Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

A.a. 2025/2026

Sessione di laurea Marzo 2026

Progettazione e sviluppo di un ambiente virtuale per la simulazione di postazioni di lavoro a supporto dell'ergonomia di processo

Relatori:

Prof. Federico Manuri

Prof. Andrea Sanna

Candidato:

Giovanna Rosace

Sommario

L'integrazione della realtà virtuale nei processi di progettazione industriale permette di anticipare le criticità ergonomiche e ottimizzare le linee di assemblaggio ancor prima della loro costruzione fisica. Tuttavia, i software tradizionali per la modellazione umana digitale presentano severi limiti legati alla rigidità cinematica e alle scarse prestazioni grafiche in tempo reale. Questa tesi illustra la progettazione e lo sviluppo di un ambiente virtuale immersivo fedele alla realtà operativa basato su Unreal Engine 5.7, finalizzato alla familiarizzazione spaziale e alla validazione ergonomica proattiva nel settore automobilistico. Il sistema supera i limiti dei manichini industriali standard integrando la tecnologia MetaHuman per una rappresentazione anatomica realistica e deformabile. Per garantire la massima flessibilità operativa, l'applicativo è dotato di un sistema di tracciamento modulare gestito tramite un'interfaccia utente dedicata. L'operatore può infatti selezionare una stima della posa basata sulla posizione di visore e controller, oppure abilitare un tracciamento inerziale dell'intero corpo tramite tuta Xsens. In questa architettura, il motore grafico gestisce unicamente la simulazione visiva e fisica, demandando il calcolo degli indici biomeccanici a software analitici esterni. Durante lo sviluppo, la necessità di renderizzare complessi modelli CAD ha evidenziato i limiti computazionali dei visori mobili. Pur ricorrendo a strumenti di profilazione e tecniche di ottimizzazione avanzate, quali l'accorpamento dei materiali e la riduzione delle chiamate di disegno grafico, si è resa necessaria la migrazione da un approccio standalone a un'architettura PCVR. La piattaforma è stata validata su due specifici casi d'uso collaborativi in architettura di rete. Il primo scenario prevede l'assemblaggio di un portellone posteriore tramite manipolatore industriale. Durante questa operazione il sistema impone una rigorosa procedura a stati e logiche di manipolazione condivisa, vincolando fisicamente le traiettorie e restituendo agli operatori una realistica percezione pseudo-aptica. Il secondo scenario riguarda l'installazione di una console, focalizzandosi sulla gestione delle interferenze spaziali all'interno di un abitacolo ristretto. I risultati confermano che la piattaforma fornisce un ambiente di simulazione affidabile, all'interno del quale gli operatori possono condurre complesse procedure industriali interagendo con una versione completamente virtualizzata della postazione di lavoro.

Indice

1 Introduzione 1
- 1.1 L'evoluzione dell'industria manifatturiera . . . 1
- 1.2 Ergonomia e simulazione virtuale . . . 2
- 1.3 Sfide tecnologiche nella realtà virtuale industriale . . . 5
- 1.4 Stellantis *ErgoLab* . . . 7
- 1.5 Obiettivo della tesi . . . 7
- 1.6 Struttura della tesi . . . 8

2 Stato dell'arte 9
- 2.1 Applicazioni industriali della realtà virtuale . . . 9
- 2.2 Metodologie per la valutazione ergonomica e il software commerciale 10
- 2.3 Sistemi di tracciamento del movimento per l'analisi posturale . . . 12
- 2.4 Motori grafici nella ricerca industriale . . . 14
- 2.5 Ambienti virtuali collaborativi . . . 16
- 2.6 Interazione fisica e simulazione dei vincoli cinematici . . . 17

3 Strumenti hardware e software 19
- 3.1 Architettura generale del sistema . . . 19
- 3.2 Dispositivi di visualizzazione e interazione - Meta Quest 3 e Meta Quest Pro . . . 21
- 3.3 Acquisizione cinematica e integrazione dati - Xsens Awinda e Live Link . . . 22
- 3.4 Ambiente di sviluppo e simulazione - Unreal Engine 5.7 . . . 24
- 3.5 Ottimizzazione delle risorse 3D - Blender . . . 26
- 3.6 Rappresentazione dell'operatore digitale - MetaHuman e CLO3D . . . 27
- 3.7 Strumenti di diagnostica e profilazione delle prestazioni . . . 28

4 Sviluppo e integrazione architetturale 31
- 4.1 Studio di fattibilità: limiti dell'architettura *standalone* e migrazione a PCVR . . . 31

4.2 Setup sperimentale definitivo e ambiente di validazione 33
4.3 Sviluppo degli asset 3D e realizzazione del *Digital Twin* 35
4.4 Integrazione dell'operatore virtuale 38
4.4.1 Sincronizzazione spaziale e calibrazione ibrida MoCap-VR 43
4.4.2 Abbigliamento virtuale in CLO3D 45
4.4.3 Logica di collisione ibrida e gestione degli scollamenti 47
4.5 Architettura del software e logiche di interazione VR 48
4.5.1 Scenari operativi e requisiti di interazione 49
4.5.2 Progettazione a componenti per l'interazione modulare . . . 51
4.5.3 Sistemi di presa multipla (*multi-grab*) e cinematica vettoriale 51
4.5.4 Sistemi di feedback sensoriale e aiuti cognitivi 55
4.5.5 Logiche di sicurezza, animazione procedurale e rilevamento predittivo 56
4.6 Architettura di rete e gestione delle sessioni collaborative 60
4.6.1 Infrastruttura di hosting e Interfaccia Utente (UI) 61
4.6.2 Assegnazione dinamica dei ruoli (*GameMode*) 62
4.6.3 Replicazione e sincronizzazione degli avatar VR 63
4.6.4 Sincronizzazione spaziale e gestione dinamica del *multi-grab* 64
4.6.5 Strumenti di analisi esterna: il *VR Spectator* 65

5 Validazione funzionale e casi di studio aziendali 67
5.1 Metodologia di collaudo industriale e validazione sul campo 67
5.2 Caso studio 1: assemblaggio del portellone (manipolazione manuale e robotica) 68
5.3 Caso studio 2: installazione della console centrale e operazioni di fissaggio 70
5.4 Discussione dei risultati 72

6 Conclusioni e sviluppi futuri 73

Ringraziamenti 74

Bibliografia 77

Capitolo 1

Introduzione

Il presente capitolo introduce il contesto tecnologico e industriale della tesi. Partendo dall'evoluzione verso l'Industria 5.0 e l'importanza dei *Digital Twin* per l'ergonomia preventiva, il testo analizza i vantaggi e le sfide ingegneristiche dell'adozione della realtà virtuale (*VR*) immersiva. Vengono infine delineati gli obiettivi della tesi, sviluppata in sinergia con l'*ErgoLab* di Stellantis, e la struttura complessiva dell'elaborato.

1.1 L'evoluzione dell'industria manifatturiera

Le fabbriche moderne sono cambiate molto negli ultimi anni. Il modello di Industria 4.0 ha introdotto l'uso massiccio di robot e macchinari interconnessi. Queste nuove funzionalità hanno accelerato la produzione. Tuttavia, il lavoratore umano è stato spesso messo in ombra dalle macchine.

Oggi stiamo passando a un nuovo modello, chiamato Industria 5.0[1]. Questo sistema rimette l'essere umano al centro del lavoro (*Human-Centric*) (figura 1.1). L'obiettivo non è più quello di far svolgere ogni compito ai robot automatici. Si cerca invece una collaborazione sicura tra persone e macchinari. La salute e il comfort del lavoratore diventano elementi essenziali nella progettazione di una fabbrica.

Nel settore automobilistico, l'assemblaggio dei veicoli richiede ancora molte operazioni manuali e ripetitive. Chi progetta postazioni di lavoro deve rispettare precise regole ergonomiche. Una stazione mal progettata affatica i muscoli e può causare lesioni fisiche nel tempo. Per questo motivo, le aziende devono valutare la comodità e la sicurezza del lavoro, prima di costruire fisicamente la catena di montaggio.

In passato venivano costruiti modelli fisici in scala reale per esaminare una nuova catena di montaggio. Questi prototipi avevano materiali costosi e richiedevano

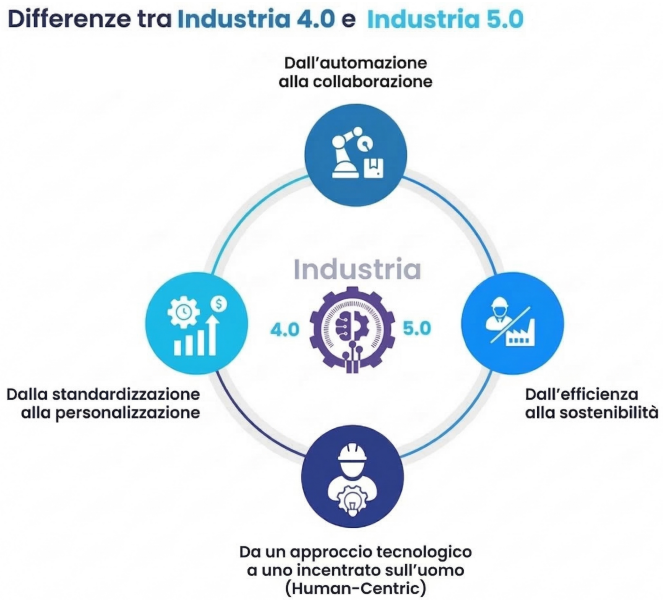


Figura 1.1: Differenze tra Industria 4.0 e Industria 5.0

molto tempo. Se i test rivelavano un problema pratico, la struttura doveva essere smantellata e rifatta. Questo processo causò enormi ritardi nella produzione.

Oggi la tecnologia offre una soluzione diversa, chiamata *Digital Twin* (figura 1.2). Il *Digital Twin* è una copia virtuale esatta di un oggetto o di un'intera fabbrica. Nell'industria automobilistica questo strumento è diventato indispensabile. Gli ingegneri ricreano al computer macchinari, utensili e automobili con misure precise al millimetro.

Grazie a questa copia virtuale è possibile effettuare prove pratiche prima di costruire gli impianti veri e propri. L'operatore può simulare il montaggio del veicolo in un ambiente tridimensionale completamente sicuro. Se si verifica un errore di progettazione o non c'è spazio per muoversi, il difetto viene corretto immediatamente nel programma. Questo metodo fa si che l'azienda riduca la spesa e renda più veloce l'arrivo di nuove auto sul mercato.

1.2 Ergonomia e simulazione virtuale

La valutazione ergonomica non è una semplice stima visiva, ma si basa su analisi precise. Gli esperti studiano come si muove il corpo umano durante il turno di lavoro. Controllano la posizione della schiena, l'inclinazione delle braccia e l'angolazione delle articolazioni. L'obiettivo è verificare che lo sforzo fisico non superi determinati limiti di sicurezza.

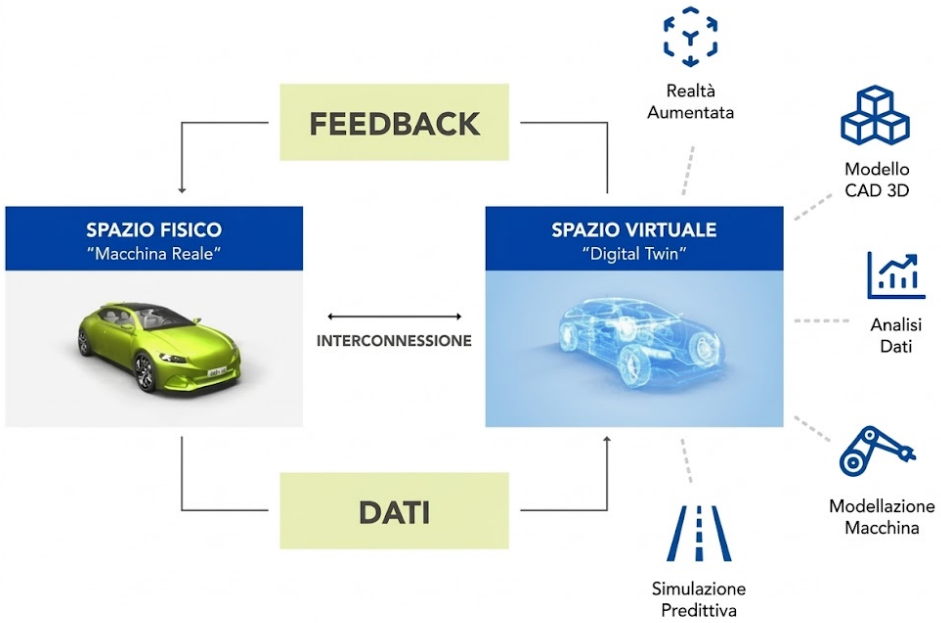


Figura 1.2: Dalla realtà al *Digital Twin*. *Fonte:[2]*

Per ottenere risultati affidabili vengono analizzate sia le posizioni fisse (posture statiche) sia i movimenti continui (posture dinamiche). Un'analisi attenta consente di decidere esattamente dove posizionare gli utensili, i carrelli e i componenti da assemblare. In questo modo, il lavoratore può raggiungere ogni elemento alla giusta altezza e distanza, lavorando in modo naturale e senza tensione muscolare.

Per effettuare un'analisi ergonomica, l'utilizzo di un *Digital Twin* visualizzato esclusivamente su un monitor tradizionale presenta dei limiti intrinseci. Come mostrato nella figura 1.3, la simulazione attraverso manichini digitali gestiti via software permette di calcolare angoli e distanze teoriche, ma non offre una valutazione del tutto affidabile della realtà operativa.

L'ingegnere o l'ergonomo che osserva la simulazione su uno schermo piatto non può percepire le reali proporzioni delle macchine né gli ingombri effettivi. Analizzare una postazione in questo modo, senza una reale interazione fisica con l'ambiente, rischia di produrre risultati parziali. Il software può indicare che un movimento è tecnicamente possibile, ma non può restituire la sensazione di sforzo, la difficoltà di accesso in spazi ristretti o la naturalezza di una postura.

In breve, simulare le postazioni basandosi solo su dati visuali bidimensionali manca della componente esperienziale necessaria per una valutazione ergonomica completa. Per questo motivo viene utilizzata la realtà virtuale.

Questa tecnologia consente all'utente di "entrare" fisicamente nel modello tridimensionale. Indossando un visore, la percezione dello spazio diventa realistica. La persona si ritrova immersa nella simulazione e può camminare intorno all'auto. Può allungare la mano, abbassarsi e stabilire immediatamente se un pezzo è difficile da raggiungere. La realtà virtuale trasforma un semplice disegno tecnico in un ambiente interattivo. Questo sistema consente di simulare decine di movimenti diversi in pochi minuti, rendendo ormai obsoleti i vecchi test su finte posizioni in legno o metallo (*mock-up*).

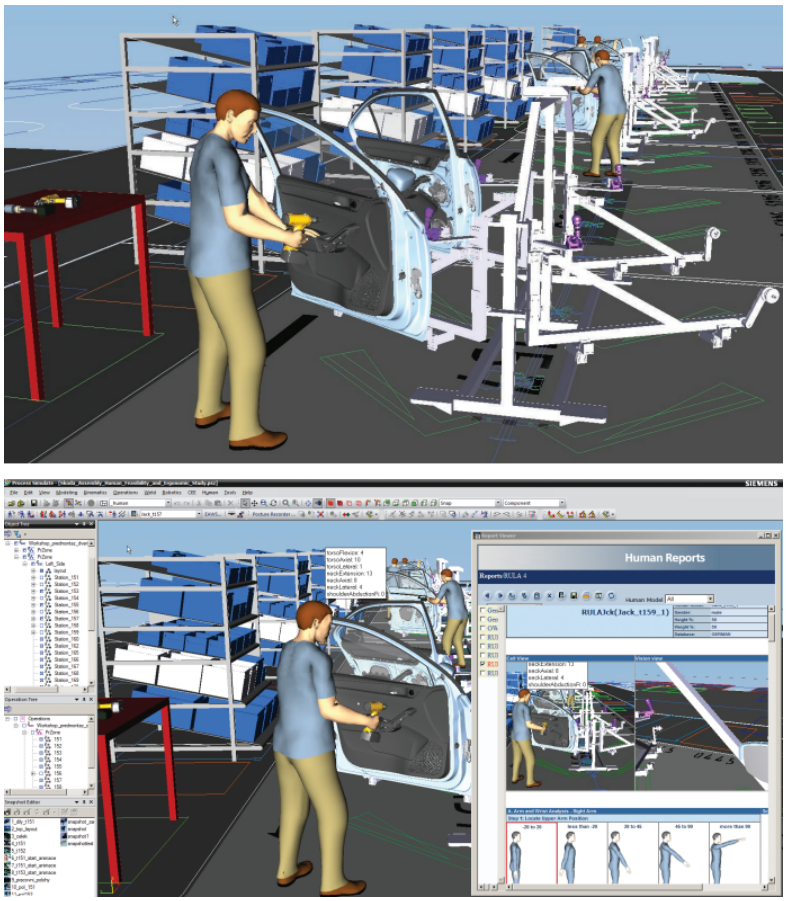


Figura 1.3: Esempio di simulazione digitale della ŠKODA Rapid tramite Process Simulate Human. *Fonte:[3]*

1.3 Sfide tecnologiche nella realtà virtuale industriale

L'integrazione della realtà virtuale nei processi industriali richiede il superamento di diversi ostacoli tecnici, a partire dalla gestione dei dati geometrici. I modelli industriali originali sono creati con programmi matematici chiamati *CAD* (*Computer-Aided Design*). Questi disegni vengono utilizzati per costruire macchinari fisici con estrema precisione. Per questo motivo, i modelli *CAD* contengono un'enorme quantità di dettagli geometrici. Questa ricchezza di dettagli rappresenta un serio ostacolo per la realtà virtuale.

I visori *VR*, come il dispositivo Meta Quest 3, hanno risorse informatiche limitate rispetto ai grandi computer industriali. L'osservatore deve disegnare due immagini tridimensionali distinte, una per ciascun occhio. Questo processo visivo deve essere ripetuto in modo continuo e fluido per decine di volte al secondo. Se il modello geometrico è troppo complesso, il processo di visualizzazione non riesce a elaborare i dati abbastanza velocemente.

Questo ritardo provoca un drastico calo delle prestazioni. Rallentando il sistema si generano immagini a scatti. In queste condizioni, l'ambiente virtuale si muove più tardi rispetto ai movimenti effettivi della testa dell'utente. Questa asincronia inganna il cervello umano e provoca un'immediata sensazione di nausea, un disturbo chiamato *motion sickness*. Di conseguenza, i file CAD originali non possono essere utilizzati direttamente nella simulazione. Devono prima essere attentamente semplificati e ottimizzati.

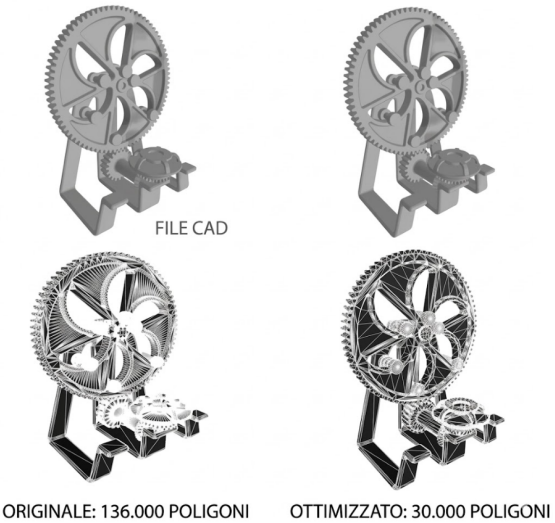


Figura 1.4: Differenza di complessità tra file CAD e mesh ottimizzate. *Fonte:[4]*

Oltre all'ottimizzazione dell'ambiente virtuale e dei macchinari, un'altra sfida cruciale riguarda la corretta rappresentazione dell'utente stesso. I programmi classici per lo studio dell'ergonomia utilizzano manichini molto semplici. Questi modelli tridimensionali sono costituiti da figure geometriche di base, come cilindri e sfere. Vengono utilizzati solo per calcolare l'angolo dei giunti su uno schermo piatto. Tuttavia, questi manichini creano un serio problema all'interno della realtà virtuale.

I manichini standard non hanno muscoli e non indossano abiti ingombranti. Non occupano lo stesso spazio fisico di un vero lavoratore. Di conseguenza, l'utente con il visore non percepisce le reali dimensioni del suo corpo virtuale. L'operatore non può dire con certezza se il suo braccio o la sua tuta toccano un ostacolo durante il montaggio dell'auto. Questa mancanza di realismo non consente di trarre conclusioni certe dalle prove di montabilità, raggiungibilità, accessibilità ed ergonomiche in generale.

Per risolvere questo problema, le nuove simulazioni utilizzano personaggi digitali fotorealistici chiamati *avatar*. Questo progetto utilizza una tecnologia avanzata chiamata MetaHuman. Questo sistema crea esseri umani digitali identici alle persone reali. Grazie a questi nuovi *avatar*, l'ingegnere vede esattamente le dimensioni reali e capisce quanto spazio serve al lavoratore per muoversi.

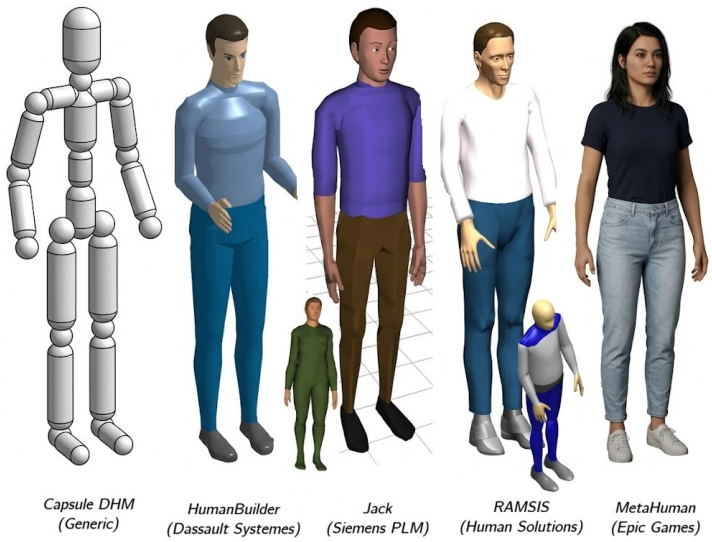


Figura 1.5: Evoluzione dei modelli DHM

Una volta raggiunto un elevato livello di realismo sia per l'ambiente che per il singolo operatore, la complessità tecnica si sposta sulla collaborazione remota. Il design industriale moderno richiede spesso il lavoro congiunto di più esperti contemporaneamente. Tuttavia, far interagire due o più persone all'interno della

stessa simulazione virtuale presenta notevoli difficoltà di calcolo. Questa modalità, chiamata *multiplayer*, richiede che ogni movimento effettuato da un utente sia visto istantaneamente da tutti gli altri partecipanti.

Il problema principale è legato alla sincronizzazione dei dati attraverso la rete. Ogni volta che un operatore muove un braccio o sposta uno strumento, il computer deve inviare queste informazioni agli altri visori collegati. Se la connessione internet non è molto veloce, si verifica un ritardo chiamato latenza. A causa della latenza, gli altri partecipanti vedono i movimenti degli utenti a scatti o nelle posizioni sbagliate. Questo difetto rovina il coordinamento e rende impossibile lavorare insieme con precisione.

Oltre al ritardo, il sistema deve gestire la continuità dei movimenti del corpo. Se il programma non è scritto correttamente, gli avatar degli altri giocatori potrebbero apparire in pose innaturali o bloccarsi durante la simulazione. Per risolvere queste sfide è necessario progettare un'architettura di rete specifica. Questo sistema deve essere in grado di pulire i dati dai disturbi della connessione e di visualizzare movimenti fluidi e naturali in tempo reale a tutti gli utenti coinvolti.

1.4 Stellantis *ErgoLab*

Il presente progetto è il risultato della collaborazione con *Stellantis*, multinazionale del settore automotive nata dall'unione dei gruppi *Fiat Chrysler Automobiles* e *PSA*.

L'attività si colloca all'interno dell'*ErgoLab*, il laboratorio di Ergonomia della region *Enlarged Europe* di Stellantis, che fornisce dati e informazioni affidabili per la progettazione di postazioni di lavoro e sistemi tecnologici nel pieno rispetto dei principi ergonomici e dei requisiti normativi. Inoltre, l'ErgoLab supporta la validazione scientifica di soluzioni innovative.

Il progetto nasce dall'esigenza aziendale di simulare in realtà virtuale immersiva alcune attività di assemblaggio particolarmente critiche dal punto di vista ergonomico. L'obiettivo è individuare, attraverso simulazioni iterative, le condizioni di lavoro ottimali e acquisire parametri posturali accurati, utili per le analisi ergonomiche.

In questo contesto, l'azienda ha fornito un contributo determinante, mettendo a disposizione strutture e attrezzature per il motion capture, oltre ai modelli CAD del veicolo, dei sistemi robotici e della strumentazione utilizzata, supportando le fasi di sviluppo e sperimentazione dell'applicativo.

1.5 Obiettivo della tesi

L'obiettivo principale del presente elaborato di tesi è la progettazione e lo sviluppo di un ambiente virtuale utile a supportare l'analisi ergonomica preventiva nelle linee

di assemblaggio industriali. Nello specifico, la ricerca mira a superare i limiti delle attuali postazioni di collaudo fisiche (costose e realizzabili solo in fasi avanzate della progettazione) e delle simulazioni digitali su schermo, proponendo un'architettura PCVR (*Personal Computer Virtual Reality*) collaborativa focalizzata sull'efficienza computazionale e sull'accuratezza dimensionale.

Attraverso l'integrazione di sistemi di tracciamento inerziale a bassissima latenza e la rappresentazione di avatar digitali antropometricamente accurati, il sistema permette agli ingegneri di processo di valutare le criticità posturali, le collisioni e le interferenze spaziali all'interno dell'abitacolo veicolare in tempo reale.

Per esaminare e validare l'efficacia del sistema, sono stati analizzati due specifici casi d'uso. Il primo scenario prevede l'assemblaggio di un portellone posteriore tramite manipolatore industriale. Durante questa operazione il sistema impone una rigorosa procedura a stati e logiche di manipolazione condivisa. Il secondo scenario riguarda l'installazione di una console, focalizzandosi sulla gestione delle interferenze spaziali all'interno di un abitacolo ristretto.

