Politecnico di Torino

Politecnico di Torino

Ingegneria del cinema e dei mezzi di comunicazione
A.A. 2022/23
Sessione di Laurea Luglio 2023

Compositing real-time in live streaming 360

Studio e applicazione di tecniche innovative per l'inserzione di oggetti sintetici su un flusso video livestream a 360°

Relatore:
Prof. Andrea Bottino

Candidato:
Brando Ramello 292400

Co-relatori:
Prof. Francesco Strada
Prof.ssa Vanessa Vozzo

Abstract

“Presence. XR Live Experiences” è un *long-term project* prodotto da Officine Sintetiche e consiste in una serie di “esplorazioni” sull’identità e la moltiplicazione del sé in rete, nella virtualità e nei social media: queste ricerche sfociano in *XR live experiences* dove realtà e virtualità si mescolano, anche grazie a tecniche di *embodied narrative* e di scrittura transmediale/crossmediale.

Per continuare a esplorare queste tematiche e concepire ulteriori sviluppi è stato intrapreso, attraverso questa tesi, un percorso di analisi e sperimentazione sulle possibilità di integrazione fra video live-action 360° e oggetti 3D.

L’intento è stato quello di sfruttare appieno il flusso video in *livestream*, per carpire informazioni in tempo reale sull’ambiente e sfruttarle in modo interattivo all’interno della scena.

E’ stato di conseguenza studiato un sistema per posizionare in modo coerente gli oggetti 3D nello spazio 360, attraverso la proiezione corretta delle ombre e mantenendo un corretto senso delle proporzioni.

E’ stata inoltre implementata l’interazione in tempo reale dell’utente con l’ambiente circostante, attraverso il tracciamento delle mani: l’intento è quello di favorire una maggiore immersione per l’utente, grazie alla sensazione di stare imprimendo un cambiamento tangibile nella scena virtuale.

Sono state anche sfruttate tecnologie di *Computer Vision* per il riconoscimento di corpi umani all’interno della scena, al fine di sfruttare i loro movimenti e azioni in modo interattivo.

Per studiare gli effetti dell’*embodiment* e la moltiplicazione del proprio corpo, tematica fondamentale analizzata nelle esperienze targate *Presence*, è stata data la possibilità di impersonare un avatar virtuale, dotato di *rig* controllato dalle mani dell’utente.

In conclusione, al fine di validare le scelte progettuali e implementative, sono stati condotti dei test con utenti, volti a raccogliere dati qualitativi e quantitativi.

Indice

1. Introduzione **6**
1.1 Il progetto “Presence” 6
1.1.1 Ispirazioni 7
1.1.2 Missing Out 14
1.1.3 Tiny Uppercase. Il nostro senso nascosto 17
1.1.4 Are you there? 20
1.2 Compositing 25
1.2.1 Cenni storici 25
1.2.2 Approccio non real-time 29
1.2.3 Approccio real-time 33
1.3 Video 360° 38
1.3.1 Definizione e tecnologie impiegate 38
1.3.2 Tipologie di proiezioni 41
1.3.3 360 Monoscopico e Stereoscopico 43
2. Stato dell’arte **46**
2.1 Sistemi per compositing a 360° in VR 46
2.1.1 MR360: Mixed Reality Rendering for 360° Panoramic Videos 46
2.1.2 Real-time Virtual Object Insertion for Moving 360° Videos 50
2.2 Body Tracking in Video 360° 52
2.2.1 Real-time object detection in 360-degree videos 53
2.2.2 A Dataset of Annotated Omnidirectional Videos for Distancing Applications 55
3. Metodo **59**
3.1 Compositing 360° 59
3.1.1 Acquisizione dei dati in livestream 59
3.1.2 Il rapporto fra spazi reali e spazi virtuali 62
3.1.3 Problematiche e soluzioni 64
3.1.4 Approccio a camera singola 68
3.1.5 Approccio a camera doppia 69
3.1.6 Illuminazione 74
3.2 Virtual Embodiment 77
3.2.1 Approccio first-person 80
3.2.2 Approccio third-person 81
3.3 Body Tracking e Pose Estimation 82
3.3.1 Utilizzo di OpenPose in 360° 82
3.3.2 Gestione della latenza 87
3.4 Scelte implementative 88
3.4.1 Stereoscopia del livestream 88
3.4.2 Gestione della doppia camera 90

3.4.3 Compromessi sull'illuminazione 92
3.4.4 L'embodiment 93
3.4.5 Applicazione di OpenPose 94
4. User Test 100
4.1 Prima parte 100
4.1.1 Fase 1 - Senso di presenza e immersione 104
4.1.2 Fase 2 - Efficacia dell'interazione 106
4.1.3 Fase 3 - Compositing 109
4.1.4 Fase 4 - Auto-coscienza corporea 112
4.2 Seconda parte 116
4.2.1 Fase 1 - Valutazione delle condizioni statiche 117
4.2.2 Fase 2 - Valutazione delle condizioni dinamiche 119
5. Primo scenario implementativo 121
5.1 La concezione 122
5.2 La formalizzazione logica 126
5.3 L'implementazione 127
5.4 Feedback 127
6. Conclusione e sviluppi futuri 129
Bibliografia e Sitografia 130
Elenco delle figure 136

1. Introduzione

Stiamo assistendo, negli ultimi anni, ad un notevole incremento di interesse nei confronti delle tecnologie *VR* e di ripresa 360°: eventi significativi come la nascita del Metaverso hanno portato a conoscenza del pubblico *mainstream* un ambito che, fino a poco tempo fa, era considerabile di nicchia e destinato a pochi interessati.
Il video 360° scardina completamente le grammatiche note ai *creator* e ai cineasti e apre loro nuovi orizzonti ancora da esplorare, consentendo allo spettatore una sensazione di immersività impossibile da ottenere con la fruizione classica.
I visori di realtà virtuale, d'altro canto, stanno via via diventando economicamente più accessibili al pubblico, e rappresentano una tecnologia applicabile virtualmente in qualsiasi campo (medicina, *training*, turismo, *gaming*, *media art*, spettacoli performativi), creando sinergie e intersezioni fra ambiti che finora parevano completamente scorrelati.
E' proprio dall'incontro fra *media art* e realtà virtuale che nasce questa tesi.
Partendo da "Presence. XR Live Experiences", un *long-term project* prodotto da Officine Sintetiche che esplora la moltiplicazione e l'autorappresentazione del sè nella virtualità, nei social e in rete tramite esperienze XR live, è stato intrapreso un percorso di studio e sperimentazione sulle tecniche di *compositing* di oggetti sintetici su video 360°.

1.1 Il progetto "Presence"

Il progetto "Presence. XR Live Experiences" si sviluppa in seno alla piattaforma Officine Sintetiche, una realtà che promuove e sviluppa forme artistiche e creative nel campo della creatività digitale e della *media art*, favorendo la creazione di relazioni fra professionalità assai diverse: performer, artisti e creativi si interfacciano con ingegneri e architetti [1].
Presence consiste in una serie di "esplorazioni" intorno all'identità, all'autorappresentazione e alla moltiplicazione del sè in rete, nei social network nella virtualità: nascono delle *XR live experiences* in cui realtà e virtualità si mescolano, dove vengono utilizzate tecniche di *embodied narrative* e di scrittura intermediale/crossmediale/transmediale.

In questo capitolo, grazie alla documentazione fornita da Vanessa Vozzo, co-fondatrice di Officine Sintetiche e co-relatrice di questa tesi, andremo ad analizzare il processo che ha permesso la nascita del progetto *Presence*.

1.1.1 Ispirazioni

Le sperimentazioni di *Presence* nascono dagli studi di gruppi di ricerca che si occupano di realtà virtuale o di scienze cognitive, come il *Brain, Body and Self Laboratory* del Group Ehrsson o l'*Event Lab* di Mel Slater ma anche da numerosi articoli accademici che trattano a loro volta di processi mentali e tecnologie immersive.
Inoltre, gruppi come *BeAnotherLab* rappresentano un'ulteriore fonte di ispirazione, lavorando al confine fra arte, scienze cognitive e tecnologia. [2]

Barbie Doll Illusion

Un esperimento molto noto è quello della *Barbie Doll Illusion* [3], condotto dal team di ricerca svedese Group Ehrsson con l'intento di rispondere a un classico quesito filosofico: la percezione del mondo può essere influenzata dalle dimensioni del proprio corpo?
I partecipanti venivano fatti sdraiare supini e dotati di visore per realtà virtuale (*HMD, Head Mounted Display*) che proiettava un flusso video live proveniente da due camere sincronizzate posizionate su un corpo artificiale di dimensioni variabili, in modo tale da rendere una prospettiva in prima persona: risultavano quindi solo visibili le gambe e il basso ventre, come si evince dalla Fig. 1.

Fig. 1 La prospettiva del partecipante: il bastoncino che tocca il piede sinistro è mosso da chi conduce l'esperimento.

L'esperimento voleva dimostrare come il proprio corpo giochi un ruolo chiave nella percezione visiva delle distanze e delle proporzioni, anche se non si tratta di quello reale ma di un'illusione percepita come veritiera.

Come mostrato in Fig. 2, sono stati utilizzati quattro corpi di materiale e dimensioni variabili, per testare quanto variasse la percezione degli oggetti e degli spazi proposti.

Il partecipante veniva stimolato sul piede da un oggetto, mentre, in perfetta sincronia, il corpo artificiale veniva toccato nello stesso punto, come mostrato in Fig. 1.

Dopo diversi minuti di questa stimolazione visuo-tattile, venivano proposti al partecipante task differenti in base alle ipotesi da verificare: poteva essere richiesto di stimare la grandezza dell'oggetto che veniva posto vicino al corpo artificiale così come la sua distanza, come si evince dalla Fig. 3.

Fig. 2 I quattro corpi artificiali utilizzati per l'esperimento Barbie Doll, rispettivamente, da sinistra a destra, di 400, 180, 80 e 30 centimetri.

Fig. 3 A sinistra, stima della dimensione dell'oggetto; a destra, stima della distanza.

I risultati hanno confermato le ipotesi iniziali: se il partecipante aveva "impersonificato" un corpo più piccolo, gli oggetti parevano più grandi e lontani; viceversa, in caso di corpo artificiale più grande, gli oggetti parevano più piccoli e vicini.

Inoltre, l'illusione di "essere dentro" un corpo artificiale si è mostrata assai efficace soprattutto grazie al tocco sincrono delle parti del corpo, e decisamente meno d'impatto in caso di tocco asincrono: addirittura è stato sperimentato uno scenario in cui il basso ventre del corpo artificiale veniva tagliato da un coltello, provocando reazioni emotive forti proprio se anticipato dalla stimolazione sincrona.

Sempre nell'ambito delle tecnologie immersive e delle scienze cognitive, vi sono articoli accademici che hanno rappresentato un motivo di interesse per le indagini del progetto *Presence* e che hanno posto la base per future sperimentazioni: ne analizziamo brevemente due, in quanto esplorano tematiche ricorrenti che vedremo in lavori successivi.

Il primo si intitola "Egocentric Smaller-person Experience through a Change in Visual Perspective" [4], ed è frutto del lavoro di ricercatori dell'Università di Tsukuba, in Giappone: tramite una videocamera indossata sulla vita, il partecipante si trovava a vedere il mondo, attraverso un *HMD*, da una prospettiva insolita, creando l'illusione di essere una persona notevolmente più bassa.

Semplici task come una stretta di mano con un interlocutore si sono rivelati estremamente difficili, in quanto il partecipante sollevava la mano molto più in alto del dovuto, percependo la propria rappresentazione corporea come più piccola.

Lo spazio personale percepito è risultato molto distorto, dal momento che l'interlocutore in avvicinamento veniva recepito con un senso di oppressione, vista la sua altezza.

Le interazioni dei partecipanti in luoghi con molte persone sono di notevole interesse: è stato notato come molti di loro iniziassero a comportarsi come bambini, cercando di proteggersi se accerchiati da adulti.

Le tre situazioni illustrate sono mostrate in Fig. 4.

Fig. 4 Da sinistra a destra: l'errore nel task della stretta di mano, la differente percezione dello spazio personale, l'istinto ad assumere una posa protettiva.

Questo studio ha quindi mostrato come si possano alterare la percezione umana, le azioni e le interazioni tramite l'illusione di avere un corpo più piccolo, come per altro già dimostrato dal *Barbie Doll Effect* [3].

Il secondo articolo si intitola "Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments" [5], ed è scritto da Mel Slater, figura chiave per le ricerche condotte nel campo delle scienze cognitive e le tecnologie immersive.
In questa ricerca, Slater si interroga sul perché le persone tendano a rispondere in modo realistico a situazioni ed eventi che sono mostrati attraverso un sistema di realtà virtuale.
Vengono definiti due parametri "ortogonali" che contribuiscono alla creazione di una reazione realistica alle situazioni in realtà virtuale:

- *Place Illusion* (*PI*): l'illusione di "essere lì", nonostante si abbia l'assoluta certezza di non essere lì.

  Questa è fortemente condizionata dalle *sensorimotor contingencies* del sistema, ovvero le azioni che l'utente può svolgere al fine di percepire l'ambiente virtuale (ad esempio, muovere o ruotare la testa): ci sarà sempre un limite oltre il quale, in un sistema VR, queste *sensorimotor contingencies* falliscono e la *PI* viene meno.

  I sistemi *VR* vengono definiti *first-order systems* se forniscono un set di azioni tali da approssimare la realtà, mentre i *second-order systems* hanno un *subset* di azioni di un *first-order*, e così via.

  La *PI* è resa possibile in un sistema *first-order* poiché conseguenza delle proprietà fisiche del set-up, mentre nei sistemi di livello inferiore può essere presente, ma solo come frutto di processi creativi mentali aggiuntivi.
- *Plausibility Illusion* (*Psi*): l'illusione che quello che si sta vedendo stia davvero accadendo, nonostante si abbia la certezza che non è così.

  La *Psi* è determinata da quanto un sistema possa produrre eventi che si relazionano con l'utente: è importante che ci sia correlazione fra eventi "esterni", non condizionati dall'utente, e le sue reazioni di quest'ultimo.

Se un *avatar* virtuale iniziasse a interagire con noi in modo realistico, il nostro corpo potrebbe reagire in modo “naturale”, quindi innalzando il battito cardiaco o facendoci arrossire.

Gli eventi proposti devono essere credibili rispetto alle aspettative dell’utente, che utilizza l’esperienza della vita reale come punto di riferimento.

Nonostante sia una condizione molto difficile da ottenere, la presenza simultanea di *PI* e *Psi* può quindi causare una reazione perfettamente realistica, da parte di un utente *VR*, a situazioni ed eventi proposti.

VR in Wonderland

VR in Wonderland è un *research setting* presentato ad Ars Electronica, in Austria nel 2019. [6]

Si tratta del lavoro che permetterà la nascita del progetto *Presence*, ed è stato concepito da Vanessa Vozzo, co-fondatrice della piattaforma Officine Sintetiche, in seguito a una ricerca svolta a partire dal 2018 con un gruppo di studenti e ricercatori dell’*Interface Culture Master* della Università di Arte e Disegno Industriale di Linz, in Austria.

In *VR in Wonderland* viene esplorato il senso della propriocezione e dell’auto-collocazione nello spazio attraverso strumenti e dispositivi 360° immersivi in *streaming real-time*.

Indossando un *HMD*, l’utente vede dalla prospettiva di un piccolo dispositivo robotico sul quale è installata una videocamera 360° *consumer*: essa si trova in tempo reale in uno spazio adiacente a quello dell’utente, il quale può spostarsi all’interno di un modello in miniatura di una città labirintica pilotando il dispositivo tramite un *joystick*.

In questo esperimento l’assottigliarsi della distanza tra l’ambiente reale e l’ambiente virtuale provoca una distorsione spazio-temporale che si amplifica nel momento in cui l’utente vede sé stesso dal basso verso l’alto percependosi, dall’esterno, come un gigante.

Nella Fig 5. Viene mostrato il *setup* completo dell’esperienza.

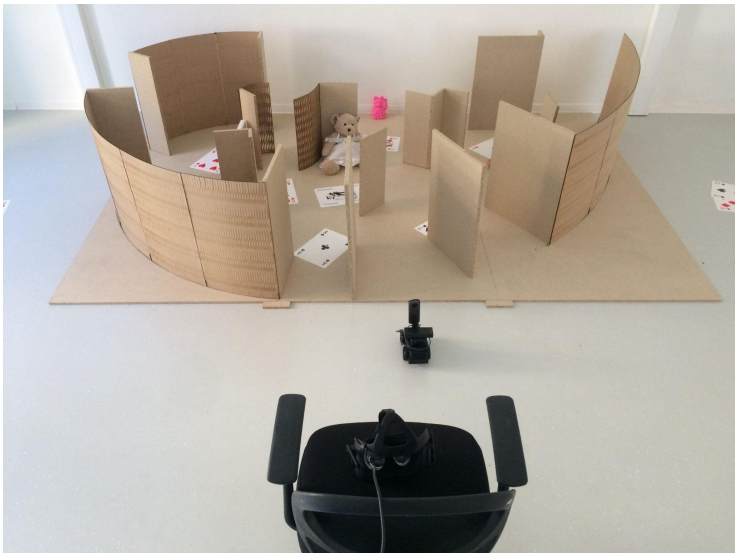

Fig. 5 Il setup di VR in Wonderland: la postazione per l'utente, il dispositivo telecomandato con la camera 360° e il "labirinto" da esplorare.

Gli utenti che hanno partecipato a questo esperimento concordano nel dire di avere avuto la sensazione corporea di alienazione e di straniamento, come se si trovassero in un "altrove" in bilico tra realtà e virtualità.

Queste sono quindi le basi da cui nasce *Presence*.
Le "esplorazioni" citate all'inizio del capitolo si svolgono a partire da un confronto con docenti e ricercatori che si occupano di *media art* o di sociologia dei nuovi media, ma si sviluppano soprattutto grazie a specifici *focus group* formati da studenti, ricercatori, docenti e artisti dai quali emergono tematiche che danno origine ai progetti artistici che andremo ad analizzare fra poco.
L'idea è di potersi approcciare a tematiche sociali e politiche ampie e in continua evoluzione da una nuova, dinamica e coinvolgente prospettiva grazie all'utilizzo di tecnologie fortemente innovative.
Proprio lo sviluppo delle tecnologie è un tema importante in *Presence*: come mostrato in Fig. 6, vengono organizzate infatti giornate di *workshop* o seminari, oppure *contest* in cui ci si confronta con le tecnologie messe in campo.
I progetti artistici, in generale, mirano non solo ad annullare la distanza fra realtà e virtualità attraverso la distorsione spazio-temporale, ma anche a posizionare il corpo dell'utente al limite tra incarnazione e alienazione, alterando volutamente il senso della propriocezione al fine di generare esperienze fortemente significative.

Fig. 6 I focus group da cui nascono le idee per i progetti artistici di Presence.

La struttura narrativa gioca un ruolo decisivo: senza di essa le esperienze *XR* sarebbero una sperimentazione tecnica fine a sé stessa.
Essa nasce dai *focus group* e si struttura intorno ai media utilizzati nei progetti: vedremo ora tre progetti artistici che rientrano nel *long-term project* che è *Presence*, dove proprio la struttura narrativa ha contribuito attivamente al coinvolgimento dell'utente.

1.1.2 Missing Out

Missing Out è un'installazione/performance in cui si utilizzano sistemi *VR* 360°, *real-time streaming* 360° e video 2D, ed è stata presentata per la prima volta presso il Circolo del Design di Torino a gennaio del 2020 [7].
Il tema centrale di *Missing Out*, emerso dal lavoro collettivo e partecipativo nei *focus group* e da alcuni studi sulle ansie sociali legate ai social media: è la FoMO, *Fear of Missing Out*, ovvero la paura di essere esclusi, di essere tagliati "fuori".
Si tratta di uno stato di ansia sociale che scaturisce dal bisogno di essere sempre informati su tutto ciò che stanno facendo gli altri, ed è caratterizzata da una preoccupazione eccessiva e ossessiva che questi stiano facendo esperienze gratificanti, interessanti, eccitanti, ma senza di noi.
Questa pervasiva sensazione di apprensione può essere esasperata dalla continua visione di messaggi e post che altri individui pubblicano sui social.

Alcuni studiosi, come Andrew Przybylski e il suo team, hanno recentemente messo in luce diversi fattori basilari della FoMO e le conseguenze date dall'aumento dell'irrequietezza se impossibilitati di controllare le notifiche dei propri *social network* [8].

In *Missing Out* la percezione sensoriale, temporale e dell'auto-collocazione nello spazio circostante viene modificata in tempo reale creando un effetto disorientante tra realtà, virtualità e ambiente *streaming*.

Attraverso un percorso che passa da una visione 2D all'immergersi nella *VR* 360° fino a vivere la realtà trasmessa da un *live-stream* 360°, viene data allo spettatore la possibilità di vivere una precisa situazione che lo porterà ai confini del proprio corpo in una situazione extra-sensoriale.

Il percorso narrativo transmediale prevede uno spazio suddiviso in 3 aree: una con un monitor, una con un visore per video 360°, una con un visore *VR.*

Area 1: Un Viaggio (Monitor 2D)

Lo spettatore viene introdotto nell'installazione da un cortometraggio in cui una donna di mezza età tenta inutilmente di coinvolgere in un viaggio in India una persona di cui non si comprende l'età e sesso, che sta chattando di schiena in una penombra dalla quale emerge solo la luce del telefono.

Fig. 7 L'Area 1, con il monitor su cui osservare il cortometraggio.

Area 2: La Mia FoMO (Video 360°)

Terminato il video lo spettatore viene invitato da una guida nell'area dei visori 360° e dotato di un vero telefono al quale giungono continue notifiche.
Indossando il visore lo spettatore si ritrova dentro la scena del cortometraggio 2D appena visto, ma nei panni della persona in ombra, passando da una prospettiva in terza ad una in prima persona.
La donna protagonista del cortometraggio, poiché ignorata dalla persona in ombra (che qui è lo spettatore stesso), si innervosisce con lei.
Lo smartphone reale, nelle mani dello spettatore/personaggio, vibra e suona continuamente.

Fig. 8 L'Area 2, con il visore per video 360°.

Area 3: Alienazione (Live Streaming 360°)

Al termine del video 360°, una guida fa entrare lo spettatore all'interno di una stanza, la stessa dei due video precedenti.
Qui viene accolto dalla donna vista nei due video precedenti (una *performer* in carne e ossa), vestita come nei video.
La donna prende lo smartphone dalle mani dello spettatore e lo ripone su un tavolo dove si trova anche un visore per *VR* che la donna gli fa indossare: improvvisamente lo spettatore si ritrova ad osservare da un punto di vista diverso, ovvero quello di una camera 360° posta nella stessa stanza.
Lo spettatore vede sé stesso in tempo reale, impotente, di fronte al telefono che continua a squillare, mentre la donna lo accarezza amplificando il senso di straniamento.
Poi, oscura la camera portando lo spettatore nel buio forzato della propria mente.

Fig. 9 L'Area 3, la stanza arredata con visore e camera 360°.

1.1.3 Tiny Uppercase. Il nostro senso nascosto

Tiny Uppercase. Il nostro senso nascosto è un'esperienza che unisce *XR* e *live dance*, prodotta dal Balletto Teatro di Torino [9] e presentata per la prima volta a maggio del 2022 presso la Lavanderia a Vapore di Collegno (TO).

Anche qui il concept nasce dal confronto attraverso un *focus group* guidato da Viola Scaglione (direttrice artistica del Balletto Teatro di Torino) e Vanessa Vozzo con ricercatori, studenti e con gli artisti del Balletto Teatro di Torino sui temi e sulla ricerca del progetto *Presence*.

Dagli incontri scaturiscono alcune domande: quali sono i limiti del nostro corpo? È possibile creare le condizioni "virtuali" per percepire il corpo di un altro? Sperimentando un altro punto di vista, "entrando" nel corpo virtuale di una persona, posso sentirmi davvero diverso/a? Quanto questo processo di immedesimazione corrisponde all'empatia? Quanto influiscono gli eventi (reali o virtuali) e lo spazio (reale o virtuale), nel processo di immedesimazione?

L'idea di *Tiny Uppercase* si sviluppa quindi a partire dalla volontà di allargare lo sguardo interpretativo sul corpo verso una nuova auto-percezione e una più consapevole auto-collocazione spazio-temporale.

L'obiettivo è incentivare la trasformazione del modo di percepire il proprio corpo nello spazio e all'interno di eventi e, di conseguenza, di relazionarsi con gli altri attraverso nuovi linguaggi e nuove modalità: viene così a svilupparsi una comunicazione con l'altro più empatica e sinergica.

Il lavoro si concentra sul senso della propriocezione per rendere visibile una delle componenti più importanti dell'esperienza umana: la costruzione del proprio essere sè e la relazione che, attraverso il corpo, costruiamo con il mondo che ci circonda.
La struttura narrativa transmediale è composta da quattro differenti fasi: a differenza di *Missing Out*, tutto avviene in un unico spazio, un palcoscenico.

Fase 1: Realtà in Bianco e Nero (Live Streaming 360°)

Quattro spettatori vengono guidati sul palcoscenico e viene loro donato un minerale che potranno tenere in mano durante i 17 minuti di spettacolo successivi.
A seguire, ogni spettatore viene invitato a indossare un HMD.
A quel punto il punto di vista dello spettatore si disloca in quello di una videocamera 360° in *live-stream* posizionata sullo stesso palcoscenico, mentre la realtà si desatura in bianco e nero.
Tutti gli spettatori osservano sé stessi dal punto di vista della stessa videocamera, mentre alcuni danzatori interagiscono con loro toccandoli: da questo momento tutti e quattro gli spettatori vivono la stessa esperienza.

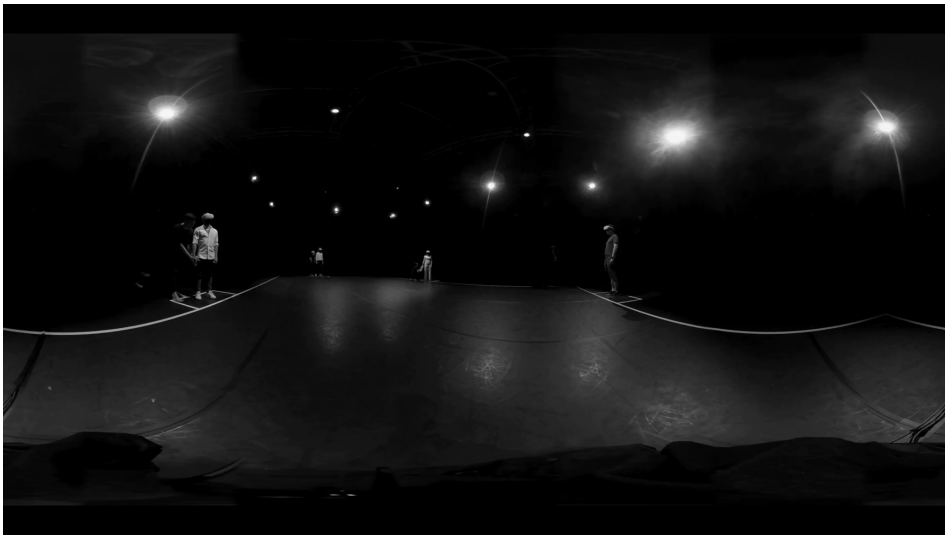

Fig. 10 Proiezione equirettangolare del punto di vista degli spettatori durante la Fase 1.

Fase 2: La Salita (VR)

Dalla realtà trasmessa in *live-stream* si passa al buio e poi in un ambiente virtuale, in grafica 3D.

Dall'interno di una grotta lo spettatore si sente sollevare verso l'alto fino a volare: vede sotto di sé un fiume avvolto in un'atmosfera lunare e vola fino a finire in uno spazio completamente bianco.

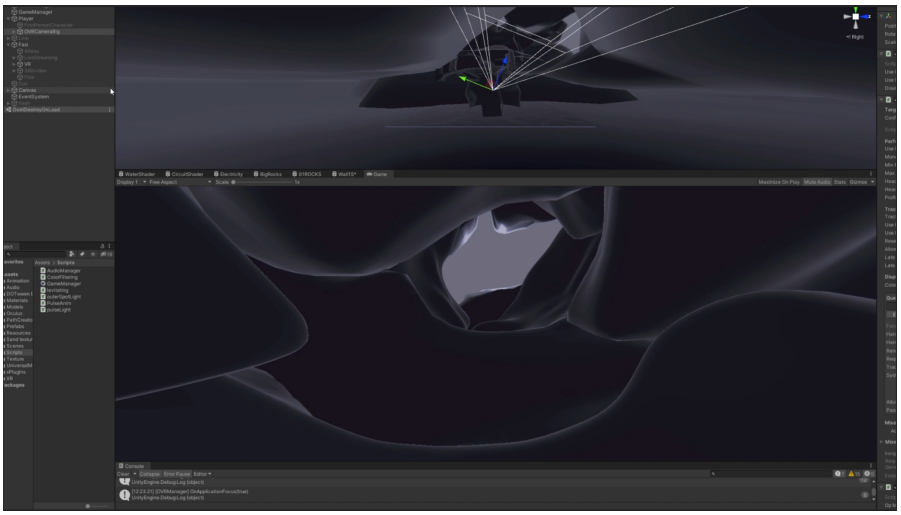

Fig. 11 Il modello 3D della grotta che si vede durante la Fase 2.

Fase 3: Il Deserto (Video 360°)

Uscito dallo spazio bianco lo spettatore si ritrova da solo, sulla riva di un fiume, in un ambiente deserto, dove lentamente compaiono "altre persone": queste iniziano a danzare e cercano lo sguardo dello spettatore per riportarlo a sé stesso.

Fig. 12 Proiezione equirettangolare del punto di vista degli spettatori durante la Fase 3.

Fase 4: Il Ritorno (Realtà)

Nell'ultima fase viene tolto il visore agli spettatori.
I danzatori reali sono nella stessa posizione e con gli stessi costumi con cui è terminato il video 360°: iniziano a danzare, in modo molto fisico, potente.

Fig. 13 La Fase 4 di Tiny Uppercase.

1.1.4 Are you there?

Are you there? è uno studio *in progress* presentato per la prima volta presso lo StudiumLab di Torino a Febbraio 2022.
Anche qui il tema nasce da un *focus group* con studenti dell'Università di Torino e del Politecnico di Torino, dal quale emerge un fatto sconcertante: l'enorme mole di fotografie che visioniamo tutti i giorni sui social è raramente accompagnata dalla nostra capacità di immaginare il contesto in cui è stata scattata.
Da questa presa di coscienza nasce il concept di *Are you there?*

> *La fuga sistematica dalla realtà attraverso i social media ci permette di vivere una virtualità in rete in cui possiamo moltiplicarci, diventare altro e comunicare online alterando, frammentando, iconizzando continuamente la nostra personalità.*
> *In questi spazi digitali si alternano continuamente istantanee di luoghi, persone, oggetti che rievocano ricordi ed eventi di questi altri noi. Ma quanto*

degli altri si può davvero afferrare da queste immagini e quanto, invece, si dissolve nella rete?
Come funziona la nostra immaginazione nella frenesia, nella velocità, nella sovrastimolazione degli scroll e dei touch in cui migliaia di fotografie si susseguono? Siamo ancora in grado di ricostruire gli spazi del sogno o della memoria nella nostra mente? Oppure la nostra mente sta diventando una sequenza di pics senza spazio?

La struttura narrativa transmediale è composta da quattro differenti fasi.

Fase 1: Il Contatto (Realtà + Smartphone)

Due spettatori vengono invitati ad entrare in una stanza da una guida: qui vengono accolti da tre *performer* che stanno apparentemente chattando ma, quando vedono i due spettatori, consegnano loro uno *smartphone*.
Gli spettatori vengono invitati, grazie ad un'applicazione realizzata ad hoc, a dare il proprio nome e inserire la propria password.
Da qui parte uno *scroll* che propone contenuti legati a "mare e parco giochi".
Lo *scroll* dura un minuto e trenta secondi, poi l'utente viene invitato ad indossare il visore posto al centro della stanza.
L'ultima immagine sullo *smartphone* è un ambiente "reticolato".

Fig. 14 Una delle immagini visionate nella Fase 1.

Fase 2: Il Mondo Popup (VR)

Gli spettatori, ognuno con il proprio visore, si ritrovano in un ambiente “reticolato” simile a quello appena visto sullo smartphone, ma in VR.
Qui vengono proposte le stesse immagini e messaggi della Fase 1, in maniera frenetica e senza pause.
Al termine, una sola immagine permane e si ingrandisce: è la foto di un parco giochi, già viste molte volte durante lo scroll, che diventa un ambiente immersivo.

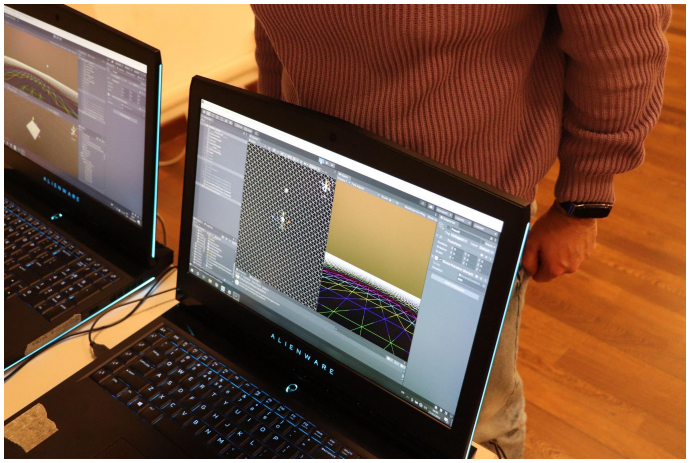

Fig. 15 L’interfaccia del motore di gioco che genera l’ambiente immersivo, in cui si può vedere il reticolato della Fase 2.

Fase 3: Il Parco Giochi (VR)

Gli spettatori entrano in un mondo virtuale in cui viene ricostruito un ambiente a partire dall’oggetto più significativo della foto del parco giochi in cui è “entrato” nella Fase 2.
Il mondo intorno si genera come se la memoria facesse fatica a ricostruire i ricordi: questa percezione frammentata emerge grazie ad effetti di sgretolamento, di dislocazione degli oggetti, di cambiamento di distanza degli oggetti dall’utente e fra di loro.
Al termine di questa fase si sente una suoneria di uno smartphone: il suono si avvicina sempre di più all’utente assieme ad una sfera.
La sfera è un ambiente nel quale gli spettatori entrano accompagnati dalle voci e dal tocco dei performer: alcune azioni degli stessi riportano gli spettatori nella realtà trasmessa in *live-stream*.

Ogni spettatore vede, infatti, dal punto di vista di una delle due videocamere 360° poste nella stanza: sono presenti i due spettatori e i *performer*.

Fig. 16 Il set-up della Fase 3: i due spettatori esplorano l'ambiente virtuale indossando l'HMD. Sono già presenti le camere 360° per la fase successiva.

Fase 4: My Real Self (Live Streaming + Realtà)

I performer sono presenti nella stanza fisica e sono visti dagli spettatori nel *live-stream* 360 ° attraverso l'alternanza dei punti di vista delle due camere, che sono poste ad altezze e punti diversi nella stanza.

Dopo alcune azioni dei *performer* le videocamere 360° vengono coperte da un telo nero: l'utente non vede più nulla, soo riverberi luminosi.

I *performer* fanno incontrare i due spettatori e tolgono loro i visori, facendoli ritrovare vicini.

Fig. 17 La Fase 4: l'incontro fra i due spettatori.

In questa sezione è stato quindi introdotto il progetto *Presence*.
Questa tesi nasce da alcune considerazioni tecniche sorte da una discussione fra l'autore, la stessa Vanessa Vozzo e l'altro co-relatore, Francesco Strada.
I primi tre progetti targati *Presence* hanno sicuramente un taglio fortemente transmediale, riuscendo a utilizzare *live-stream* 360°, VR, arti performative e 2D classico, ma mancano di momenti che permettano di "intrecciare" senza soluzione di continuità le "fasi" delle esperienze immersive.
L'intento di questa tesi è quindi quello di esplorare le possibilità del *compositing* di oggetti sintetici su un video 360° *live-stream*, al fine di poter creare esperienze più organiche, che possano passare da *live-stream 360°* a VR grazie a momenti transitori, in cui appunto oggetti 3D appaiono già nell'ambiente 360°.
Presence, come definito in precedenza, è un *long-term project* e, in quanto tale, è caratterizzato da un processo di ricerca continua che viaggia su due "piani paralleli", ovvero quello scientifico/sociologico e quello tecnologico: questa tesi si focalizza sul secondo, al fine di fornire mezzi tecnici innovativi che andranno ad essere utilizzati in futuri lavori.

Nella seconda parte di questo capitolo viene quindi spiegato il concetto di *compositing*, evidenziando le differenze fra l'approccio non *real-time*, tipico del cinema classico, con quello *real-time*, visto nelle esperienze XR.

1.2 Compositing

Il *compositing* è “la combinazione di elementi visivi provenienti da fonti separate in singole immagini, spesso per creare l'illusione che tutti questi elementi siano parte della stessa scena”. [10]
In questo capitolo il termine *compositing* verrà inteso e trattato come un sinonimo di “sovrapposizione di oggetti sintetici su un flusso video *live-action*”, ignorando quindi le sue accezioni più “semplicistiche”, ovvero, fra le altre, le sovraimpressioni di scritte o immagini sopra a flussi video (i trafiletti che passano durante i notiziari ne sono un esempio lampante).
Dal cinema su pellicola a quello in digitale, si è sempre fatto uso di questo principio, al fine di ottenere effetti non realizzabili, per i motivi più vari, fisicamente sul set.
Con l’avvento della tecnologia *AR* (realtà aumentata), si è visto un aumento di popolarità verso l’accezione *real-time* del *compositing*, in quanto gli oggetti sintetici vengono applicati sul video di sfondo in tempo reale, senza bisogno di estenuanti tempi di *rendering*, dal momento che le applicazioni che ne fanno uso hanno bisogno di *frame-rates* elevati.

