ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

STUDIO E SVILUPPO PROTOTIPALE DI UN WORKBENCH VIRTUALE IN REALTÀ MISTA A SUPPORTO DEL CHIRURGO IN SALA OPERATORIA

Elaborato in
SISTEMI EMBEDDED E INTERNET OF THINGS

Relatore
Prof. ALESSANDRO RICCI

Corelatore
Dott. Ing. ANGELO CROATTI

Presentata da
FILIPPO BENVENUTI

Anno Accademico 2020 – 2021

Alla mia famiglia

Indice

Introduzione

Il recente sviluppo tecnologico relativo a realtà aumentata e realtà mista rende possibile la loro applicazione in svariati ambiti applicativi, come forma di Human Computer Interaction. Fra questi ambiti, questa tesi prende come contesto di riferimento quello clinico. In questa tesi verranno esplorate le definizioni e le tecnologie di realtà aumentata, mista e virtuale con l'intento di districare e unificare la complessa nomenclatura dell'ambito, in seguito l'attenzione verrà riposta sulle potenzialità di questi sistemi per l'*healthcare*, evidenziandone la validità e l'efficacia a fronte dell'analisi di articoli pubblicati in merito, mostrando attraverso esempi le diverse modalità con cui i visori sono stati usati, indossati direttamente dal paziente in sostituzione alle terapie tradizionali oppure dal medico curante per la visualizzazione di informazioni in tempo reale del paziente in cura.

Buona parte della raccolta sullo stato dell'arte riguarda l'uso della realtà mista attraverso Hololens 2 in sala operatoria, i risultati descritti in questi articoli e l'opinione degli esperti del dominio è ciò che ha motivato la realizzazione del progetto di questa tesi, il cui obiettivo è quello di progettare e sviluppare un sistema per la realizzazione di un **workbench virtuale** utilizzabile dal chirurgo in sala operatoria, essa dovrà contenere tutta una serie di strumenti utili in ogni tipologia d'intervento come la possibilità di leggere in tempo reale i parametri vitali del paziente o richiedere un consulto in video chiamata ad un medico esterno alla sala, che supportino senza mai essere d'intralcio l'operazione in svolgimento.

Il sistema proposto si pone come base per lo sviluppo di applicazioni in realtà mista a supporto del chirurgo, integra strumenti di generale necessità in sala operatoria ed è progettato per accoglierne di più specifici ad ausilio di particolari interventi. La tesi si articola in tre macro capitoli seguiti dalla validazione del progetto:

1. Per capire a pieno le potenzialità della realtà mista nel primo capitolo esploreremo come prima cosa le definizioni con cui familiarizzare per entrare nell'argomento, classificheremo i visori ad oggi esistenti secondo quale tecnica di combinazione usano per mescolare realtà e virtua-

lità, distinguendoli inoltre secondo la posizione nella quale essi vengono indossati o sono posti relativamente all'utente. Negli ultimi paragrafi verranno analizzate le tecniche per la realizzazione concreta della realtà mista scendendo nel dettaglio delle componenti *hardware* di Hololens 2 ed esplorandone le caratteristiche *software* necessarie all'elaborazione delle informazioni sull'ambiente e la visualizzazione di ologrammi.

2. Nel secondo capitolo è sottolineata l'importanza dei visori in realtà aumentata e virtuale nell'ambito sanitario, usati in un caso dai pazienti in sostituzione ai classici trattamenti farmacologici oppure direttamente dai medici come supporto visivo agevole durante il loro lavoro o come strumento di formazione in sostituzione ai metodi tradizionali che alle volte risultano costosi e di scarsa personalizzazione. Infine vengono presentate diverse applicazioni in realtà mista trovate in letteratura a supporto del chirurgo in sala operatoria per specifici interventi, grazie alle quali è stato possibile definire accompagnate dall'opinione degli esperti del dominio quali fossero gli strumenti utili in sala operatoria da integrare nel **workbench virtuale**.

3. Il terzo capitolo è volto all'analisi, alla progettazione e in seguito allo sviluppo del sistema proposto, specifica inizialmente quali sono gli obiettivi del progetto motivandone l'importanza, dopo una parte di ingegneria in cui il sistema è definito attraverso l'uso di diagrammi UML sono descritte le tecnologie utilizzate per svilupparlo e le tecniche, infine alcune assunzioni che si presuppone l'ospedale rispetti per un corretto funzionamento del sistema progettato.

4. Il quarto ed ultimo capitolo è quello di validazione, sono raccolte le opinioni dello sviluppatore e degli esperti del dominio sul sistema proposto, per definirne la correttezza e delineare i possibili sviluppi futuri da implementare.

Capitolo 1

Mixed Reality

Mentre pochi anni fa il pensiero di virtualizzare l'ambiente in cui viviamo era il futuro, ad oggi grazie alla ricerca e all'evoluzione tecnologica è divenuto il presente, non è passato molto tempo dall'uscita nelle sale cinematografiche del film "Iron Man" che ci ha fatto sognare un mondo aumentato ricco di ologrammi, eppure già troviamo nel mercato visori che ci permettono di vivere esperienze di realtà mista o totalmente virtuale.

1.1 Reality-Virtuality Continuum

Per introdurre e spiegare la terminologia possiamo rifarci al concetto di *reality-virtuality continuum*[15] introdotto da Milgram e Fumio Kishino nel 1994 secondo il quale la virtualizzazione è un'estensione del mondo in cui viviamo come rappresentato in figura 1.1, quest'ultimo assieme alla totale virtualizzazione(VR) rappresentano i due estremi della realtà, tutto ciò che si pone tra questi è parte della realtà mista(MR), più vicino al mondo reale parliamo di realtà aumentata(AR) opposta alla virtualità aumentata (AV) invece più vicino alla realtà virtuale.

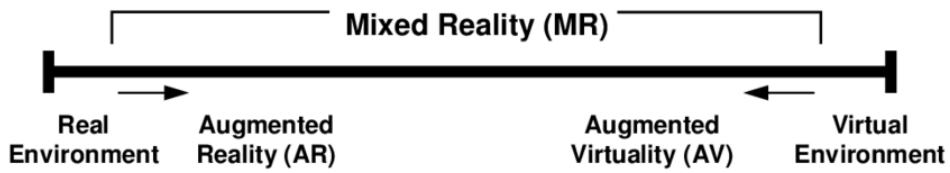

Figura 1.1: Reality-virtuality (RV) continuum

1.2 Definizioni

// breve intro.

1.2.1 Realtà aumentata

Ronald T. Azuma concorda con la definizione di Milgram pone la realtà aumentata da qualche parte al centro tra la realtà e l'ambiente virtuale, introduce un nuovo termine *virtual environment* (VE) e spiega essere solo un altro modo per riferirsi alla *virtual reality* (VR) un ambiente totalmente sintetico, il contributo alla definizione di realtà aumentata risiede nell'averla sganciata dai limiti delle tecnologie utilizzate per realizzarla astraendola ad un qualsiasi sistema dotato di queste tre caratteristiche [2]:

1. **Combina realtà e virtuale**: in grado di disporre elementi virtuali nello stesso contesto della realtà.

2. **Interattivo in tempo reale**: in grado di interagire in tempo reale con l'utente ed apportare modifiche agli elementi virtuali in risposta ad esso.

3. **Sviluppato in 3 dimensioni**: in grado di porre elementi virtuali nella realtà mostrandoli immersi in essa, ad esempio un oggetto virtuale posto dietro uno fisico verrà occluso da quello fisico rendendolo non visibile o visibile in parte, allo stesso modo vale il contrario per un oggetto fisico dietro ad un oggetto virtuale come mostrato in figura 1.2.

Figura 1.2: Desktop fisico con una lampada virtuale e due sedie virtuali

Leggendo queste caratteristiche verrà naturale pensare facciano riferimento alla sola vista, è invece importante notare che tale astrazione non è strettamente legata ad alcuna tecnologia e nemmeno ad un solo senso, prendendo ad esempio l'udito e rileggendolo nelle caratteristiche notiamo che secondo la definizione è importante essere in grado di sentire sia i suoni reali che i virtuali e che il suono virtuale venga percepito dalla persona in base a come si muove e alle occlusioni ambientali sia fisiche che virtuali (un suono generato dietro una parete risulterà meno forte di uno diretto alla persona senza occlusioni).

1.2.2 Realtà mista

Nonostante la definizione sul *continuum* 1.1 e la precisazione di Azuma 1.2.1 il concetto di realtà mista rimane comunque confuso, lo dimostra la pubblicazione scientifica "What is Mixed Reality"[14] scritta a valle di un'interrogazione condotta su dieci esperti del settore (ricercatori universitari e industriali) e l'analisi letteraria di 68 articoli, dalla quale dopo un'accurata elaborazione dei dati raccolti sono state stabilite sei definizioni più comunemente accettate e note:

1. **Continuum**: questa è la definizione "tradizionale" in accordo con Reality Virtuality continuum [15] di Milgram secondo la quale la realtà mista è tutto ciò che si pone tra i due estremi quali realtà e realtà virtuale quindi includendo la realtà aumentata ma escludendone gli estremi.

2. **Sinonimo**: semplicemente il concetto di realtà mista viene visto come sinonimo di realtà aumentata, usando quindi i due termini in modo equivalente e intercambiale.

3. **Collaborazione**: in questo caso descrive l'interazione tra realtà aumentata e realtà virtuale imponendo quindi la presenza di almeno due utenti, la comunicazione in due stanze separate di una persona in AR e una in VR virtualizzando la realtà del primo nel visore del secondo è un esempio di collaborazione in realtà mista.

4. **Combinazione**: un sistema in grado di combinare parti distinte di realtà aumentata e realtà virtuale facendole collaborare tra di loro oppure un sistema in grado di passare da una all'altra all'occorrenza (es. Pokemon GO).

5. **Allineamento**: una sincronizzazione tra un mondo fisico ed uno virtuale o l'allineamento di una rappresentazione virtuale sopra un ambiente fisico, una forte immersione in un mondo aumentato o virtuale che rispecchia la realtà in cui ci si trova.

6. **AR potenziata**: caratterizzata da un'avanzata comprensione dell'ambiente circostante, una migliore interazione con oggetti virtuali e una corretta integrazione del virtuale nella realtà, questo può significare limitare la realtà mista a una determinata tecnologia in grado di fornire queste funzionalità che la "classica" realtà aumentata non supporta e che quindi la realtà mista sia un'evoluzione di quella aumentata.

Come sottolineato da questo studio non esiste ad oggi una definizione univoca di realtà mista, questo perchè ne assume una diversa per ogni ambito in cui viene usata e vorrebbe descrivere un concetto ancora in via di sviluppo quindi mutabile. Per quanto riguarda questa tesi la definizione a cui farà riferimento è quella di Milgram e il continuum 1.1, vedremo poi alla sezione 1.6 perchè il visore Hololens 2 viene descritto dalla stessa casa madre (Microsoft) un visore a realtà mista.

1.3 Combinazione elementi reali e virtuali

Combinare in modo efficace elementi virtuali nella realtà è un processo complicato [4] sono infatti necessari diversi passaggi prima di ottenere l'effetto voluto, per iniziare è necessaria la **camera calibration** è un'operazione per far combaciare i parametri della camera virtuale con quelli della camera fisica o di un modello ricostruito sulla visione dell'utente, in modo che le immagini virtuali renderizzate siano allineate con la vista sul mondo reale, esistono due tipi di parametri di una camera:

- **Interni**: determinano come una scena in 3D venga proiettata in un'immagine bidimensionale, possono essere calcolati usando un insieme di immagini di calibrazione contenenti *pattern* conosciuti a priori osservando come vengono proiettati nell'immagine in 2D, tutto ciò viene solitamente effettuato a priori dell'uso ma esistono anche sistemi che si calibrano *on the fly* [10], nel caso non fosse presente una camera fisica tali parametri vengono decisi sfruttando un modello geometrico che mette in relazione gli occhi dell'utente e lo schermo.

- **Esterni**: determinano la posizione e l'orientazione della camera, quando la scena fisica è statica solo la posizione della camera rispetto all'ambiente deve essere tracciata, se invece la scena fosse dinamica allora anche tutti gli oggetti d'interesse dovrebbero essere tracciati di modo che i cambiamenti nel mondo fisico siano riflessi nella scena virtuale.

Le tecnologie di **tracking** provvedono a recupare la posizione e l'orientamento di un determinato oggetto relativamente a un sistema di coordinate

scelto da lui stesso, per allineare correttamente il mondo virtuale al mondo fisico è necessario convertire queste coordinate nelle coordinate del mondo fisico con un processo chiamato **registration**, alla fine di tutto è finalmente possibile sovrapporre l'immagine virtuale al mondo fisico attraverso un processo chiamato **composition**.
Le tecnologie al giorno d'oggi in grado di mettere in atto tutti questi passaggi, tenendo in considerazione come combinano il mondo fisico con gli elementi virtuali, possono essere categorizzate in quattro gruppi: *video-based*1.3.1, *optical see-through*1.3.2, *projection-based*1.3.3, *eye multiplexed*1.3.4.

1.3.1 Video-based

Questi visori per prima cosa digitalizzano il mondo reale usando una videocamera per poi comporre le immagini virtuali con il video e mostrare il risultato attraverso lo schermo su cui la persona sta guardando 1.3.

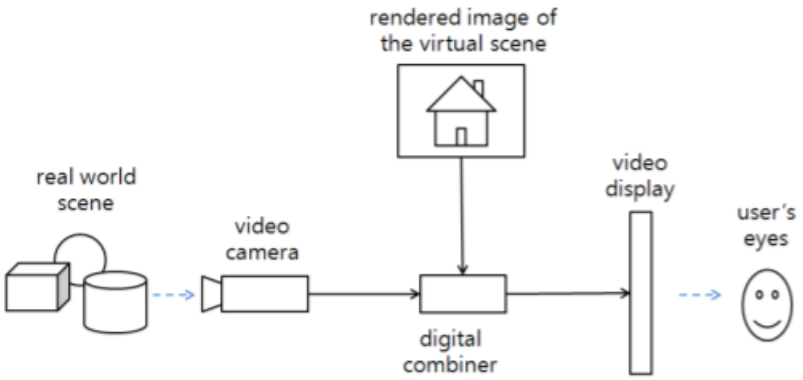

Figura 1.3: Struttura di un visore video-based

Solitamente la videocamera viene posta nel retro dello schermo, in questo modo l'utente avrà l'impressione di vedere attraverso lo schermo e gli verrà naturale muoverlo per adattarlo a ciò che vuole vedere, questa tecnica prende proprio il nome dall'effetto che sfrutta: **video see-through**.

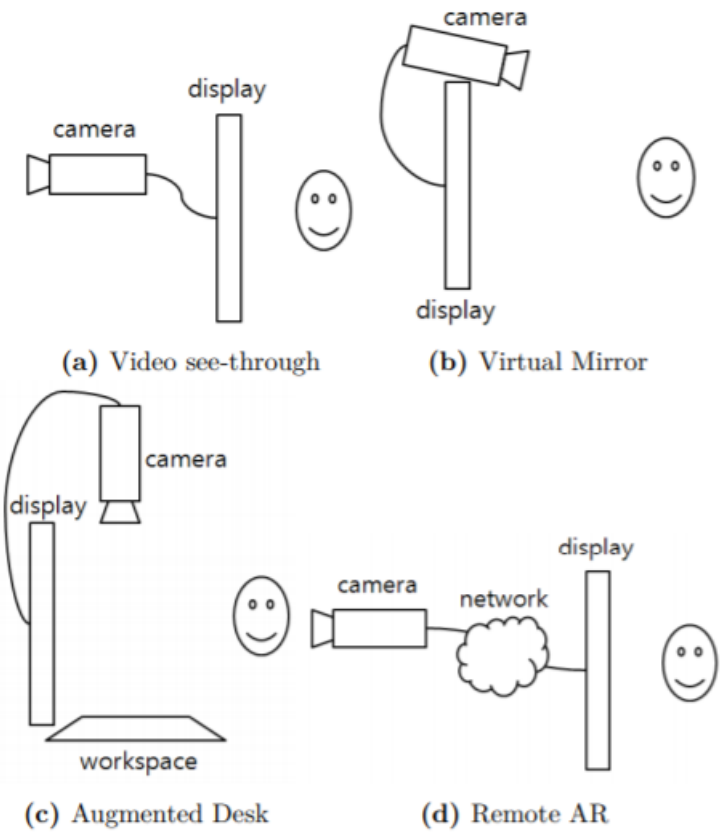

Figura 1.4: Configurazioni camera in un visore video-based

Altri modalità in cui impostare la videocamera sono rappresentati in figura 1.4, **virtual mirror** la posiziona verso l'utente per creare una sorta di specchio virtuale, **augmented desk** invece la punta verso l'ambiente di lavoro per aumentarne il contenuto evidenziando ad esempio il pezzo da montare o il cacciavite da usare, **remote AR** invece sfrutta una connessione per ricevere il video da una fonte remota, in questo modo è possibile immergersi in un ambiente lontano senza doverlo fisicamente raggiungere.
Il punto forte di queste tecniche risiede nell'alta precisione con cui è possibile combinare elementi virtuali e mondo fisico grazie all'uso di accurati algoritmi di visione artificiale, sommato al fatto che hanno un costo di realizzazione molto basso rendono questi visori i più diffusi nel mercato e i più facili da costruirsi in casa, infatti gli unici componenti di cui si ha bisogno sono una camera e uno schermo, tecnologia presente ovunque al giorno d'oggi.

1.3.2 Optical see-through

Questi display sfruttano tecniche ottiche per combinare gli elementi virtuali con il mondo fisico, solitamente si fa uso di *beam splitter* (semi specchi o prismi combinati) 1.5 che uniscono la realtà vista direttamente attraverso lo splitter e il riflesso di un'immagine presa da uno schermo, i visori *head up displays* (HUD) sono solitamente presenti nell'abitacolo di un aeroplano o in macchine moderne, anche se in questo caso è il mondo fisico ad essere riflesso e il display visto attraverso lo specchio.

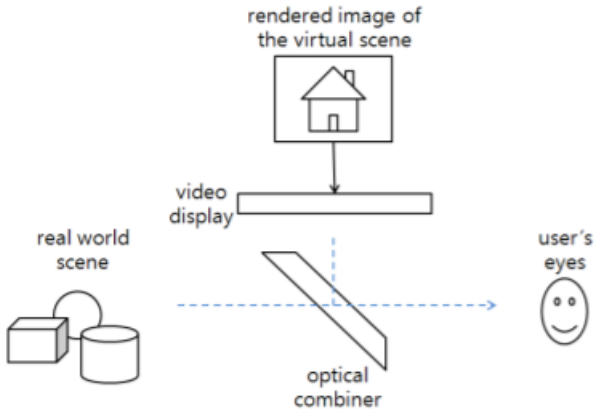

Figura 1.5: Schema riassuntivo funzionamento di un display optical see through

Un'altra tecnica ottica per la sovrapposizione di immagini sono i *transparent projection films* 1.6 noti non solo perchè in grado di diffondere la luce per mostrare l'immagine proiettata ma anche in quanto transparenti per permmettere la visione della realtà retrostante, l'uso di tal tecnica ha contribuito a semplificare e miniaturizzare la struttura e la dimensione dei visori. A confronto con i visori *video-based* 1.3.1 il vantaggio degli *optical see through* risiede nella libera e diretta visione della realtà, questa può essere una caratteristica molto importante in tutti quegli ambiti in cui la vista sugli oggetti fisici è fondamentale come ad esempio in applicazioni mediche o militari.

Figura 1.6: Esempi di display di tipo projection film

Al contrario dei *video-based* 1.3.1 invece gli *optical see through* sono meno accurati quando si parla di *registration* 1.3 e necessitano di accurati sistemi di tracciamento tridimensionale degli occhi per calcolare i parametri di calibrazione in modo corretto da evitare il disallineamento delle immagini virtuali nel caso in cui il visore cambi di posizione relativamente agli occhi, un'altra peculiarità di questi visori è data dalla loro natura in quanto transparenti per la quale sarà sempre possibile notare dietro la rappresentazione di un'immagine virtuale l'ambiente fisico retrostante, tale problematica è stata praticamente risolta da Kiyokawa et al. [12] svilippando una maschera elttronicamente controllata che occlude la visione sul mondo fisico nella zona in cui si trova l'oggetto virtuale. Le condizioni di luce nel mondo reale possono ulteriormente influire sulla qualità degli elementi virtuali, questo può essere risolto attraverso l'uso di diversi display più o meno scuri oppure sfruttando speciali schermi LCD in scala di grigi per controllare elettronicamente quanta luce esterna far arrivare agli occhi.

1.3.3 Projection-based

Mentre altri tipi di visori combinano la realtà e gli elementi virtuali sul piano del display i *projection-based* non fanno altro che proiettare le immagini

virtuali direttamente sulla superficie degli oggetti fisici 1.7, combinando questo con il tracciamento dell'utente e degli oggetti stessi è possibile creare un ambiente aumentato e interattivo. Spesso questi sistemi vengono realizzati con l'ausilio di proiettori e videocamere fissate a muro o sul soffitto lasciando libere le persone dall'indossare visori in testa o portarli in mano ma limitando in questo modo l'ambiente aumentato ai confini del proiettore, in altri casi invece si fa uso di proiettori indossabili [13] per rendere la scena dinamica e permettere una virtualizzazione mobile a seconda degli spostamenti della persona che lo indossa.