I risultati confermano che la piattaforma fornisce un ambiente di simulazione affidabile, all'interno del quale gli operatori possono condurre complesse procedure industriali interagendo con una versione completamente virtualizzata della postazione di lavoro. L'esito finale è, dunque, la fornitura di uno strumento ingegneristico avanzato, in grado di supportare la validazione *multi-utente* dei processi produttivi in fabbrica ancor prima della loro effettiva realizzazione fisica.

1.6 Struttura della tesi

L'elaborato è organizzato in sei capitoli, che descrivono l'intero percorso di ricerca, sviluppo e validazione del sistema.

Il capitolo 2 presenta lo stato dell'arte relativo alla realtà virtuale e ai sistemi di tracciamento nell'industria. Viene approfondito il concetto di *Digital Twin* e l'evoluzione dei modelli umani digitali per l'ergonomia.

Il capitolo 3 descrive nel dettaglio gli strumenti utilizzati. Vengono analizzate le caratteristiche hardware dei visori e della tuta inerziale, insieme alle funzionalità del motore grafico Unreal Engine 5.7 (UE) e della tecnologia MetaHuman.

Il capitolo 4 espone il cuore tecnico del progetto. In questa parte viene descritto il processo di ottimizzazione, la programmazione delle logiche di interazione e lo sviluppo dell'architettura di rete per il multigiocatore.

Il capitolo 5 è dedicato alla validazione funzionale. Vengono presentati i test condotti sui casi d'uso del portellone automobilistico e della console centrale, riassumendo i risultati ottenuti.

Infine, il capitolo 6 sintetizza i traguardi raggiunti e propone possibili sviluppi futuri per l'integrazione di nuove funzionalità all'interno della piattaforma virtuale.

Capitolo 2

Stato dell'arte

Sulla base delle premesse delineate nel capitolo precedente, che hanno evidenziato la necessità di mettere il lavoratore al centro del progetto tramite i *Digital Twin*, questo capitolo analizza le tecnologie attuali per la simulazione dell'ergonomia. Il testo illustra le applicazioni della realtà virtuale nell'industria, il funzionamento dei manichini digitali e i sistemi di tracciamento del corpo, evidenziando i pregi e i difetti delle soluzioni oggi disponibili. Vengono inoltre esaminati il ruolo dei motori grafici, le architetture per la collaborazione tra più utenti e i metodi per simulare i vincoli fisici dei macchinari. L'analisi dei limiti tecnologici attuali costituisce il presupposto fondamentale per le scelte di progettazione e sviluppo descritte nei capitoli successivi.

2.1 Applicazioni industriali della realtà virtuale

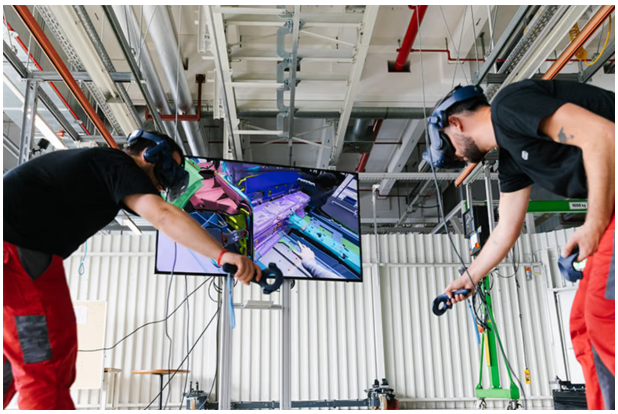
L'integrazione delle tecnologie immersive nei settori ad alta complessità sta ridefinendo le metodologie di progettazione. La letteratura scientifica più recente evidenzia come l'uso combinato della realtà virtuale e dei modelli umani digitali ci consenta di anticipare problemi fisici critici fin dalle primissime fasi di sviluppo del prodotto [5]. Questo cambiamento di paradigma trasforma l'analisi posturale da una semplice attività correttiva a un processo rigorosamente proattivo.

Nel settore aerospaziale, la NASA utilizza motori grafici per ricreare ambienti a gravità zero o spazi confinati [6], come si può osservare nella figura 2.1a. Gli ingegneri utilizzano queste simulazioni per studiare le interazioni degli astronauti con gli strumenti di bordo. La fedeltà spaziale garantita dai visori consente una mappatura accurata dei limiti di raggiungibilità manuale e visiva. Questi parametri sono essenziali per operare in habitat in cui la tolleranza agli errori è minima.

Nell'industria automobilistica, la realtà virtuale viene ora utilizzata per supportare tabelle di pianificazione collaborativa, note nell'industria come laboratori 3P

(*Processo di Preparazione della Produzione*) [7]. In questi scenari virtuali, i team di produzione e i progettisti analizzano le future stazioni di assemblaggio in scala 1:1, un esempio nella figura 2.1b. L'operatore testa le sequenze di assemblaggio, valuta le dimensioni e verifica l'interazione con l'apparecchiatura. Recenti studi di settore dimostrano che l'immersione tridimensionale fornisce dati biomeccanici molto più affidabili rispetto alle tradizionali stime dei monitor bidimensionali [8]. L'uso sistematico di queste tecnologie accelera i tempi di risoluzione delle linee di produzione e garantisce un ambiente di lavoro ergonomicamente validato ancor prima della sua costruzione.

(a) Design e training NASA



(b) Test dei processi di assemblaggio e logistica del gruppo Audi. *Fonte:[9]*

Figura 2.1: Esempi di utilizzo della tecnologia VR in ambito industriale

2.2 Metodologie per la valutazione ergonomica e il software commerciale

La prevenzione dei disturbi muscolo-scheletrici richiede l'applicazione di rigorose metodologie di calcolo. Nel corso degli anni, la ricerca ha sviluppato diversi protocolli per quantificare il rischio biomeccanico sul posto di lavoro. L'integrazione di questi protocolli all'interno di ambienti virtuali rappresenta oggi una delle principali sfide ingegneristiche. Alla base di queste misurazioni vi è la valutazione del rischio posturale, che si basa su standard internazionali ampiamente documentati. Nel settore manifatturiero, i metodi di osservazione più diffusi includono RULA, REBA ed EAWS. Il protocollo RULA (*Rapid Upper Limb Assessment*) analizza specificamente il carico su braccia, polsi e collo. Il metodo REBA (*Rapid Entire Body Assessment*) estende invece la valutazione all'intero corpo, compresa l'analisi degli arti inferiori.

Figura 2.2: Criteri di valutazione EAWS. *Fonte:[10]*

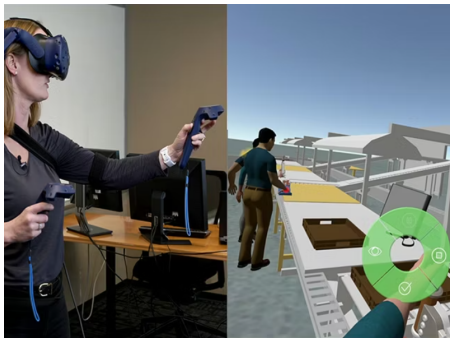
Nel settore automobilistico, lo strumento di riferimento è l'*EAWS* (*Ergonomic Assessment Worksheet*) [11], un metodo di valutazione ergonomica sviluppato dall'*MTM Association* e ampiamente utilizzato in ambito industriale per analizzare il carico biomeccanico a cui è sottoposto un operatore durante un'attività lavorativa.

L'EAWS valuta il carico biomeccanico attraverso quattro sezioni principali: posture, forza, movimentazione dei carichi e arti superiori. A ciascuna sezione viene attribuito un punteggio che contribuisce alla determinazione di un punteggio complessivo, rappresentato tramite un sistema semaforico che consente di identificare il livello di rischio ergonomico dell'attività analizzata. I criteri di valutazione sono riportati nella figura 2.2.

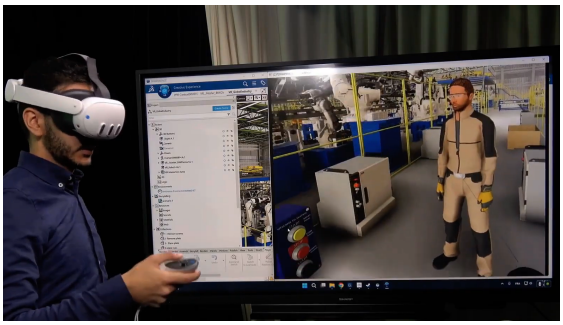
Tuttavia, l'applicazione manuale di questi metodi è lenta e soggetta a errori di misurazione da parte dell'analista. Proprio per superare queste inefficienze e automatizzare il calcolo degli indici ergonomici, l'industria adotta sistemi di *Digital Human Modeling* (DHM). Le piattaforme commerciali storiche, come Siemens Tecnomatix Jack [12] o Dassault Systèmes DELMIA [13], rappresentano lo standard per l'ingegneria di processo. Questi strumenti offrono calcoli biomeccanici matematicamente impeccabili. La figura 2.3 mostra due esempi del loro utilizzo in ambito VR.

Tuttavia, la letteratura scientifica evidenzia diverse limitazioni operative nell'uso di questi software in scenari immersivi [15]. I manichini digitali tradizionali possiedono una cinematica rigida. I loro movimenti sono spesso robotici e innaturali. Inoltre, i motori di rendering di queste applicazioni ingegneristiche non sono ottimizzati per la realtà virtuale. Hanno difficoltà a gestire la quantità di calcoli necessari per mantenere gli *fps* (fotogrammi per secondo) alti. Una bassa frequenza di aggiornamento visivo genera latenza e provoca inevitabilmente *motion sickness* nell'utente.

Per ovviare a queste problematiche e superare i limiti cinematici dei DHM



(a) Siemens Tecnomatix Jack



(b) Dassault Systèmes DELMIA. *Fonte:[14]*

Figura 2.3: Esempi di aziende che utilizzano DHM

tradizionali, la recente letteratura scientifica evidenzia il progressivo abbandono dei modelli a corpo rigido (*rigid-body*) a favore di rappresentazioni anatomiche deformabili [16]. Nei software DHM classici, il corpo umano viene approssimato utilizzando primitive geometriche, come cilindri ed ellissoidi, collegate da giunti cinematici. Questa struttura rigida non simula la compressione naturale dei tessuti molli durante il movimento o la massa dinamica dell'abbigliamento da lavoro.

Oggi, le architetture moderne come i modelli SMPL (*Skinned Multi-Person Linear model*) e i *MetaHuman* integrano algoritmi avanzati di skinning. Le implementazioni più recenti sfruttano l'apprendimento automatico (*Machine Learning Deformer*) (figura 2.5)[17] per approssimare la deformazione muscolare in tempo reale, senza il costo computazionale di una simulazione fisica pura. Questo livello di fedeltà garantisce che il calcolo delle collisioni tra l'operatore e la scocca dell'automobile sia preciso al millimetro, requisito indispensabile per condurre analisi ergonomiche affidabili in ambienti industriali confinati.

2.3 Sistemi di tracciamento del movimento per l'analisi posturale

Il tracciamento accurato dei segmenti corporei è un requisito fondamentale per la valutazione ergonomica nella realtà virtuale. La letteratura scientifica classifica i sistemi *Motion Capture* (MoCap) in due macrocategorie principali: sistemi ottici e sistemi inerziali (figura 2.6). Ognuna di queste tecnologie presenta vantaggi e limiti specifici in campo industriale.[18]

Analizzando la prima categoria, si osserva che i visori commerciali di ultima generazione utilizzano algoritmi di tracciamento ottico del tipo *inside-out*. Le telecamere integrate nel dispositivo calcolano autonomamente la posizione della

Figura 2.4: Modello SMPL a confronto con MetaHuman

Figura 2.5: Addestramento del modello ML Deformer (UE) per la deformazione anatomica

testa e delle mani nello spazio tridimensionale. Questo approccio è estremamente pratico poiché non richiede l'installazione di sensori esterni nell'ambiente di lavoro.

Tuttavia, il tracciamento limitato a soli tre punti (testa e due controller) non è

Figura 2.6: Esempio di MoCap ottico (a sinistra) e inerziale (a destra)

sufficiente per un'analisi biomeccanica rigorosa [19]. Per animare l'intero *avatar*, il software deve stimare la posizione dei gomiti, del tronco e delle gambe utilizzando algoritmi di cinematica inversa (*IK*). Sebbene i moderni algoritmi di *Deep Learning* abbiano migliorato la stima posturale di gomiti e gambe, queste approssimazioni matematiche introducono ancora errori angolari inaccettabili per i rigidi protocolli industriali. Inoltre, le telecamere ottiche soffrono di occlusione visiva: se l'operatore posiziona le mani dietro un pannello o un macchinario, il tracciamento si interrompe.

Proprio per superare i limiti dovuti all'occlusione visiva e all'imprecisione della stima algoritmica, la ricerca industriale adotta oggi la seconda categoria tecnologica, ovvero i sistemi di tracciamento inerziale [20]. Questa tecnologia impiega reti di sensori IMU (*Inertial Measurement Unit*) (figura 2.7) posizionate direttamente sui segmenti corporei del lavoratore. Dispositivi professionali, come le tute Xsens[21], misurano l'accelerazione, la rotazione e l'orientamento magnetico di ogni singolo giunto del corpo.

L'approccio inerziale garantisce un tracciamento continuo a frequenze di campionamento molto elevate, consentendo il cosiddetto *Full-Body Tracking*. I dati cinematici non subiscono alcuna interruzione quando l'operatore lavora all'interno di abitacoli ristretti o interagisce con geometrie complesse. Recenti validazioni sperimentali confermano che la precisione angolare fornita dai sensori IMU soddisfa pienamente i rigorosi requisiti dei protocolli ergonomici industriali, come EAWS [22]. L'integrazione visiva di questi dati nel motore grafico aggira la necessità della cinematica inversa e crea una replica posturale esatta, permettendo all'operatore di percepire realisticamente il proprio corpo nello spazio virtuale.

2.4 Motori grafici nella ricerca industriale

I motori grafici sono nati originariamente per l'industria dell'intrattenimento. Oggi rappresentano uno strumento fondamentale per la ricerca e lo sviluppo ingegneristico.

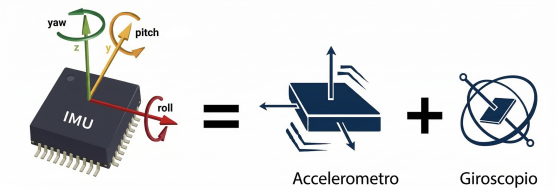


Figura 2.7: Sensore IMU. *Fonte:[23]*

La recente letteratura scientifica conferma il crescente utilizzo di piattaforme come *Unreal Engine*[24] per la simulazione industriale avanzata.

Per abilitare queste simulazioni avanzate, oggi la ricerca accademica e industriale si concentra sull'automazione dei flussi di lavoro tra dipartimenti di ingegneria e piattaforme immersive. Nel moderno *Digital Twin*, i modelli matematici originali *NURBS (Non-Uniform Rational Basis-Splines)* subiscono complessi processi di tassellazione [1] e decimazione poligonale [2]. La principale sfida tecnologica evidenziata in letteratura è quella di bilanciare la drastica riduzione del peso computazionale con la rigorosa conservazione dei dettagli topologici, come fori o giunti [25].

Studi recenti propongono l'integrazione di algoritmi avanzati per la generazione automatica di livelli di dettaglio (*LOD*). Queste tecniche consentono ai motori grafici di scalare dinamicamente la complessità geometrica in base alla distanza dell'osservatore. L'adozione di *pipeline* dedicate per l'importazione diretta e automatizzata dei formati *CAD* ha standardizzato il trasferimento dei dati. Attualmente, lo stato dell'arte per l'interoperabilità dei dati industriali si basa su protocolli standardizzati come *OpenUSD* (*Universal Scene Description*) [26]. Questa architettura consente la sincronizzazione in tempo reale tra le applicazioni *CAD* in esecuzione su computer desktop e gli ambienti *VR*, annullando i tempi di conversione e importazione dei modelli tridimensionali [27]. Questo approccio metodologico garantisce che eventuali modifiche al progetto originale si riflettano immediatamente nell'ambiente virtuale, ottimizzando le risorse hardware senza compromettere l'ispezione visiva dei componenti.

Nonostante queste ottimizzazioni software, la gestione di modelli così complessi richiede motori grafici di altissimo livello. Tra le soluzioni disponibili, *Unreal Engine 5*[28] si distingue per le sue avanzate capacità di calcolo. La piattaforma integra sistemi fisici complessi e algoritmi di illuminazione dinamica in tempo reale; caratteristiche che aumentano il realismo della simulazione e migliorano la

[1] La *tassellazione* converte le superfici matematiche continue in una maglia di poligoni per il rendering 3D.

[2] La *decimazione poligonale* ne riduce il numero per alleggerire il carico computazionale, preservando la geometria originale.

percezione della profondità spaziale da parte dell'utente [29].

Tuttavia, l'elaborazione grafica di una stazione di assemblaggio automobilistica con questo livello di realismo richiede notevoli risorse hardware. I processori mobili integrati nei visori *standalone*, come il Meta Quest 3, non possiedono una potenza di calcolo sufficiente per gestire milioni di poligoni. Per questo motivo, le applicazioni industriali adottano l'architettura *PCVR*. Il visore viene collegato a una *workstation* esterna ad alte prestazioni.

In configurazione PCVR, l'elaborazione dell'intera logica grafica e fisica è a carico della workstation, mentre il visore funge esclusivamente da periferica di input/output, trasmettendo i comandi dell'utente e ricevendo il flusso video in tempo reale. Questa netta separazione dei carichi computazionali permette di massimizzare le prestazioni hardware aggirando i limiti dei dispositivi *standalone* [30]. All'interno di questo paradigma modulare, il motore grafico è delegato unicamente al rendering fotorealistico dell'ambiente. Parallelamente, l'elaborazione dei dati cinematici per le valutazioni biomeccaniche ed ergonomiche viene demandata a software analitici esterni, svincolando così il rendering visivo dalla complessità dell'analisi matematica.

2.5 Ambienti virtuali collaborativi

La progettazione di una catena di montaggio richiede la collaborazione simultanea di più figure professionali. L'ingegneria di processo odierna si basa su ambienti virtuali collaborativi, fondati su architetture eterogenee. La letteratura scientifica evidenzia come la simulazione distribuita rappresenti lo standard per la validazione preventiva delle *workstation* [31].

All'interno di questo paradigma, la simulazione immersiva condivisa diventa essenziale per analizzare le operazioni di assemblaggio in spazi ristretti. In questi scenari, il sistema deve garantire che due o più operatori possano coordinare i propri movimenti senza incorrere in interferenze spaziali. Come descritto in precedenza, l'uso di architetture basate su *OpenUSD* non solo ottimizza le risorse hardware, ma abilita la creazione di veri e propri ambienti virtuali collaborativi, dove il team di progettazione può visualizzare e valutare istantaneamente ogni modifica apportata al modello originale.

Oltre alla condivisione in tempo reale del dato ingegneristico, un ulteriore e fondamentale sviluppo tecnologico per il lavoro di squadra è rappresentato dalla *Co-location*. Utilizzando ancore spaziali (*spatial anchors*), il sistema calibra lo stesso sistema di riferimento per più visori presenti nello stesso ambiente fisico. Ciò consente a due operatori di lavorare fianco a fianco nella realtà, interagendo sullo stesso *Digital Twin* olografico senza il rischio di collisioni fisiche accidentali.

Se la *Co-location* risolve il problema della condivisione dello spazio fisico, la gestione della precisione posturale e della latenza di rete richiede accorgimenti

software altrettanto complessi. Per affrontare questa sfida, la ricerca ha progressivamente superato i classici algoritmi di interpolazione lineare. Le piattaforme industriali più recenti implementano modelli di compensazione predittiva basati sull'intelligenza artificiale. In caso di fluttuazioni o ritardi nella trasmissione dei pacchetti di rete, il sistema stima algoritmicamente la traiettoria cinematica degli operatori. Questo approccio previene la compenetrazione visiva dei modelli, garantendo agli utenti una percezione degli ingombri costantemente affidabile e matematicamente coerente.

Sebbene questa sincronizzazione predittiva rappresenti l'attuale frontiera della ricerca, la maggior parte delle applicazioni industriali odierne si basa ancora sulle robuste architetture di replicazione native (*client-server*) fornite dai motori grafici. Queste soluzioni standardizzate, se opportunamente ottimizzate, garantiscono tuttora il miglior compromesso tra costi di sviluppo, stabilità della rete e accuratezza della simulazione ergonomica.

2.6 Interazione fisica e simulazione dei vincoli cinematici

La movimentazione di componenti automobilistici ingombranti e pesanti, come i portelloni posteriori, richiede l'uso di manipolatori industriali. Questi ausili meccanici sono progettati per liberare l'operatore dal carico, limitando al contempo le traiettorie di assemblaggio a binari e assi di rotazione specifici. Replicare fedelmente questa interazione all'interno della realtà virtuale rappresenta una delle sfide biomeccaniche più complesse.

Nei sistemi immersivi tradizionali, l'assenza di feedback di forza (*force feedback*) consente all'utente di spostare liberamente i *controller* nello spazio, violando le leggi della fisica e la meccanica dei macchinari. La letteratura scientifica propone due approcci principali per affrontare questa criticità. Il primo prevede l'utilizzo di interfacce tattili attive, come esoscheletri o bracci robotici, in grado di ripristinare una reale resistenza fisica, un esempio è visibile nella figura 2.8. Tuttavia, gli elevati costi di manutenzione e le limitazioni degli spazi di lavoro rendono queste tecnologie difficili da scalare negli attuali laboratori industriali [32].

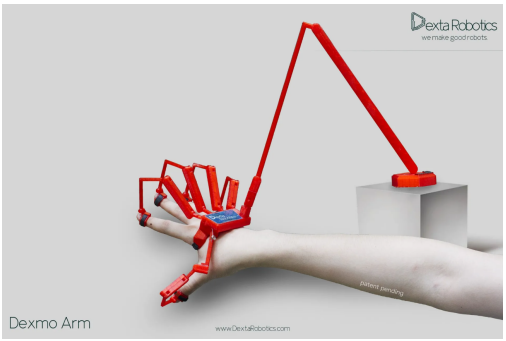


Figura 2.8: Esempio di dispositivo aptico indossabile (esoscheletro). *Fonte:[33]*

L'approccio attualmente prevalente e ampiamente standardizzato è di natura puramente software. Utilizza motori fisici in tempo reale integrati nel *Game Engine*[3] per applicare rigorosi vincoli cinematici (*physics joints*) a modelli tridimensionali. Il manipolatore virtuale è programmato per rispettare i veri gradi di libertà della macchina, bloccandone la traslazione o la rotazione al di fuori delle tolleranze ingegneristiche previste [34].

Quando l'operatore tenta di forzare una traiettoria non autorizzata, il sistema interviene bloccando l'oggetto tridimensionale. In queste situazioni si genera una discrepanza visiva temporanea tra la posizione effettiva delle mani dell'utente e la posizione delle mani dell'avatar, legate alla macchina. Recenti studi neurocognitivi dimostrano che questo disallineamento visuo-motorio genera un feedback pseudo-aptico. Il cervello umano interpreta la limitazione visiva come una reale resistenza meccanica, ripristinando la percezione da parte dell'operatore del peso e dell'inerzia del macchinario senza l'ausilio di periferiche hardware dedicate [35]. Questa simulazione matematica dei vincoli è essenziale per formare i lavoratori nelle corrette posture di spinta e trazione.

[3]Un *game engine* (o motore grafico) è un framework software che gestisce il rendering 3D in tempo reale, la simulazione fisica e l'interattività degli ambienti virtuali.

Capitolo 3

Strumenti hardware e software

Sulla base dei limiti tecnologici analizzati nel capitolo precedente, sono state operate precise scelte di progettazione per lo sviluppo del sistema. Questo capitolo illustra l'architettura hardware e software selezionata per garantire prestazioni elevate e accuratezza cinematica. Il testo descrive l'integrazione tra i visori, i sistemi di tracciamento inerziale e il motore grafico Unreal Engine 5.7, approfondendo inoltre gli strumenti utilizzati per l'ottimizzazione degli asset e la creazione degli avatar. La configurazione di questi componenti costituisce la base tecnica necessaria per l'implementazione delle logiche funzionali e dei casi d'uso descritti nei capitoli successivi.

3.1 Architettura generale del sistema

La creazione dell'ambiente virtuale immersivo ha richiesto la progettazione di un'infrastruttura hardware e software dedicata, strutturata per garantire un elevato realismo visivo e una trasposizione cinematica a bassissima latenza. Sebbene nelle prime fasi di sviluppo l'applicazione fosse stata progettata per funzionare secondo un'architettura *standalone*, i severi requisiti prestazionali associati all'importazione di modelli industriali hanno guidato la migrazione definitiva verso il paradigma *PCVR* (Personal Computer Virtual Reality). In questa configurazione finale, il carico computazionale è nettamente separato dai dispositivi indossati dall'utente.

A livello macroscopico, il sistema è suddiviso in tre nodi operativi principali, strettamente interconnessi per garantire uno scambio di dati continuo e ad alte prestazioni:

- **Nodo di elaborazione centrale:** costituito da una workstation ad alte prestazioni. Questo nodo rappresenta il "motore" del simulatore, in quanto ospita l'ambiente di sviluppo Unreal Engine 5.7, gestisce in tempo reale le interazioni, elabora la logica di rete per le sessioni collaborative, collegato a un router LAN (Local Area Network) e genera il flusso di rendering grafico.
- **Nodo di acquisizione dati (input):** rappresenta il sistema sensoriale del simulatore. È composto parallelamente dalla tuta inerziale, responsabile del campionamento continuo dei movimenti corporei dell'operatore, e dal sistema di tracciamento integrato nel visore. Quest'ultimo, infatti, agisce come sensore primario rilevando la posizione e la rotazione della testa dell'utente (tracciamento a 6 gradi di libertà), lavorando in sinergia con i dispositivi di controllo manuale per la gestione delle interazioni.
- **Nodo di visualizzazione (output):** costituito dai display integrati nel visore stesso, che funge da terminale ricevente bidirezionale. Svincolato dal calcolo grafico, il dispositivo decodifica e proietta sulle proprie lenti il flusso video generato dalla workstation, restituendo all'utente un riscontro visivo immersivo costantemente aggiornato in base ai propri movimenti reali della testa.