1.2.1 Cenni storici

Il *compositing*, nella sua accezione classica, non quindi necessariamente di oggetti CG, nasce pochi anni dopo l’avvento del cinema stesso: la pratica dell’esposizione multipla è uno dei primi esempi di *visual effects* della storia, e consisteva nell’impressionare una seconda volta/più volte la pellicola su cui si stava girando, per ottenere effetti visivi direttamente *in-camera* [10].
Uno dei primissimi esempi di *compositing* ottenuto attraverso l’esposizione multipla è visibile nel film “The Great Train Robbery”, diretto da Edwin S.Porter nel 1903 [11]: nella prima scena, il *footage* del treno in movimento che si può vedere nella finestra a destra dell’inquadratura è stato impressionato sulla pellicola in una fase successiva, come indicato in Fig. 18.
In fase di ripresa, infatti, lo spazio circoscritto dentro la finestra è stato dipinto di nero, per far sì che la pellicola non venisse impressionata; durante la seconda esposizione è stato poi utilizzato un *matte* (mascherino) per coprire la porzione già esposta ed è stato filmato un treno in movimento per impressionare la parte di pellicola rimanente.

Fig. 18 L'ambiente interno è stato girato in una prima fase, il treno all'esterno in una seconda.

Altre tecniche che si sono sviluppate negli anni sono il *glass shot*, che consiste nel posizionare un vetro trasparente davanti alla camera disegnato in alcune porzioni specifiche al fine di contestualizzare il *footage* in un ambiente diverso [10], oppure la *background projection*, ovvero il proiettare dietro gli attori uno sfondo in movimento, come visto spesso nelle scene di guida dei veicoli, ottenendo una fusione perfetta fra le due immagini [10].

Fig. 19 L'utilizzo del glass shot in Tempi Moderni di Charlie Chaplin. - Immagine presa dal canale YT "Petr Pechar" [12]

Fig. 20 Background projection sul set di Eyes Wide Shut: a sinistra il footage finale, a destra l'attore su un tapis roulant cammina davanti allo sfondo proiettato. - Immagini prese dal canale YT "whospuss" [13]

Una delle più celebri tecniche classiche di *compositing* è quella del *blue/green screen*: l'azione del soggetto viene ripresa davanti a un telo di colore uniforme, come il blu o il verde, che viene poi rimosso in post produzione e sostituito con un girato di sfondo, chiamato *plate* [10].

Prima dell'avvento del digitale, tutto avveniva su pellicola attraverso l'utilizzo di un *traveling matte*, una versione del girato del soggetto in movimento da utilizzare in fase di post-produzione: facendo le riprese del *foreground* su sfondo blu e applicando un filtro sulla lente della camera, il soggetto veniva impressionato sulla pellicola mentre lo sfondo no.

Veniva poi fatta una copia ad alto contrasto del negativo, in modo da registrare l'area del background come opaca e quella del soggetto in primo piano come "pulita", e una seconda copia dalla prima con procedimento opposto.

Utilizzando una stampante ottica, ovvero un dispositivo composto da uno o più proiettori che permetteva di compositare elementi provenienti da diversi "strati" su un'unica pellicola, veniva posta nella parte inferiore la copia "vergine" della bobina, sopra di essa il mascherino con sfondo opaco e in cima il mascherino con soggetto opaco: in questa fase, veniva copiata sulla pellicola "vergine" l'azione del soggetto perfettamente scontornata dallo sfondo, che rimaneva quindi non impressionato.

Il processo veniva quindi ripetuto, ma disponendo in cima la scena di sfondo e sotto di essa il mascherino con soggetto opaco: in questo modo la silhouette del soggetto

andava a coprire le parti già esposte in precedenza e quindi venivano toccate solo quelle rimanenti.
Il risultato è quindi una copia “in positivo” dei contributi di sfondo e soggetto uniti realisticamente.

Fig. 21 Il matte che si ottiene durante il procedimento. - Immagine presa dal canale YT “Tom Scott” [14]

Con l’avvento del digitale, il processo di separazione fra il soggetto e lo sfondo è stato reso meno “fisico”, spostando tutta la complessità sul potere computazionale del software di *compositing*.
Non crediamo però che il *workflow* sia stato quindi semplificato: vi sono infatti numerosi “tricks” da utilizzare al fine di ottenere un soggetto perfettamente scontornato e poi correttamente integrato con il *background* [10].

1. *Relighting*: la “risistemazione” delle luci in maniera virtuale, in modo da simulare situazioni che non sono state rese possibili sul set.
2. Modulazione della luce: l’aggiunta di elementi chiari o scuri di luce sul soggetto, per dare una maggiore vita al girato (ad esempio, nel caso di una scena in una giungla, si possono simulare delle chiazze di luci e ombre sul soggetto).
3. *Edge Wrap*: una tecnica per integrare il soggetto, al fine di simulare cosa succederebbe se questo fosse ripreso in un ambiente reale (lo sfondo *straborda* sopra gli elementi frontali).

Si utilizza un *sobel matte*, ovvero un operatore differenziale discreto, che calcola un'approssimazione del gradiente della funzione dell'intensità dell'immagine: in poche parole, riconosce i bordi del soggetto e crea un contorno.

Utilizzando il *sobel matte* come guida, è possibile fondere con maggiore realismo lo sfondo con il *foreground*, con la consapevolezza che la tecnica dell'*edge wrap* deve essere utilizzata con parsimonia, in quanto può facilmente risultare esagerata e irrealistica.

4. *Edge Blending*: più di nostro interesse, in quanto utilizzato principalmente su oggetti CG, è una tecnica che si utilizza per simulare la luce che si disperde e i bordi degli oggetti che diventano più morbidi. Un soggetto compositato che presenta bordi netti sara probabilmente riconosciuto e ritenuto poco realistico
   Si utilizza sempre un *sobel matte*, più sottile rispetto all'*edge wrap*, che viene utilizzato per pilotare una composizione secondaria.

Questa è sola una parte delle tecniche che permettono di sovrapporre un soggetto ad uno sfondo: per questioni di tempo e di pertinenza, andremo ora a focalizzarci sull'argomento di nostro interesse, ovvero il *compositing* di oggetti sintetici/3D su un *background live action.*

1.2.2 Approccio non real-time

Il *compositing CG* prevede l'inserimento, all'interno del processo produttivo, di numerose figure professionali aggiuntive, come il *modelling artist*, l'*animator*, il *texture artist*, il *matchmove artist,* il *lighting artist*, l'*environment artist* e il *layout artist*, i quali sono supervisionati da il *look-development artist* e dal *CG supervisor*: ognuno ha il proprio specifico ruolo all'interno della *pipeline* e non è difficile capire quanto la complessità del processo aumenti.

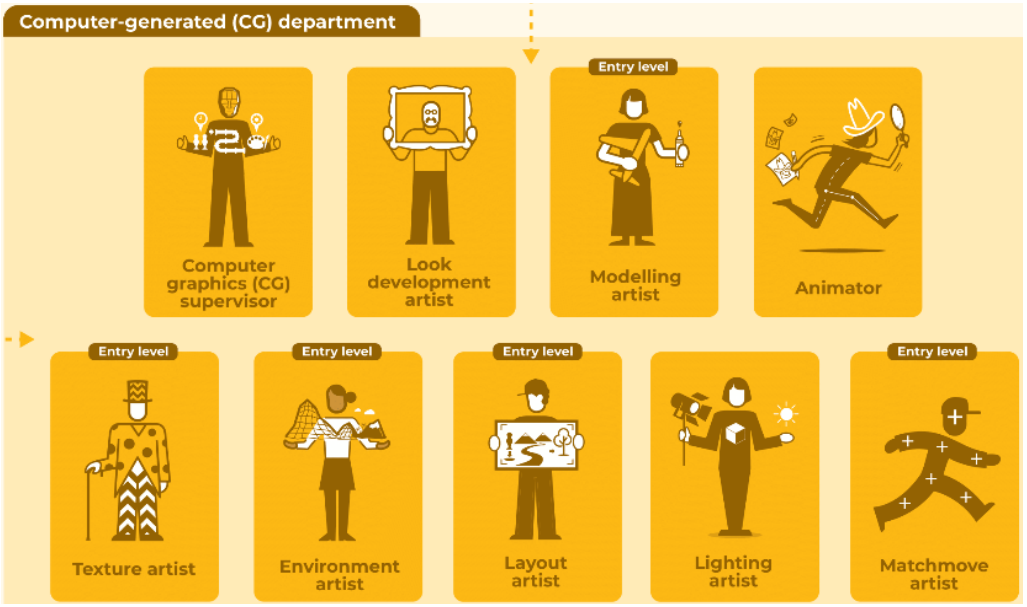

Fig. 22 Il reparto, all'interno di uno studio di VFX, che si occupa della computer grafica. - Immagine presa dal sito "Screen Skills" [15]

Una volta che un elemento CG è stato correttamente modellato, ha ricevuto le *texture,* è stato illuminato in modo coerente ed eventualmente animato, viene "passato" al *compositor*, colui che si occupa dell'unire i vari *layers* per formare l'immagine finale.

In caso di un errore o un'incongruenza di luce, capita spesso di dover renderizzare nuovamente l'oggetto o la scena: si tratta di un procedimento fatto di numerose iterazioni e quindi dispendioso a livello di tempo [10].

Nonostante le tecnologie si stiano evolvendo e le tempistiche quindi diminuendo, ogni *shot* può impiegare diverse ore per essere renderizzato.

Inoltre, ovviamente, non si può solo pensare di "appoggiare" sul girato *live action* l'oggetto sintetico: questo, se si tratta di un personaggio animato, probabilmente andrà ad interagire con l'ambiente, rendendo complessa la sua "fusione" con lo stesso.

Il *compositor* riceve le ombre proiettate dai modelli 3D sul terreno e deve essere abile a correggere il colore per rendere il risultato realistico (banalmente, i livelli di nero di ombre e *background* possono essere differenti); ancora, il *footage* di sfondo *live action* avra probabilmente del rumore o della granularità, che deve essere "estratta" e applicata anche sugli oggetti sintetici per uniformare il tutto.

Ovviamente, non tutto il processo di *compositing* avviene in post-produzione, ma è necessario un grande sforzo anche da parte del reparto di pre-produzione e produzione: bisogna correttamente pianificare le riprese che avranno bisogno di un contributo CG, al fine di facilitare il lavoro del *compositor*.

Nasce quindi una disciplina nuova, che vuole aiutare l'integrazione fra la produzione *live action* e la post produzione digitale, la *on-set data acquisition* (acquisizione dei dati sul set).

Un importantissimo *tool* che può essere utilizzato al fine di acquisire informazioni sul "mondo reale" all'interno del set è la tecnologia *lidar* (*Light Detection And Range*) [10].

Questa permette di effettuare un *laser scanning* sfruttando la velocità della luce: emettendo un fascio luminoso, si può calcolare quanto tempo questa impiega per tornare indietro e quindi registrare le coordinate x, y e z nello spazio per ogni punto riflesso.

Vengono quindi generate delle nuvole di punti che, se opportunamente filtrate, generano un'ottima approssimazione della topologia dell'ambiente di ripresa.

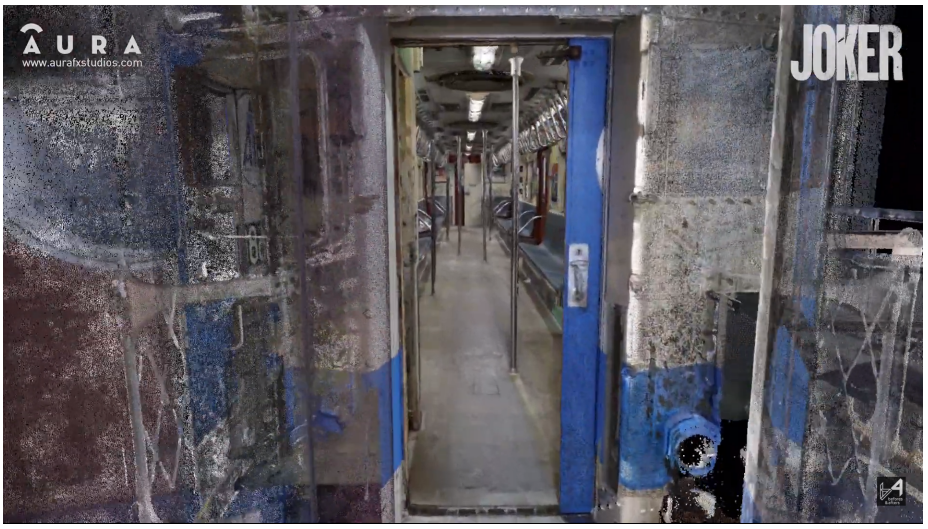

Fig. 23 La ricostruzione ottenuta con tecnologia lidar, ad opera del team Aura FX [16], di una metropolitana, utilizzata poi nel film Joker del 2019. - Immagine presa dal canale YT "before & afters" [17]

I dati raccolti in questo modo possono essere utilizzati in numerose fasi della *pipeline*, dalla pre-visualizzazione alla creazione di *visual effects*: commentiamo brevemente due applicazioni che frequentemente vediamo nei prodotti cinematografici odierni.

- *Set extension*: si tratta di un'evoluzione del *matte painting* o del *glass shot*, in quanto consiste nell'estendere l'ambiente di ripresa con elementi architettonici altrimenti troppo dispendiosi da costruire fisicamente.

  La porzione "estesa" dell'inquadratura deve essere perfettamente "allineata" dal punto di vista artistico e geometrico con il girato *live-action* e la tecnologia *lidar* può essere di grande aiuto: fornendo un'approssimazione della geometria dell'ambiente di ripresa, risulta molto più facile gestire gli spazi e le proporzioni, così come le sorgenti di luce virtuali.
- Personaggi CG: come sopra, la nuvola di punti fornita dal *lidar* consente di creare una zona di azione confinata per i personaggi sintetici animati presenti in scena, ma anche di proiettare ombre coerenti sul modello 3D o sulla geometria del mondo reale.

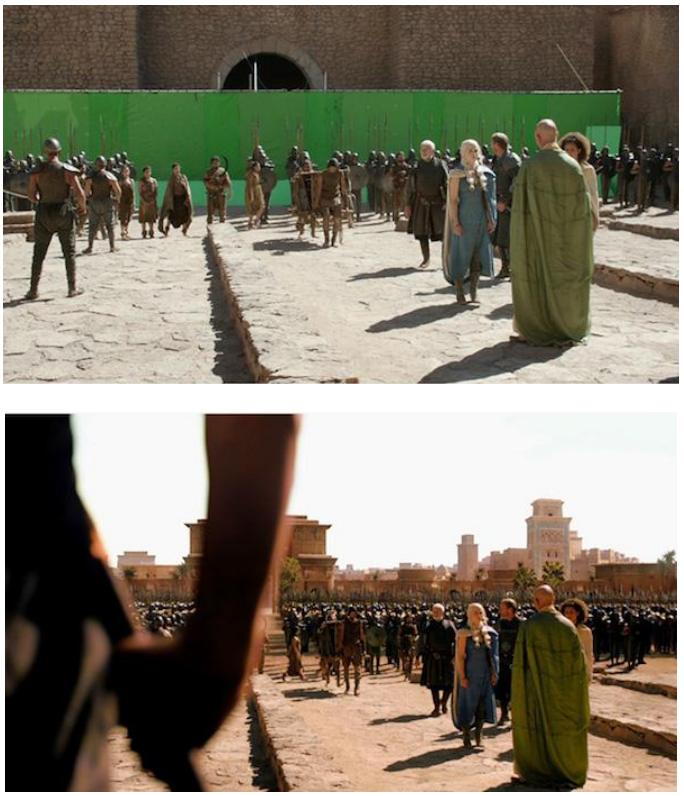

Fig. 24 La tecnica del set-extension utilizzata per "Game of Thrones". - Immagine presa dal sito "Amanda Fullwood" [18]

In questa sezione è stato quindi brevemente analizzato l'approccio non *real-time* del *compositing*: dalla pre-produzione alla post, ogni reparto è coinvolto al fine di produrre il miglior risultato, che deve essere il più fotorealistico possibile per soddisfare gli standard cinematografici.
Questo prevede inevitabilmente lunghi tempi di ripresa, elaborazione dati e *rendering* e non risulta, quindi, performabile in tempo reale.

1.2.3 Approccio real-time

Il *compositing* eseguito in tempo reale è già utilizzato nell'industria cinematografica e permette di ottenere *live* un'approssimazione di ciò che sarà il risultato finale.
Si tratta, quindi, di un *tool* estremamente utile per i cineasti che fanno uso di *VFX*: il girato *live-action* viene visualizzato sui classici video-monitor con gli effetti sovraimposti in modo sincronizzato, risultando di grande aiuto al reparto di fotografia, che si occupa delle riprese. [10]
Risulta importante non confondere questa tecnica con le *virtual cameras* utilizzate nella *virtual production*, in quanto queste ultime si focalizzano nell'inserire elementi *live action* all'interno di ambienti virtuali.
I vantaggi del *compositing real-time* sono innumerevoli, in quanto la possibilità di avere una *preview* degli effetti visivi direttamente sul set può aiutare anche il regista nel scegliere l'inquadratura migliore o *il blocking* dei personaggi.
I dati raccolti da questo *tracking real-time* contengono informazioni importanti, come i movimenti della camera sui tre assi, le rotazioni, gli zoom e i cambi di fuoco: il movimento della camera può essere quindi ricostruito in un ambiente CG e poi utilizzato per il *compositing* finale.
Si tratta, come già detto, di una tecnologia in continuo sviluppo, che attualmente non consente una simulazione fotorealistica degli effetti particellari o dei personaggi CG, ma promette di essere un tool sempre più presente sui set cinematografici che fanno uso di *CGI.*

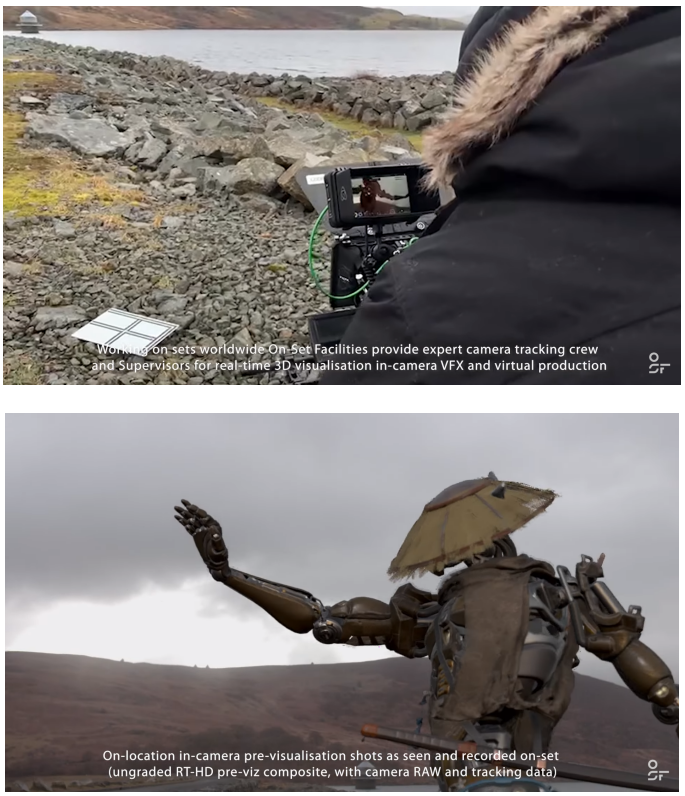

Fig. 25 Il real-time camera tracking: l'operatore può già visualizzare nel monitor il modello 3D animato e contestualizzato nello spazio live action. - Immagine presa dal canale YT dell'agenzia "On-Set Facilities" [19]

Terminato l'excursus sulle applicazioni cinematografiche, un altro esempio, più vicino all'argomento di questa tesi, sono le applicazioni in realtà aumentata.

La realtà aumentata, come noto, prevede che a un *background live action* siano sovraimposti in tempo reale oggetti sintetici: si tratta di una tecnologia in forte espansione che vedrà, con ogni probabilità, un picco di *hype* con l'uscita dell'Apple Vision Pro nel 2024 [20].

L'*AR* deve rispettare alcuni requisiti tecnici fondamentali affinché risulti verosimile all'occhio umano: deve riconoscere correttamente la superficie del terreno o, più in generale, comprendere la topologia dell'ambiente e stimare correttamente l'illuminazione affinchè l'oggetto sintetico risulti efficacemente compositato.

Vi sono numerose applicazioni *AR* che hanno avuto successo commerciale, una fra tutte *Pokemon Go*, ed è sempre più facile per gli sviluppatori trovare *tools* che permettano uno sviluppo intuitivo e integrato di applicazioni in realtà aumentata.

Fig. 26 La realtà aumentata in Pokemon Go, gioco mobile pubblicato da Niantic nel 2016.

Dal momento che questo progetto di tesi ha visto la sua applicazione sul motore di gioco *Unity* [21], si ritiene interessante parlare brevemente di *ARCore*, una piattaforma di Google che permette la creazione di esperienze in realtà aumentata, che è appunto supportata sui *game engine* più popolari come *Unreal Engine* e, appunto, *Unity.*

ARCore utilizza la camera del telefono Android per identificare punti di interesse, *features*, nell'ambiente reale e tracciare come questi punti si muovono nel tempo: unendo questo contributo ai sensori integrati dello smartphone è in grado di determinare sia l'orientamento che la posizione del telefono nello spazio [22].

In questo modo, la camera virtuale che renderizza i contenuti 3D si allinea con quella fisica del dispositivo *mobile*, consentendo una sovrapposizione verosimile del contenuto sintetico con lo sfondo reale.

Può inoltre riconoscere le superfici piane, come un tavolo o il pavimento, cercando *clusters* di *features* che sembrano giacere sullo stessa superficie orizzontale o verticale: *ARCore* genera quindi dei piani geometrici che possono essere usati come base per apporre oggetti virtuali.

L’ultima caratteristica che rende questa tecnologia interessante e particolarmente inerente per i nostri scopi è la *light estimation*, ovvero la stima della posizione e intensità delle sorgenti di luce presenti nell’ambiente reale.
ARCore fa uso della *Lighting Estimation API* per ottenere informazioni dettagliate che permettono di imitare in modo efficace numerose caratteristiche che gli oggetti illuminati da una sorgente posseggono, ovvero la presenza di ombre direzionalmente coerenti e di illuminazione tramite luce ambientale, lo *shading* (l’intensità della luce), gli *specular highlights* (porzioni di superficie che riflettono la sorgente di luce), le riflessioni (in quanto una superficie può essere speculare o diffusa).
La stima dell’illuminazione può venire svolta su *Unity* tramite due modalità, la *Ambient Intensity Mode* e la *Environmental HDR Mode*.

- *Ambient Intensity Mode*: per una data immagine, determina l’intensità media dei pixel e gli scalari di correzione colore.
  Si tratta di un *setup* “grezzo”, pensato per quando un’illuminazione precisa non è critica, ad esempio se gli oggetti hanno un’illuminazione *baked-in*.
- *Environmental HDR Mode*: utilizza il *machine learning* per analizzare le immagini della camera in *real-time* e creare l’illuminazione virtuale.
  Questa modalità fornisce:
    - La luce direzionale principale, utilizzata per proiettare le ombre in modo coerente (Fig. 27).
    - Le armoniche sferiche ambientali, che contribuiscono in modo più sottile all’illuminazione, come mostrato in Fig. 28
    - Una *cubemap HDR*, che può essere utilizzata per le riflessioni di oggetti metallici.

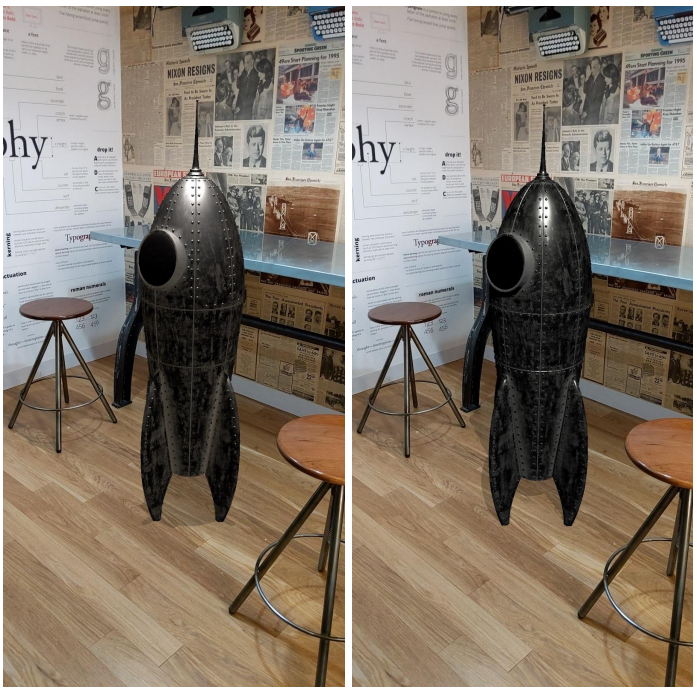

Fig. 27 A sinistra, un modello con un'ombra che non corrisponde a quella della scena reale; a destra l'ombra è correttamente posizionata. - Immagine presa dal sito di ARCore [22]

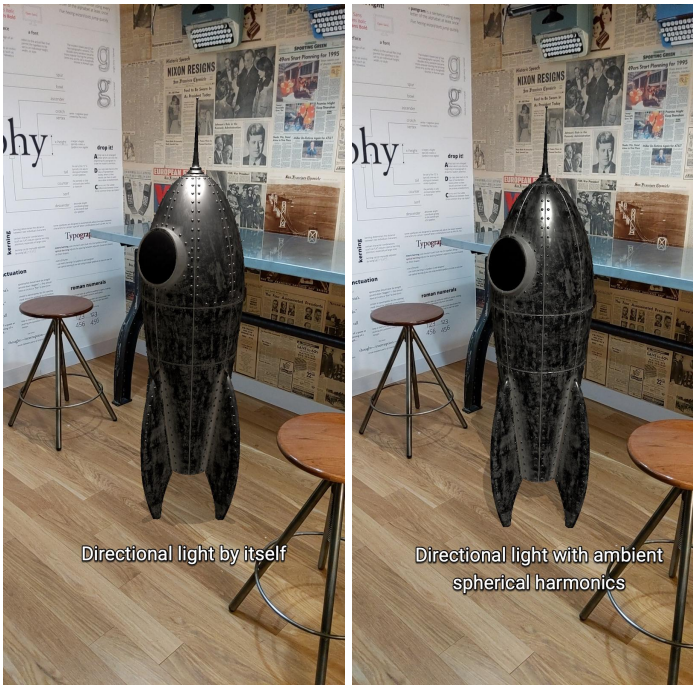

Fig. 28 A destra, il contributo delle ambient spherical harmonics. - Immagine presa dal sito di ARCore.

ARCore è solo un esempio di *tool* per sviluppatori che volessero cimentarsi nella realtà aumentata: esistono altre realtà come *ARKit*, destinato per le piattaforme iOS, ma anche *Vuforia* e *Spark AR*, strumenti relativamente semplici da imparare ma funzionali.

Si conclude quindi questo excursus sulle possibilità del compositing nell'industria cinematografica e nelle applicazioni real-time.
I principi spiegati in queste pagine torneranno utili quando, nel Capitolo 3, verranno ripresi per giustificare alcune scelte implementative.

1.3 Video 360°

Il video 360° è una tecnologia relativamente recente, che ha visto uno sviluppo importante a partire dal 2015, quando Youtube ha annunciato che avrebbe supportato questo formato all'interno della sua applicazione *mobile* [23].
A settembre dello stesso anno Facebook ha iniziato a sua volta a supportare i video 360° [24], mentre Vimeo nel 2017 [25].

Ma in cosa consiste un video 360°?
In questa porzione del capitolo andremo a introdurre alcuni concetti indispensabili per la comprensione dei tecnicismi di questa tesi, analizzando la tecnologia alla base ma anche le implicazioni e le limitazioni di questo mezzo.

1.3.1 Definizione e tecnologie impiegate

Un video 360°, come suggerisce il nome, è un flusso video che inquadra contemporaneamente l'intera area in cui è situata la videocamera.
Può essere visionato sia su un classico *display* 2D, dove l'utente ha la possibilità di controllare il punto di vista "manualmente", che su un *HMD*, rendendo l'esperienza estremamente immersiva in quanto lo spettatore orienta la camera virtuale con il naturale movimento della propria testa.
Le tecnologie *HMD* hanno dei "gradi di libertà" (*Degree of Freedom, DoF)*, che rappresentano su quanti assi è possibile orientare la visuale e se questa prevede anche delle "traslazioni".

Ci interessiamo essenzialmente a due tipologie [27]:

- 3-DoF: la visuale può soltanto essere ruotata sui 3 assi, non è previsto anche lo spostamento.
- 6-DoF: la visuale può sia ruotare che muoversi nello spazio.

Il video 360° è per definizione 3-DoF, in quanto uno spostamento su uno degli assi porterebbe il punto di vista lontano dal centro della sfera immaginaria di proiezione, "rompendo" la visualizzazione.

Le videocamere utilizzate per la produzione di questi contenuti dispongono di due o più lenti, uguali per caratteristiche fra di loro, che forniscono singoli contributi che vengono poi "uniti", in un processo chiamato *stitching* [26], per formare l'inquadratura definitiva.
Le tipologie di queste camere presenti sul mercato sono essenzialmente due [27]:

- A due lenti (*Dual Fisheye*): compatte e accessibili a livello di budget, presentano soltanto due lenti grandangolari con campo visivo intorno a 200°.
  A causa del limitato numero di sensori, i bordi del girato risultano poco nitidi e le aberrazioni ottiche sono evidenti, ma il processo di allineamento dei due contributi video risulta più semplice.
  Due esempi commerciali possono essere la *Insta360 One X2* o la *GoPro Max*.
- Con più di due lenti: in questo caso le lenti presenti sulla camera possono essere 4, 6, 16 o addirittura 36.
  Aumentando il numero di sensori, cresce ovviamente la risoluzione video e, inevitabilmente, anche il costo.
  La fase di *stitching* risulta quindi più laboriosa, ma sensibilmente più precisa se ben performata.
  Un esempio commerciale, che tra l'altro è stato impiegato per le ricerche di questa tesi, è la *Insta360 Pro*.

Fig. 29 Da sinistra a destra, la Insta360 One X2, la GoPro Max e la Insta360 Pro.

Se gli esempi forniti qui sopra si possono contestualizzare nella categoria *all-in-one*, in quanto le lenti fanno parte dello stesso compatto corpo macchina, esiste anche un approccio più complesso, ingombrante e assai costoso, che va però a massimizzare la qualità del prodotto finale: si tratta del *rig* multi-camera.
L'azienda *GoPro* ha commercializzato *Omni* [28], un *rig* che accoglie 6 singole camere *GoPro*, le sincronizza e fornisce un *software* per lo *stitching* manuale.
Nonostante la maggiore complessità, si tratta comunque di un prodotto compatto, seppur estremamente costoso (prevede di dover acquistare separatamente 6 camere *GoPro* per funzionare).

Fig. 30 Omni prodotto da GoPro.

Il processo di *stitching* sopracitato può avvenire sia in tempo reale che in fase di post-produzione: nel caso di un *live-stream* come quello impiegato in questa tesi, viene svolto in automatico, producendo buoni ma, ovviamente, non eccelsi risultati.

1.3.2 Tipologie di proiezioni

Dal momento che i video 360 sono immagini in movimento mappate su una sfera, è necessario escogitare un modo per proiettare il contributo "piatto" registrato dai sensori delle camere su una superficie che piatta non è.
Esistono numerosi tipi di proiezione di video 360° ma, per semplicità, tratteremo delle due più utilizzate, in quanto verranno citate successivamente all'interno di questo elaborato.

Equirettangolare

Si tratta di una tipologia di proiezione antichissima, risalente al 100 d.c [29], ed è la più popolare quando si tratta di video 360°.
La trasformazione consiste in una proiezione della superficie di una sfera su quella di un cilindro, e quindi mappa i meridiani su linee verticali equispaziate e i paralleli su linee rette orizzontali disposte a intervalli uguali.
I poli risultano quindi rappresentati grandi come l'equatore, causando una notevole distorsione se ci si avvicina agli estremi superiori e inferiori dell'immagine.

Le formule che vengono utilizzate per trasformare le coordinate sferiche in cartesiane sono [28]:

$$x = R(\lambda - \lambda_0)cos\varphi_1$$

$$y = R(\varphi - \varphi_0)$$

Con:

- λ la longitudine del punto da proiettare.
- φ la latitudine del punto da proiettare.
- φ_1 i paralleli standard dove la scala della proiezione è corretta.
- φ_0 il parallelo centrale.
- λ_0 il meridiano centrale.
- x la coordinata orizzontale del punto proiettato.
- y la coordinata verticale del punto proiettato.
- R il raggio della sfera.

La formula inversa è:

$$\lambda = \frac{x}{R\cos\varphi_1} + \lambda_0$$

$$\varphi = \frac{y}{R} + \varphi_0$$

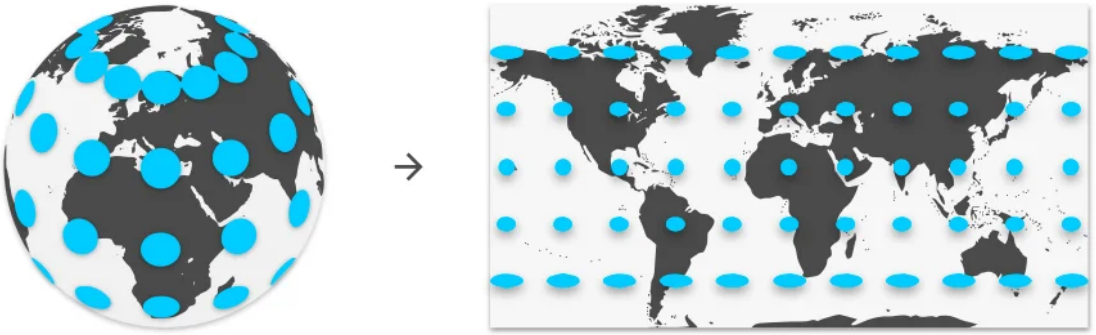

Fig. 31 La proiezione equirettangolare con indicate le evidenti distorsioni ai poli. - Immagine presa da blog.google [30]

Fig. 32 Un frame estratto da un video 360° con proiezione equirettangolare: da notare l'assenza di distorsioni al centro dell'inquadratura.

La convenzione cartografica prevede che vi siano 360 meridiani e 180 paralleli, che si trovano quindi in un rapporto 2:1: i video 360° vengono quindi *stitchati* in equirettangolare con un *aspect ratio* proprio di 2:1.

Cubemap

Si tratta di una proiezione frequentemente utilizzata nell'industria del *gaming*, che prevede di mappare l'ambiente sulle sei facce di un cubo.
Il vantaggio principale è piuttosto evidente: viene ridotta notevolmente la distorsione che per natura è presente nella proiezione *equirettangolare*.
Si ottiene con una semplice proiezione radiale, inserendo la sfera in un cubo e proiettando l'immagine presente sulla sfera sulla superficie del cubo stesso, come si può vedere in Fig. 33.

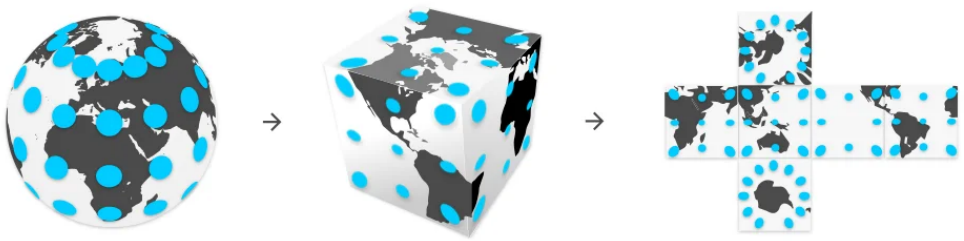

Fig. 33 La proiezione cubemap illustrata. - Immagine presa da blog.google

Questo approccio migliora il rapporto della densità dei *pixel*, ovvero quanti *pixel* vengono destinati alla stessa superficie dell'immagine: come visibile confrontando la Fig. 31 con la 33, nella equirettangolare vengono destinati tantissimi *pixel* per le aree dei poli, "sprecandoli" [30].
Tuttavia, vista la facilità di visualizzazione e manipolazione del formato equirettangolare, questo è sempre stato preferito alla *cubemap*, ed è il più utilizzato: anche in questo progetto di tesi, il flusso video è sempre stato manipolato nel formato equirettangolare.

1.3.3 360 Monoscopico e Stereoscopico

Alcune camere 360° presenti sul mercato consentono di registrare e visualizzare i video in formato stereoscopico (3D), favorendo un maggior senso di immersione in quanto entra in gioco la percezione della profondità.
Come sappiamo gli occhi umani forniscono al cervello due contributi bidimensionali, che combinati assieme formano una visione stereoscopica, la quale contiene

informazioni sulla tridimensionalità del mondo, grazie a una serie di indizi chiamati *depth cues*.

I *depth cues* sono [10]:

- Occlusioni: lungo la linea della vista, un oggetto vicino occlude quello lontano.
- Altezza relativa: oggetti simili ma a distanza differente appariranno più piccoli o più grandi.
- Ombre e ombreggiature: forniscono informazioni sull'orientamento rispetto alla luce del sole.
- Parallasse: oggetti vicini sembrano muoversi più velocemente di quelli distanti
- Prospettiva atmosferica: gli oggetti distanti perdono di definizione in quando è presente l'atmosfera.

Queste *cues* esistono già nell'immagine 2D ma, se uniti al contributo stereoscopico degli occhi, permettono di percepire la profondità in un'immagine.

Analizzando più nel dettaglio il funzionamento della visione umana, descriviamo brevemente i concetti di accomodazione e convergenza.

- Convergenza: la rotazione degli occhi quando essi guardano verso un oggetto.
- Accomodazione: il processo che porta il cristallino dell'occhio a contrarsi per focalizzarsi sull'oggetto.

Questi due processi lavorano assieme ma appartengono a differenti sistemi muscolari [10].

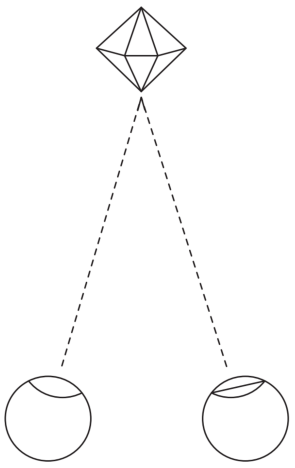

Fig. 34 - La convergenza. - Immagine di Jenny Lipton

L’ultimo concetto da definire è quello di “parallasse”, che è la differenza orizzontale fra le due immagini che gli occhi captano e dipende dalla *IPD* (distanza interpupillare): è proprio la parallasse a creare, come anticipato, il senso di tridimensionalità ed è il processo che le camere 360° imitano per le riprese in 3D.
L’*output* video stereoscopico non è più quindi soltanto un video equirettangolare, ma l’unione di due “strisce” equirettangolari orizzontali, che vengono proiettate singolarmente sugli occhi e producono l’effetto desiderato.