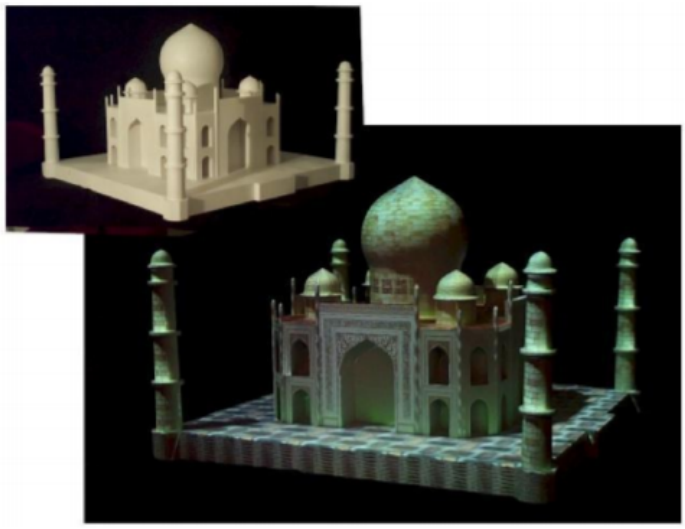

Figura 1.7: Modello di TajMahal aumentato attraverso l'uso di una proiezione [17]

Per ottenere i risultati desiderati è necessario che le superfici su cui si proietta siano fisicamente adatte a tale scopo e che la distanza tra il proiettore e l'oggetto fisico non sia troppo elevata, le problematiche maggiori dell'uso di questa tecnica risiedono nell'illuminazione dell'ambiente circostante e nell'occlusione data dalla presenza di oggetti tra il proiettore e la superficie (ad esempio le mani o la persona stessa), nel caso di un ambiente troppo luminoso otteniamo proiezioni poco visibili e desaturate mentre occludendo il proiettore si verranno a generare zone d'ombra che possono risultare fastidiose o comunque andare a rovinare l'esperienza di realtà aumentata.

1.3.4 Eye multiplexed

Al contrario dei tre metodi appena descritti questo non propone all'utente il risultato finale del mondo aumentato ma gli demanda l'onere di comporre

gli elementi virtuali nel mondo fisico, per fare ciò si sfrutta l'abilità del nostro cervello nel comporre le due diverse immagini provenienti dai nostri occhi attraverso un sistema composto come descritto nell'immagine 1.8.

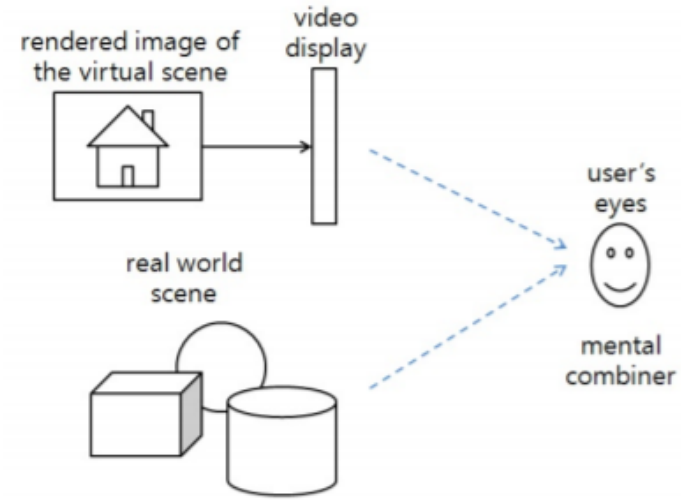

Figura 1.8: Struttura di un modello eye multiplexed

Un occhio dell'utente è lasciato libero di visualizzare l'ambiente circostante mentre uno è totalmente o quasi totalmente occluso dallo schermo rappresentante il mondo virtuale, nonostante questo sia registrato 1.3 nel mondo fisico l'immagine risultante non è combinata con esso lasciando alla mente della persona il lavoro di sovrapporli richiedendo quindi una minore potenza computazionale rispetto alle altre metodologie. La precisione al pixel nella rappresentazione degli oggetti virtuali non è necessaria, persino la registrazione nell'ambiente fisico può essere imperfetta, questo è possibile grazie alla forte correlazione che il nostro cervello impone alle immagini ricevute dagli occhi e che entro certi limiti è in grado di mostrarci come fossero appartenenti alla stessa scena.

1.4 Classificazione tipi di visori

Abbiamo visto nella sezione 1.3 le diverse tecniche di combinazione per aumentare la realtà fisica, classifichiamo ora questi visori in base a come vengono posti in relazione all'utente secondo la scala di Bimber e Raskar [5] suddividendoli in tre categorie considerando la distanza tra gli occhi e il visore stesso come rappresentato in figura 1.9.

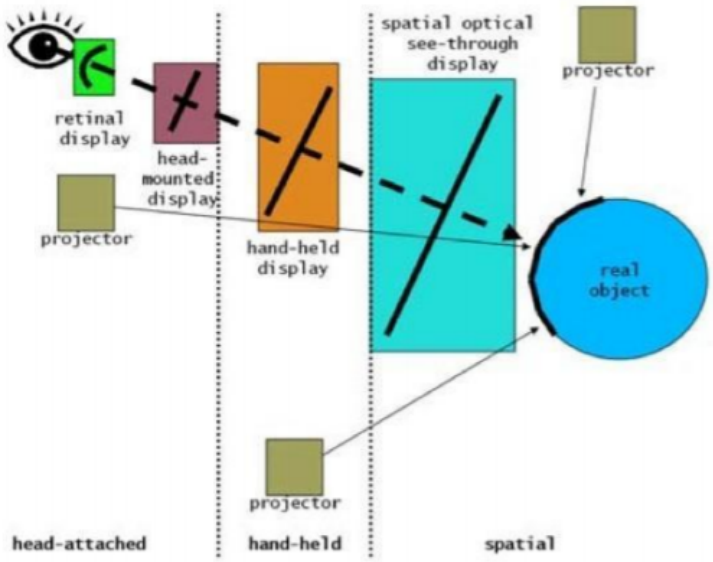

Figura 1.9: Tipi di visori secondo Bimber e Raskar [5]

1.4.1 Head-attached displays

Questi visori visualizzano le immagini virtuali proprio davanti agli occhi dell'utente impedendo che alcun altro oggetto si interporrà tra vista e schermo garantendo in questo modo l'assenza di occlusioni viste nel capitolo 1.3.3, possono essere di diverse misure a partire da caschetti fino ad arrivare al peso di semplici occhiali come siamo abituati a vederli tutti i giorni o addirittura grazie a recenti sviluppi persino a indossarli direttamente sugli occhi come lenti a contatto.

Figura 1.10: Diversi tipi di head-attached displays partendo da Mojo lenses (www.mojo.vision/mojo-lens), Trivisio (www.trivisio.com) e infine Google Glass (www.google.com/glass)

Ci riferiamo a questi visori come **Head mounted displays** (HMD), in letteratura troviamo riferimenti a questi anche con il nome di *Near eye displays* oppure *Face mounted displays* che fanno proprio riferimento al fatto di avere uno schermo appoggiato alla faccia o comunque molto vicino agli occhi. Questa tipologia di visori è preferibile nella realizzazione di applicazioni immersive cioè dove vogliamo integrare il mondo virtuale a quello fisico facendolo risultare il più naturale possibile, questo perché spesso sono in grado di offrire un *Field of view* (FOV) abbastanza ampio da coprire la maggior parte della vista della persona immergendolo in un mondo più o meno aumentato a seconda delle esigenze.

1.4.2 Handheld and body-attached displays

Come si intuisce dal titolo questi sono visori da tenere in mano offrendo quindi una grande mobilità e libertà di spostarlo praticamente in qualsiasi punto, ma impongono di base l'avere almeno una mano occupata dal visore stesso rendendoli poco utilizzabili mentre si lavora bensì preferiti quando l'unico risultato che si vuole ottenere è la vista aumentata sul mondo reale senza una vera necessità di interagirci in tempo reale.

Figura 1.11: Un esempio di display hand held con sensore di profondità Google Tango

Come vediamo in figura 1.11 una funzionalità molto ambita in diversi campi e realizzabile grazie a sensori di profondità è la ricostruzione in 3D di ambienti in tempo reale realizzata ad esempio da *Google Project Tango*[1] sfruttando avanzate tecniche di tracciamento e visualizzazione. A questo punto è importante notare che per realizzare questi visori è necessario e sufficiente un qualsiasi dispositivo dotato di schermo, videocamera e potenza computazionale, considerando che queste caratteristiche al giorno d'oggi le ritroviamo in un qualsiasi dispositivo già presente in mano alle persone rende applicazioni basate su questo concetto quelle con più potenzialità d'utilizzo da parte di tutti.

[1] `www.support.google.com/faqs/faq/6029402?hl=en`

Alcune compagnie elettroniche hanno prodotto schermi LCD e OLED trasparenti che potrebbero avere applicazioni molto interessanti nell'ambito dei visori *hand held* perchè non costringono l'utente a vedere il mondo attraverso una videocamera ma lasciano libera visione dell'ambiente retrostante visualizzando sullo schermo solo gli elementi virtuali, diminuendo in questo modo il divario tra realtà e virtualità.

1.4.3 Spatial displays

Questa è la categoria di visori posizionati più distanti dagli occhi della persona al contrario delle altre due tipologie in questo caso parliamo di visori senza mobilità in quanto solitamente fissati in una determinata posizione, tendono ad offrire un'ampia visione virtualizzata rendendo questo l'ottima tipologia per applicazioni di condivisione dove due o più persone osservano la scena attraverso lo stesso schermo. Il tipico esempio d'utilizzo è rappresentato nella figura 1.12 utilizzando un *beam splitter* 1.3.2 si va a creare un workbench virtuale sulla quale possiamo liberamente lavorare con le mani e allo stesso tempo vedere immerse in essa immagini virtuali, in questo caso particolare vediamo in atto anche l'uso di un'interfaccia *haptic* [3] con la quale è possibile muovere un oggetto virtuale molto precisamente senza l'uso eccessivo di potenza computazionale, gli occhiali che indossa la persona nella figura 1.12 vengono chiamati *stereo shutter* attraverso specifiche tecniche ottiche decidono cosa mostrare o meno agli occhi per rendere possibile un effetto di visione tridimensionale, sono gli stessi che vengono forniti all'ingresso di una sala cinematografica quando si vuole vedere un film proiettato in 3D.

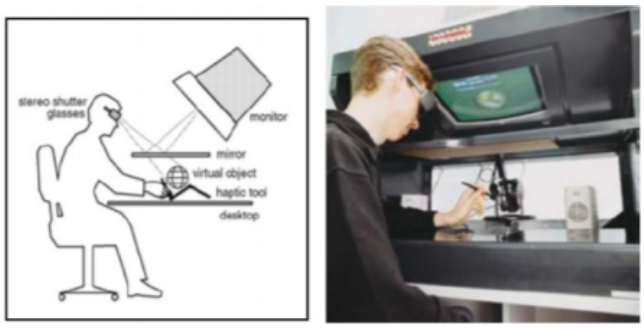

Figura 1.12: Un esempio di scrivania aumentata e *haptic* [7]

Anche in questo caso come per gli *handheld displays* 1.4.2 schermi LCD o OLED trasparenti possono raffigurare un notevole miglioramento sia per la visualizzazione che implementazione di questi sistemi, in generale gli *spatial displays* possono essere costruiti secondo diverse metodologie di combinazione

come abbiamo visto alla sezione 1.3, proiettando le immagini virtuali direttamente sulla superficie degli oggetti fisici 1.3.3 oppure facendo uso di tecniche video-based mettendo a disposizione uno schermo collegato ad una videocamera che mostra la scena aumentata direttamente agli utenti o ribaltare la camera verso le persone per creare uno specchio virtuale aumentato 1.3.1.

1.5 Uno sguardo software alla MR

Nella sezione 1.1 abbiamo definito il *continuum* secondo Milgram come un insieme continuo di ambienti partendo dalla realtà fisica fino a quella virtuale passando per la realtà aumentata e quella mista, ciò che li contraddistingue a livello di design è la percezione che ne ha la persona, nel caso della AR l'utente rimane sempre nel mondo fisico e non gli viene mai fatto credere di starlo abbandonando al contrario della VR nella quale l'utente è completamente immerso in un mondo sintetico ed è totalmente ignaro dell'ambiente fisico, nel caso della MR i due estremi sono perfettamente fusi e l'utente percepisce la realtà e la virtualità come un unico mondo, a rendere possibile quest'ultimo scenario è stato l'avanzamento degli studi nella *human computer interaction* (HCI) grazie ai quali sono stati sviluppati algoritmi in grado di elaborare informazioni sull'ambiente derivanti da fotocamere e altri sensori per realizzare sistemi di *head tracking*[2]1.5.1, *spatial mapping*[3]1.5.2 e *scene understanding*[4]1.5.3, l'unione dell'input dell'ambiente, di quello umano e la loro elaborazione è ciò che sta alla base della realtà mista.

1.5.1 Sistema di coordinate

Al loro interno le applicazioni in realtà mista usano le **coordinate cartesiane** per posizionare e tracciare gli ologrammi 1.5.4,le nostre mani, i nostri occhi e l'ambiente che ci circonda, essendo per natura legate alla realtà sono rappresentate da tre assi X, Y e Z e ci riferiamo ad esse con il nome ***spatial coordinates***, in tutte le applicazioni di realtà mista di Windows queste sono *right-handed* cioè l'asse X punta a destra, l'asse Y in alto e l'asse Z avanti di fronte all'utente, inoltre Le coordinate spaziali esprimono i loro valori in metri, in questo modo posizionando due elementi distanti 2 unità nella realtà risulteranno distare 2 metri. La base della renderizzazione degli ologrammi è aggiornarne la vista al variare della posizione e orientamento dell'utente, in

[2] `shorturl.at/ezGMO`
[3] `shorturl.at/kmANX`
[4] `shorturl.at/cxLYZ`

base a quale tipo di esperienza si vuole creare esistono diverse metodologie di fissaggio del sistema di coordinate al mondo o alla persona stessa:

- **Stationary frame of reference**: un sistema adatto per tutte quelle applicazioni in cui si vogliono posizionare ologrammi nel mondo entro 5 metri dall'origine inizialmente fissata all'avvio dell'applicazione corrispondente alla testa dell'utente, utile per tutte quelle esperienze *seated-scale* in cui si presume che l'utente sia seduto o comunque non si sposti, se ciò dovesse succedere potrebbero verificarsi disallineamenti degli ologrammi rispetto alla realtà causando imprecisione nella loro rappresentazione, se invece l'utente dovesse spostarsi volontariamente per cambiare postazione e volesse far riapparire gli ologrammi attorno a lui come prima, è possibile aggiornare l'origine del mondo sulla nuova posizione della testa e ottenere l'effetto desiderato. Un ologramma viene detto *world-locked* quando la sua posizione o orientamento non variano allo spostarsi dell'utente, ma dipendono solamente dall'origine del mondo.

- **Attached frame of reference**: un sistema adatto alle applicazioni in cui non è necessario fissare gli ologrammi al mondo bensì averli sempre a "portata di mano" ovunque l'utente si sposta (anche oltre i 5 metri dall'origine), le coordinate hanno la loro origine nel corpo della persona che le usa, essa si sposta nel mondo seguendo l'utente nei suoi movimenti mantenendo così fisse le distanze tra gli oggetti e l'utente stesso, per questa caratteristica questa tipologia di coordinate non ha bisogno di riconoscere il mondo circostante, rendendo di fatto questo sistema l'unico funzionante nel caso in cui il riconoscimento del mondo non dovesse avere successo e quindi un'ottima strategia per visualizzare errori relativi allo *spatial mapping* 1.5.2 e alla *scene understanding* 1.5.3. Un ologramma viene detto *body-locked* quando la sua posizione o orientamento dipendono dall'origine posta nel corpo dell'utente.

- **Stage frame of reference**: un sistema adatto alle applicazioni immersive, l'utente prima di iniziare ad usare l'applicazione deve definire attorno a se un'area nella quale può muoversi liberamente (sempre rispettando un massimo raggio di 5 metri) e l'altezza del pavimento, gli ologrammi avranno la loro origine ai piedi dell'utente di conseguenza Y=0 indica un ologramma appoggiato a terra, utile in tutte quelle applicazioni *standing-scale* in cui l'utente può camminare e spostarsi ma sempre rimanendo all'interno dell'area definita.

In un mondo completamente virtuale registrare la posizione degli ologrammi è molto semplice, una volta istanziati se non spostati infatti manterranno

sempre le stesse identiche coordinate, questo perché il mondo in cui si trovano non è soggetto a cambiamenti indesiderati, in un'applicazione in realtà mista invece le coordinate dipendono da una ricostruzione dinamica dell'ambiente che ci circonda, questo vuol dire che due oggetti a distanza di 4 metri dopo un'eventuale rivalutazione dell'area circostante potrebbero trovarsi a 3.9 metri, qui si nasconde il limite dei 5 metri di raggio al di fuori dei quali Microsoft ci dice che gli ologrammi perdano di precisione, l'aggiornamento dell'ambiente ad una tale distanza non è preciso e ciò causa inevitabilmente discrepanze tra realtà e mondo aumentato, fortunatamente però per sorpassare questo limite un altro sistema di fissaggio è stato definito: le **ancore spaziali**. Le *spatial anchor* sono il sistema con cui si risolve il problema appena descritto, esse permettono di definire punti importanti del mondo di cui l'applicazione dovrebbe curarsi di tener traccia, esse aggiustano la loro posizione al variare dell'ambiente circostante mantenendo però costanti le distanze tra un'ancora e l'alta costruendo uno scheletro solido virtuale a cui gli ologrammi possono fissarsi, questo rende possibile la creazione di applicazioni in esperienza *world-scale*, ma bisogna prestare attenzione al fatto che comunque sia un ologramma deve rimanere vicino alla sua origine, come indicato dalle linee guida non dovrebbe mai allontanarsi oltre i 3 metri dall'ancora a cui è fissato, ovviamente si possono definire ulteriori ancore, ma questo significa che per spostare un oggetto nell'ambiente è necessario assegnargli l'ancora a cui è fissato in modo dinamico. Un'altra caratteristica di questo sistema di coordinate è la presenza dello *spatial anchor store*, esso permette di salvare le ancore permanentemente su disco per poterle ricaricare tra una sessione e l'altra semplicemente conoscendo la loro stringa identificativa, inoltre la persistenza di un'ancora si porta dietro anche gli ologrammi ad essa associati, in questo modo è possibile fissare oggetti nel mondo che tra una sessione e l'altra permarranno come ci si aspetta. Il salvataggio e persistenza delle ancore apre il mondo anche allo *spatial anchor sharing*, sfruttando i servizi di *Azure spatial anchors*[5] è possibile condividere anche in tempo reale le ancore di un ambiente, permettendo quindi l'interazione di più persone sugli stessi ologrammi e la possibilità di entrare nella stessa stanza virtuale anche quando il dispositivo che l'ha creata non è presente o acceso.

1.5.2 Spatial mapping

Lo ***spatial mapping*** è il sistema con cui l'applicazione prende conoscenza delle superfici componenti l'ambiente circostante, con queste informazioni è possibile adattare gli ologrammi al mondo fisico per ricreare situazioni realistiche che facciano percepire all'utente gli ologrammi come appartenenti al

[5] `shorturl.at/htyDY`

posto in cui si trova, in generale possiamo analizzare i vantaggi della *spatial awareness* sotto diversi punti di vista:

- **Placement**: caratteristica propria di un ologramma è che non ha vere proprietà fisiche ed è compito dello sviluppatore definirle, questo rende possibile creare ologrammi che volano, ologrammi che non subiscono l'effetto della gravità o ologrammi che passano attraverso i muri, ma quando vogliamo realizzare oggetti che rappresentino la realtà è importante che le basiche leggi della fisica vengano applicate anche a loro, una volta definito una forma e un peso è importante definire anche le superfici su cui si può appoggiare, in questo modo un ologramma lasciato sul tavolo non cadrà sotto ad esso ma ci si appoggerà sopra. Durante la realizzazione degli ologrammi è importante definire la forma che hanno, ma non è necessario essere estremamente precisi difatti in casi come per una sedia in cui si hanno zone appuntite è preferibile addolcirne la superficie per evitare interazioni erronee con l'ambiente circostante, in questo modo durante il posizionamento su una superficie da parte dell'utente si evitano errori di compenetrazione e strani effetti visivi come scatti, per semplificarne ulteriormente il posizionamento è utile all'utente illuminare la superficie alla quale l'oggetto che ha in mano si sta avvicinando, in questo modo l'utente ha un'immediata consapevolezza di una corretta interazione e può rilasciare la presa tranquillo che l'oggetto rimanga appoggiato o non cada per una sbagliata valutazione della distanza tra ologramma e superficie. Un altro metodo discusso per il posizionamento degli ologrammi è quello automatico, all'avvicinarsi dell'oggetto alla superficie questo viene automaticamente posizionato su di essa evitando compenetrazioni o distaccamenti involontari, è poi possibile aggiustarne la posizione sulla superficie semplicemente spostandolo, questo metodo risulta molto comodo in molti casi ma è necessario tenere a mente che l'automatismo potrebbe non essere quello che si aspetta l'utente, ad esempio in questo caso si sta togliendo la possibilità di rilasciare un oggetto vicino ad una superficie senza che la tocchi, uno scenario decisamente raro ma che comunque impone un comportamento non naturale degli ologrammi.

