Alma Mater Studiorum · Università di Bologna

Corso di Laurea in Ingegneria e Scienze Informatiche

Mixed Reality e AI nella chirurgia ortopedica: un ecosistema a supporto pre- e intraoperatorio per l'inserimento di protesi al ginocchio

Relatore:
Chiar.mo Prof.
Alessandro Ricci

Correlatore:
Dott.
Andrea Colombelli

Presentata da:
Alessandro Magnani

Sessione I
Anno Accademico 2024/2025

A Chri, a mia mamma e mio babbo, per tutto ciò che non si può spiegare a parole, ma che ogni giorno mi ha fatto sentire supportato e al mio posto.

A Andy e Monta, per aver reso lo studio qualcosa di condiviso, concreto e anche divertente.

Agli amici più stretti, per esserci stati nei modi più veri, con presenza, leggerezza e naturalezza — come solo chi sa starti vicino sa fare.

Grazie, davvero — per aver reso tutto questo qualcosa che valeva la pena vivere. È un piacere poter condividere la vita con voi.

Un ringraziamento doveroso anche al professor Alessandro Ricci per la disponibilità, e al dott. Andrea Colombelli per aver condiviso la sua esperienza in questo lavoro.

Introduzione

La chirurgia ortopedica rappresenta una delle aree specialistiche della medicina moderna, con un impatto diretto sulla mobilità e sulla qualità della vita dei pazienti. Tra gli interventi più comuni, la *protesi totale del ginocchio* (*Total Knee Arthroplasty*, TKA) viene eseguita su pazienti affetti da forme avanzate di *osteoartrite*, una patologia degenerativa che provoca deterioramento della cartilagine, dolore cronico e riduzione della funzionalità dell'articolazione.

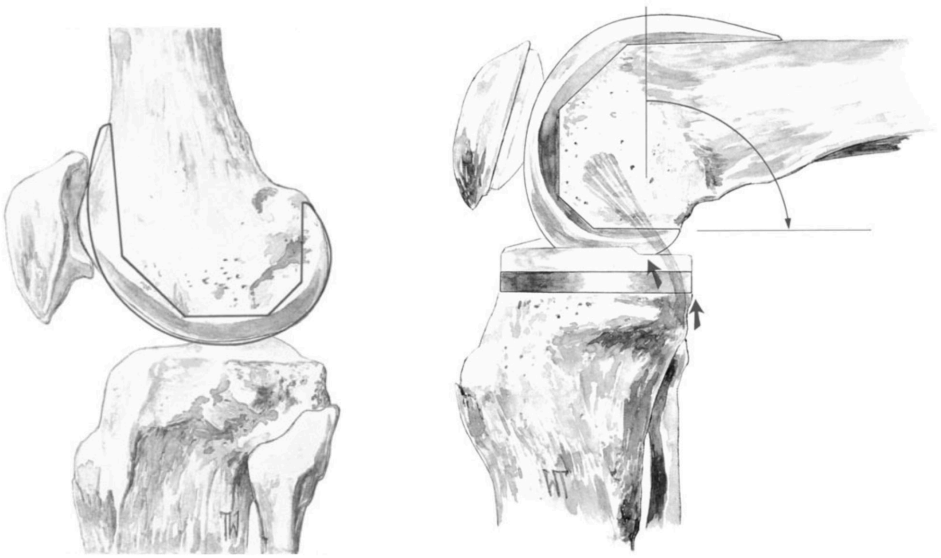

Figura 1: Anatomia del ginocchio e meccanismo di movimento dell'articolazione [1].

L'intervento chirurgico ha l'obiettivo di sostituire l'articolazione danneggiata con una protesi artificiale, attraverso una serie di *tagli ossei* che devono essere eseguiti con grande precisione. Attualmente, questi tagli vengono effettuati **a mano libera dal chirurgo**, seguendo la pianificazione preoperatoria e utilizzando strumenti meccanici. Tuttavia, questo metodo presenta alcune difficoltà: la precisione dell'intervento dipende molto **dall'esperienza del chirurgo**, dalla sua capacità di interpretare immagini preoperatorie bidimensionali e dalla memoria visiva, con il rischio di commettere errori durante l'esecuzione dei tagli.