Fig. 35 Un frame di un video 360 stereoscopico: notare come le prospettive siano leggermente sfalsate per la parallasse.

Nonostante la *Insta360 Pro* consenta di *streammare* in stereoscopia, nell’implementazione di questo progetto di tesi è stato utilizzato un flusso *livestream* monoscopico, in quanto l’*autostitching* stereoscopico presentava delle limitazioni nelle aree di sovrapposizione delle lenti: vedremo meglio questa implicazione nella Sezione 3.4 del Capitolo 3.

Si conclude questa lunga introduzione, che ha toccato numerosi concetti chiave che risulteranno utili nei capitoli successivi.
Ora analizzeremo i sistemi già presenti in letteratura che hanno utilizzato un video 360° per estrapolare informazioni e sfruttarle in modo dinamico, dal punto di vista del *compositing* e del *body tracking*.

2. Stato dell'arte

In letteratura non è attualmente presente un sistema che utilizzi un *livestream 360°* nel modo che è approcciato all'interno di questo progetto di tesi, ma vi sono alcune ricerche e studi che si sono "avvicinati" all'argomento da vicino.
Questo capitolo è diviso in due parti, le prima dedicata ai sistemi che hanno studiato le possibilità del *compositing* su un flusso video a 360°, la seconda dedicata invece alle metodologie di tracciamento di corpi in un video immersivo.

2.1 Sistemi per *compositing* a 360° in VR

Come anticipato, verranno ora analizzati i due sistemi trovati in letteratura che hanno studiato e applicato tecniche innovative per il *compositing* di oggetti sintetici su un video 360°.

2.1.1 MR360: Mixed Reality Rendering for 360° Panoramic Videos

MR360 [31] è stato presentato ufficialmente in un *paper* pubblicato nel gennaio del 2017, ed è un sistema che ha molto influenzato le primissime fasi di questa ricerca.
Implementato su *Unreal Engine*, si tratta di un sistema immersivo che utilizza un flusso video 360 convenzionale *LDR* (*low dynamic range*), ovvero dotato di una gamma dinamica bassa, per illuminare e compositare coerentemente oggetti sintetici nell'ambiente 360°.
Per fare ciò, viene applicato il principio dell'*image based lighting* [32], che sfrutta una mappa di radianza *HDR* (*high dynamic range*) per illuminare gli oggetti presenti in scena: dal momento che il flusso video 360 è *LDR*, e non sono previste fasi di precomputazione in quanto il sistema si prefigge l'idea di poter supportare un *livestream*, è stato necessario utilizzare un processo chiamato *inverse tone mapping*, per "elevare" la gamma dinamica e ottenere mappe di radianza più precise.
Ottenuta una mappa di radianza *HDR*, è necessario sottocampionarla per renderla fruibile in tempo reale: grazie agli studi condotti dagli stessi autori [33], una risoluzione di 64x32 o addirittura 32x16 *pixel* risulta percettivamente indistinguibile da una di 256x128 (come si vede in Fig. 36).

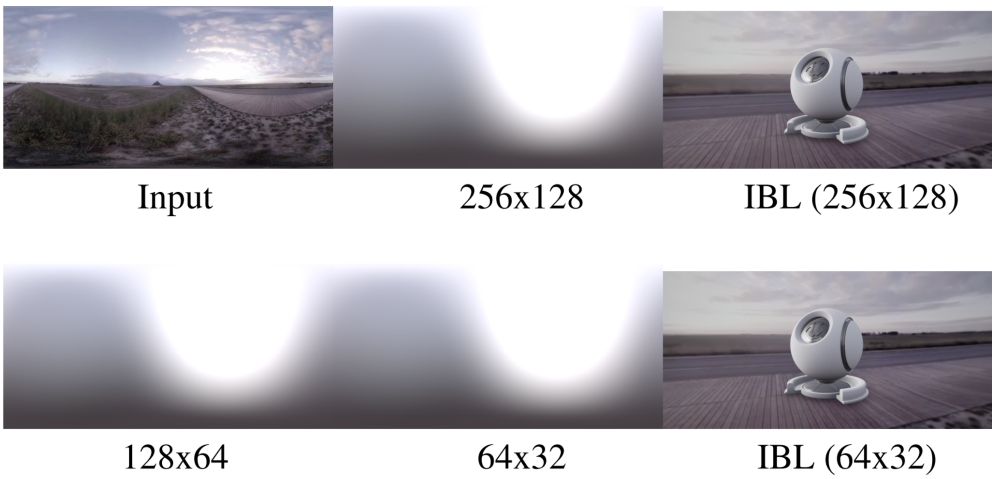

Fig. 36 Le radiance maps a differenti risoluzioni. - Immagine presa dal paper in questione [31]

Avviene quindi il processo di *light detection*, per riconoscere le sorgenti di luce all'interno del video equirettangolare e poter poi proiettare ombre in modo coerente, attraverso l'*IBS* (*image-based shadowing*): la sola *IBL* ha infatti delle limitazioni per quanto riguarda questo aspetto.

Partendo dalla mappa di radianza, viene confrontato il valore di luminanza di ogni *pixel* con una soglia prestabilita e, se viene superata, il *pixel* viene considerato come parte di una sorgente di luce.

Dal momento che, in scenari realistici, le sorgenti di luce sono situate nella parte superiore dell'orizzonte (sul soffitto nel caso di lampade, nel cielo nel caso del sole), viene scartata la metà inferiore del *frame*, che potrebbe presentare riflessioni provenienti dal terreno erronee.

Come visibile nella seconda riga della Fig. 37, viene creato un mascherino, il quale viene processato con un algoritmo di ricerca volto a produrre un *set* limitato di luci direzionali.

Per ogni luce rilevata viene calcolato il suo valore di irradianza, sommando i singoli contributi dei *pixel.*

Infine, viene calcolata la posizione della luce in coordinate sferiche.

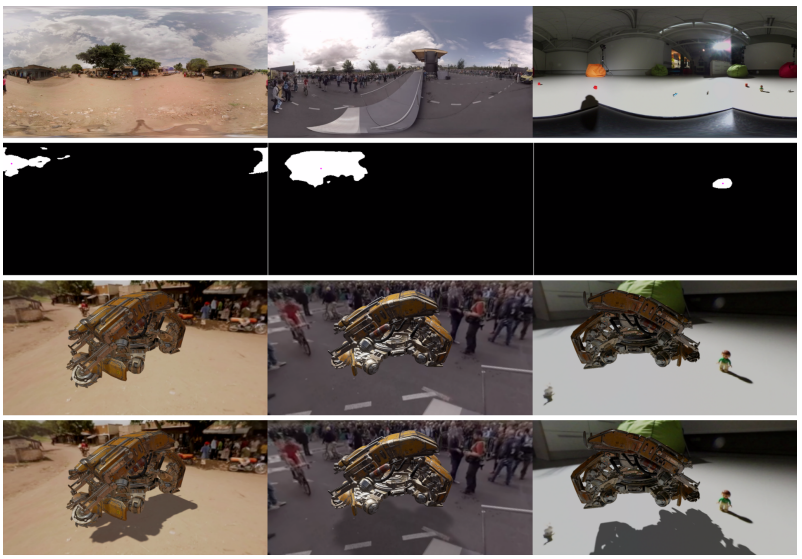

Fig.37 Il procedimento di Light Detection. Dall'alto al basso: il frame di input, il mascherino di pixel che superano la soglia, il render usando solo IBL, il render usando anche la IBS. - Immagine presa dal paper in questione

Viene poi utilizzata la tecnica del *differential rendering* [34] per unire i vari contributi al fine di creare l'immagine compositata.

Il procedimento prevede di renderizzare la *local scene*, una geometria che rappresenta la superficie di *compositing*, una volta con gli oggetti sintetici e una volta senza.

Facendo la differenza di *pixel* fra i due *renderings* si ottiene l'influenza degli oggetti sulla *local scene*, come le ombre e le riflessioni.

In Fig. 38 viene illustrato l'intero procedimento.

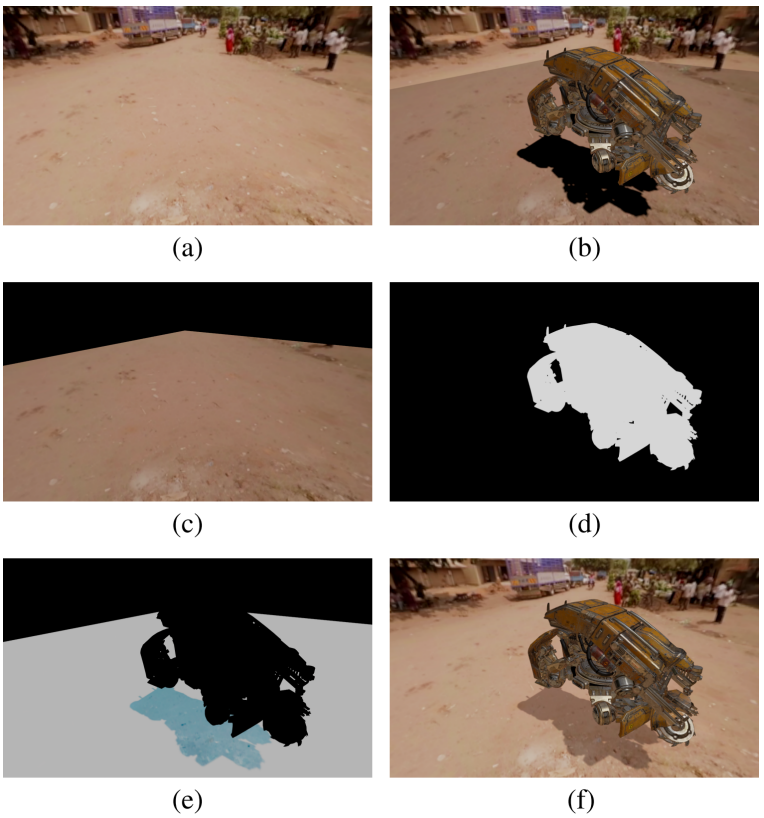

Fig. 38 dall'alto al basso, da sinistra a destra: il video 360° di sfondo, lo sfondo + la *local scene* + gli oggetti sintetici, la *local scene*, il mascherino generato dagli oggetti sintetici, il risultato della formula *b * (1-d) - c*, il risultato finale.

Nella sezione 8, dedicata alle conclusioni, viene fatta una considerazione importante per quanto riguarda l'implementazione su *Unreal Engine*.

Come interfaccia *HMD* è stato utilizzato un *HTC Vive*, che supporta solamente il *rendering* stereoscopico in caso di utilizzo di *controller Vive*: nasce un importante problema di prospettiva (*parallax mismatch*), in quanto gli oggetti virtuali sono renderizzati in 3D, mentre il video 360° è monoscopico, e quindi questi oggetti sembrano "fluttuare" sul *background* e non correttamente compositati.

Per risolvere questa problematica, gli oggetti virtuali sono stati allontanati dall'utente il più possibile, per ridurre al minimo la parallasse.

Si tratta di un'implicazione importantissima, in quanto l'effetto rischia di azzerare l'illusione di un *compositing* verosimile: questo problema è stato affrontato e risolto durante le mie sperimentazioni, come vedremo nel Capitolo 3.

Questa ricerca è stata la base da cui è partito questo progetto di tesi, sia per gli spunti riguardanti l'algoritmo di *light detection* che per l'aiuto diretto fornito da uno

degli autori, Andrew Chalmers, che ha gentilmente risposto ad alcuni miei quesiti (fra i quali quello sopracitato del *parallax mismatch*) tramite mail durante le fasi iniziali dei miei studi sull'argomento.
Accenneremo a questi quesiti, come anticipato, nel Capitolo 3.

2.1.2 Real-time Virtual Object Insertion for Moving 360° Videos

Questo *paper* [35] è stato pubblicato nel novembre del 2019 e espande quanto studiato e applicato dal *team* di *MR360*.
Vengono, infatti, applicati i principi di *IBL, tone mapping*, *light estimation* e *IBS* citati nel *paper* antecedente, seppur vengano performati in una fase di *pre-process*, che non rende il *compositing* realmente in tempo reale.
Viene, inoltre, implementato il *camera tracking* per supportare video omnidirezionali in movimento, attraverso una *pipeline structure-from-motion 360°* innovativa: viene quindi ribadito che il processo non può essere eseguito in *real-time*.

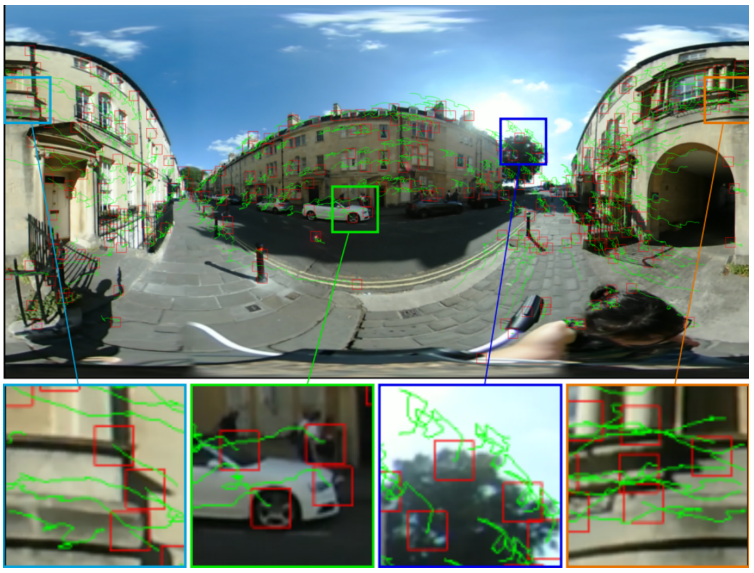

Fig. 39 Il feature tracking 360° implementato in pre-process. - Immagine presa dal paper in questione [35]

Dal momento che nell'implementazione scelta per il progetto di tesi la camera 360° deve restare ferma, si ritiene non sia particolarmente interessante dilungarsi su questo aspetto.

Risulta invece importante spiegare la modalità di applicazione del modello di *illumination estimation*, in quanto implementato sul *game engine Unity*.
Vengono infatti utilizzate delle *environment maps*, ovvero *light probes*, equispaziate lungo il percorso della camera 360° in movimento, in quanto di poca complessità: l'intero sistema di illuminazione si basa su questo principio.
Le *environment maps* vengono facilmente ottenute dai *frames* del video 360°, e convertite in *HDR* col procedimento dell'*inverse tone mapping*.

Fig. 40 A sinistra, un light probe con LDR; a destra, una versione HDR, che si nota essere meglio esposta. - Immagine presa dal paper in questione

Per le scene in esterna, viene applicata manualmente una *light source* per rappresentare il sole, al fine di generare le ombre: sembra, tuttavia, che non siano state fatte prove *indoor*, quindi non è chiaro in che modo si ottenga l'ombreggiatura in questo caso, soprattutto in presenza di multiple sorgenti di luce.

Fig. 41 Le scene di dimostrazione: il risultato è efficace, ma è pre-computato e la sorgente di luce rappresentante il sole è posizionata manualmente. - Immagine presa dal paper in questione

Il *compositing* finale viene poi performato con il *differential rendering*, in modo concettualmente identico all'applicazione *MR360*.

Si tratta di una ricerca sicuramente interessante, ma non particolarmente calzante per le specifiche tecniche definite da questo progetto di tesi, ovvero la necessità di lavorare in *livestream* e con camera fissa.
E' stato comunque utile il suggerimento di utilizzare le *probes*, implementate come spiegato nel Capitolo 3.

2.2 Body Tracking in Video 360°

Per aumentare le possibilità espressive e narrative dei progetti *Presence* che faranno utilizzo di questo sistema, si è voluto esplorare la possibilità di integrare una qualche forma di *body tracking* sul video 360°, sfruttando poi le informazioni raccolte in modo interattivo.
Vi sono due ricerche che hanno trattato di quest'ambito, evidenziando potenziali problematiche e proponendo soluzioni.
Andiamo a parlarne.

2.2.1 Real-time object detection in 360-degree videos

In questo *paper* [36], pubblicato nel 2021, viene analizzato il processo di *object detection* su un video 360° proiettato in formato equirettangolare.
La motivazione di fondo di questa ricerca nasce da una riflessione sulla larghezza di banda che i video 360° occupano se *streammati* e dalla necessità di una latenza ridotta.
Una soluzione proposta per ridurre la pesantezza di questi flussi video *live* è di fornire al visualizzatore soltanto il *field of view* che sta venendo visionato in quel momento, che è circa il 20% dell'area totale del frame: vi è tuttavia un problema di latenza fra il *server* e lo spettatore, che rende impossibile al *server* il sapere in tempo reale le informazioni sul *FoV* visualizzato dall'utente.
Vengono citati alcuni studi che hanno suggerito di predire il *FoV* dell'utente facendo uso della regressione lineare utilizzando dati sul comportamento passato dello spettatore [37]: questi tuttavia non sono utili se il *network delay* è maggiore dell'orizzonte di previsione.
Può quindi diventare utile utilizzare algoritmi di *object detection*, in quanto l'attenzione dell'utente è spesso influenzata dalla presenza di oggetti in scena, che tendono ad essere osservati per lunghi periodi di tempo.
Questi algoritmi nascono per rilevare oggetti su video "convenzionali", e utilizzano quindi griglie rettangolari.
Come abbiamo visto nel Capitolo 1, Sezione 1.3.2, la maggior parte dei contenuti 360° è mappato il proiezione equirettangolare, fortemente distorta ai poli (fare riferimento alle Fig. 31 e 42): il rilevamento di oggetti nelle aree deformate risulta molto più difficile, poiché il modello di *machine learning* è allenato, come già detto, su video "convenzionali".

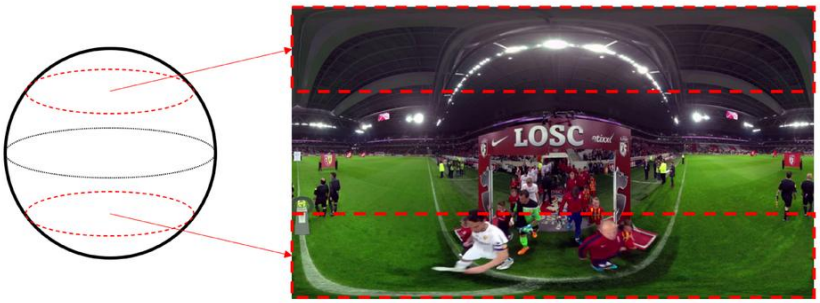

Fig. 42 La porzione centrale non presenta distorsioni, a differenza delle due aree in rosso. - Immagine presa dal paper in questione [36]

Fig. 43 Un esempio ottenuto con un algoritmo di object detection basato su YOLO [38]: una persona, indicata in blu, non è stata rilevata. - Immagine presa dal paper in questione

Viene quindi portato un confronto con la proiezione *cubemap* che, come spiegato nel Capitolo 1, Sezione 1.3.2, presenta un quantitativo di distorsione assai minore, ma anche delle discontinuità fra le sei facce del cubo, rendendo il riconoscimento di oggetti più difficoltoso.
Viene quindi introdotta una tecnica di *multi-directional projection* (*MDP*), che consiste nel generare molteplici proiezioni equirettangolari con diversi angoli di proiezione.
Il processo prevede tre *steps*:

1. Proiezione dell'immagine equirettangolare in formato *cubemap*.
2. Rotazione della *cubemap* in differenti direzioni.
3. Proiezione della *cubemap* nel formato equirettangolare originario.

In questo modo, gli oggetti vicino ai bordi del *frame* vengono spostati verso il centro, riducendo notevolmente la distorsione.

Fig. 44 A sinistra, i differenti angoli di proiezione; a destra, il frame composto dai vari contributi. - Immagine presa dal paper in questione

Le proiezioni con *Angle-1* e *Angle-2* vengono utilizzate per evitare discontinuità ai bordi destri o sinistri dell'immagine: se un oggetto si trova al limite sinistro del contributo *Angle-1*, risulterà al centro del contributo *Angle-2*.
I contributi *Angle-3* e *Angle-4* sono generati per spostare gli oggetti che originariamente si trovano ai poli verso il centro del *frame*, per mitigare le distorsioni.
Questo procedimento risulta efficace, ma anche più dispendioso in quanto devono essere processate quattro proiezioni equirettangolari per volta: il tempo impiegato resta comunque sotto 1 secondo, consentendo una *detection real-time*.

Questo *paper* è risultato particolarmente interessante in quanto ha confermato ciò che già era noto in letteratura: le porzioni centrali del *frame* equirettangolare non risentono di distorsioni e performano bene l'*object detection*.
Come vedremo nel Capitolo 3, questa implicazione è stata tenuta in considerazione per alcune scelte implementative.

2.2.2 A Dataset of Annotated Omnidirectional Videos for Distancing Applications

Questo *paper* [39] è stato pubblicato nel 2021 da un gruppo di ricercatori dell'Università degli Studi di Palermo.
Viene proposto un *dataset*, chiamato *CVIP360*, di video 360° *outdoor* e *indoor* che contengono informazioni sui pedoni inquadrati (*bounding boxes*) e sulle distanze di alcuni punti segnalati da *markers* e, *feature* interessante per i nostri scopi, viene

anche presentato un algoritmo per rilevare le distanze delle persone dalla camera 360°.
Questo progetto è stato concepito per applicazioni che possono utilizzare le informazioni sulla distanza degli oggetti, come quelle di videosorveglianza.
Come già visto nel Capitolo 1, Sezione 1.3.2, viene spiegato che la proiezione equirettangolare rappresenta l'equatore della sfera come riga che taglia orizzontalmente il *frame* e i poli come le righe superiori o inferiori, e presenta distorsioni quando ci si avvicina a questi ultimi (Fig. 31 e 42).
Il processi di *depth estimation* si può dividere essenzialmente in due categorie:

- Attivi: se utilizzano tecnologie come *Kinect* per costruire una mappa di profondità della scena. Sono molto accurati ma dispendiosi e non possono essere utilizzati in tutte le circostanze.
- Passivi: se si basano unicamente su informazioni visive fornite dal flusso video. Sono meno dispendiosi e hanno meno costrizioni, ma più inaccurati.
  Si possono a loro volta in due sotto-categorie:
  - *Stereo-based* [40]: se fanno uso di due camere che riprendono da due punti di vista differenti la scena.
  - *Monocular* [41]: se l'informazione di profondità è estratta da un'unica immagine.

I metodi già esistenti in letteratura utilizzano, nella maggior parte dei casi, tecniche *deep network*, le quali necessitano di *datasets* per la fase di *training*: ecco che viene quindi introdotto *CVIP360*, aspetto del *paper* che lasceremo da parte in quanto lontano dai nostri scopi.

Algoritmo di *depth estimation*

Risulta invece molto interessante il sistema impiegato per calcolare la distanza degli oggetti dalla camera.
Si parte da un presupposto chiave: tutti i punti presenti su una riga del frame sono equidistanti dalla camera nel mondo 3D, nel caso che la camera 360° sia stata correttamente posizionata "in bolla", ovvero l'asse sia parallelo al terreno (ipotesi di "orizzontalità").

Deviazioni di pochi gradi rispetto agli assi possono portare a shift delle coordinate equirettangolari degli oggetti di decine di pixel [39], come si può vedere in Fig. 45.
Queste distorsioni vengono corrette su *Adobe Premiere CC* con un *tool* apposito, che ovviamente non può essere utilizzato nel nostro caso *livestream*.

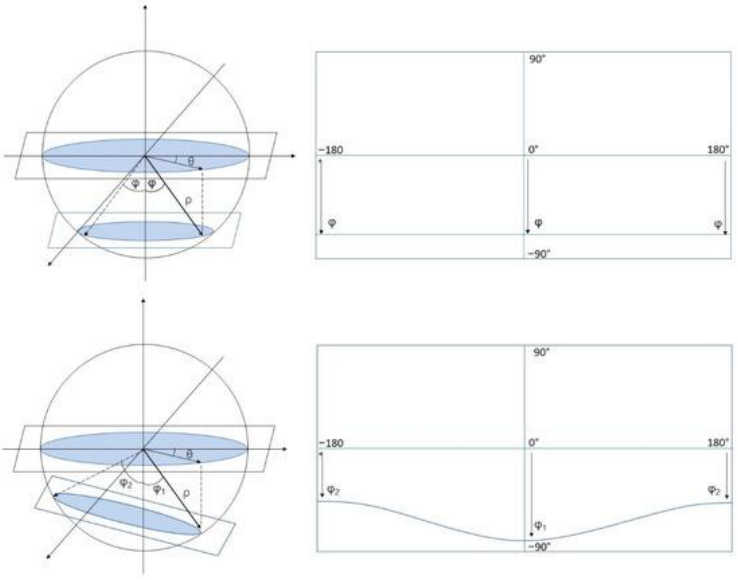

Fig. 45 Se viene a mancare l'ipotesi di "orizzontalità", punti alla stessa distanza nel mondo 3D verranno shiftati notevolmente in equirettangolare. - Immagine presa dal paper in questione [39]

La *feature* più utile per i nostri scopi è che l'unico parametro che serve conoscere per utilizzare questo algoritmo è l'altezza della camera dal terreno, che può essere ottenuta con una misurazione durante l'allestimento.
Data una persona presente in scena che sta toccando il terreno (ad esempio, una persona con i propri piedi), e supponendo che sussista l'ipotesi di "orizzontalità", la distanza *d* del punto dalla camera può essere stimata con:

$$d = h_c cot\alpha$$

Dove h_c è l'altezza nota della camera e α l'angolo fra la linea che congiunge il centro della camera e l'orizzonte con la linea che congiunge il centro della camera con il punto di contatto sul terreno dell'oggetto: si tratta di una semplice formula trigonometrica.

L’angolo *alfa* si calcola direttamente dai *frames* del video equirettangolare:

$$\alpha = \frac{y_L - \frac{h}{2}}{\frac{h}{2}} * 90°$$

Dove y_L è la coordinata y in *pixel* del punto inferiore dell’oggetto (in caso di una persona, il punto in cui il piede tocca il terreno) e h l’altezza in *pixel* dell’immagine equirettangolare.

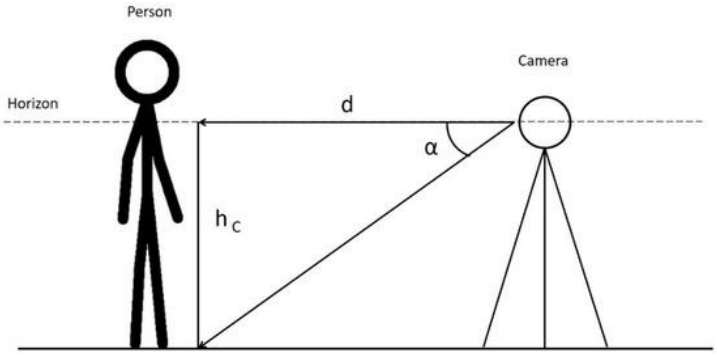

Fig. 46 Le variabili in gioco nel calcolo della distanza: d è semplicemente un cateto di un triangolo rettangolo. - Immagine presa dal paper in questione

Questo metodo presenta in media un errore di 5 cm per video *outdoor* e 2.5 cm per video *indoor*, che cresce leggermente allontanandosi dalla camera: si tratta comunque di percentuali trascurabili.
La soluzione qui presentata ben si presterebbe, a livello teorico, ad un’implementazione pratica in questo progetto di tesi, tuttavia il fatto che il flusso video sia *livestream* non consente di gestire in *post* la distorsione generata dalla proiezione equirettangolare e l’algoritmo di *object detection* utilizzato non è abbastanza stabile da consentire una misura precisa.
Maggiori dettagli nella sezione dedicata del Capitolo 3.

Si conclude il capitolo relativo allo stato dell’arte, dove sono stati analizzati alcuni studi e progetti che hanno fornito spunti interessanti per lo sviluppo di questo progetto.
Ora andremo a vedere, finalmente, il metodo utilizzato per il *compositing* di oggetti sintetici su un *livestream* a 360°

3. Metodo

In questo capitolo verrà trattato il metodo che è stato impiegato per ottenere il risultato finale, analizzando le problematiche sorte e poi risolte ma anche gli inevitabili limiti.
Il motore di gioco che è stato utilizzato è *Unity* e potranno capitare terminologie specifiche di questo *engine*, che verranno spiegate in caso non fossero immediatamente comprensibili.

3.1 Compositing 360°

Il primo passo compiuto nell'implementazione è stato rendere disponibile il video 360° sul *game engine*.

3.1.1 Acquisizione dei dati in *livestream*

Unity fornisce un *Video Player* che può essere utilizzato per ricevere un flusso video tramite *URL* o *file* importato: è possibile scegliere come *render target* una *render texture* creata appositamente, della stessa risoluzione del video 360°.
Creando un materiale di tipo *Skybox/Panoramic* e utilizzandolo come *Skybox* dell'ambiente, il video 360° verrà proiettato come se fosse, appunto, una *skybox*.
Si tratta di un approccio consigliato da *Unity* stesso [42], che però presenta alcune limitazioni, la prima delle quali è l'assenza di supporto per file *livestream*.

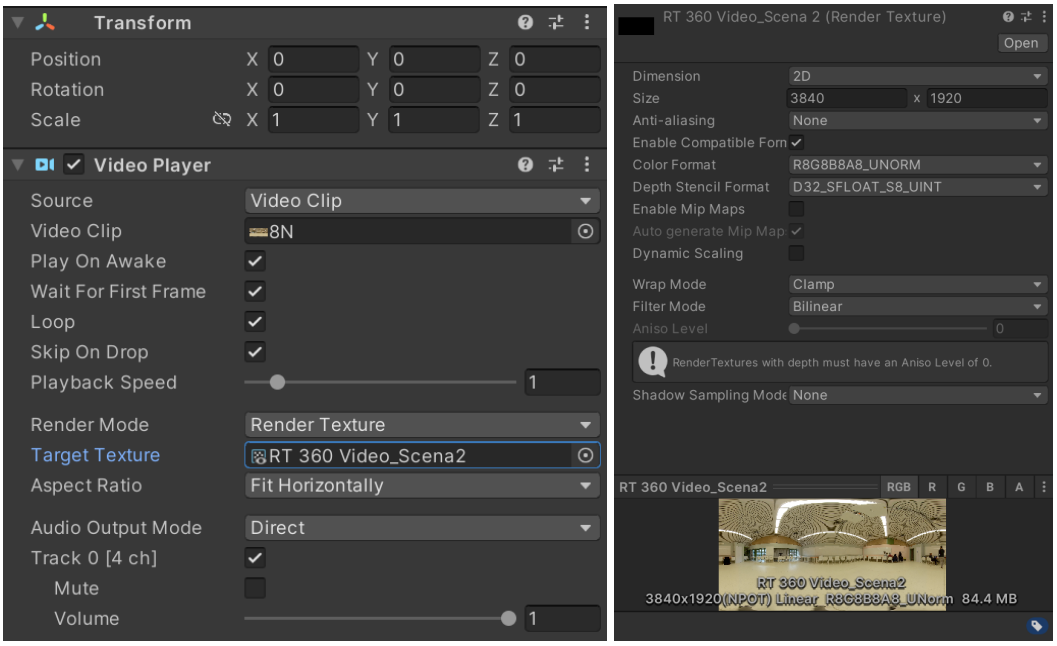

Fig. 47 Il Video Player e la Render Texture, con dimensioni settate in base alla risoluzione nota.

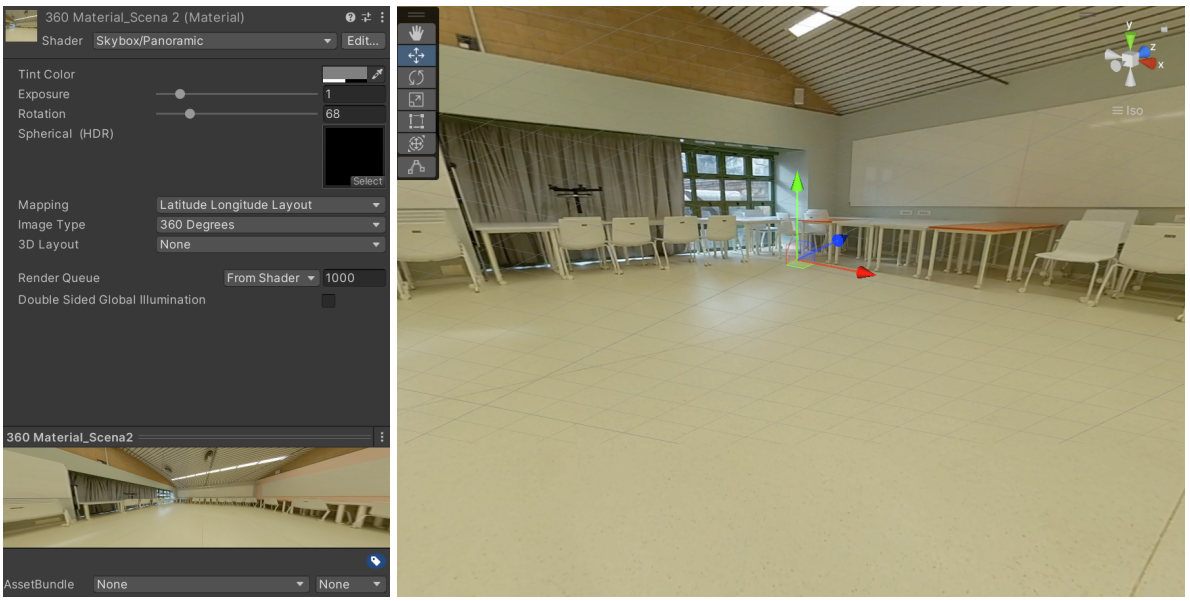

Fig. 48 Il materiale Skybox/Panoramic e l'esito finale, in cui il video 360° viene utilizzato come skybox.

L'approccio che è stato invece scelto, più versatile e adatto ai requisiti, è stato l'utilizzo della libreria *UMP* (*Universal Media Player)*.

Questa fornisce alcuni *tools* indispensabili per il *playback* di video immersivi, come lo script *UniversalMediaPlayer* (che consente anche di riprodurre flussi video ricevuti

da un *server RTMP*) e lo *shader UMP_Equirectangular* (che, applicato a un oggetto di forma sferica presente in scena, mappa il video immersivo al suo interno).
Il flusso video *livestream* trasmesso dalla *Insta360 Pro*, infatti, viaggia su un *server RTMP* che genera un *URL* fisso, il quale viene quindi inserito manualmente per ottenere il video.

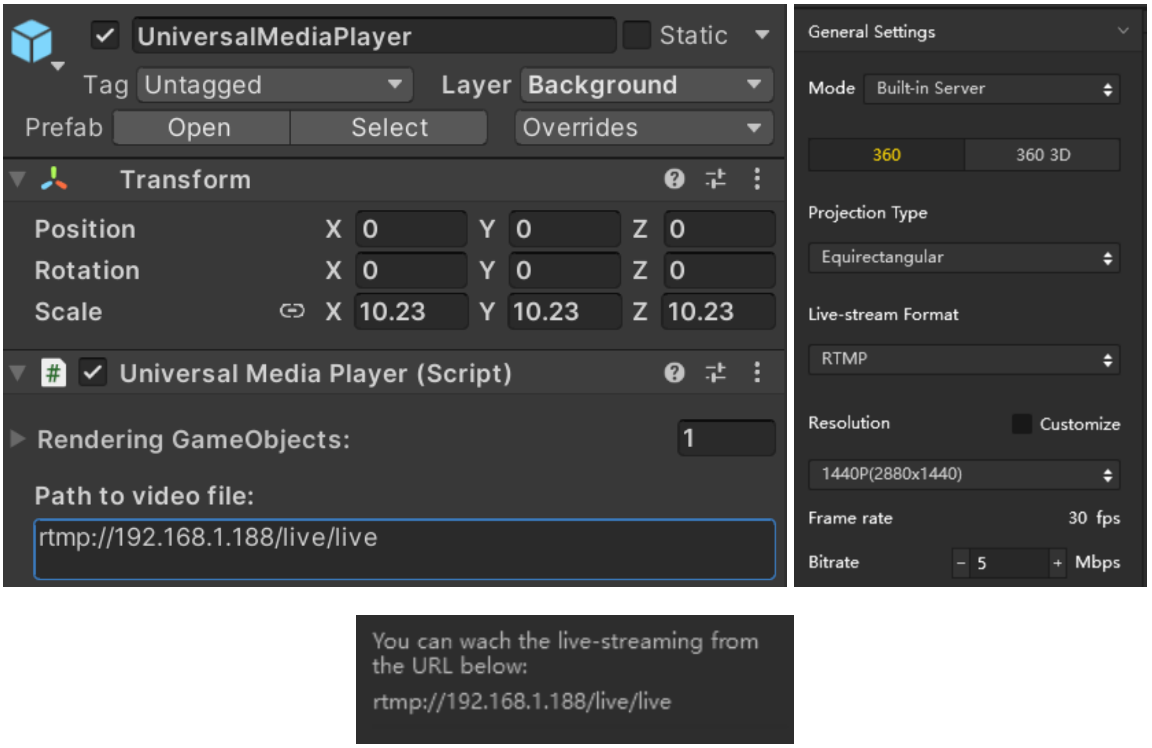

Fig. 49 Lo script UniversalMediaPlayer e i settings della Insta360 Pro, dove viene fornito l'URL da incollare su Unity.

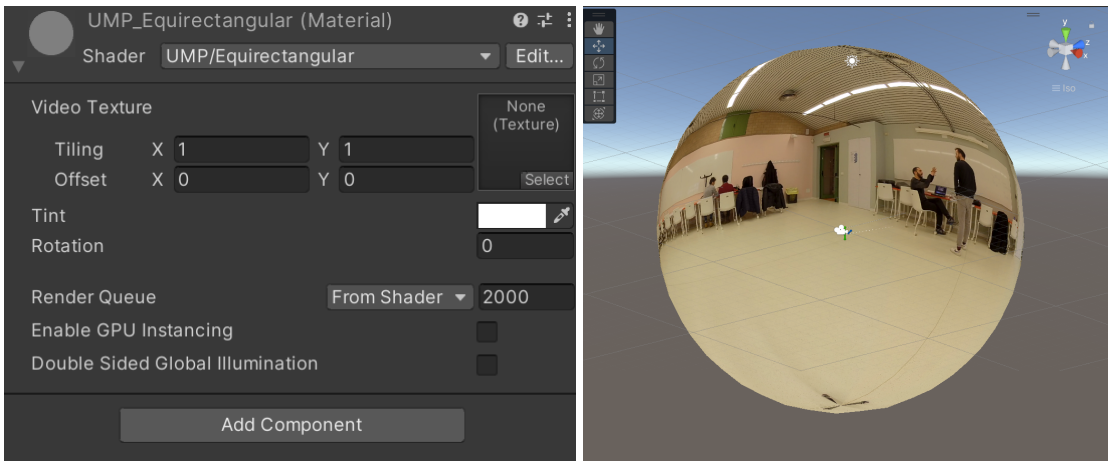

Fig. 50 Il materiale che fa uso dello shader UMP_Equirectangular e il risultato finale, mappato all'interno di una sfera.