- **Occlusion**: uno dei principali utilizzi delle superfici è quello di occludere la vista in parte o totalmente degli ologrammi posti dietro ad esse, vedere gli ologrammi attraverso i muri non è un comportamento sempre desiderato, soprattutto quando vogliamo creare ambienti in realtà mista che non si allontanino dalla realtà è importante tenere conto dei muri e della loro solidità, se un ologramma non è visibile dietro ad una superficie ci si aspetta inoltre che non ci possa passare attraverso. La corretta occlusione degli ologrammi fornisce inoltre all'utente informazioni più precise

sulla loro posizione, vedere un oggetto spuntare in parte da dietro un altro ci fa capire immediatamente dove si trovi e come raggiungerlo, un oggetto invece rappresentato dietro un muro potrebbe andare a rovinare la percezione che abbiamo della realtà mettendoci anche in situazioni più o meno pericolose, un utente distratto potrebbe tentare di afferrare l'ologramma sbattendo la mano contro il muro, in generale però questo non è un comportamento da scartare infatti esistono applicazioni in cui è necessaria la "vista a raggi X", casi in cui l'interazione con l'ologramma non deve interrompersi nemmeno se occluso da un muro, per rendere questo comportamento sicuro e più naturale alla vista è preferibile inscurire l'ologramma occluso per informare l'utente che esso si trova dietro ad una superficie solida e che per interagirci non può afferrarlo direttamente.

- **Physics**: simulare la fisica degli ologrammi è uno dei migliori modi per integrare gli oggetti virtuali con la realtà, vedere una gomma per cancellare cadere dalla scrivania, rimbalzare per terra e infilarsi sotto ad un armadio ci da l'impressione che sia reale, seguirla nel suo tragitto per recuperarla ci porta ad abbassarci a terra e a cercarla esattamente come se fosse una vera gomma. Spostare un divano, far rotolare una penna o giocare con una pallina sono tutte gesta possibili e percepibili naturali solo grazie ad un motore fisico che ne elabora l'interazione con le superfici, a tal proposito è necessario assicurarsi che queste non presentino buchi per evitare che gli oggetti cadano nel nulla, inoltre è necessario scegliere se queste devono essere aggiornate dinamicamente al variare degli oggetti fisici nella stanza, quindi rispecchiando gli spostamenti di oggetti come sedie e tavoli in tempo reale oppure se data una configurazione iniziale delle superfici mantenere quella accettando i disallineamenti con la realtà, entrambe le soluzioni sono valide e la scelta è legata alla tipologia di applicazione che si sta sviluppando, in tutti i casi è comunque necessario definire il comportamento degli oggetti arrivati al bordo delle superfici digitalizzate, se lasciare che cadano nel nulla, se bloccarli al limite o se farli sparire, tutte scelte da fare sulla base dello scopo dell'applicativo.

- **Navigation**: un altro uso dello *spatial mapping* è la creazione di percorsi sui quali gli ologrammi possono muoversi, su cui *avatar* di vario tipo possono guidarsi per compiere tragitti lungo le diverse superfici in qualunque modo si voglia (camminando, arrampicandosi, di lato..), una volta create le mappe di navigazione è poi possibile utilizzarle anche per gli utenti, mostrando segnaletica che indica come muoversi, dove passare o come spostarsi è possibile guidare le persone lungo percorsi sicuri nel caso di ambienti ostili come può essere in cantiere in costruzione. La maggiore difficoltà derivante da questa tipologia d'esperienza è l'elabora-

zione in tempo reale delle superfici, una persona non può camminare sui tavoli è quindi necessario distinguere su quali superfici si può passare e su quali è meglio evitare, inoltre l'ambiente circostante potrebbe variare è quindi essenziale che l'applicazione sia in grado di far fronte a questi cambiamenti, aggiornando il percorso o consigliando l'apertura di una porta nel caso dovesse essere stata chiusa da qualcun altro, il carico di lavoro non è poco è quindi fondamentale creare algoritmi di navigazione che facciano uso di *mesh* particolarmente leggere e che non abbiano complessità computazionali elevate da rallentare l'elaborazione delle superfici in percorso, a tal proposito esistono diverse discussioni aperte e una vasta collezione di articoli in letteratura che trattano di queste problematiche e di alcune soluzioni applicabili.

- **Visualization**: la maggior parte delle volte vogliamo che l'applicativo nasconda completamente la rappresentazione virtuale delle superfici, questo perché occlude la vista sulla realtà e spesso si preferisce lasciarne libera visione di modo che sia il mondo a parlare per se stesso, ma ci sono occasioni come quella che abbiamo visto per il *placement* in cui è preferibile visualizzare una porzione di superficie per aiutare l'utente nelle sue intenzioni. In generale tendiamo a nascondere le superfici in quanto la realtà è più brava a comunicarcele, ma alle volte può essere utile ad esempio per rappresentare i muri di una stanza o la mobilia in essa contenuta dietro un muro, quindi per darci un'idea della composizione di un ambiente di cui non abbiamo diretta visione, più in generale visualizzare le superfici aiuta l'utente a prendere consapevolezza di come l'applicativo sia stato in grado di percepire la realtà e nel caso tentare di migliorarne i risultati spostandosi nell'ambiente. Quando la stanza è buia o comunque c'è scarsa visibilità data da un qualsiasi fattore può essere utile visualizzare le superfici delle stanze salvate da una vecchia scansione per permettere comunque all'utente di muoversi agevolmente nell'ambiente, bisogna ovviamente tener conto che la situazione potrebbe essere differente dall'ultima registrazione e che quindi usare questo metodo è solo un aiuto e non uno strumento su cui fare affidamento ciecamente.

1.5.3 Scene understanding

La ***scene understanding*** è un'elaborazione dello *spatial mapping* che sfrutta tecnolgie di intelligenza artificiale (AI) per trasformare i triangoli generati precedentemente in superfici piatte, senza fori, che offrono un'API ad alto livello dello *spatial awareness* per lo sviluppo di applicazioni che non hanno

grosse esigenze in termini di latenza, ma che prediligono la precisione e risultati più accettabili, in particolare le funzioni messe a disposizione sono le stesse dello *spatial mapping* ma con tempistiche e qualità diverse:

- **Placement**: i muri e gli oggetti sono ottimizzati per accogliere il posizionamento di ologrammi, espongono tramite API funzionalità per appendere oggetti ai muri o appoggiarli su tavoli, in questo caso non dobbiamo preoccuparci di riempire i buchi nelle superfici in quanto di coprirli se n'è occupata l'AI, nel caso disattivassimo l'auto inferenza delle superfici invece avremmo la possibilità di posizionare gli oggetti solo dove lo *spatial mapping* ha avuto successo.

- **Occlusion**: Il modo migliore di usarle dinamicamente è quello di sfruttare lo *spatial mapping*, ma nel caso l'applicazione dovesse sopportare ritardi in termini di latenza allora è fortemente consigliato l'uso di *scene understanding* per una maggiore qualità in termini di piani e riempimento di buchi, il caso migliore è sfruttare entrambe le tecnologie per avere un primo risultato rozzo inizialmente dallo *spatial mapping* in attesa dell'elaborazione da parte dell'AI.

- **Physics**: quando implementiamo ologrammi sottoposti alle leggi della fisica dobbiamo preoccuparci della qualità delle superfici rilevate, in questo caso ha molto senso sfruttare la *scene understanding* non solo per ottenere superfici migliori e senza buchi, ma anche per la possibilità di avere superfici più ampie nello spazio, riducendo quindi i casi in cui un oggetto finisce ai bordi del mondo rilevato.

In generale questo approccio è computazionalmente più oneroso e richiede più tempo per dare risultati, ma in tutti i casi in cui questo non dovesse essere un problema allora è fortemente consigliato l'uso di *scene understanding* per una migliore esperienza da parte dell'utente e minori preoccupazioni implementative da parte del programmatore, il caso ideale è quello di inizialmente sfruttare i risultati grezzi dello *spatial mapping* per dare all'utente una prima percezione dell'ambiente, poi una volta pronti i risultati di *scene understanding* migliorare le superfici virtuali con quelle informazioni.

1.5.4 Gli ologrammi

Alla base della realtà mista ci sono gli ologrammi, ogni cosa visualizzata, ogni elemento aggiunto alla realtà sono uno o più ologrammi, questi vengono generati da luce e proiettati di fronte agli occhi dell'utente, non è infatti possibile avere ologrammi neri, questi non vengono visualizzati e possiamo pensarli

come trasparenti, esistono ologrammi di ogni tipo variando in apparenza e comportamento, alcuni sembrano solidi e reali altri invece cartonati ed eterei, possono essere usati per evidenziare caratteristiche nell'ambiente o per rappresentare immagini e modelli tridimensionali, in generale sono molto flessibili e possono essere sfruttati per tantissime funzionalità diverse. Gli ologrammi possono essere *world-locked* 1.5.1 nel caso in cui siano fissi nel mondo indipendentemente dalla posizione dell'utente o *body-locked* quando invece seguono l'utente nei suoi movimenti, nel caso siano ancorati ad una *spatial anchor* allora possiamo aspettarci di ritrovarli nella stessa posizione anche tra una sessione e l'altra dell'applicazione, in generale alcuni consigli da seguire per evitare di far vivere un'esperienza frustrante all'utente che li usa:

- Evitare l'uso di ologrammi *display locked*, questi spesso risultano essere in mezzo e appiccicosi per l'utente il quale spesso tenterà di scrollarseli di dosso per non vederli più, in generale l'uso è sconsigliato.

- Al suo posto preferire l'uso di ologrammi *body locked*, questi seguono la persona e si posizionano attorno all'utente, in questo modo esso può spostarsi tranquillamente nell'ambiente senza temere di perderli e sa sempre dove trovarli semplicemente abbassando lo sguardo, per implementare il *follow* spesso si preferisce dare una sorta di effetto elastico agli ologrammi che inizialmente rimangono indietro per poi recuperare posizione più lentamente, in questo modo non risultano incollati e la loro presenza non disturba l'utente.

- Posizionare gli ologrammi ad una distanza minima di 1-2 metri, in questo modo si evitano fastidiosi tagli degli elementi e la nauseabonda sensazione che essi ti stiano entrando in testa, è buona norma applicare effetti di *fading* agli ologrammi che si avvicinano troppo per evitare di vederli sparire tutti in una volta o di entrarci dentro.

Sin dalla realtà aumentata il posizionamento degli ologrammi e lo *spatial awareness* sono stati suddivisi in due grandi categorie, perseguendo lo stesso obiettivo ma realizzandolo in modi diversi [6]:

- **Marker-based**: questa tipologia implica l'uso di *marker* per funzionare, solitamente si fa uso di simboli o immagini con un elevato numero di caratteristiche da estrarre come QRCode, in questo modo è più accurato il risultato finale, questo metodo osserva l'ambiente alla ricerca dei *marker*, una volta averne identificato uno ne tiene traccia con algoritmi di *tracking* e impone sopra ad esso l'ologramma che si vuole rappresentare. Per realizzare questo sistema l'applicativo ha bisogno di conoscere la

posizione, l'orientamento e le dimensioni che dovrà avere l'oggetto nella scena, informazioni che vengono ricavate dall'analisi dell'estrazione di *feature* dal marker preposto, poi attraverso adatti algoritmi si modella l'ologramma da rappresentare e lo si impone nella scena.

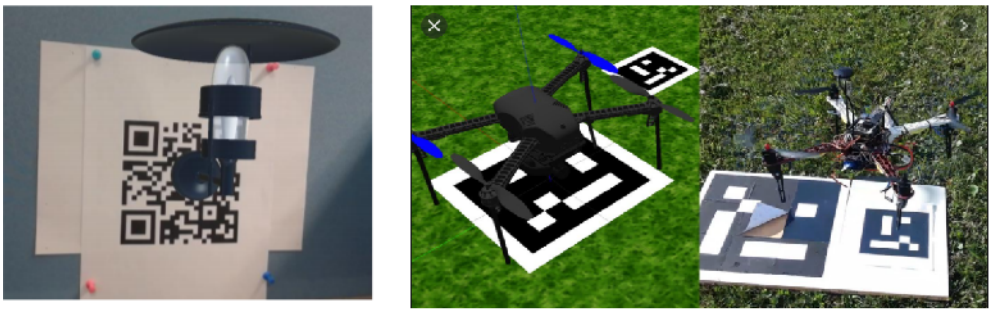

Figura 1.13: Esempi di *marker*

Dalla figura 1.13 vediamo un uso classico dei *marker*, cioè quello di appenderli ad un muro per offrire contenuti aggiuntivi all'utente che visita il posto, nel secondo caso invece come punto di riferimento per un drone che deve volare in un determinato territorio e "mappare" l'ambiente similmente allo *spatial mapping* 1.5.2 visto in precedenza, questo ci fa capire quanto possa essere importante un meccanismo *marker-based*, inoltre considerando note le dimensioni del *marker* e l'altezza del drone è possibile effettuare misure reali del territorio.

- **Markerless**: l'esatto opposto del *marker-based*, non necessita di *marker* appositamente predisposti per il suo funzionamento, ma funziona grazie all'estrazione di *feature* realizzata tramite algoritmi di riconoscimento degli oggetti e in casi più recenti all'uso di intelligenze artificiali. Grazie a questo metodo è possibile posizionare nella stanza ologrammi senza elaborate preparazioni precedenti, questo lo rende il metodo preferito di fissaggio nell'ambiente di ologrammi per la realtà mista, infatti quando parliamo di MR intendiamo una perfetta collaborazione tra realtà fisica e realtà virtuale, non vogliamo che la realtà si adatti a quella virtuale piuttosto il contrario. Una volta posizionato l'ologramma questo non è necessariamente legato al punto iniziale in cui è stato generato, è infatti libero di muoversi all'interno della scena, questo implica l'uso di potenti algoritmi di visione artificiale per essere realizzato, tant'è vero che spesso si tende ad usare un sistema *marker-based* proprio per risparmiare sull'uso della *CPU*, considerando però i recenti algoritmi di AI per realizzare lo *spatial awareness* se presente si può demandare il lavoro alla scheda video e ottenere risultati molto promettenti.

In generale gli obiettivi raggiungibili sono gli stessi con anche la stessa qualità, probabilmente un sistema *marker-based* risulterà sempre più stabile rispetto ad uno senza *marker*, ma non è questo su cui bisogna basare la scelta di quale meccanismo usare, piuttosto è necessario considerare che in un caso abbiamo un sistema leggero, preciso ma limitato nelle possibilità, mentre con un sistema *markerless* non si hanno limiti dati dai *marker* ma è necessaria una notevole potenza computazionale non sempre presente su tutti i dispositivi.

1.6 Hololens 2

Hololens 2 è un visore *Head-attached* 1.4.1 che mostra ologrammi all'utente proiettandoli su lenti poste direttamente davanti ai suoi occhi e trasparenti per non impedire la vista sulla realtà 1.3.2, grazie ai sensori presenti su di esso è in grado di riconoscere l'ambiente che lo circonda realizzando la *spatial awareness* o le mani della persona che lo indossa per farla interagire secondo una metodologia *hands-free* con il visore stesso, l'unione di queste caratteristiche rappresenta un nuovo sistema di *human-computer interaction* (HCI) in grado di immergere l'utente in un mondo che rappresenta il perfetto equilibrio tra realtà fisica e realtà virtuale, in un ambiente in cui è possibile spostare ologrammi come fossero oggetti fisicamente presenti o modellarli come fossero immagini in un computer, osservarli obbedire alle leggi della fisica cadendo e rimbalzando su superfici reali oppure usarli per realizzare situazioni impossibili come oggetti che fluttuano in assenza di gravità o vedere attraverso i muri. Una caratteristica molto usata in ambito professionale è la possibilità di riconoscere gli strumenti fisici presenti davanti all'utente e attraverso ologrammi evidenziare quelli utili e come usarli per guidare la persona nel suo lavoro, in alcuni casi grazie a questo metodo si son registrati incrementi nella produzione addirittura fino al 30% senza alcun effetto collaterale derivante dalla lunga esposizione al visore.

1.6.1 Componenti hardware

Come si vede dalla figura 1.14 il visore è composto da diverse componenti ognuna delle quali collabora per realizzare le funzionalità appena descritte, di seguito suddivise per categoria un riassunto sulle specifiche dell'*hardware*:

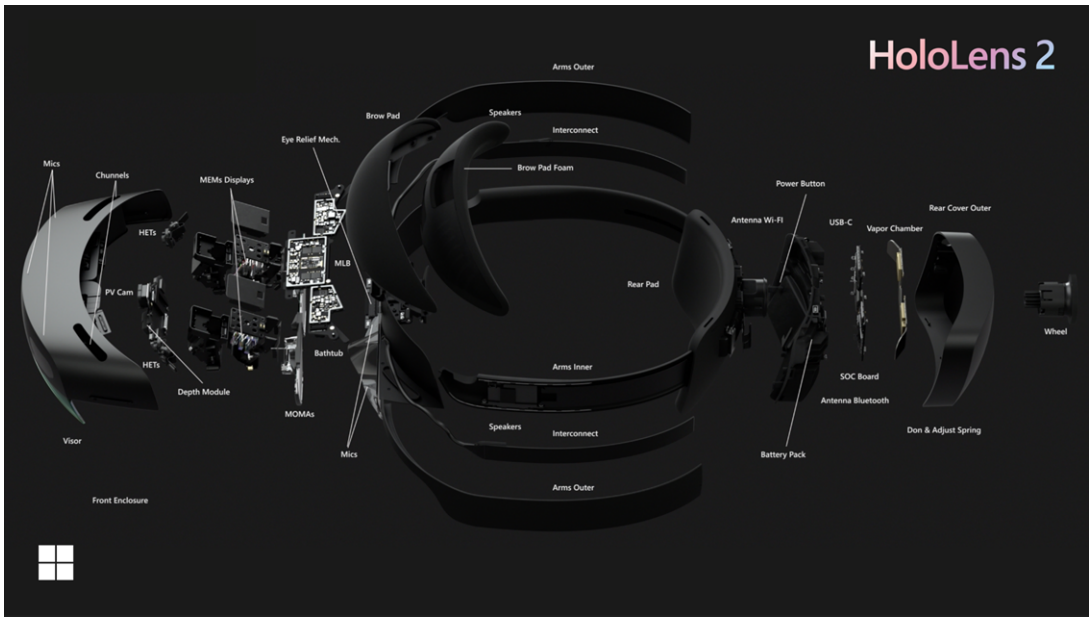

Figura 1.14: Esploso di Hololens 2

- **Display**: lenti olografiche *see-through* 1.3.2 con risoluzione 2K e rapporto 3:2, ottimizzate per la visualizzazione 3D in base alla posizione degli occhi.

- **Sensori**: una video camera da 8 Mp per registrare video a 30 fps, quattro camere visibili per il tracciamento della testa, due camere a infrarossi (IR) per il tracciamento degli occhi, un sensore di profondità *time-of-flight* (ToF) a 1 Mp, accelerometro, giroscopio e magnetometro.

- **Audio e microfono**: casse integrate per la riproduzione spaziale, un *array* di microfoni a cinque canali.

- **Computazione e connettività**: SoC *Qualcomm Snapdragon 850*, *Holographic Processing Unit* (HPU) di seconda generazione, 4 GB LPDDR4x di DRAM, 64 GB di UFS 2.1 per la memoria non volatile, Wi-Fi 5 (802.11ac 2x2), bluetooth 5 e USB Type-C.

- **Forma**: taglia unica adattabile tramite manopola posteriore, indossabile con occhiali da vista e pesante 566g.

- **Alimentazione**: alimentato da batterie al litio in grado di fornire un'autonomia di 2-3 ore di utilizzo attivo, USB-PD per ricaricare velocemente e raffreddamento passivo cioè senza uso di ventole.

1.6.2 Elementi software

Al paragrafo 1.6.1 abbiamo visto quali sono le specifiche hardware dei componenti all'interno del visore, di seguito invece esploriamo le potenzialità applicative che mette a disposizione, le stesse che rendono Hololens 2 un visore a realtà mista:

- **Tracciamento utente**: Hololens 2 non tiene traccia solamente della posizione nel mondo della testa dell'utente, ma grazie ad un modello completamente articolato di due mani è in grado di riconoscere persino le dita della persona, grazie a questo è possibile manipolare gli ologrammi in tempo reale senza il bisogno di ulteriori impedimenti fisici. Grazie alle camere IR è possibile tracciare lo spostamento degli occhi per realizzare sistemi di HCI basati sul *tracking* degli occhi, in questo modo è possibile realizzare applicazioni con le quali si può interagire semplicemente osservando gli ologrammi. Per semplificare ulteriormente l'interazione utente-applicazione è possibile sfruttare i servizi di riconoscimento vocale limitando al minimo i gesti fisici che l'utente è costretto a fare.

- **Comprensione dell'ambiente**: il visore realizza un tracciamento a 6 gradi di libertà per conoscere in tempo reale la sua posizione nel mondo, inoltre tramite le camere ToF è in grado di mettere in atto lo *spatial mapping* per registrare l'ambiente che lo circonda e nel caso mostrare *mesh* di ologrammi per evidenziare le superfici, l'unione di questi due aspetti è ciò che permette di tracciare nel mondo la posizione del visore ad alta precisione e di conseguenza visualizzare gli ologrammi in maniera stabile e affidabile. La camera a 8 Mp presente sul visore permette di registrare video a 30 fps, è inoltre possibile decidere se nelle fotografie scattate o nel video ripreso debbano comparire o meno gli ologrammi in quel momento visibili, i *file* generati possono poi essere condivisi con altri dispositivi per prenderne visione in un secondo momento oppure in *real time* per implementare video chiamate o simili.