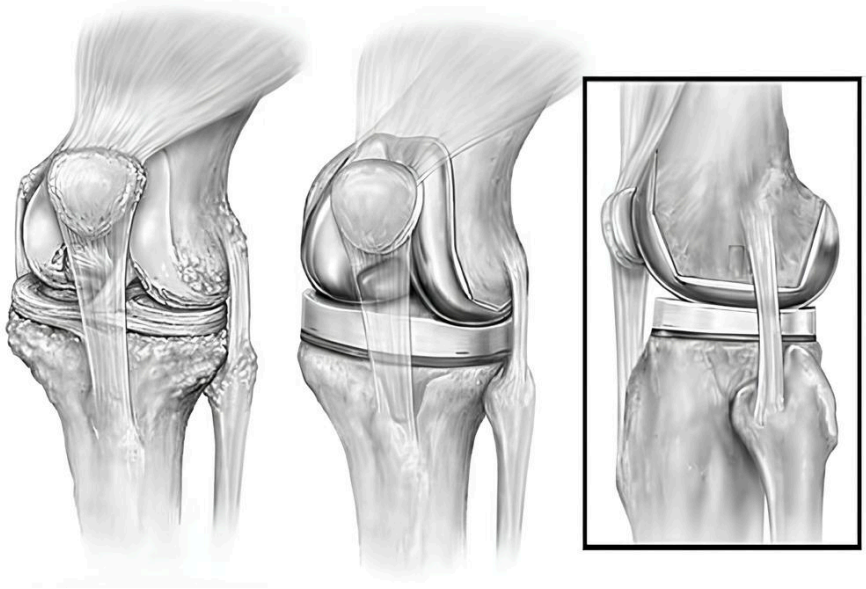

Figura 2: Confronto tra ginocchio con osteoartrite e risultato finale dopo impianto di protesi totale [2].

Durante una conferenza chirurgica alla quale ho avuto l'opportunità di partecipare, ho osservato una dimostrazione del sistema *Mako*, un braccio meccanico robotizzato già in commercio, che guida la mano del chirurgo durante l'intervento e si blocca automaticamente se si supera il piano di taglio preimpostato. Il sistema è stato fatto provare agli aspiranti chirurghi presenti. Si tratta di una tecnologia estremamente avanzata, già adottata in alcune strutture ospedaliere, ma **caratterizzata da costi molto elevati** sia per l'acquisto che per la manutenzione e l'utilizzo, rendendola **inaccessibile per molte realtà sanitarie**, soprattutto in contesti pubblici italiani dove le risorse destinate alla sanità sono spesso limitate.

In questo scenario emerge con forza il bisogno di **soluzioni tecnologiche alternative, più accessibili ed efficienti**, in grado di supportare il chirurgo nella pianificazione e nell'esecuzione dell'intervento. Tecnologie emergenti come la *Mixed Reality* e l'*Intelligenza Artificiale* offrono nuove possibilità in questa direzione, introducendo strumenti più intuitivi e intelligenti per migliorare l'accuratezza e la personalizzazione del trattamento.

Questa tesi nasce proprio dall'esigenza di esplorare soluzioni prototipali che possano contribuire a colmare il divario tra le attuali pratiche chirurgiche e le possibilità offerte dalle tecnologie digitali. Il progetto presentato è stato sviluppato in **collaborazione con due chirurghi ortopedici**, i quali hanno contribuito a identificare i problemi concreti riscontrati in sala operatoria e hanno fornito preziosi feedback sullo sviluppo del sistema. Ho inoltre assistito a un'intervento chirurgico, in modo da osservare e comprendere da vicino il processo di lavoro dei chirurghi.

Obiettivi del lavoro

L'obiettivo generale è quello di proporre un **ecosistema software prototipale**, concepito per essere utilizzato sia nella *fase preoperatoria*, attraverso l'analisi delle immagini TAC e la definizione dei piani di taglio, sia nella *fase intraoperatoria*, grazie alla visualizzazione in *realtà mista* direttamente in sala operatoria. L'ambizione è offrire un supporto più economico, flessibile e intuitivo, pur mantenendo l'accuratezza necessaria per un intervento chirurgico di alta precisione.