Arrivati a questo punto, il flusso video *live* viene correttamente visionato su *Unity*.
Si ritiene importante far notare come sia presente un'inevitabile latenza, dovuta all'elaborazione da parte del motore di gioco del flusso video in entrata: si tratta di un paio di secondi, ed è una problematica che verrà ripresa nella Sezione 3.3.

3.1.2 Il rapporto fra spazi reali e spazi virtuali

La prima fase di questo periodo di ricerca è stata dedicata alla comprensione del rapporto fra gli spazi reali (la distanza fra oggetti nella stanza "fisica") e virtuali (la distanza fra oggetti sintetici nel motore di gioco).
Dopo alcuni esperimenti empirici, utilizzando oggetti reali di dimensioni note e distanza nota, la conclusione è stata che la scala in metri che si utilizza nella "realtà" corrisponde, con un ottimo grado di approssimazione, a quella di *Unity*.
In Fig. 51 si può osservare una sedia reale alta 94 cm e distante 4.45 metri dalla camera e un cubo sintetico alto, a sua volta, 94 cm e distante 4.45 unità dall'origine del mondo: la percezione prospettica funziona e l'errore è minimo.

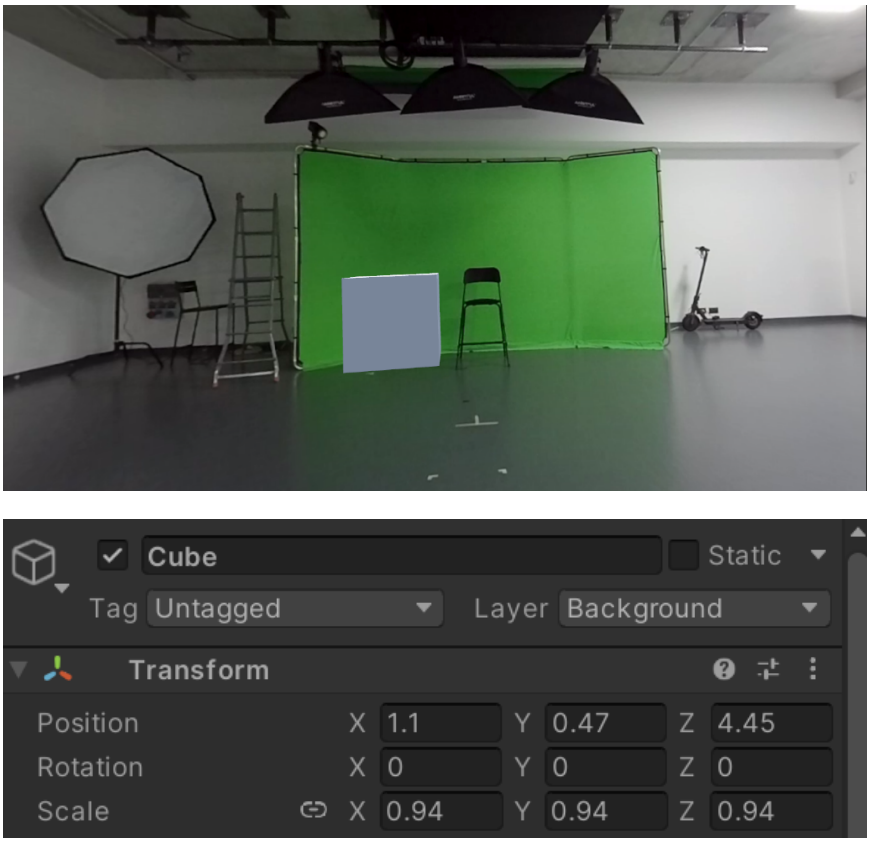

Fig. 51 Sopra, il cubo sintetico; sotto, la sua transform (posizione, rotazione, scala).

Un aspetto molto importante da considerare è l'altezza della camera fisica 360°: questa va misurata in fase di allestimento, poiché va utilizzata per "abbassare" tutti gli oggetti sintetici della stessa unità.

Infatti, dal momento che la camera virtuale che renderizza il punto di vista dell'utente è fissata nell'origine del mondo (0,0,0), tutti gli oggetti che vengono compositati sul pavimento virtuale non devono ovviamente stare alla sua stessa altezza, ma devono essere traslati verso il basso per mantenere la coerenza prospettica.

L'approccio generale che è stato impiegato in questo progetto di tesi è stato quello di creare un oggetto *Floor*, situato a (*-altezza camera*) sull'asse y, ed utilizzare come *parent* di tutti gli oggetti che devono essere compositati sul suolo.

L'oggetto *Floor* inoltre, è dotato di un *collider*, che lo rende un pavimento a tutti gli effetti, in grado di bloccare gli oggetti che ci cadono sopra.

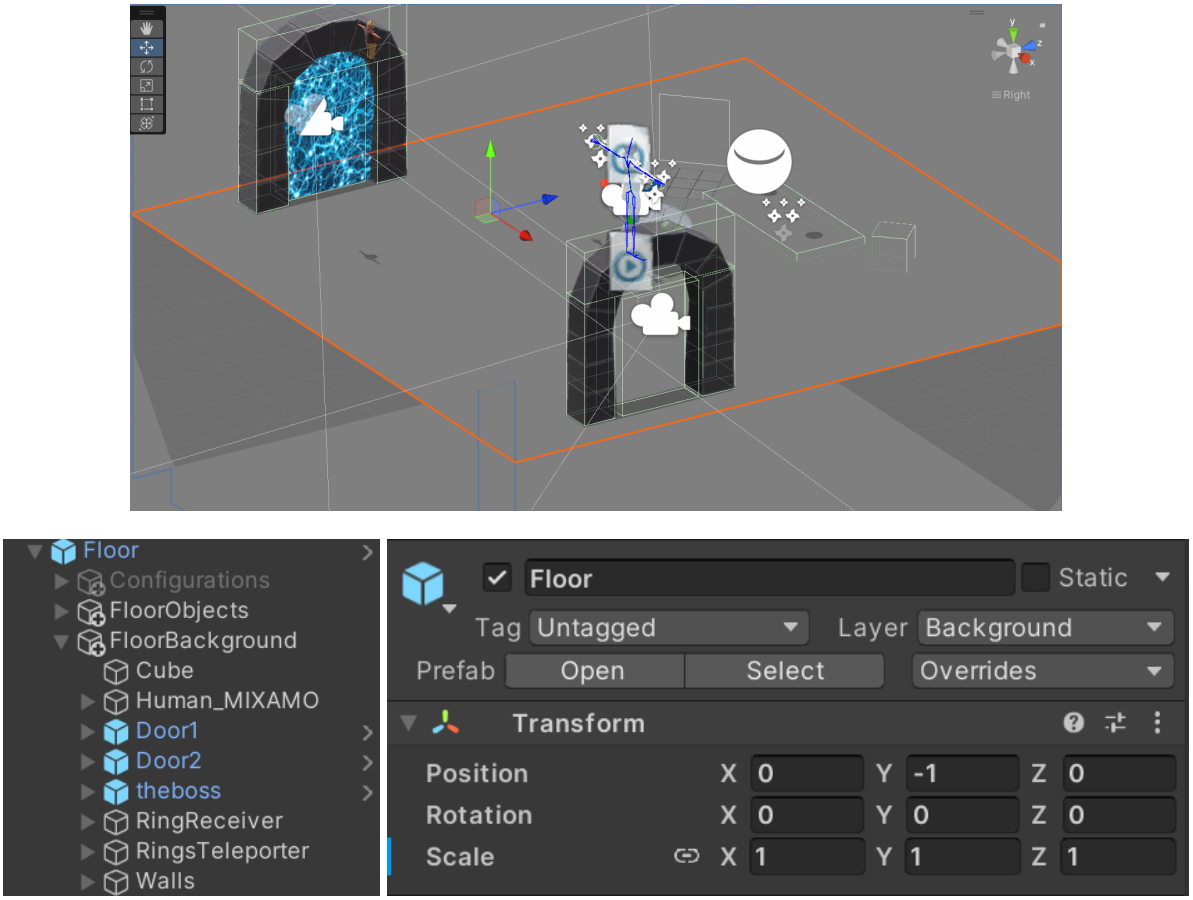

Fig. 52 Sopra, evidenziato in arancione, l'oggetto Floor; sotto, la sua gerarchia e la sua transform (Y = -1 in quanto la camera è situata a 1 metro da terra).

In questo modo, si ottiene un'impostazione pulita e efficace della scena.

3.1.3 Problematiche e soluzioni

In questa fase dell'implementazione, gli oggetti sintetici possono essere posizionati all'interno della scena con la consapevolezza che posseggano proporzioni e prospettiva credibili.

Tuttavia, vi sono almeno due problematiche da tenere in conto e da risolvere: la prima è la necessità di rendere invisibile l'oggetto *Floor*, affinché si veda il pavimento reale della stanza all'interno del video 360° e la seconda è la gestione del *parallax mismatch*, accennato nel Capitolo 2, Sezione 2.1.2.

L'oggetto *Floor* inoltre deve sì essere invisibile, ma deve comunque ricevere le ombre degli oggetti, al fine di fornire una *depth cue* [10] essenziale per la percezione della profondità.

L'invisibilità del Floor

Cercando sul *web*, è stato trovato uno *shader* [43] che permette di rendere invisibile l'oggetto a cui è applicato, preservando però le ombre che altri oggetti proiettano su di esso: è stato preferito questo approccio, più immediato, rispetto al modificare la *pipeline* di *rendering* al fine di implementare il *differential rendering* [34] come nei paper citati nella Sezione 2.1 del Capitolo 2.

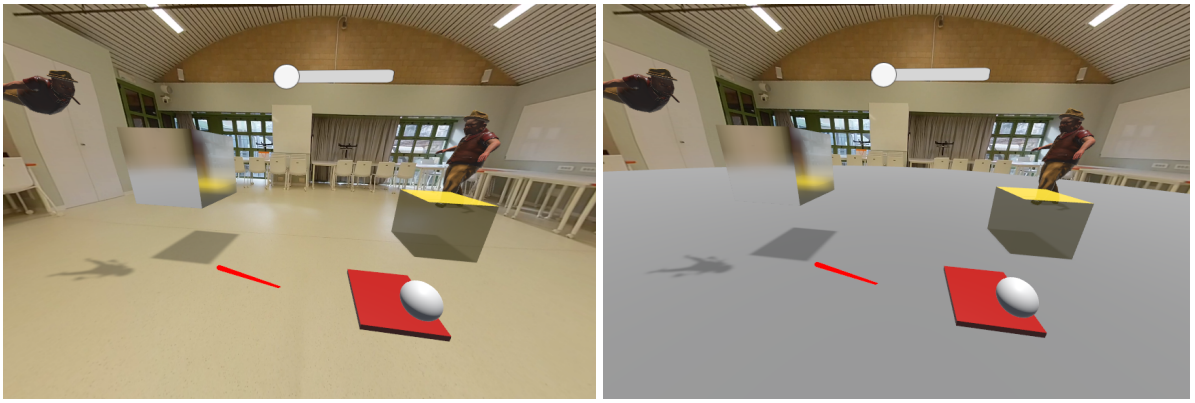

Fig. 53 A sinistra, l'oggetto Floor con lo shader applicato; a destra lo stesso oggetto con un materiale di default.

L'effetto ottenuto è l'illusione che il pavimento della stanza reale riceva le ombre proiettate dagli oggetti sintetici, ottenendo un *compositing* efficace.

Alla luce di questo e alla consapevolezza che le dimensioni spaziali su *Unity* corrispondono, con ottimo grado di approssimazione, a quelle del mondo reale, è possibili istanziare le quattro mura della stanza virtuale alla stessa distanza dalla camera delle mura della stanza "reale", applicando poi su di essere lo *shader* in questione: si ottengono quindi dei muri invisibili, che però accettano le ombre degli oggetti sintetici su di essi e forniscono un'ostruzione se gli oggetti stessi vi entrano in collisione.

Questo *shader* presenta un'essenziale limitazione: supporta soltanto una sorgente di luce di tipo direzionale.

Questa peculiarità ha delle importanti implicazioni, che tratteremo fra poco, nella Sezione 3.1.6 dedicata alla *light estimation*.

Allo *shader* in questione ne è stato affiancato un altro, molto semplice, che ha il compito di rendere "occludenti" le superfici trasparenti su cui è applicato.

In questo modo, il *Floor* trasparente allo stesso tempo riceve le ombre degli altri oggetti e occlude tutto ciò che "sta dietro": diventa così possibile "far fuoriuscire" oggetti dal pavimento, come mostrato in Fig. 54.

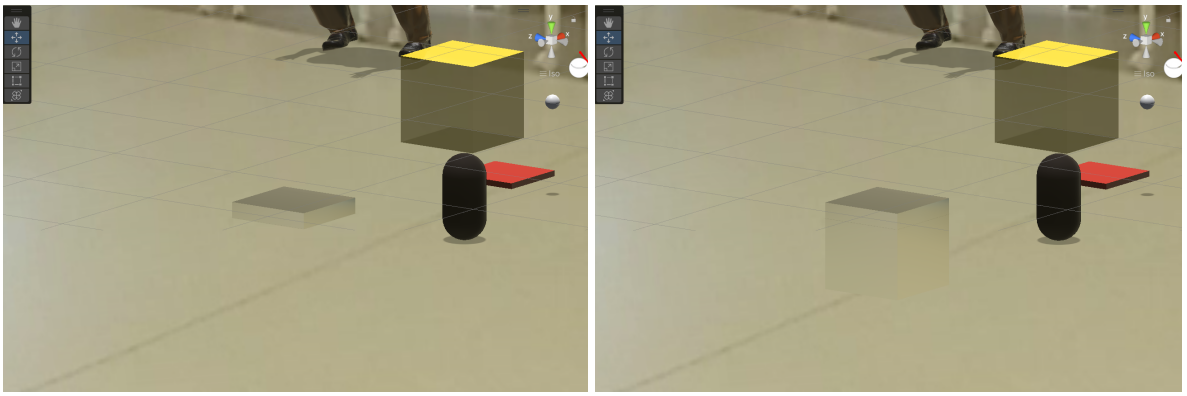

Fig. 54 A sinistra, l'oggetto Floor con lo shader applicato; a destra, senza shader.

Parallax mismatch

Questo problema è stato citato dal *team* di *MR360* [31], come abbiamo visto nel Capitolo 2, Sezione 2.1.1.

Si tratta del più grande ostacolo all'ottenimento di un *compositing* verosimile e nasce dal contrasto fra due elementi in gioco:

- La *virtual camera* di *Unity*, se utilizzata per il *playback* in *VR* (come in questo caso), è stereoscopica e mostra quindi gli oggetti sintetici in 3D, renderizzando contributi diversi per i due *display* dell'*HMD*.
- Il video 360° è monoscopico, quindi "piatto", ed è mappato, come già visto, all'interno di una sfera: esso non possiede in alcun modo informazioni di profondità.

Ne consegue che, senza intervenire, gli oggetti sintetici sembreranno sempre "fluttuare" sul video 360° di sfondo: risulta impossibile mostrare l'effetto sulle pagine bidimensionali di questa tesi, ma è una condizione che "rompe" completamente il *compositing*.
Inizialmente è stato trovato il compromesso, come indicato dal team di *MR360*, di utilizzare soltanto oggetti lontani che minimizzino l'effetto di parallasse: se per uno scenario *outdoor* potrebbe anche andare bene, risulta impossibile rendere l'effetto realistico in piccole stanze reali.
Chiedendo a novembre delucidazioni per *mail* a Andrew Chalmers, uno dei co-creatori di di *MR360*, è stata proposta una soluzione che non era attuabile con la loro configurazione *HTC Vive* + *Unreal Engine*: disattivare il *rendering* stereoscopico della camera virtuale.
Non avendo trovato nella documentazione ufficiale un modo per renderizzare in modo monoscopico, è stato trovata su un *forum* [44] una soluzione semplice ma efficace, ovvero ridurre la *scale* della camera VR a 0
Non è stata trovata una spiegazione "scientifica", ma è molto probabile che portando la dimensione lungo l'asse x a zero si annulli la distanza interpupillare (*IPD*), rimuovendo la stereoscopia per le motivazioni spiegate nella sezione 1.3.3.
Un altro suggerimento fornito da Chalmers è stato quello di bloccare il tracciamento della posizione della camera VR, trasformandola di fatto da un sistema 6-DoF a uno 3-DoF [27], differenza citata nella Sezione 1.3.1: si tratta di un accorgimento necessario, in quanto vogliamo garantire che la visuale dell'utente rimanga sempre al centro della sfera su cui è mappato il video 360°.

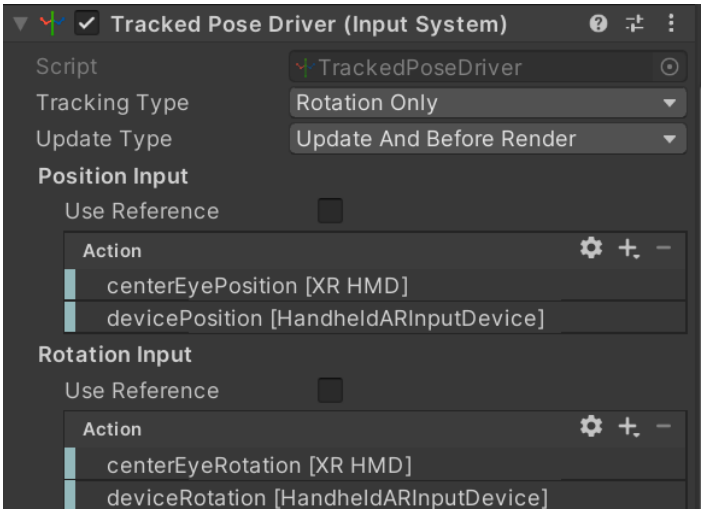

Fig. 55 Il componente Tracked Pose Driver, a cui nella proprietà Tracking Type è stato inserito Rotation Only.

Arrivati a questo punto dell'implementazione, non sarebbe esagerato definire il *compositing* come "efficace".

Tuttavia, l'immagine finale è inevitabilmente "piatta": sembra di guardare un video 360° compositato su un qualche software di post-produzione, dal momento che è stata rimossa ogni componente stereoscopica.

Sarebbe interessante mantenere la stereoscopia quando possibile, soprattutto se si vuole implementare l'interazione da parte dell'utente, in quanto può essere necessaria una percezione degli spazi che solo il 3D riesce a dare.

Inoltre, sempre in ottica di voler permettere una qualche forma di interazione, sarebbe utile preservare anche i 6 *Degrees of Freedom*, per permettere all'utente di compiere piccoli movimenti con la testa per gestire l'inquadratura in modo più dinamico.

Nella prossima sezione andremo a formalizzare i due approcci possibili per la gestione delle camere, al fine di ottenere un *compositing* 360 di qualità.

3.1.4 Approccio a camera singola

Si tratta della metodologia descritta finora: utilizzare un'unica camera monoscopica *3-DoF*.

I sistemi studiati nello stato dell'arte attuano questo approccio, rinunciando ad ogni tipo di componente stereoscopica e al tracciamento della posizione della testa dell'utente.

In Fig. 56 si può vedere come la camera sia in una posizione fissa, nell'origine del mondo, in grado solo di ruotare sui 3 assi x, y e z.

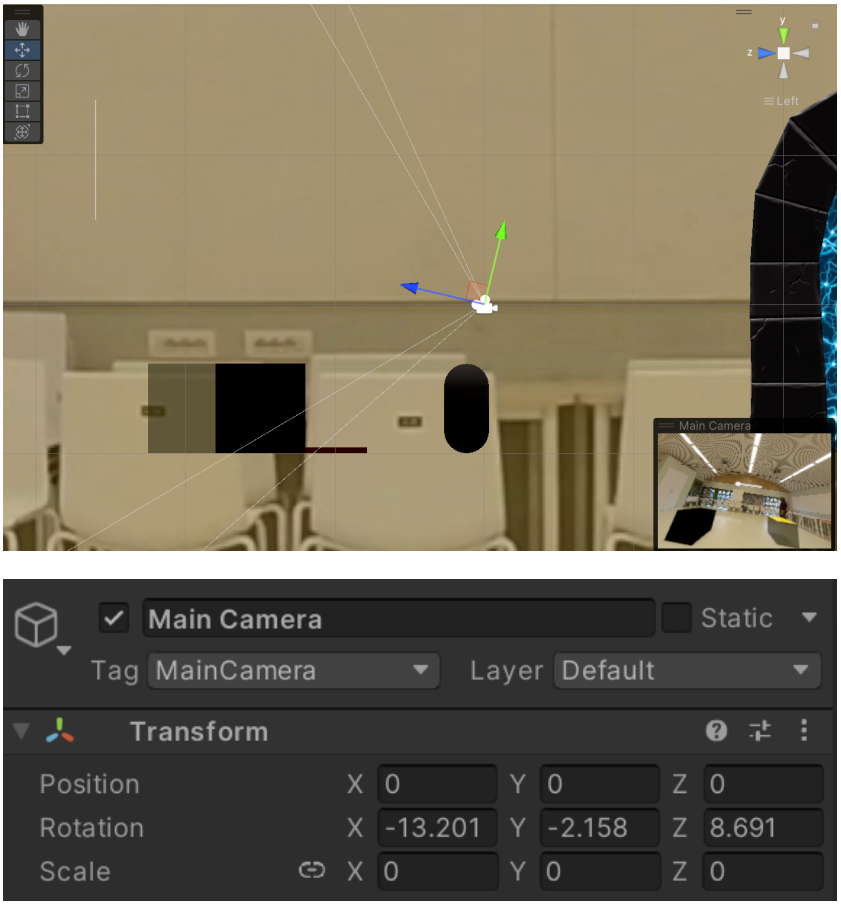

Fig. 56 La singola camera utilizzata con questo approccio: la posizione rimane fissa nell'origine e varia solo la rotazione. Notare inoltre come la scale sia 0 per garantire la monoscopia.

In questo progetto di tesi è stato anche tentato un differente approccio, illustrato nella prossima sezione, che prevede l'utilizzo di una doppia camera: una stereoscopica e una monoscopica.

3.1.5 Approccio a camera doppia

Il sistema proposto in questo capitolo prevede, come già accennato, l'utilizzo di due camere virtuali (una monoscopica e una stereoscopica).
Vediamo le loro *features*:

- Camera monoscopica: si comporta allo stesso modo di quella descritta nella sezione precedente, ovvero ha *scale 0* per rimuovere l'effetto 3D e possiede solo 3 *DoF*.
  Inoltre, renderizza soltanto gli oggetti presenti su un *layer* chiamato *Background*.
- Camera stereoscopica: mantiene la classica *scale* di 1 e ha 6 *DoF*, ovvero permette anche il tracciamento dello spostamento della testa dell'utente.
  Renderizza soltanto gli oggetti situati sul *layer* chiamato *Objects*.

Il concetto alla base è piuttosto semplice: gli oggetti sintetici vicini all'utente, ovvero quelli con cui può interagire, verranno renderizzati in 3D per aumentare la percezione della profondità e il rapporto fra le loro posizioni, mentre quelli lontani (sullo sfondo) verranno renderizzati dalla camera monoscopica per "fonderli" al meglio con il video 360°, presente anch'esso sul *layer Background* (Fig. 57).
Ne consegue che ora l'utente ha la possibilità di compiere piccoli movimenti della testa e vedere gli effetti sugli oggetti presenti sul *layer* stereoscopico, assicurandosi allo stesso tempo di mantenere invariata la prospettiva degli oggetti sullo sfondo.

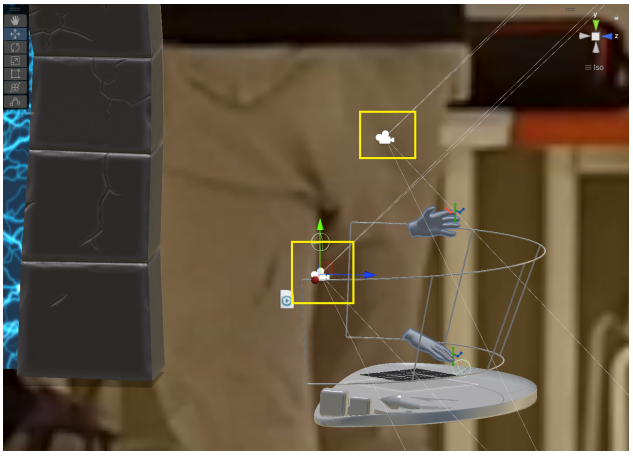

Fig. 57 La camera mono rimane sempre nell'origine del mondo, mentre quella stereo può muoversi se l'utente muove la testa.

Fig. 58 Dall'alto al basso: il contributo del layer Background, il contributo del layer Objects e l'unione di essi visto dall'utente. Ovviamente in queste immagini non è possibile apprezzare la differenza percettiva dei due contributi.

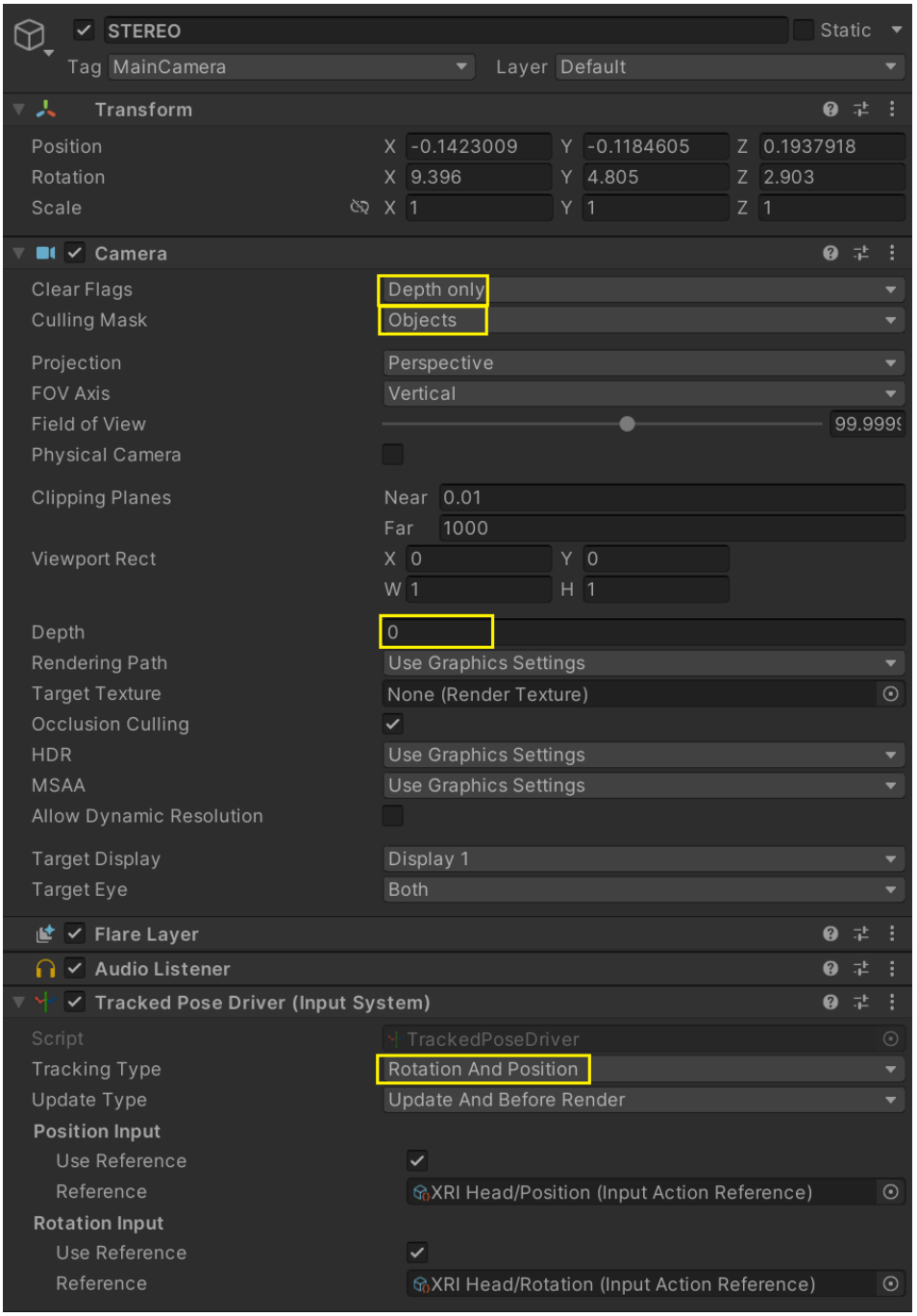

Fig. 59a Il gameobject della camera chiamata "STEREO".

Come visibile in Fig. 58a, la camera stereo mostrerà soltanto il *layer Objects*, possiede 6 *DoF* e ha il parametro *Depth* settato a 0: verrà renderizzata davanti a quelle con valore minore.

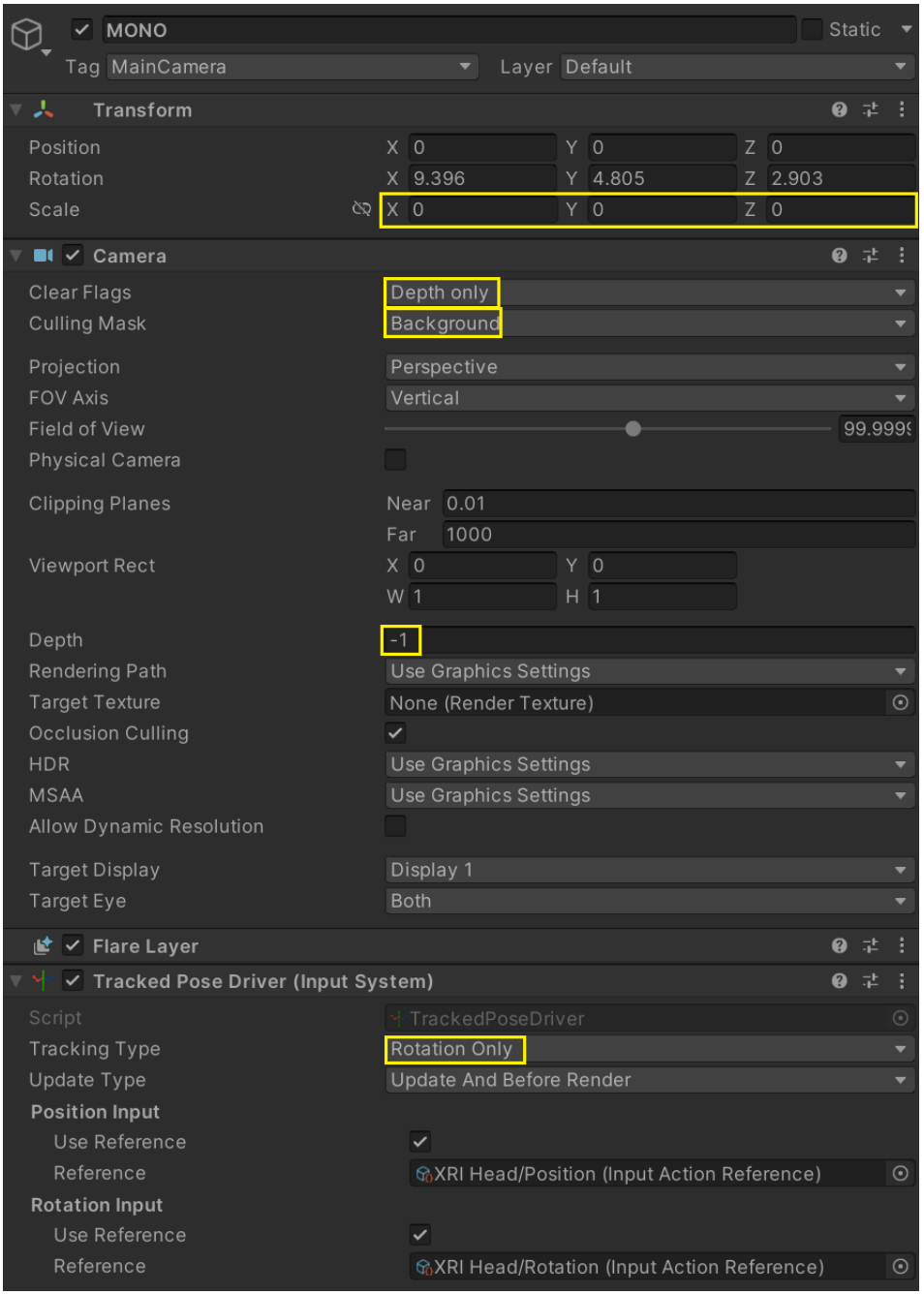

Fig. 59b Il gameobject della camera chiamata "MONO".

La camera monoscopica mostra solo gli oggetti appartenenti al *layer Background*, possiede 3 *DoF* e ha *Depth* -1: verrà renderizzata "dietro" quella stereoscopica.

La necessità di utilizzare due camere contemporaneamente nasce nel momento in cui si è scelto di implementare l'interazione dell'utente tramite il tracciamento delle mani fornito da *Meta Quest 2*, l'*HMD* utilizzato in questo progetto di tesi: per una manipolazione precisa degli oggetti e un corretto *hand tracking* la stereoscopia è senz'altro consigliata ed è un aspetto affrontato nella prima parte dello *user test* nel Capitolo 4.

Un altro aspetto importante da considerare è la gestione delle ombre.
La presenza di due *render layer* fa sì che gli oggetti posti su *Objects* non proiettino ombre su quelli in *Background* e viceversa, causando una discontinuità: si tratta di una problematica che non si può risolvere ma può essere mitigata, come verrà spiegato nella Sezione 3.4.

Fig. 60 A sinistra l'approccio a camera singola (con il modello della mano sullo stesso layer Background del Floor); a destra l'approccio a camera doppia (il modello della mano si trova sul layer Objects mentre il Floor su Background): da notare l'assenza dell'ombra sul pavimento.

Infine ritorna, solamente per gli oggetti situati sul *layer Objects*, il problema del *parallax mismatch* spiegato nella Sezione 3.1.3: questi infatti, essendo visualizzati in modo stereoscopico, sembreranno fluttuare se appoggiati sul pavimento (che si trova sul *layer Background*).
Si tratta di un effetto che rischia di rompere il senso di *compositing* ma può essere mitigato con uno stratagemma, spiegato nella Sezione 3.4.

Nel complesso, l'idea di utilizzare due camere risulta un buon compromesso fra l'efficacia dell'interazione dell'utente e la verosimiglianza del compositing: nel Capitolo 4 verranno testate diverse configurazioni per provare questo assunto.

3.1.6 Illuminazione

La gestione dell'illuminazione è stata un punto affrontato nelle fasi iniziali di questa ricerca, prendendo ispirazione dai concetti espressi nel *paper* di *MR360* [31]: è infatti stato elaborato un algoritmo per riconoscere le sorgenti di luce all'interno del *livestream*, al fine di generare le *light sources* nella scena virtuale.
I risultati ottenuti sono funzionanti, ma resi inutilizzabili in uno scenario reale a causa di alcune limitazioni di *Unity* e dallo *shader* utilizzato.
Per completezza verrà comunque illustrato il procedimento per estrarre le sorgenti di luce, in quanto semplice ma efficace.
Estraendo il primo *frame* dal flusso video, vengono scansionati tutti i *pixel* presenti nella metà superiore dell'immagine equirettangolare: è raro infatti trovare sorgenti di luce provenienti dal basso e viene quindi ignorata la porzione inferiore, che anzi potrebbe contenere, come indicato dal team di *MR360* [31], delle riflessioni che causerebbero falsi positivi.
Vengono selezionati i *pixel* che superano una soglia di *brightness* scelta manualmente e poi settati di colore bianco, mentre tutti gli altri vengono portati a nero.
A questo punto si ottiene un'immagine della stessa risoluzione del video equirettangolare con i soli *pixel* più luminosi, che presumibilmente rappresentano sorgenti di luce.
Ovviamente non è possibile associare una sorgente di luce su *Unity* ad ogni *pixel* bianco dell'immagine, quindi è stato attuato un semplice approccio per raccogliere gruppi di *pixel* significativi: viene sottocampionata l'immagine *black and white*, portandola a 1/64 della risoluzione originale (Fig. 61).