Questa chiara compartimentazione dei ruoli ha permesso di superare i limiti dell'hardware mobile, supportando i pesanti oneri di elaborazione derivanti dall'uso di modelli CAD complessi e avatar fotorealistici. Nelle sezioni seguenti verranno analizzati in dettaglio i singoli componenti tecnologici e software che costituiscono le basi di questa architettura.

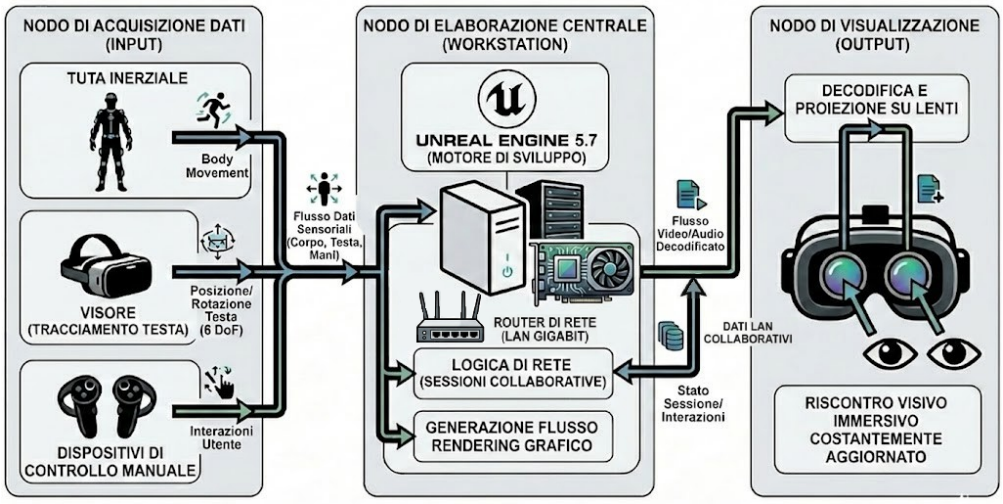


Figura 3.1: Diagramma schematico dell'architettura

3.2 Dispositivi di visualizzazione e interazione - Meta Quest 3 e Meta Quest Pro

I principali dispositivi di visualizzazione e interazione selezionati per l'ambiente virtuale sono i visori Meta Quest 3 [36] e Meta Quest Pro[37]. La scelta di questi specifici *Head-Mounted Display* (HMD) è stata dettata dalla necessità di bilanciare un'elevata fedeltà visiva con un'ergonomia adeguata per sessioni di simulazione industriale prolungate. Dal punto di vista dell'output visivo, entrambi i visori integrano un sistema ottico basato su lenti Pancake, che consentono una drastica riduzione dello spessore e del peso del dispositivo rispetto alle tradizionali lenti Fresnel, migliorando contemporaneamente la nitidezza dell'immagine e riducendo le distorsioni ottiche ai bordi. I display restituiscono risoluzioni elevate (2064x2208 pixel per occhio per il Quest 3 e 1800x1920 pixel per il Quest Pro), un livello di dettaglio fondamentale per consentire all'operatore di analizzare con precisione le geometrie e le dimensioni dei modelli CAD automobilistici. Altrettanto cruciale è la frequenza di aggiornamento (*refresh rate*) impostata a 90 fotogrammi al secondo, parametro tecnico essenziale per garantire la fluidità dell'esperienza immersiva e prevenire l'insorgenza della chinetosi (*motion sickness*) durante i movimenti della testa.

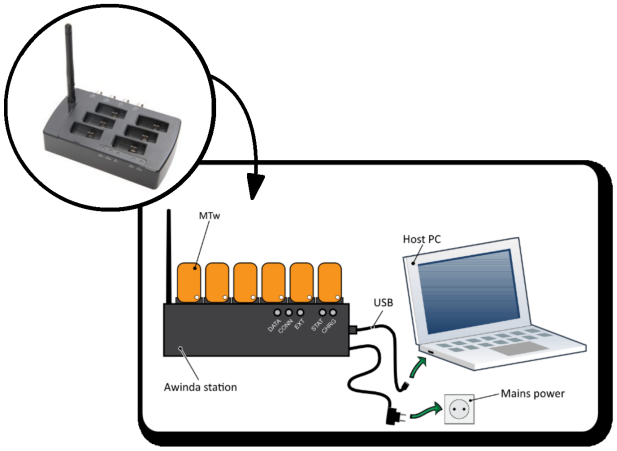
Oltre a fungere da terminali di visualizzazione, i due Meta Quest funzionano come sensori di input spaziale primari. I dispositivi utilizzano un'architettura di tracciamento *inside-out* gestita da una serie di sensori e telecamere integrate, che elaborano la posizione e la rotazione della testa dell'utente in tempo reale nello spazio fisico con sei gradi di libertà (6-DoF). Parallelamente, ai controller forniti viene affidata l'interazione manuale standard: mentre il Quest 3 utilizza controller tracciati otticamente dalle telecamere del visore, il Quest Pro si avvale dei controller Touch Pro, dotati di telecamere integrate per un tracciamento autonomo a 360 gradi. I dati posizionali di visore e controller permettono inoltre di stimare l'intera posa corporea tramite IK, senza ricorrere all'uso delle telecamere e quindi occlusioni. Questa soluzione viene impiegata per tracciare eventuali utenti collaborativi all'interno della simulazione, mentre per l'operatore principale l'acquisizione cinematica è affidata alla tuta inerziale, rilegando l'uso dei controller alla sola interazione con l'interfaccia (UI Widget), alla manipolazione degli elementi di scena e alla eventuale navigazione.

Infine, impiegando l'applicazione Meta Link [38] per interfacciare i visori con la workstation esterna, l'unità di calcolo integrata nei caschetti funge esclusivamente da nodo di comunicazione. Nello specifico, essa si occupa di trasmettere i dati di tracciamento ad Unreal Engine e di decodificare, a bassissima latenza, il flusso video ad alta risoluzione renderizzato dal computer.

3.3 Acquisizione cinematica e integrazione dati - Xsens Awinda e Live Link

Il cuore tecnologico del monitoraggio ergonomico della postura è il sistema di tracciamento inerziale Xsens Awinda [39]. A differenza dei sistemi ottici *motion capture*, che si basano su telecamere esterne soggette a frequenti occlusioni visive in spazi ristretti, la tecnologia inerziale funziona tramite una rete di diciassette sensori indossabili *IMU*. Ogni sensore integra un accelerometro triassiale, un giroscopio e un magnetometro, consentendo di calcolare con estrema precisione l'orientamento e l'accelerazione di ciascun segmento corporeo dell'operatore. I dati grezzi vengono trasmessi in tempo reale tramite protocollo wireless a una stazione ricevente collegata alla postazione di lavoro, che elabora la cinematica dell'intero corpo con sei gradi di libertà per ogni articolazione. La configurazione completa è osservabile nella figura 3.2.

(a) Sensori **(b)** Posizione sensori



(c) Awinda Sistem - ricevitore

Figura 3.2: Sistema di tracciamento MoCap Xsens Awinda

L'integrazione di questo enorme flusso di dati cinematici nell'ambiente di simulazione è resa possibile dal software *MVN Analyze*(figura 3.3) e dal protocollo *Live Link* [40]. Il software proprietario Xsens funge da motore di elaborazione intermedio, riceve segnali dai sensori, applica algoritmi di filtraggio per correggere eventuali derive magnetiche e ricostruisce lo scheletro biomeccanico dell'utente. Una volta convalidati, i dati vengono instradati all'Unreal Engine tramite una connessione di rete locale basata su pacchetti *UDP* (*User Datagram Protocol*), scelta per massimizzare la velocità di trasmissione e ridurre al minimo il ritardo tra il movimento effettivo e la risposta dell'avatar virtuale.

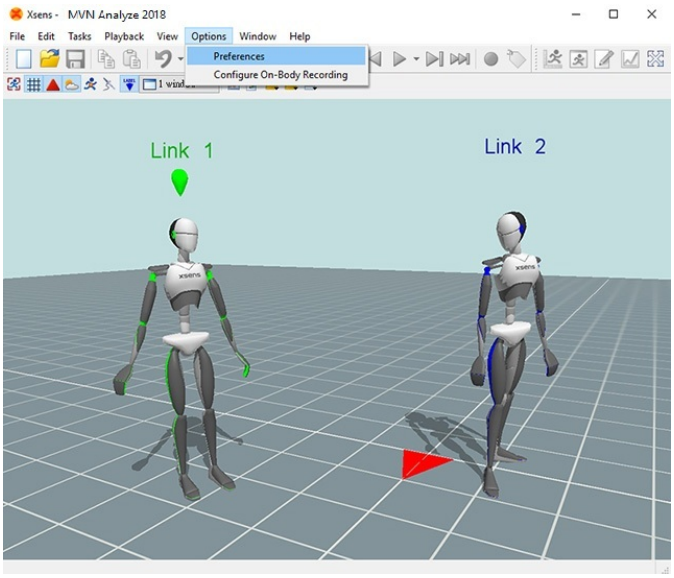
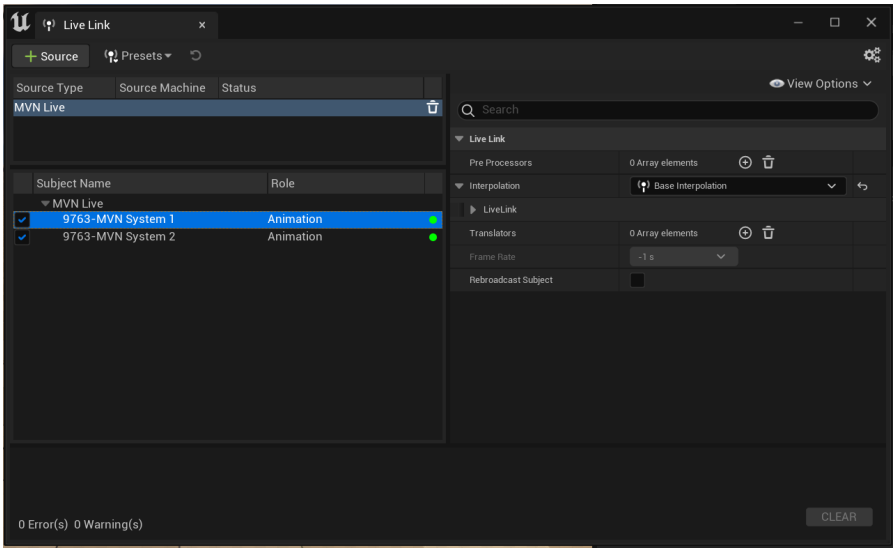
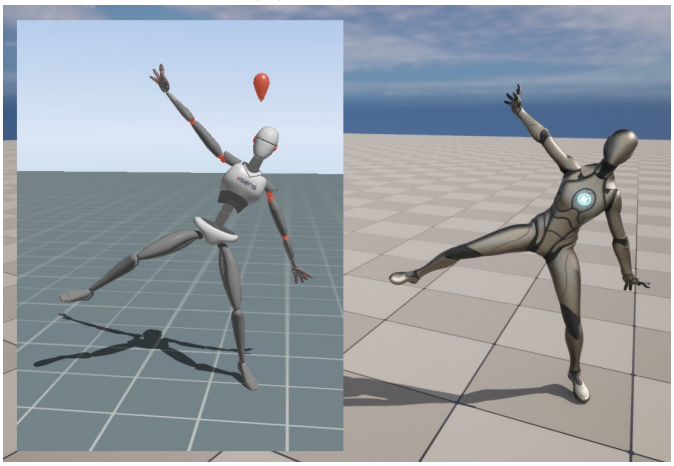


Figura 3.3: Interfaccia software MVN Analyze. *Fonte:[41]*

All'interno del motore grafico, il modulo *Live Link* [42], attraverso il plugin *Xsens Unreal Live Link Plugin*[43] (figura 3.4a), funge da interfaccia di ricezione dinamica. Permette di mappare istantaneamente le rotazioni delle ossa dello scheletro di Xsens sulle articolazioni corrispondenti del modello umano digitale. Questa architettura di comunicazione trasparente consente una netta separazione della fase di acquisizione fisica, dalla logica di visualizzazione. L'animazione dell'operatore virtuale non è precalcolata, ma è il risultato diretto e sincrono dei movimenti eseguiti nel mondo reale, come è visibile nella figura 3.4b. Questa accuratezza è un requisito fondamentale per la tesi, in quanto consente una visualizzazione e analisi ergonomica fedele, garantendo che qualsiasi interferenza o sforzo posturale rilevato nella simulazione corrisponda ad una reale criticità sulla linea di assemblaggio.



(a) Live Link



(b) Risultato del tracciamento reale in UE

Figura 3.4: Integrazione di Xsens in Unreal Engine

3.4 Ambiente di sviluppo e simulazione - Unreal Engine 5.7

Sviluppato da Epic Games, Unreal Engine[24] è uno dei motori grafici 3D in tempo reale più avanzati e diffusi al mondo. Sebbene nato originariamente per l'industria videoludica, nel corso degli anni si è evoluto in una piattaforma di simulazione trasversale, diventando un vero e proprio standard industriale per la creazione di *Digital Twin*, per la realtà virtuale e per la validazione ergonomica nei settori

manifatturiero e automobilistico. Il software mette a disposizione degli sviluppatori un ambiente di calcolo ad alte prestazioni, capace di elaborare simulazioni fisiche complesse, illuminazione dinamica e geometrie poligonali ad altissima densità.

La scelta della versione 5.7 [28] come piattaforma di sviluppo per questo lavoro è stata determinata proprio dalla sua architettura modulare e dalle elevate prestazioni del suo rendering in tempo reale. A differenza di altri motori grafici presenti sul mercato, che spesso richiedono *plugin* di terze parti per gestire flussi operativi complessi, Unreal Engine offre un ecosistema integrato in modo nativo. Questa funzionalità consente di gestire contemporaneamente, e all'interno dello stesso ambiente di sviluppo, l'acquisizione di dati cinematici esterni (tramite il protocollo *Live Link*) e il framework proprietario per la generazione di avatar fotorealistici.

A livello di interfaccia hardware, un requisito fondamentale per l'architettura del sistema è stato il supporto nativo e ottimizzato per lo standard *OpenXR*, un'API pubblica che funge da interfaccia standardizzata tra il motore grafico e i dispositivi immersivi. All'interno dell'ecosistema Unreal, questo livello di astrazione ha garantito la piena compatibilità *plug-and-play* con l'infrastruttura Meta Quest e PCVR, ottimizzando l'invio dei frame ai display degli spettatori e riducendo al minimo la latenza (la cosiddetta latenza *motion-to-photon*).

Dal punto di vista fisico e cinematico, un pilastro fondamentale del software è rappresentato dal motore interno *Chaos*. Sebbene non siano state attivate simulazioni fisiche dinamiche di massa (come il calcolo gravitazionale su ogni singolo componente CAD) per ottimizzare le risorse di calcolo, il motore *Chaos* è stato sfruttato per definire i limiti di movimento dei macchinari industriali e valutare le intersezioni dei volumi spaziali. Questa rigorosa gestione di semplici collisioni e vincoli ha permesso di implementare l'effetto pseudo-aptico. Alterando la reattività visiva degli oggetti vincolati rispetto alla posizione reale della mano dell'utente, il sistema inganna la percezione sensoriale, inducendo una sensazione tangibile di peso e inerzia senza l'ausilio di esoscheletri o attuatori tattili hardware.

Un ulteriore aspetto cruciale, essenziale per la validazione di procedure industriali collaborative, riguarda l'infrastruttura di rete nativa basata su un'architettura *client-server*. Attraverso il sistema di replicazione delle variabili e l'uso di *Remote Procedure Calls* (RPC), il motore garantisce la sincronizzazione a livello millimetrico dello stato dell'oggetto tra diverse istanze dell'applicazione. Questa implementazione consente a più operatori di coesistere nello stesso ambiente virtuale e di manipolare simultaneamente lo stesso componente meccanico, supportando la logica avanzata *multi-grab*.

Per lo sviluppo della logica applicativa si è scelto di adottare il sistema di *Visual Scripting* nativo del motore, noto come *Blueprints*, un esempio nella figura 3.5. Questi offrono un livello di astrazione ideale per la prototipazione rapida e l'implementazione di architetture *event-driven*, tipiche delle interazioni in realtà virtuale. La natura visiva a nodi ha enormemente facilitato la fase di *debugging* in

tempo reale, permettendo di tracciare visivamente il flusso di esecuzione durante le complesse transizioni della macchina a stati direttiva e durante la gestione della logica di *multi-grab* in rete. Tale approccio ha consentito di mantenere una struttura logica pulita, modulare e facilmente scalabile, senza incorrere nei tempi di compilazione prolungati tipici dello sviluppo tradizionale in C++.

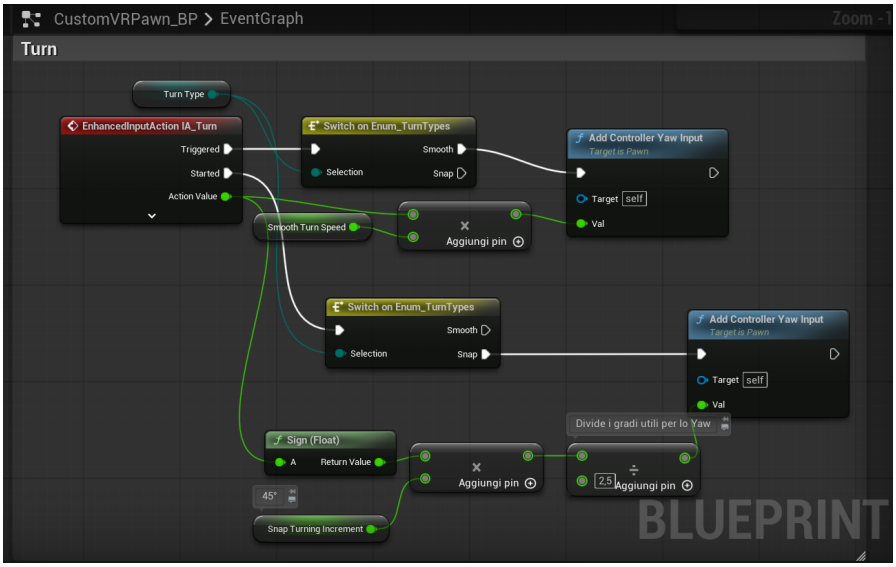


Figura 3.5: Esempio di Blueprint per ruotare il pawn

3.5 Ottimizzazione delle risorse 3D - Blender

Blender [44] è una suite software *open-source* avanzata dedicata alla modellazione tridimensionale, ampiamente utilizzata sia in ambito videoludico che industriale per la gestione e la manipolazione di geometrie poligonali.

La transizione dei modelli ingegneristici dal formato CAD originale all'ambiente di simulazione in tempo reale ha richiesto l'uso di *Blender* come software ponte per la preparazione e l'ottimizzazione di *asset* tridimensionali. I modelli industriali nativi, come le carrozzerie automobilistiche o i manipolatori robotici, sono tipicamente caratterizzati da geometrie matematiche complesse (NURBS) o mesh poligonali ad altissima densità. Tali formati sono intrinsecamente incompatibili con i rigidi vincoli prestazionali imposti dalla realtà virtuale, dove mantenere fps alti è fondamentale per la stabilità dei visori Meta Quest e per il comfort dell'operatore.

Per colmare questa lacuna tecnica, Blender è stato utilizzato per eseguire una drastica semplificazione geometrica dei macchinari e dei componenti del veicolo.

Attraverso tecniche di decimazione poligonale e *retopology*[1], è stato possibile ridurre significativamente il conteggio complessivo di vertici e triangoli (il cosiddetto *polycount*), alleggerendo il carico sulla memoria video senza tuttavia compromettere la fedeltà volumetrica e dimensionale degli oggetti, essenziale per la corretta valutazione delle dimensioni spaziali.

Parallelamente all'ottimizzazione geometrica, il software ha svolto un ruolo cruciale nella gestione dell'aspetto visivo dei modelli attraverso i processi *Texture Baking* e *Material Merging*. Assegnare più materiali distinti a un singolo componente CAD genera infatti un numero eccessivo di chiamate di disegno (*draw calls*), che rappresenta uno dei principali colli di bottiglia per lo sviluppo VR. Utilizzando Blender, le proprietà fisiche e visive delle superfici (come colore, rugosità e riflessione metallica) sono state "cotte" e trasferite in mappe di texture bidimensionali unificate e leggere.

Questa rigorosa preparazione preliminare degli asset si è rivelata una scelta ingegneristica fondamentale, consentendo l'importazione di geometrie altamente ottimizzate in Unreal Engine 5.7. Questo ha garantito che le risorse di calcolo della workstation potessero essere interamente dedicate al rendering stereoscopico a bassa latenza, alla logica di rete *client-server* e all'elaborazione in tempo reale dei dati cinematici dal sistema Xsens.

3.6 Rappresentazione dell'operatore digitale - MetaHuman e CLO3D

Per condurre un'analisi ergonomica affidabile all'interno dell'ambiente virtuale, è fondamentale disporre di una rappresentazione digitale dell'operatore che rispetti rigorosi canoni di fedeltà anatomica e cinematica. A questo scopo è stato adottato il framework *MetaHuman* [45] integrato in Unreal Engine. Questo ecosistema ci consente di generare *avatar* tridimensionali fotorealistici con una topologia poligonale ottimizzata e, fattore cruciale per questo lavoro, una struttura scheletrica complessa (*rigging*) progettata nativamente per interfacciarsi con il flusso di dati inerziale proveniente dal modulo Xsens. L'uso di un modello antropometricamente accurato garantisce che gli angoli articolari, le torsioni e le posture assunte durante la simulazione spaziale riflettano fedelmente la biomeccanica umana reale.

Tuttavia, l'accuratezza dell'analisi degli ingombri in ambito automobilistico non dipende esclusivamente dalla corporatura dell'operatore, ma anche dall'abbigliamento da lavoro indossato, che introduce volumi aggiuntivi negli spazi ristretti

[1]La *retopology* è il processo di ricostruzione della maglia poligonale di un modello 3D, volto a riorganizzarne la topologia per ottimizzare le prestazioni computazionali e la qualità visiva nel rendering in tempo reale.

dell'abitacolo o della catena di montaggio. Per modellare questi elementi specifici è stato utilizzato *CLO3D* [46], un software avanzato di progettazione e simulazione sartoriale. Attraverso questa applicazione è stato possibile ricreare con estrema precisione indumenti industriali, valutando i volumi effettivamente occupati dai tessuti sul corpo del MetaHuman, come si osserva dall'immagine di esempio 3.6.

Dal punto di vista dell'ottimizzazione delle prestazioni per la realtà virtuale, l'inserimento di una simulazione fisica dinamica dei tessuti (*cloth simulation*) nel motore grafico avrebbe comportato un costo computazionale proibitivo per la CPU, compromettendo la stabilità degli fps. Per affrontare questa criticità, è stata scelta una tecnica di *skinning*, in cui le geometrie degli indumenti, una volta modellate e posizionate in CLO3D, sono state esportate e vincolate matematicamente alle ossa dello scheletro MetaHuman.

In questo modo, le superfici dei tessuti seguono la rotazione delle articolazioni attraverso una deformazione superficiale controllata (*mesh deformation*), senza richiedere alcun calcolo in tempo reale della dinamica dei fluidi o dei tessuti. Questa strategia di ottimizzazione ci ha permesso di preservare l'impronta volumetrica realistica, essenziale per il corretto rilevamento di collisioni e interferenze spaziali tra operatore e veicolo, eliminando al contempo il costo computazionale associato alla fisica dell'abbigliamento.

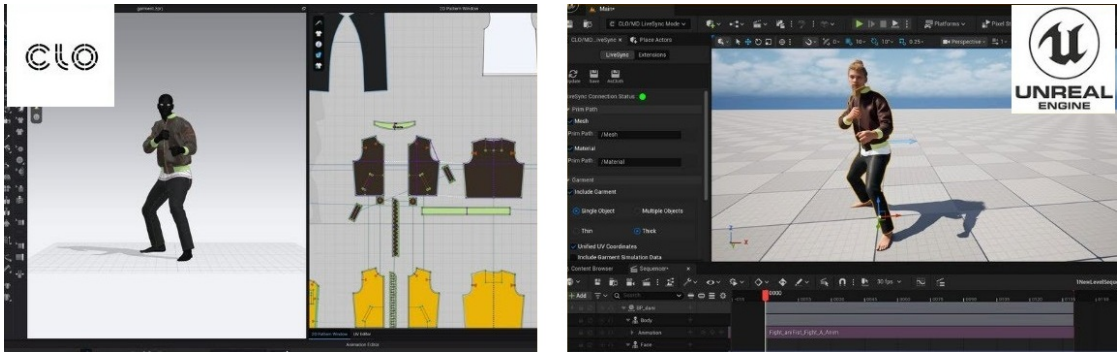


Figura 3.6: Da CLO3D a Unreal Engine e Metahuman. *Fonte:[47]*

3.7 Strumenti di diagnostica e profilazione delle prestazioni

Al fine di validare l'efficienza computazionale del sistema e giustificare le scelte architetturali descritte nelle sezioni precedenti, l'applicativo è stato sottoposto a una rigorosa fase di telemetria. L'analisi quantitativa delle prestazioni ha richiesto l'impiego di strumenti di profilazione avanzati, indispensabili per monitorare il

consumo di risorse hardware e identificare con precisione eventuali colli di bottiglia durante l'esecuzione in realtà virtuale.

Per garantire la validità scientifica e la totale riproducibilità dei test prestazionali, è stato impiegato il tool *NVIDIA VR Capture and Replay* (VCR) [48], un estratto della demo di presentazione nella figura 3.7. Questo strumento si è rivelato fondamentale per isolare le variabili umane durante la raccolta dati. Infatti registrando i dati di tracciamento esatti del visore e dei controller durante una sessione di prova, VCR ha permesso di rieseguire la medesima sequenza interattiva (*replay*) in modo automatizzato. Ciò ha consentito di misurare l'impatto prestazionale esattamente sullo stesso identico scenario operativo, garantendo che le fluttuazioni del *frame rate* o della latenza fossero imputabili esclusivamente al carico computazionale e non a variazioni casuali nei movimenti dell'utente.