Nella fase preoperatoria, il prototipo consente di **importare ed elaborare immagini TAC** del ginocchio del paziente, fornendo strumenti per la visualizzazione tridimensionale e per l'interazione con i dati anatomici. In particolare, è possibile generare un *modello 3D dell'articolazione* e definire **piani di taglio personalizzati**, composti da marker tridimensionali, che saranno successivamente utilizzati come riferimento durante l'intervento.

Nella fase intraoperatoria, il chirurgo può usufruire del supporto fornito da un'applicazione per **visore in realtà mista** (*visionOS*), che permette di visualizzare il modello 3D precedentemente elaborato, con i relativi piani di taglio, direttamente nell'ambiente operatorio. Il modello è interattivo e può essere ruotato, ingrandito o spostato a piacimento, offrendo al chirurgo un riferimento tridimensionale intuitivo e costante durante l'operazione. Inoltre, è stata implementata una funzionalità che consente, tramite *pinch gesture*, l'**inserimento di marker virtuali direttamente sul ginocchio reale del paziente**, generando in tempo reale un piano di taglio visibile in Mixed Reality.

A supporto del contesto clinico preoperatorio, è stato inoltre sviluppato un **sistema software complementare**, non oggetto diretto di questa tesi ma strettamente collegato: un'*applicazione web* rivolta a medici e pazienti, che consente la gestione di radiografie, la pianificazione chirurgica e l'utilizzo di un **modello di intelligenza artificiale** per la classificazione automatica del grado di *osteoartrite*, corredato di *heatmap* esplicative. Questo sistema include anche un *modello generativo* capace di produrre radiografie artificiali, utili per l'addestramento e la sperimentazione senza compromettere la privacy del paziente. Questo progetto complementare è stato sviluppato in collaborazione con i colleghi **Andrea Matteucci** e **Simone Montanari**.

Metodologia adottata

Lo sviluppo del prototipo è stato guidato da un approccio pratico e orientato ai problemi reali, adattandosi progressivamente alle esigenze emerse durante il progetto e al confronto con i chirurghi.

Il lavoro è partito da un'analisi del contesto clinico reale, svolta attraverso incontri diretti con due chirurghi ortopedici, la partecipazione a una conferenza del settore e l'osservazione di un intervento in sala operatoria. Queste

attività hanno permesso di comprendere da vicino le criticità legate all'inserimento di protesi al ginocchio, in particolare l'esecuzione manuale dei tagli ossei, spesso soggetta a imprecisioni.

Sulla base delle osservazioni emerse dal confronto con i chirurghi, sono stati definiti i **requisiti principali** del sistema, sia funzionali (come la gestione delle TAC, la definizione dei piani di taglio e la visualizzazione in realtà mista), sia non funzionali (come la compatibilità tra dispositivi, l'usabilità e l'affidabilità del prototipo).

Il lavoro è stato organizzato in due componenti principali: un'*applicazione* per *macOS*, sviluppata in `Swift`, per la fase preoperatoria, e un'*applicazione* per *visionOS*, progettata per l'uso su visore durante l'intervento. La generazione del modello 3D avviene attraverso l'elaborazione delle TAC, ricostruendo l'anatomia del ginocchio. I dati elaborati, inclusi i piani di taglio, vengono poi esportati in un formato compatibile con l'*app* in realtà mista. Le interfacce sono state progettate in base al contesto d'uso: più ricche e dettagliate su desktop, più essenziali e gestuali su visore.

L'intero prototipo è stato sviluppato seguendo un approccio **incrementale e iterativo**, ispirato ai principi del *metodo Agile*, con cicli brevi di sviluppo, test e revisione. Ogni funzionalità è stata introdotta, testata e affinata progressivamente, sulla base dei feedback ricevuti dai chirurghi coinvolti. Questo metodo ha permesso di adattare rapidamente il progetto alle esigenze emerse in corso d'opera, mantenendolo costantemente allineato con i bisogni concreti del contesto clinico.