- **Applicazioni preinstallate**: quando si avvia Hololens 2 per la prima volta nel menù messo a disposizione dal sistema operativo *Windows Holographic Operating System* sono presenti diverse applicazioni già installate utili a dimostrare le potenzialità del visore, tra queste troviamo "3D Viewer" che permette di osservare modelli tridimensionali anche in movimento, "Dynamics 365 Remote Assist" per la realizzazione di video chiamate mirate al supporto di chi indossa il visore e il classico "Microsoft Store" per scaricarne altre nel caso quelle già presenti non siano

sufficienti. Un elenco completo delle applicazioni preinstallate lo si trova nel sito ufficiale di Microsoft[6].

[6] https://docs.microsoft.com/en-us/hololens/hololens2-hardware

Capitolo 2

Applicazioni di Mixed Reality in ambito sanitario

Dopo aver definito la *Mixed Reality* e classificato le varie tecnologie ad essa inerenti, si descrivono di seguito esempi di applicazioni note in letteratura che adoperano questi strumenti in ambito *healthcare*, con l'obiettivo di sostituirsi ai classici trattamenti farmacologici oppure di aiutare il medico nel suo lavoro, mostrandogli informazioni inerenti al paziente e guidandolo attraverso ologrammi ad una corretta e precisa esecuzione di ciò che deve fare. L'esplorazione in questo senso è stata fondamentale per poter definire il progetto di tesi e le sue componenti, orientandosi verso le applicazioni a supporto dei medici si denota una caratteristica in comune a tutti i progetti nel guidarli attraverso ologrammi, evidenziando caratteristiche sull'interno del paziente 2.2.3, mostrandogli dove incidere 2.3.1, dove impiantare viti 2.3.2 o video derivanti da sonde 2.3.3, in generale lasciando in secondo piano la possibilità di visualizzare informazioni da sensori. È invece obiettivo del progetto di questa tesi fornire una base di partenza nella quale sono presenti informazioni utili e strumenti comuni ad ogni intervento come la possibilità di leggere i parametri vitali del paziente o effettuare video chiamate verso l'esterno della sala operatoria, lasciando la possibilità a sviluppatori futuri di integrare il sistema con funzionalità particolari a supporto di specifici interventi, eludendo alla necessità di partire da zero ogni volta che si vuole sviluppare un'applicazione di questo tipo.

2.1 Trattamento di patologie e fobie

Il nostro cervello è molto bravo a immagazzinare e recuperare ricordi [1] basandosi su suoni, sentimenti, odori e altri aspetti dell'ambiente che ci circonda, un esempio lampante di questa capacità lo abbiamo ascoltando un brano mu-

sicale risalente alla nostra gioventù, subito nei nostri pensieri affiorano ricordi che spesso dimentichiamo addirittura di avere, nella letteratura ci si riferisce a questo effetto con il termine medico *episodic autobiographical memories* (EAMs) funzioni involontarie del sistema percettivo e memoria dell'umano che possono portare a grandi benefici se legati a bei ricordi o a terribili conseguenze se in relazione ad un ambiente ed un'esperienza traumatica come ad esempio un campo di battaglia in guerra. Caratteristiche del nostro cervello come questa sono quelle che si tentano di sfruttare attraverso l'uso di un visore in realtà virtuale, immergendo la persona in ambienti particolari, affiorando ricordi, plasmandone di migliori.

2.1.1 Disturbi post traumatici

L'esposizione a eventi di forte stress, campi di combattimento o a tutta una serie di esperienze traumatiche come lo stupro, l'abuso, la violenza fisica o le minacce a mano armata sono solo alcuni degli esempi a seguito dei quali è possibile sviluppare una malattia mentale nota come *post-traumatic stress disorder* (PTSD). Il trattamento più usato ed empiricamente dimostrato essere il più efficace è noto in letteratura come **prolonged-exposure therapy** una procedura suddivisa in *imaginal* sotto la guida di un terapista raccontare gradualmente, controllatamente e ripetutamente la propria esperienza traumatica ed *in vivo* l'esposizione a situazioni, attività o oggetti caratteristici della scena traumatica vissuta, entrambi le parti sono necessarie alla persona in questione per valutare ed emotivamente elaborare ciò che ha vissuto ed arrivare infine a superare eccessivi livelli di ansia e/o paura. L'applicazione di visori a realtà virtuale entra in gioco proprio sul secondo punto *in vivo*, è obiettivo di **Bravemind**[1] immergere il soggetto in questione in un ambiente totalmente virtuale sfruttando non solo rappresentazioni e suoni tipici dei campi di battaglia ma anche display tattili e olfattivi, mettendo a disposizione un sistema dinamico e regolabile in grado di adattarsi alle esigenze del paziente, per rivivere esperienze di guerra traumatiche in modo controllato e sicuro ottenendo risultati promettenti in termini di totale o parziale guarigione del paziente. Studi su questa metodologia hanno dimostrato una maggiore efficienza rispetto ai metodi tradizionali, portando alla diffusione di Bravemind in più di 50 strutture in tutti gli Stati Uniti tra cui ospedali e università per la ricerca sulla cura dei PTSD.

[1] `https://ict.usc.edu/prototypes/pts/`

2.1.2 Fobie

Chiunque abbia mai sofferto di una fobia e sia riuscito poi a risolverla sa perfettamente quanto sia importante l'esposizione a ciò che ci spaventa gradualmente e senza alcun tipo di pretesa, la cura di una fobia è un processo lento e complicato, proprio perché le paure caratterizzanti una fobia sono spesso irrazionali non è sufficiente immergere la persona nell'ambiente che più lo mette in soggezione e aspettarsi che razionalmente capisca che non ha nulla di cui temere, ma è necessaria una progressiva e controllata esposizione, esperienze che con il passare del tempo instaurano un rapporto diverso tra il soggetto in questione e la sua fobia, in letteratura nella psicologia ci si riferisce all'esposizione controllata per guarire una fobia con il termine **desensibilizzazione sistematica** che sottolinea come questa sia graduale, ripetitiva e imponga al nostro cervello un cambiamento di prospettiva con il passare del tempo. In questi casi la realtà virtuale prende piede proprio nell'aspetto dell'esposizione controllata, la *Virtual reality exposure therapy* (VRET) viene utilizzata per immergere il paziente in ambienti particolarmente significanti rendendo inoltre possibile la realizzazione di scenari che nella realtà sarebbero troppo costosi o di situazioni particolari che non si potrebbero simulare altrimenti, l'azienda **Virtually better**[2] lavora in questo settore da più di 20 anni e ha realizzato diversi ambienti in realtà virtuale proprio per sconfiggere paure come il volare 2.1, le altezze, i temporali e altre fobie ricorrenti.

Figura 2.1: Esempio di Virtual reality exposure therapy

Una ricerca svolta nel 2015 [1] ha sottolineato due aspetti positivi della VRET, nei questionari somministrati al paziente dopo questa tipologia d'esposizione si vedono miglioramenti rispetto a prima del trattamento e nell'osservazione a lungo termine delle condizioni del paziente non si notano discrepanze tra il benessere di uno che è guarito tramite tecniche tradizionali d'esposizione

[2] `https://www.virtuallybetter.com/`

in vivo e chi ha fatto uso di VRET, questo unito al fatto che un'esposizione virtuale è decisamente meno costosa di molte dal vivo, rendono di fatto l'uso della realtà virtuale una scelta furba, intelligente ed efficace.

2.1.3 Ingannare il cervello

Quando indossiamo un visore in realtà virtuale abbiamo la sensazione di essere in un mondo totalmente differente e non solo, il nostro cervello si lascia convincere che ciò che vediamo rappresenti la realtà, facendosi condizionare e rispondendo agli stimoli visivi di conseguenza, questa forte caratteristica dei visori virtuali apre la strada ad una serie di *brain tricks*[8] rivolti alla salute della persona spesso sostituendosi ad una cara cura farmacologica tra cui:

- **Gestione del dolore da bruciatura**: l'immersione di un paziente vittima di scottature in un campo di neve dove può giocare con essa e divertirsi ha dato gli stessi effetti sulla gestione del dolore della morfina, il Dr. Brennan Spiegel dice: *"If the brain is anxious or upset, it wants to keep track of pain. If the brain is calm and relaxed, it doesn't have time to feel pain."*, una mente distratta distratta dal dolore è un cervello che non sente più quel dolore, il visore in realtà virtuale funziona esattamente come gli oppiacei, interrompe il flusso di dati passante per i nervi impedendo di fatto al paziente di percepire dolore. Anche solo giocare a *Beat Saber* che con le bruciature non ha nulla a che fare ha dato ottimi risultati, il paziente conferma di aver smesso di provare dolore mentre giocava e che persino dopo la rimozione del visore questo effetto anestetico non sia svanito, risultati promettenti come questo in sostituzione alla morfina significano soldi risparmiati in medicine e soprattutto l'azzeramento del rischio di creare dipendenze al paziente da droghe.

- **Demenza**: una condizione neurologica che colpisce principalmente la memoria a lungo e a breve termine, causando di conseguenza altre complicazioni come la perdita della capacità di parlare e solitamente legato ad uno stato depressivo del paziente conscio di quello che sta vivendo, il metodo tradizionale per aiutare le persone che soffrono di questa malattia è quello di fargli vedere immagini o oggetti che li riportino alla loro infanzia, questa somministrazione di ricordi visivi aiuta il cervello a ricreare collegamenti persi e di conseguenza a rallentare l'effetto invasivo della malattia, l'uso di un visore a realtà virtuale può immergere totalmente il paziente nella sua infanzia ricostruendo non solo immagini o oggetti, ma tutto l'ambiente circostante, questa metodologia si è dimostrata molto efficace in termini di velocità di recupero dei ricordi e benessere del paziente.

- **schizofrenia**: tutti noi abbiamo in testa una voce, il nostro pensiero, se ci pensiamo siamo anche in grado di modificarne il tono simulando quella di un'altra persona, questo capita addirittura involontariamente mentre leggiamo un libro o pensiamo a canzoni che ascoltiamo spesso, la schizofrenia è quella condizione per cui una persona ha in testa pensieri e voci a cui non sta esplicitamente pensando, ma che sente come se altre una o più entità pensassero con il suo cervello, solitamente seguito dall'incapacità di svolgere le normali attività di vita portano il paziente ad allontanarsi dalla realtà e a chiudersi in una tutta loro, l'uso di un visore a realtà virtuale da la possibilità al paziente di visualizzare *avatar* rappresentanti le loro voci immaginarie per trasformarle da qualcosa di estraneo a volti conosciuti dando loro una faccia che possa aiutarli, come confermato dal Dr. Spiegel questa metodologia ha dato risultati statisticamente migliori del classico uso di farmaci di inibizione del pensiero.

- **Partorire**: non importa quanto il dolore sia forte, persino durante il parto il cervello può essere distratto in un effetto anestetico, l'uso di un visore durante questa procedura ha portato non solo ad una diminuzione del dolore percepito, ma anche ad una percezione più veloce del tempo passato in sala, lo stesso modo in cui un videogiocatore perde la cognizione del tempo durante una partita al suo gioco preferito.

- **Ictus**: causato dall'imprevista chiusura o rottura di un vaso cerebrale è la mancata ossigenazione di parte del cervello che comporta temporanee perdite dell'uso degli arti o nei casi peggiori di immobilizzazioni di parti del corpo anche permanenti, il Dr. Sook-Lei Liew direttore del laboratorio di *Neural Plasticity and Neurorehabilitation* usa a sua volta la rappresentazione di *avatar* comandati dall'EEG del paziente per mostrargli il loro arto paralizzato muoversi, ha dimostrato poi come tale trattamento porti in alcuni casi il paziente a riprendere le abilità perse durante l'ictus e quindi a muovere nuovamente l'arto incriminato.

- **Disturbi alimentari**: uno dei modi per curare un disturbo alimentare è dare la consapevolezza al paziente di quello che si sta facendo, non solo in termini di aspetto fisico, ma dar loro una prospettiva più dettagliata di quello che succede all'interno del loro corpo, essere a conoscenza delle conseguenze delle proprie azioni è parte fondamentale per correggerle, finché la pressione alta è solo un numero il paziente non è invogliato a seguire una dieta povera di sale, prendendo visione di una ricostruzione del suo corpo interno e osservando gli organi degradare si instaura invece nel suo cervello un sistema di sopravvivenza che porta inevitabilmente

a regolare lo stile con cui mangia. Un esempio d'uso di un visore a realtà virtuale è quello usato con pazienti che soffrono d'anoressia e che si giustificano dicendo: "mi vedo troppo grasso/a", una volta averlo indossato il paziente vedrà rappresentato al posto del suo corpo uno con fattezze normali, in questo modo si cambia la prospettiva sulla visione del proprio aspetto riadattando l'idea di fisico sano e abituando la persona a vedersi in un corpo sano, in questo modo alla rimozione del visore il soggetto osserverà il proprio fisico da un altro punto di vista finalmente notando di non essere grasso bensì estremamente magro.

2.2 Addestramento del personale medico

Un buon medico si distingue anche per il tempo impiegato ad allenarsi e dalla qualità degli strumenti che ha usato per farlo, ovviamente studiare è fondamentale alla realizzazione di un dottore, ma sopratutto nella chirurgia la pratica è necessaria a formare un medico in grado di operare tranquillamente e in sicurezza. Un grosso limite della chirurgia risiede nel fatto che per far pratica è necessario un paziente e che non è sicuro far operare una persona da un chirurgo inesperto, questo inevitabilmente richiede l'ausilio di simulatori che sostituiscano la persona sotto i ferri per permettere al medico di sbagliare e imparare dai propri errori e quindi di migliorare, le prime simulazioni sono state fatte sfruttando oggetti che rispecchiassero il più possibile la realtà, per poi arrivare alla virtualizzazione come vedremo di seguito:

2.2.1 HelpMeSee

In accordo con *World Health Organization* (WHO) la maggior causa di cecità nel mondo è legata alla non cura della cataratta, una patologia che porta all'annebbiamento delle lenti naturali dell'occhio riducendo la trasmissione di luce alla retina, un dato risalente al 2014 mostra che la metà delle persone non vedenti al mondo soffrono di questa patologia e sono circa 20 milioni, nella maggior parte dei casi come conseguenza dell'invecchiamento della persona e in alcuni bambini nati con questo problema. Normalmente per risolvere questo problema è sufficiente un intervento di breve durata di sostituzione delle lenti, ma questo non è sempre possibile nei paesi meno sviluppati e la continua crescita dell'esigenza di questo intervento rendono pochi i chirurghi in grado di farli, per risolvere questo divario l'organizzazione nonprofit **HelpMeSee**[3] ha sviluppato un simulatore chirurgico usato per insegnare velocemente ed efficacemente le procedure per la sostituzione delle lenti e ristabilire la corretta

[3] `https://helpmesee.org/`

vista. Il simulatore successivamente chiamato *Manual Small Incision Cataract Surgery* (MSICS) è usato per formare specialisti in grado di operare la cataratta in soli 5 minuti negli adulti e 15 per i bambini, per raggiungere questo obiettivo il sistema è dotato di un display stereoscopico ad alta definizione attraverso il quale la persona che sta studiando vede rappresentato un preciso modello dell'occhio umano ed è in grado di interagirci con attraverso i classici strumenti che si userebbero in un vero intervento.

Figura 2.2: Simulatore Manual Small Incision Cataract Surgery MSICS

Come si vede dalla figura 2.2 lo specialista è seduto sulla sedia e guarda dentro al visore esattamente come succederebbe nella realtà, questo unito a precisi *haptic displays* [3] rendono la simulazione pressoché identica ad una situazione reale, considerando inoltre le 240 diverse simulazioni installate nel dispositivo rappresentanti tutti i casi che lo specialista potrebbe trovarsi ad affrontare in un vero intervento, si arriva ad un livello di preparazione tale da poter operare tranquillamente pazienti veri in sicurezza.

2.2.2 Simodont

In tutte le scuole per dentisti nel mondo gli studenti sono soliti sviluppare le loro doti cliniche utilizzando trapani e altri strumenti su denti di plastica montati nei famosi *phantom heads*, questa pratica è innanzitutto costosa e comporta perdite di tempo nella preparazione del modello, inoltre sotto un

punto di vista educativo è limitato nelle capacità di rappresentazione di situazioni reali, limitandosi ai casi base dell'odontoiatria risulta essere una metodologia non sufficiente alla formazione di veri dentisti, a questo proposito la collaborazione tra *Moog Industrial Group* e *Academic Centre for Dentistry in Amsterdam* (ACTA) ha portato a sviluppare un sistema meno costoso e più efficace per l'addestramento chiamato **Simodont**[4]. Combinando la visualizzazione 3D e tecnologie di *force feedback* hanno realizzato un sistema virtuale 2.3 osservabile tramite occhiali stereo polarizzati rappresentante un modello tridimensionale della bocca di un paziente con il quale attraverso strumenti realistici si può interagire e vedere in tempo reale l'effetto dei trapani sui denti, ricevendo indietro risposte tattili come la vibrazione o suoni emessi dall'operazione, considerando inoltre che la posizione tenuta dallo studente che usa questo strumento è la stessa che avrebbe un dentista in una vera operazione otteniamo una simulazione pressoché uguale alla realtà.

Figura 2.3: Simodont dental trainer

Anche in questo caso ACTA ha sviluppato e messo a disposizione svariati scenari rappresentanti situazioni reali attraverso i quali gli studenti possono formarsi come se stessero operando su un vero paziente, inoltre il professore che li segue è in grado di rivedere passaggi effettuati in precedenza per valutarne l'operato o dar loro consigli su come migliorare, questo sistema rappresenta un'innovazione provata essere talmente di successo da essere stato adottata ormai nella maggior parte delle scuole in tutto il mondo.

[4] `https://www.simodontdentaltrainer.com/`

2.2.3 Vascular imaging

Una delle operazioni al giorno d'oggi più effettuata è accedere ad una vena per estrarre il sangue di un paziente o iniettare terapie farmacologiche, nonostante la frequenza con cui venga fatta questa semplice manovra rimane in determinati casi comunque una sfida riuscire a trovare il punto adatto dove forare. Certamente esistono metodi che risaltano le vene interrompendo il flusso sanguigno o aprendo e chiudendo il pugno, ma non sarebbe più comodo un sistema che ti mostra esattamente dove sono? Questo è l'obiettivo di **Eyes-On Glass**[5] sviluppati da Evena, occhiali in realtà aumentata che rappresentano direttamente sul paziente le vene evidenziate di modo che chi li indossa ne abbia visione certa, questi per sapere l'esatta posizione delle vene sfruttano 4 fonti luminose *near-infrared* (NIR) con una lunghezza d'onda tra i 600 e i 1000 micrometri per illuminare l'area del corpo dove forare, è noto che tale tipologia di luce venga assorbita dal sangue presente nelle vene è quindi sufficiente analizzare attraverso camere adatte il ritorno di questi fasci luminosi per scoprire che le vene coincidono esattamente con i punti più scuri dell'immagine percepita, a questo punto gli occhiali elaborano i dati ricevuti e mostrano al medico ologrammi sulle vene del paziente.

Figura 2.4: Evena Eyes-on glass affiancati dalle immagini catturate da speciali sensori

Come si vede in figura 2.4 le due camere montate sul display percepiscono immagini in bianco e nero di quel tipo, utilizzando algoritmi di visione artificiale è semplice separare le vene dal braccio e mostrarne al medico una ricostruzione tridimensionale basata su dati raccolti in tempo reale, considerando che tra tutti i prelievi fatti in più del 50% dei casi è necessario forare più di una volta proprio per una mancanza di visibilità è facile pensare quanto

[5] `https://evenamed.com/products/glasses/`

questi occhiali possano fare la differenza in termini di velocità, precisione e di conseguenza minori complicazioni.

2.3 Realtà Mista a supporto di operazioni chirurgiche

L'uso di Hololens in sala operatoria si è dimostrato essere una metodologia promettente, l'integrazione di ologrammi per guidare il chirurgo nell'operazione o per mostrare informazioni in modo agevole ha portato in diversi casi ad un miglioramento sia in termini di tempo che precisione e qualità dell'intervento non scontati né ignorabili, in questo capitolo si approfondisce qualche esempio preso dalla letteratura dell'uso di questa tecnologia evidenziando obiettivi e risultati ottenuti.

2.3.1 Chirurgia ricostruttiva con vasi perforanti

L'obiettivo principale di questa applicazione [16] è quello, attraverso la realtà mista di Hololens, di identificare con precisione il punto dove incidere per mettere in atto una *vascular pedunculated flaps* visualizzando la struttura interna del paziente direttamente nel suo corpo in una sorta di vista a raggi X. Per rendere possibile tutto questo è necessario prima di entrare in sala operatoria che il paziente venga sottoposto ad una *Computed Tomography Angiography* (CTA) con contrasto, i dati risultanti dalla CT grazie al software di Vitrea[6] vengono segmentati in pelle, ossa, muscoli e modello vascolare per poi essere riuniti in un modello DICOM e manualmente colorati grazie all'uso di *mesh* per riconoscere al volo quello che si sta guardando, a questo punto è possibile caricare il risultato finale nel visore Hololens ed entrare in sala operatoria. Una volta avviata l'applicazione il visore mette a disposizione del chirurgo il modello 3D della gamba del paziente che dovrà far combaciare ruotandolo e traslandolo alla vera gamba, da notare che la scalatura non è un'operazione citata appositamente in quanto il modello virtuale è già adatto alla realtà, da questo momento in poi il chirurgo è in grado di vedere l'interno della gamba senza mai averla ancora incisa.