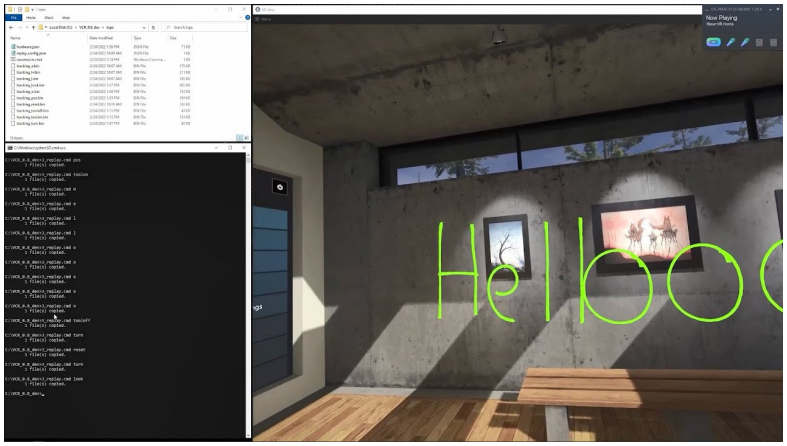


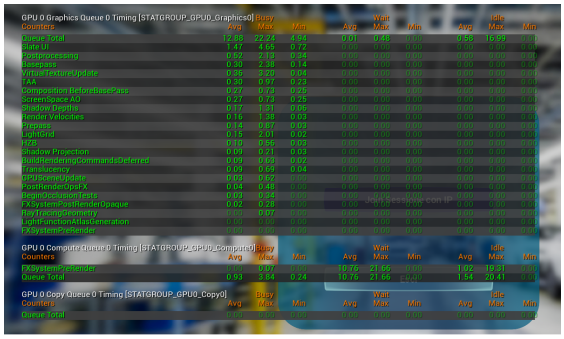
Figura 3.7: Demo di NVIDIA VR Capture and Replay

Durante lo sviluppo, il monitoraggio in tempo reale è stato condotto utilizzando comandi diagnostici nativi (*Stats comand*, come *stat unit* e *stat RHI*, vedi figura 3.8a), utili per ottenere un feedback immediato sull'occupazione dei *thread* della CPU e della GPU. Per un'analisi più approfondita dei colli di bottiglia è stato utilizzato *Unreal Insights*[49] (figura 3.8c). Questo potente strumento di profilazione consente di registrare e analizzare l'intera *pipeline* dell'esecuzione del motore grafico fotogramma per fotogramma, identificando con precisione millimetrica i picchi di carico derivanti da un numero eccessivo di *draw calls* o da calcoli fisici e logici eccessivi.

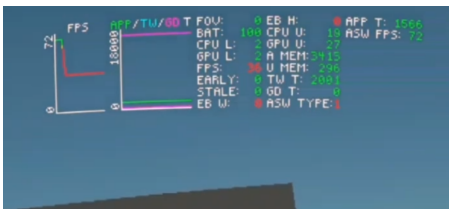
Parallelamente, per monitorare le prestazioni sul lato hardware del visore, si è utilizzato l'ecosistema di sviluppo ufficiale fornito da Meta, in particolare *OVR Metrics Tool*[50] (figura 3.8b) accessibile tramite il *Meta Quest Developer Hub* (MQDH)[51]. Questa applicazione di overlay ci ha permesso di raccogliere dati quantitativi fondamentali direttamente sul dispositivo indossato, monitorando

parametri critici come la percentuale di utilizzo dei core logici (CPU/GPU), le temperature operative, *frame drop*[2] e la latenza di decodifica del flusso video proveniente dalla rete.

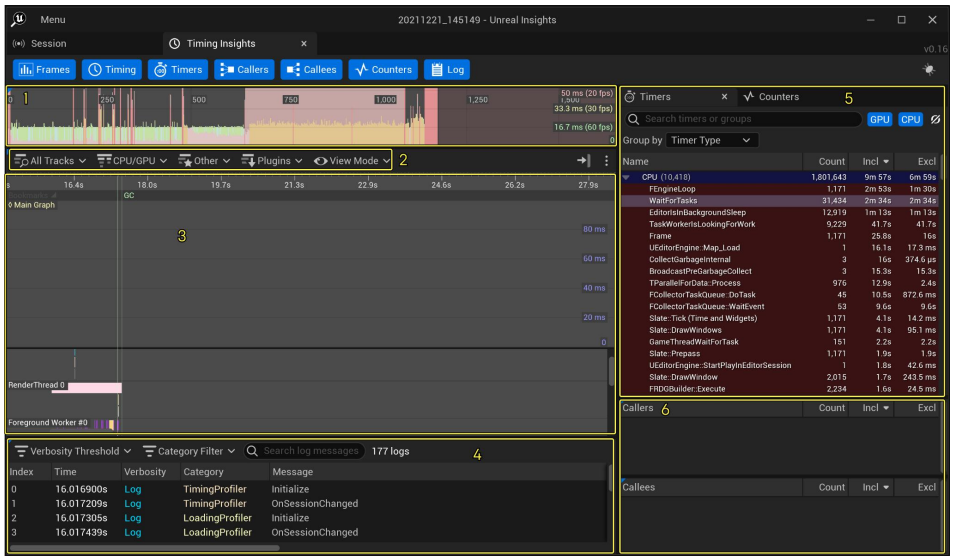
L'uso sinergico di questi strumenti diagnostici ha fornito il set di dati quantitativi necessario per convalidare scientificamente le prestazioni del simulatore. Come verrà ampiamente discusso nel prossimo capitolo, l'analisi incrociata di queste metriche è stata la chiave di volta ingegneristica che ha motivato l'abbandono dell'architettura *standalone* a favore dell'ecosistema PCVR più performante.



(a) Esempio di Stat comand: *stat gpu*



(b) Overlay di OVR Metrics Tool. *Fonte:[52]*



(c) Panoramica di Unreal Insight *Fonte:[49]*

Figura 3.8: Software utilizzati per la profilazione delle prestazioni

[2]Il *frame drop* è la perdita di fotogrammi dovuta a sovraccarico hardware, con conseguente latenza e perdita di fluidità.

Capitolo 4

Sviluppo e integrazione architetturale

Sulla base della configurazione hardware e software precedentemente definita, questo capitolo descrive le fasi di sviluppo e integrazione che hanno portato alla realizzazione del simulatore. Il testo illustra inizialmente lo studio di fattibilità che ha motivato la migrazione dall'architettura *standalone* a quella PCVR, per poi approfondire la creazione del *Digital Twin* attraverso l'ottimizzazione radicale dei modelli CAD industriali. Viene esaminata in dettaglio l'integrazione degli operatori virtuali, con particolare attenzione agli algoritmi di sincronizzazione tra il tracciamento inerziale e l'ambiente VR. Infine, il capitolo analizza l'architettura delle logiche di interazione modulare, descrivendo i sistemi di manipolazione collaborativa (*multi-grab*) e le infrastrutture di rete necessarie per la gestione delle sessioni multi-utente. La descrizione di queste implementazioni fornisce il quadro tecnologico necessario per la successiva fase di validazione ergonomica delle procedure industriali.

4.1 Studio di fattibilità: limiti dell'architettura *standalone* e migrazione a PCVR

La fase iniziale di progettazione del simulatore ha previsto uno studio di fattibilità per valutare l'esecuzione dell'applicativo in modalità *standalone*, ovvero delegando l'intero carico computazionale, grafico e logico al processore mobile integrato nel Meta Quest 3 e Meta Quest Pro. Questo primo approccio esplorativo è stato adottato con l'intento di esaminare empiricamente i potenziali vantaggi di un'architettura del genere, verificando la possibilità di massimizzare la libertà di movimento dell'operatore (grazie all'assenza di cavi di collegamento) e, parallelamente, di mantenere al minimo i costi dell'infrastruttura hardware necessaria per l'azienda.

Tuttavia, i test di telemetria condotti tramite l'impiego dei tool descritti nella sezione 3.7, hanno evidenziato criticità insormontabili legate ai limiti fisici del chipset mobile. Il solo rendering dei modelli CAD industriali (seppur precedentemente ottimizzati) si è rivelato sufficiente a saturare le capacità di calcolo del visore, provocando un drastico calo delle prestazioni. Questa limitazione hardware ha precluso fin da subito l'implementazione dei passaggi successivi.

In un contesto applicativo ergonomico, un'esperienza visiva non fluida è inaccettabile, in quanto invalida la naturalezza dei movimenti dell'utente e induce rapidamente chinetosi (motion sickness). Di seguito, nella figura 4.1, sono riportate le evidenze quantitative raccolte durante i test in modalità *standalone*.

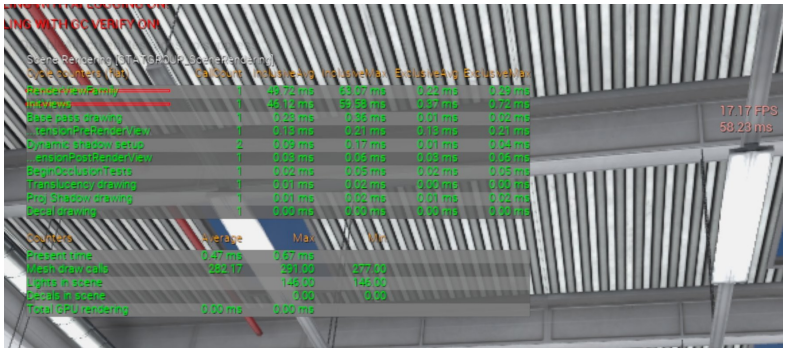


Figura 4.1: Analisi delle prestazioni in modalità *standalone*, con una sola delle mesh ottimizzate

Queste evidenze quantitative hanno reso necessaria una revisione architetturale, motivando la migrazione verso un ecosistema PCVR. In questa configurazione definitiva, il visore è stato declassato a terminale di input spaziale e display stereoscopico, mentre l'onere computazionale (rendering, gestione interazioni, ricezione dati da Xsens) è stato interamente trasferito a una workstation esterna ad alte prestazioni, comunicante con il Quest 3 e Quest Pro, tramite l'applicativo Meta Link.

Nonostante la transizione verso un hardware di calcolo nettamente superiore, si è scelto di mantenere un approccio ingegneristico orientato alla massimizzazione dell'efficienza. Un primo intervento fondamentale in tal senso ha riguardato l'aggiornamento dell'ambiente di sviluppo, migrando il progetto dalla versione 5.5 di Unreal Engine alla release 5.7. Questa decisione è stata dettata dai notevoli miglioramenti architetturali introdotti nell'ottimizzazione del rendering stereoscopico e nella gestione della memoria video, che hanno permesso di abbattere il frametime complessivo, come visibile nel confronto prestazionale nella figura 4.2.

(Simulazione effettuata con NVIDIA VR Capture & Replay)

Versione / Metrica (stat unit)	FPS Medi	Frame Time (Latenza)	Game Thread (CPU)	Draw Thread (Preparazione)	RHIT (Driver/API)	GPU Time	Draw Calls (Chiamate)	Primitives (Poligoni)
UE **5.5**	56 FPS	21.01 ms	19.36 ms	20.80 ms	13.73 ms	18.41 ms	742	2,439.6 K
UE **5.7**	64 FPS	18.41 ms	17.22 ms	18.40 ms	11.33 ms	16.07 ms	551	706.4 K
Delta miglioramenti	**+14%** *(Più fluido)*	**-2.60 ms** *(Più reattivo)*	**-2.14 ms** *(Cpu meno stressata)*	**-2.40 ms** *(Prep. veloce)*	**-2.40 ms** *(Comunicazione Veloce)*	**-2.34 ms** *(Rendering Veloce)*	**-26%** *(Meno Overhead)*	**-71%** *(Culling Ottimizzato)*

Figura 4.2: Confronto prestazioni tra UE 5.5 e 5.7 a parità di scena PCVR

Inoltre, non conoscendo le specifiche tecniche esatte della postazione PC che sarebbe stata fornita dall'azienda in fase di installazione finale, si è adottata una rigorosa strategia di scalabilità per assicurare che l'applicativo operi fluidamente anche su macchine industriali di fascia media. Si è scelto consapevolmente di disattivare le feature avanzate di Unreal Engine particolarmente esose di risorse, come [53] (illuminazione globale dinamica) e Nanite [54] (virtualizzazione geometrica). Al loro posto, si è optato per l'utilizzo di un'illuminazione statica pre-calcolata (Baked Lighting) e per un'aggressiva ottimizzazione poligonale degli asset 3D.

4.2 Setup sperimentale definitivo e ambiente di validazione

Al fine di garantire la stabilità computazionale della simulazione immersiva e la riproducibilità scientifica dei test prestazionali ed ergonomici, l'infrastruttura di sistema è stata riprogettata e dimensionata secondo i requisiti dell'architettura PCVR. Di seguito si dettaglia la configurazione hardware, software e di rete implementata come ambiente di test per la validazione del simulatore, schematizzata nella figura 4.3:

Configurazione hardware minima

- **Workstation di calcolo:** processore Intel Core i7 (o equivalente), 16 GB di memoria RAM e scheda grafica dedicata con almeno 8 GB di VRAM (es. NVIDIA RTX 2080 o superiore).

- **Sistema di visualizzazione e input:** visore Meta Quest 3 o Meta Quest Pro, impiegato esclusivamente come terminale di tracciamento spaziale a 6 gradi di libertà (6-DoF) e monitor immersivo.

- **Acquisizione cinematica:** sistema inerziale Xsens Awinda, composto da 17 sensori IMU e relative antenne riceventi, per il tracciamento full-body dell'operatore.

Configurazione software

- **Motore grafico:** Unreal Engine 5.7.3, utilizzato per l'esecuzione e il rendering dell'ambiente virtuale immersivo.

- **Elaborazione biomeccanica:** software Xsens MVN Analyze 2019, utilizzato per la lettura, la pulizia e la calibrazione dei dati grezzi provenienti dall'interfaccia Awinda.

- **Plugin di integrazione:** plugin Xsens Live Link (versione 5.4-5.7), modulo di interfaccia installato all'interno di Unreal Engine, necessario per ricevere e tradurre in tempo reale i dati cinematici provenienti da MVN Analyze.

- **Applicativo di analisi ergonomica:** software eseguibile di supporto, utilizzato dagli ingegneri di processo per estrapolare e analizzare gli indici biomeccanici durante la simulazione.

Infrastruttura di rete e comunicazione

- **Sincronizzazione multi-utente:** la sincronizzazione a bassissima latenza richiede che tutti i terminali hardware operino all'interno della medesima sottorete (LAN). Per scongiurare la perdita di pacchetti dati durante le sessioni di collaborazione virtuale, è stata predisposta un'infrastruttura di rete cablata tra le workstation, unitamente a una corretta configurazione dei permessi del *Firewall* di sistema per le specifiche porte di comunicazione software.

- **Streaming PCVR:** il collegamento ad alta larghezza di banda tra la workstation e il visore è stato stabilito tramite l'applicativo Meta Link, mediante connessione cablata USB 3.0, al fine di garantire la trasmissione del flusso video compresso con una latenza motion-to-photon impercettibile.

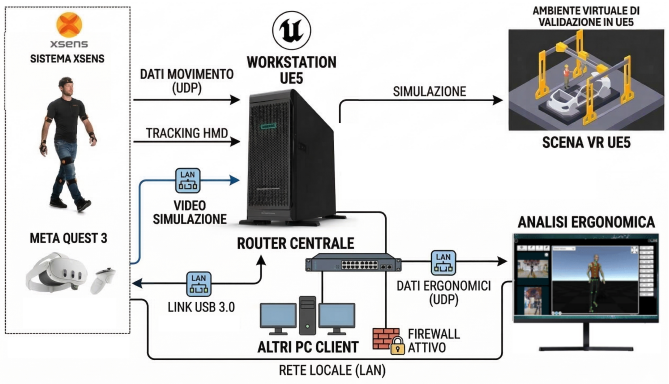
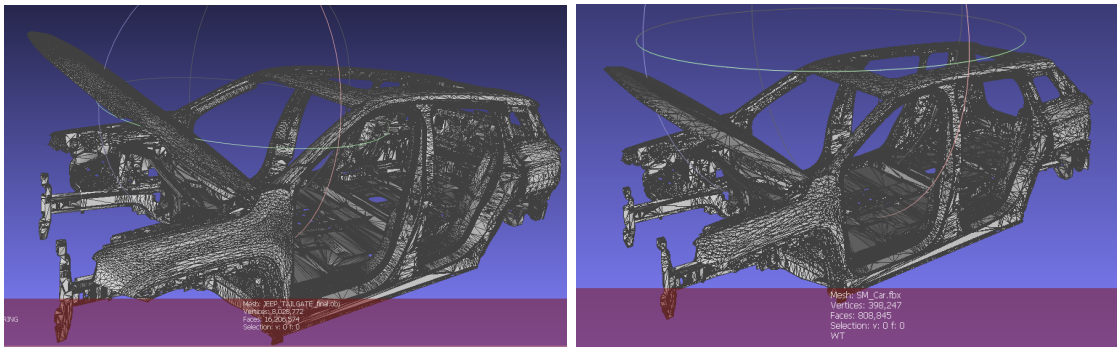


Figura 4.3: Schema dell'architettura definitiva del sistema

4.3 Sviluppo degli asset 3D e realizzazione del *Digital Twin*

Un intervento di ottimizzazione radicale è stato applicato direttamente ai modelli tridimensionali, affrontando il ben noto problema della conversione dei modelli CAD per il real-time rendering. In una fase preliminare, l'importazione e una prima semplificazione delle geometrie sono state gestite tramite il plugin Datasmith[55]. Sebbene questo strumento abbia fornito un'ottima base di partenza per la trasposizione dei formati industriali, l'ottimizzazione automatica faticava a trovare il giusto equilibrio: da un lato, forzando la riduzione poligonale, si rischiava la perdita di dettagli e componenti geometriche importanti; dall'altro, preservando la fedeltà visiva, i modelli mantenevano un peso eccessivo. Pertanto, i macchinari industriali e le scocche sono stati successivamente elaborati all'interno dell'ambiente di modellazione Blender. Qui, attraverso algoritmi manuali di decimazione geometrica, processi di retopology e la rimozione accurata di tutte le parti interne non visibili, si è proceduto ad abbattere il numero complessivo di vertici e triangoli, come si evince dalla tabella 4.1. Questa operazione è stata condotta prestando la massima attenzione a preservare la fedeltà volumetrica e la silhouette degli oggetti, parametri imprescindibili per garantire l'accuratezza dei calcoli ergonomici, vedi figura 4.4.

A questa operazione di ricostruzione geometrica è seguita una fase di shading e texturizzazione. Poiché i file CAD originali erano sprovvisti di informazioni cromatiche e fisiche, sono stati creati dei materiali personalizzati basandosi scrupolosamente sui riferimenti fotografici della linea (vedi figura 4.5). Questo lavoro ha permesso di riprodurre non solo i colori corretti, ma anche dettagli minuti come le texture delle superfici, il grado di riflettenza dei metalli e le etichette di segnaletica applicate sui macchinari, come si può osservare nella figura 4.6. Tale accuratezza



(a) Modello non ottimizzato
Numero di vertici: 8.028.772

(b) Modello dopo l'ottimizzazione
Numero di vertici: 398.247

Figura 4.4: Confronto del modello dell'auto originale e ottimizzato

Tabella 4.1: Confronto tra i modelli originali e le mesh ottimizzate

	Mesh originale		Mesh ottimizzata		
Modello	Vertici	Triangoli	Vertici	Triangoli	Riduzione %
Auto (per console)	13.318.024	26.726.495	507.878	1.037.466	96.1%
Auto (per portellone)	8.028.772	16.206.574	398.247	808.845	95.0%
Console posteriore	5.026.638	10.108.185	90.373	209.572	97.9%
Portellone	4.069.945	8.186.061	45.317	93.811	98.9%
Console anteriore	3.752.916	7.527.496	74.927	168.964	97.8%
Supporto	2.475.860	4.973.865	43.034	92.049	98.1%
Manipolatore	399.798	800.709	17.348	33.850	95.8%
Dima superiore	277.860	561.104	15.843	31.402	94.4%
Avvitatore	276.403	552.911	5.440	10.804	98.0%
Dima inferiore	174.406	356.182	10.318	20.276	94.3%

visiva è risultata indispensabile per garantire all'operatore un elevato senso di presenza e per facilitare l'identificazione immediata dei comandi e dei potenziali ostacoli durante la simulazione.

Successivamente, si è affrontato il principale collo di bottiglia prestazionale e cioè l'eccessivo numero di *draw calls*, ovvero le singole istruzioni di disegno inviate dal processore alla scheda video. I modelli CAD nativi, infatti, presentano tipicamente centinaia di componenti separati, ciascuno assegnato a un materiale distinto. Per risolvere questa criticità, si è adottata la tecnica del Texture Baking combinata al Material Merging, osservabile nella figura 4.7

Le molteplici geometrie che componevano un singolo macchinario sono state accorpate e le rispettive coordinate UV sono state riorganizzate in un unico spazio bidimensionale, generando un cosiddetto Texture Atlas. Attraverso il processo

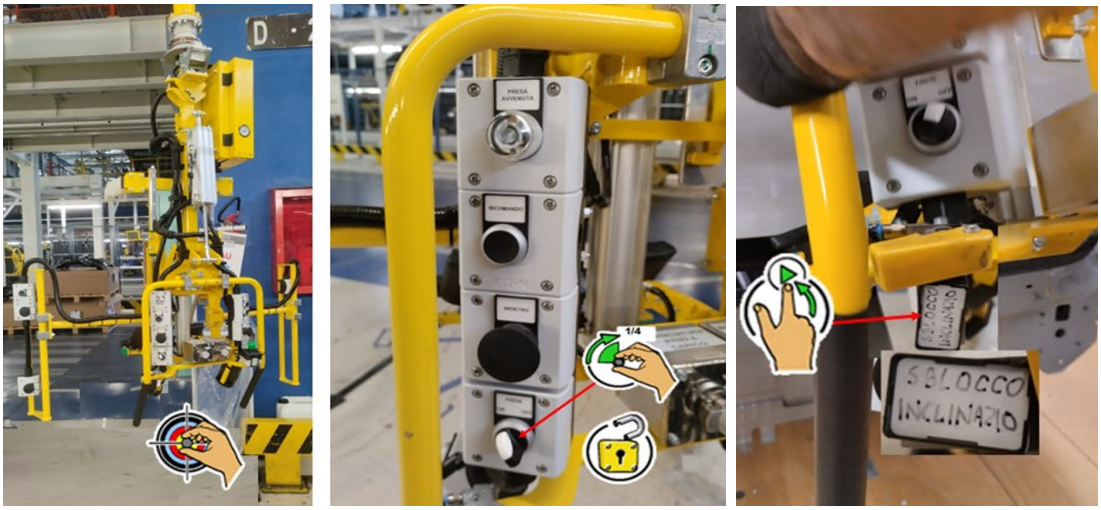


Figura 4.5: Alcune immagini di reference del manipolatore industriale

di Baking, le proprietà complesse dei materiali originali (come il colore base, la rugosità superficiale, il livello di metallicità e i dettagli di rilievo) sono state "cotte" e trasferite su un set ridotto di mappe texture 2D ad alta risoluzione.

Questa rigorosa pipeline di preparazione ha permesso di sostituire centinaia di materiali computazionalmente onerosi con un singolo materiale PBR (Physically Based Rendering) padre altamente ottimizzato. Il risultato è stato un abbattimento drastico delle *draw calls*, liberando preziose risorse di calcolo per la gestione della logica di rete e l'elaborazione del tracciamento inerziale in Unreal Engine.

Al fine di garantire la validità dell'analisi ergonomica, era imperativo che l'operatore percepisse gli esatti vincoli spaziali e gli ingombri dell'ambiente di fabbrica. Basandosi su un set di immagini fotografiche di reference fornite dall'azienda (figura 4.8), numerosi asset ambientali e di utensileria sono stati modellati da zero, integrandoli con alcuni modelli tridimensionali di elementi standard reperiti online. Questo processo di ricostruzione contestuale ha permesso di realizzare un *Digital Twin* della linea di assemblaggio fedele alla realtà, come si evince dalla figura 4.9. Tale fedeltà non è da intendersi unicamente sotto il profilo della resa visiva, ma soprattutto dal punto di vista dimensionale. Riprodurre esattamente la distanza tra un banco da lavoro e l'abitacolo veicolare è un requisito imprescindibile per valutare correttamente le rotazioni del busto e le estensioni degli arti dell'operatore durante le simulazioni.

Naturalmente, per mantenere la coerenza con la strategia di ottimizzazione descritta, anche questi elementi ambientali aggiuntivi sono stati modellati prestando attenzione al polycount e sottoposti alla medesima pipeline di Texture Baking per minimizzare l'impatto sulle *draw calls*.