Nella progettazione del sistema sono stati adottati anche alcuni principi guida fondamentali del design dell'interazione, come il paradigma *less is more*, il principio *KISS* (Keep It Simple, Stupid) e alcuni dei *dieci principi euristici di Nielsen*, con l'obiettivo di garantire semplicità d'uso, chiarezza e coerenza tra le due applicazioni.

Struttura della tesi

La tesi è organizzata in sette capitoli, ciascuno dei quali affronta una fase specifica del progetto, partendo dal contesto clinico e tecnologico fino ad arrivare alla realizzazione tecnica del prototipo, alla sua valutazione e alle prospettive future.

Il **Capitolo 1** introduce il contesto clinico e tecnologico, descrivendo le pratiche attuali per l'inserimento di protesi al ginocchio, i limiti delle tecnologie esistenti, l'uso di visori in sala operatoria e il ruolo del feedback dei chirurghi nella definizione degli obiettivi del sistema.

Il **Capitolo 2** presenta il sistema software integrato per il supporto decisionale, composto dall'applicazione web per medici e pazienti e dai modelli di intelligenza artificiale per la classificazione dell'osteoartrite e la generazione

di radiografie sintetiche.

Il **Capitolo 3** analizza i requisiti funzionali e non funzionali del sistema, i criteri decisionali per la scelta delle tecnologie e i vincoli tecnici, operativi e clinici che condizionano l'applicabilità del prototipo.

Il **Capitolo 4** descrive l'architettura complessiva del sistema bimodale, il design specifico delle applicazioni *macOS* e *visionOS*, e le scelte progettuali per l'interfaccia utente ottimizzata per i diversi contesti d'uso.

Il **Capitolo 5** si concentra sui dettagli implementativi, dalle tecnologie utilizzate alla gestione dei dati, dalla ricostruzione 3D fino all'interazione gestuale in realtà mista.

Il **Capitolo 6** presenta la validazione del sistema attraverso la verifica del soddisfacimento dei requisiti, il confronto con le pratiche chirurgiche attuali e l'analisi delle limitazioni e delle criticità identificate.

Il **Capitolo 7** conclude la tesi con una sintesi dei risultati conseguiti, i contributi originali della ricerca e le direzioni per gli sviluppi futuri, delineando l'impatto potenziale del sistema nel panorama della chirurgia ortopedica assistita.

Indice

Elenco delle figure

Elenco dei listati

Capitolo 1

Contesto clinico e tecnologico

Per comprendere le motivazioni del sistema sviluppato è necessario analizzare il quadro di riferimento clinico e tecnologico. Nel presente capitolo vengono esaminate le criticità del processo attuale degli interventi di protesi totale del ginocchio, presentate le tecnologie disponibili con i relativi vincoli economici e normativi, e descritta la collaborazione con chirurghi ortopedici che ha guidato lo sviluppo del prototipo.

1.1 Il processo attuale per l'inserimento di protesi al ginocchio

La *protesi totale del ginocchio (Total Knee Arthroplasty, TKA)* è un intervento chirurgico finalizzato a sostituire l'articolazione del ginocchio danneggiata, nella maggior parte dei casi a causa di *osteoartrite avanzata* — una patologia degenerativa che colpisce l'articolazione e rappresenta una delle principali cause di disabilità tra gli anziani — con una protesi artificiale. L'obiettivo è quello di ridurre il dolore, ripristinare la mobilità articolare e migliorare la qualità della vita del paziente [3, 4].

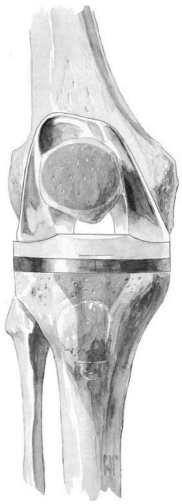

Figura 1.1: Struttura ossea del ginocchio e esempio di protesi impiantata.

Il percorso clinico che porta all'intervento si articola in diverse fasi. Inizialmente, il paziente si sottopone a esami diagnostici, nello specifico **ra-**

diografie e **TAC**, per valutare lo stato dell'articolazione e la gravità della patologia. Sulla base delle immagini e dei sintomi riportati, il chirurgo decide se procedere con l'intervento, definendo un piano preoperatorio che include la scelta della protesi e l'impostazione dei *piani di taglio osseo*.