[6] `https://www.vitalimages.com/enterprise-imaging-solution/advanced-visualization/`

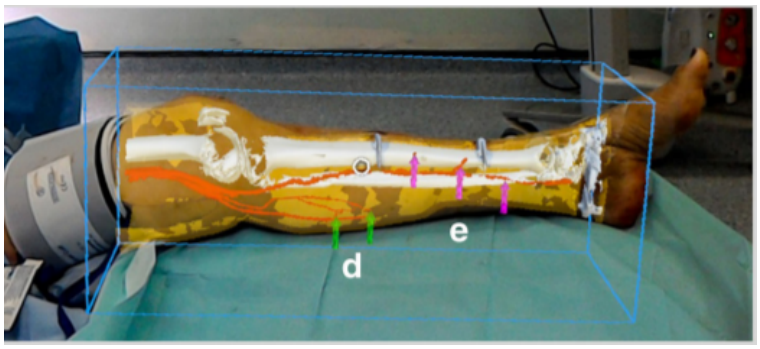

Figura 2.5: Risultato finale di sovrapposizione del modello sulla gamba

A questo punto il chirurgo può trasferire da Hololens alla pelle del paziente il punto esatto dove incidere utilizzando un pennarello sterile e iniziare l'intervento, questa procedura è stata effettuata su 6 diversi pazienti consensuali, solo in un caso in cui determinate condizioni hanno portato allo spostamento dei tessuti interni si è registrata una discrepanza di 1 cm tra la rappresentazione dell'ologramma e la realtà, in tutti gli altri casi la posizione data da Hololens e quella verificata tramite *audible Doppler ultrasound* non hanno presentato incongruenze visibilmente percettibili. I chirurghi che hanno partecipato a questo esperimento hanno evidenziato quanto questo sistema sia molto più accurato e veloce rispetto all'uso delle vecchie strumentazioni, le implementazioni future che gli sviluppatori pensano di mettere in atto sono una semplificazione dell'iter dalla CTA fino al caricamento del modello in Hololens e il posizionamento automatico del modello 3D sopra alla gamba del paziente per farla combaciare perfettamente.

2.3.2 Navigazione per viti peduncolari

Un intervento noto e spesso effettuato è la *spinal fusion*, un'operazione consigliata per correggere la scoliosi e necessaria in pazienti aventi situazioni degenerative, in tutti i casi la precisione nel posizionare e impiantare le viti peduncolari deve essere elevata per evitare di arrecare danni permanenti alla persona operata, a tal proposito nascono i sistemi di navigazione in supporto al chirurgo che mostrano la posizione esatta dove incidere, questo è il punto in cui si colloca il progetto di questo articolo [9] l'implementazione di un sistema di navigazione per l'impianto di viti peduncolari realizzato tramite la visualizzazione *in situ* offerta da Hololens. Come altre tecniche preesistenti anche questa applicazione necessita di una CT preoperatoria per raccogliere informazioni sull'anatomia della spina dorsale del paziente, ottenuti questi risultati e caricati sul dispositivo è possibile entrare in sala operatoria e iniziare l'intervento, una volta esposte le vertebre sulle quali intervenire (in questo

caso nella zona lombare) il chirurgo deve marcare le vertebre visibili utilizzando lo strumento raffigurato nella figura 2.6.A e confermare verbalmente la selezione fatta, a questo punto il sistema è in grado di localizzare i fori dove effettuare l'intervento e di mostrarli attraverso *marker* direttamente sulle ossa del paziente 2.6.D.

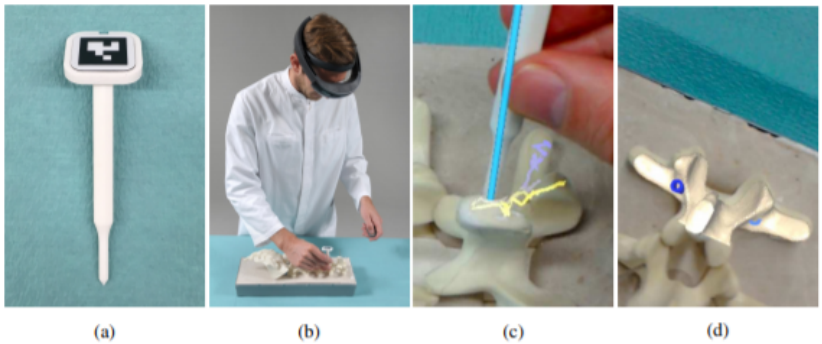

Figura 2.6: Passaggi preparazione intervento di fusione spinale

Da questo momento in poi il sistema ha tutte le informazioni necessarie per guidare il chirurgo durante il piazzamento delle viti peduncolari, tenendo in una mano il trapano per il posizionamento dei *K-wire* e nell'altra lo strumento 2.7.A di navigazione è possibile applicare comodamente e precisamente i *K-wire* nei punti evidenziati 2.7.C facendo uso delle informazioni sull'angolazione ottenuta dallo strumento di navigazione 2.7.D

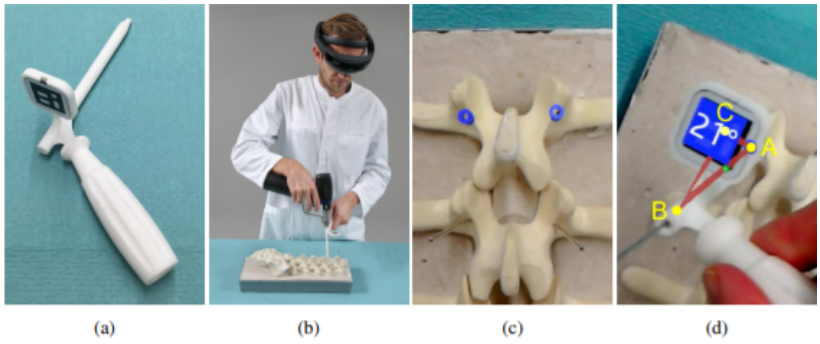

Figura 2.7: Punti di navigazione per l'intervento di fusione spinale

Le sperimentazioni di questo sistema sono state fatte solamente su manichini e mai in una vera sala operatoria, i dati raccolti durante questi test indicano un errore di 3.38±1.73 gradi nella traiettoria d'orientamento e 2.77±1.46 mm nella localizzazione dei punti d'ingresso, le tempistiche per la digitalizzazione della superficie dell'osso si aggirano intorno a 125±27 secondi, per l'impianto

di un *K-wire* 147±55 secondi e per il completamento di una vertebra 419 secondi in media. In 5 casi è stato necessario un riavvio della sperimentazione a causa di un *crash* non identificato del sistema operativo e solo in un caso il chirurgo ha deciso spontaneamente di riavviare valutando la registrazione dei punti poco precisa. I test sui manichini hanno raccolto risultati promettenti, il prossimo passo è la sperimentazione su cadaveri per confermare la precisione su pazienti reali.

2.3.3 Procedure image-guided minimamente invasive di Philips

Philips è un'azienda multinazionale olandese fondata nel 1891 la quale dopo più di 120 anni tra i leader del settore elettronico ha optato nel 2011 ad un cambio di strategia per concentrarsi sempre più sulle nuove tecnologie nell'ambito *healthcare*, ricerca e sviluppo sugli interventi minimamente invasivi è ciò che li ha portati a realizzare IntraSight e Azurion[7] dispositivi montati attorno al letto chirurgico in grado di fornire immagini ad alta risoluzione dell'interno del paziente e allo stesso tempo non essere d'intralcio per il personale medico che sta intervenendo, tali attrezzature sono la porta d'ingresso alla realizzazione di procedure *image-guided* operazioni che non necessitano di chirurgia aperta per la visione diretta dell'intervento, ma che sfruttano le immagini generate dalle macchine per guidare il medico attraverso piccoli fori fino alla zona chirurgica d'interesse nell'operazione.

[7] `https://www.usa.philips.com/healthcare/resources/landing/azurion?npagination=1`

Figura 2.8: Sistema Azurion per la chirurgia cardiovascolare

I brevi tempi di recupero, le ridotte complicanze e l'uso di anestesia locale rendono questa tipologia di interventi preferibili a quelli classici, un paziente post operatorio è noto dover aspettare giorni o addirittura settimane in ricovero all'ospedale per dar tempo alle ferite di richiudersi, oltre ad essere un letto meno a disposizione comporta un peso ulteriore allo staff infermieristico che dovrà occuparsi di medicare le ferite, la rimozione delle grosse incisioni dall'intervento è ciò che ha consentito ad alcuni pazienti di tornare a casa solamente dopo poche ore dall'operazione. Ad oggi non è sempre possibile mettere in pratica questo tipo d'intervento su ogni paziente, ma sono proprio i benefici derivanti da questo approccio ad aver spinto Philips e Microsoft a collaborare nella realizzazione di un sistema più sicuro e preciso per le procedure image-guided minimamente invasive, il contributo di Azurion 2.8 con la sua abilità nel raccogliere informazioni in tempo reale e mostrarle comodamente aveva già giocato un forte ruolo nella fattibilità dell'intervento, ma l'integrazione con Hololens 2 potrebbe essere l'arma finale alla riuscita di queste operazioni, prendere visione dell'interno del paziente attraverso uno schermo e guidarsi con quelle informazioni è ciò che ha consentito fino ad oggi di intervenire abbastanza in sicurezza, ma la rappresentazione di ologrammi "dentro" al paziente simulando una visione di un intervento classico (incidendo e aprendo la zona chirurgica) andrebbe a colmare definitivamente il divario tra schermo e realtà, portando la precisione di queste tecniche ad un livello tale da consentirne l'usufruibilità anche in altri campi chirurgici diversi da quelli attuali. Purtroppo

le ultime novità[8] risalgono al 2019 e non ho trovato altre fonti che ne parlino, al tempo il progetto era solo un'idea e la mancanza di articoli più recenti può far pensare che non sia stato portato avanti, anche se personalmente ritengo sia più opportuno immaginare che data la complessità del sistema non siano ancora giunti ad una prima conclusione. Dai video *concept*[9] pubblicati da Philips si notano altre caratteristiche interessanti del progetto, innanzitutto una forte interazione tra Hololens e gli apparecchi fisici circostanti come ad esempio "prendere" un'immagine da uno schermo e spostarla come ologramma, al momento dell'ecografia vederne l'output virtuale direttamente sul corpo oppure attraverso bottoni virtuali agire su attrezzature come Azurion 2.8, da non sottovalutare poi tutta la predisposizione di informazioni per il chirurgo come i parametri vitali del pazienti, immagini utili all'intervento e i modelli tridimensionali aggiornati in tempo reale, dal video si nota poi che la rappresentazione della guida all'interno del paziente è posta di fronte al chirurgo e non direttamente dentro al corpo dove è naturale pensare che sia, questo probabilmente deriva dal fatto che per un chirurgo è più comodo visualizzarlo in questo modo.

2.3.4 Sistema MR in supporto alla chirurgia addominale

Come ci fa notare immediatamente questo articolo [11] in un intervento addominale il chirurgo ha necessariamente bisogno di avere sott'occhio tutta una serie di informazioni in tempo reale sullo stato del paziente e il paziente stesso, solitamente queste informazioni sono messe a disposizione del chirurgo tramite monitor fisici presenti nella sala operatoria, il problema principale derivante da questo metodo di visualizzazione risiede nel fatto che i monitor non possono essere posti vicini al paziente in quanto andrebbero ad ostruire la libertà del chirurgo costringendolo ogni volta che ne vuole leggere il contenuto a spostare la testa e distogliere lo sguardo dal paziente, questo inevitabilmente interrompe la concentrazione del chirurgo su quello che sta facendo e inoltre costringe i suoi occhi ad un lavoro continuo di messa a fuoco tra monitor e paziente. L'obiettivo principale del sistema di questo articolo è quello di scongiurare i problemi appena descritti relativi alla visualizzazione delle informazioni mettendo in gioco Hololens come visore, DICOM per la virtualizzazione di referti medici 2D o 3D e Spectator View per consentire ad altre persone di visualizzare quello che vede il chirurgo creando una collaborazione tra più medici sullo stesso paziente condividendo gli ologrammi sia attraverso altri visori Hololens sia attraverso cellulari.

[8] `shorturl.at/mHOS1`

[9] `https://www.youtube.com/watch?v=kG9wKUd-jPo`

Figura 2.9: Chirurgia addominale con visori Hololens

Questo sistema è stato sperimentato attivamente su 10 interventi di chirurgia addominale 2.9, in ognuno di essi almeno due persone indossavano un visore e alla fine di ogni intervento è stato somministrato un questionario ai chirurghi per trarre vantaggi e vantaggi del sistema utilizzato, tra i vantaggi si è riscontrata un'opinione abbastanza comune su:

- **MR e registrazione video**: uno degli aspetti cruciali per cui ai chirurghi è piaciuto usare Hololens in sala operatoria è stata la possibilità di visualizzare e maneggiare immagini e modelli 3D, in particolare la possibilità di ancorare al paziente indicatori presenti in tutti i visori per condividere con precisione un punto d'interesse inoltre tale funzionalità è stata sfruttata per marcare le operazioni fatte durante l'intervento, questo assieme alla funzionalità di Hololens di acquisire video ha creato un sistema di *training* del personale chirurgico molto efficace, attraverso appunti, punti d'interesse e registrazioni vocali uno studente è in grado di visualizzare un intervento con gli occhi del chirurgo che lo ha fatto e in più seguire passo passo ogni passaggio rifacendosi agli appunti lasciati durante l'operazione.

- **DICOM viewer**: un componente del sistema in grado di caricare dinamicamente referti medici di test eseguiti prima dell'intervento virtualizzandoli in modelli 3D che il chirurgo può successivamente manovrare a piacimento, ruotandoli o segmentandoli per prendere visione nel modo più efficace, considerando che le tempistiche per caricare un modello di

questo tipo erano di pochi secondi non è del tutto inimmaginabile pensare anche durante l'intervento di poter svolgere esami a raggi X e in tempo reale visualizzarli su tutti i visori.

Purtroppo questo progetto è stato sviluppato utilizzando visori Hololens e non Hololens 2, difatti tra i difetti riscontrati dai chirurghi compaiono nella maggior parte dei casi problemi relativi all'hardware utilizzato, ad esempio il FOV ridotto o l'errore di parallasse nel posizionamento di indicatori ad una distanza minore di 50 cm dal visore lasciando al chirurgo la percezione che l'oggetto si spostasse al variare del punto di vista, un altro problema da non sottovalutare è l'autonomia del visore che in questo caso era di 2 ore su interventi che possono durare anche fino a 5 ore, infine è stata criticata anche l'ergonomia del visore il quale avendo un peso differente davanti e dietro andava a pesare sul collo fino a diventare fastidioso. Tutti questi problemi sono stati parzialmente o interamente risolti grazie ad Hololens 2, l'unico appiglio importante rimane il peso del visore che adesso è bilanciato tra fronte e retro, ma che se usato con la testa inclinata come si vede dalla figura 2.9 a lungo andare può comunque portare a dolori cervicali da sforzo, inoltre non bisogna sottovalutare il problema dell'autonomia della batteria del visore quindi tenere in considerazione che potrebbe scaricarsi.

Capitolo 3

Progettazione e sviluppo di un workbench virtuale in sala operatoria

Alla luce di quanto osservato dallo stato dell'arte 2 dove si denota una predisposizione alla raffigurazione di ologrammi col solo obiettivo di guidare l'operazione in sala operatoria, ho deciso di realizzare in accordo con l'opinione degli esperti del dominio un **workbench virtuale** fruibile tramite Hololens 2 in realtà mista che mostri al chirurgo informazioni relative all'intervento e metta a disposizione funzionalità utili in sala operatoria in maniera più agevole rispetto ai metodi tradizionali o altrimenti non realizzabili per problemi di spazio o intralcio.

La scelta della realtà mista non è casuale, essa infatti offre nuovi sistemi di *human computer interaction* (HCI) che attraverso l'uso di strumenti virtuali interfacciano l'utente ad un ambiente aumentato in modalità *hands-free*, tale caratteristica in ambito sanitario e soprattutto in quello chirurgico è di fondamentale importanza, la possibilità di utilizzare strumenti virtuali è sinonimo di velocità, sicurezza e portabilità, un ologramma non si rovina nel tempo, non può essere urtato, non può scivolare e non ha peso, non è un oggetto fisico quindi non può essere né d'intralcio né fonte di germi o batteri eludendo alla necessità di essere sterilizzato per entrare in sala operatoria. L'unione di tutte queste caratteristiche implementate in un visore comodo e agevole come Hololens 2 è ciò che rende la scelta della realtà mista tra le migliori in ambito sanitario e chirurgico.

3.1 Requisiti e motivazioni del progetto

In accordo con la letteratura 2.3 e validate da esperti del dominio, sono state definite alcune funzionalità utili in sala operatoria che dovrebbero essere presenti nel **workbench virtuale**, di seguito spiegati i requisiti che sono emersi correlati dall'importanza con cui contribuiscono al sistema:

- **Parametri vitali**: leggere in modo agevole e in tempo reale i parametri vitali del paziente, come sottolineato dall'articolo 2.3.4 questo semplice gesto può essere fonte di perdita di tempo e affaticamento degli occhi nel continuo cambio di messa a fuoco, questo senza considerare che il chirurgo potrebbe non avere nemmeno sott'occhio tutti i monitor ed essere costretto a chiedere ad un assistente riferirgli i valori desiderati, in un momento d'emergenza questo scambio di messaggi potrebbe rivelarsi fatale.

- **Immagini 2D e 3D**: prendere visione di immagini è possibile anche in formato cartaceo, per i modelli tridimensionali si possono usare stampanti 3D, sistemi efficienti finché la quantità di questi elementi non supera le 2 o 3 immagini, limite che attraverso la realtà mista può essere superato, inoltre rendendo possibile la visualizzazione di modelli generati in tempo reale, risparmiando anche su carta o plastica per effettuare le stampe.

- **Video chiamate**: effettuare video chiamate facendo prendere visione dell'intervento ad un medico esterno in tempo reale rende possibile richiedere consulti di qualità come se il consulente si trovasse all'interno della sala operatoria anche quando questo non è possibile, evitando perdite di tempo derivanti dal raggiungimento della sala operatoria da parte di una persona esterna ed eludendo alla necessità di utilizzare un telefono.

- **Appunti o note**: la raccolta di appunti o note registrati temporalmente e cronologicamente su eventi particolari come un'incisione o un'iniezione offrono al chirurgo uno storico di quello che è successo sino a quel momento durante l'intervento, ciò può essere utile per ricontrollare i passaggi effettuati o per tenere conto di quanto tempo tempo è passato da una determinata azione, qui possono venir registrati anche eventi catturati in automatico dal sistema come il crollo della pressione del paziente o l'aumento imprevisto dei battiti cardiaci. Un registro eventi realizzabile anche attraverso carte e penna, ma di certo migliore sfruttando meccanismi di riconoscimento vocale o automatismi del sistema.

L'importanza di un sistema di questo tipo è in parte rappresentata dagli elementi che lo compongono, ogni requisito è studiato per agevolare il chirurgo durante l'intervento offrendogli un'interfaccia *hands-free* con la quale può

comunicare senza essere costretto ad impugnare alcun tipo di strumentazione particolare e può prendere visione di informazioni senza che queste siano fisicamente d'intralcio per l'intervento. L'altra parte che ha motivato l'ideazione e realizzazione di questo progetto è il contributo che può dare non solo in termini di sistema generico utilizzabile dai chirurghi per ogni intervento, ma anche come base aperta ad essere ampliata dall'integrazione di funzionalità specifiche che la raffinano per supportare particolari interventi, questa caratteristica permette a futuri sviluppatori in quest'ambito di concentrarsi sull'implementazione degli strumenti strettamente necessari ad uno specifico intervento senza doversi preoccupare di implementare da capo ogni funzionalità utile in sala operatoria.

3.2 Analisi e modello del dominio

Dopo essersi posizionato per l'intervento e aver indossato il visore Hololens 2 il chirurgo può avviare l'applicazione preventivamente installata sul dispositivo, il primo passo necessario è scegliere la sala operatoria in cui ci si trova, a questo punto il sistema comunicando con il *backend* si occuperà di caricare tutti gli elementi necessari all'operazione, da questo momento il chirurgo può iniziare ad operare senza obbligo alcuno di interagire con il sistema, all'occorrenza può però consultare immagini precedentemente disposte per l'intervento o modelli tridimensionali. Informazioni importanti come i parametri vitali del paziente sono sempre presenti dove preferisce il chirurgo aggiornati in tempo reale per una lettura comoda e agevole. In qualsiasi momento è sempre possibile prendere nota di quello che si sta facendo, il sistema si occuperà di memorizzare questi appunti e metterli a disposizione per essere riletti affiancati dal loro *timestamp*. Nell'ipotesi della necessità di un consulto esterno il chirurgo può chiamare un altro medico esterno alla sala operatoria, questo dopo aver risposto avrà visione dell'intervento e potrà supportare il chirurgo in difficoltà. A fine intervento il chirurgo deve chiudere l'applicazione e togliersi il visore.