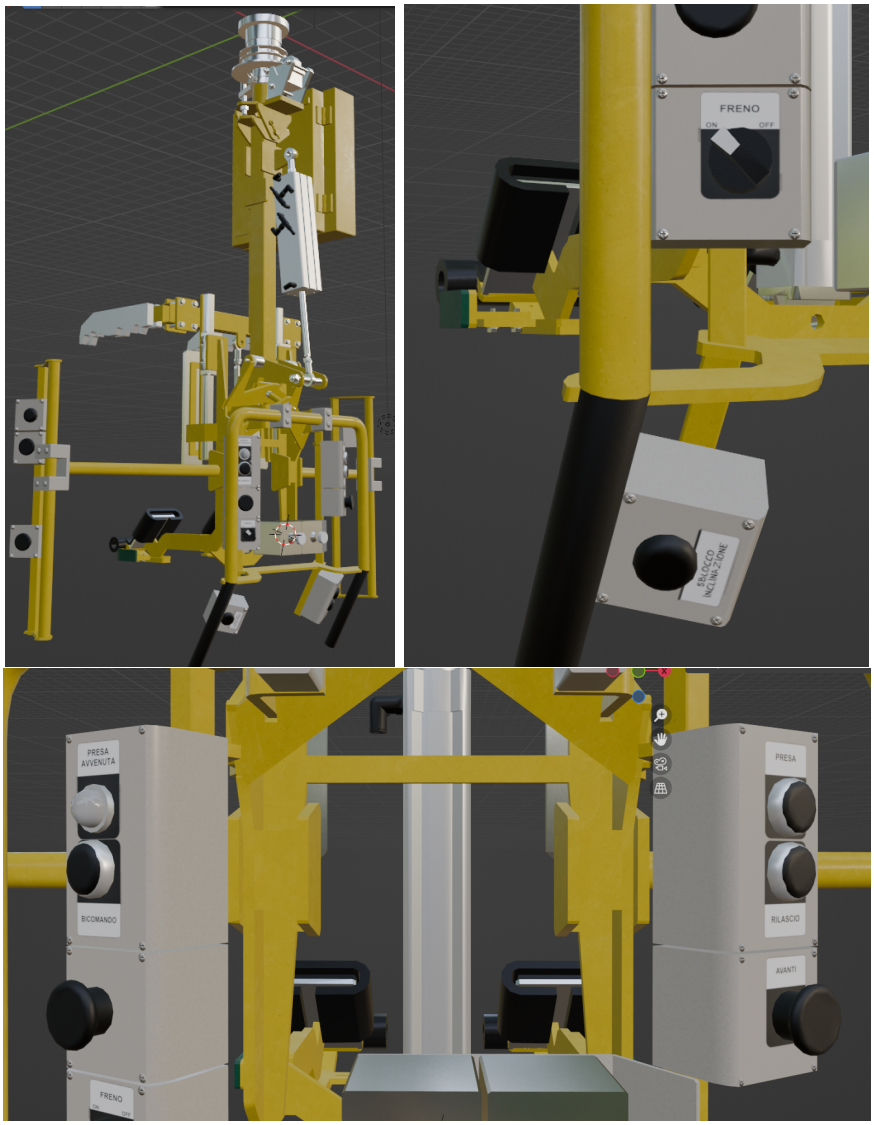
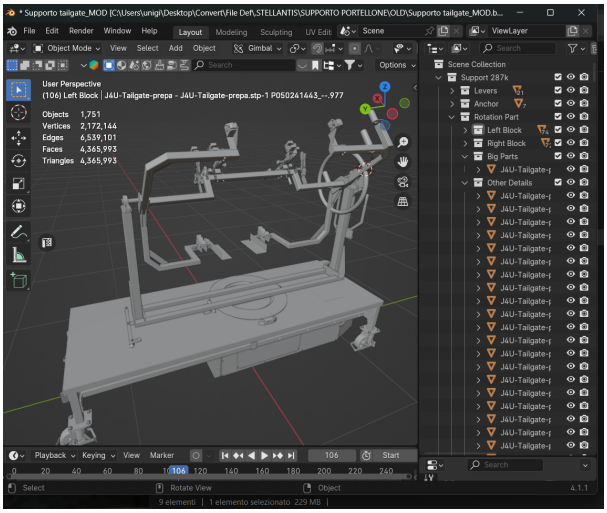


Figura 4.6: Riproduzione virtuale con materiali del manipolatore industriale

L'azione sinergica di questi fattori ha creato un ambiente virtuale estremamente leggero e reattivo, pronto per accogliere la complessità logica e biomeccanica dell'operatore e dell'ambiente.

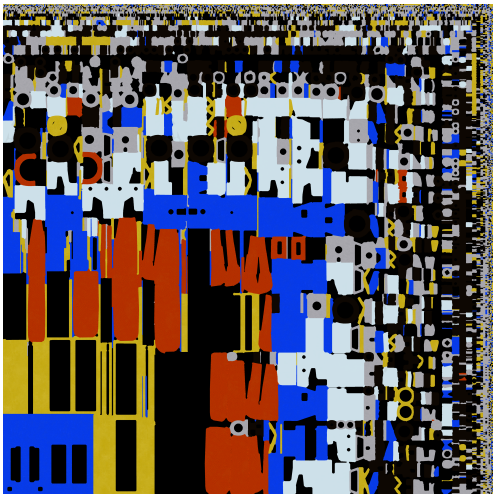
4.4 Integrazione dell'operatore virtuale

L'integrazione degli operatori all'interno dell'ambiente virtuale rappresenta il fulcro tecnologico del simulatore, poiché dalla fedeltà di questa rappresentazione dipende



(a) Mesh originale del supporto portelloni

Numero di elementi: 1751



(b) Esempio di Texture Backing per un unico materiale

Figura 4.7: Tecniche di ottimizzazione dei materiali

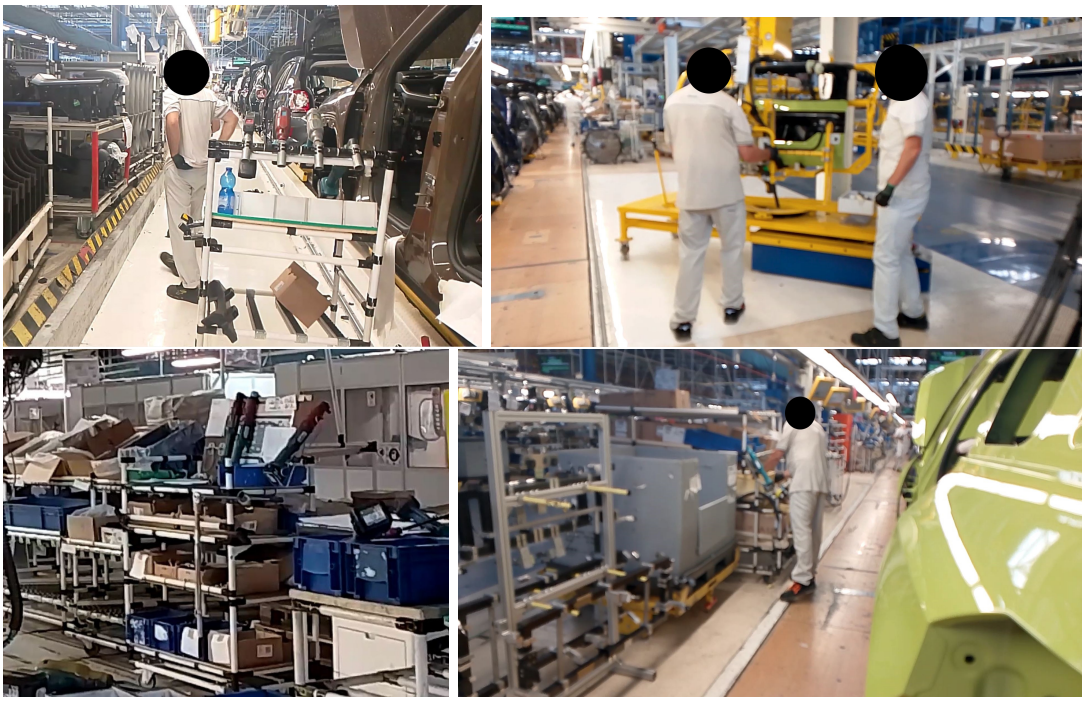


Figura 4.8: Alcune immagini di reference dell'ambiente industriale

Figura 4.9: Ambiente industriale ricreato in virtuale

la validità dell'analisi ergonomica e spaziale. Per garantire un'elevata accuratezza biomeccanica nelle sessioni collaborative, gli utenti vengono incarnati tramite avatar basati sull'architettura *MetaHuman* (figura 4.10). Al fine di facilitare l'identificazione reciproca durante le manovre congiunte, a ciascun avatar è stata assegnata una fisionomia distinta.

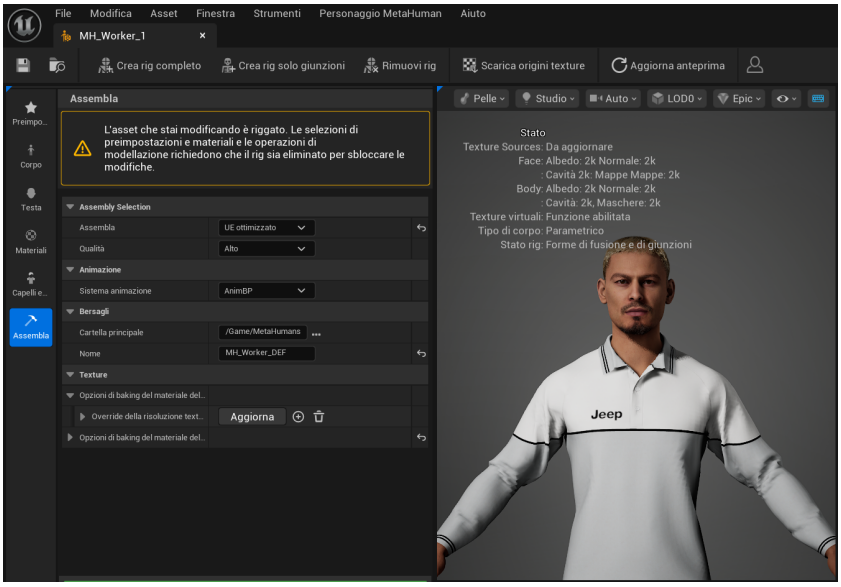


Figura 4.10: Creazione del MetaHuman

In linea con le strategie descritte precedentemente, relative all'ottimizzazione generale dell'applicativo, si è ritenuto opportuno bilanciare il livello di dettaglio visivo con le fisiologiche esigenze di fluidità della realtà virtuale. Pur non operando in condizioni di stringenti limitazioni hardware, si è scelto di non esportare i modelli alla massima risoluzione geometrica (*cinematic quality*). Il sistema è stato invece configurato per simulare gli avatar forzandone la renderizzazione al Livello di Dettaglio 1 (LOD 1). Questa accortezza progettuale ottimizza le risorse computazionali, garantendo un'eccellente stabilità prestazionale senza sacrificare il realismo e la fedeltà anatomica necessari ai fini dello studio.

Per supportare le dinamiche multi-utente tenendo conto dei vincoli hardware tipici di questi progetti (come la disponibilità di una singola tuta di *motion capture*), l'architettura di sistema è stata ingegnerizzata per gestire simultaneamente differenti tipologie di tracciamento posturale. Questa configurazione ibrida e asimmetrica permette la compresenza in scena di operatori animati con metodologie diverse:

- **Tracciamento full-body (Xsens):** i movimenti reali dell'operatore equipaggiato con la tuta inerziale Xsens vengono trasmessi in tempo reale al motore grafico attraverso il plugin ufficiale *Xsens Unreal Live Link [43]*. Questo

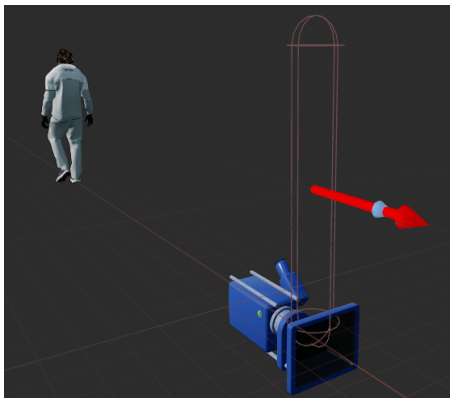
strumento gestisce in modo nativo e automatizzato il *retargeting* cinematico, mappando i dati dell'armatura sorgente direttamente sullo scheletro di destinazione del MetaHuman, fondamentale per le valutazioni posturali di precisione.

- **Tracciamento tramite cinematica inversa:** gli operatori sprovvisti di tuta inerziale vengono animati impiegando un sistema ibrido basato sull'IK. La postura della parte superiore del corpo del MetaHuman viene calcolata e interpolata dinamicamente dal motore grafico, basandosi esclusivamente sui tre punti di input spaziale forniti dall'hardware VR: il tracciamento della testa (tramite il visore) e quello delle mani (tramite i due *controller*). Parallelamente, per compensare la mancanza di tracciamento posizionale sugli arti inferiori, il sistema delega la gestione della locomozione a uno spazio di fusione procedurale (*Blend Space* [56]). Poiché la simulazione sfrutta appieno un approccio *room-scale*, in cui l'operatore si muove fisicamente all'interno dello spazio di tracciamento reale, il sistema calcola vettorialmente lo spostamento differenziale della telecamera VR rispetto al fotogramma precedente, per ottenere la velocità e la direzione di movimento. Questa reale traslazione fisica alimenta i parametri direzionali del *Blend Space*, il quale interpola in modo fluido e coerente le animazioni cicliche degli arti inferiori (come i passi in avanti o laterali), raccordandole senza soluzione di continuità con i movimenti del tronco generati dall'algoritmo IK.

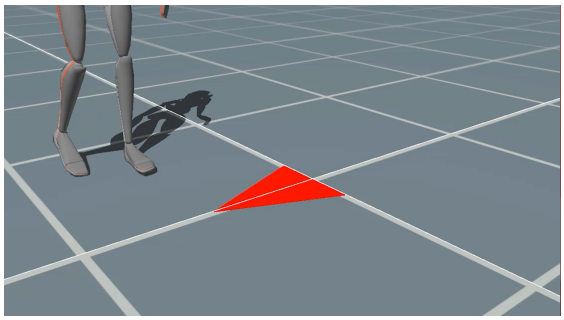
Locomozione ibrida e gestione dello spazio fisico. Indipendentemente dalla tipologia di tracciamento adottata (Xsens o IK), la simulazione è progettata per sfruttare nativamente un approccio *room-scale*. L'operatore è incoraggiato a muoversi fisicamente all'interno dell'area di tracciamento reale, poiché lo spostamento corporeo effettivo garantisce la massima fedeltà nella valutazione degli ingombri e delle posture. Tuttavia, per compensare le inevitabili discrepanze volumetriche tra i confini della stanza reale e l'estensione dell'impianto industriale virtuale, è stato integrato un sistema ausiliario di locomozione e rotazione artificiale. Tramite gli *input* analogici dei *controller* (*thumbstick*), l'operatore può traslare e riorientare istantaneamente il proprio sistema di coordinate virtuale (*VROrigin*). Questa flessibilità operativa assicura che l'utente possa navigare senza interruzioni per l'intera postazione di lavoro, riposizionandosi in modo coerente qualora raggiunga i limiti fisici dell'area di test.

4.4.1 Sincronizzazione spaziale e calibrazione ibrida MoCap-VR

Un aspetto fondamentale del simulatore risiede nel sistema di allineamento dinamico tra il flusso dati inerziale (*Xsens*) e il sistema di riferimento spaziale del visore VR. Poiché i due sistemi operano su origini e orientazioni differenti (figura 4.11), è stata sviluppata una logica di sincronizzazione all'interno del *Blueprint* dell'operatore (`CustomVRPawn_BP_Xsens`), articolata in tre fasi critiche: allineamento rotazionale, compensazione posizionale continua e calibrazione antropometrica.



(a) Unreal: disallineamento mesh con camera e capsula



(b) MVN Analyze: sistema di coordinate locale non sincronizzato con l'ambiente virtuale

Figura 4.11: Analisi dell'offset di posizione e rotazione tra i diversi sistemi di riferimento

Allineamento rotazionale e scaling iniziale

In fase di inizializzazione (*BeginPlay*), il sistema esegue una procedura di allineamento dello yaw. Attraverso il calcolo del delta angolare tra l'orientamento della camera VR (*HMD*) e il *socket* della testa della *mesh* MoCap, il sistema ruota l'intero scheletro affinché la direzione dello sguardo virtuale coincida con quella reale dell'utente.

Contestualmente, viene eseguita la funzione `CalibrateAndScaleAvatar`, la quale normalizza l'altezza dell'avatar MetaHuman. Il sistema rileva l'altezza reale del giocatore (tramite la coordinata Z della camera) e calcola un fattore di scala basato su l'altezza del MetaHuman originale (figura 4.12). Questo garantisce che le proporzioni dell'operatore virtuale siano coerenti con la statura fisica dell'utente, parametro fondamentale per la validità dei test ergonomici.

(a) Statura elevata **(b)** Bassa statura

Figura 4.12: Calibrazione in altezza dell'operatore

Algoritmo di compensazione posizionale (*tick*)

Per mantenere la perfetta coerenza spaziale durante il movimento, è stato implementato un algoritmo di compensazione che opera ad ogni *frame* (*tick*). La logica segue una catena di trasformazioni vettoriali invertite:

- **Gestione della capsula:** la capsula di collisione viene vincolata alle coordinate XY della camera, mentre la *VROrigin* subisce una traslazione inversa per stabilizzare la visuale del giocatore.
- **Ancoraggio mesh-HMD:** il sistema calcola costantemente l'offset tra il *socket* POV (posto sulla testa del MetaHuman) e la camera VR. Applicando questo scostamento negato tramite *AddWorldOffset*, la testa dell'avatar viene letteralmente "trascinata" per coincidere millimetricamente con la posizione dell'HMD.

Calibrazione antropometrica degli arti

Per correggere le discrepanze tra la lunghezza delle braccia dell'utente reale e quelle del modello 3D, è stata introdotta la funzione `CalibrateArmsXsens`. Tale procedura confronta le distanze euclidee tra i giunti (*clavicle*, *upper arm*, *lower arm*) e genera fattori di scala dinamici (figura 4.13). Questi parametri vengono

passati all'*Animation Blueprint* [57], che tramite nodi di trasformazione dei *bone*, adatta le lunghezze dei segmenti dell'avatar in tempo reale, assicurando che la mano virtuale raggiunga esattamente i punti di interazione toccati dall'operatore.

(a) Arto lungo **(b)** Arto corto

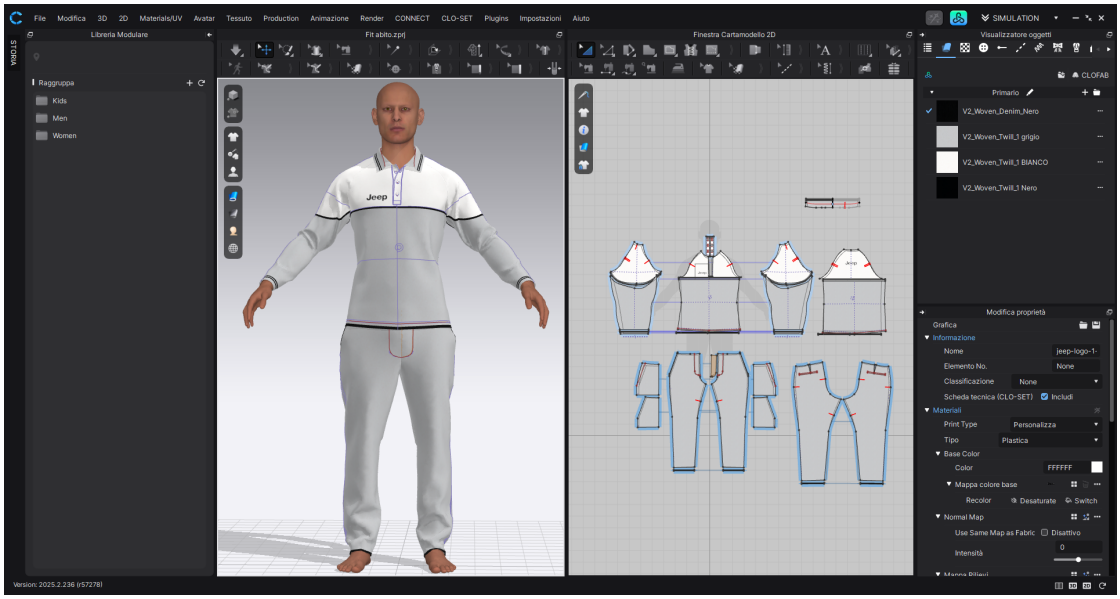
Figura 4.13: Calibrazione della lunghezza delle braccia dell'operatore

4.4.2 Abbigliamento virtuale in CLO3D

Al fine di massimizzare l'immersività e condurre valutazioni spaziali rigorose sui volumi occupati dall'abbigliamento da lavoro, le divise tecniche aziendali sono state riprodotte all'interno del software *CLO3D*, partendo da un'attenta analisi di diverse *reference* fotografiche reperite online.

Per garantire una corrispondenza geometrica impeccabile tra i capi d'abbigliamento e la morfologia degli avatar, la *mesh* del corpo del MetaHuman è stata preventivamente esportata da Unreal Engine e reimportata in *CLO3D* per fungere da manichino di riferimento. Sfruttando la simulazione fisica e il *tool* di vestibilità automatica offerti dal software, i tessuti sono stati adattati millimetricamente alle specifiche proporzioni anatomiche del modello. Questo approccio ha garantito un *fit* realistico e aderente, requisito fondamentale per prevenire fastidiose compenetrazioni poligonali durante i movimenti di collaudo. Si può osservare il risultato nella figura 4.14.

(a) Riferimenti trovati in rete



(b) Riproduzione virtuale in CLO3D

Figura 4.14: Design della divisa da lavoro con CLO3D

Tuttavia, l'impiego di *mesh* tessili ad alta densità in un ambiente in realtà virtuale rappresenta una potenziale criticità prestazionale. Pertanto, in fase di esportazione, non è stata generata la topologia alla massima risoluzione. Si è optato per un rigoroso compromesso tecnico: un'ottimizzazione del numero di vertici ha ridotto significativamente il carico computazionale, preservando al contempo una geometria sufficientemente dettagliata da consentire un'accurata stima visiva degli ingombri periferici all'interno dell'abitacolo.

Infine, per l'integrazione nel motore grafico, è stato adottato il flusso di lavoro avanzato basato sul formato USD. Le divise ottimizzate sono state importate in Unreal Engine e configurate direttamente come *Cloth Asset* [58]. Questa procedura ha permesso di vincolare automaticamente la geometria tessile all'armatura scheletrica dell'avatar. In questo modo, l'abbigliamento eredita fluidamente le deformazioni cinematiche del corpo dell'operatore senza richiedere la configurazione manuale dei pesi di deformazione (*skin weights*), garantendo un'elevata fedeltà visiva e interattiva.

4.4.3 Logica di collisione ibrida e gestione degli scollamenti

Una delle sfide ingegneristiche più complesse nell'integrazione degli operatori ha riguardato la gestione delle interazioni fisiche tra l'avatar e l'ambiente industriale. Per preservare l'assoluta fluidità dei sistemi di tracciamento (sia inerziale Xsens che IK), i corpi virtuali dei MetaHuman sono stati volutamente esclusi dai calcoli di collisione rigida bloccante. L'adozione di questo approccio a collisione morbida (*soft collision*), ha consentito di abilitare una validazione visiva: gli ingegneri esterni possono osservare direttamente a schermo le compenetrazioni poligonali tra l'operatore e le geometrie circostanti, mappando con precisione millimetrica le aree di potenziale urto senza che la simulazione interrompa bruscamente il movimento reale dell'utente.

Tuttavia, l'assenza di vincoli cinematici assoluti genera una criticità quando l'operatore afferra un componente provvisto di vincoli fisici propri (ad esempio, un elemento che urta contro la scocca dell'auto, come illustrato nella figura 4.15). In tali circostanze, la continuazione del movimento fisico delle mani crea un inevitabile disallineamento spaziale (*offset*) rispetto al punto di presa virtuale. Per gestire questa discrepanza senza ricorrere all'uso di esoscheletri o al blocco innaturale della simulazione, il sistema compensa la mancanza di resistenza meccanica introducendo aiuti sensoriali sostitutivi: un *feedback* visivo di tensione, che verrà illustrato dettagliatamente nella sezione 4.5.4, e un feedback acustico spazializzato.

Per quanto riguarda il feedback sonoro, invece di attivare un semplice allarme generico, l'applicativo calcola dinamicamente la posizione e l'entità dell'urto in tempo reale. Quando il motore fisico rileva una compenetrazione volumetrica tramite l'evento `OnComponentBeginOverlap` [59] generato dalla *mesh* del MetaHuman, si innesca una routine di calcolo iterativa. Un algoritmo ciclico (`ForEachLoop`) analizza un *array* predefinito di giunti scheletrici critici (es. testa, gomiti, gambe), interrogando per ciascuno la funzione `GetClosestPointOnCollision` [60]. Questa operazione matematica restituisce la coordinata esatta sulla superficie dell'ostacolo più vicina all'osso in esame, calcolandone la distanza vettoriale euclidea.

Una volta isolato il punto di massimo contatto, il sistema estrapola un valore scalare rappresentativo della profondità, ovvero della severità, della compenetrazione.

Figura 4.15: Rappresentazione del disallineamento tra la mano reale dell'operatore e la posizione di presa vincolata, evidenziato dal feedback visivo rosso

Tale variabile indipendente viene processata da una funzione matematica di ricalibrazione (`Map Range Clamped` [61]), la quale modula dinamicamente il parametro *Volume Multiplier* del segnale audio. A un'interferenza marginale corrisponde un suono lieve, mentre a una compenetrazione profonda fa seguito un impatto acustico di massima intensità.

Infine, per massimizzare il realismo immersivo, il nodo `Play Sound at Location` [62] istanzia l'effetto sonoro esattamente alle coordinate tridimensionali dell'impatto calcolato. Questo garantisce una propagazione audio direzionale e spazializzata, permettendo all'utente di percepire istintivamente da quale direzione, e su quale arto, si sia verificato l'urto, dandogli la possibilità di correggere la propria postura in modo naturale. Per prevenire fenomeni di affaticamento uditivo e saturazione del segnale durante le sovrapposizioni continue, la logica seleziona casualmente il campione sonoro da una libreria di effetti (simulanti dei "tonfi") e implementa un rigoroso sistema di ritardo temporale tra le singole attivazioni.

4.5 Architettura del software e logiche di interazione VR

Completata l'integrazione e la calibrazione spaziale dell'operatore all'interno del simulatore, lo sviluppo si è concentrato sulla progettazione delle logiche di interazione tra l'utente e l'ambiente produttivo virtuale.

Al fine di garantire un'elevata manutenibilità del codice e favorire la futura scalabilità del progetto, si è evitato l'incapsulamento di logiche ridondanti all'interno dei singoli attori della scena (*spaghetti code*), lo sviluppo è stato strutturato attorno alla creazione di componenti modulari e interamente riutilizzabili.

Questa specifica scelta progettuale ha consentito di disaccoppiare in modo netto le logiche di presa, il calcolo dei vincoli cinematici e i sistemi di feedback aptico, visivo e sonoro. Ciò permette di estendere le medesime funzionalità a nuovi strumenti o interfacce di fabbrica semplicemente "agganciando" i componenti preesistenti, senza la necessità di riscrivere l'infrastruttura di base.

Per comprendere appieno le sfide tecniche che hanno guidato tali scelte architetturali, è opportuno definire preliminarmente le procedure di assemblaggio che l'operatore è chiamato a validare. Pertanto, prima di analizzare nel dettaglio l'infrastruttura dei singoli componenti sviluppati, verranno delineati i principali scenari operativi implementati all'interno del *Digital Twin*.