Durante l'intervento, il chirurgo esegue una serie di passaggi ben definiti, mirati a rimuovere le superfici articolari danneggiate e a preparare l'alloggiamento per i componenti protesici.

L'intervento inizia con l'apertura del ginocchio: la pelle e i tessuti molli vengono delicatamente spostati per esporre l'articolazione sottostante.

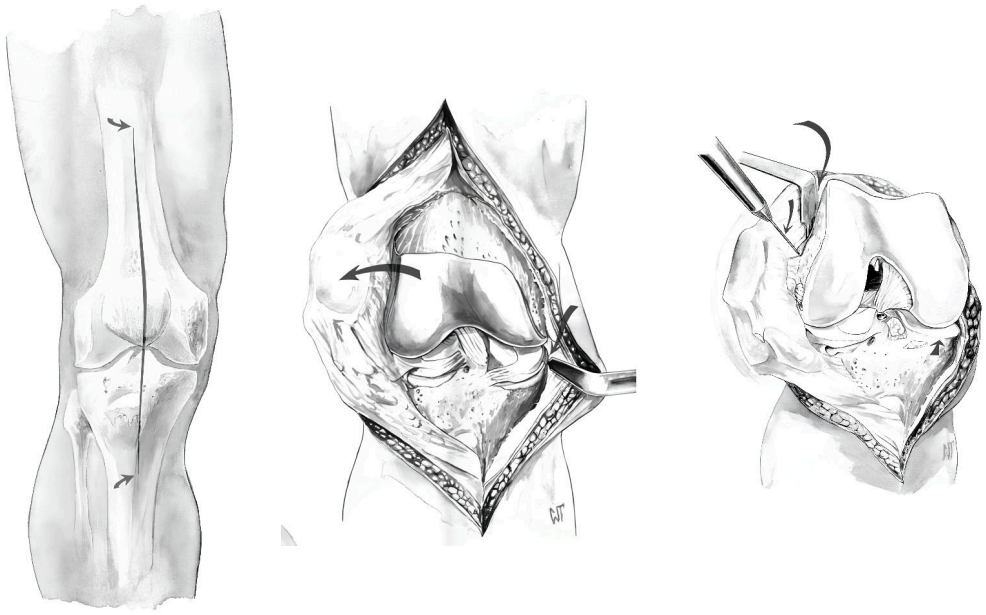

Figura 1.2: Apertura chirurgica del ginocchio ed esposizione dell'articolazione.

Successivamente, il chirurgo procede con l'esecuzione dei tagli ossei sul femore, iniziando dalla sua porzione inferiore e proseguendo con i tagli inclinati. Questi interventi contribuiscono a modellare l'estremità femorale in modo da renderla compatibile con la geometria della protesi da impiantare. In entrambi i casi si utilizzano delle guide meccaniche per assicurarsi che il taglio sia eseguito con l'angolazione e la profondità corrette.

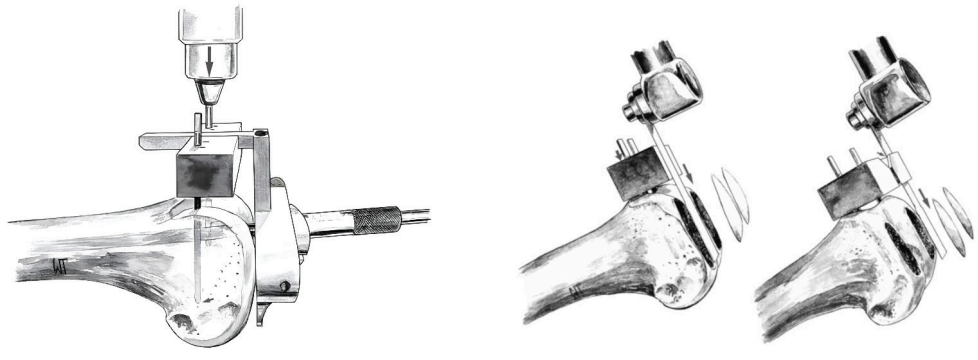

Figura 1.3: Tagli ossei sul femore con guide chirurgiche.