3.2.1 Casi d'uso

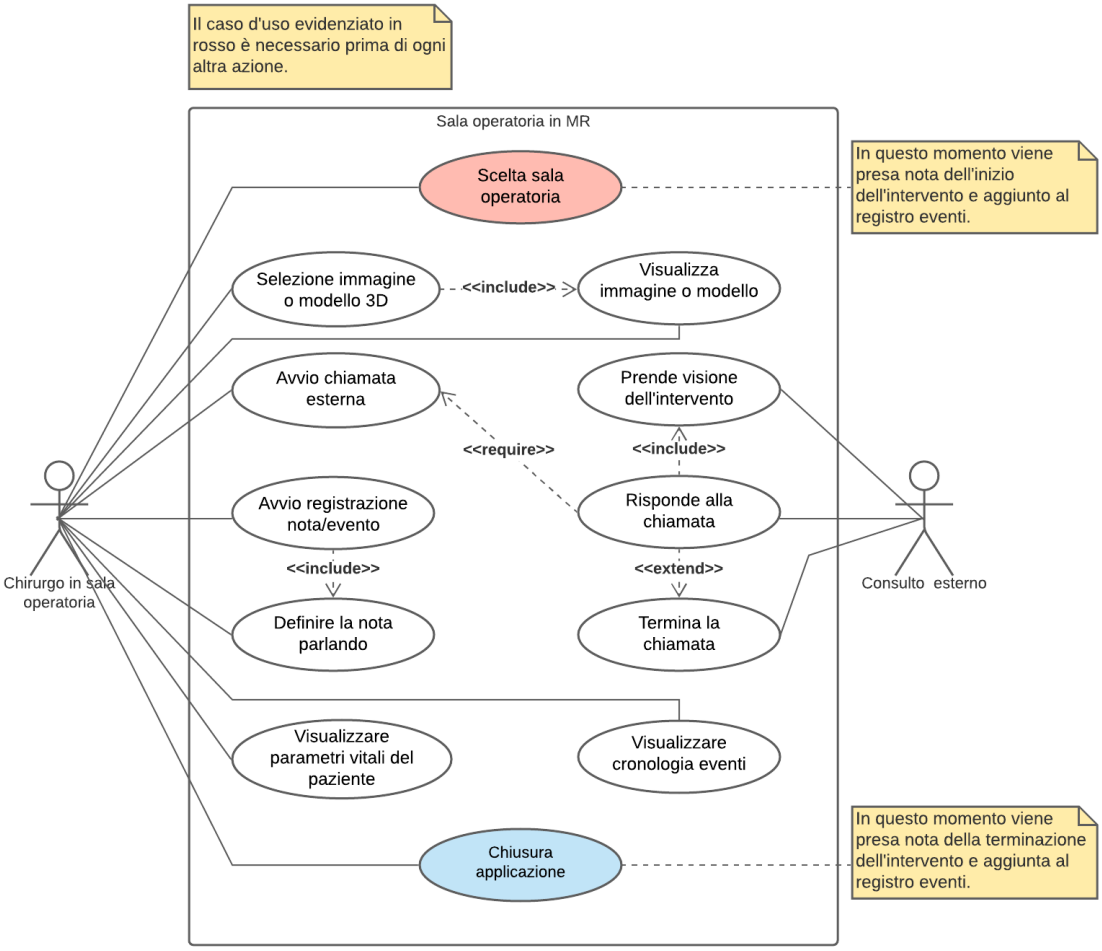

Figura 3.1: Casi d'uso del sistema

Per implementare i requisiti 3.1 secondo i casi d'uso 3.1 è stato deciso di separarle in diversi pannelli che li raggruppano per funzionalità secondo quanto descritto di seguito:

- **Archivio immagini 2D**: si occupa di contenere la *preview* di tutte le immagini 2D a disposizione dell'intervento, interagendo con una di queste essa viene generata più grande di fronte al chirurgo, sarà quindi possibile spostarla o scalarla in base all'esigenza, in tutti i casi l'immagine aggiornerà la sua rotazione per essere sempre visibile nel migliore dei modi con la facciata frontale rivolta verso il chirurgo. Nel momento

in cui questa non dovesse più servire è possibile rimuoverla dalla sala operatoria con un bottone rappresentante una 'X' sempre presente sopra l'angolo in alto a destra dell'immagine stessa.

- **Archivio immagini 3D**: molto simile all'*archivio immagini 2D* contiene rappresentazioni tridimensionali di vario tipo, dall'arto o organo interno del paziente ricostruito da scansioni effettuate da altre macchine a "semplici" modelli generici illustrativi. L'interazione con questo pannello è simile a quella vista precedentemente con l'unica differenza che la rotazione non sarà impostata automaticamente lasciando al chirurgo la libertà di ruotarlo e osservarlo da qualsiasi punto di vista.

- **Registratore appunti/eventi**: un pannello contenente in ordine cronologico quello che è successo in sala operatoria, ogni evento verrà visualizzato affiancato dal suo *timestamp* per sapere con precisione quando è avvenuto e da un'etichetta che ne contraddistingue la categoria (incisione, iniezione, ecc.), è quindi possibile scorrere lungo tutti gli eventi per recuperare informazioni su quanto avvenuto. Oltre ad essere visualizzato sul pannello ogni evento è in tempo reale condiviso per essere memorizzato in un *log* e a disposizione di possibili ulteriori monitor (ad esempio per gli spettatori al di fuori della sala operatoria). Per memorizzare un evento basterà interagire con il bottone *START* sempre presente a lato del pannello per mettere in ascolto il sistema, a questo punto basterà parlare e in automatico verranno comprese le parole attraverso un sistema di *speech to text* (STT), finito di registrare l'appunto non sarà necessaria alcun'altra azione infatti il sistema capirà che non si sta più parlando e analizzerà il testo calcolato cercando al suo interno la presenza di parole chiave per assegnare l'evento a una determinata categoria, se nessuna delle parole chiave dovesse essere trovata la categoria selezionata sarà quella di semplice appunto.

- **Tele-consulto**: grazie a questo pannello è possibile effettuare video chiamate in tempo reale con altri medici, chi risponde non per forza deve indossare un visore Hololens 2 ma può comunque vedere quello che vede il chirurgo durante l'operazione attraverso la videocamera già presente nel visore che indossa, tale funzionalità risulta comoda nel momento in cui si ha bisogno di un consulto esterno da parte di un altro dottore, anziché fargli raggiungere fisicamente la sala operatoria gli basterà rispondere alla chiamata e prendere visione dell'operazione, eliminando in questo modo tutta una serie di problemi logistici e perdite di tempo, ma comunque mantenendo elevata la qualità dell'opinione come se fosse stato in presenza. Per interagire con questo pannello ed effettuare una

chiamata si hanno a disposizione pulsanti rapidi per mettersi in comunicazione direttamente con il reparto di competenza di cui si ha bisogno, se ad esempio fosse necessario richiedere un parare da neurologia basterebbe premere il bottone ad esso associato e aspettare una risposta, una volta conclusa la discussione chi ha risposto ha l'incarico di chiudere la chiamata per lasciare libero il chirurgo di effettuare altre chiamate nel caso dovesse averne bisogno.

- **Visualizzatore parametri vitali**: in questo pannello sono raggruppati i parametri vitali del paziente, in particolare per ognuno di essi è presente il grafico dell'andamento nel tempo affiancato dal valore assunto in quell'istante, data la sua esigenza di essere visualizzato velocemente e di non dover essere d'intralcio è possibile spostarlo e ridimensionarlo a piacere, inoltre in questo caso la rotazione non è fissata verso il chirurgo per evitare che muovendo la testa l'elemento vada a ruotarsi rischiando di intralciare la vista sul paziente. Per evitare fraintendimenti e per adattarsi agli standard ogni parametro è visualizzato affiancato dalla sua unità di misura e del colore specifico che lo rappresenta, in particolare: Frequenza Cardiaca - Verde [bpm/min], Saturazione Ossigeno nel sangue (SpO2) - Bianco/Grigio [%], End-Tidal CO2 (EtCO2) - Giallo [mm(hg)], Pressione Arteriosa (SYS – DIA) - Sistolica: Rossa [mm(hg)] - Diastolica: Arancio [mm(hg)] e Temperatura (Temp) - Azzurro [°C]. Nel momento in cui un parametro esce dai limiti normali in cui dovrebbe trovarsi un'icona di allerta viene visualizzata al suo fianco per segnalarne l'anomalia.

- **Bottone chiusura intervento**: questo è il bottone predisposto alla chiusura dell'intervento, esso automaticamente scrive la nota di fine intervento negli appunti e chiude l'applicazione lasciando libero il chirurgo di rimuovere il visore, data la sua intrinseca importanza un messaggio di conferma di voler davvero terminare l'applicazione viene mostrato prima di uscire, per evitare di essere premuto per sbaglio o che possa essere d'intralcio è posizionato alle spalle del chirurgo, in questo modo l'unico modo col quale può interagirci impone che si debba voltare di 180°.

3.2.2 Scenari

Scelta sala operatoria

Scenario base	1. Il chirurgo si posiziona e indossa il visore. 2. Il chirurgo avvia l'applicazione. 3. Il chirurgo sceglie la sala operatoria.
Varianti	• Se il sistema non riesce a comunicare con il backend viene visualizzato un messaggio d'errore.

Nel caso in cui non si riesca a comunicare con il backend è necessario assicurarsi la connessione Wi-fi sia attiva e riavviare l'applicazione.

Visualizzare immagine bidimensionale o modello tridimensionale

Prerequisiti	• La sala operatoria deve essere stata scelta.
Scenario base	1. Il chirurgo seleziona un'immagine o modello da visualizzare. 2. Il chirurgo sposta e scala a piacimento l'oggetto. 3. Quando il chirurgo non ha più bisogno dell'oggetto lo elimina.
Varianti	• È possibile visualizzare più oggetti contemporaneamente selezionandoli uno dopo l'altro. • Un oggetto eliminato può sempre essere recuperato dall'archivio.

Oltre a visualizzare più oggetti contemporaneamente è inoltre possibile creare lo stesso oggetto più volte, nel caso di modelli tridimensionali questa scelta dona al chirurgo la possibilità non solo di osservarne la fisionomia da diversi punti di vista ma anche di prenderne visione da più prospettive simultaneamente.

Registrare una nota o evento

Prerequisiti	• La sala operatoria deve essere stata scelta.
Scenario base	1. Il chirurgo avvia la registrazione della voce. 2. Il chirurgo parla normalmente esponendo i dettagli della nota. 3. La nota viene automaticamente aggiunta al registro.
Varianti	• Se il chirurgo avvia la registrazione e non parla dopo qualche secondo verrà automaticamente interrotta e nulla verrà registrato.

Avviare una chiamata

Prerequisiti	• La sala operatoria deve essere stata scelta.
Scenario base	1. Il chirurgo sceglie a quale reparto inoltrare la chiamata. 2. Il chirurgo chiama il reparto tramite apposito bottone. 3. Il medico esterno risponde alla chiamata. 4. Il medico esterno ha visione sulla sala operatoria e comunica con il chirurgo. 5. A fine conversazione il medico esterno chiude la chiamata.
Varianti	• Se il chirurgo dopo aver avviato la chiamata non dovesse ricevere risposta può decidere di terminarla anticipatamente.

Chiudere la sala operatoria

Prerequisiti	• La sala operatoria deve essere stata scelta.
Scenario base	1. Il chirurgo si volta verso il bottone d'uscita. 2. Il chirurgo preme il bottone e conferma di voler chiudere la sala operatoria. 3. Il chirurgo si toglie il visore dalla testa.
Varianti	• Se dopo aver premuto il bottone il chirurgo non conferma di voler chiudere la sala è possibile continuare ad operare come se non lo si fosse premuto.

Per scongiurarne la pressione involontaria e considerata la non necessità di accederci agilmente il bottone è posto esattamente dietro al chirurgo, così facendo il chirurgo è costretto a voltarsi per poterci interagire.

3.3 Architettura del sistema

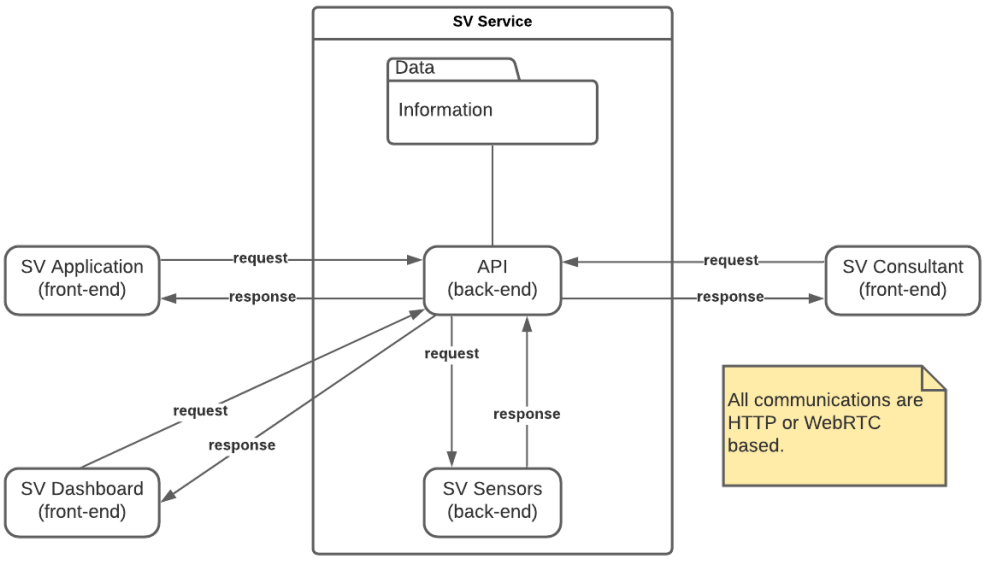

Figura 3.2: Modello architetturale

Per semplicità ci riferiamo nei modelli al **workbench virtuale** col suo acronimo **SV**, di seguito spiegate le principali entità del sistema:

- **SV Application**: applicazione per Hololens 2 utilizzata dal chirurgo in sala operatoria per fruire dei vantaggi offerti dal sistema. Comunica con *SV Service* tramite richieste HTTP oppure WebRTC per instaurare video chiamate.

- **SV Service**: si occupa di mantenere salvate le informazioni utili ai vari interventi e di metterli a disposizione comodamente tramite API accessibile per via HTTP, inoltre gestisce le chiamate WebRTC tra chirurgo e consulente e comunica con l'interfaccia *SV Sensors* per la raccolta dei dati in tempo reali dai macchinari presenti in sala operatoria.

- **SV Consultant**: applicativo che usa il consulente esterno per prendere visione dell'intervento in tempo reale e comunicare la propria opinione al chirurgo che ne ha richiesto le competenze.

- **SV Sensors**: interfaccia tra *SV Service* e le macchine presenti in sala operatoria da cui si ricavano i dati in tempo reali riguardanti i parametri vitali del paziente.

- **SV Dashboard**: applicativo Web per la gestione del *backend*, da qui è possibile visualizzare e modificare le informazioni relative agli interventi, aggiungere o rimuovere immagini e modelli tridimensionali dagli appunti e visualizzare lo storico degli interventi.

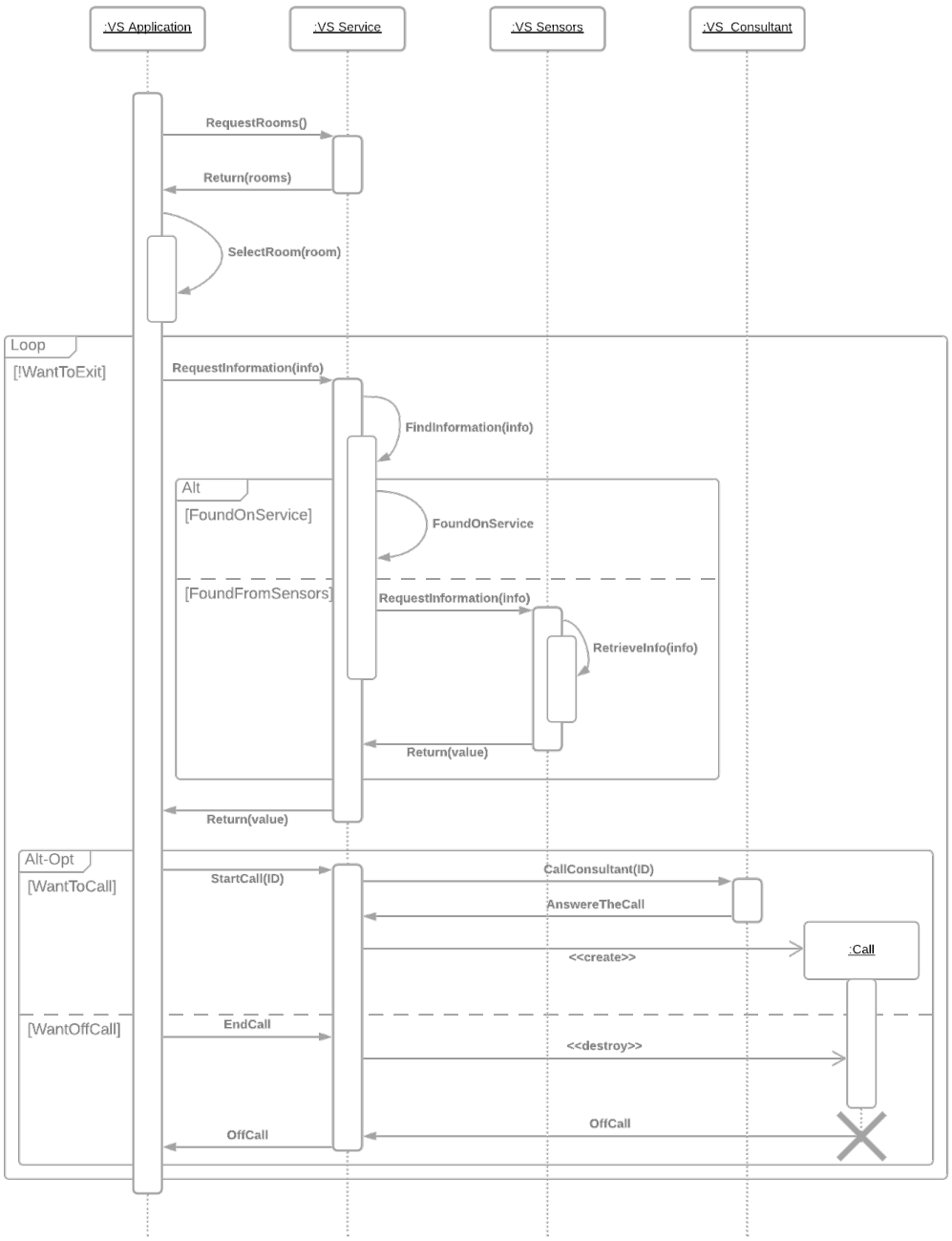

Figura 3.3: Diagramma di sequenza, avvio del sistema

Come si nota dal diagramma di sequenza del sistema 3.3 l'applicazione su Hololens per funzionare ha necessariamente bisogno di una connessione funzionante con il *backend*, questo potrebbe sembrare una caratteristica a sfavore del

sistema realizzato, in realtà questo permette di demandare tutte le operazioni che non riguardano la rappresentazione dei dati in realtà mista al *backend* lasciando così più potenza computazionale a disposizione del *frontend* per rappresentare al meglio l'interfaccia grafica del chirurgo, inoltre nel caso in cui la batteria del visore dovesse scaricarsi e ci fosse bisogno di cambiarlo, questa separazione dei lavori consente di rimpiazzarlo con un altro carico in modo naturale senza perdere alcun dato.

3.3.1 Design dell'API

Di seguito elencati i percorsi di navigazione per la comunicazione tra *frontend* e *backend* descrivendone il significato partendo dal metodo usato secondo il protocollo HTTP:

Metodo	Percorso
GET	/rooms
POST	/rooms/RoomName/Note
GET	/rooms/RoomName/2DImages
GET	/rooms/RoomName/3DImages
GET	/rooms/RoomName/2-3DImages/ImageName
GET	/history
GET	/history/InterventionID

Effettuando una GET sul percorso */rooms* viene restituito l'elenco in formato JSON delle sale operatorie registrate nel sistema, una volta scelta la sala in cui si sta operando è possibile accedere alle immagini utili all'intervento mediante richieste GET ai percorsi */rooms/RoomName/2-3DImages* le quali restituiscono gli elenchi delle immagini e modelli tridimensionali presenti, da questi poi specificando un nome preso dall'elenco è possibile scaricare il file associato. Quando il chirurgo prende un appunto, registra un evento o il sistema stesso prende nota di una particolare situazione è necessario fare una POST con al suo interno il contenuto dell'appunto in questo modo il *backend* è sempre aggiornato, quando viene presa nota della fine dell'intervento esso viene registrato assieme a tutti i suoi dati nella *history*. Non durante un intervento ma lato *SV Dashboard* è possibile visualizzare l'elenco di tutti gli interventi

fatti tramite GET al percorso */history*, specificando poi l'identificativo di un intervento si accede al suo contenuto. Per evitare inutili ripetizioni ho evitato di specificare anche i metodi POST e DELETE alle chiamate GET in */rooms**, il *backend* risponde a queste chiamate eliminando l'elemento selezionato (sala operatoria, immagine 2D o 3D) in caso della DELETE oppure aggiungendo l'elemento passato nel caso di una POST, in questo modo lato *SV Dashboard* è possibile interagire con il *backend* per modificare, aggiungere e rimuovere le informazioni predisposte all'intervento o addirittura le sale operatorie stesse.

3.4 Tecnologie impiegate

Per la realizzazione del sistema sono state impiegate diverse tecnologie, Unity e MRFT per lo sviluppo dell'applicazione che gira direttamente su Hololens 2, mentre python e WebRTC per l'implementazione dei servizi di *backend* che si interfacciano con i vari macchinari in sala operatoria e rendono possibile la comunicazione in video chiamata tra chirurgo e medico consulente.