4.5.1 Scenari operativi e requisiti di interazione

All'interno del simulatore sono stati implementati due scenari operativi principali, corrispondenti a due distinte stazioni della linea di montaggio. Le logiche software sono state progettate per rispondere alle specifiche sfide ergonomiche e di manipolazione imposte da queste due procedure:

- **Zona 1 - Stazione portellone:** l'obiettivo di questa postazione è l'assemblaggio del portellone posteriore sulla scocca del veicolo. La procedura inizia con una fase preparatoria, in cui gli operatori devono posizionare e agganciare delle specifiche dime di supporto negli incavi della scocca dell'auto, le quali fungeranno da guide spaziali per il successivo appoggio del componente. Per garantire la massima precisione, l'avvicinamento delle dime ai punti di ancoraggio è assistito da un sistema di guida aptica. Questo incrementa progressivamente l'intensità della vibrazione sui controller, culminando in un inequivocabile riscontro sonoro ("click") ad avvenuto allineamento. A questo punto, il collaudo virtuale si divide in due possibili varianti di movimentazione:
  - **Variante A (manipolazione manuale collaborativa):** due operatori fisici sorreggono il portellone, lo sganciano dai due fermi del supporto statico e, sfruttando il sistema di presa simultanea (*multi-grab*), lo trasportano manualmente verso il veicolo per tentare l'aggancio sui due punti di ancoraggio della scocca.
  - **Variante B (manipolazione tramite robot industriale):** l'operatore si avvale di un manipolatore industriale. L'interazione con il macchinario segue una rigorosa macchina a stati che replica le logiche di sicurezza

reali: l'utente deve disattivare il freno (*Manopola Freno OFF/ON*), abilitare il movimento in avanti (pulsante *Avanti*) e guidare il braccio verso il portellone. Anche in questo frangente, il corretto allineamento della pinza del manipolatore con il portellone è facilitato dal medesimo sistema di feedback aptico-sonoro, che segnala il raggiungimento della posizione corretta per la presa. L'aggancio e lo sgancio del portellone richiedono l'attivazione di una procedura di sicurezza bimanuale (pulsante *BiComando* unito al tasto di *Presa* o *Rilascio*), che gestisce la logica di controllo per la chiusura o l'apertura della pinza di presa. Durante tutte le fasi di traslazione, l'operatore può premere un pulsante dedicato per sbloccare l'inclinazione del carico (*sblocco inclinazione*). Conclusa l'operazione, il braccio deve essere riportato nella zona di parcheggio e bloccato.

In entrambe le varianti di movimentazione, la fase di accoppiamento finale del portellone alla scocca è ugualmente supportata dalla guida aptica per facilitare l'innesto preciso sui ganci di fissaggio. Parallelamente, l'intera manovra di avvicinamento è governata da un sistema di rilevamento delle collisioni: in caso di impatto o di posizionamento errato, l'algoritmo genera una retroazione atta a respingere fisicamente il carico o il manipolatore all'indietro, impedendo compenetrazioni irrealistiche e segnalando visivamente l'errore all'utente.

- **Zona 2 - Stazione console:** l'obiettivo di questa stazione è l'installazione di una console centrale all'interno dell'abitacolo. Al fine di valutare l'impatto posturale delle operazioni, l'ambiente virtuale permette di configurare la scocca dell'auto in tre differenti assetti di studio: a 400 mm da terra, a 700 mm, oppure ruotata con altimetria di circa 300 mm e rotazione del gancio girevole di 30°. Tali configurazioni sono selezionabili in tempo reale attraverso un pannello di controllo dedicato integrato nell'ambiente virtuale, permettendo una scelta rapida tra i diversi scenari di test ergonomico. La procedura prevede che l'operatore prelevi il primo componente e lo introduca nella vettura. Durante questa fase, il motore fisico esegue un calcolo continuo delle collisioni volumetriche (sweep) tra la console e l'auto, costringendo l'utente (eventualmente aiutato da un secondo operatore in *multi-grab*) a manovrare con estrema precisione. Durante l'avvicinamento ai punti di ancoraggio corretti, il sistema fornisce una guida aptica la cui intensità di vibrazione aumenta progressivamente, culminando in un riscontro sonoro ("click") ad avvenuto incastro. Ripetuta l'operazione per la seconda componente della console, il sistema genera le viti di bloccaggio: gli operatori impugnano a due mani un avvitatore elettrico per il serraggio finale, ricevendo un feedback acustico realistico dell'utensile e un feedback visivo della vite che penetra in sede.

4.5.2 Progettazione a componenti per l'interazione modulare

L'architettura software del simulatore si fonda sul paradigma a componenti nativo di Unreal Engine, impiegato per astrarre e generalizzare l'intera logica di manipolazione e interazione in realtà virtuale. Invece di sviluppare porzioni di codice monolitiche e ridondanti all'interno dei singoli attori della scena (es. avvitatore, portellone o manipolatore), si è scelto di ingegnerizzare un'infrastruttura di interazione modulare. Tale architettura permette di trasformare qualsiasi modello geometrico statico in un oggetto interattivo e valutabile ergonomicamente, semplicemente assegnandovi un set di componenti pre-programmati.

Il cuore di questo ecosistema è costituito da tre componenti personalizzati e interdipendenti:

- **`BPC_GrabSolver`:** è il modulo computazionale primario, delegato alla risoluzione della cinematica spaziale. Gestisce l'interazione di una o più mani calcolando la trasformazione risultante dell'oggetto. Le specifiche implementative di questo modulo verranno approfondite nella sezione 4.5.3.
- **`BPC_TensionMaterial`:** rappresenta il modulo di feedback visivo procedurale. Leggendo i dati di scostamento generati dal *GrabSolver*, modula dinamicamente i parametri dei materiali degli oggetti per comunicare visivamente l'errore di posizionamento delle mani. Il suo funzionamento verrà analizzato nel dettaglio nella sezione 4.5.4.
- **`AC_HapticGuide`:** gestisce il feedback aptico e sonoro, guidando intuitivamente l'operatore nel centraggio spaziale degli oggetti verso i rispettivi punti di ancoraggio. Le logiche algoritmiche di questo componente sono descritte nella sezione 4.5.4.

Grazie a questa tripartizione logica, lo sviluppo delle stazioni di lavoro (come la Zona 1 relativa al portellone e la Zona 2 relativa alla console) è risultato estremamente scalabile. È stato sufficiente assegnare questi tre componenti ai vari attori (come `BP_Portellone`, `BP_Dima`, `BP_FrontConsole`) per dotarli istantaneamente di logiche di manipolazione bimanuale, gestione della fisica e fornire all'utente un feedback completo (tatto, vista, udito) e immediato, abbattendo drasticamente i tempi di sviluppo e garantendo un comportamento coerente in tutto il simulatore.

4.5.3 Sistemi di presa multipla (*multi-grab*) e cinematica vettoriale

La complessità computazionale del *GrabSolver* si manifesta nel calcolo continuo della posizione e della rotazione virtuale dell'oggetto quando questo viene manipolato

simultaneamente da più mani.

Calcolo della posizione virtuale. La posizione target dell'oggetto viene determinata come media pesata delle posizioni nel sistema di riferimento globale dei controller attivi nella presa. Il sistema distingue tra controller principali e secondari: ai primi viene assegnato un peso $w_i = 1{,}0$, mentre ai secondari un peso ridotto $w_i = 0{,}2$. Questa scelta consente di attenuare l'influenza di prese accessorie rispetto all'asse di controllo principale.

La posizione virtuale risultante è definita come

$$\mathbf{p}_{\text{virt}} = \frac{\sum_{i=1}^{n} w_i \mathbf{p}_i}{\sum_{i=1}^{n} w_i} \tag{4.1}$$

dove $\mathbf{p}_i$ rappresenta la posizione globale del controller i-esimo.

Calcolo della rotazione con offset di presa. Per il calcolo dell'orientamento dell'oggetto si evita l'interpolazione diretta degli angoli di Eulero, che può introdurre singolarità cinematiche. Il sistema utilizza invece un approccio basato sull'interpolazione vettoriale degli assi direzionali.

Per ciascun controller i viene ricostruita la trasformazione compensata rispetto all'offset registrato al momento della presa:

$$T_{\text{comp},i} = \text{ComposeTransforms}\left(T_{\text{grab},i}, T_{\text{world},i}\right) \tag{4.2}$$

Da tale trasformazione vengono estratti il vettore *Forward* $\mathbf{f}_i$ e il vettore *Up* $\mathbf{u}_i$. I due insiemi di vettori vengono quindi combinati tramite media pesata e normalizzazione:

$$\mathbf{f}_{\text{avg}} = \frac{\sum_i w_i \mathbf{f}_i}{\left\|\sum_i w_i \mathbf{f}_i\right\|} \tag{4.3}$$

$$\mathbf{u}_{\text{avg}} = \frac{\sum_i w_i \mathbf{u}_i}{\left\|\sum_i w_i \mathbf{u}_i\right\|} \tag{4.4}$$

L'orientamento finale dell'oggetto è quindi ottenuto come

$$R = \text{MakeRotFromZX}\left(\mathbf{u}_{\text{avg}}, \mathbf{f}_{\text{avg}}\right) \tag{4.5}$$

Rotazione geometrica bimanuale. Nel caso di due controller attivi, la rotazione può essere calcolata direttamente dalla configurazione geometrica delle mani nello spazio.

L'asse Y viene definito come il vettore normalizzato che collega le due posizioni dei controller:

$$\mathbf{y} = \frac{\mathbf{p}_B - \mathbf{p}_A}{\|\mathbf{p}_B - \mathbf{p}_A\|} \tag{4.6}$$

L'asse Z viene invece ottenuto come somma dei vettori *Up* dei due controller:

$$\mathbf{z} = \mathbf{u}_A + \mathbf{u}_B \tag{4.7}$$

L'orientamento risultante è quindi

$$R = \text{MakeRotFromYZ}\,(\mathbf{y}, \mathbf{z}) \tag{4.8}$$

Le posizioni $\mathbf{p}_A$ e $\mathbf{p}_B$ sono determinate come centroidi pesati dei grabber appartenenti ai lati logici sinistro e destro. Questo calcolo è gestito dalla funzione ausiliaria `GetTwoHandVectors`.

Applicazione 1: oggetto rigido libero (es. `BP_Avvitatore`) Un oggetto non vincolato deve seguire le mani sia in posizione sia in orientamento. Al momento della presa viene registrato l'offset iniziale tra la trasformazione dell'oggetto e la trasformazione virtuale delle mani:

$$T_{\text{off}} = \text{MakeRelativeTransform}\,(T_{\text{actor}}, \text{MakeTransform}\,(\mathbf{p}_{\text{virt}}, R_{\text{global}})) \tag{4.9}$$

Ad ogni frame viene quindi calcolata la trasformazione target:

$$T_{\text{target}} = \text{ComposeTransforms}\,(T_{\text{off}}, \text{MakeTransform}\,(\mathbf{p}_{\text{virt}}, R_{\text{global}})) \tag{4.10}$$

La trasformazione viene applicata all'attore con rilevamento delle collisioni attivo (`Sweep = true`). In questo modo l'oggetto mantiene lo scostamento geometrico iniziale rispetto alle mani, evitando salti improvvisi al momento della presa.

Applicazione 2: manipolatore articolato vincolato (`BP_Manipulator`) Un manipolatore meccanico ancorato al soffitto possiede due gradi di libertà: rotazione orizzontale (*Yaw*) e inclinazione (*Pitch*) sul bone `Rot`. Il `GrabSolver` fornisce in questo caso delle *intenzioni di movimento* che vengono successivamente proiettate e vincolate ai gradi di libertà del sistema meccanico.

Lo *Yaw* viene calcolato dal vettore planare tra la base del manipolatore e la posizione virtuale della mano:

$$\psi = \text{MakeRotFromX}\left(\mathbf{p}_{\text{virt}} - \mathbf{p}_{\text{base}}\right).\text{Yaw} - \psi_0 \tag{4.11}$$

dove ψ_0 rappresenta lo yaw della mano registrato al momento della presa. Nel caso di interazione bimanuale viene invece utilizzata la direzione inter-manuale $\mathbf{p}_B - \mathbf{p}_A$ con un offset angolare iniziale.

La traslazione viene calcolata come differenza rispetto al punto di ancoraggio locale e decomposta sugli assi dell'attore.

Il calcolo del *Pitch* utilizza la funzione `CalculateVirtualRotation`. A partire dal vettore *Forward* medio compensato si ricava l'inclinazione del polso:

$$\theta_{\text{wrist}} = -\arctan\left(\frac{f_{\text{avg}} \cdot r_{\text{actor}}}{f_{\text{avg}} \cdot u_{\text{actor}}}\right) \tag{4.12}$$

L'angolo finale è quindi definito come

$$\theta_{\text{final}} = \text{NormalizeAxis}\left(\theta_{\text{wrist}} - \theta_{\text{wrist},0}\right) + \theta_{\text{bone},0}, \qquad \theta_{\text{final}} \in [-50^\circ,\, 0^\circ] \tag{4.13}$$

Quando l'inclinazione è sbloccata e sono presenti due o più mani, il risultato viene applicato come *rotazione orbitale* attorno all'asse Up del bone `Rot`. Si costruisce innanzitutto una rotazione attorno all'asse u_{bone} con angolo θ_{final}, che viene applicata al vettore raggio dell'attore rispetto alla posizione virtuale delle mani:

$$p_{\text{new}} = R_{\text{orb}}\left(p_{\text{actor}} - p_{\text{virt}}\right) + p_{\text{virt}} \tag{4.14}$$

La rotazione del mesh viene aggiornata simultaneamente componendo la rotazione corrente con la rotazione orbitale.

Sintesi In conclusione, il `GrabSolver` viene utilizzato secondo due modalità distinte:

- nel caso del `BP_Avvitatore`, come *sistema di posizionamento diretto*, in cui posizione e orientamento dell'oggetto seguono direttamente i controller dell'utente;
- nel caso del `BP_Manipulator`, come *sistema di input indiretto*, in cui i movimenti delle mani vengono interpretati come intenzioni di controllo (yaw, pitch e traslazione) e successivamente vincolati ai gradi di libertà meccanici del manipolatore.

4.5.4 Sistemi di feedback sensoriale e aiuti cognitivi

L'assenza di vincoli fisici reali nella manipolazione di oggetti virtuali richiede l'implementazione di robusti sistemi di *feedback* per restituire all'operatore la percezione degli ingombri, delle tolleranze e degli sforzi meccanici. A tale scopo, l'infrastruttura si avvale dei componenti `BPC_TensionMaterial` e `AC_HapticGuide`.

Feedback visivo procedurale (`BPC_TensionMaterial`) .

Questo componente ingegnerizza un sistema di tensione dinamica basato sui parametri del materiale, comunicando all'operatore non solo l'errore di posizionamento, ma anche il punto esatto e l'entità vettoriale della trazione (vedi figura 4.15). L'architettura si articola in tre fasi logiche:

- **Gestione collaborativa e *slot pooling*:** per supportare le dinamiche di *multi-grab* senza causare cali di fotogrammi critici per la VR, il componente evita la continua creazione e distruzione di istanze a *runtime*. Al contrario, genera un singolo materiale dinamico (`MaterialInstanceDynamic`[63]) in fase di inizializzazione, a cui è associato un *pool* di memoria pre-allocato di sei *slot* parametrici (permettendo l'interazione simultanea di un massimo di 6 mani). Sottoscrivendosi agli eventi delegati del `BPC_GrabSolver`, il sistema si limita ad assegnare uno *slot* libero a ogni nuova mano interattiva e a sovrascriverne le coordinate. Questa operazione ha un costo computazionale quasi nullo, garantendo la massima fluidità visiva.
- **Calcolo differenziale della tensione:** ad ogni fotogramma di calcolo (*tick*), il componente estrae le coordinate tridimensionali dell'ancoraggio virtuale e calcola la distanza euclidea tra questo punto ideale e la reale posizione spaziale della mano fisica (disallineamento).
- **Modulazione parametrica dello *shader*:** lo scostamento metrico rilevato viene normalizzato matematicamente in un valore scalare. Questo dato, unito alle coordinate del punto di ancoraggio, viene trasmesso in tempo reale ai parametri dello *shader* grafico. Il risultato è una sfumatura cromatica progressiva verso il rosso: più l'operatore "tira" l'oggetto bloccato, più l'alone si espande, fornendo un *feedback* visivo proporzionale allo sforzo applicato. Un risultato visivo è osservabile nella figura 4.16.

Guida aptica di allineamento (`AC_HapticGuide`) . Questo componente gestisce l'orientamento spaziale invisibile, fondamentale per replicare la sensazione di incastro meccanico. Si attiva automaticamente non appena l'oggetto manipolato varca una specifica soglia di prossimità rispetto al bersaglio. Tale area di tolleranza viene rilevata tramite l'intersezione dei volumi di controllo (*collision box*) dei due

Figura 4.16: *Feedback* visivo: l'intensità del colore rosso aumenta proporzionalmente all'errore di posizionamento

attori coinvolti, permettendo al sistema di calcolare il punto esatto di aggancio, come si osserva nella figura 4.17.

Durante l'avvicinamento, un algoritmo calcola la distanza vettoriale residua e modula l'intensità della vibrazione sui *controller* in modo inversamente proporzionale: la vibrazione aumenta guidando intuitivamente l'operatore verso il centro esatto. Raggiunta la tolleranza ottimale, la vibrazione cessa per lasciare spazio a un riscontro acustico (un "*click*") che conferma l'avvenuto allineamento e l'abilitazione al fissaggio. Viceversa, un allontanamento oltre la soglia comporta un decadimento progressivo del *feedback* aptico, comunicando la perdita del riferimento spaziale.

4.5.5 Logiche di sicurezza, animazione procedurale e rilevamento predittivo

La corretta simulazione del manipolatore industriale ha richiesto il superamento del tradizionale approccio di interazione degli applicativi ludici, per abbracciare logiche tipiche della robotica e dei controllori programmabili. Invece di affidare il movimento alla fisica simulata del motore grafico, il controllo del macchinario è stato ingegnerizzato attraverso tre sottosistemi deterministici: l'animazione procedurale, i blocchi logici di sicurezza e il rilevamento predittivo delle collisioni.

Macchina a Stati Finiti (FSM) e flusso operativo. Come illustrato nella figura 4.18, il comportamento complessivo del manipolatore è governato da una rigorosa macchina a stati finiti. L'operatore è vincolato a una sequenza operativa deterministica che prevede lo sblocco iniziale del freno, l'abilitazione al movimento e il raggiungimento delle specifiche tolleranze di prossimità necessarie per l'aggancio o lo sgancio del carico. Tale architettura logica fa sì che input specifici, come lo sblocco dell'inclinazione o l'attivazione della pinza, vengano elaborati dal sistema

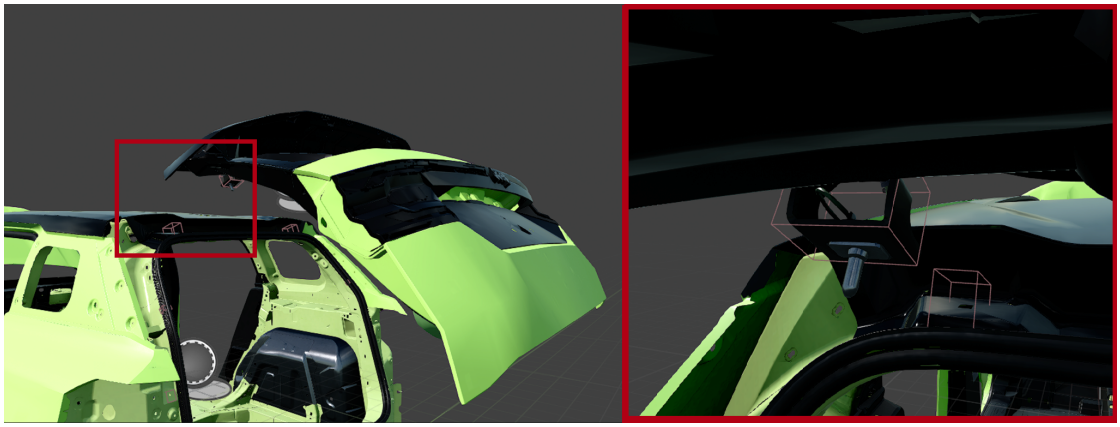


Figura 4.17: Collision box che identificano il punto di aggancio e il corretto allineamento degli oggetti

solo se il macchinario si trova nello stato corretto, replicando in tutto e per tutto la programmazione di un reale PLC di fabbrica.

Cinematica diretta e animazione procedurale (*Control Rig*)[64]. A differenza degli avatar umani, la cui struttura richiede complessi calcoli di cinematica inversa, il manipolatore costituisce una catena cinematica seriale semplice, priva di giunti ridondanti. Per tale motivo lo scheletro, visibile nella figura 4.19, è stato animato impiegando unicamente la cinematica diretta (*Forward Kinematics*).

L'infrastruttura di animazione (*Animation Blueprint* [57]) è strutturata su due livelli. Il primo applica le rotazioni assolute calcolate dalla logica di gioco ai singoli giunti tramite nodi *Modify Bone* (ad esempio per le manopole e i freni), imponendo rigorosi limiti angolari (*clamp*) programmati per prevenire le compenetrazioni interne. Il secondo livello demanda a un *Control Rig* parametrico l'interpolazione delle animazioni complesse, come l'apertura graduale della pinza e la compensazione automatica dell'inclinazione (*Pitch*) in risposta alla rotazione del corpo principale.

Attuazione bimanuale e interazione fisica (Bi-comando). Per replicare fedelmente le normative di sicurezza industriale, l'attuazione delle ganasce non è stata mappata sui pulsanti dei *controller* fisici, bensì richiede l'interazione spaziale con i pulsanti tridimensionali modellati sul manipolatore. Tali pulsanti sono configurati come ossa dello scheletro meccanico.

Durante l'interazione, un algoritmo monitora costantemente la traslazione sull'asse Z del pulsante nello spazio locale del giunto padre (Z_{local}). Quando la mano virtuale spinge il componente fisico, il valore scende sotto una soglia di attivazione

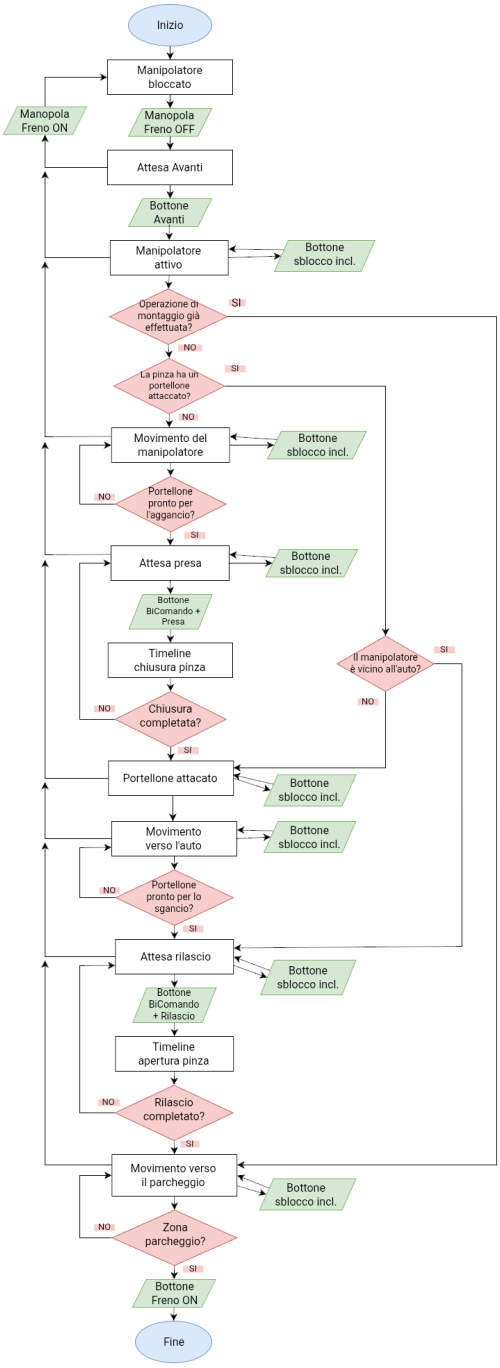


Figura 4.18: Diagramma di flusso della macchina a stati e delle logiche di interazione del manipolatore industriale.

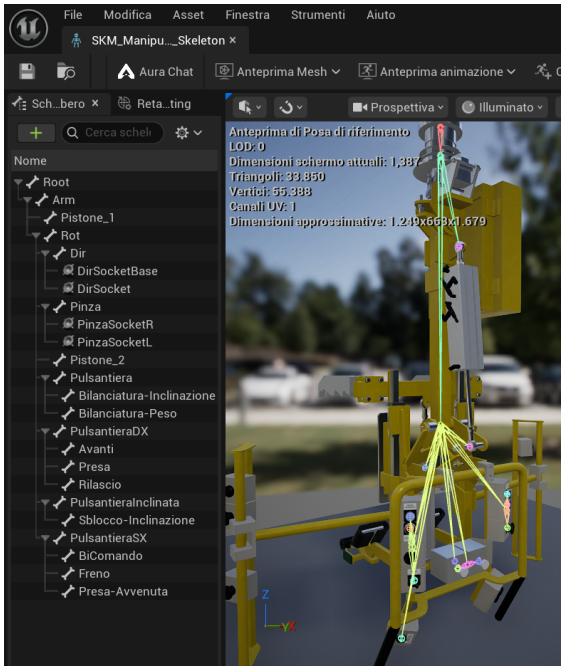


Figura 4.19: Scheletro creato per il manipolatore industriale

prefissata, innescando un meccanismo a ritenuta logica (*latching*) per evitare letture multiple.

L'apertura o la chiusura della pinza avviene esclusivamente se il sistema verifica l'asserzione booleana simultanea (operatore logico *AND*) tra lo stato del pulsante operativo (*Presa* o *Rilascio*) e un interruttore di sicurezza. Questa configurazione implementa il cosiddetto *Bi-comando*, costringendo l'operatore a utilizzare fisicamente entrambe le mani sulla pulsantiera della macchina e riproducendo il vincolo ergonomico presente nei sistemi reali.

Algoritmo predittivo di collisione (`CheckCollisionGhost`). Il controllo delle compenetrazioni tra il macchinario e la scocca dell'auto non si affida alla reazione fisica post-impatto (che genererebbe instabilità nel tracciamento VR), ma implementa un algoritmo di collisione predittiva. Prima di applicare la trasformazione spaziale richiesta dall'utente, il sistema simula il movimento istanziando volumi di controllo (*BoxTrace*) lungo le traiettorie dei giunti principali della pinza e dell'asse direzionale, come osservabile nella figura 4.20.