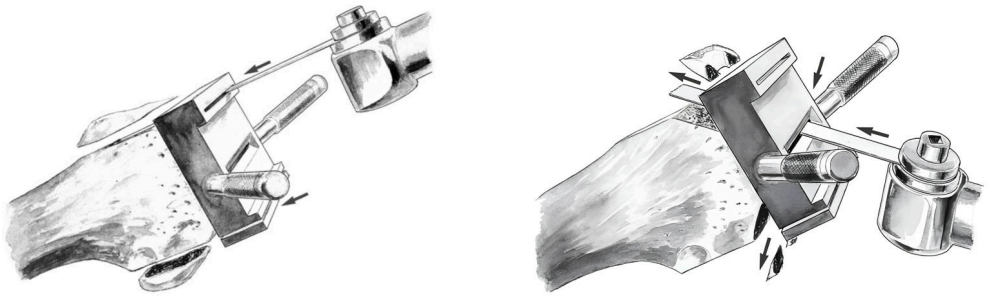

Figura 1.4: Tagli inclinati del femore per l'alloggiamento della protesi.

Una volta completati i tagli sul femore, si passa alla rimozione della parte superiore della tibia.

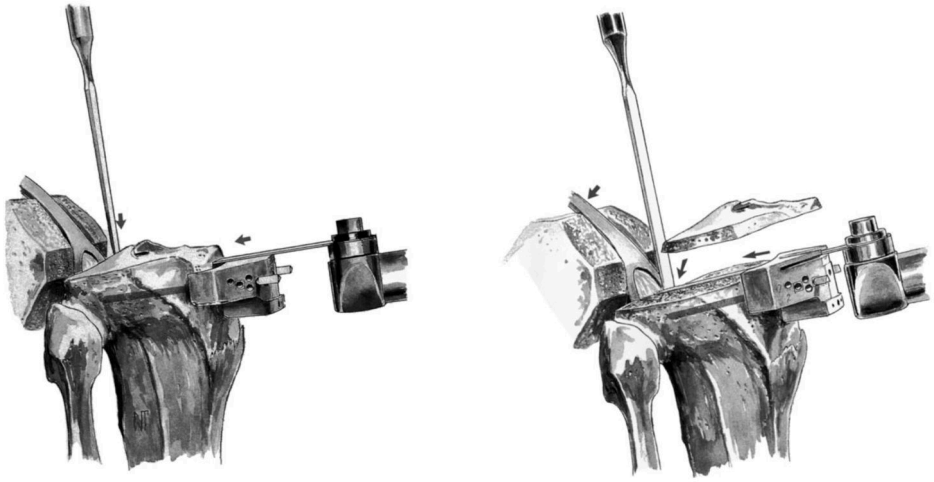

Figura 1.5: Rimozione della parte superiore della tibia.

Infine viene posizionata e fissata la protesi al femore e alla tibia.

Figura 1.6: Posizionamento finale della protesi.

1.2 Problematiche e margini di miglioramento

Sebbene il processo sia supportato da strumenti meccanici di guida, la precisione finale dipende ancora in larga parte **dall'esperienza e dalla mano del chirurgo** e dalla sua capacità di trasporre nella realtà le informazioni ottenute durante la pianificazione preoperatoria. Nella maggior parte degli interventi, i tagli vengono infatti eseguiti **a mano libera**, con margini di errore legati alla percezione visiva e all'interpretazione tridimensionale dell'anatomia.

Errori anche minimi nell'orientamento, nella profondità o nell'angolazione dei tagli possono causare **disallineamenti della protesi**, generando instabilità, dolore residuo o ridotta funzionalità. In alcuni casi, questi errori possono richiedere interventi correttivi, noti come *revisioni chirurgiche*, con conseguente aumento dei costi sanitari e impatto negativo sulla qualità della vita del paziente. Secondo un'analisi pubblicata su *ScienceDirect*, il tasso medio di revisione a 10 anni dopo una TKA si aggira attorno al **6,2%**, con valori che possono arrivare fino al **7,8%** in alcune casistiche [5].