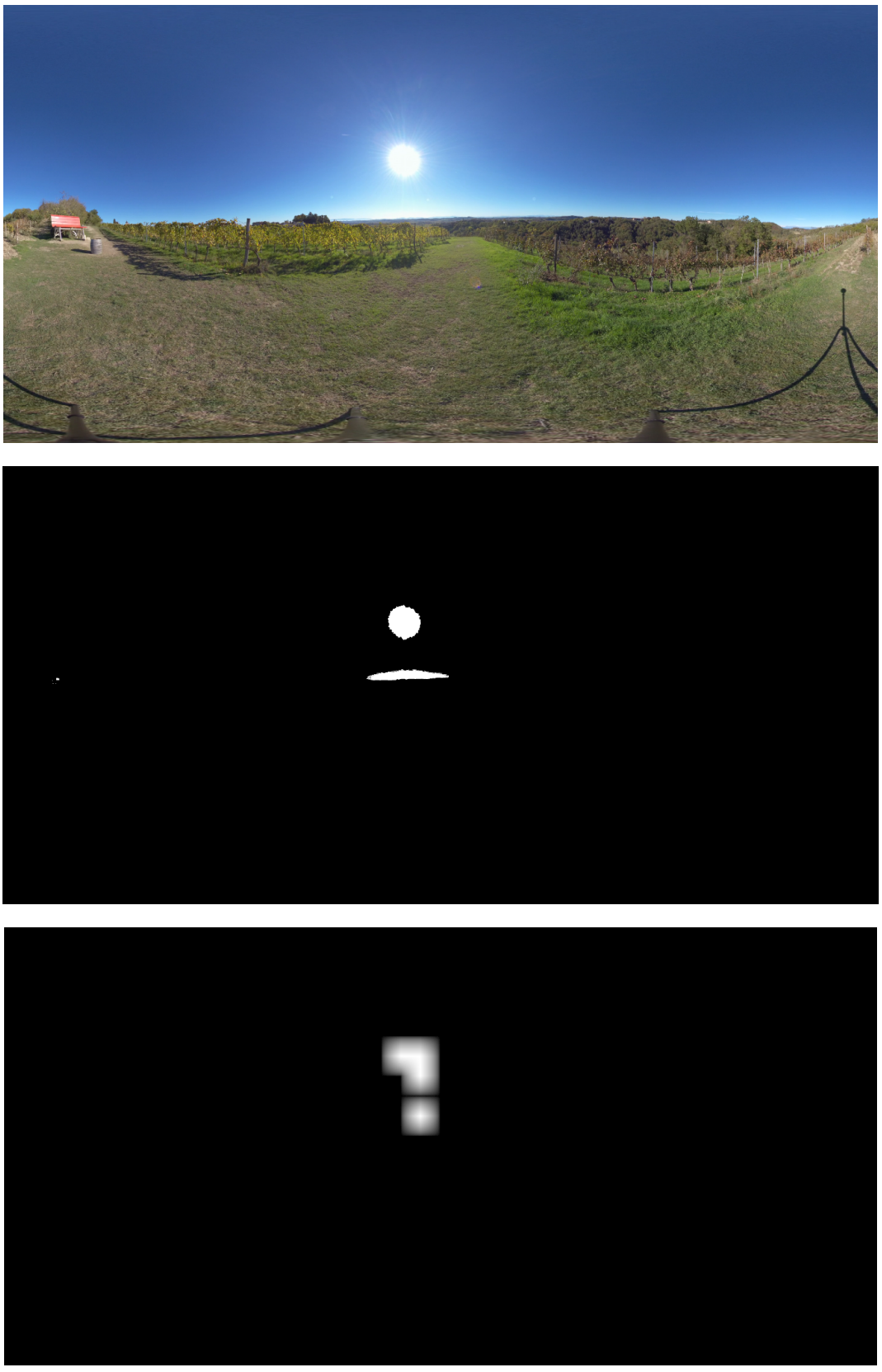

Fig. 61 Dall'alto al basso: un'immagine outdoor equirettangolare 2880x1440, il risultato del filtraggio dei pixel, l'immagine 45x22 frutto del sottocampionamento.

A questo punto si ha un'immagine equirettangolare dove i *pixel* bianchi rappresentano ancora, a grandi linee, le coordinate nello spazio delle sorgenti di luce.

Mappando le coordinate cartesiane in coordinate sferiche si possono creare delle *directional light sources* su *Unity* sulla superficie della sfera che renderizza il video 360°, puntate verso il centro di essa, che idealmente creeranno delle ombre coerenti.

Fig. 62 Le quattro light sources create dai quattro pixel bianchi presenti in basso in Fig. 61.

Vi sono tuttavia, come anticipato, delle problematiche che hanno reso impossibile l'implementazione di questo algoritmo.

- *Unity* riesce a renderizzare soltanto 2 ombre (generate da luci direzionali) per oggetto alla volta: nel caso di tre sorgenti di luce presenti nello stesso momento, una di queste non genererebbe ombre.
- Lo *shader* utilizzato per il *Floor* supporta, come indicato nella Sezione 3.1.3, una sola luce direzionale, limitando ancora di più le possibilità.

All’aperto il sole rappresenta l’unica fonte di luce, come si può vedere in Fig. 61, e quindi si può approcciare la problematica imponendo all’algoritmo di attivare soltanto una delle luci rilevate e istanziate: il risultato è efficace, come mostrato in Fig. 63.
Tuttavia, lo scenario implementativo classico dei progetti *Presence* prevede di utilizzare *location indoor*, dove solitamente le sorgenti di luce sono più di una (o comunque, l’imposizione di una luce singola complicherebbe l’organizzazione e limiterebbe le possibilità narrative).
Non si può pensare quindi di attuare l’approccio *outdoor*, ovvero attivare una sola luce su *Unity,* in quanto il risultato sarebbe parziale.
Nella Sezione 3.4, dedicata alle scelte implementative, verrà ripreso questo aspetto.

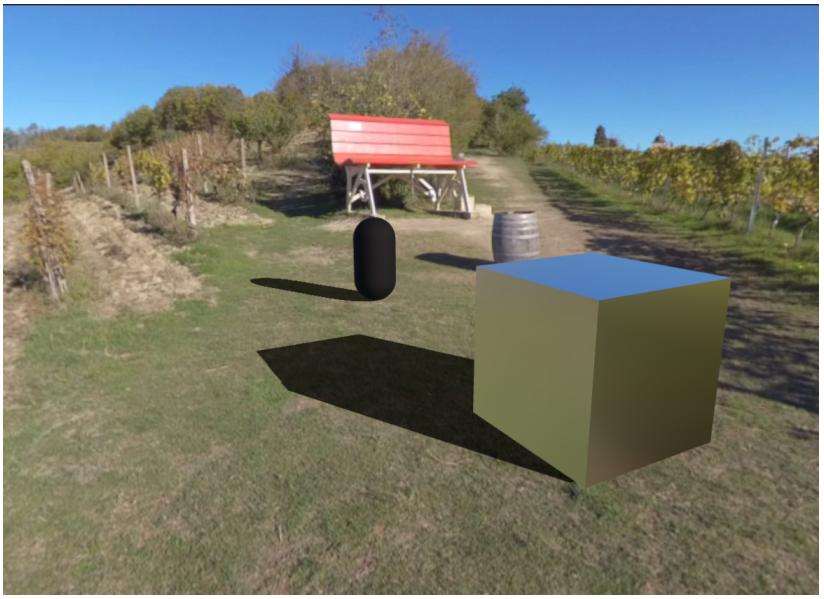

Fig. 63 Scenario outdoor con una sola sorgente di luce attivata: le ombre proiettate dal cubo e dalla capsula sono coerenti con quella del barile di legno sullo sfondo.

3.2 Virtual Embodiment

Il progetto *Presence*, come esplicato nella Sezione 1.1, affronta le tematiche della moltiplicazione e l'autorappresentazione del sé in rete, nella virtualità e nei social.
Un modo per ampliare le possibilità di questa ricerca è senz’altro il permettere all’utente di controllare un *avatar*, attraverso i movimenti delle proprie mani tracciate.
Una volta scelto, sulla piattaforma *Mixamo*, un modello umanoide base per effettuare i test, è stata effettuata l’associazione fra le mani e la testa dell’utente e il *rig* fornito dal sito stesso.

Prendendo spunto da un contributo video [46], è stato utilizzato il *package* chiamato *Animation Rigging* [45] per attuare il *rigging* e implementare il concetto di cinematica inversa (*IK*).

Innanzitutto, è necessario definire dei *constraints* per indicare al modello umanoide quali movimenti sono consentiti e quali no:

- *Head Constraint*: impone, tramite un *Multi-Parent Constraint* [45], che il *bone* della testa sia influenzato dallo stesso *Head Constraint*.
- *Right Arm / Left Arm IK*: vengono definiti gli elementi chiave della cinematica inversa, ovvero il *target* che viene utilizzato per ricostruire la catena cinematica (composta da braccio, avambraccio e mano del modello umano) [47] e l'*hint*, che pone un vincolo al movimento che la catena cinematica può compiere.

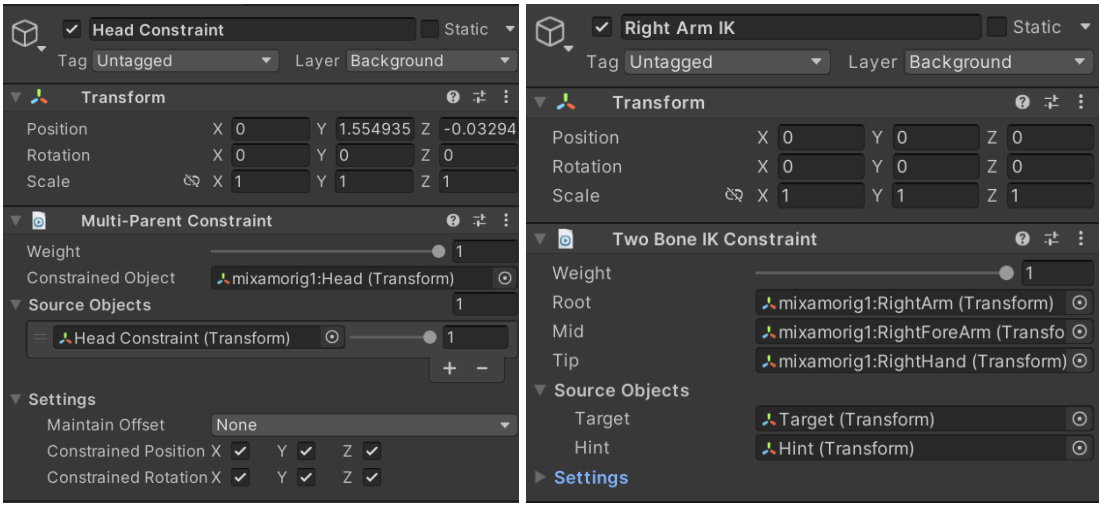

Fig. 64 Da sinistra a destra: Head Constraint e Right Arm IK.

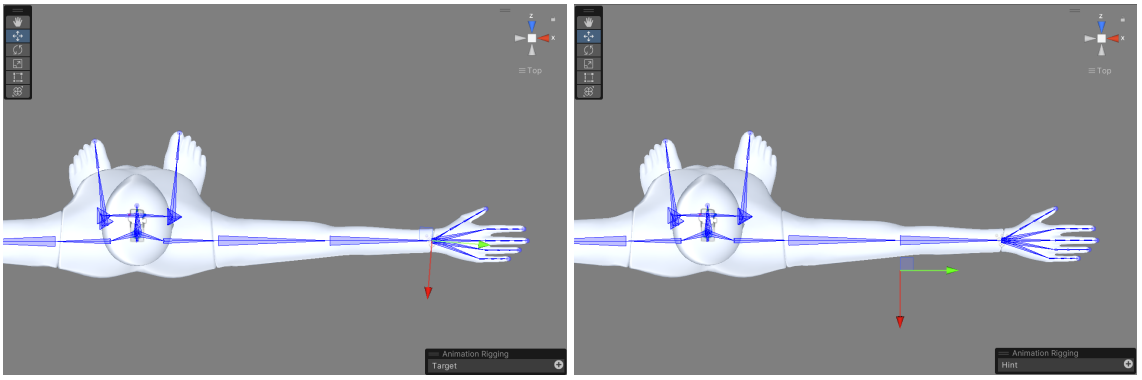

Fig. 65 A sinistra il target utilizzato in Right Arm IK, a destra l'hint: per il braccio sinistro si procede nello stesso modo.

Andando a indicare all'elemento *parent* del rig che esistono dei *constraints*, questo andrà a generare il *rig* proprio con un componente chiamato *Rig Builder*.

A questo punto, non resta che associare gli *anchors VR* forniti dalle mani ai *targets* (visibili in Fig. 65) e la camera virtuale all'*Head Constraint* per ottenere una visuale in prima persona dell'*avatar*: questo procedimento si attua utilizzando uno *script* chiamato *Head Body Rig*.

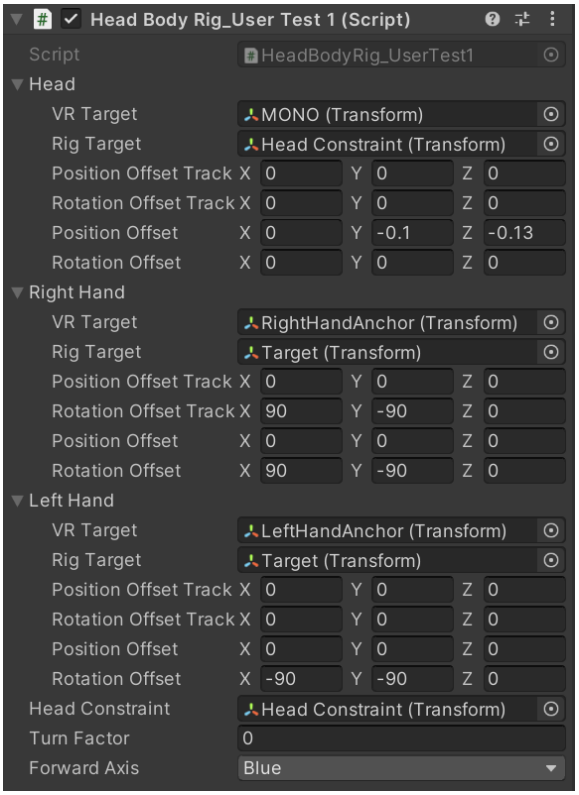

Fig. 66 Head Body Rig: i VR Target sono le componenti VR di testa e mani, i Rig Target sono i constraints creati in fase di setup.

Dal momento che l'utente fruisce delle esperienze *Presence* da fermo, non è stata implementata la cinematica inversa per le gambe, in quanto l'*avatar* rimane sempre immobile.
Riguardo questo, è da notare, nella Fig. 66, il valore *Turn Factor* responsabile della rotazione sull'asse Y dell'*avatar*: è stato settato a 0 per far sì che questo rimanga fermo se l'utente ruota la testa.
Infine, per ottenere il tracciamento delle dita dell'utente, è stato utilizzato uno *script* [48] che associa ad ogni *bone* della mano dell'utente una *bone* del *rig* dell'avatar.

3.2.1 Approccio *first-person*

Quanto descritto finora è il metodo impiegato per ottenere un *embodiment* in prima persona (*POV*).

Si tratta di una possibilità per immedesimare di più l'utente all'interno del mondo immersivo, in quanto ottiene un corpo e non è più solo dotato di mani "fluttuanti", come visto ad esempio in Fig. 60.

Allo stesso tempo si tratta di una situazione piuttosto delicata, in quanto è molto facile che le condizioni di immedesimazione vadano a rompersi, per alcuni motivi:

- Il modello umanoide che l'utente impersona ha delle dimensioni diverse dal corpo dell'utente stesso: può capitare che, distendendo le braccia, l'*avatar* "non arrivi" al punto desiderato.
- La cinematica inversa rileva solo il posizionamento delle mani e cerca quindi di ricostruire il movimento di braccio e avambraccio: eventuali rotazioni del gomito dell'utente non verranno registrate e quindi si potrebbero creare situazioni di "disallineamento" fra la posa reale e quella "ricostruita" dalla cinematica inversa.

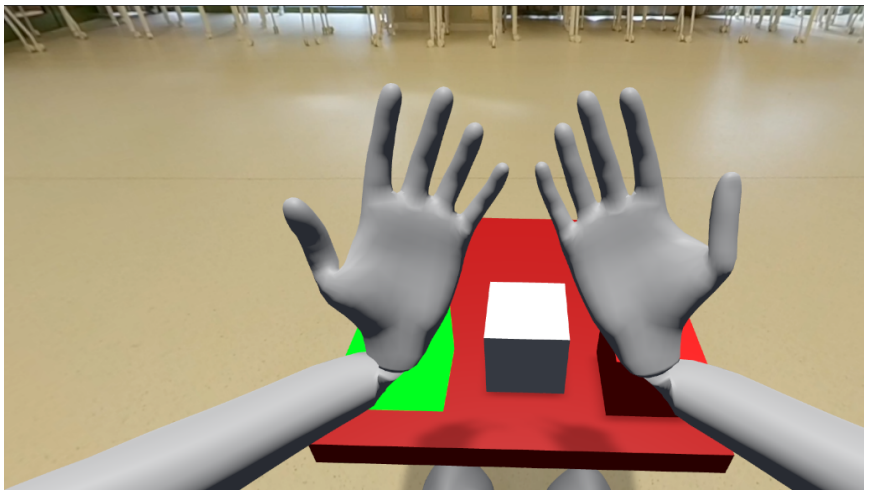

Fig. 67 L'approccio first-person.

Implementando questo approccio nella configurazione a doppia camera, illustrata nella Sezione 3.1.5, la percezione finale è comunque accettabile, poiché si percepisce efficacemente un senso di profondità; d'altro canto, con l'approccio a camera singola (Sezione 3.1.4) il rapporto di proporzioni fra le mani e l'ambiente risulta "strano": ne parleremo nella Sezione 3.4.

3.2.2 Approccio *third-person*

Per testare situazioni diverse al classico *POV*, è stato scelto di sperimentare con punti di vista alternativi.

Nella Fig. 68 è possibile vedere l'approccio *third-person* che è stato implementato: l'*avatar* che si controlla è al di fuori dal corpo dell'utente ed è quindi possibile osservarne il modello 3D.

Il senso di straniamento risulta accresciuto, dal momento che si ha l'impressione di avere un "clone" di sé stessi davanti.

Allo stesso tempo, la cinematica inversa del braccio rimane non precisissima come illustrato poco fa, ma viene percepita come meno "strana", in quanto il punto di vista dell'utente non è più "all'interno" dell'*avatar*.

Una limitazione aggiuntiva è l'assenza del tracciamento delle dita se la prospettiva in terza persona è ruotata rispetto a quella standard, come nell'esempio della Fig. 68, dove vi è un *avatar* che ci osserva (rotazione di 180 gradi) e uno ruotato lateralmente (90 gradi): si tratta di un problema probabilmente risolvibile, ma a causa di difficoltà di implementazione dello script non è stato per ora possibile.

A livello implementativo, attuare lo spostamento dell'*avatar* risulta molto facile, in quanto basta spostare l'oggetto *rig* nella gerarchia del *gameobject* del modello 3D.

Fig. 68 L'approccio *third person*, con l'*avatar* sia rivolto verso l'utente che ruotato di 90 gradi.

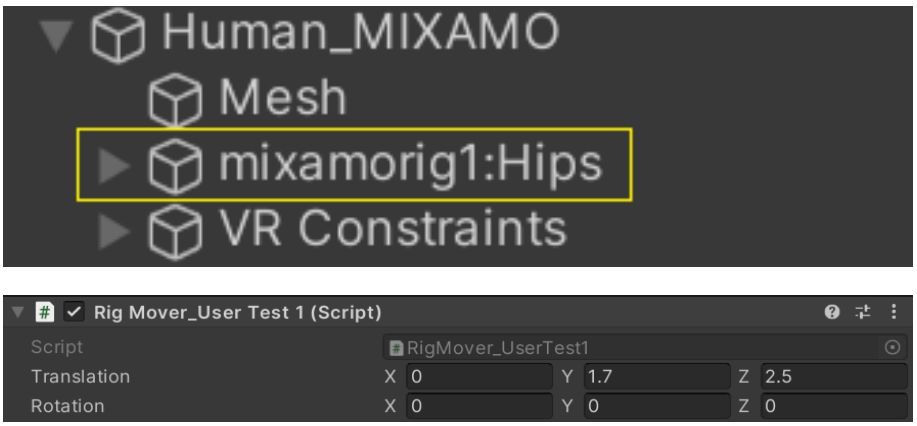

Fig. 69 In alto, la gerarchia dell'avatar con evidenziato il rig; in basso un semplice script applicato al rig per muoverlo e ruotarlo dove desiderato.

Nel Capitolo 4 vedremo uno scenario implementativo dove è stata impiegata quest'ultima modalità *third-person*.

3.3 Body Tracking e Pose Estimation

Un importante aspetto di questa tesi, nell'ottica di voler estrarre più informazioni possibili dal flusso video *live* da utilizzare in modo interattivo, è stato il tentare di tracciare i corpi presenti nel video 360°.
Si tratta di un'implementazione non presente attualmente in letteratura ed è quindi stato necessario tentare diversi approcci con un processo di *trial and error*.
La libreria che è stata scelta per attuare il *body tracking* è *OpenPose* [49], che è possibile implementare su *Unity* grazie a un *plugin* [50].

3.3.1 Utilizzo di *OpenPose* in 360°

OpenPose è una libreria multi-persona che permette di rilevare la posa di un corpo umano in tempo reale: in particolare può riconoscere dei punti chiave del corpo, della faccia e della mano, per un totale di 135 *keypoints* [49].
Per l'implementazione di questo progetto, il focus è stato dato al rilevamento dei punti del corpo umano, tralasciando il volto e le mani.
Vengono estrapolati i *frames* di un video, forniti a una rete neurale convoluzionale (*CNN*) [51] che li scompone in una griglia di regioni di interesse, le quali vengono analizzate per rilevare caratteristiche delle parti del corpo umano (proprio i *keypoints*

sopracitati): una volta rilevati, viene utilizzato un algoritmo per collegarli in una sorta di scheletro (fase di *association*).

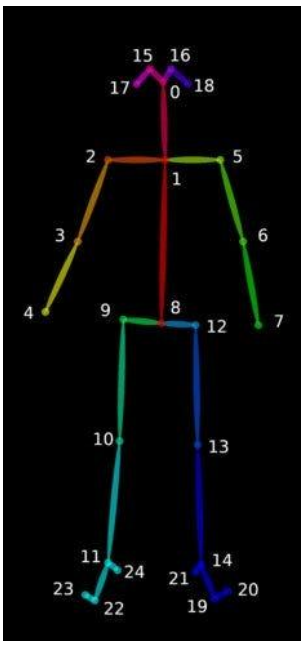

Fig. 70 I 25 keypoints del corpo riconosciuti con il modello chiamato BODY_25.

Il plugin per *Unity* viene fornito di un *wrapper*, ovvero un "contenitore" creato per incapsulare la complessità della libreria sottostante al fine di fornire un'interfaccia più semplice e intuitiva per gli sviluppatori.
Si possono quindi creare degli *script* personalizzati, più o meno complessi, che utilizzino metodi forniti proprio dal *wrapper*, al fine di velocizzare il procedimento.
Nell'implementazione di questo progetto di tesi viene dato allo *script* di *OpenPose* l'*URL* (Fig. 49) del *livestream* tramite una *producer string*, da cui verrà estrapolato il flusso video poi processato dalla libreria.

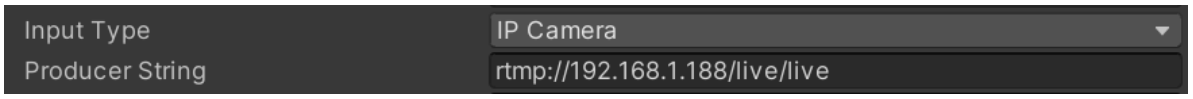

Fig. 71 OpenPose interpreta il livestream come se provenisse da un IP Camera.

Nello *script* personalizzato, una volta attivato *OpenPose* tramite il metodo *wrapper* .OPRun(), la libreria inizierà a scansionare il flusso video trovato all'indirizzo fornito dalla *producer string* vista in Fig. 71, fornendo in ritorno al metodo *OPGetOutput*() un oggetto di tipo *datum* che contiene tutte le informazioni rilevate nel *frame*.

```
if (OPWrapperBodyTracking.OPGetOutput(out datum))
{

    if (datum.poseKeypoints == null || datum.poseKeypoints.Empty()) numberPeople = 0;
    else numberPeople = datum.poseKeypoints.GetSize(0);
    peopleText.text = "People: " + numberPeople;

    // Update framerate in UI
    frameTimeQueue.Enqueue(Time.time);
    frameCounter++;
    if (frameTimeQueue.Count > queueMaxCount)
    {
        // overflow
        frameTimeQueue.Dequeue();
    }

    if (frameCounter >= queueMaxCount || frameTimeQueue.Count <= 5)
    {
        // update frame rate
        frameCounter = 0;
        avgFrameRate = frameTimeQueue.Count / (Time.time - frameTimeQueue.Peek());
        fpsText.text = avgFrameRate.ToString("F1") + " FPS";
    }

}
```

Fig. 72 All'interno del metodo Update(), l'utilizzo di datum per ottenere il numero di persone nell'immagine e calcolare gli FPS di elaborazione.

Prima di procedere con l'implementazione, è necessario fare una constatazione sull'*input* video che viene ricevuto da *OpenPose*.

Come già anticipato nella Sezione 1.3.2, il video 360° viene proiettato in equirettangolare, che è quindi il formato ricevuto dalla rete neurale convoluzionale.

Il *paper* descritto nella Sezione 2.2.1 indicava, come già noto, delle evidenti distorsioni ai poli (Fig. 43) che potrebbero ovviamente causare una mancata o scorretta *pose estimation*.

Piuttosto che risolvere con una *multi-directional projection* [36], si è constatato che, negli scenari implementativi di *Presence*, non è previsto che *performer* o persone nella scena siano in posizioni particolari e quindi inquadrate in modo distorto: esse si trovano sempre a una distanza di almeno 1 metro dalla camera, quindi sicuramente non affette da deformazioni [52], e sempre a contatto con il terreno (Fig. 73).

Alla luce di queste constatazioni, si può assumere che dare in *input* alla libreria *OpenPose* l'immagine equirettangolare inalterata non risulti un problema.

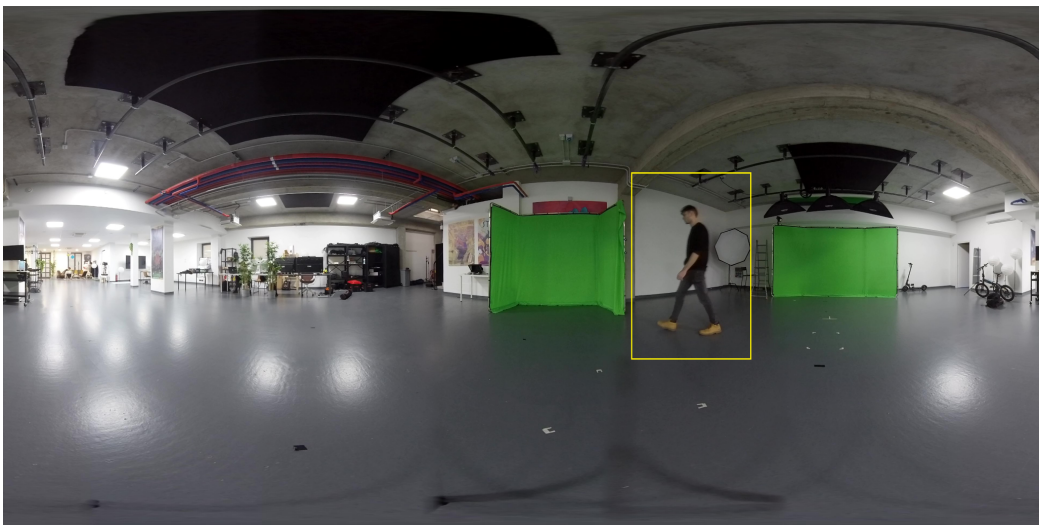

Fig. 73 Uno scenario standard: la persona indicata è distante qualche metro dalla camera, è a contatto con il terreno e quindi non presenta distorsioni.

A questo punto *OpenPose*, durante l'esecuzione, sta fornendo dei *keypoints* in *output* nel formato *datum*, illustrato in Fig. 74.

```
// Common parameters needed
const auto numberPeopleDetected = poseKeypoints.getSize(0);
const auto numberBodyParts = poseKeypoints.getSize(1);
// Easy version
const auto x = poseKeypoints[{person, part, 0}];
const auto y = poseKeypoints[{person, part, 1}];
const auto score = poseKeypoints[{person, part, 2}];
// Slightly more efficient version
// If you want to access these elements on a huge loop, you can get the index
// by your own, but it is usually not faster enough to be worthy
const auto baseIndex = poseKeypoints.getSize(2)*(person*numberBodyParts + part);
const auto x = poseKeypoints[baseIndex];
const auto y = poseKeypoints[baseIndex + 1];
const auto score = poseKeypoints[baseIndex + 2];
```

Fig. 74 Il formato Datum: per ogni persona in scena si possono ottenere le coordinate x e y di una parte del corpo sfruttando i dati contenuti nell'array poseKeypoints.

Sempre in Fig. 74 si può notare il parametro *score*, che è una valutazione di accuratezza che *OpenPose* offre per ogni *keypoint* rilevato: si può utilizzare questa informazione per renderizzare soltanto i punti con *score* più alto, che probabilmente rappresentano davvero il giunto della persona.

A questo punto le coordinate cartesiane di ogni *keypoints* possono essere rimappate in coordinate sferiche per poter essere sovraimpresse al video 360° e utilizzate per pilotare la posizione di semplici oggetti *Sphere*.

Come si potrà intuire, i *keypoints* sono piatti e "spalmati" sulla superficie di una sfera (Fig. 75) e non posseggono quindi informazioni di profondità, ma da momento che, come spiegato nella Sezione 3.1.3, video 360° e oggetti sintetici sono a loro volta visualizzati in modo monoscopico, l'illusione è che i *keypoints* siano correttamente sovraimposti al corpo della persona, come illustrato in Fig. 76.

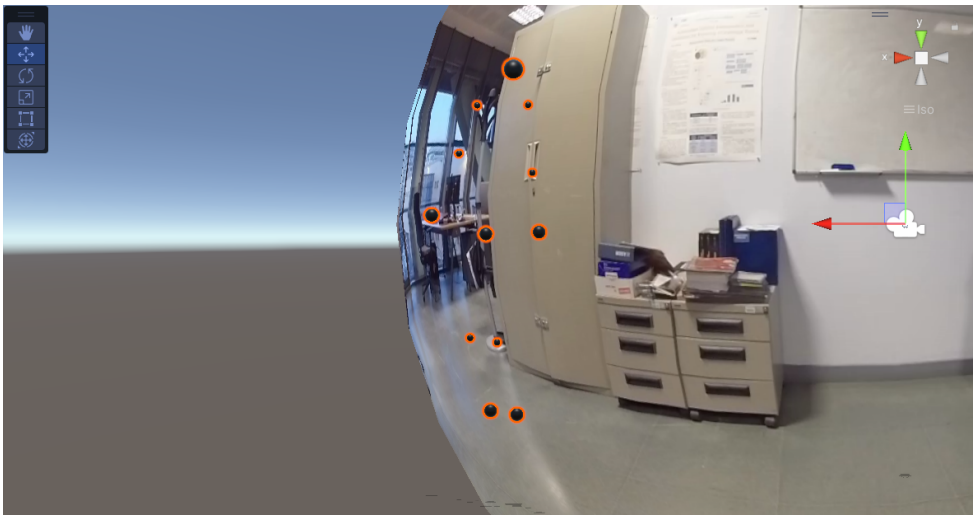

Fig. 75 I keypoints si trovano sulla superficie della sfera che renderizza il video 360°.

Fig. 76 Vista la bidimensionalità dell'immagine 360°, l'assenza di profondità dei keypoints non è rilevante.

Le informazioni ottenute dall'array *poseKeypoints* possono essere utilizzate in vari modi, come spiegheremo nella Sezione 3.4, ma va sicuramente evidenziata una grossa limitazione.

3.3.2 Gestione della latenza

Come anticipato nella Sezione 3.1.1, il flusso video *live* che giunge su *Unity* dal *server RTMP* è soggetto a latenza: si tratta di un "ostacolo" inevitabile, almeno attraverso l'utilizzo della libreria *UMP*.
Oltre a questo, *OpenPose* ha ovviamente dei tempi di elaborazione, più o meno grandi a seconda delle impostazioni utilizzate.
Ne consegue che i *keypoints* trovati e renderizzati non saranno, in condizioni standard, mai davvero sovrapposti in tempo reale al corpo della persona sul video 360°.
Il flusso video viene processato e renderizzato sull'oggetto sfera tramite l'*URL* al *server RTMP* (Fig. 49, 50) e, allo stesso tempo, processato in parallelo dallo *script OpenPose* (Fig. 71): questo implica che si possa "giocare" con le impostazioni di *pose estimation* per mitigare questo effetto, cercando dei compromessi.
Andiamo a vedere le variabili in gioco:

- *Net Resolution*: questo parametro indica la risoluzione con cui i *frames* del video *livestream* vengono processati dalla rete neurale.
  Si indica con multipli di 16 e ovviamente, accrescendo il suo valore, le *features* dei corpi verranno rilevate con più efficacia, ma la complessità computazionale aumenta.
- *Scales Number*: indica quante volte viene riscalata a risoluzioni più basse l'immagine di *input*.
  Utilizzando un numero superiore a 1, *OpenPose* potrà rilevare informazioni in modo più robusto, poiché i dettagli delle pose possono variare in base alla distanza, all'orientamento e alla risoluzione dell'immagine.
  Anche in questo caso, il tempo di elaborazione aumenta al crescere della variabile.
- *Number People Max*: quante persone possono venire rilevate contemporaneamente.
  Come intuibile, la complessità cresce proporzionalmente a questo numero.

- Dimensione dell'immagine in *input*: la risoluzione del flusso video *live*, che determina ovviamente più *pixel* da scansionare.
- *Realtime Processing*: un'impostazione che permette di *skippare* dei *frames* per mantenere gli *FPS* di elaborazione il più vicino possibile a quelli del *producer*.

Andando a testare varie configurazioni di questi parametri, è possibile trovare compromessi che migliorino la resa finale, come vedremo nella Sezione 3.4.

Le sezioni affrontate finora in questo Capitolo 3 hanno spiegato in che modo si possa ottenere un *compositing* efficace di oggetti sintetici su un video 360°, sfruttando in modo interattivo le informazioni fornite dal *livestream*.
La prossima sezione, abbondantemente anticipata all'interno dei capitoli precedenti, andrà a illustrare alcune scelte implementative che mirano a "tirare fuori" il massimo da ogni componente.

3.4 Scelte implementative

Questa sezione andrà a "ripassare" i vari aspetti trattati durante il capitolo corrente, illustrando le scelte tecniche che sono state impiegate e testate in uno *user test* diviso in due parti e un primo scenario implementativo, esplicati nel Capitolo 4 e 5.

3.4.1 Stereoscopia del livestream

Come anticipato nella sezione 1.3.3 del Capitolo 1, la videocamera 360° *Insta360 Pro* utilizzata per questo progetto di tesi consente di streammare il flusso video in stereoscopia, fornendo a *Unity* una doppia immagine equirettangolare (Fig. 35) che può essere renderizzata tramite due *shader* identici, ma con *tiling* e *offsets* differenti per considerare solo la parte superiore o inferiore del *frame*, come mostrato in Fig. 77.

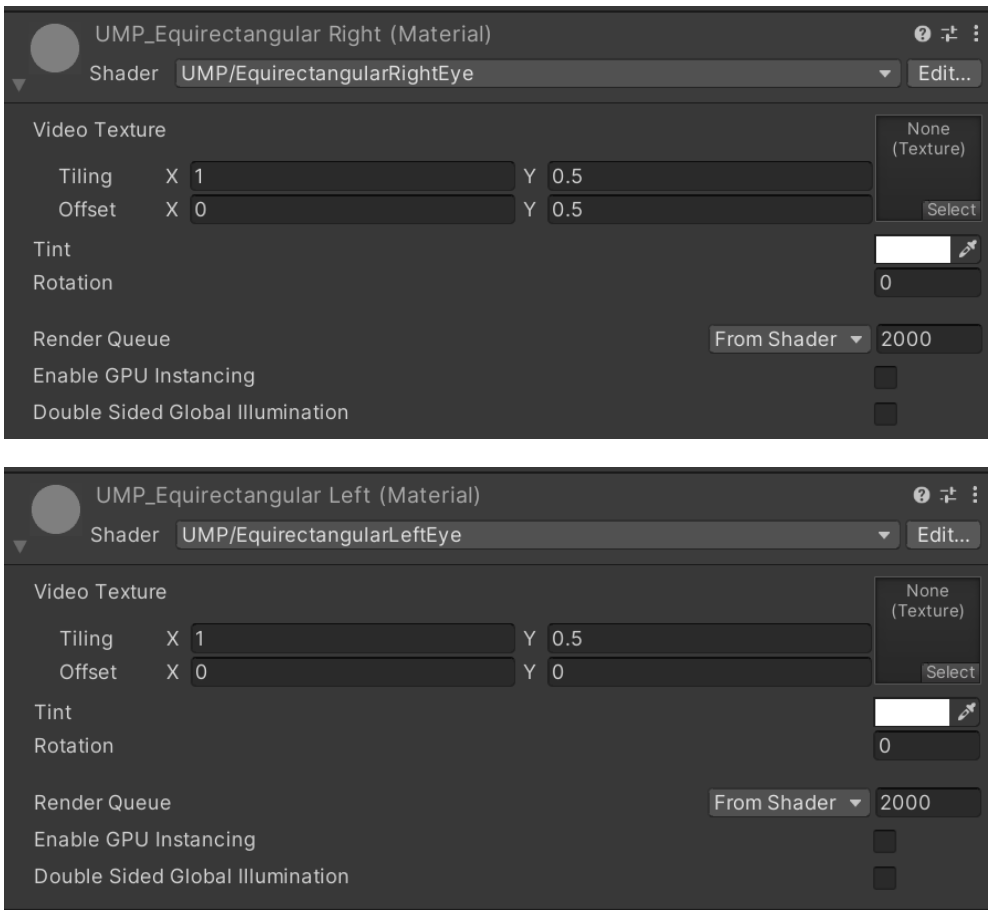

Fig. 77 I due shader che renderizzano occhio destro e sinistro su due oggetti sfera differenti.

Il risultato non è soddisfacente, probabilmente perché l'*autostitching* performato in *live* dalla camera non è preciso come quello che si può attuare *offline*: sulle linee di *stitching*, ovvero dove i contributi di due lenti si uniscono, l'effetto stereoscopico si va completamente a perdere.

Inoltre, la percezione della profondità non va a mitigare l'effetto di *parallax mismatch* illustrato nella Sezione 3.1.3.

Alla luce di queste problematiche, la scelta è stata quella di streammare sempre in modalità monoscopica.

3.4.2 Gestione della doppia camera

Come anticipato nella Sezione 3.1.5, è stata implementata una modalità di rendering "a camera doppia", per separare in *layers* differenti gli oggetti virtuali vicini (renderizzati in 3D) e quelli lontani (renderizzati in 2D).

Come vedremo nella prima parte dello *user test*, l'interazione con oggetti vicini risulta sensibilmente migliore rispetto a un approccio a camera singola monoscopica.

Resta da affrontare un problema di proiezione di ombre e di parziale *parallax mismatch*, come anticipato.

Gli oggetti situati sul *layer Objects* infatti non proiettano ombre sul *Floor*, in quanto questo si trova sul *layer Background* (Fig. 60).

Inoltre, se si trattasse di oggetti dotati di fisica o comunque in grado di avvicinarsi al terreno, questi fluttuerebbero in quanto situati sulla versione stereoscopica dell'oggetto *Floor*.

Nella prima parte dello *user test* questi spiacevoli effetti sono stati mitigati ponendo fra l'utente e il *Floor* sotto di lui un tavolo di lavoro molto ampio, in modo che fosse molto difficile per gli oggetti stereoscopici capitare sopra il *Floor*, come illustrato in Fig. 78.