Se per la traslazione il tracciamento avviene in modo lineare, per i movimenti angolari il calcolo predittivo richiede una trasformazione degli spazi vettoriali. Il vettore posizione di ciascun giunto ($\vec{P}_{socket}$) viene de-ruotato nello spazio locale dell'attore tramite l'inversa della rotazione corrente ($R^{-1}_{current}$), moltiplicato per la

(a) Box Trace senza collisione **(b)** Identificazione della collisione

Figura 4.20: Gestione delle collisioni del manipolatore industriale

nuova matrice di rotazione candidata (R_{new}), e successivamente ri-proiettato nello spazio globale per ottenere la posizione futura predetta ($\vec{P}_{pred}$):

$$\vec{V}_{local} = R_{current}^{-1} \cdot (\vec{P}_{socket} - \vec{P}_{actor}) \tag{4.15}$$

$$\vec{P}_{pred} = \vec{P}_{actor} + (R_{new} \cdot \vec{V}_{local}) \tag{4.16}$$

Il sistema esegue quindi un controllo volumetrico dalla posizione di inizio fotogramma fino a $\vec{P}_{pred}$. Qualora l'operatore logico *OR* tra tutti i controlli restituisca un impatto positivo, la trasformazione richiesta viene negata.

In tal caso l'algoritmo incrementa un contatore di *stress* e attiva un meccanismo di *rollback* cinematico: viene interrogato un buffer temporale (`TransformHistory`) contenente gli stati sicuri degli ultimi fotogrammi e il braccio viene interpolato fluidamente verso l'ultima posizione valida. Questo produce un feedback visivo di "repulsione" elastica che segnala all'operatore l'errore di manovra.

4.6 Architettura di rete e gestione delle sessioni collaborative

Affinché il simulatore potesse fungere da vero e proprio ambiente di validazione industriale collaborativo, è stata sviluppata una solida infrastruttura di rete basata sul paradigma *client–server* (vedi figura 4.21). Per garantire la massima sicurezza dei dati aziendali e minimizzare la latenza, il sistema adotta una topologia *Listen*

Server su rete locale (LAN), appoggiandosi alle librerie `OnlineSubsystemUtils`[65] di Unreal Engine.

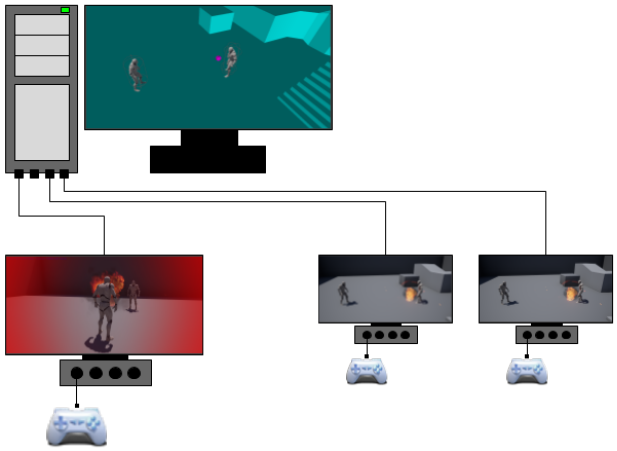


Figura 4.21: Esempio di gameplay client-server *Fonte:[28]*

4.6.1 Infrastruttura di hosting e Interfaccia Utente (UI)

L'accesso all'ambiente virtuale è regolato da un'interfaccia utente tridimensionale integrata nel livello di partenza, osservabile nella figura 4.22. Il sistema si basa su un'architettura a stati (`WidgetSwitcher`) orchestrata dal modulo `WB_ServerMangerMaster`, che consente all'utente di navigare fluidamente tra i menu di creazione o ricerca delle sessioni.

(a) Menu principale

(b) Scelta simulazione e tracciamento

(c) Unirsi a una sessione

Figura 4.22: Menu per la gestione della sessione

L'inizializzazione di una nuova sessione invoca chiamate asincrone al sottosistema di rete, componendo dinamicamente una stringa di opzioni URL (ad esempio `listen?MoCap=1?SimTag=Z1`). Tali parametri indicano al motore grafico di istanziare il livello come *Listen Server* e trasmettono ai *client* connessi le direttive sulla tipologia di simulazione e sull'eventuale utilizzo di tute inerziali per il *motion capture*.

Per quanto concerne l'accesso a una sessione esistente (*Join*), il sistema implementa una connessione diretta tramite indirizzo IP (`WB_JoinSessionNEW`). A tale scopo è stata ingegnerizzata una tastiera numerica virtuale personalizzata, ottimizzata per l'interazione immersiva.

Utilizzando i propri *controller*, l'operatore compone l'indirizzo di rete dell'host locale (es. 192.168.1.x); gli input aggiornano in tempo reale una variabile stringa visualizzata all'interno di un'apposita casella di testo.

Alla pressione del pulsante di conferma, la logica di rete innesca un comando di *ClientTravel* (tramite il nodo `OpenLevel` [66]), indirizzando il *client* all'IP digitato. Questo approccio progettuale non solo assicura una connessione rapida e mirata, azzerando l'*overhead* prestazionale derivante dal *polling* continuo, ma risolve elegantemente una nota criticità del sottosistema di rete di Unreal Engine legata alla scoperta automatica in LAN (*broadcast discovery*). La connessione diretta, infatti, previene i conflitti di *routing* causati dalla presenza di interfacce di rete multiple (fisiche o virtuali) sulla macchina *client*, le quali tendono a intercettare i pacchetti di ricerca impedendo il corretto rilevamento della sessione attiva nella sottorete desiderata. In tal modo, si garantisce l'accesso immediato alla simulazione senza costringere l'operatore a disabilitare manualmente le schede di rete secondarie a livello di sistema operativo.

4.6.2 Assegnazione dinamica dei ruoli (*GameMode*) [67]

La logica di smistamento dei giocatori è centralizzata nel `CustomVRPawn_GameMode`. Al momento della connessione di un nuovo *PlayerController*, il server esegue un controllo hardware (`IsHeadMountedDisplayEnabled`) e analizza i parametri URL della sessione. In base a queste discriminanti, il sistema interroga liste precompilate (*arrays*) per assegnare la classe corretta:

- Se la sessione prevede il tracciamento inerziale (`MoCap=1`), viene istanziato l'avatar avanzato `CustomVRPawn_BP_Xsens`.
- Se si tratta di un operatore VR standard, il sistema estrae un avatar dalla lista `PawnSimpleAssignmentList`.

Per evitare sovrapposizioni visive in scenari collaborativi, l'estrazione dalla lista avviene tramite un algoritmo *round-robin* (incremento dell'indice con operatore modulo), garantendo che ogni operatore riceva una fisionomia MetaHuman univoca.

Infine, un *override* della funzione `ChoosePlayerStart` [68] legge il *Tag* della simulazione per far comparire l'utente esattamente nella postazione di lavoro richiesta.

4.6.3 Replicazione e sincronizzazione degli avatar VR

La coerenza visiva e spaziale degli operatori all'interno dell'ambiente collaborativo è garantita da un sistema di replicazione di rete altamente ottimizzato. Per gestire le differenti periferiche di tracciamento (visore VR standard e tuta inerziale), l'architettura dei *Pawn* è stata strutturata secondo un paradigma orientato agli oggetti basato sull'ereditarietà. Una classe genitore (`CustomVRPawn_BP`) definisce le variabili replicate condivise e le logiche di rete fondamentali, mentre una classe figlia (`CustomVRPawn_BP_Xsens`) ne esegue l'*override* per adattare l'interpolazione ai dati del *motion capture*.

Ottimizzazione della banda e sistema a timer. Per prevenire la saturazione della banda passante, l'invio delle trasformate spaziali dal *client* locale al server non avviene a ogni *frame* di calcolo (*tick*), bensì è regolato da un *timer* asincrono a frequenza configurabile (impostato di *default* a 30 Hz). Ad ogni ciclo temporale, il sistema raccoglie i dati di tracciamento, quantizza i vettori di posizione per ridurne il peso in byte (`FVector_NetQuantize`) e li trasmette al server tramite una chiamata RPC *Unreliable* (`VR Transforms to Server`). L'utilizzo di pacchetti non garantiti è bilanciato dall'elevata frequenza di invio. Infatti la perdita di un singolo pacchetto viene istantaneamente mascherata dal pacchetto successivo e dall'algoritmo di interpolazione eseguito sui *client* remoti.

Replicazione dell'avatar VR standard (classe genitore). Nel caso dell'operatore dotato unicamente di visore e *controller*, l'evento di replicazione raccoglie tre sorgenti di tracciamento primarie: la posizione e rotazione della testa (*Camera*) e le trasformate dei due *controller* manuali. A questi si aggiungono metadati cruciali quali la velocità di locomozione e il parametro di *Spine Blend*, calcolato dinamicamente in base alla distanza planare tra la testa e il centro della capsula di collisione, utile per piegare proceduralmente la schiena dell'avatar.

Poiché la cinematica delle dita richiede aggiornamenti legati alla pressione analogica dei grilletti, i valori di *Grip*, *Index* e *Thumb* vengono inviati al server tramite RPC *Unreliable* dedicati. Quando i dati giungono ai *client* remoti, una funzione eseguita nel *tick* (`Interpolate Tracking Sources`) non impone le trasformate in modo istantaneo, ma utilizza funzioni matematiche di smussamento (`TInterpTo` con velocità 15) per muovere in modo fluido la testa e le mani dell'avatar, ricostruendo l'azione dell'utente originario senza i tipici scatti (*jittering*) derivanti dalla latenza di rete.

Replicazione dell'avatar motion capture (classe figlia). L'assegnazione della classe figlia `CustomVRPawn_BP_Xsens` disabilita l'animazione della locomozione basata sul visore e altera radicalmente il pacchetto di dati trasmesso in rete. Poiché in questo scenario l'intera postura del corpo (incluse le braccia e le mani) è dettata dai dati inerziali del *motion capture* gestiti tramite *LiveLink*, risulta ridondante replicare spazialmente i *controller* VR.

Di conseguenza, il pacchetto di rete viene snellito: la classe figlia invia un'unica posizione (quella della *Camera*) e tre rotazioni (la *Camera*, la rotazione globale del corpo fornita dal MoCap e l'orientamento della capsula). Sui *client* remoti, la funzione di ricostruzione viene sovrascritta (*override* di `Interpolate Camera`) per interpolare esclusivamente la traslazione del corpo virtuale sotto la posizione della testa, mentre l'allineamento degli arti viene demandato direttamente all' *Animation Blueprint* dedicato (`SK_Mannequin_AB_Xsens`), il quale riceve il flusso dati inerziale e genera l'animazione *full-body* in tempo reale.

4.6.4 Sincronizzazione spaziale e gestione dinamica del *multi-grab*

Il fulcro tecnologico dell'esperienza collaborativa risiede nella sincronizzazione degli oggetti manipolati in rete (come l'avvitatore o la *console* centrale). Per garantire la massima fluidità e prevenire conflitti di calcolo, l'architettura di rete adotta una gestione dinamica dell'*Authority* (l'autorità di calcolo) che varia in base al numero di utenti coinvolti nell'interazione, distinguendo tra manipolazione singola e collaborativa (*multi-grab*).

Manipolazione singola (predizione lato client). Quando un singolo utente afferra un oggetto, il sistema privilegia l'assenza di latenza percepita in realtà virtuale adottando un approccio di predizione lato *client*. Il sistema esegue una chiamata RPC di tipo `Server_SetOwner`, trasferendo temporaneamente la proprietà di rete dell'attore al *client* locale. Da questo momento, il calcolo della cinematica viene eseguito localmente dal componente `BPC_GrabSolver`.

Per preservare l'integrità della banda passante, la complessa trasformazione risultante non viene replicata tramite eventi garantiti, bensì inviata al server a ogni *tick* di calcolo tramite una RPC *unreliable* (`SRV_UpdateTransform`). Il server si limita ad aggiornare la variabile replicata `NetTransform`, la quale viene successivamente propagata a tutti gli altri *client* in ascolto. Per mascherare l'inevitabile perdita di pacchetti e la discontinuità temporale della trasmissione, i *client* remoti non applicano la trasformazione spaziale istantaneamente, ma utilizzano algoritmi matematici di smussamento (tramite i nodi `VInterpTo` e `RInterpTo`) per interpolare in modo fluido e continuo la posizione e la rotazione dell'oggetto verso il nuovo *target* di rete.

Manipolazione collaborativa *multi-grab* (calcolo lato server). Le dinamiche mutano radicalmente quando più operatori afferrano simultaneamente il medesimo oggetto. In scenari di *multi-grab*, per evitare che le predizioni dei singoli *client* entrino in conflitto generando scatti o desincronizzazioni, l'architettura passa a un approccio basato sulla stretta autorità del server.

In questo caso, il server mantiene (o riacquisisce) l'autorità sull'oggetto. I *controller* remoti degli utenti coinvolti vengono registrati in un *array*, ma è esclusivamente il server a processare la logica del `BPC_GrabSolver`. Il server ricalcola in tempo reale l'offset spaziale applicando una media pesata delle posizioni lette dai vari *client*. Nello specifico, l'algoritmo attribuisce un peso predominante pari a 1,0 al movimento generato dalla presa principale, mentre le interazioni secondarie (le mani aggiuntive degli altri *client*) influiscono sul calcolo vettoriale con un peso ridotto pari a 0,2. Una volta estrapolata la posizione virtuale risultante, il server impone il movimento propagando le trasformazioni ai *client* in modo nativo e centralizzato, sfruttando il sistema di *Replicate Movement* [69] offerto dal motore grafico, senza necessitare di ulteriori interpolazioni manuali sui singoli *client*.

4.6.5 Strumenti di analisi esterna: il *VR Spectator*

Poiché lo scopo ultimo dell'applicativo è la valutazione ergonomica da parte di personale ingegneristico, è stato previsto un ruolo specifico per l'osservazione esterna. Il *GameMode*, rilevando l'assenza di un visore VR sul PC connesso (`IsHeadMountedDisplayEnabled == false`), bypassa l'assegnazione degli avatar MetaHuman istanziando un `VRSpectatorPawn`.

Questa entità invisibile è controllabile tramite classiche periferiche *desktop* (tastiera e mouse) e fornisce all'ingegnere di processo controlli di volo libero (*Fly Mode*) e manipolazione del campo visivo (FOV). L'osservatore può così navigare all'interno del *Digital Twin*, esaminando da qualsiasi angolazione le tolleranze di montaggio e le posture assunte dagli operatori durante il collaudo virtuale, senza causare collisioni o interferire con lo svolgimento della simulazione.

In aggiunta alla navigazione spaziale autonoma, il sistema implementa una meccanica di commutazione visiva dinamica. Tramite un apposito comando da tastiera (barra spaziatrice), l'osservatore può scorrere ciclicamente e ancorare il proprio punto di vista alla telecamera in prima persona di ciascun operatore attivo nella simulazione. Questa transizione prospettica risulta di fondamentale importanza ai fini della validazione ergonomica: consente infatti all'ingegnere di abbandonare temporaneamente la visuale panoramica per immedesimarsi nell'operatore, verificandone l'effettiva linea di vista in tempo reale e valutando con precisione le eventuali occlusioni visive generate dalle geometrie del veicolo o dagli ingombri della strumentazione.

Capitolo 5

Validazione funzionale e casi di studio aziendali

Dopo aver analizzato l'architettura tecnica e le logiche di integrazione, questo capitolo presenta la fase di validazione funzionale del simulatore, condotta attraverso dei test sul campo presso gli spazi aziendali. L'obiettivo è dimostrare l'efficacia del sistema come strumento di supporto decisionale per l'ingegneria di processo e l'ergonomia preventiva. Attraverso l'analisi di due casi di studio specifici (l'assemblaggio del portellone posteriore e l'installazione della console centrale), vengono esaminate le prestazioni del sistema in scenari di manipolazione collaborativa e in diverse configurazioni di linea. Il capitolo approfondisce l'utilità dei feedback sensoriali nel guidare l'operatore e conclude con una discussione sui risultati ottenuti.

5.1 Metodologia di collaudo industriale e validazione sul campo

Conclusa la fase di sviluppo, il simulatore è stato sottoposto a una campagna di validazione qualitativa e funzionale direttamente presso gli spazi dell'azienda. L'obiettivo di questa fase conclusiva risiedeva nella dimostrazione pratica dell'efficacia dell'applicativo come strumento di supporto decisionale per gli ingegneri di processo e gli ergonomi.

I test sul campo sono stati strutturati per replicare fedelmente reali procedure di assemblaggio e collaudo. L'esecuzione materiale delle prove è stata condotta in prima persona, indossando il sistema di tracciamento inerziale Xsens e il visore Meta Quest Pro, si è proceduto a interagire direttamente con la replica virtuale della stazione di lavoro. Il compito assegnato consisteva nell'interagire con specifici

componenti del veicolo, come illustrato nella sezione 4.5.1, al fine di valutarne l'accessibilità, verificare le collisioni, esaminare le interazioni e analizzare gli ingombri spaziali e trarre in real time parametri per le valutazioni ergonomiche.

Durante le sessioni di prova, il team di Ergolab ha potuto monitorare l'operato dell'utente in tempo reale attraverso la proiezione delle diverse visuali del simulatore. Questa modalità di osservazione multi-prospettica ha permesso agli esperti di valutare immediatamente le criticità posturali, le difficoltà di raggiungimento della componentistica e le eventuali interferenze tra il corpo dell'avatar e la scocca del veicolo, confermando la validità del sistema come ambiente di test preventivo.

Per illustrare nel dettaglio l'esito di questa validazione sul campo, nei paragrafi successivi vengono presentati i due casi di studio specifici, documentati attraverso il materiale multimediale, raccolto durante le prove aziendali.

5.2 Caso studio 1: assemblaggio del portellone (manipolazione manuale e robotica)

Il primo collaudo funzionale ha riguardato la procedura di assemblaggio del portellone posteriore alla scocca del veicolo. L'obiettivo del test era verificare le condizioni di lavoro per il/gli operatore/i analizzando due differenti approcci operativi, acquisendo dalla scena virtuale parametri necessari per la stima a progetto degli indici di sollevamento per il rischio da movimentazione manuale dei carichi, nonché informazioni posturali. Questo è stato svolto, analizzando due differenti approcci operativi: la manipolazione manuale collaborativa (eseguita da due operatori) e la movimentazione assistita tramite manipolatore robotico industriale. In entrambi gli scenari, la procedura prevedeva un'azione preparatoria comune di posizionamento e aggancio delle dime di supporto all'interno dell'incavo della scocca.

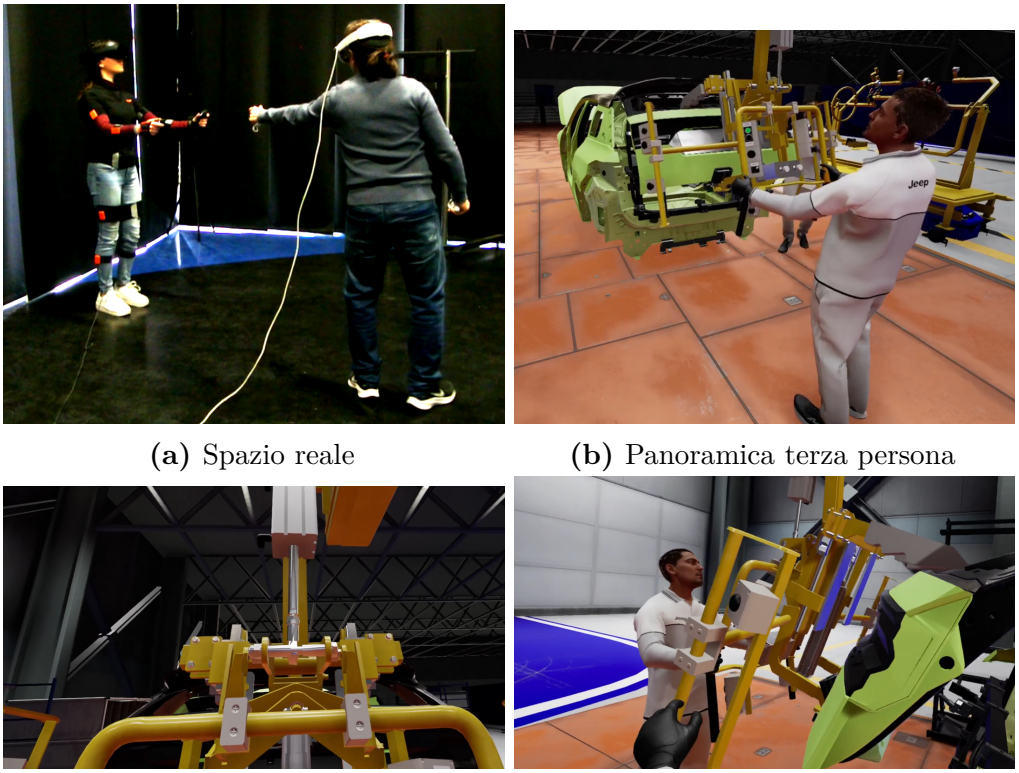
Durante l'esecuzione della variante manuale, l'azione in *multi-grab* ha permesso agli ingegneri di valutare in tempo reale le criticità posturali. Un aspetto cruciale di questo test è stata la valutazione delle tolleranze spaziali. Come dettagliato in precedenza (4.4.3), si è scelto di adottare una programmazione logico-cinematica priva di collisioni rigide per i corpi degli operatori. Durante il test in azienda, questa scelta si è rivelata ottimale per preservare il tracciamento ergonomico, gli ingegneri hanno potuto osservare le compenetrazioni visive tra le braccia dell'avatar e la carrozzeria, mappando i colli di bottiglia spaziali senza interrompere l'azione dell'operatore.

Per monitorare la corretta manipolazione del portellone (e del manipolatore robotico), gli esperti si sono affidati al sistema di soft collision e ai feedback aptico-visivi precedentemente implementati (4.5.4). Durante la simulazione, ogni volta che il componente entrava in collisione con la scocca, la repulsione visiva dell'oggetto permetteva agli ingegneri di individuare chiaramente i punti di urto.

Contestualmente, l'attivazione del materiale rosso nei punti di aggancio delle mani virtuali, ha permesso di monitorare i disallineamenti posturali dovuti allo scostamento tra la mano reale e la presa virtuale.

Questa rete di aiuti sensoriali si è rivelata cruciale soprattutto per l'operatore immerso in VR, che doveva compensare i severi limiti di visibilità riscontrati. In particolare, durante la movimentazione tramite il manipolatore robotico, gli ingombri meccanici occludevano quasi del tutto la linea di vista sui punti di aggancio. Per superare questa criticità, il sistema ha permesso un intervento collaborativo anche sulla macchina. Il secondo operatore, agendo in *multi-grab* direttamente sul manipolatore, ha aiutato il primo a guidare l'estremità robotica verso i punti giusti. Affidandosi poi unicamente al feedback aptico progressivo e al riscontro audio (*click*) di avvenuto aggancio, la coppia di operatori è stata in grado di orientarsi spazialmente e completare l'inserimento agevolmente.

La validità di questo collaudo è ampiamente documentata dalle acquisizioni multimediali. Come si evince dalla figura 5.1, l'infrastruttura ha permesso di monitorare l'operazione su più livelli visivi simultaneamente.



(a) Spazio reale **(b)** Panoramica terza persona

(c) Prima persona operatore 1 **(d)** Prima persona operatore 2

Figura 5.1: Confronto sincronizzato della simulazione nelle varie viste (caso 1)

5.3 Caso studio 2: installazione della console centrale e operazioni di fissaggio

Il secondo collaudo funzionale si è concentrato sulla simulazione del compito di assemblaggio della console centrale (divisa in due sub componenti) in vettura. A differenza dell'operazione sul portellone, questa fase ha richiesto un'analisi ergonomica differenziata in base a tre specifiche configurazioni di linea. L'obiettivo era simulare le diverse quote a cui la scocca del veicolo può essere posizionata lungo la linea di montaggio per agevolare il lavoro degli operatori. Le varianti di test hanno previsto: una prova con altimetria della vettura a 400 mm da terra, una con altimetria a 700 mm, e una con configurazione della vettura ruotata, ovvero con altimetria di circa 300 mm e rotazione del gancio girevole di 30°. Queste tre configurazioni si sono rese necessarie per arrivare a identificare nella terza configurazione la migliore configurazione altimetrica di lavoro, poiché minimizzavano le posture incongrue statiche dell'operatore e il rischio da movimentazione manuale dei carichi.

La procedura operativa richiedeva inizialmente il prelievo della prima console e il suo inserimento all'interno dell'abitacolo. Per questo componente sono state mantenute attive le collisioni geometriche contro la scocca dell'auto. Questo vincolo è stato cruciale per la validazione del test, obbligando l'utente a prestare la massima attenzione durante l'attraversamento del vano portiera, impedendo movimenti irrealistici e permettendo agli ingegneri di valutare i reali spazi di manovra. Per agevolare l'inserimento ed evitare posture incongrue prolungate, l'operazione è stata condotta in modo collaborativo da due operatori tramite la logica *multi-grab*.

Durante queste manovre negli spazi ristretti dell'abitacolo, il mantenimento del tracciamento posturale esigeva che il corpo dell'operatore fosse privo di blocchi fisici (4.4.3). Per consentire all'utente di percepire comunque gli urti contro la carrozzeria (ad esempio con la testa o le gambe durante l'accesso al veicolo), l'operatore si è affidato al sistema di feedback acustico spazializzato precedentemente implementato. I tonfi sonori dinamici hanno fornito all'utente una percezione istintiva della gravità dell'urto, permettendogli di correggere la postura, mentre gli ingegneri osservavano visivamente le compenetrazioni a schermo.

Per il corretto alloggiamento della console, gli operatori si sono lasciati guidare dagli aiuti sensoriali (vibrazione e click acustico) per trovare i punti di incastro esatti. Una volta posizionate le console, la simulazione ha previsto la fase di fissaggio meccanico. Gli operatori hanno prelevato un avvitatore simulato, vincolato a un'impugnatura a due mani per replicarne l'ingombro reale. Durante l'operazione di serraggio, il sistema ha restituito un duplice feedback a conferma dell'interazione: la simulazione riproduceva il suono del motore elettrico dell'utensile, accompagnato

dal riscontro visivo della vite che scendeva progressivamente nella propria sede, segnalando all'operatore l'effettivo fissaggio del componente.