3.4.1 Unity 3D

Unity 3D è un *game engine* che permette lo sviluppo di giochi in un ambiente tridimensionale, ci riferiamo ad ogni elemento contenuto in una scena con *gameobject*, essi contengono sia *script* messi a disposizione da Unity per il salvataggio di informazioni generiche come il posizionamento nel mondo tramite coordinate o per l'implementazione di comportamenti tipici come cadere per effetto della gravità e la gestione delle collisioni con altri *gameobject* sia *script* scritti in C# direttamente dal programmatore per la realizzazione di comportamenti specifici per l'applicazione che si sta sviluppando. I vantaggi derivanti dalla semplicità di sviluppo in C# e dalla sua natura orientata agli oggetti uniti alle potenzialità di Unity in quanto motore tridimensionale è ciò che rende questa accoppiata un'ottima metodologia per lo sviluppo di applicazioni in realtà mista, ogni *gameobject* viene visualizzato tramite Hololens 2 come ologramma, il sistema di coordinate valutato in metri permettete di realizzare applicazioni in scala mondiale in modo semplice e intuitivo azzerando il gap tra sviluppo e risultato finale. Dentro l'*editor* di Unity senza bisogno di possedere alcun hardware specifico è possibile simulare l'uso dell'applicazione come se stessimo interagendo attraverso Hololens 2 direttamente dalla piattaforma su cui stiamo sviluppando, questo riduce drasticamente i tempi di debugging e permette di sviluppare in realtà mista anche a programmatori che non possiedono il visore necessario per testare l'applicazione, la differenza tra il testing nell'editor e su Hololens 2 è ulteriormente assottogliata dalla possibilità di

caricare le mappe generate dal visore nell'editor per simulare anche lo *spatial awareness*. Ovviamente il *testing* su dispositivo è fondamentale per assicurarsi che ogni ogni ologramma e funzionalità implementata funzioni correttamente, a tal proposito è ovviamente possibile installare l'applicazione sul dispositivo e validarla come fossimo un utente, ma è inoltre possibile avviare il *debugging* da remoto tramite Wi-Fi per poter leggere i log dell'applicazione in tempo reale e assicurarsi che tutto funzioni secondo quanto previsto, in caso contrario è possibile prendere visione delle eccezioni lanciate dall'applicativo e nel caso scovare eventuali bug che il simulatore da *editor* non aveva evidenziato.

3.4.2 MRFT e MRTK

Il *Mixed Reality Feature Tool* (MRFT) è lo strumento col quale vengono scoperti, installati e aggiornati i componenti della realtà mista all'interno di un progetto preesistente di Unity, con esso è possibile cercare i pacchetti da installare per nome o per categoria, vedere le loro dipendenze e persino controllare le modifiche che verranno applicate al *manifest* ancor prima di importarli, una volta averli validati il MRFT si occuperà di scaricarli all'interno del progetto scelto inizialmente. Uno dei componenti installati è il *Mixed Reality ToolKit* un *kit* per sviluppatori *open-source* e *cross-platform* per lo sviluppo di applicazioni in realtà mista, esso contiene le interazioni base di *input* al sistema e le fondamenta per la realizzazione di interazioni spaziali, l'obiettivo di questo pacchetto è quello di velocizzare lo sviluppo di applicazioni in MR riducendo l'effetto di *reinventing the wheel* mettendo a disposizione alcune componenti fondamentali. Analizziamo ora alcuni vantaggi di usare il MRTK:

- **Modular**: per sfruttarne le potenzialità non è necessario scaricarlo per intero, ma solo quelle funzionalità che si vogliono sfruttare all'interno dell'applicazione, questo permette di mantenere leggero il progetto che si sta sviluppando e una gestione più intuitiva dei vari pacchetti. È inoltre possibile modificarne il contenuto per adattare i componenti al sistema che si sta realizzando.

- **Cross-platform**: questo non significa che ogni singola piattaforma è supportata, ma assicura che cambiando piattaforma target di sviluppo nessuna porzione di codice del *kit* dia problemi, tale robustezza ed elasticità permette di sviluppare su diverse piattaforme come ARCore, ARKit, e OpenVR.

- **Performant**: lo sviluppo di questo *kit* è stato incentrato sulle performance, per evitare che l'uso delle sue componenti vada ad impattare negativamente sulle prestazioni del sistema.

Per sviluppare un'applicazione in realtà mista è necessario prima di tutto crearne una nuova in 3D dall'hub di Unity, poi con il MRFT installare le componenti che si ritengono necessarie (tra cui il MRTK), a questo punto è necessario scegliere quale plugin XR utilizzare, *Legacy Built-in XR* per Unity 2019.4 raccomandato in quanto versione stabile del plugin, *Windows XR* con Unity 2020.3 per una versione più aggiornata ma dichiarata avere problemi di stabilità degli ologrammi ed infine *Mixed Reality OpenXR* sempre per Unity 2020.3 conforme agli standard di *OpenXR* per la realizzazione di applicazioni che possono essere eseguite da tutti i dispositivi che lo supportano, per ovvi motivi questa scelta è quella su cui ci si vuole uniformare anche se ad oggi presenta ancora qualche instabilità.

3.4.3 Python e http.server

Python è un linguaggio interpretato ad alto livello *general-purpose* studiato per essere leggibile grazie anche all'uso significativo dell'indentazione, viene definito come un linguaggio "*batteries included*" per la sua ricca *standard library*, per quasi tutte le altre funzionalità lì non presenti esistono quanto meno librerie scaricabili e facilmente installabili (anche attraverso utili strumenti come pip o anaconda) che ne completano il funzionamento, queste caratteristiche unite rendono la scelta di questo linguaggio ottima per lo sviluppo veloce di semplici *backend* che non necessitino di onerose e particolarmente esigenti prestazioni computazionali. In particolare sono interessanti le implementazioni di web server http messe a disposizione, quelle base che troviamo sono tre:

- **BaseHTTPRequestHandler**: questa implementazione è la più basilare di tutte, senza ulteriori sviluppi è solamente in grado di gestire le richieste che riceve dai client ma non di rispondergli, questo *handler* non fa altro che tradurre le richieste e gli *headers* che gli arrivano per poi richiamare il giusto metodo in base al tipo di richiesta effettuata, il nome con cui viene invocato un metodo è semplicemente composto dal prefisso *do_* e il nome della *request*, ad esempio per una richiesta GET il nome del metodo chiamato risulta: do_GET().

- **SimpleHTTPRequestHandler**: funziona esattamente come l'implementazione di base, ma inoltre implementa i metodi do_GET() e do_HEAD() per rispondere a particolari richieste da parte del client relative al *file system*, cioè mette a disposizione la possibilità di navigare nell'albero di cartelle in cui è situato il server e di leggere o scaricare tutti i file presenti al suo interno. È inoltre possibile modificare il percorso dal quale il server ricava la porzione di *file system* attraverso il parametro "*directory*",

nel caso volessimo esplorare il contenuto di una cartella diversa da quella in cui si trova il server stesso.

- **CGIHTTPRequestHandler**: anche questa implementazione elabora le richieste come la versione base e mette a disposizione il *file system* come appena visto, ciò che lo differenzia risiede nella gestione del contenuto delle cartelle "*/cgi-bin*" e "*/htbin*", questo infatti non è semplicemente restituito come file, si presuppone che al loro interno siano contenuti *script Common Gateway Interface* (CGI), essi vengono invocati e all'utente viene restituito il risultato prodotto dalla loro esecuzione. In questo caso anche il metodo do_POST è implementato, ma solo nel caso le richieste facciano riferimento alle cartelle di CGI, altrimenti viene restituito un errore 501 "*Can only POST to CGI scripts*".

Per ognuna di queste implementazioni è possibile sovrascrivere o estendere i metodi invocati dall'*handler* per modificare o implementare come deve rispondere il web server alle richieste del client, questo è vero soprattutto per la prima versione analizzata che altrimenti sarebbe totalmente inutile e priva di significato.

3.4.4 WebRTC e node-dss

WebRTC è un protocollo basato su uno standard aperto che permette la condivisione di voce, video e altri messaggi in tempo reale per dare la possibilità agli sviluppatori di integrare nella propria applicazione funzionalità di messaggistica o video chiamata, nativamente nasce per essere utilizzato da pagine web, infatti le interazioni con l'API sono realizzate tramite Javascript, ma non sono stati esclusi sistemi nativi come Android o iOS per i quali esistono librerie sviluppate appositamente. Le potenzialità di questo protocollo sono evidenti, abbastanza da trovarne un'implementazione *ad hoc* all'interno del MRFT per lo sviluppo con Unity, infatti per integrare la nostra applicazione con questo componente è sufficiente installarlo tramite il *tool* e si avranno a disposizione tutti gli script necessari per realizzare una vera comunicazione video e vocale in tempo reale, il tutto supportato da un server che fa da intermediario per rendere la comunicazione possibile. In questo caso la guida di Microsoft consiglia l'utilizzo di un server *node-dss* del quale possiamo trovare un'implementazione su Github[1], esso realizza uno scambio di messaggi semplificato in un ambiente dove entrambi i client conoscono l'uno l'identità dell'altro, in questo felice scenario possiamo risparmiarci di implementare una complessa logica di controllo dei client "vivi" e la comunicazione degli eventi di *join* e *leave* dei

[1] `https://github.com/bengreenier/node-dss`

pari, inoltre per semplificare ulteriormente il server di *signalling* la consegna dei messaggi al client destinatario viene effettuata solamente quando esso lo richiede, costringendo in questo modo però la comunicazione a mettere in atto un sistema di *polling* con tutti i vantaggi e svantaggi che esso comporta. In accordo con la documentazione ufficiale le componenti di WebRTC messe a disposizione dal MRFT funzionano solamente se l'applicazione che si sta sviluppando viene esportata in Hololens 2 basandosi su un'architettura ARM e non ARM64, di per sé questo non sarebbe un problema infatti il processore di Hololens 2 a 64 bit può tranquillamente eseguire applicazioni sviluppate a 32 bit, se non fosse per un bug conosciuto di Unity 2020 che impedisce il *rendering* delle applicazioni sviluppate per architettura ARM una volta caricate su Hololens 2, questo sfortunatamente ha costretto l'uso di una versione più vecchia di Unity per riuscire a implementare il progetto di questa tesi senza ulteriori perdite di tempo.

3.5 Sviluppo applicazione per Hololens 2

Per risolvere il problema descritto alla fine del paragrafo 3.4.4 si è deciso di implementare il progetto utilizzando la versione di Unity 2019.4 che non presenta lo stesso bug riscontrato con la versione più aggiornata, questo però si porta dietro di conseguenza il dover abbandonare la possibilità di sviluppare usando il *plugin OpenXR*, una grossa perdita a livello di compatibilità ma non troppo importante per il prototipo che è stato sviluppato, in generale non appena questo bug dovesse essere risolto sarà possibile riportare il sistema su Unity 2020 e uniformarlo allo standard in evoluzione. Per evitare ulteriori problematiche ho deciso di usare come *plugin XR* il più stabile cioè il *legacy* con le impostazioni predefinite del MRTK, per il *deploy* del progetto da Unity è necessario scegliere come piattaforma destinazione *Universal Windows Platform* (UWP), come dispositivo *target* Hololens e infine come architettura ARM, in questo modo viene generata una soluzione di Visual Studio con la quale poter compilare i sorgenti per generare semplicemente l'applicativo finale oppure per avviare il *debugging* in uno dei tanti modi messi a disposizione dallo strumento.

3.5.1 Implementazione pattern producer–consumer

Nel particolare di questa applicazione abbiamo la necessità di scaricare e visualizzare informazioni provenienti dal *backend*, per evitare di bloccare l'aggiornamento degli ologrammi durante il *download* tale operazione deve necessariamente essere svolta da un *thread* asincrono mentre per poter aggiungere nuovi elementi grafici alla scena è necessario farlo dal *thread* principale sincrono,

è facile a questo punto astrarre queste due operazioni in processo produttore (asincrono) e processo consumatore (sincrono) sfruttando un'implementazione del problema *producer-consumer* per realizzare il comportamento appena descritto. Di seguito un'implementazione del problema basata sull'architettura *Super-loop* degli *script* di Unity:

```
private Queue<Action> actionQ;
private GameObject locker;

void Start()
{
   actionQ = new Queue<Action>();
    locker = new GameObject();

    StartCoroutine(DownloadData());
}

IEnumerator DownloadData()
{
   // Assegnazione path e metodo da usare.
    var uwr = new UnityWebRequest("{APIdataPath}",
        UnityWebRequest.kHttpVerbGET);
    // Handler che gestisce i risultati del download.
    uwr.downloadHandler = new DownloadHandlerFile(imagePath);
    // Invio richiesta al backend.
    yield return uwr.SendWebRequest();

   // Risultati ricevuti correttamente.
    if (uwr.isDone)
    {
        lock (locker)
        {
         // Aggiungo alla coda delle azioni il lavoro sincrono per
             questi dati.
            actionQ.Enqueue({azione});
        }
    }
}

void Update()
{
    lock (locker)
    {
      // Se la coda contiene almeno un'azione.
```

```
        if (actionQ.Count > 0)
        {
         // Eseguo l'azione sincronamente nel thread principale e la
             rimuovo dalla coda.
            actionQ.Dequeue().Invoke();
        }
    }
}
```

Listato 3.1: Download.cs

Nel codice il consumatore, cioè il *thread* principale che si occupa di visualizzare le informazioni, viene realizzato usando il metodo base *Update* presente in ogni *script* che implementa il modello *Super-loop* di Unity da *Monobehaviour*, mentre il produttore asincrono, cioè il *thread* che si occupa di scaricare le informazioni dal *backend*, viene realizzato invocando il metodo *StartCoroutine* che si occupa di gestire la creazione e distruzione di operazioni asincrone. Grazie al sistema di *lock* realizzato è possibile lanciare contemporaneamente anche più *coroutine* senza preoccuparci di finire in *race condition*.

3.5.2 Implementazione WebRTC

Una volta installati i componenti di WebRTC tramite MRFT saranno a disposizione gli *script* per la realizzazione di comunicazioni in tempo reale sia video che vocale o semplicemente scritta tramite messaggi, di seguito un semplice riassunto tratto dalla guida ufficiale[2] per l'implementazione di una semplice video chiamata:

1. **Creazione PeerConnection**: creiamo un nuovo *gameobject* vuoto, aggiungiamo al suo interno lo script "MixedReality-WebRTC - PeerConnection", a questo punto le proprietà di questo componente non hanno bisogno di essere modificate, così facendo abbiamo creato l'oggetto che fa da interfaccia tra la comunicazione e le risorse locali.

2. **Creazione Signaler**: creiamo un ulteriore *gameobject* vuoto e gli aggiungiamo il componente "MixedReality-WebRTC - NodeDssSignaler", importante impostare correttamente l'indirizzo del server *node-dss* come *default* si presuppone che esso sia in esecuzione localmente alla macchina, successivamente è necessario definire il tempo di *polling* per il controllo della presenza di messaggi sul server, il valore di default è di 500 millisecondi.

[2] `https://microsoft.github.io/MixedReality-WebRTC/versions/release/1.0/manual/helloworld-unity.html`

3. **Connessione Signaler**: nelle proprietà della *PeerConnection* è ora possibile impostare a *Signaling* in *Signaler* il *gameobject Signaler* appena creato.

4. **Aggiunta video locale**: per attivare la camera locale creiamo un altro *gameobject* e ci aggiungiamo il componente "MixedReality-WebRTC - LocalVideoSource", esso si occupa di ottenere il video dalla camera principale e di spedirlo alla *PeerConnection* impostata sotto *Video Track*, se la prorietà *Enable mixed reality capture* è abilitata nel video inviato saranno presenti anche gli ologrammi.

5. **Aggiunta video remoto**: per ricevere il video da remoto è sufficiente creare un *gameobject* con all'interno un "MixedReality-WebRTC - RemoteVideoSource" e connetterlo alla *PeerConnection* come fatto per il video locale, per mostrare il video remoto ricevuto è necessario aggiungere altre tre componenti, una *Mesh filter* con la *mesh* che desideriamo (consigliata *quad*), un *Mesh renderer* con materiale "/Materials/YUVFeedMaterial" e un "MixedReality-WebRTC - MediaPlayer" con fonte di video impostata a quella presente nello stesso *gameobject*, se fosse necessario visualizzare anche il video locale è sufficiente aggiungere gli stessi elementi al *gameobject* con la fonte del video locale e impostarli come appena descritto.

6. **Realizzazione connessione**: una volta arrivati a questo punto è necessario decidere un ID col quale verremo identificati dal server nella proprietà del *Signaler: Local Peer Id*, per decidere chi chiamare è necessario impostare anche il *Remote Peer Id* con l'ID di chi stiamo chiamando, a questo punto è possibile stabilire una connessione invocando il metodo "PeerConnection.CreateOffer()".

Da notare che in questo modo solo il video viene condiviso e visualizzato, nel caso volessimo invece anche l'audio allora è necessario creare altri due *gameobject* simili a quelli per il video ma usando *LocalMicrophoneSource* e *RemoteMicrophoneSource* al posto delle fonti di video e *AudioSource* al posto dei *renderer* dei video, ovviamente evitando di riprodurre l'audio locale che molto spesso non è il comportamento che uno si aspetta.

3.5.3 Interfaccia utente

All'avvio dell'applicazione su Hololens 2 viene mostrato all'utente il menù 3.4 per scegliere la sala operatoria in cui si sta operando.

Figura 3.4: Menù scelta sala operatoria

Una volta scelta la sala operatoria il sistema scarica dal *backend* le immagini salvate per quel determinato intervento e le mostra in pannelli, inoltre carica il blocco note, i parametri vitali e predispone il pannello per effettuare video chiamate. In automatico viene presa nota dell'inizio dell'intervento come mostrato in figura 3.5, inoltre dal pannello dei parametri vitali possiamo vedere il segnale d'emergenza posto a fianco dei valori fuori dalla norma.

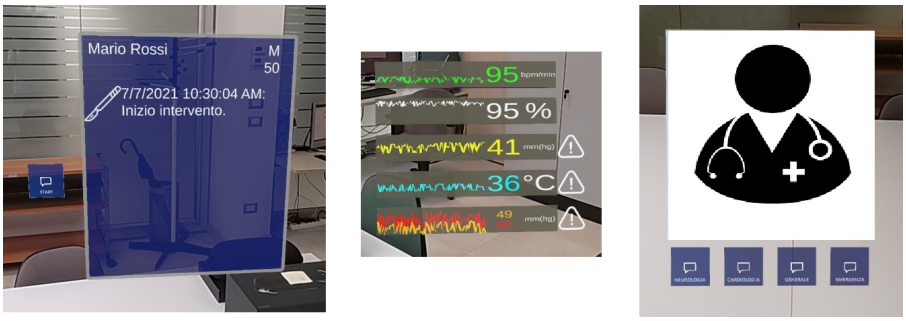

Figura 3.5: Funzionalità sala operatoria

Toccando un'immagine o un modello tridimensionale posti all'interno di librerie rispettivamente a sinistra e destra dell'utente viene generata dinanzi a lui lo stesso elemento 3.6 ma con la possibilità di essere traslato e ridimensionato, quando poi non serve più è possibile rimuoverlo attraverso l'apposito bottone di chiusura.

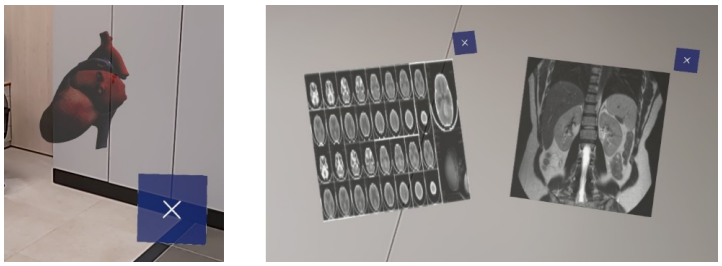

Figura 3.6: Esempio elementi 2D e 3D

3.6 Sviluppo backend

Il *backend* ricopre due ruole importanti, realizzare la comunicazione tra applicativi e mettere a disposizione tramite API i dati raccolti dai macchinari presenti in sala operatoria.

3.6.1 Implementazione server Node-dss

WebRTC nasce come standard aperto per la realizzazione di comunicazioni in tempo reale sul web, Node-dss è un'implementazione di server basata su questo standard scritta in Javascript (JS) e implementabile appunto grazie a Node.js una piattaforma che permette di eseguire codice JS al di fuori di un classico browser, grazie a questa implementazione è possibile crearsi un proprio server locale per WebRTC installando Node.js e avviandolo tramite un semplice comando:

```
npm install
npm start
```

Listato 3.2: Comando per lanciare Node-dss

Avviando in questo modo il terminale che si apre non mostra alcun tipo di *output* ed è difficile in fase di sviluppo capire se tutto sta funzionando correttamente, fortunatamente gli sviluppatori di questa implementazione han messo a disposizione la possibilità di rendere verboso ogni evento semplicemente impostando una variabile d'ambiente DEBUG prim di eseguire in questo modo:

```
set DEBUG=dss*
npm install
npm start
```

Listato 3.3: Comando per lanciare Node-dss

Grazie a questo piccolo accorgimento siamo in grado di vedere quando un client si connette e tutte le volte in cui fa *polling* per controllare se ci sono messaggi, da questi due semplici eventi è possibile ricavare l'ID con il quale chi si è connesso ha deciso di identificarsi, inoltre vengono visualizzati anche tutti i messaggi che i client si scambiano in modo da semplificare la verifica di un corretto flusso o nel caso qualcosa non andasse a buon fine poter leggere la traccia degli errori che si sono verificati.