Un ulteriore elemento critico è rappresentato dalla **variabilità tra chirurghi**, anche esperti: la pianificazione preoperatoria viene eseguita su immagini bidimensionali (radiografie o TAC) e tradotta in appunti, linee guida e misure, che il chirurgo deve poi interpretare durante l'intervento. La **trasposizione dalla pianificazione digitale alla realtà tridimensionale del corpo umano** non è banale e dipende fortemente dall'esperienza, dalla precisione manuale e dalla capacità del chirurgo di orientarsi spazialmente nel contesto anatomico reale.

A complicare ulteriormente il processo c'è la quasi totale **assenza di strumenti visivi tridimensionali in tempo reale** che possano assistere il chirurgo direttamente sul campo operatorio. Attualmente, le immagini preoperatorie sono visualizzate su un monitor separato e non sono integrate visivamente con l'anatomia del paziente. Questo costringe il chirurgo a una continua astrazione mentale tra ciò che osserva sullo schermo e ciò che esegue manualmente, aumentando il rischio di errori.

Per ridurre tali problematiche, alcune strutture ricorrono a tecnologie robotiche avanzate come il sistema *Mako*, in grado di guidare attivamente la mano del chirurgo durante l'intervento e bloccare i movimenti in caso di deviazioni dai piani di taglio prestabiliti. Tuttavia, questi sistemi presentano **costi molto elevati**: si stima che l'acquisto del robot *Mako* si aggiri attorno a **1,25 milioni di dollari**, con costi annuali di manutenzione superiori a **100.000 dollari** e un incremento di circa **2.400 dollari per intervento** rispetto alle TKA convenzionali [6, 7].

Questi numeri rendono l'adozione di sistemi robotici sostenibile solo per alcune strutture ad alta disponibilità economica, lasciando fuori molte realtà

ospedaliere, in particolare quelle pubbliche, dove i budget per l'innovazione tecnologica sono limitati o assenti.

In questo scenario emerge uno **spazio concreto per soluzioni alternative**, che sappiano coniugare semplicità d'uso, costi contenuti e reale utilità clinica.

1.3 Tecnologie esistenti e utilizzo di visori in sala operatoria

Negli ultimi anni la chirurgia ortopedica ha mostrato un crescente interesse verso l'integrazione di tecnologie digitali avanzate. Come spiegato precedentemente, tra quelle attualmente impiegate si possono identificare i **sistemi robotici**.

Parallelamente, si è affacciata una nuova frontiera: l'utilizzo della **Realtà Aumentata (AR)** e della **Realtà Mista (MR)** in ambito chirurgico. Queste tecnologie permettono di sovrapporre informazioni digitali tridimensionali alla scena reale, migliorando l'integrazione tra pianificazione e operatività.

Dispositivi come il **Microsoft HoloLens 2** e il **Magic Leap 2** sono già stati testati per visualizzare strutture anatomiche in tempo reale o per scopi formativi in neurochirurgia, ortopedia e chirurgia maxillo-facciale [8]. Il **Vision Pro** di Apple, introdotto nel 2024, offre prestazioni grafiche avanzate e supporto al *rendering* spaziale.

1.3.1 Aspetti regolatori e marcatura CE

L'utilizzo di visori in sala operatoria è subordinato al rispetto della normativa sui dispositivi medici. In Europa, ciò è regolato dal **Regolamento (UE) 2017/745** (MDR), che impone la marcatura **CE** per qualsiasi dispositivo destinato a finalità diagnostiche o terapeutiche [9]. Per ottenere tale marcatura, un dispositivo deve superare una serie di valutazioni cliniche, test di sicurezza e requisiti specifici.

Attualmente, solo alcuni visori, come *HoloLens 2* (nella versione per *Remote Assist*), sono stati utilizzati in contesti clinici, ma con finalità indirette, ad esempio per la formazione o il supporto remoto. In questi scenari, la certificazione CE è meno stringente, in quanto non viene coinvolto direttamente l'atto chirurgico.

Altri visori, come il **Magic Leap** o l'**Apple Vision Pro**, non sono ancora certificati come dispositivi medici. Le ragioni principali risiedono nell'assenza di una dichiarazione d'uso terapeutico da parte del produttore e nel fatto che non rispettano ancora gli standard tecnici previsti per i dispositivi

medicali (come le norme IEC 60601 o ISO 13485).