L'osservazione di queste fasi, come si può notare nella figura 5.2, ha consentito agli ingegneri di processo di valutare l'accessibilità dei punti di fissaggio e delle sedi di alloggiamento. Il test è stato particolarmente significativo poiché la console, essendo collocata in una posizione molto interna rispetto alla sagoma del veicolo, richiedeva all'operatore di protendersi notevolmente verso il centro dell'abitacolo. L'analisi ha permesso di verificare in quale delle tre quote di sollevamento della vettura l'operatore, limitato dagli ingombri del telaio, dalla profondità della sede e vincolato alla presa a due mani, riuscisse a mantenere le posture più sicure e a completare l'installazione subendo il minor carico biomeccanico possibile.

(a) Spazio reale

(b) Panoramica terza persona

(c) Prima persona operatore 1

(d) Prima persona operatore 2

Figura 5.2: Confronto sincronizzato della simulazione nelle varie viste (caso 2)

5.4 Discussione dei risultati

La campagna di collaudo industriale descritta nei casi di studio ha permesso di validare a pieno l'architettura software sviluppata, confermando l'efficacia della realtà virtuale come strumento predittivo per l'ingegneria di processo. L'infrastruttura si è dimostrata robusta e capace di rispondere alle complesse esigenze dell'ergonomia preventiva, superando i limiti dei tradizionali software CAD e dei manichini digitali statici.

Dal punto di vista funzionale, le scelte ingegneristiche adottate si sono rivelate determinanti per il successo delle simulazioni. In primo luogo, l'implementazione del sistema *multi-grab* in un ambiente di rete sincronizzato ha permesso di replicare fedelmente le dinamiche di lavoro di squadra, un aspetto spesso trascurato nei simulatori VR mono-utente.

In secondo luogo, l'approccio ibrido alla gestione delle collisioni (*soft collision* per i corpi e collisioni rigide per i componenti) ha risolto efficacemente il compromesso tra la necessità di mantenere un tracciamento fluido (garantito dai sensori Xsens e dalla cinematica inversa) e l'esigenza di valutare gli ingombri spaziali reali. La validazione ha dimostrato che l'assenza di un blocco fisico sul corpo dell'operatore non inficia la simulazione, purché sia supportata da un solido ecosistema di feedback sensoriali.

Infine, il sistema di visualizzazione multi-prospettica (inclusa la telecamera spettatore) ha fornito agli ergonomi uno strumento di indagine potente, consentendo di individuare tempestivamente i colli di bottiglia spaziali e di valutare il carico biomeccanico prima ancora della realizzazione delle linee di montaggio fisiche.

Capitolo 6

Conclusioni e sviluppi futuri

In conclusione, il presente lavoro di tesi dimostra come l'integrazione profonda tra motori grafici real-time, sistemi di tracciamento avanzati e logiche di interazione ingegnerizzate possa trasformare radicalmente il paradigma della progettazione industriale. Il simulatore realizzato non rappresenta solo un traguardo tecnico, ma si configura come un tassello fondamentale verso un'Industria 5.0 sempre più sicura, efficiente e incentrata sull'essere umano.

Nonostante gli ottimi risultati ottenuti, il prototipo sviluppato pone le basi per ulteriori implementazioni tecnologiche. L'architettura modulare dell'applicativo si presta a diverse traiettorie di sviluppo futuro:

- **Caricamento dinamico degli ambienti di simulazione:** l'architettura modulare del software si presta in modo ottimale all'integrazione di nuove funzionalità operative dirette. Sarà infatti possibile implementare un sistema per l'importazione a *runtime* di intere scene tridimensionali, consentendo all'utente di scegliere dinamicamente in quale specifico *layout* di fabbrica posizionare la simulazione e gli oggetti fisici manipolabili, senza la necessità di ricompilare l'applicativo.
- **Parametrizzazione ergonomica in tempo reale:** si prevede inoltre lo sviluppo di un'interfaccia di controllo (GUI) dedicata alla modifica *live* delle misure, delle quote e delle altezze degli elementi presenti nella postazione. Questa funzionalità permetterà di riadattare istantaneamente l'ambiente di lavoro alle specifiche necessità procedurali e alle differenti caratteristiche antropometriche (percentili) degli operatori testati, massimizzando la flessibilità analitica del *Digital Twin*.
- **Valutazione ergonomica quantitativa automatizzata:** attualmente, la determinazione degli indici posturali è vincolata all'utilizzo di applicativi

esterni. Un naturale sviluppo futuro prevede l'integrazione di algoritmi in grado di processare i dati cinematici estratti dai sensori Xsens in tempo reale, al fine di calcolare automaticamente gli indici ergonomici standardizzati (come RULA, REBA o EAWS), fornendo una reportistica numerica istantanea del rischio biomeccanico.

- **Integrazione di periferiche aptiche avanzate:** per colmare ulteriormente il divario sensoriale legato alla percezione del peso, i futuri sviluppi potrebbero prevedere l'integrazione di esoscheletri o guanti aptici a resistenza di forza. Questo permetterebbe di restituire non solo la sensazione di contatto, ma anche lo sforzo reale necessario per la manipolazione di oggetti pesanti come il portellone.

Ringraziamenti

Desidero ringraziare tutto il personale di Stellantis con cui ho collaborato durante lo svolgimento del progetto per la disponibilità, la cortesia e il supporto fornito durante le attività di simulazione. La loro accoglienza e il tempo dedicato a illustrarmi alcuni aspetti della realtà aziendale hanno reso questa esperienza particolarmente formativa e mi hanno permesso di lavorare in un clima sereno e professionale.

Bibliografia

[1] Innovapptive. *Industry 4.0 vs. Industry 5.0: What's the Difference? And Why Should I Care?* Article explaining the main differences between Industry 4.0 and Industry 5.0, including automation, human–machine collaboration, sustainability, and personalized production. 2024. URL: https://www.innovapptive.com/blog/differences-between-industry-4.0-and-industry-5.0 (cit. a p. 1).

[2] NXP Semiconductors. *What Are Digital Twins?* Blog article explaining the concept of digital twins and their applications in industry, healthcare, and supply chains. 2023. URL: https://www.nxp.com/company/about-nxp/smarter-world-blog/BL-ARE-DIGITAL-TWINS (cit. a p. 3).

[3] Siemens PLM Software. *Use of Tecnomatix Tools Helps Spur ŠKODA's Automotive Market Growth.* Case study describing how ŠKODA Auto uses Tecnomatix tools for virtual commissioning, manufacturing simulation, and optimization of robotic production lines. 2013. URL: https://www.plm.automation.siemens.com/media/global/cz/Siemens-PLM-Skoda-Auto-Robotics-cs_tcm84-54134.pdf (cit. a p. 4).

[4] Michael Alba. *From Native CAD to Low-Poly FBX.* Article describing the workflow for converting native CAD models into optimized low-polygon FBX assets suitable for real-time applications such as games, VR, AR, and mobile visualization. 2018. URL: https://www.engineering.com/from-native-cad-to-low-poly-fbx/ (cit. a p. 5).

[5] L. Garotti et al. «Design Guidelines for Combining Digital Human Modeling and Virtual Reality to Foresee Workplaces Ergonomics Issues During Product Development». In: *Applied Sciences* 15.13 (2025), p. 7083 (cit. a p. 9).

[6] Epic Games e NASA. *Unreal Engine: How NASA is Using Immersive Technology for Design and Training.* Unreal Fest Orlando 2025. Ultimo accesso: 16 Febbraio 2026. 2025. URL: https://dev.epicgames.com/community/learning/talks-and-demos/jZqe/unreal-engine-how-nasa-is-using-immersive-technology-for-design-and-training-unreal-fest-orlando-2025 (cit. a p. 9).

[7] M. Schmid et al. «Using virtual reality to support the design of work systems in 3P workshops: a use case from the automotive industry». In: *Frontiers in Virtual Reality* 5 (2024), p. 1268780 (cit. a p. 10).

[8] L. Natrayan et al. «A Novel Approach for Integrating Virtual Reality in the Ergonomic Testing of Automotive Interior Designs». In: *SAE Technical Paper*. SAE International. 2024 (cit. a p. 10).

[9] Autocar Professional. *Audi Develops VR Software to Test Assembly Line and Logistics Processes*. Article describing how Audi developed virtual reality software to test assembly line workflows and logistics processes for the Audi e-tron GT production planning. 2019. URL: https://www.autocarpro.in/news-international/audi-develops-vr-software-to-test-assembly-line-and-logistics-processes-43571 (cit. a p. 10).

[10] BPR Group. *Ottimizzazione dei Tempi e delle Performance con i Sistemi UAS e EAWS*. https://www.bprgroup.it/blog/ottimizzazione-dei-tempi-e-delle-performance-con-i-sistemi-uas-e-eaws/. Published: September 27, 2024; Accessed: 2026-03-15. 2024 (cit. a p. 11).

[11] Fondazione Ergo-MTM Italia. *EAWS – Ergonomic Assessment Worksheet*. Sistema internazionale per la valutazione del carico biomeccanico e la progettazione ergonomica del posto di lavoro, conforme agli standard ISO di riferimento. 2024. URL: https://www.eaws.it/ (cit. a p. 11).

[12] Siemens AG. *Siemens – Company Overview*. Official Siemens website providing information about the company, its technologies, and business areas. 2025. URL: https://www.siemens.com/en-us/ (cit. a p. 11).

[13] Dassault Systèmes. *DELMIA Manufacturing Software*. Official page describing DELMIA, a digital manufacturing software platform for planning, simulating, and optimizing industrial production processes. 2024. URL: https://discover.3ds.com/it/delmia-manufacturing-software (cit. a p. 11).

[14] 3DS DELMIA. *How VR Transforms Manufacturing | Immersive Factory Validation with 3DEXPERIENCE*. Video demonstrating how virtual reality and the 3DEXPERIENCE platform are used to simulate and validate manufacturing processes in an immersive environment. 2024. URL: https://www.youtube.com/watch?v=yhp-RfyrhkI (cit. a p. 12).

[15] A. G. da Silva et al. «Virtual Reality and Digital Human Modeling for Ergonomic Assessment in Industrial Product Development: A Patent and Literature Review». In: *Applied Sciences* 12 (2022), p. 1084 (cit. a p. 11).

[16] L. Wang e H. Y. K. Lau. «Digital human modeling for physiological factors evaluation in work system design». In: *International Journal of Industrial Ergonomics* 88 (2022), p. 103281. DOI: 10.1016/j.ergon.2022.103281 (cit. a p. 12).

[17] *How to Use the Machine Learning Deformer in Unreal Engine*. https://dev.epicgames.com/documentation/en-us/unreal-engine/how-to-use-the-machine-learning-deformer-in-unreal-engine. Accessed: 2026-03-15. Epic Games Documentation, 2026 (cit. a p. 12).

[18] Ståle Andreas Skogstad, Kristian Nymoen e Mats Høvin. «Comparing Inertial and Optical MoCap Technologies for Synthesis Control». In: *fourMs - Music, Mind, Motion, Machines, Department of Informatics, University of Oslo* (2011). Figure: The NaturalPoint OptiTrack system (left) and the Xsens MVN system (right). Accessed via ResearchGate. URL: https://www.researchgate.net/publication/252066778_Comparing_Inertial_and_Optical_MoCap_Technologies_for_Synthesis_Control (cit. a p. 12).

[19] Jan P. Vox, A. Weber, K. I. Wolf, K. Izdebski, T. Schüler, P. König e F. Wallhoff. «An Evaluation of Motion Trackers with Virtual Reality Sensor Technology in Comparison to a Marker-Based Motion Capture System Based on Joint Angles for Ergonomic Risk Assessment». In: *Sensors* 21.9 (2021), p. 3145. DOI: 10.3390/s21093145 (cit. a p. 14).

[20] S. Salisu, S. A. Mohd Zaki et al. «Motion Capture Technologies for Ergonomics: A Systematic Literature Review». In: *Diagnostics* 13.15 (2023), p. 2593. DOI: 10.3390/diagnostics13152593 (cit. a p. 14).

[21] *Xsens –Motion Capture & 3D Motion Tracking*. https://www.xsens.com/. Accessed: 2026-03-15. Xsens by Movella, 2026 (cit. a p. 14).

[22] Maria Lua Nunes, Duarte Folgado, Carlos Fujão, André V. Carreiro, Sara Madeira e Hugo Gamboa. «Posture Risk Assessment in an Automotive Assembly Line Using Inertial Sensors». In: *IEEE Access* 10 (2022), pp. 88374–88389. DOI: 10.1109/ACCESS.2022.3196473 (cit. a p. 14).

[23] Lydia Qiang. *IMU Working Principle & Tactical Grade IMU Product Recommendations*. https://www.linkedin.com/pulse/imu-working-principle-tactical-grade-product-roro-yang-ygkhc/. Published: Dec 20, 2023; Accessed: 2026-03-15. 2023 (cit. a p. 15).

[24] *Unreal Engine –The Most Powerful Real-Time 3D Creation Tool*. https://www.unrealengine.com/en-US. Accessed: 2026-03-15. Epic Games, 2026 (cit. alle pp. 15, 24).

[25] Mario Lorenz et al. «CAD to VR–a methodology for the automated conversion of kinematic CAD models to virtual reality». In: *Procedia CIRP* 41 (2016), pp. 358–363. DOI: 10.1016/j.procir.2015.12.115 (cit. a p. 15).

[26] Pixar Animation Studios. *OpenUSD Documentation and Release Information*. https://openusd.org/release/index.html. Accessed: 2026-03-12. 2024 (cit. a p. 15).

[27] Khang Quang Tran, Ernst L. Leiss, Nikolaos V. Tsekos e Jose Daniel Velazco-Garcia. «A Real-Time Immersive Augmented Reality Interface for Large-Scale USD-Based Digital Twins». In: *Virtual Worlds* 4.4 (2025), p. 50. DOI: 10.3390/virtualworlds4040050 (cit. a p. 15).

[28] Epic Games, Inc. *Unreal Engine 5.7: The World's Most Open and Advanced Real-Time 3D Creation Tool*. Unreal Engine Documentation Portal. Official technical documentation and release notes for version 5.7. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/ (cit. alle pp. 15, 25, 61).

[29] Oleksandra Sobchyshak, Santiago Berrezueta-Guzman e Stefan Wagner. «Pushing the boundaries of immersion and storytelling: A technical review of Unreal Engine». In: *Displays* 91 (2026), p. 103268. DOI: 10.1016/j.displa.2025.103268 (cit. a p. 16).

[30] Nikola Rendevski, Diana Trajcevska, Mario Dimovski, Konstantin Veljanovski, Aleksandar Popov, Naile Emini e Dimitar Veljanovski. «PC VR vs Standalone VR Fully-Immersive Applications: History, Technical Aspects and Performance». In: *2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*. IEEE. 2022, pp. 1–4. DOI: 10.1109/ICEST55168.2022.9828656 (cit. a p. 16).

[31] SeyedehTina Zaringhalam, Mohammad Khalilzadeh e Omid Fatahi Valilai. «An integrated framework for collaborative product design and assembly based on STEP compatible ontology model in cloud manufacturing». In: *Discover Applied Sciences* 7.1 (2024), p. 37. DOI: 10.1007/s42452-024-06430-y (cit. a p. 16).

[32] Claudio Pacchierotti, Stephen Sinclair, Massimiliano Solazzi, Antonio Frisoli, Vincent Hayward e Domenico Prattichizzo. «Wearable haptic systems for the fingertip and the hand: taxonomy, review, and perspectives». In: *IEEE Transactions on Haptics* 10.4 (2017), pp. 580–600. DOI: 10.1109/TOH.2017.2689006 (cit. a p. 17).

[33] Paul James. *Dexta Shows Off Latest Exoskeleton Gloves That Let You Touch VR*. https://www.roadtovr.com/dexta-dexmo-exoskeleton-vr-glove-haptic-force-feedback-touch-vr/. Published: August 23, 2016; Accessed: 2026-03-15. Road to VR, 2016 (cit. a p. 18).

[34] L. Pérez, E. Diez, R. Usamentiaga e D. F. García. «Industrial robot control and collision detection using virtual reality». In: *Robotics and Computer-Integrated Manufacturing* 55 (2019), pp. 332–343. DOI: 10.1016/j.rcim.2018.03.015 (cit. a p. 18).

[35] Chiwoong Hwang, Tiare Feuchtner, Ian Oakley e Kaj Grønbæk. «Enriching Industrial Training Experience in Virtual Reality with Pseudo-Haptics and Vibrotactile Stimulation». In: *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*. ACM, 2024, pp. 1–11. DOI: 10.1145/3641825.3687728 (cit. a p. 18).

[36] Meta Platforms, Inc. *Meta Quest 3: Breakthrough Mixed Reality - Technical Specifications*. Official Meta Product Page. Official hardware specifications and product overview. 2023. URL: https://www.meta.com/quest/quest-3/ (cit. a p. 21).

[37] Meta. «Meta Quest Pro: price, release date, and specs». In: *Meta Blog* (2022). Official Meta blog post describing the Meta Quest Pro headset, its release date, pricing, and technical specifications. URL: https://www.meta.com/it-it/blog/meta-quest-pro-price-release-date-specs/ (cit. a p. 21).

[38] Meta Platforms, Inc. *Meta Link: Connect Meta Quest to a PC with Link and Air Link*. Meta Quest Help Center. Official technical documentation for PC VR connectivity. 2024. URL: https://www.meta.com/help/quest/articles/headsets-and-accessories/oculus-link/ (cit. a p. 21).

[39] Movella Inc. *MVN Analyze: Professional Human Motion Analysis Software*. Official Product Page. Software for full-body kinematic data analysis and motion capture. 2024. URL: https://www.movella.com/motion-capture/xsens-mvn-analyze (cit. a p. 22).

[40] Movella Inc. *MVN Real-time Network Streaming Protocol Specification*. Technical documentation for real-time data streaming to third-party engines like Unreal Engine via Live Link. Xsens Technologies B.V. 2023. URL: https://base.xsens.com/s/article/MVN-real-time-network-streaming-Protocol-Specification (cit. a p. 23).

[41] *Device Settings - Xsens (Motion LIVE Plugin for iClone)*. https://manual.reallusion.com/Motion_LIVE_Plugin/Content/Motion_LIVE/1/Xsens_for_iClone.htm. Accessed: 2026-03-15. Reallusion, 2025 (cit. a p. 23).

[42] Epic Games, Inc. *Live Link: Synchronizing Real-Time Data into Unreal Engine*. Unreal Engine Documentation. Official technical documentation for the Live Link framework and plugins. 2024. URL: https://docs.unrealengine.com/5.3/en-US/live-link-in-unreal-engine/ (cit. a p. 23).

[43] Movella Technologies B.V. *Xsens Unreal Live Link Plugin UE5*. Xsens Knowledge Base. Accessed: 2026-03-12. 2024. URL: https://base.xsens.com/s/article/Xsens-Unreal-Live-Link-Plugin-UE5?language=en_US (cit. alle pp. 23, 41).

[44] Blender Online Community. *Blender: A Free and Open Source 3D Creation Suite*. Blender Institute, Amsterdam. Official software documentation and reference manual. Blender Foundation, 2026. URL: https://www.blender.org/ (cit. a p. 26).

[45] Epic Games, Inc. *MetaHuman: High-Fidelity Digital Humans for Unreal Engine and Beyond*. Official MetaHuman Documentation and Creator Portal. Framework for the creation and animation of high-fidelity digital humans. 2026. URL: https://www.unrealengine.com/metahuman (cit. a p. 27).

[46] CLO Virtual Fashion Inc. *CLO3D: 3D Fashion Design Software for Advanced Garment Simulation*. Official CLO3D Global Website. Industry-standard software for 3D apparel design and real-time cloth simulation. 2026. URL: https://www.clo3d.com/ (cit. a p. 28).

[47] CLO Virtual Fashion. *CLO 3D Garment Animation to Metahuman | Unreal Engine Workflow*. https://www.youtube.com/watch?v=bTZn-8P9ffk. YouTube video; Accessed: 2026-03-15. 2025 (cit. a p. 28).

[48] NVIDIA Corporation. *NVIDIA VR Capture and Replay (VCR): Precision Testing and Performance Analysis for XR*. NVIDIA Developer Portal. SDK for capturing and replaying VR sessions to enable repeatable performance profiling and automated testing. 2026. URL: https://developer.nvidia.com/vr-capture-and-replay (cit. a p. 29).

[49] Epic Games. *Unreal Insights in Unreal Engine*. Official documentation describing Unreal Insights, a profiling and performance analysis tool used to collect, visualize, and analyze runtime data in Unreal Engine. 2024. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-insights-in-unreal-engine (cit. alle pp. 29, 30).

[50] Meta. *OVR Metrics Tool for Unreal Engine*. Official Meta developer documentation describing the use of the OVR Metrics Tool for monitoring performance metrics in VR applications developed with Unreal Engine. 2024. URL: https://developers.meta.com/horizon/documentation/unreal/ts-ovrmetricstool/ (cit. a p. 29).

[51] Meta. *Meta Quest Developer Hub (MQDH) for Unreal Engine*. Official Meta developer documentation describing the use of Meta Quest Developer Hub (MQDH) for developing, testing, and managing VR applications on Meta Quest devices with Unreal Engine. 2024. URL: https://developers.meta.com/horizon/documentation/unreal/ts-mqdh/ (cit. a p. 29).

[52] Skarredghost. *All you need to know on Application SpaceWarp on Unity: how-to and review*. https://skarredghost.com/2021/12/12/application-spacewarp-unity-how-to-review/. Published: December 12, 2021; Accessed: 2026-03-15. 2021 (cit. a p. 30).

[53] Epic Games. *Lumen Global Illumination and Reflections*. Official technical documentation for Lumen, Unreal Engine's dynamic global illumination and reflections system, providing real-time bounce lighting and high-fidelity environmental lighting. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine (cit. a p. 33).

[54] Epic Games. *Nanite Virtualized Geometry*. Official technical documentation for Nanite, a virtualized geometry system that enables the rendering of high-poly meshes with massive triangle counts in real-time, optimizing performance through intelligent level-of-detail management. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine (cit. a p. 33).

[55] Epic Games. *Datasmith Plugins for Unreal Engine*. Official documentation describing Datasmith tools and exporter plugins used to import scenes and assets from CAD and 3D design software into Unreal Engine. 2024. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/datasmith-plugins-for-unreal-engine (cit. a p. 35).

[56] Epic Games. *Blend Spaces*. Official documentation for Blend Spaces, assets that allow for smooth blending of animations based on multiple input parameters to create dynamic and responsive character movement. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/blend-spaces-in-unreal-engine (cit. a p. 42).

[57] Epic Games. *Animation Blueprints*. Official documentation for Animation Blueprints, specialized graphs used to control character animation logic, manage state machines, and perform complex pose blending in real-time. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-blueprints-in-unreal-engine (cit. alle pp. 45, 57).

[58] Epic Games. *Cloth Asset Editor*. Official documentation for Cloth Assets and the Chaos Cloth solver, detailing the setup of physical properties for fabric simulation and vertex-based collision constraints for skeletal meshes. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/cloth-simulation-in-unreal-engine (cit. a p. 47).

[59] Epic Games. *UPrimitiveComponent::OnComponentBeginOverlap*. Official API reference detailing the OnComponentBeginOverlap event, triggered when a component begins overlapping another component in Unreal Engine. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Components/UPrimitiveComponent/OnComponentBeginOverlap (cit. a p. 47).

[60] Epic Games. *UPrimitiveComponent::GetClosestPointOnCollision*. Official API reference detailing the GetClosestPointOnCollision function, used to calculate the closest point on a collision surface to a given world location. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Components/UPrimitiveComponent/GetClosestPointOnCollision (cit. a p. 47).

[61] Epic Games. *Map Range Clamped*. Official Blueprint API reference for the Map Range Clamped node, used to remap a value from a specified input range to a target output range while strictly clamping the result within the output limits. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Math/Float/MapRangeClamped (cit. a p. 48).

[62] Epic Games. *Play Sound at Location*. Official Blueprint API reference for the Play Sound at Location node, used to spawn and play an audio sound cue or wave at a specified world location without attaching it to an actor. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Audio/PlaySoundatLocation (cit. a p. 48).

[63] Epic Games. *UMaterialInstanceDynamic*. Official API reference for the UMaterialInstanceDynamic class, detailing how to instantiate and modify material parameters at runtime during gameplay or simulation in Unreal Engine. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/Materials/UMaterialInstanceDynamic (cit. a p. 55).

[64] Epic Games. *Control Rig in Unreal Engine*. Official documentation for Control Rig, a node-based rigging and animation system used to create custom character controls, procedural animations, and retargeting logic directly within Unreal Engine. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/control-rig-in-unreal-engine (cit. a p. 57).

[65] Epic Games. *Online Subsystem in Unreal Engine*. Official documentation covering the Online Subsystem (OSS) architecture and the OnlineSubsystemUtils module, explaining the network layers for session management, matchmaking, and platform-specific services. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/online-subsystem-in-unreal-engine (cit. a p. 61).

[66] Epic Games. *Loading and Unloading Levels*. Official documentation for the OpenLevel functionality and level streaming, detailing the process of transitioning between world maps and managing memory during level loading. 2026. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/loading-and-unloading-levels-in-unreal-engine (cit. a p. 62).

[67] Epic Games. *Game Mode and Game State*. Official documentation for the GameMode class, detailing its role in defining game rules, managing player spawning, and controlling the overall flow and logic of the simulation. 2026. URL: `https://dev.epicgames.com/documentation/en-us/unreal-engine/game-mode-and-game-state-in-unreal-engine` (cit. a p. 62).

[68] Epic Games. *AGameModeBase::ChoosePlayerStart*. Official API reference for the ChoosePlayerStart function, used within the GameMode to programmatically determine the most appropriate starting point for a player or agent during spawning. 2026. URL: `https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/GameFramework/AGameModeBase/ChoosePlayerStart` (cit. a p. 63).

[69] Epic Games. *Actor Movement Replication*. Official documentation for the Replicate Movement property, detailing the built-in system for synchronizing actor transformations, including location, rotation, and velocity, across a network. 2026. URL: `https://dev.epicgames.com/documentation/en-us/unreal-engine/replicating-actor-movement-in-unreal-engine` (cit. a p. 65).