3.6.2 Implementazione http.server

Esistono tantissimi modi per realizzare uno scambio di informazioni in rete, da quelli di più basso livello che sfruttano *sockets* per creare comunicazioni *ad hoc* all'applicazione che si sta sviluppando fino a quelli di più alto livello che si appoggiano su protocolli del *layer* applicativo come http, tra questi troviamo http.server di Python oppure Vert.x scritto in Java che inglobano il protocollo sottostante fornendo un'API per realizzare *Web server* partendo da una base funzionante. In particolare in questo progetto è presente l'implementazione in Python di *SimpleHTTPRequestHandler* che mette nativamente a disposizione la possibilità di scaricare i file presenti nel server come nel nostro caso le immagini relative all'intervento, per avviarlo basta semplicemente da terminale posizionarsi nella cartella dove si desidera aprirlo ed eseguire questo semplice comando:

```
python -m http.server 8000
```

Listato 3.4: Comando per lanciare http.server

Ovviamente il server avviato in questo modo non permette alcuna personalizzazione del suo funzionamento se non cambiare il numero di porta (8000) o la *directory* da quale prende i file impostando il parametro –directory, nel momento in cui nasce il bisogno di una maggiore flessibilità per realizzare comportamenti aggiuntivi, come in questo caso comunicare con i macchinari in sala operatoria per la raccolta dei dati in tempo reale dai sensori, è necessario estendere l'implementazione base per abilitare il server a rispondere ad altre tipologie di richieste e a farne lui a sua volta ad altri sistemi, di seguito un'implementazione che mostra come sia possibile farlo:

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import logging

port = 8000

class S(BaseHTTPRequestHandler):
```

```
    def _set_response(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def do_GET(self):
        logging.info("GET request,\nPath: %s\nHeaders:\n%s\n",
            str(self.path), str(self.headers))
        self._set_response()
        self.wfile.write("GET request for
            {}".format(self.path).encode('utf-8'))

    def do_POST(self):
        content_length = int(self.headers['Content-Length']) #
            Lunghezza del contenuto POST.
        post_data = self.rfile.read(content_length) # Lettura del
            contenuto nella POST.
        logging.info("POST request,\nPath:
            %s\nHeaders:\n%s\n\nBody:\n%s\n",
                str(self.path), str(self.headers),
                    post_data.decode('utf-8'))

        self._set_response()
        self.wfile.write("POST request for
            {}".format(self.path).encode('utf-8'))

logging.basicConfig(level=logging.INFO)
server_address = ('', port)
httpd = server_class(server_address, S)
logging.info('Starting httpd...\n')
try:
    httpd.serve_forever()
except KeyboardInterrupt:
    pass
httpd.server_close()
logging.info('Stopping httpd...\n')
```

Listato 3.5: HttpExtended.py

Questo esempio mostra una possibile base[3] da cui è possibile partire per capire come estendere le funzionalità di http.server, qui è mostrato una semplice implementazione che non fa altro che ascoltare tutte le richieste GET e

[3] https://gist.github.com/mdonkers/63e115cc0c79b4f6b8b3a6b797e485c7

POST rispondendo con HTML contenente le informazioni della richiesta stessa, partendo da questo file è possibile integrare funzionalità come la lettura dei sensori sul paziente e restituire di seguito risposte significative.

3.7 Interazioni in MR con l'applicazione

Per prima cosa è necessario comprendere che il visore è dotato di fotocamere con le quali è in grado di riconoscere le mani di chi lo indossa tracciandole in uno spazio tridimensionale, questo significa che per compiere qualsiasi azione è obbligatorio "vedere" quello che si sta facendo, non è importante guardarsi direttamente le mani ma è necessario che i sensori ne abbiano libera visione. Hololens 2 è in grado di riconoscere tutte le dita singolarmente, ma nella maggior parte dei casi sono importanti solo pollice ed indice delle rispettive mani infatti se non implementato in altro modo il supporto originale basa tutte le interazioni sulla posizione di queste due. Di seguito elencate le interazioni possibili con il sistema sviluppato in questa tesi:

- **Il tocco**: usato per selezionare immagini o modelli tridimensionali ed interagire con i pulsanti è il più intuitivo e vuole simulare quello che siamo abituati a fare con uno *schermo touch*, all'interno del mondo di Hololens ci ritroviamo infatti elementi simili a quelli che già conosciamo grazie agli *smartphone* e intuitivamente siamo in grado di interagirci, per marcare una checkbox o premere un pulsante è sufficiente "appoggiarci" l'indice sopra e toglierlo, davanti ad una lista o collezione di oggetti in cui è possibile effettuare lo *scrolling* basterà toccare e trascinare come già siamo abituati a fare.

- **La presa**: utilizzata per manipolare tutti gli elementi che si possono spostare, come le immagini generate o i pannelli delle funzionalità utile in generale tutte le volte in cui vogliamo prendere un oggetto in mano, per farlo ci basterà entrare all'interno di esso con la mano (anche parzialmente) e congiungere l'indice con il pollice, in questo modo se l'ologramma in questione è predisposto a rispondere a questa interazione la sua posizione verrà direttamente collegata alla posizione delle due dita, i movimenti di traslazione o rotazione creati dalla mano verranno riflessi sull'oggetto, per interrompere l'interazione è sufficiente separare le due dita. Se avvicinandosi ad un oggetto compare una scatola evidenziata nei bordi intorno ad esso l'interazione appena descritta non funziona più ma dobbiamo far riferimento a lati ed angoli del contorno, interagendo con i vertici infatti è possibile ingrandire o rimpicciolire l'oggetto contenuto oppure attraverso gli spigoli ruotarlo sull'asse parallelo ad esso passante per il centro.

- **Doppia presa**: utilizzato per ridimensionare gli oggetti che possiamo afferrare attraverso l'uso di due prese contemporaneamente, prendendo un oggetto sia con la mano destra che con la sinistra è possibile ruotarlo spostarlo e scalarlo a piacimento, l'oggetto avrà due perni a cui fare riferimento e si adatterà ai movimenti di modo da mantenere i punti iniziali di presa fissi sulle mani (se allontaniamo le mani tra di loro l'oggetto sarà costretto ad allargarsi), questa interazione risulta molto comoda perché ci permette di interagire in tutti i modi possibili con l'oggetto, ma ovviamente è limitata ad un oggetto per volta al contrario della presa singola con la quale possiamo trascinare due oggetti assieme e indipendentemente.

- **Interazione da remoto**: tutte le interazioni che abbiamo appena visto possono essere riprodotte da remoto, cioè senza il bisogno di avvicinarci all'oggetto con cui vogliamo interagire, guardandosi una mano si può notare un indicatore che parte da essa e punta verso l'esterno, indicando un oggetto e interagendo con la *presa* l'oggetto si legherà alle dita e si muoverà di conseguenza secondo quanto appena descritto. È inoltre possibile effettuare un *tocco* da lontano, se ad esempio dovessimo trovarci un bottone distante basterà indicarlo con la mano e interagirci come se avessimo un pulsante sul pollice, premendolo con l'indice e rilasciandolo subito dopo.

- **Menù start**: in ogni momento è sempre possibile accedere al menù di Hololens 2, guardandosi il polso della mano destra si noterà apparire su di esso la classica icona *start* di Windows, premendola con l'indice della mano sinistra si apre di fronte alla persona il menù iniziale di Hololens, qui possiamo decidere di uscire dall'applicazione corrente toccando il pulsante con l'icona di una casetta in basso oppure di aprirne una dal menù rapido o qualsiasi altra dall'elenco di tutte le applicazioni, apribile con il bottone a destra del menù.

Ovviamente in questo elenco non sono presenti tutte le funzionalità e interazioni possibili con Hololens 2, ma solamente quelle di interesse per il progetto di questa tesi, per una panoramica generale si può sempre far riferimento al sito ufficiale[4] oppure al video di presentazione di Hololens 2[5].

[4] `https://docs.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals`

[5] `https://www.youtube.com/watch?v=uIHPPtPBgHk`

3.8 Assunzioni e sicurezza

Per come è stato progettato il sistema proposto è suddiviso in parti, ognuna per poter funzionare deve avere una connessione con quella principale *SV Service*, questo impone che esso sia installato in una rete raggiungibile dalle altre componenti e quindi la necessità di avere una rete su cui comunicare, nel caso di Hololens in sala operatoria è necessaria una connessione *wireless* Wi-Fi alla quale collegarsi. Affinché il riconoscimento vocale presente in Hololens 2 possa funzionare è necessaria una connessione ad internet in quanto la traduzione da segnale sonoro in testo scritto non è fatta in locale dal visore ma usufruisce di un servizio *cloud* apposito, tal fatto da una parte ci alleggerisce il carico computazionale del visore dall'altro invece ci costringe a fornire al sistema una connessione verso l'esterno dell'ospedale, questo significa aprire un'ulteriore potenziale porta d'ingresso ai male intenzionati, che non dovrebbe però destare problemi considerando i già esistenti sistemi di sicurezza presenti negli ospedali in cui questo sistema potrebbe venir installato. Uno dei punti chiave del sistema è la possibilità di leggere i dati in tempo reale dai macchinari presenti in sala operatoria per mostrare i parametri vitali del paziente al chirurgo, *SV Sensors* è la parte che si occupa di rendere questi dati disponibili, assumiamo che i macchinari siano dotati di un'interfaccia API con la quale comunicare per richiedere informazioni e che possano essere collegati alla stessa rete a cui è collegato tutto il sistema.

Capitolo 4

Validazione e sviluppi futuri

Nel seguente capitolo sono descritte le validazioni effettuate sul prototipo sviluppato. Infine tenendo conto delle considerazioni raccolte sono descritti alcuni possibili sviluppi futuri come l'aggiunta di nuove funzionalità o il consolidamento di quelle già presenti.

4.1 Validazione

Il sistema è stato testato per intero sia attraverso controlli empirici cioè usando direttamente l'applicativo finale, sia attraverso meccanismi di controllo automatici sviluppati *ad hoc* per verificare il corretto funzionamento del *back-end*. In particolare il *back-end* è stato progettato per fornire un servizio di qualità come ci si aspetta in un ospedale, ma il prototipo sviluppato per questioni di tempo è stato realizzato usando componenti non *production-ready*, nello specifico le parti che si occupano della comunicazione non fanno uso di sicurezza data dalla crittografia e nel caso del *signaler* per *WebRTC* è stata utilizzata una semplice implementazione fornita dalla documentazione che non offre tutte le funzionalità limitandone il comportamento a quelle basilari anche a scapito della scalabilità. Per quanto riguarda l'applicativo per Hololens 2 vediamo di seguito qualche risultato interessante derivante dalle sperimentazioni empiriche effettuate su di esso per valutarne la validità:

- Sia in caso di forte luminosità (luci non dirette nel fronte del visore) che in caso di ambienti con scarsa visibilità il sistema è in grado di tener traccia di dove si trova mostrando ologrammi fissi e stabili, se nell'ambiente in cui ci troviamo c'è troppa luce gli ologrammi visualizzati appaiono sbiaditi quindi poco visibili.
- Se durante l'uso dovesse saltare la connessione, tralasciando la possibilità di visualizzare e muovere le immagini e i modelli tridimensionali presenti

prima che succedesse, tutti gli altri pannelli smettono di funzionare, questo è dato dal fatto che ognuno di essi è dipendente in un qualche modo dal *back-end* o da internet. Quando la connessione viene ristabilita tutto torna a funzionare correttamente come ci si aspetta.

- Nel caso di ostruzione ai sensori di Hololens 2 per lo *spatial awareness* la stabilità degli ologrammi ne risente, usando solamente il sistema interno di *tracking* realizzato attraverso sensori IMU è possibile notare leggeri scostamenti degli ologrammi rispetto a dove ci si aspetterebbe di trovarli, tali errori di rappresentazione vengono corretti non appena il sensore ostruito torna a funzionare liberamente.

- Anche sforzando l'applicativo quindi caricando tante immagini o modelli tridimensionali e utilizzando le diverse funzionalità contemporaneamente le prestazioni del sistema non calano e gli ologrammi mostrati rimangono *responsive*.

- Il *rendering* del video in arrivo derivante da una video chiamata comporta un drastico calo delle prestazioni del sistema, in particolare gli FPS diminuiscono rendendo difficile continuare ad utilizzare l'applicativo, se gli esperti del dominio dovessero essere d'accordo tale funzionalità potrebbe essere rimossa, lasciando solo la possibilità di effettuare chiamate mostrando a chi risponde la sala operatoria, ma senza che il chirurgo possa prendere visione di chi sta parlando.

Allo stato corrente il sistema è in grado di gestire contemporaneamente più visori in diverse sale operatorie, ciò è stato testato simulando *client* che fingessero di essere visori facendo richieste al *back-end* e confrontando i risultati ottenuti con quelli aspettati, in tutti i casi questi *test* hanno dato esito positivo anche su lunga durata con tempi di risposta più che accettabili. In generale il sistema proposto, salvo le complicazioni derivanti dalle video chiamate, risulta una valida base di partenza prototipale per il sistema studiato in questa tesi.

Purtroppo durante la stesura di questa tesi la grave situazione epidemiologica di questi tempi ha limitato le occasioni di interazione con gli esperti del dominio, rendendo complicato far testare il sistema proposto ai diretti interessati, di seguito sono elencate le metodologie proposte per effettuare la validazione con il contributo dei chirurghi:

- **Sessioni finte**: il chirurgo indossa il visore e fa partire l'applicativo senza operare, in questo modo è possibile raccogliere i primi *feedback* relativi all'interfaccia proposta in termini di interazioni ed intuitività del sistema, raccogliendo inoltre le opinioni sulla scelta dei colori degli elementi e della loro disposizione iniziale.

- **Sessioni simulate**: sfruttare un ambiente di *training* solitamente usato dai chirurghi per allenarsi o studiare per un intervento, dove i soggetti operati sono manichini o rappresentazioni di organi finti eludendo a recar danno alle persone, in questo modo è possibile raccogliere i *feedback* relativi all'usabilità del sistema e all'utilità effettiva delle funzionalità messe a disposizione.

- **Sessioni reali controllate**: una volta verificata l'utilità del sistema è necessario verificare che in una situazione reale la possibilità di interagire con esso venga sfruttata naturalmente o risulti essere d'intralcio, per capire su interventi di lunga durata l'efficacia del sistema proposto. Per realizzare i primi test sarà necessario sfruttare interventi semplici e controllati da ulteriori chirurghi per prevenire qualsiasi forma di danno al paziente.

4.2 Sviluppi futuri

Possiamo vedere i possibili sviluppi futuri divisi in due categorie, quelli che riguardano uno specifico intervento e quindi aggiungono strumenti o funzionalità utili a quella particolare operazione oppure quelli che riguardano la base di partenza cioè che aggiungono o modificano comportamenti e funzionalità degli strumenti comuni a tutti gli interventi, in particolare in questa sezione tratteremo solo di quest'ultima categoria di aggiornamenti in quanto riguardano direttamente il progetto di tesi, mentre gli altri sono sviluppi futuri specifici e possibili a seguito di una profonda collaborazione con lo staff chirurgico, che rappresentano comunque uno sviluppo del sistema ma che non fanno parte del caso di studio di questa tesi, di seguito alcuni raffinamenti proposti a migliorare il progetto sviluppato:

- Autenticazione per l'accesso a tutte le componenti del sistema per identificare chi le sta usando e permettere la gestione dei permessi di modifica lettura o scrittura nei vari ambiti, ad esempio tutti possono prendere visione dello storico degli interventi ma solo il chirurgo assegnato ha la possibilità di modificarne uno non ancora svolto, garantendo in questo modo sicurezza e tracciabilità, tutte le operazioni svolte possono essere salvate in un *log* seguite dal nominativo di chi le ha compiute.

- Ampliamento del sistema per la richiesta di informazioni anche a macchinari diversi dai sensori per i parametri vitali, ad esempio l'integrazione di macchine per effettuare in tempo reale analisi al paziente e ottenerne i risultati in modo agevole direttamente sul **workbench virtuale**.

- Integrazione con gli strumenti in realtà mista, prendere visione ad esempio delle immagini generate da ecografi o doppler tramite il visore oppure la possibilità di osservare allo stesso modo il video generato da sonde interne per le procedure come gli interventi mini-invasivi.

- Interfacciamento con la sala operatoria, poter quindi consultare i dati relativi all'ambiente in termini di luminosità, temperatura, umidità relativa e assoluta, percentuale d'ossigeno nell'aria e tutto quello che potrebbe tornare utile conoscere, nel caso in cui la sala operatoria sia dotata di specifici attuatori è ragionevole pensare di integrare nel *workbench virtuale* la possibilità di modificare a piacimento ognuno di questi parametri.

- Rimozione del video in ingresso di una video chiamata, esaminando questa funzionalità si è giunti a conclusione che vedere il medico consulente in faccia non è utile ai fini dell'intervento, inoltre rimuovendo questa possibilità si va a risparmiare una notevole quantità di potenza computazionale, rendendo questa scelta inevitabilmente corretta.

L'idea del **workbench virtuale** è quella di racchiudere tutti gli strumenti comuni alla maggior parte degli interventi in un unico sistema agevole, lasciando implementazioni specifiche a sviluppi futuri, questo non significa che un chirurgo sia costretto ad usare in ogni intervento tutti gli strumenti messi a disposizione dalla versione base, in realtà l'obiettivo è quello di non costringere **mai** il chirurgo a perdere tempo per interagire con il sistema e allo stesso tempo fornirgli funzionalità comode che può sfruttare durante l'intervento come, quando e quanto ritiene opportuno.

Conclusioni

In questa tesi è proposto un sistema per l'implementazione di un **workbench virtuale** in realtà mista per l'ambito *healthcare* tramite l'uso di Hololens 2, essa ha l'obiettivo di supportare il chirurgo in sala operatoria, mettendo quindi a disposizione tutta una serie di funzionalità e strumenti che, in accordo con la letteratura analizzata 2 e l'opinione degli esperti del dominio, possono essere utili durante un intervento. Come sottolineato dagli sviluppi futuri 4.2 il sistema è stato ideato per essere ampliato sia nelle sue funzioni base sia per l'implementazione di un supporto specifico a particolari interventi, la modularità con cui è stato progettato consente di aggiungere o modificare comportamenti in modo semplice e intuitivo rendendolo di fatto un'ottima base di partenza per le applicazioni in realtà mista in sala operatoria. A fronte degli sviluppi futuri proposti, grazie alle opinioni positive raccolte, risulta promettente proseguire lungo la strada intrapresa da questa tesi, arricchendo di funzionalità il **workbench virtuale** e perseguendo l'obiettivo di questo progetto: supportare il chirurgo in sala operatoria con strumenti agevoli che non gli siano mai d'intralcio.

Ringraziamenti

Vorrei ringraziare Lucia Sacchetti, la quale mi ha sempre supportato durante questi anni di studio motivandomi a non arrendermi mai.

Bibliografia

[1] Steve Aukstakalnis. *Practical Augmented Reality A Guide to the Technologies, Applications, and Human Factors for AR and VR.* Addison-Wesley, 2016.

[2] Ronald T. Azuma. A survey of augmented reality. 1997.

[3] Olivier Bau and Ivan Poupyrev. Tactile feedback technology for augmented reality. 2012.

[4] Mark Billinghurst, Adrian Clark, and Gun Lee. A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8, 2015.

[5] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality Merging Real and Virtual Worlds.* A K Peters Wellesley, Massachusetts, 2006.

[6] Jack C.P. Cheng Keyu Chen and Weiwei Chen. Comparison of marker-based ar and markerless ar: A case study on indoor decoration system. 2017.

[7] Gunn Chris, Hutchins Matthew, Adcock Matt, and Hawkins Rhys. *Trans-World Haptic Collaboration.* Association for Computing Machinery, 2003.

[8] Irena Cronin e Robert Scoble. *The Infinite Retina.* Packt, BIRMINGHAM - MUMBAI, 2020.

[9] Florentin Liebmann e Simon Roner e Marco von Atzigen e Davide Scaramuzza e Reto Sutter e Jess Snedeker e Mazda Farshad e Philipp Furnstah. Pedicle screw navigation using surface digitization on the microsoft hololens.

[10] Simon G., Fitzgibbon A.W., and Zisserman A. *Markerless tracking using planar structures in the scene.* 2000.

[11] R. Galati, M. Simone, G. Barile, R. De Luca, C. Cartanese, and G. Grassi. Experimental setup employed in the operating room based on virtual and mixed reality: Analysis of pros and cons in open abdomen surgery. *Journal of Healthcare Engineering*, 2020, 2020.

[12] Kiyoshi Kiyokawa, Yoshinori Kurata, and Hiroyuki Ohno. An optical see-through display for mutual occlusion with a real-time stereovision system. *Computers & Graphics*, 2001.

[13] David M. Krum, Evan A. Suma, and Mark Bolas. Augmented reality using personal projection and retroreflection. 2012.

[14] Brian D. Hall Maximilian Speicher and Micheal Nebeling. What is mixed reality? 2019.

[15] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. 1994.

[16] Philip Pratt, Matthew Ives, Graham Lawton, Jonathan Simmons, Nasko Radev, Liana Spyropoulou, and Dimitri Amiras. Through the HoloLens™ looking glass: augmented reality for extremity reconstruction surgery using 3d vascular models with perforating vessels. *European Radiology Experimental*, 2018.

[17] Ramesh Raskar, Greg Welch, Kok-Lim Low, and Deepak Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. 2001.