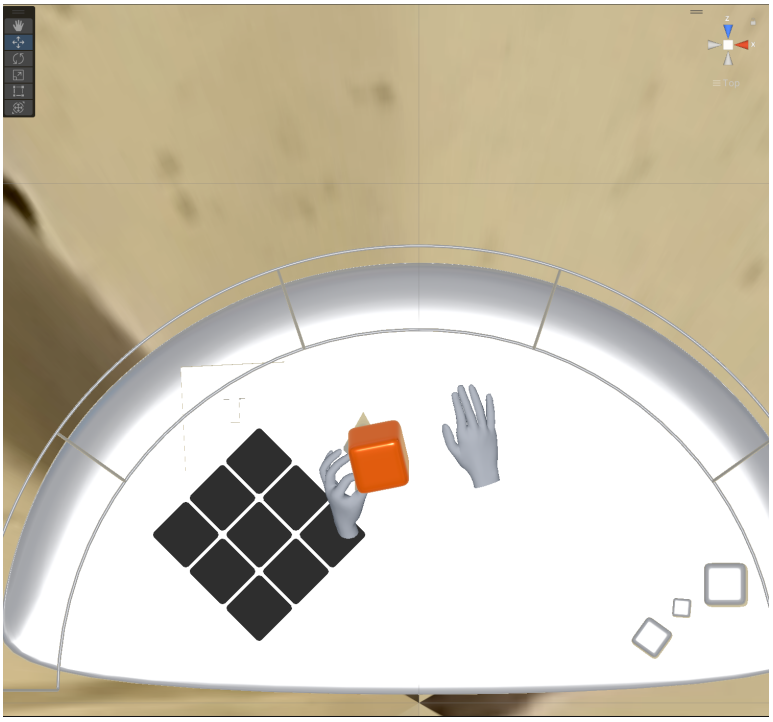

Fig. 78 Vista dall'alto dell'ampio tavolo da lavoro, che occlude quasi completamente il pavimento sottostante.

Per lo scenario implementativo, è stato tentato un approccio più “drastico”.
L’utente si trova in una stanza virtuale, un modello 3D, che lo circonda a 360°: attraverso una vetrata è possibile vedere il video 360° di sfondo, dando l’illusione di due mondi separati ma interconnessi (Fig. 79).
In questo modo, vengono ovviamente bypassati i problemi di proiezione ombre (gli oggetti stereoscopici proiettano ombre sul pavimento della stanza) e del *parallax mismatch* (il *Floor* non è visibile sotto l’utente ed è molto lontano in caso contrario).
Si tratta ovviamente di un *escamotage* piuttosto invasivo, ma di grande effetto visivo se gestito bene.

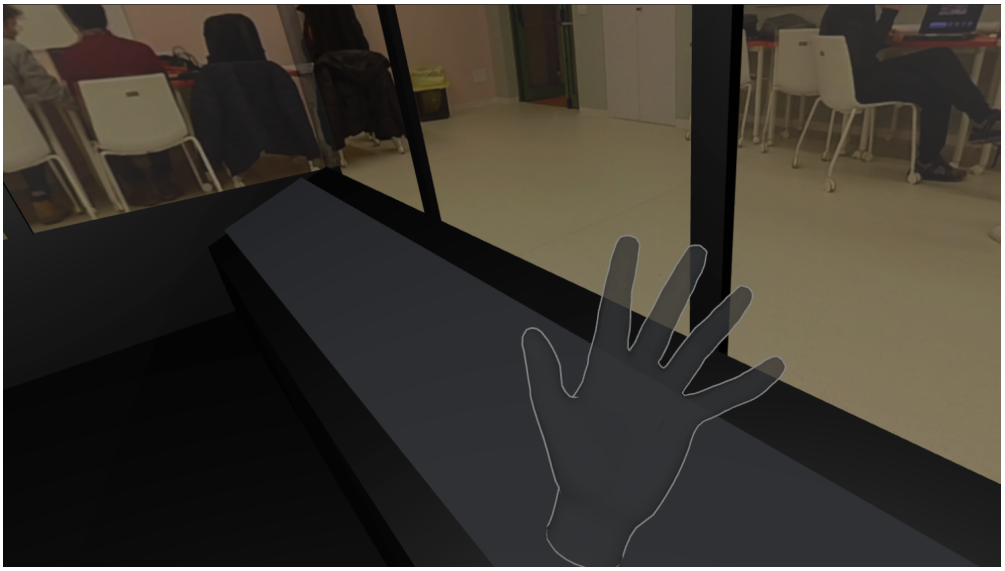

Fig. 79 La stanza in cui l’utente vive lo scenario implementativo: si ha l’impressione di essere in un laboratorio che si affaccia su una stanza reale.

Sempre nella prima parte dello *user test*, è stata testata anche un’alternativa completamente opposta a quelle presentate qui sopra: non vi sono interazioni con oggetti vicini, vi è una singola camera monoscopica e l’utente si ritrova con il *Floor* sotto di sé, senza barriere.
Il fatto di avere una sola camera monoscopica consente di proiettare le ombre dei modelli delle mani virtuali sul terreno, migliorando il senso di presenza; tuttavia le interazioni risultano meno naturali in quanto non prevedono il contatto diretto con gli oggetti.

I risultati ottenuti con questa configurazione sono comunque molto positivi e li analizzeremo nel Capitolo 4.

3.4.3 Compromessi sull'illuminazione

Come visto nella Sezione 3.1.6, è stato elaborato un algoritmo di *light detection* che però si scontra con delle limitazioni importanti.
Infatti, lo shader utilizzato per proiettare le ombre sul *Floor* invisibile supporta una sola luce direzionale per volta, mentre il motore di gioco *Unity* riesce a renderizzare solo due ombre per oggetto in contemporanea: vista l'impossibilità di fare *baking* delle luci, poiché gli oggetti in scena sono dinamici, si tratta di problematiche attualmente non risolvibili.
Le opzioni disponibili sono essenzialmente due:

- Illuminare l'ambiente reale in cui verrà performata l'esperienza *Presence* con una sola sorgente di luce, quindi:
    - Una *location outdoor*, dove il sole rappresenta l'unica sorgente (Fig. 63).
    - Una *location indoor* con una sola luce, direzionata in qualsivoglia modo.
- Utilizzare le ombre in modo non fotorealistico, ma soltanto come *depth cues* per migliorare la sensazione di *compositing*, posizionando quindi la luce direzionale a zenit, che punta quindi verso il basso.

La scelta è ricaduta sulla seconda opzione, poiché la prima sarebbe andata a creare dei vincoli evidentemente troppo stringenti.
Tramite la prima parte dello *user test* si è verificato come un'ombra proveniente dall'alto, nei frequentissimi casi in cui l'illuminazione della scena sia diffusa e non fortemente direzionale, "faccia il suo lavoro" e porti efficacemente l'illusione che l'oggetto si trovi nell'ambiente reale.
L'utilizzo di *reflection probes* migliora inoltre la resa finale del *compositing* degli oggetti metallici, fortemente riflettenti: i *probes* sono in grado di catturare tutti i *layers* della scena contemporaneamente e utilizzarli all'interno delle riflessioni degli oggetti metallici, come illustrato in Fig. 80.

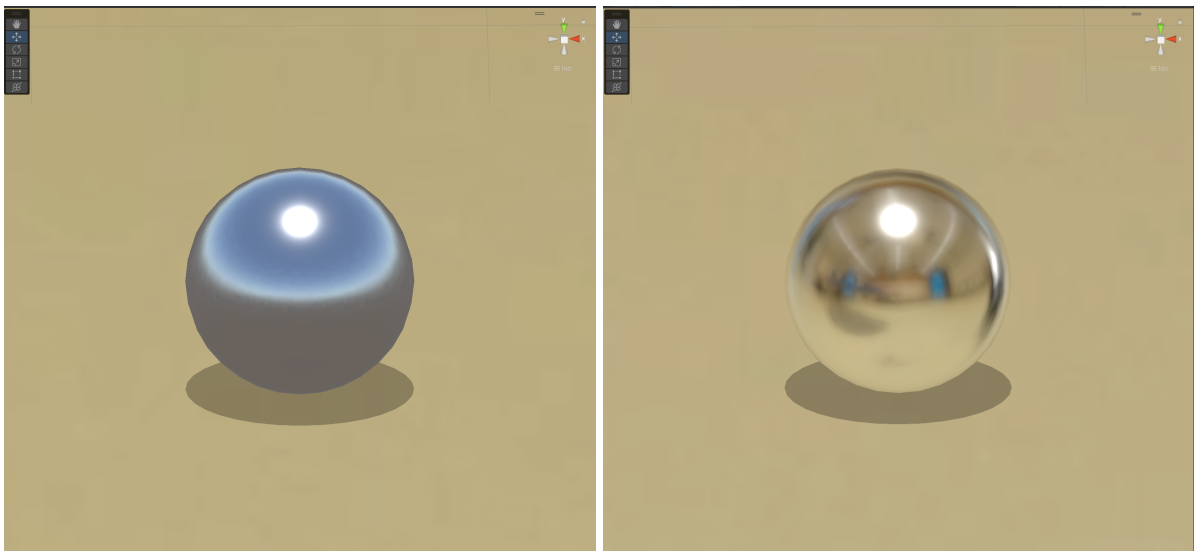

Fig. 80 A sinistra, un oggetto metallico se in scena non c'è un reflection probe; a destra le riflessioni generate grazie al probe.

3.4.4 L'*embodiment*

Come visto poc'anzi, il *virtual embodiment* è senza dubbio una *feature* interessante, che accresce le possibilità del progetto *Presence* per sperimentare sulla moltiplicazione del sé.

Nella Sezione 3.2.1 sono state evidenziate delle limitazioni per quanto riguarda l'approccio *first-person*: il modello 3D dell'*avatar* può non avere le stesse identiche dimensioni degli arti dell'utente e la cinematica inversa che ricostruisce il movimento della catena braccio-avambraccio-mano non può rappresentare correttamente la posizione reale dell'utilizzatore.

Se si volesse scegliere di attuare l'approccio a camera singola, inoltre, si dovrebbe rinunciare al tracciamento della posizione dell'utente (vedere Sezione 3.1.4), e quindi a piccoli movimenti delle mani corrisponderebbero grandi traslazioni degli arti, vanificando ancora di più il realismo dell'*embodiment*.

Si è quindi scelto, nella prima parte dello *user test* e nello scenario implementativo, di utilizzare l'*avatar third-person*, seppur vi sia l'assenza del tracciamento delle dita (Sezione 3.2.2): questa limitazione può essere risolta a livello narrativo, proponendo un modello 3D dell'*avatar* semplificato oppure volutamente poco "umano", come nel caso del manichino in Fig. 68, utilizzato in questo progetto.

3.4.5 Applicazione di *OpenPose*

Nella Sezione 3.3 è stato spiegato, a grandi linee, il funzionamento della libreria *OpenPose* per *Unity* e sono state illustrate le numerose variabili che entrano in gioco per poter ridurre l'inevitabile latenza presente.
Tuttavia, non è stato ancora spiegato in che modo usufruire efficacemente delle informazioni ottenute dal *plugin*.
Si tratta di fare dei compromessi fra le variabili che sono state definite nella Sezione 3.3.2, al fine di ottenere un risultato vicino a quello che è l'obiettivo.

Scelta delle impostazioni

Tutto passa da il *bool* chiamato *Realtime Processing* che, come accennato in precedenza, permette di far saltare l'elaborazione di alcuni *frames* in *input* al fine di mantenere gli stessi *FPS* del *livestream*.
Settando questo *flag* come *true*, l'intera implementazione *OpenPose* risulta più leggera e veloce: gli *FPS* della *PlayMode* di *Unity* si mantengono abbondantemente sopra i 70, tenendo conto dell'hardware utilizzato (Laptop Alienware 17 R5).
Inoltre, è possibile alzare di molto la *Net Resolution*, che è una variabile chiave per una *pose estimation* più efficace, in quanto indica la risoluzione dei *frames* dati in *input* alla libreria.
Vi sono due dettagli non trascurabili, ovvero la fluidità dell'aggiornamento dei *keypoints* e la velocità di esecuzione del *plugin*:

- Fluidità di aggiornamento: i *keypoints* non si aggiornano con la frequenza necessaria per far sembrare il loro movimento come "fluido", dal momento che, come accennato, molti di loro vengono saltati per mantenere gli *FPS*.
- Velocità di esecuzione: come già detto, il flusso video viene processato in uno script differente da quello responsabile del suo *rendering*.
  In questa circostanza di *Realtime Processing*, *OpenPose* riesce addirittura a calcolare la posa in anticipo rispetto al rendering del video, quindi sembra che i *keypoints* "siano nel futuro" (Fig. 81).

Se invece il *bool* è settato su *false*, verranno processati tutti i *frames* del *livestream*, andando ad appesantire notevolmente il sistema: gli *FPS* possono scendere fino a sotto i 30.

Ovviamente è necessario abbassare la *Net Resolution*, causando problemi di rilevamento delle persone, soprattutto se lontane dalla camera.

Andando a rivalutare i punti esposti prima:

- Fluidità di aggiornamento: siccome ogni *frame* viene processato, i *keypoints* si aggiornano al *rate* scelto, quindi il loro movimento è decisamente fluido.
- Velocità di esecuzione: *OpenPose* fa, in questo caso, un'enorme fatica a processare il *livestream*, causando un ritardo considerevole nell'aggiornamento dei dati (Fig. 81).

  Nella seconda parte dello *user test* viene testata la sopportabilità di questo ritardo, con esiti prevedibili.

Fig. 81 Sopra: i keypoints se il bool Realtime Processing è settato su true. Sotto: l'importante ritardo se il bool è false.

Nel complesso, si ritiene che il *flag Realtime Processing* debba sempre essere *true* se si vuole una maggior aderenza all'aspetto temporale dell'esperienza, mentre può essere settato su *false* se si vuole una maggiore fluidità del tracciamento, considerando che la persona deve essere piuttosto vicina alla camera e il tutto non sarà in tempo reale.

Ora verranno illustrate ora alcune applicazioni, che sono poi state messe alla prova nella seconda parte dello *user test* e nello scenario implementativo.

Oggetti statici

Il modo più semplice per sfruttare i dati forniti da *OpenPose* è sicuramente il tracciare un corpo umano statico, non in movimento.
In questo modo la latenza non è percepibile, in quanto i *keypoints* sono fissi nello spazio.
Nello scenario implementativo è stato utilizzato questo approccio per tracciare la posizione statica dell'utente e poi far comparire una sfera di fumo nel punto rilevato (Fig. 82).

Fig. 82 Lo scenario implementativo inizia in VR e la sfera di fumo si trova nel punto in cui l'utente è situato all'interno della stanza reale.

Nella seconda parte dello *user test*, è stata invece sovraimpressa una maschera al volto della persona tracciata, che ovviamente rimane ferma (Fig. 83).
Come mostreranno i risultati del test, l'effetto è convincente.

Fig. 83 Una maschera sovraimpressa al volto della persona.

In Fig. 76 è stato illustrato un terzo utilizzo delle condizioni statiche, dove i *keypoints* vengono sovraimposti al corpo della persona.
Si tratta di un effetto non adatto, ora come ora, ad un'implementazione narrativa complessa (come, ad esempio, utilizzare i punti per creare la posa di un modello umano 3D): come già mostrato in Fig. 75, i *keypoints* sono mappati in modo piatto su una sfera e non contengono informazioni sulla profondità.
L'unico stratagemma che si può attuare è diminuire il raggio della sfera su cui vengono distribuiti i punti: in questo modo, se è nota la distanza del *performer* dalla camera, è possibile utilizzare il dato proprio per pilotare il raggio della sfera.
L'effetto è che i *keypoints* saranno posizionati all'interno della scena 3D (Fig. 84) e potranno essere utilizzati, come vedremo fra poco, anche per pilotare oggetti in modo dinamico.

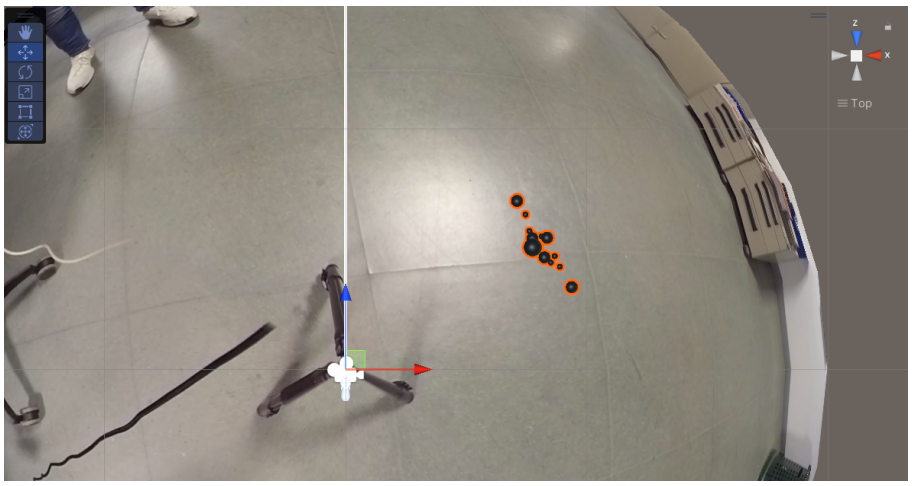

Fig. 84 I keypoints sono mappati su una sfera di raggio ridotto e quindi acquisiscono un'informazione di profondità.

Oggetti dinamici

Questo tipo di implementazione prevede di scendere a compromessi con le variabili descritte in precedenza.

Sia che *Realtime Processing* sia *true* o *false*, è impossibile immaginare di vedere i *keypoints* sovrapposti in tempo reale alla persona in movimento, come illustrato in Fig. 81.

Nel caso del *bool* settato su *false* il ritardo di elaborazione sarà troppo, nel caso opposto i *keypoints* si muoveranno in anticipo e a scatti.

Tuttavia, è possibile tentare un approccio che non dipenda così tanto dalla sovrapposizione precisa dei punti sul corpo della persona.

Come vedremo nella seconda parte dello *user test*, è possibile sfruttare il movimento dei *keypoints* per creare un *target* che un oggetto in scena possa seguire.

L'oggetto in questione può essere di qualsiasi tipo: nel nostro caso è stato scelto il modello 3D utilizzato per il *virtual embodiment*.

Il punto definito come *target* è invisibile, quindi non influenza la percezione dello spettatore con eventuali ritardi o anticipi.

Uno *script* calcola in tempo reale la distanza fra la posizione del *target* (che ricordiamo essere tridimensionale come visto in Fig. 84) e la posizione del modello 3D e, se questa supera una certa soglia, attiva un'animazione di camminata verso il punto dello spazio del *target*: come dimostrano i risultati, la resa è più che soddisfacente.

Fig. 85 Il modello umanoide sembra correttamente muoversi verso il punto nello spazio 3D dove si trova il performer.

È stato tentato anche di modificare in modo dinamico il raggio della sfera su cui vengono mappati i *keypoints* (Fig.84) sfruttando il metodo indicato nel *paper* [39] analizzato nella Sezione 2.2.2: è infatti possibile, in linea teorica, utilizzare la coordinata y del *keypoint* di uno dei piedi all'interno della formula che rileva la distanza della persona dalla camera (Fig. 46).
All'atto pratico, l'*output* fornito da *OpenPose* non è abbastanza stabile e preciso da consentire questa applicazione: errori di pochi *pixel* possono portare ad un'errata valutazione della distanza della persona.

Concludendo, la libreria *OpenPose* può essere sfruttata con efficacia in numerosi modi, evitando con astuzia i limiti noti e sfruttando alcuni stratagemmi implementativi.

Il capitolo dedicato alla metodologia impiegata per questo progetto di tesi può considerarsi terminato.
Tutte le scelte implementative spiegate in questa ultima sezione sono state "messe alla prova", come già detto, in uno *user test* diviso in due parti e uno scenario implementativo, che vedremo rispettivamente nel prossimo capitolo e in quello dopo.

4. User Test

Al fine di validare le scelte implementative illustrate nella Sezione 3.4, è stato condotto un test con utenti.
Questo test è stato somministrato in due parti, a 14 persone, nell'aula 8N del Politecnico di Torino.
Queste verranno ora illustrate, definiti i loro obiettivi e commentati i risultati.

4.1 Prima parte

L'intento di questa prima parte è stato quello di valutare le possibilità implementative dell'approccio a camera doppia, visto nella Sezione 3.1.5, e metterlo a confronto con una classica fruizione a camera singola.
In particolare, si è andato a esplorare quanto il cambio di configurazione potesse andare a influenzare il senso di presenza dell'utente, la sua efficacia di interazione con oggetti sintetici, la percezione complessiva del *compositing* e il suo livello di auto-coscienza e auto-localizzazione.
Il *pool* di domande che è stato somministrato per questa prima parte è composto da contributi provenienti sia da *VRUSE* [53], un questionario introdotto nel 1999 per misurare l'usabilità dei sistemi *VR*, che da domande utilizzate dal *team* di *MR360* [31] per valutare la qualità del *compositing* 360°, ma anche da quesiti contenuti nel *paper* "Spatial aspects of bodily self-consciousness" [54], volti a studiare quanto il visualizzare il proprio corpo da una prospettiva differente possa influenzare la propria auto-coscienza corporea.
Le domande sono state poste a voce, durante l'esecuzione dei *task*, e la risposta è stata data verbalmente e in tempo reale.
Ogni quesito prevede una risposta da 1 a 5, con 1 che rappresenta "per niente" e 5 "moltissimo".

È stato creato un ambiente che prevedesse un ampio tavolo da lavoro di fronte all'utente (Fig. 78) e numerosi oggetti virtuali situati lontano da esso.
Sul tavolo da lavoro, in particolare, sono presenti:

- Cubi di dimensione crescente, manovrabili tramite il gesto di *pinch* delle mani.
- Una forma poligonale tridimensionale e una lastra metallica con una fessura della stessa forma: queste, se utilizzate con entrambe le mani, possono essere scalate per cambiarne la dimensione.
- Una pulsantiera con nove bottoni, premibili con l'indice della mano.

Nell'ambiente attorno, invece, si trovano:

- Il gioco "acchiappa la talpa", con il quale si interagisce con la pulsantiera sopracitata: ogni tasto corrisponde a uno spazio in cui può apparire una talpa.
- Un *avatar* comandato dall'utente in terza persona (approccio illustrato nella Sezione 3.2.2), con il quale è possibile toccare delle sfere che cadono dal cielo.
- Elementi sintetici per testare la qualità del *compositing*: due portali *fantasy*, una sfera riflettente, un cubo trasparente, un modello umano che vola.

Nella Fig. 86 sono mostrati gli oggetti elencati sopra.

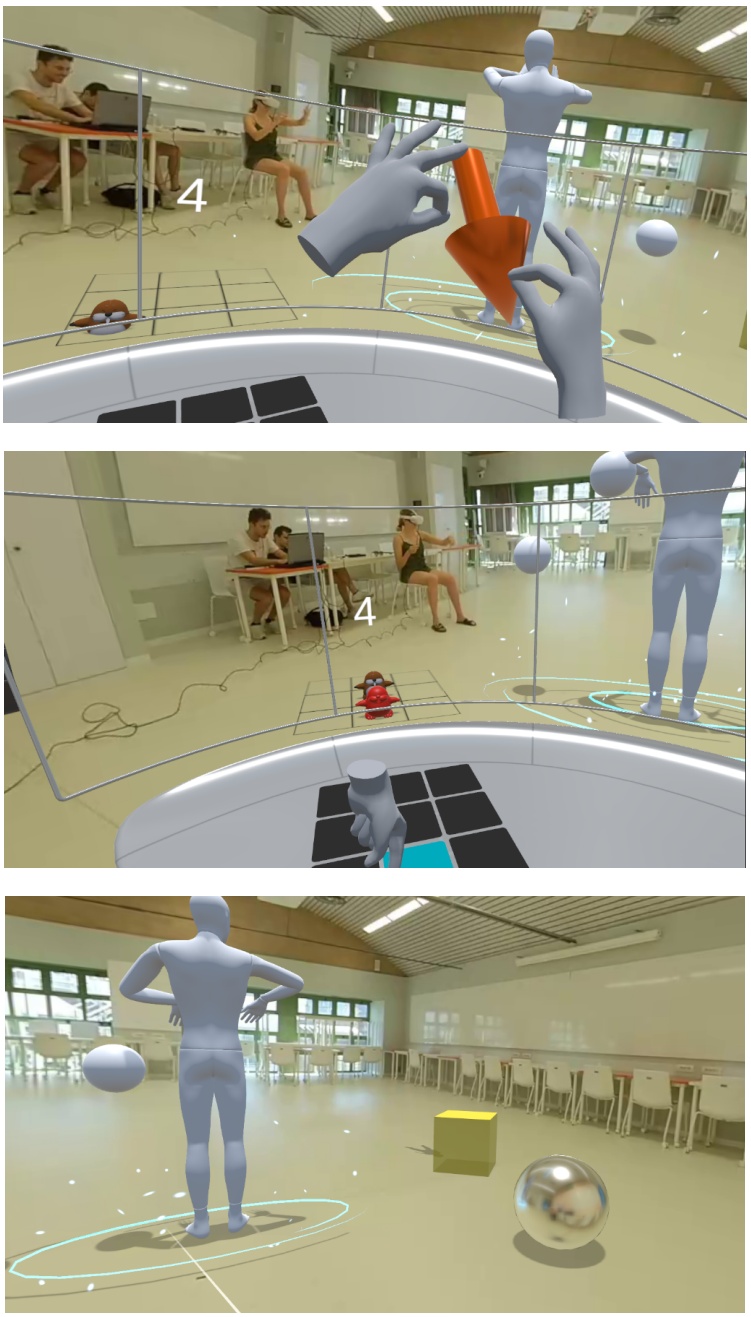

Fig. 86 Dall'alto al basso: l'oggetto poligonale scalabile, la pulsantiera con cui giocare ad "acchiappa la talpa", l'avatar tracciato con due oggetti sintetici.

Questa prima parte del test è stata divisa in quattro fasi che, come accennato, vanno a indagare aspetti differenti e viene ripetuta tre volte, ognuna con una configurazione diversa:

- Configurazione 1: sfrutta l'approccio a camera doppia (Sezione 3.1.5, Fig. 57) e quindi gli oggetti vicini (tavolo di lavoro e mani virtuali) vengono visualizzati in stereoscopia, mentre tutto ciò che è sullo sfondo è monoscopico.
- Configurazione 2: utilizza una singola camera monoscopica e quindi tutti gli oggetti sintetici (comprese le mani virtuali) vengono mostrati in 2D.
- Configurazione 3: anticipata nella Sezione 3.4.2, prevede una singola camera monoscopica ma anche l'assenza del tavolo da lavoro e, quindi, delle interazioni dirette da parte dell'utente.

  In questo caso il gioco "acchiappa la talpa" viene eseguito tramite un *ray interactor*, facendo il gesto di *pinch* con la mano (Fig. 87).

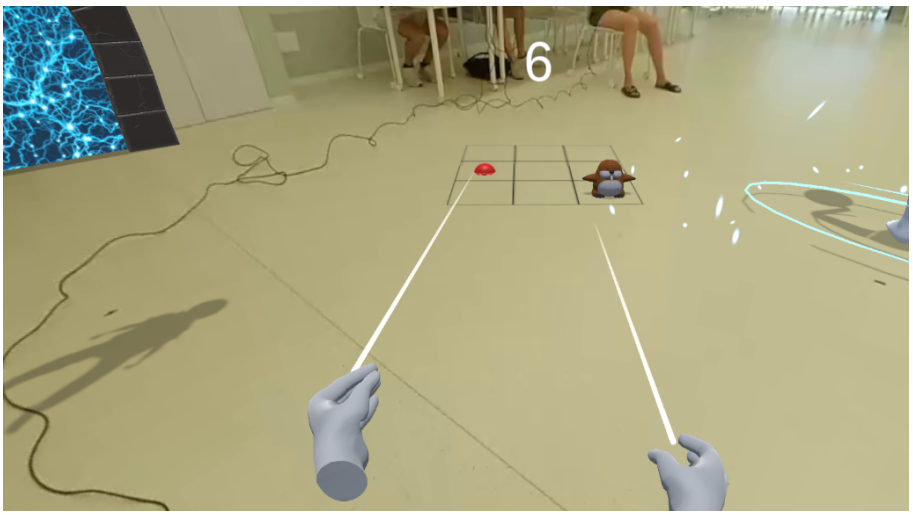

Fig. 87 La configurazione 3 prevede che le interazioni con le talpe avvengano tramite un ray casting.

Le prime due configurazioni sono state sottoposte agli utenti in ordine casuale, al fine di proporre un'esperienza diversa e non influenzare la valutazione finale.

Le quattro fasi sopracitate sono state invece affrontate con un ordine preciso e vanno a esplorare gli aspetti ritenuti più importanti fra quelli studiati: verranno ora spiegate singolarmente e commentate.

4.1.1 Fase 1 - Senso di presenza e immersione

L'utente inizia l'esperienza trovandosi immerso nell'ambiente 360° che riprende, in tempo reale, la stanza in cui si trova in quel momento.
Viene dapprima invitato a esplorare il banco da lavoro davanti a sé e a condurre delle piccole prove con ogni elemento presente, poi a visualizzare gli oggetti distanti (che vengono elencati) e provare l'*avatar* pilotato dalle proprie mani.
In questa prima fase è stato chiesto di focalizzarsi sul proprio corpo virtuale, rappresentato unicamente dai modelli 3D delle mani tracciate, e rispondere a delle domande.
È stato quindi sottoposto un primo blocco di 5 quesiti, estratto da *VRUSE Factor 9 (Sense of Immersion/Presence)* e *Factor 3 (System Output)*, in cui si interroga l'utente sul suo senso di presenza e immersione durante l'esperienza.
Le domande spaziavano da "Sto percependo un senso di presenza (ovvero, "mi sento lì")" a "Sto percependo, all'interno dell'ambiente, un corretto senso delle proporzioni".

Risultati

I risultati sono nel complesso positivi per tutte e tre le configurazioni, come mostrato in Fig. 88a.

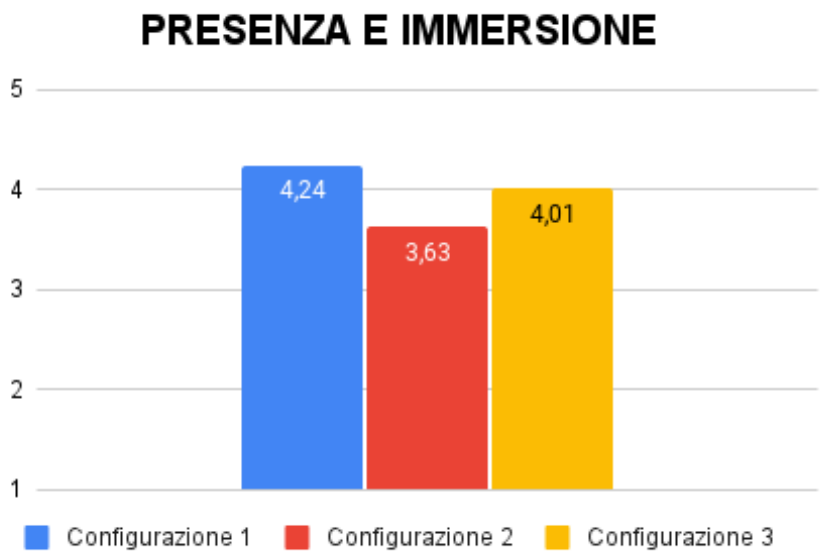

Fig. 88a La media arrotondata delle 5 domande sul senso di presenza e immersione.

La Configurazione 1 risulta la più efficace, soprattutto se confrontata con la Configurazione 2 sul quesito Q4 "Sto percependo, all'interno dell'ambiente, un buon senso delle proporzioni" (Fig. 88b): la motivazione, più che evidente, è che l'approccio a camera doppia prevede la visualizzazione 3D anche delle mani virtuali, le quali vengono tracciate con maggiore precisione grazie ai 6-*DOF* della camera stereoscopica (Fig. 59a) e risultano correttamente proporzionate.

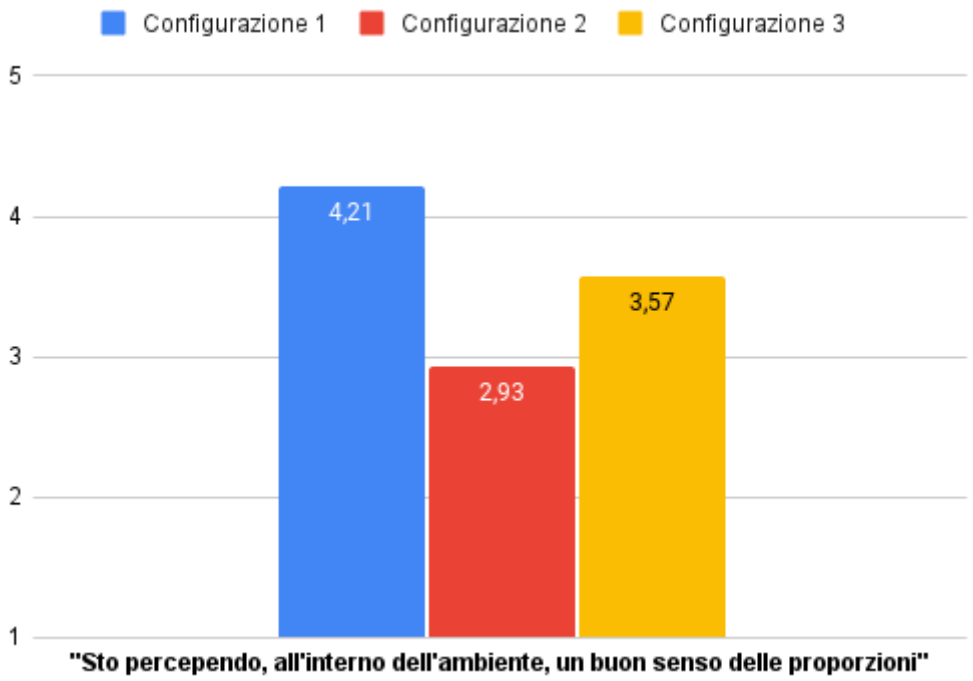

Fig. 88b Domanda Q4.

Dalla Fig 88a risulta anche una notevole efficacia della Configurazione 3, che addirittura supera la Configurazione 1 nella domanda Q3 "Mi sento isolato/a e non parte dell'ambiente virtuale" (Fig. 88c): probabilmente l'assenza del tavolo da lavoro ha rimosso un elemento distraente e ha permesso all'utente di sentirsi più immerso nell'ambiente.

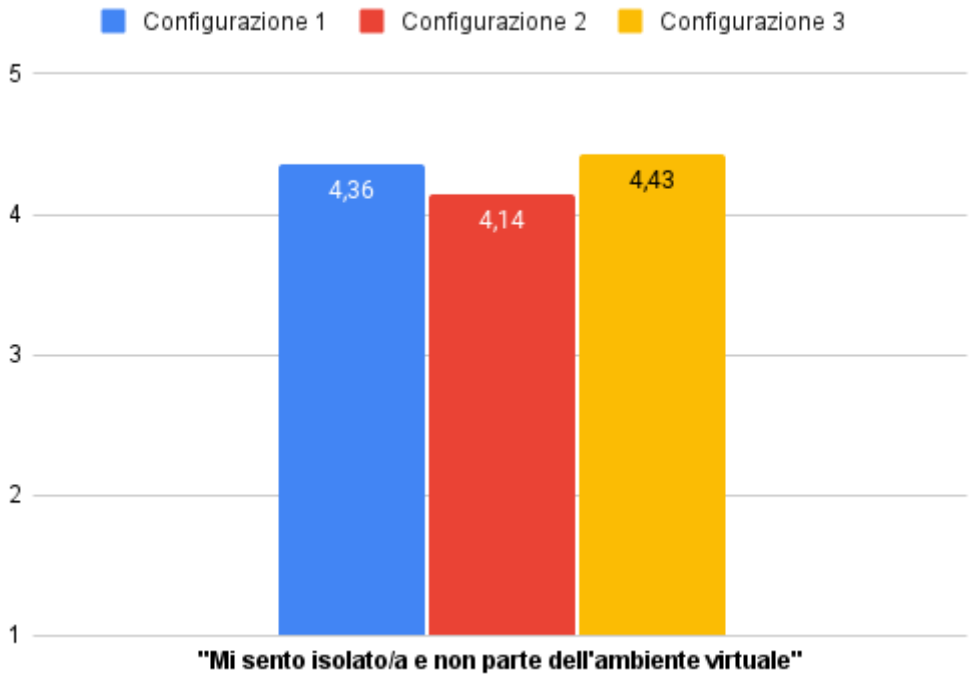

Fig. 88c Domanda Q3. Il risultato è stato invertito per mantenere la coerenza con le restanti domande e quindi un valore alto corrisponde a una valutazione positiva.

4.1.2 Fase 2 - Efficacia dell'interazione

Terminata la Fase 1, l'utente è stato invitato ad eseguire alcuni *task*:

- Fare un determinato punteggio ad "acchiappa la talpa".
- Impilare i cubi in modo crescente.
- Far passare la forma poligonale nella fessura della lastra metallica.
- Toccare le sfere che cadono dal cielo comandando le mani dell'*avatar.*

Durante questo processo è stato sottoposto il secondo blocco di 5 quesiti, provenienti da *VRUSE Factor 10 (Overall System Usability)*, in cui si interroga l'utente sulla facilità e il piacere dell'utilizzo del sistema.
Alcuni esempi di domande sono "Sto trovando difficile imparare a usare il sistema" oppure "Mi sento in controllo del sistema".

Come mostra la Fig. 89a, la Configurazione 1 si è di nuovo elevata sopra le altre.

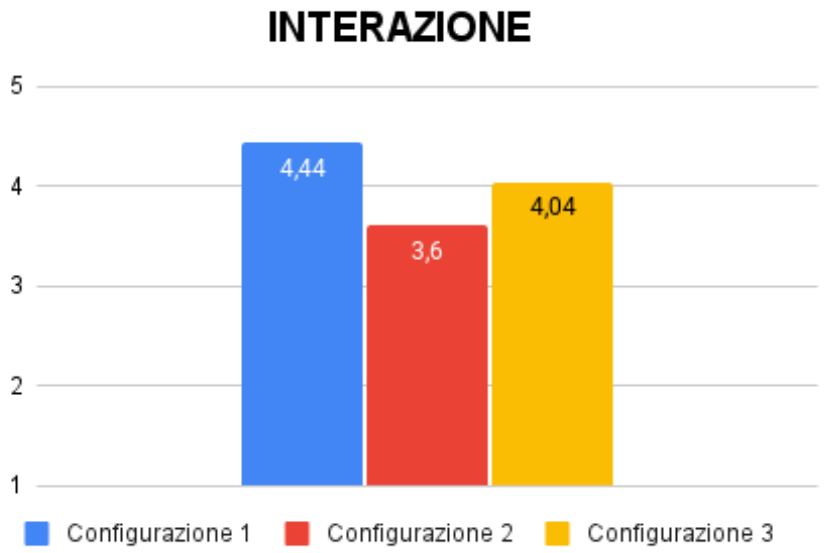

Fig. 89a La media arrotondata delle 5 domande sull'efficacia dell'interazione.

Questo risultato era già stato auspicato nella Sezione 3.1.5, dove si ipotizzava che visualizzare gli oggetti vicini con una camera stereoscopica avrebbe facilitato di molto l'interazione con essi.
Lo dimostrano le risposte al quesito Q2 "Mi sento in controllo del sistema", in Fig. 89b, dove la Configurazione 1 stacca di più di un punto la Configurazione 2.
L'utente si sente addirittura più in controllo del sistema nella Configurazione 3, dove non sono presenti interazioni vicine, rispetto all'avere un tavolo da lavoro visualizzato in monoscopia.

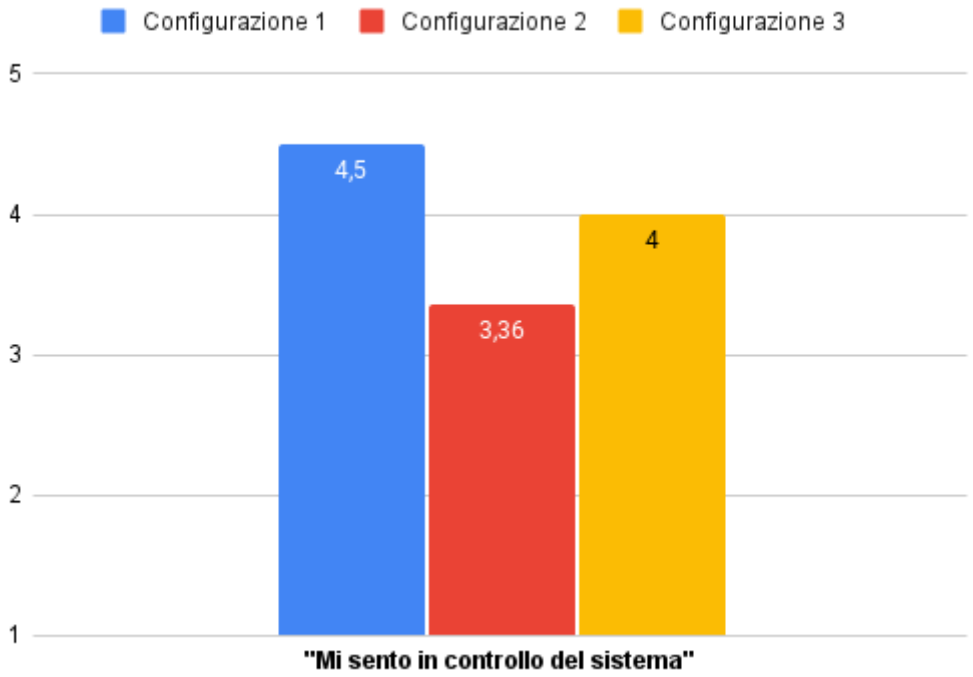

Fig. 89b Domanda Q2

Risulta interessante commentare l'esito della domanda Q5 "Credo che sia confortevole fare uso di questo sistema per lunghi periodi di tempo", in Fig. 89c.
Si può notare come la tendenza rimanga uguale, con una preferenza per la Configurazione 1, ma i numeri siano più bassi: il fatto di indossare un *HMD* per le prima volta (per molti degli utenti) ha sicuramente influenzato l'esito, ma è anche probabilmente dovuto all'utilizzo delle mani tracciate, più intenso dell'uso di un comune *controller*.

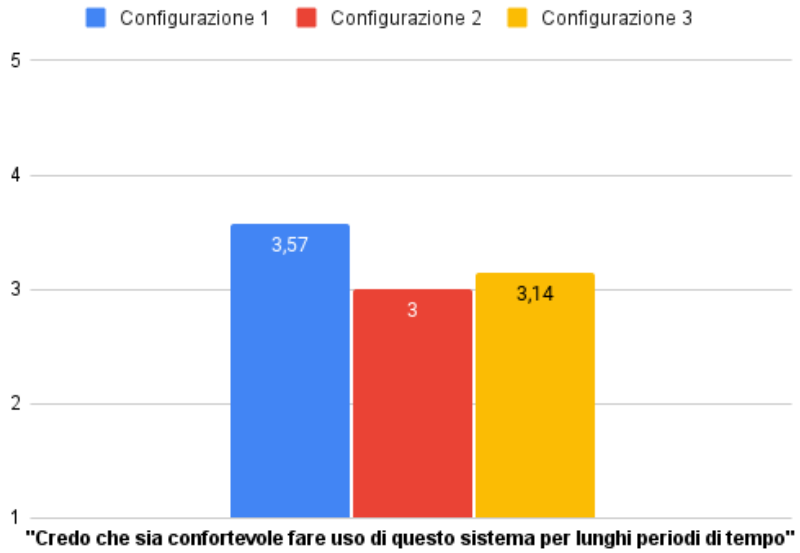

Fig. 89c Domanda Q5

4.1.3 Fase 3 - Compositing

Per questa fase l'utente è stato invitato a focalizzarsi sul rapporto fra il video 360° di sfondo e i vari oggetti sintetici, al fine di valutare la qualità complessiva del *compositing*.

Il terzo blocco di quesiti ha utilizzato le 3 domande che sono state sottoposte agli utenti per lo *user test* di *MR360* [31], il sistema che più si avvicinava a quello ideato per questo progetto di tesi, al fine di fare un confronto.

Si tratta di:

- Q1: "Mi sembra di essere situato/a nell'ambiente del video 360°"
- Q2: "Mi sembra che gli oggetti virtuali siano situati nello stesso ambiente del video 360°"
- Q3: "Mi sembra che la qualità del *compositing* sia nel complesso verosimile"

La scala lineare è stata convertita da [1,5] a [-2,2], al fine di poter confrontare al meglio i risultati con quelli di *MR360*.

Nella Fig. 90a è possibile vedere i risultati ottenuti da questo *user test*, nella Fig. 90b i risultati di *MR360*.

MR360 considerava due momenti, uno con e uno senza interazione: nel nostro *user test* l'interazione è sempre presente.

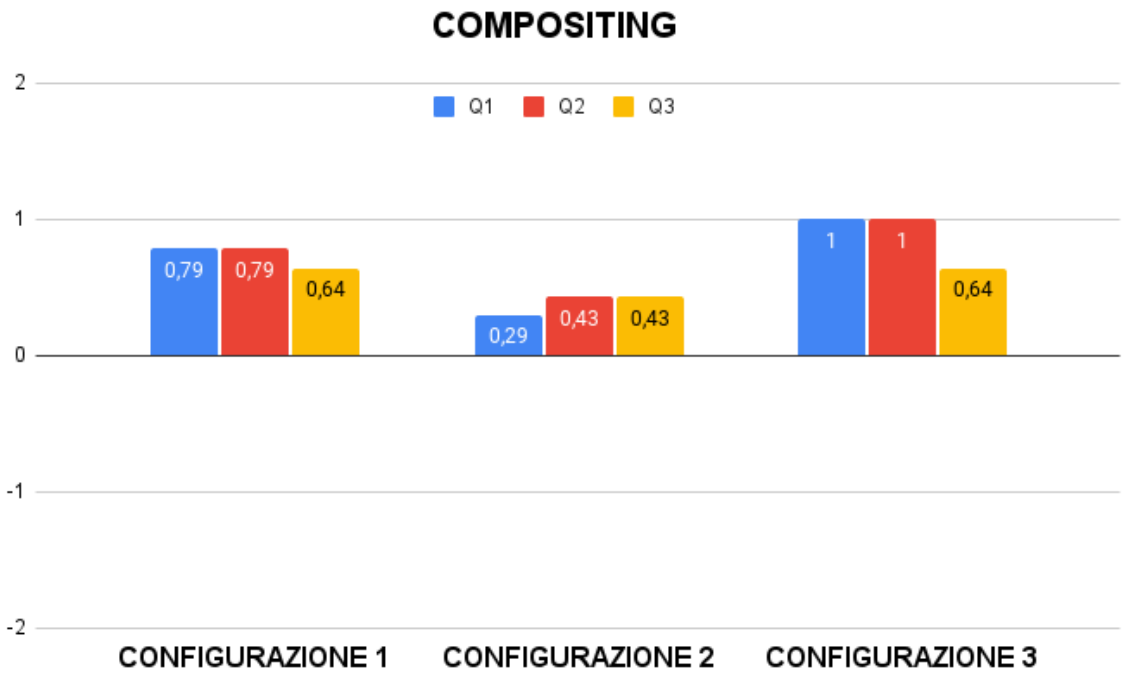

Fig 90a La media arrotondata delle 3 domande sul compositing, divisa per configurazioni.

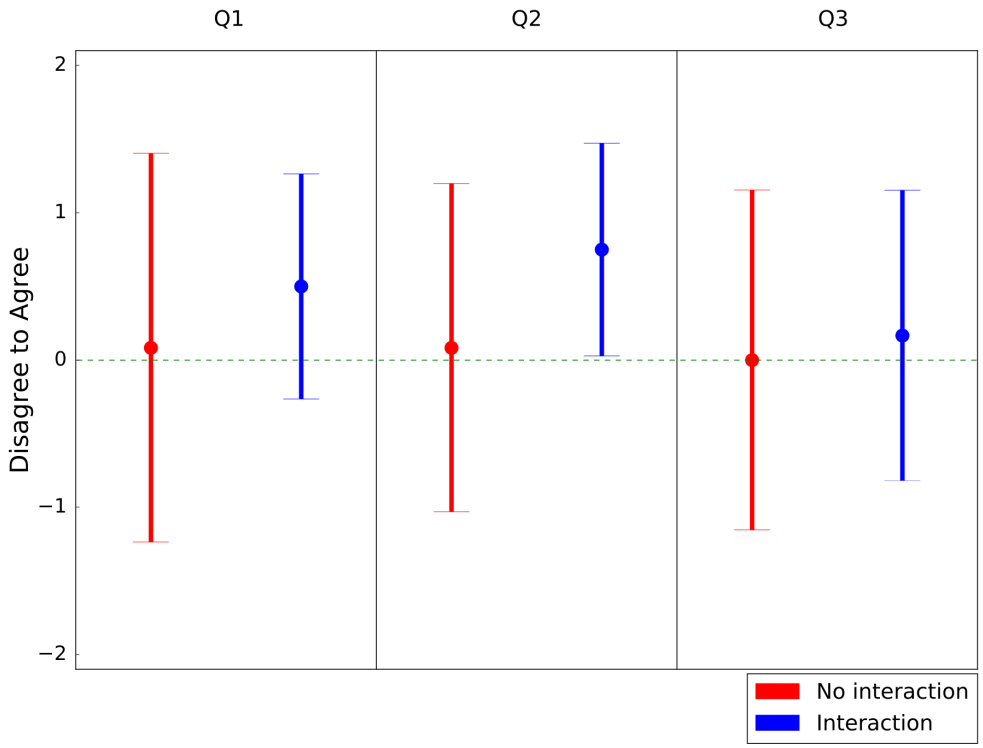

Fig 90b I risultati di MR360. - Immagine presa dal paper in questione

Le Configurazioni 1 e 3 raggiungono e superano i risultati di *MR360* per tutte e tre le domande, mentre la Configurazione 2 raggiunge punteggi minori, ma comunque vicini.

Questo esito porta ad una constatazione: la Configurazione 2 era stata creata per facilitare il *compositing* degli oggetti vicini con il *floor*, in quanto questi proiettano le ombre a terra (Fig. 60) e non hanno problemi di *parallax mismatch*, come illustrato nella Sezione 3.1.5, ma gli utenti non hanno notato questo cambiamento, addirittura valutando in modo maggiormente negativo la qualità generale.

Vale la pena notare che, per la domanda Q1 “Mi sembra di essere situato/a nell’ambiente del video 360°” la Configurazione 3 ha superato la Configurazione 1 (Fig. 90a), come già era successo per la Q3 della Fase 1: sembra sempre più confermata l’idea che l’assenza di un tavolo da lavoro possa favorire l’immersione nell’ambiente virtuale.

4.1.4 Fase 4 - Auto-coscienza corporea

Quest'ultima fase ha visto l'utente focalizzarsi principalmente sulla presenza del suo corpo reale all'interno dell'ambiente 360°: per tutto il tempo trascorso durante questa parte dello *user test*, infatti, la persona è sempre stata inquadrata dalla camera 360°. Colui che ha somministrato fino ad ora il questionario si alza e inizia a toccare periodicamente il braccio disteso dell'utente, al fine di creare uno stimolo visivo-tattile (Fig. 91).

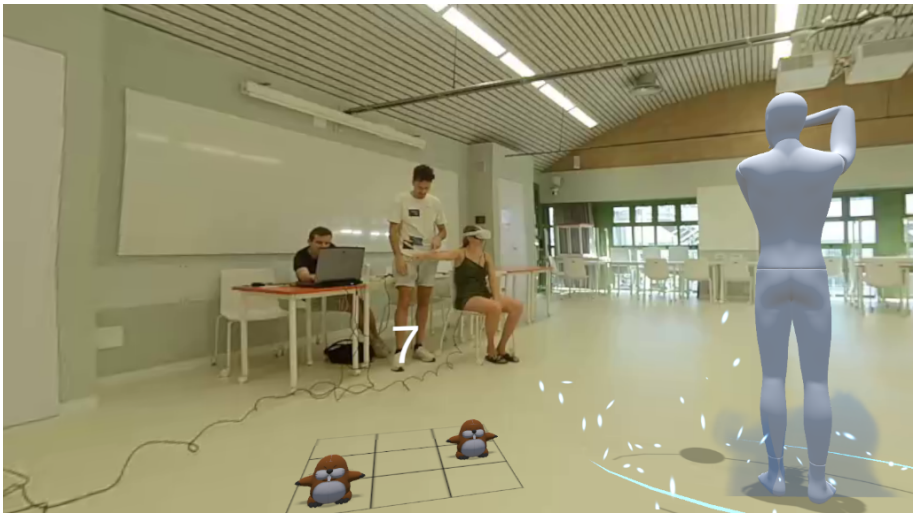

Fig. 91 L'utente viene toccato sul braccio mentre gli vengono sottoposte delle domande.

Le domande che sono state poste all'utente sono 5, prese da un *paper* [54] che ha studiato come il livello di autocoscienza possa variare in base al conflitto di input multisensoriali.
Infatti, sfruttando la latenza che è presente all'interno del *livestream* (Sezione 3.1.1), è stato possibile fornire proprio degli stimoli in disaccordo fra di loro e si è voluto curiosare sugli effetti di questa condizione.
Alcune domande sono state volontariamente scelte per "fallire", come la Q4 "Quanto ti sembra che il tocco che stai percependo si trovi dove tu stai vedendo il tocco?", altre per testare se fosse possibile creare un senso di dissociazione corporea a causa della latenza, come la Q3 "Quanto ti sembra che tu sia dissociato/a dal tuo corpo?".

Come si evince dalla Fig. 92a, i risultati non sono particolarmente entusiasmanti: la media è, per tutte le configurazioni, attorno a 2.5/5.
Andando ad analizzare alcune domande nello specifico si può notare come, in parte, le ipotesi espresse trovino un riscontro: la domanda Q3 "Quanto ti sembra che tu sia dissociato/a dal tuo corpo?" è quella con una valutazione media più alta, mentre la Q4 "Quanto ti sembra che il tocco che stai percependo si trovi dove tu stai vedendo il tocco?" ha la media più bassa (Fig. 92b).

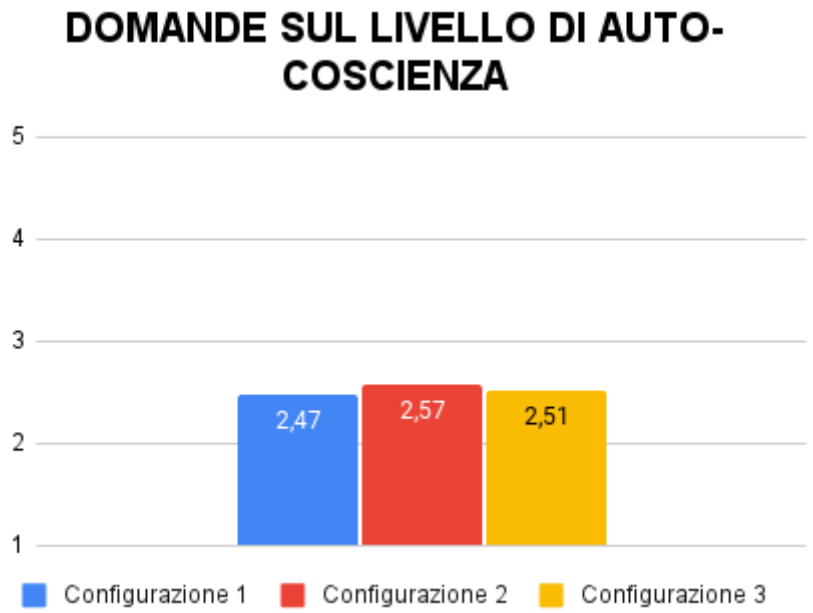

Fig. 92a La media arrotondata delle 5 domande sul livello di auto-coscienza.

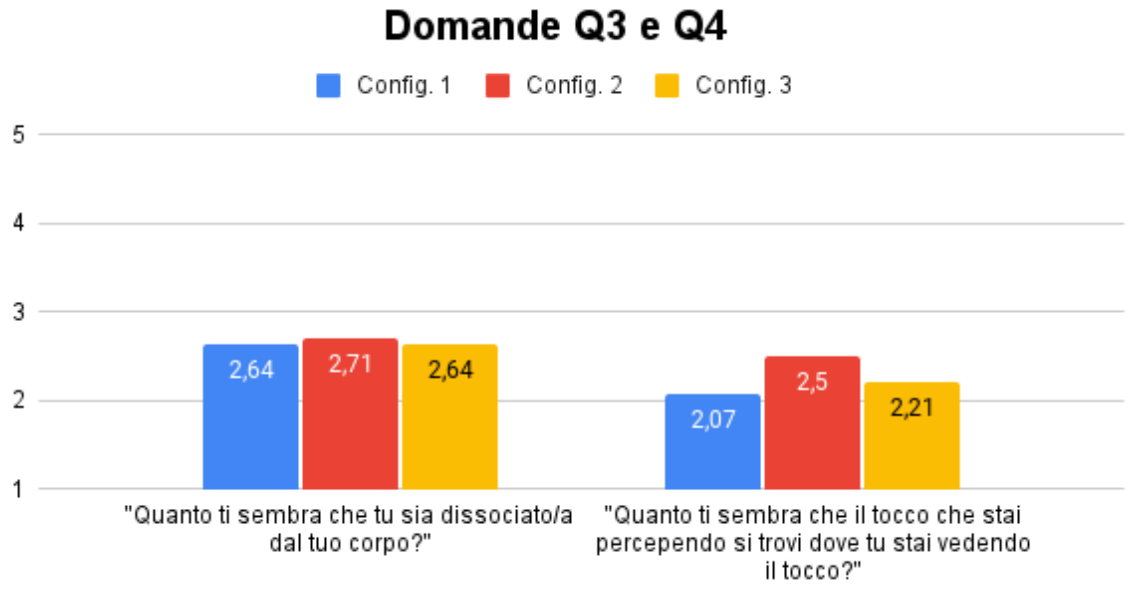

Fig. 92b Le domande Q3 e Q4.

Alla luce di questi risultati, si possono fare un paio di riflessioni:

- La qualità del *livestream* può sicuramente aver influenzato l'esito di questa fase del test.

  Per garantire un *playback* fluido, infatti, è stata settata una risoluzione di 2880 x 1440 *pixel* con bitrate di 3 Mbps: si tratta di settaggi più che accettabili, ma possono aver causato una cattiva percezione del proprio corpo ripreso.
- La distanza dell'utente dalla camera: come visto in Fig. 91, la persona si trova a diversi metri dalla camera e quindi, probabilmente, percepisce meno un "collegamento" con il proprio corpo visualizzato, complice anche la presenza di numerosi oggetti sintetici distraenti.

  Nel test condotto nel *paper* di riferimento [54], l'utente era posizionato davanti alla camera e visionava il flusso video senza modifiche.
- La latenza del video: sempre nel *paper* sopracitato, sono state tentate delle configurazioni asincrone (paragonabili alle condizioni di questo *user test*).

  I risultati evidenziano come il tocco fosse percepito non dove "visto" visivamente ma dove "sentito" tattilmente: i punteggi bassi per la domanda Q4 sembrano allinearsi con questo risultato.

  In condizioni sincrone, impossibili da replicare nel nostro caso, l'immedesimazione nel corpo "rappresentato" aumentava invece di molto.

Questa prima parte del test viene quindi terminata.

Nel complesso, i risultati confermano quanto spiegato nella Sezione 3.4 dedicata alle scelte implementative.
L'approccio consigliato per favorire l'interazione vicina dell'utente è sicuramente quello a camera doppia, che aumenta anche il senso di presenza e la percezione di un *compositing* corretto.
L'approccio a camera singola è funzionale solo se le interazioni dell'utente sono lontane, mentre il livello di presenza e di percezione del *compositing* sono paragonabili alla configurazione a camera doppia.

Ora verrà illustrata la seconda parte del test, dedicata all'implementazione di *OpenPose*.

4.2 Seconda parte

L'intento di questa seconda parte del test è stato valutare quanto le scelte implementative illustrate nella Sezione 3.1.5 fossero efficaci.
In particolare, si è voluto studiare quanta differenza ci sia fra il tracciamento di oggetti statici e dinamici e quanto il modificare i parametri della libreria cambi la velocità di esecuzione.
Anche questa parte ha visto la creazione di due configurazioni differenti che, come anticipato nella Sezione 3.4.5 "Scelta delle impostazioni", consistono nell'utilizzo o meno del *bool Realtime Processing* (Fig. 93).
Elenchiamo ora le differenze e le implicazioni:

- Configurazione *Realtime*:
    - *Bool Realtime Processing*: attivato
    - *Net Resolution*: 800 x 400

    In questo caso, il tracciamento sarà più accurato anche a maggiori distanza grazie alla *net resolution* migliore, ma i *keypoints* verranno aggiornati "a scatti" e saranno in anticipo rispetto al movimento del *performer*.
    Inoltre, gli *FPS* in *Play Mode* rimangono attorno a 70.

- Configurazione *No Realtime*:
    - *Bool Realtime Processing*: disattivato
    - *Net Resolution*: 384 x 192

    Per questa configurazione il tracciamento sarà meno accurato all'aumentare della distanza e l'aggiornamento dei *keypoints* sarà in ritardo rispetto al video, ma più fluido rispetto al caso precedente.

Anche in questo caso il test è diviso in fasi, da ripetere una volta con la Configurazione *Realtime* e una volta con la *No Realtime* (scelte randomicamente per non influenzare l'utente).
Le domande sono state create appositamente per l'occasione, al fine di indagare in modo chiaro sulla percezione dell'utente, e prevedono come prima una risposta da 1 (Per niente) a 5 (Moltissimo).

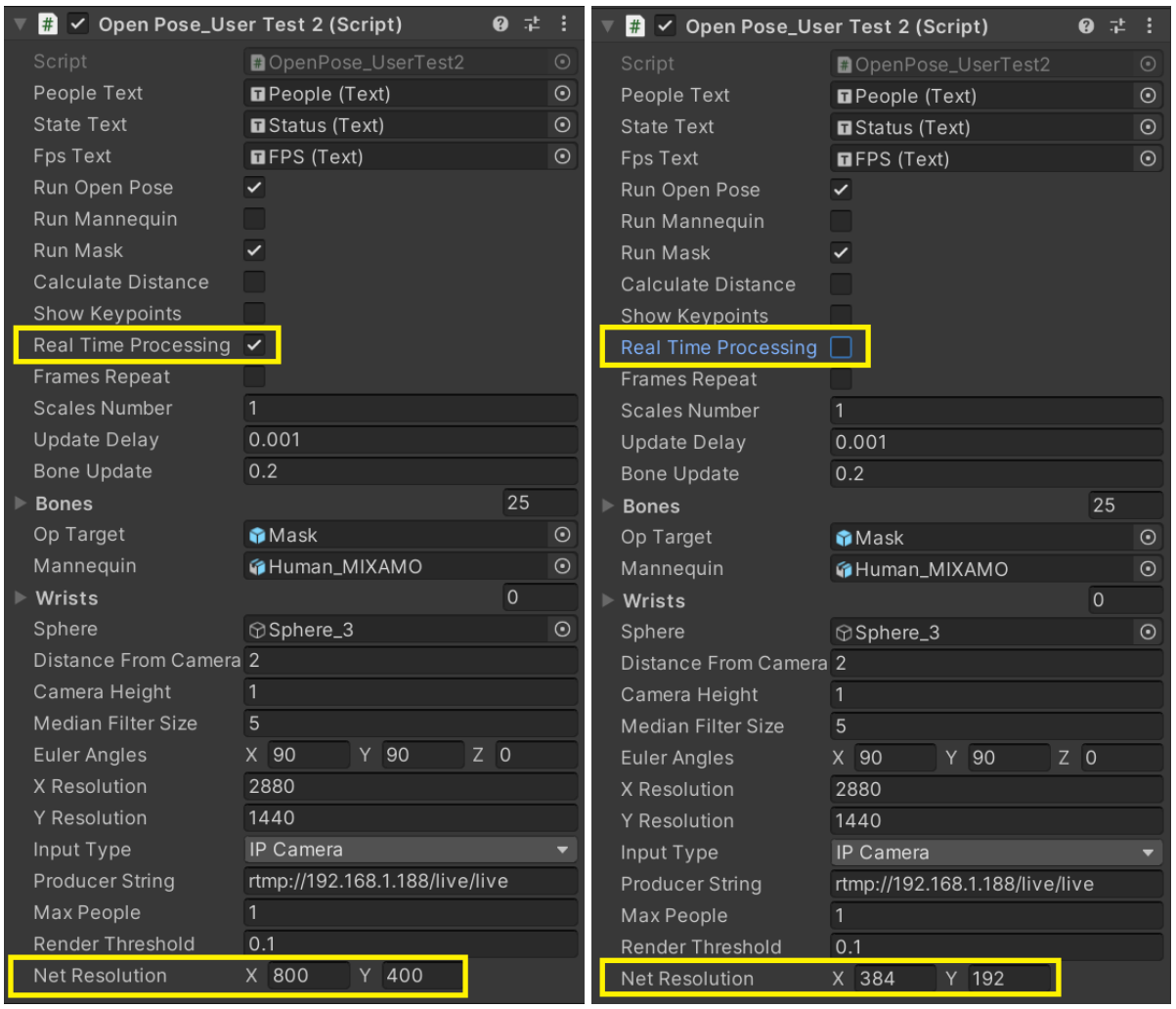

Fig. 93 A sinistra, la configurazione Realtime; a destra, quella No Realtime.

4.2.1 Fase 1 - Valutazione delle condizioni statiche

Viene chiesto al *tester* di recarsi fuori dalla stanza in cui si trova, di sedersi su una sedia e di chiudere gli occhi, al fine di completare i primi settaggi.

Quando gli occhi vengono riaperti l'utente vede, come di consueto, dal punto di vista della camera 360° *live*: vi è un *performer* nella stanza, immobile, con una maschera sovraimpressa sul volto (visibile in Fig. 83).

Viene posta la domanda Q1 "Mi sembra che la maschera sia sovraimpressa correttamente al volto della persona"

L'utente chiude di nuovo gli occhi e viene attivato il tracciamento dei *keypoints*.

Il *performer* sceglie una posa e rimane fermo (Fig. 76).

All’apertura degli occhi viene posta la domanda Q2 "Mi sembra che i giunti siano sovraimpressi correttamente al corpo della persona".
Questa prima fase è considerata la *baseline*, il punto di riferimento, ed è performata una sola volta con la configurazione *Realtime*, in quanto dotata di *net resolution* maggiore.

Risultati

I risultati sono nel complesso soddisfacenti, confermando le teorie espresse nella Sezione 3.4.5 “Oggetti statici”.
Come si può vedere in Fig. 94, i *keypoints* hanno convinto più della maschera: una probabile motivazione è che i punti sono un elemento chiaramente estraneo all’ambiente virtuale, mentre la maschera ha un suo contesto “diegetico” che spinge l’utente a essere più “stretto” nella valutazione.
È probabile che, con una maschera meno astratta e più vicina all’aspetto umano, l’effetto possa migliorare.

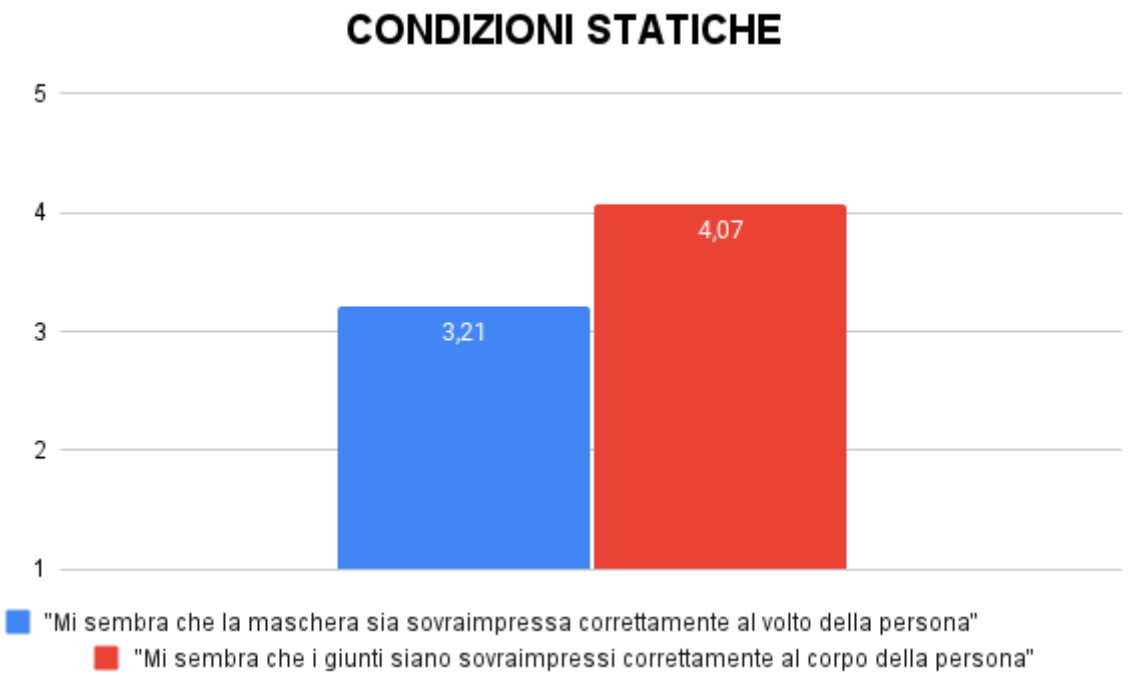

Fig. 94 Domande Q1 e Q2.

4.2.2 Fase 2 - Valutazione delle condizioni dinamiche

In questa fase vengono ripetuti gli step precedenti, ma con il *performer* in movimento.

Viene inoltre aggiunto un terzo elemento da valutare, che differisce dagli altri due per la modalità di implementazione: si tratta di un *avatar* virtuale che segue il movimento del *performer* (spiegato nella Sezione 3.4.5 “Oggetti dinamici”, Fig. 85).

Le domande Q1 e Q2 sono identiche alle precedenti, la Q3 riguarda l’*avatar* ed è formulato come segue “Mi sembra che l’*avatar* segua correttamente il movimento della persona, la quale rappresenta un *target* a cui vuole avvicinarsi”.

Risultati

Nella Fig. 95 vengono illustrati i risultati per le configurazioni *Realtime* e *No Realtime*.

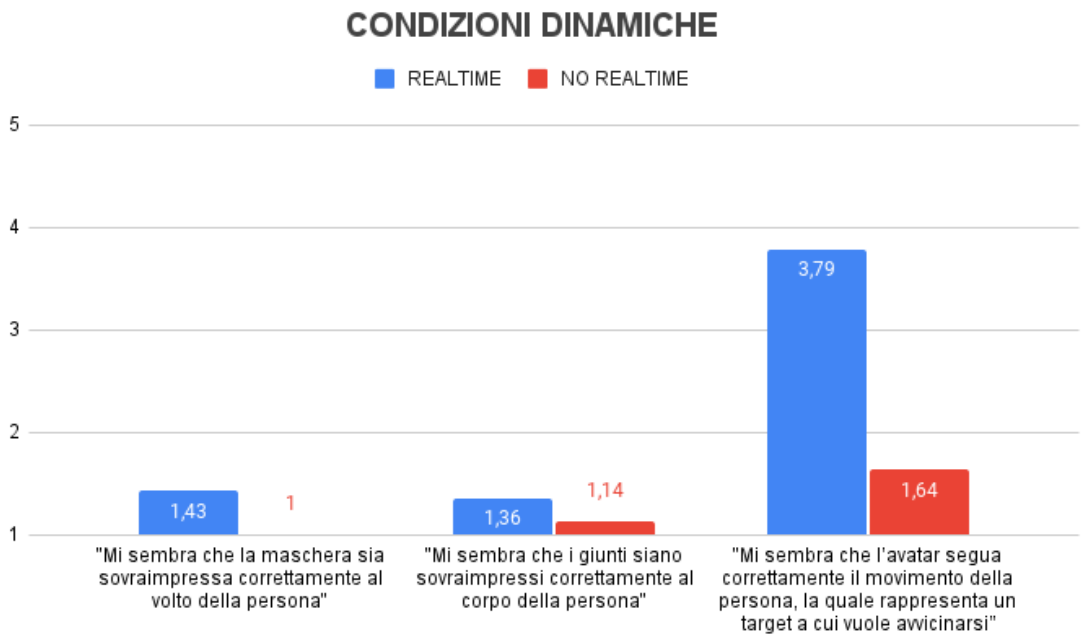

Fig. 95 La media arrotondata delle 3 domande sulle condizioni dinamiche.

Come previsto nella Sezione 3.4.5, il *feedback* da parte degli utenti è generalmente negativo: il ritardo causato dalla configurazione *No Realtime* e la poca fluidità unita all’anticipo di elaborazione della *Realtime* rendono impossibile la sovrapposizione corretta di oggetti su corpi in movimento.

L’unica nota positiva è, come anticipato, il risultato ottenuto dall’*avatar* nella configurazione *Realtime*.
Risulta evidente che, rimuovendo i *keypoints* dal corpo della persona tracciata e rinunciando quindi a una sovraimposizione in tempo reale, risulti fattibile ed efficace puntare sull’approccio “oggetto segue un *target*”: l’avatar infatti ha un suo set di animazioni che lo rendono indipendente dalla precisione dei *keypoints*, i quali servono solo per aggiornare il punto in cui dovrà andare a recarsi.
Resta da notare che, nonostante gli scarsi risultati per le domande Q1 e Q2, la configurazione *Realtime* sia risultata sempre leggermente meglio della sua controparte *No Realtime*.

Si conclude quindi anche la seconda parte dello *user test*, raccogliendo risultati già in parte previsti, confermando le teorie elaborate nella Sezione 3.4.5.

5. Primo scenario implementativo

Come già esplicato nel capitolo di introduzione, questo progetto di tesi nasce per far progredire i mezzi tecnologici poi sfruttati nel progetto *Presence*.
È stato tuttavia molto stimolante, su consiglio della stessa Vanessa Vozzo, iniziare a concepire anche un'applicazione narrativa di tutte le implementazioni finora testate.
Presso lo Studiumlab a Palazzo Nuovo, è stata organizzata una giornata di "scambio di saperi", in cui diversi studenti dell'Accademia Albertina di Belle Arti si sono recati per provare un primo scenario implementativo e fornire dei *feedback*.
L'incontro è stato introdotto da Vanessa Vozzo, che ha spiegato le basi del progetto *Presence* e illustrato le esperienze che hanno visto la luce in questi anni (Fig. 96).
Gli studenti hanno poi potuto provare l'esperienza, che è scritta e implementata per l'occasione.
Andremo ora a illustrare la sua struttura e il suo svolgimento.

Fig. 96 Vanessa Vozzo introduce il progetto Presence presso lo Studiumlab.

5.1 La concezione

L'idea di fondo è stata quella di indagare sulle proprietà del corpo umano e cosa significhi davvero "essere in controllo" di un corpo.
Al fine di ottenere questo, sono state utilizzate tutte le funzionalità testate durante la fase di ricerca, unendole in uno scenario che fosse tenuto insieme da un filo logico-narrativo.
È stato quindi creato dapprima uno *storyboard 360°*, che aiutasse a visualizzare le varie fasi dell'esperienza, denominate "Azioni": si tratta di un *tool* estremamente importante per definire con chiarezza quali elementi in scena sono sintetici e quali *live action*, ma anche esplicitare se l'ambiente rappresentato è un *livestream* 360° o puramente virtuale.
Nella Fig. 97 è possibile vedere l'impostazione di una pagina dello *storyboard 360°*, che si è dimostrato molto utile per comunicare con tutte le "maestranze" coinvolte nell'esperienza.
Seguendo la struttura del *modello misto* suggerito da Vanessa Vozzo, in alto a sinistra è presente un ovale, che rappresenta una visuale dall'alto dell'ambiente, dove al centro è situato il punto di vista dell'utente.
Con vari colori sono indicati i differenti elementi che fanno parte della scena, poi specificati nella legenda, al fine di rendere comprensibile l'impostazione "spaziale".
Proprio nella legenda è stato infatti scelto di indicare chiaramente quali elementi siano *VR*, ovvero completamente sintetici, e quali facciano parte del *360 LS (LiveStream)*, quindi ripresi in tempo reale dalla camera.
In basso a destra si trova il "cilindro srotolato", che viene utilizzato per visualizzare in anteprima il punto di vista dell'utente: nella parte centrale vi è il punto di interesse della scena e allontanandosi da questo si raggiunge, ai bordi, la parte posteriore dell'ambiente.
È inoltre indispensabile fornire una descrizione dell'Azione che l'utente dovrà svolgere oppure osservare: si tratta di un modo di illustrare la "trama" dell'esperienza, se presente.
Allo stesso modo, è importante anche definire la *user experience*, ovvero ciò che si vuol far provare sensorialmente all'utente durante l'Azione e le reazioni attese da parte dello stesso.

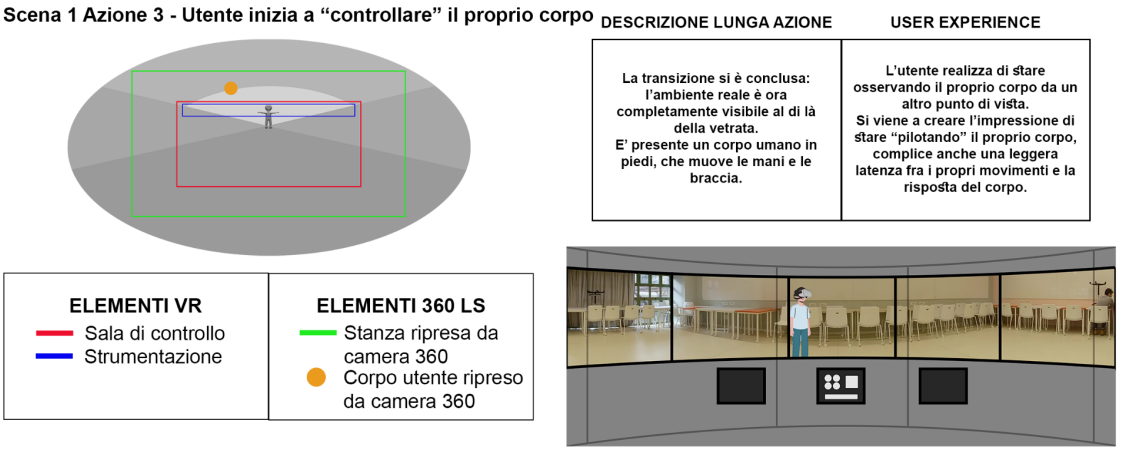

Fig. 97 Una pagina dello storyboard 360.

In modo molto riassuntivo, andiamo a elencare le Azioni che fanno progredire l’esperienza dell’utente:

1. Inizio: l’utente inizia l’esperienza trovandosi in una sala di laboratorio *VR*, con una vetrata che separa l’ambiente da una stanza più grande, vuota (Fig. 79).
   Un monitor rivolge la domanda "Dov'è il tuo corpo?” e invita a interagire con una leva.
2. Transizione: da ambiente *VR*, le pareti della stanza esterna iniziano a smaterializzarsi rivelando l’ambiente 360° dove si trova l’utente.
   Nel frattempo compare una sfera di fumo nel punto dove si trova l’utente stesso (Fig. 82).
3. Controllo del corpo: ora è visibile, oltre la vetrata, unicamente l’ambiente 360°. L’utente muove le mani virtuali e si rende conto che il proprio corpo ripreso “reagisce in ritardo” (causa latenza del *livestream*) e l’impressione è quella di star “pilotando” il proprio corpo reale.
4. Comparsa manichino: alla pressione di un tasto, appare dal pavimento un manichino sintetico, rivolto verso l’utente e controllato dalle mani dello stesso.
   Il monitor chiede “Qual è il tuo corpo?”, accentuando il senso di straniamento.
   Una *performer* entra nell’ambiente reale e tocca il corpo dell’utente, sussurrandogli la stessa frase del monitor (Fig. 98).
5. Contatto: l’utente viene invitato ad “entrare in contatto con il vero sé" attraverso un testo sul monitor.

Allungando le proprie mani di fronte a sé, dal manichino iniziano a uscire dei raggi di energia che lentamente raggiungono il corpo 360° dell'utente.

6. Allarme: il manichino smette di essere tracciato, diventa "senziente" e inizia a camminare per la stanza, avvicinandosi all'utente oltre la vetrata (Fig. 99).
   Il monitor segnala "ALLARME, HAI PERSO IL CONTROLLO!", mentre le luci d'ambiente diventano rosse lampeggianti.
   Un bottone di emergenza diventa premibile.
7. Scelta del finale: alla pressione del bottone di emergenza, l'utente ha l'occasione di fermare il manichino scegliendo fra 3 finali differenti.

Fig. 98 Una performer tocca il corpo dell'utente durante l'Azione 4.

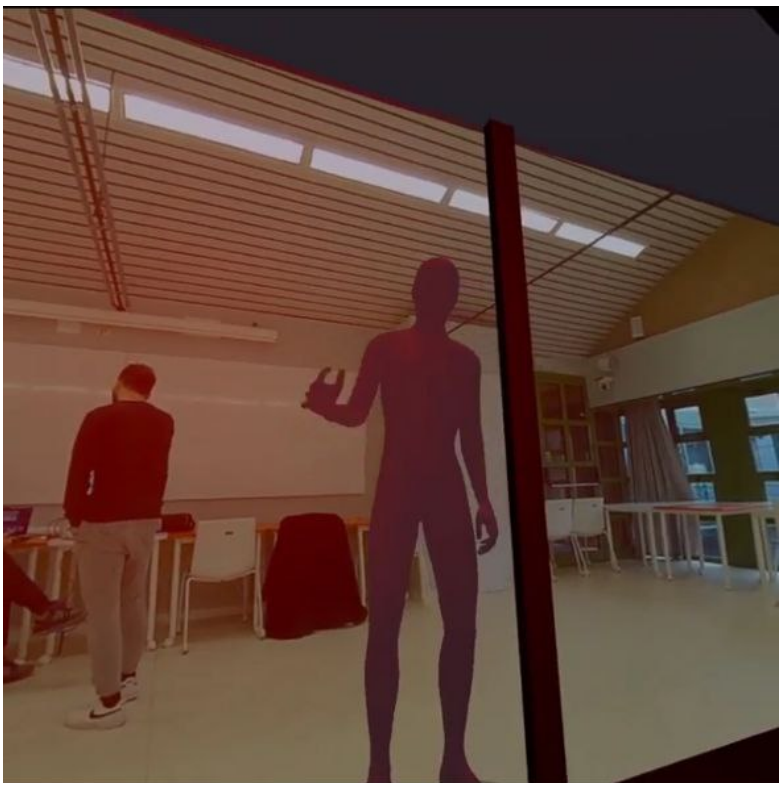

Fig. 99 Nell'Azione 6 il manichino "diventa senziente" e minaccia l'utente.

5.2 La formalizzazione logica

Definito il progredire dell'esperienza dal punto di vista "narrativo", è stato necessario formalizzare la logica che permetterà la creazione dell'applicativo sul motore di gioco *Unity*.

È stato quindi creato un *flow chart* che andasse a definire i vari elementi che entrano in gioco nelle varie Azioni e in che modo questi interagiscono fra di loro.

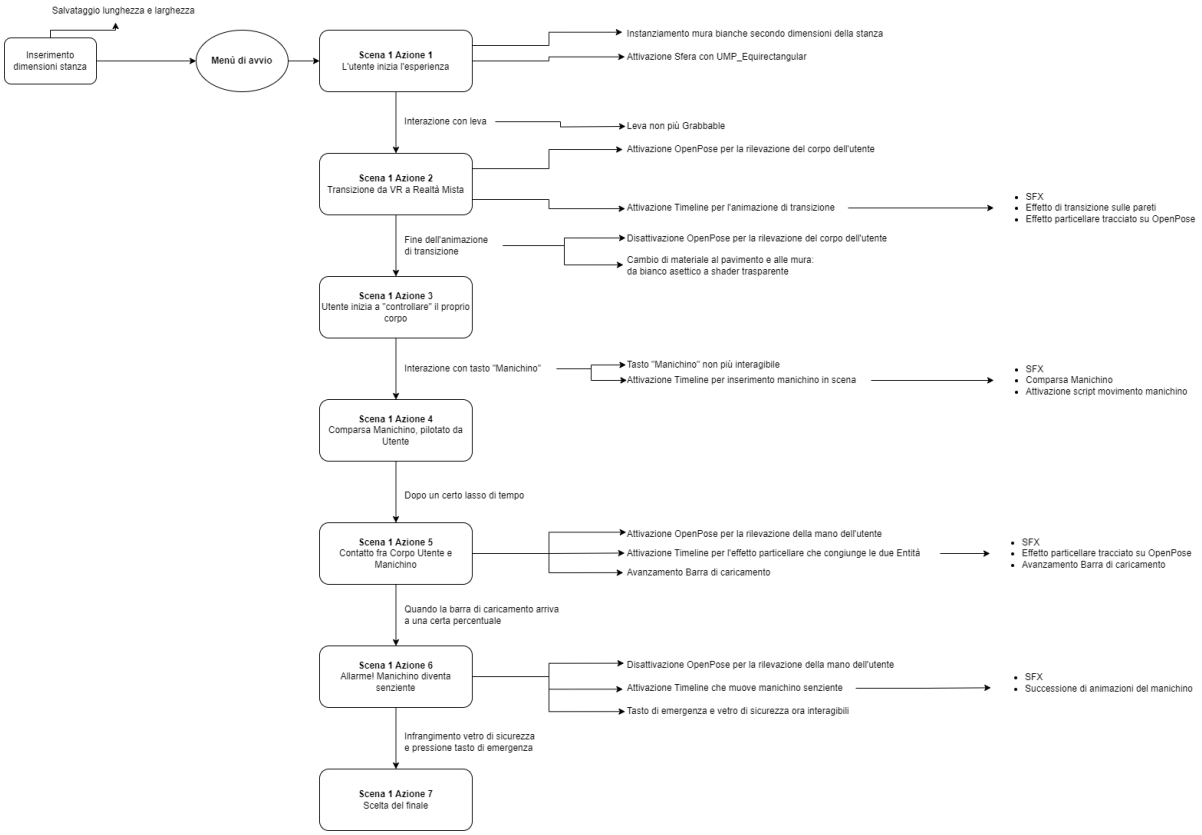

Fig. 100 Il flow chart dello scenario implementativo.

Il *flow chart* è un elemento molto utile poiché, come lo *storyboard*, rappresenta uno strumento per poter comunicare efficacemente le proprie idee alle persone coinvolte, senza dover entrare in tecnicismi magari poco comprensibili.

Una volta formalizzata la logica dietro l'esperienza, si è passati all'implementazione su *Unity.*

5.3 L’implementazione

Attraverso un *script* chiamato *FlowManager* è stato gestito il susseguirsi della Azioni componenti la “trama”: si tratta di un componente implementativo fondamentale, soprattutto se la “storia” narrata procede in modo lineare, tramite step definiti.
Tutte le funzioni illustrate nella Sezione 3.4 sono state implementate, costruendo una sorta di narrativa attorno a pregi e difetti di ognuna.
In particolare:

- È stata implementata la doppia camera, per visualizzare l’intera stanza di laboratorio in stereoscopia e l’ambiente oltre la vetrata in monoscopia.
- È stata utilizzata l’illuminazione con luce dall’alto, compromesso necessario viste le limitazioni di *Unity* e dello *shader.*
- È stato utilizzato l’*embodiment* in terza persona attraverso il manichino.
- È stata utilizzata la libreria di *OpenPose* per tracciare il corpo statico dell’utente durante la fase di transizione iniziale.
- È stata sfruttata la latenza del *livestream* per indurre una dissociazione corporea nell’utente, sia quando egli si rende conto che il proprio corpo “reale” reagisce in ritardo rispetto al manichino, sia quando la *performer* entra in scena a toccare il suo corpo.

L’esperienza dura circa 3 minuti, ma può essere riproposta per testare i 3 differenti finali.

5.4 Feedback

Al termine dell’esperienza è stato sottoposto un questionario ideato per l’occasione, che prevedeva delle domande su alcuni aspetti specifici ma anche dello spazio per commenti aperti.
Nel complesso il *feedback* è stato positivo: sono stati apprezzati l’idea e la realizzazione, così come la sensazione di straniamento dovuta alla presenza della *performer* e al ritardo del flusso video.
Sono stati fatti anche numerosi commenti sulla realizzazione tecnica dello scenario implementativo, in quanto è stata notata una eccessiva semplicità dei modelli 3D e della caratterizzazione dell’ambiente: si tratta di opinioni più che lecite, dal momento

che è stata data, in fase di implementazione, maggiore importanza al “flusso” della storia che alla qualità grafica.
Le interazioni per far proseguire la “storia” sono state considerate chiare e ben spiegate, ma è stata criticata la scelta utilizzare il movimento di *pinch* per attivare una leva: anche in questo caso il commento è lecito, tuttavia la libreria che implementa l’*hand tracking* su *Unity* attualmente supporta solo i movimenti di *pinch* e *poke*.
È stato mostrato grande interesse per ulteriori sviluppi del progetto, auspicando una durata aumentata e una maggiore “completezza” generale.

Nel prossimo capitolo verranno tirate le somme sul progetto di tesi e tracciati alcuni percorsi per sviluppi futuri.

6. Conclusione e sviluppi futuri

Durante questi mesi di ricerca e implementazione sono state sviluppate, come illustrato, delle funzionalità valide e innovative.
L'implementazione di una doppia camera, ad esempio, si è dimostrata efficace nel facilitare l'interazione da parte dell'utente, mantenendo un effetto di *compositing* verosimile: non è più obbligatorio, come suggerito in letteratura [31], rendere l'intero ambiente virtuale compositato monoscopico.
La *feature* di pilotare un *avatar*, sia in prima che in terza persona, spalanca le porte a nuove ricerche sulle scienze cognitive e sul grado di auto-coscienza nella virtualità.
L'utilizzo della libreria *OpenPose* su *Unity* è stato più che un esperimento: per la prima volta in letteratura è stato utilizzato un video 360° *livestream* per rilevare la presenza di corpi o il loro movimento, sfruttandoli poi in modo interattivo nel motore di gioco.
Lo scenario implementativo ha unito le varie *features* in quella che può essere definita una prima *demo* delle nuove potenzialità del progetto *Presence*.

Il progetto di ricerca può ovviamente proseguire ed essere ampliato, andando ad esplorare alcuni aspetti più in profondità, sperimentando con:

- Tecnologie di ripresa 360° differenti.
- Una libreria alternativa per il *playback 360°* che possa permettere una minore latenza del flusso video *live*.
- Una *rendering pipeline* differente da quella standard, che possa migliorare la resa grafica.
- Differenti *features* che la libreria *OpenPose* offre, come il riconoscimento della posa delle mani e delle espressioni facciali o il tracciamento multi-persona.

Si tratta di aspetti che vale la pena approfondire, al fine di migliorare ulteriormente il prodotto finale.
Il progetto *Presence* fa quindi un grande passo avanti nel percorso di innovazione tecnologica che lo contraddistingue: l'autore non vede l'ora di vedere quali esperienze potranno nascere grazie all'utilizzo delle *features* studiate in questo progetto di tesi.

Bibliografia e Sitografia

[1] http://www.officinesintetiche.it/

[2] Beanotherlab. (n.d). *BeAnotherLab – Empathy & VR*. http://beanotherlab.org/

[3] Van Der Hoort, B., Guterstam, A., & Ehrsson, H. H. (2011). Being Barbie: The Size of One's Own Body Determines the Perceived Size of the World. *PLOS ONE*, *6*(5), e20195. https://doi.org/10.1371/journal.pone.0020195

[4] Nishida, J., Matsuda, S., Oki, M., Takatori, H., Sato, K., & Suzuki, K. (2019). *Egocentric Smaller-person Experience through a Change in Visual Perspective*. https://doi.org/10.1145/3290605.3300926

[5] Slater, M. (2009). Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philos Trans R Soc Lond B Biol Sci*, *364*(1535), 3549–3557. https://doi.org/10.1098/rstb.2009.0138

[6] *VR in Wonderland#1*. (2019). https://kunstuni-linz.at/VR-in-Wonderland-1.16603.0.html

[7] *Circolo Del Design — Missing Out*. (2020). Circolo Del Design — Missing Out. https://www.circolodeldesign.it/whats-on/programma/missing-out

[8] Przybylski, A. K., Murayama, K., DeHaan, C. R., & Gladwell, V. (2013). Motivational, emotional, and behavioral correlates of fear of missing out. *Computers in Human Behavior*, *29*(4), 1841–1848. https://doi.org/10.1016/j.chb.2013.02.014

[9] Balletto Teatro di Torino. (2023, June 14). *TINY UPPERCASE - BTT Balletto Teatro di Torino*. BTT Balletto Teatro Di Torino. https://www.ballettoteatroditorino.it/tiny-uppercase-2/

[10] Okun, J. A., Zwerman, S., & Society, V. E. (2010). *The VES Handbook of Visual Effects: Industry Standard VFX Practices and Procedures*. Taylor & Francis.

[11] Library of Congress. (2017, December 11). *The Great Train Robbery* [Video]. YouTube. https://www.youtube.com/watch?v=In3mRDX0uqk

[12] https://www.youtube.com/@PetrPechar1975/videos

[13] https://www.youtube.com/@whospuss

[14] https://www.youtube.com/@TomScottGop

[15] ScreenSkills. (2022). VFX career map. *ScreenSkills*. https://www.screenskills.com/starting-your-career/career-maps/vfx-career-map/

[16] *AuraFx*. (n.d.). https://www.aurafxstudios.com/

[17] https://www.youtube.com/@beforesmag

[18] Amandafullwoodma. (n.d.). *set extension – Amanda Fullwood's Film and Visual Art Journal*. Amanda Fullwood's Film and Visual Art Journal. https://amandafullwoodma.wordpress.com/tag/set-extension/

[19] *On-Set Facilities | Virtual Production Talent & Service | North Wales*. (n.d.). On-Set Facilities. https://www.onsetfacilities.com/

[20] Barbera, D. (2023, June 5). Il visore Apple Vision Pro e le altre novità del Wwdc 2023. *Wired Italia*. https://www.wired.it/article/visore-apple-vision-pro-prezzo-uscita/

[21] *Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine*. (n.d.). https://unity.com/

[22] Overview of ARCore and supported development environments. (n.d.). *Google for Developers*. https://developers.google.com/ar/develop?hl=en

[23] Bonnington, C. (2015, March 13). You Can Now Watch and Upload 360-Degree Videos on YouTube. *WIRED*. https://www.wired.com/2015/03/youtube-360-degree-video/

[24] Company, F., & Meta. (2019, November 7). Introducing Facebook 360 For Gear VR. *Meta*. https://about.fb.com/news/2017/03/introducing-facebook-360-for-gear-vr/

[25] HDblog.It. (2017). Vimeo aggiunge il supporto ai video a 360 gradi. *HDblog.it*. https://www.hdblog.it/2017/03/08/Vimeo-supporto-video-360-gradi/

[26] Nielsen, F. (2005). Surround video: a multihead camera approach. *The Visual Computer*, *21*(1–2), 92–103. https://doi.org/10.1007/s00371-004-0273-z

[27] AnotheReality. (2020, November 13). *Differenze tra VR e Video 360: facciamo chiarezza - AnotheReality*. AnotheReality. https://www.anothereality.io/differenze-tra-vr-e-video-a-360-gradi-facciamo-chiarezza/?lang=it

[28] GoPro Tips. (2016, August 10). *GoPro: Introducing Omni* [Video]. YouTube. https://www.youtube.com/watch?v=Xo6LZNkNLTg

[29] Snyder, J. P. (1997). *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press.

[30] Brown, C. (2017, March 14). Bringing pixels front and center in VR video. *Google*. https://blog.google/products/google-ar-vr/bringing-pixels-front-and-center-vr-video/

[31] Rhee, T., Petikam, L., Allen, B. L., & Chalmers, A. D. (2017). MR360: Mixed Reality Rendering for 360° Panoramic Videos. *IEEE Transactions on Visualization*

and Computer Graphics, *23*(4), 1379–1388. https://doi.org/10.1109/tvcg.2017.2657178

[32] Debevec, P. (2002). Image-based lighting. *IEEE Computer Graphics and Applications*, *22*(2), 26–34. https://doi.org/10.1109/38.988744

[33] Chalmers, A. D., Choi, J. J., & Rhee, T. (2014). Perceptually Optimised Illumination for Seamless Composites. *PG (Short Papers)*. https://doi.org/10.2312/pgs.20141268

[34] Debevec, P. (2008). *Rendering synthetic objects into real scenes*. https://doi.org/10.1145/1401132.1401175

[35] Tarko, J., Tompkin, J., & Richardt, C. (2019). *Real-time Virtual Object Insertion for Moving 360° Videos*. https://doi.org/10.1145/3359997.3365708

[36] Park, J. (2021). *Real-time object detection in 360-degree videos*. https://doi.org/10.1117/12.2586403

[37] Xie, L., Zhang, X., & Guo, Z. (2018). *CLS*. https://doi.org/10.1145/3240508.3240556

[38] Redmon, J. (n.d.). *YOLO: Real-Time Object Detection*. https://pjreddie.com/darknet/yolo/

[39] Mazzola, G., Lo Presti, L., Ardizzone, E., & La Cascia, M. (2021). A Dataset of Annotated Omnidirectional Videos for Distancing Applications. *Journal of Imaging*, *7*(8), 158. https://doi.org/10.3390/jimaging7080158

[40] Laga, H., Jospin, L. V., Boussaid, F., & Bennamoun, M. (2022). A Survey on Deep Learning Techniques for Stereo-Based Depth Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(4), 1738–1764. https://doi.org/10.1109/tpami.2020.3032602

[41] Bhoi, A. (2019). Monocular depth estimation: A survey. *arXiv preprint arXiv:1901.09402*.

[42] Technologies, U. (n.d.). *360 video*. Unity. https://unity.com/solutions/360video

[43] Contributo dell'utente "monkeyscience". https://forum.unity.com/threads/matte-shadow.14438/

[44] Contributo dell'utente "AnKOu" https://forum.unity.com/threads/disabling-stereo-rendering-for-a-camera.508549/

[45] *Animation Rigging | Animation Rigging | 1.1.1*. (n.d.). https://docs.unity3d.com/Packages/com.unity.animation.rigging@1.1/manual/index.html

[46] RealaryVR. (2022, December 22). *VR Full Body Avatar | Unity Tutorial for Oculus Quest* [Video]. YouTube. https://www.youtube.com/watch?v=Iya-gKJ0V74

[47] Wikipedia contributors. (2023). Inverse kinematics. *Wikipedia*. https://en.wikipedia.org/wiki/Inverse_kinematics

[48] Contributo dell'utente "alexchesser" https://forum.unity.com/threads/oculus-handtracking-with-custom-hand-model.810678/

[49] Boesch, G. (2023). The Complete Guide to OpenPose in 2023. *viso.ai*. https://viso.ai/deep-learning/openpose/#:~:text=OpenPose%20is%20a%20real%2Dtime,facial%20keypoints%20on%20single%20images.

[50] Cmu-Perceptual-Computing-Lab. (n.d.). *GitHub - CMU-Perceptual-Computing-Lab/openpose_unity_plugin: OpenPose's Unity Plugin for Unity users*. GitHub. https://github.com/CMU-Perceptual-Computing-Lab/openpose_unity_plugin

[51] *Che cos'è una rete neurale convoluzionale?* | *3 cose da sapere*. (n.d.). MATLAB & Simulink. https://it.mathworks.com/discovery/convolutional-neural-network-matlab.html

[52] Uptale. (2021). 5 mistakes not to make when shooting in 360°. *Uptale*. https://www.uptale.io/en/for-creators/5-mistakes-not-to-make-when-shooting-360-videos/#:~:text=Indeed%2C%20the%20fisheye%20lenses%20used,one%20meter%20of%20the%20camera.

[53] Kalawsky, R. S. (1999). VRUSE—a computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems. *Applied Ergonomics*, *30*(1), 11–25. https://doi.org/10.1016/s0003-6870(98)00047-7

[54] Lenggenhager, B., Mouthon, M., & Blanke, O. (2009). Spatial aspects of bodily self-consciousness. *Consciousness and Cognition*, *18*(1), 110–117. https://doi.org/10.1016/j.concog.2008.11.003

Elenco delle figure

Fig. 1 La prospettiva del partecipante: il bastoncino che tocca il piede sinistro è mosso da chi conduce l'esperimento. - Pagina 8
Fig. 2 I quattro corpi artificiali utilizzati per l'esperimento Barbie Doll, rispettivamente, da sinistra a destra, di 400, 180, 80 e 30 centimetri. - Pagina 9
Fig. 3 A sinistra, stima della dimensione dell'oggetto; a destra, stima della distanza. - Pagina 9
Fig. 4 Da sinistra a destra: l'errore nel task della stretta di mano, la differente percezione dello spazio personale, l'istinto ad assumere una posa protettiva. - Pagina 10
Fig. 5 Il setup di VR in Wonderland: la postazione per l'utente, il dispositivo telecomandato con la camera 360° e il "labirinto" da esplorare. - Pagina 13
Fig. 6 I focus group da cui nascono le idee per i progetti artistici di Presence. - Pagina 14
Fig. 7 L'Area 1, con il monitor su cui osservare il cortometraggio. - Pagina 15
Fig. 8 L'Area 2, con il visore per video 360°. - Pagina 16
Fig. 9 L'Area 3, la stanza arredata con visore e camera 360°. - Pagina 17
Fig. 10 Proiezione equirettangolare del punto di vista degli spettatori durante la Fase 1. - Pagina 18
Fig. 11 Il modello 3D della grotta che si vede durante la Fase 2. - Pagina 19
Fig. 12 Proiezione equirettangolare del punto di vista degli spettatori durante la Fase 3. - Pagina 19
Fig. 13 La Fase 4 di Tiny Uppercase. - Pagina 20
Fig. 14 Una delle immagini visionate nella Fase 1. - Pagina 21
Fig. 15 L'interfaccia del motore di gioco che genera l'ambiente immersivo, in cui si può vedere il reticolato della Fase 2. - Pagina 22
Fig. 16 Il set-up della Fase 3: i due spettatori esplorano l'ambiente virtuale indossando l'HMD. Sono già presenti le camere 360° per la fase successiva. - Pagina 23
Fig. 17 La Fase 4: l'incontro fra i due spettatori. - Pagina 23
Fig. 18 L'ambiente interno è stato girato in una prima fase, il treno all'esterno in una seconda. - Pagina 26

Fig. 19 L’utilizzo del glass shot in Tempi Moderni di Charlie Chaplin. - Immagine presa dal canale YT “Petr Pechar” [12] - Pagina 26
Fig. 20 Background projection sul set di Eyes Wide Shut: a sinistra il footage finale, a destra l’attore su un tapis roulant cammina davanti allo sfondo proiettato. - Immagini prese dal canale YT “whospuss” [13] - Pagina 27
Fig. 21 Il matte che si ottiene durante il procedimento. - Immagine presa dal canale YT “Tom Scott” [14] - Pagina 28
Fig. 22 Il reparto, all’interno di uno studio di VFX, che si occupa della computer grafica. - Immagine presa dal sito “Screen Skills” [15] - Pagina 30
Fig. 23 La ricostruzione ottenuta con tecnologia lidar, ad opera del team Aura FX [16], di una metropolitana, utilizzata poi nel film Joker del 2019. - Immagine presa dal canale YT “before & afters” [17] - Pagina 31
Fig. 24 La tecnica del set-extension utilizzata per “Game of Thrones”. - Immagine presa dal sito “Amanda Fullwood” [18] - Pagina 32
Fig. 25 Il real-time camera tracking: l’operatore può già visualizzare nel monitor il modello 3D animato e contestualizzato nello spazio live action. - Immagine presa dal canale YT dell’agenzia “On-Set Facilities” [19] - Pagina 34
Fig. 26 La realtà aumentata in Pokemon Go, gioco mobile pubblicato da Niantic nel 2016. - Pagina 35
Fig. 27 A sinistra, un modello con un’ombra che non corrisponde a quella della scena reale; a destra l’ombra è correttamente posizionata. - Immagine presa dal sito di ARCore [22] - Pagina 37
Fig. 28 A destra, il contributo delle ambient spherical harmonics. - Immagine presa dal sito di ARCore. - Pagina 37
Fig. 29 Da sinistra a destra, la Insta360 One X2, la GoPro Max e la Insta360 Pro. - Pagina 40
Fig. 30 Omni prodotto da GoPro. - Pagina 40
Fig. 31 La proiezione equirettangolare con indicate le evidenti distorsioni ai poli. - Immagine presa da blog.google [30] - Pagina 42
Fig. 32 Un frame estratto da un video 360° con proiezione equirettangolare: da notare l’assenza di distorsioni al centro dell’inquadratura.- Pagina 42
Fig. 33 La proiezione cubemap illustrata. - Immagine presa da blog.google - Pagina 43

Fig. 34 - La convergenza. - Immagine di Jenny Lipton - Pagina 44
Fig. 35 Un frame di un video 360 stereoscopico: notare come le prospettive siano leggermente sfalsate per la parallasse. - Pagina 45
Fig. 36 Le radiance maps a differenti risoluzioni. - Immagine presa dal paper in questione [31] - Pagina 47
Fig.37 Il procedimento di Light Detection. Dall'alto al basso: il frame di input, il mascherino di pixel che superano la soglia, il render usando solo IBL, il render usando anche la IBS. - Immagine presa dal paper in questione - Pagina 48
Fig. 38 dall'alto al basso, da sinistra a destra: il video 360° di sfondo, lo sfondo + la local scene + gli oggetti sintetici, la local scene, il mascherino generato dagli oggetti sintetici, il risultato della formula b * (1-d) - c, il risultato finale. - Pagina 49
Fig. 39 Il feature tracking 360° implementato in pre-process. - Immagine presa dal paper in questione [35] - Pagina 50
Fig. 40 A sinistra, un light probe con LDR; a destra, una versione HDR, che si nota essere meglio esposta. - Immagine presa dal paper in questione - Pagina 51
Fig. 41 Le scene di dimostrazione: il risultato è efficace, ma è pre-computato e la sorgente di luce rappresentante il sole è posizionata manualmente. - Immagine presa dal paper in questione - Pagina 52
Fig. 42 La porzione centrale non presenta distorsioni, a differenza delle due aree in rosso. - Immagine presa dal paper in questione [36] - Pagina 54
Fig. 43 Un esempio ottenuto con un algoritmo di object detection basato su YOLO [38]: una persona, indicata in blu, non è stata rilevata. - Immagine presa dal paper in questione - Pagina 54
Fig. 44 A sinistra, i differenti angoli di proiezione; a destra, il frame composto dai vari contributi. - Immagine presa dal paper in questione - Pagina 55
Fig. 45 Se viene a mancare l'ipotesi di "orizzontalità", punti alla stessa distanza nel mondo 3D verranno shiftati notevolmente in equirettangolare. - Immagine presa dal paper in questione [39] - Pagina 57
Fig. 46 Le variabili in gioco nel calcolo della distanza: d è semplicemente un cateto di un triangolo rettangolo. - Immagine presa dal paper in questione - Pagina 58
Fig. 47 Il Video Player e la Render Texture, con dimensioni settate in base alla risoluzione nota. - Pagina 60

Fig. 48 Il materiale Skybox/Panoramic e l’esito finale, in cui il video 360° viene utilizzato come skybox. - Pagina 60
Fig. 49 Lo script UniversalMediaPlayer e i settings della Insta360 Pro, dove viene fornito l’URL da incollare su Unity. - Pagina 61
Fig. 50 Il materiale che fa uso dello shader UMP_Equirectangular e il risultato finale, mappato all’interno di una sfera. - Pagina 61
Fig. 51 Sopra, il cubo sintetico; sotto, la sua transform (posizione, rotazione, scala). - Pagina 62
Fig. 52 Sopra, evidenziato in arancione, l’oggetto Floor; sotto, la sua gerarchia e la sua transform (Y = -1 in quanto la camera è situata a 1 metro da terra). - Pagina 63
Fig. 53 A sinistra, l’oggetto Floor con lo shader applicato; a destra lo stesso oggetto con un materiale di default. - Pagina 64
Fig. 54 A sinistra, l’oggetto Floor con lo shader applicato; a destra, senza shader. - Pagina 65
Fig. 55 Il componente Tracked Pose Driver, a cui nella proprietà Tracking Type è stato inserito Rotation Only. - Pagina 67
Fig. 56 La singola camera utilizzata con questo approccio: la posizione rimane fissa nell’origine e varia solo la rotazione. Notare inoltre come la scale sia 0 per garantire la monoscopia. - Pagina 68
Fig. 57 La camera mono rimane sempre nell’origine del mondo, mentre quella stereo può muoversi se l’utente muove la testa. - Pagina 69
Fig. 58 Dall’alto al basso: il contributo del layer Background, il contributo del layer Objects e l’unione di essi visto dall’utente. Ovviamente in queste immagini non è possibile apprezzare la differenza percettiva dei due contributi. - Pagina 70
Fig. 59a Il gameobject della camera chiamata “STEREO”. - Pagina 71
Fig. 59b Il gameobject della camera chiamata “MONO”. - Pagina 72
Fig. 60 A sinistra l’approccio a camera singola (con il modello della mano sullo stesso layer Background del Floor); a destra l’approccio a camera doppia (il modello della mano si trova sul layer Objects mentre il Floor su Background): da notare l’assenza dell’ombra sul pavimento. - Pagina 73
Fig. 61 Dall’alto al basso: un’immagine outdoor equirettangolare 2880x1440, il risultato del filtraggio dei pixel, l’immagine 45x22 frutto del sottocampionamento. - Pagina 75

Fig. 62 Le quattro light sources create dai quattro pixel bianchi presenti in basso in Fig. 61. - Pagina 76
Fig. 63 Scenario outdoor con una sola sorgente di luce attivata: le ombre proiettate dal cubo e dalla capsula sono coerenti con quella del barile di legno sullo sfondo. - Pagina 77
Fig. 64 Da sinistra a destra: Head Constraint e Right Arm IK. - Pagina 78
Fig. 65 A sinistra il target utilizzato in Right Arm IK, a destra l'hint: per il braccio sinistro si procede nello stesso modo. - Pagina 78
Fig. 66 Head Body Rig: i VR Target sono le componenti VR di testa e mani, i Rig Target sono i constraints creati in fase di setup. - Pagina 79
Fig. 67 L'approccio first-person. - Pagina 80
Fig. 68 L'approccio third person, con l'avatar sia rivolto verso l'utente che ruotato di 90 gradi. - Pagina 81
Fig. 69 In alto, la gerarchia dell'avatar con evidenziato il rig; in basso un semplice script applicato al rig per muoverlo e ruotarlo dove desiderato. - Pagina 82
Fig. 70 I 25 keypoints del corpo riconosciuti con il modello chiamato BODY_25. - Pagina 83
Fig. 71 OpenPose interpreta il livestream come se provenisse da un IP Camera. - Pagina 83
Fig. 72 All'interno del metodo Update(), l'utilizzo di datum per ottenere il numero di persone nell'immagine e calcolare gli FPS di elaborazione. - Pagina 84
Fig. 73 Uno scenario standard: la persona è distante qualche metro dalla camera, è a contatto con il terreno e quindi non presenta distorsioni. - Pagina 85
Fig. 74 Il formato Datum: per ogni persona in scena si possono ottenere le coordinate x e y di una parte del corpo sfruttando i dati contenuti nell'array poseKeypoints. - Pagina 85
Fig. 75 I keypoints si trovano sulla superficie della sfera che renderizza il video 360°. - Pagina 86
Fig. 76 Vista la bidimensionalità dell'immagine 360°, l'assenza di profondità dei keypoints non è rilevante. - Pagina 86
Fig. 77 I due shader che renderizzano occhio destro e sinistro su due oggetti sfera differenti. - Pagina 89

Fig. 78 Vista dall’alto dell’ampio tavolo da lavoro, che occlude quasi completamente il pavimento sottostante. - Pagina 90
Fig. 79 La stanza in cui l’utente vive lo scenario implementativo: si ha l’impressione di essere in un laboratorio che si affaccia su una stanza reale. - Pagina 91
Fig. 80 A sinistra, un oggetto metallico se in scena non c’è un reflection probe; a destra le riflessioni generate grazie al probe. - Pagina 93
Fig. 81 Sopra: i keypoints se il bool Realtime Processing è settato su true. Sotto: l’importante ritardo se il bool è false. - Pagina 95
Fig. 82 Lo scenario implementativo inizia in VR e la sfera di fumo si trova nel punto in cui l’utente è situato all’interno della stanza reale. - Pagina 96
Fig. 83 Una maschera sovraimpressa al volto della persona. - Pagina 97
Fig. 84 I keypoints sono mappati su una sfera di raggio ridotto e quindi acquisiscono un’informazione di profondità. - Pagina 98
Fig. 85 Il modello umanoide sembra correttamente muoversi verso il punto nello spazio 3D dove si trova il performer. - Pagina 99
Fig. 86 Dall’alto al basso: l’oggetto poligonale scalabile, la pulsantiera con cui giocare ad “acchiappa la talpa”, l’avatar tracciato con due oggetti sintetici. - Pagina 102
Fig. 87 La configurazione 3 prevede che le interazioni con le talpe avvengano tramite un ray casting. - Pagina 103
Fig. 88a La media arrotondata delle 5 domande sul senso di presenza e immersione. - Pagina 104
Fig. 88b Domanda Q4. - Pagina 105
Fig. 88c Domanda Q3. Il risultato è stato invertito per mantenere la coerenza con le restanti domande e quindi un valore alto corrisponde a una valutazione positiva. - Pagina 106
Fig. 89a La media arrotondata delle 5 domande sull’efficacia dell’interazione. - Pagina 107
Fig. 89b Domanda Q2 - Pagina 108
Fig. 89c Domanda Q5 - Pagina 109
Fig 90a La media arrotondata delle 3 domande sul compositing, divisa per configurazioni. - Pagina 110
Fig 90b I risultati di MR360. - Immagine presa dal paper in questione - Pagina 111

Fig. 91 L’utente viene toccato sul braccio mentre gli vengono sottoposte delle domande. - Pagina 112
Fig. 92a La media arrotondata delle 5 domande sul livello di auto-coscienza. - Pagina 113
Fig. 92b Le domande Q3 e Q4. - Pagina 114
Fig. 93 A sinistra, la configurazione Realtime; a destra, quella No Realtime. - Pagina 117
Fig. 94 Domande Q1 e Q2. - Pagina 118
Fig. 95 La media arrotondata delle 3 domande sulle condizioni dinamiche. - Pagina 119
Fig. 96 Vanessa Vozzo introduce il progetto Presence presso lo Studiumlab. - Pagina 121
Fig. 97 Una pagina dello storyboard 360. - Pagina 123
Fig. 98 Una performer tocca il corpo dell’utente durante l’Azione 4. - Pagina 124
Fig. 99 Nell’Azione 6 il manichino “diventa senziente” e minaccia l’utente. - Pagina 125
Fig. 100 Il flow chart dello scenario implementativo. - Pagina 126