Nel caso specifico dell'*Apple Vision Pro*, va sottolineata una differenza tecnica rilevante rispetto agli altri visori. Il dispositivo utilizza una tecnologia di *video passthrough*: la realtà viene catturata da telecamere esterne e poi riprodotta sugli schermi interni del visore. Questo approccio introduce inevitabilmente una latenza visiva (stimata intorno agli 11 ms) tra il mondo reale e la percezione dell'utente [10]. Questo può rappresentare un limite in contesti chirurgici ad alta precisione. Al contrario, dispositivi come l'*HoloLens 2* utilizzano una tecnologia *optical see-through*, che permette di vedere direttamente l'ambiente reale attraverso lenti trasparenti, senza mediazione da parte di una fotocamera, e quindi senza ritardi visivi percepibili.

All'estero, il *Vision Pro* è stato comunque impiegato in alcuni studi sperimentali, come presso la *University of California, San Diego*, dove è stato utilizzato per mostrare al chirurgo immagini 3D e parametri vitali durante procedure minimamente invasive [11]. Tuttavia, il visore non è stato usato per guidare strumenti chirurgici o interagire fisicamente con il paziente. L'uso era limitato a scopo di visualizzazione e supporto informativo, all'interno di un protocollo etico autorizzato.

In Italia e in Europa, un utilizzo simile richiederebbe comunque l'adesione a specifici iter regolatori e l'ottenimento della certificazione come *dispositivo medico*.

In questo scenario in evoluzione si inserisce il prototipo presentato in questa tesi. Sebbene non ancora destinato all'uso clinico certificato, il progetto si propone come base per soluzioni future conformi ai requisiti di sicurezza, usabilità e validazione richiesti dalla normativa vigente.

1.4 Collaborazione con i chirurghi e ruolo del feedback clinico

Fin dalle fasi iniziali del progetto, il lavoro è stato sviluppato in stretto contatto con due chirurghi ortopedici: il dott. **Andrea Colombelli**, Primario presso l'Ospedale di Lugo, e il dott. **Federico Polidoro**, in servizio presso l'Ospedale di Ravenna. La collaborazione con loro ha avuto inizio già durante progetti precedenti alla tesi — in particolare nello sviluppo di un' *applicazione web* e di modelli di intelligenza artificiale per la classificazione dell'osteoartrite — condotti insieme ai colleghi **Andrea Matteucci** e **Simone Montanari**.

Durante lo sviluppo di questa tesi ho avuto occasione di approfondire ulteriormente il rapporto con i professionisti coinvolti, prendendo parte, insieme ai miei colleghi, a una conferenza tematica svoltasi a Bologna e rivolta a giovani chirurghi ortopedici. In tale evento, organizzato anche con il supporto del dott. Colombelli, ho potuto osservare in azione tecnologie avanzate per

l'intervento di protesi al ginocchio, tra cui il sistema robotico *Mako* di cui già discusso precedentemente. Le dimostrazioni su modelli anatomici artificiali hanno permesso di comprendere concretamente il funzionamento di questi strumenti e di raccogliere ulteriori spunti sul processo chirurgico reale.

L'esperienza più significativa si è poi svolta in ambito clinico: sempre su invito del dott. Colombelli ho assistito a un intervento reale di *protesi totale di ginocchio (TKA)*, seguendo tutte le fasi dell'operazione. Dopo aver visionato, insieme al chirurgo, le radiografie e TAC su cui era stata pianificata l'operazione, abbiamo osservato in diretta l'esecuzione dei tagli ossei e l'inserimento della protesi, comprendendo le difficoltà tecniche legate all'esecuzione manuale dell'intervento e all'uso di monitor distaccati dal campo visivo del paziente.

In quell'occasione, il dott. Colombelli ha espresso chiaramente il proprio interesse per soluzioni digitali immersive, sottolineando come il futuro della chirurgia ortopedica — a suo parere — sarà legato all'uso di visori di realtà aumentata o mista. Secondo lui, dispositivi come *Mako*, pur molto efficaci, sono oggi economicamente insostenibili per molte strutture pubbliche. I visori potrebbero rappresentare una valida alternativa più accessibile, capace di offrire un supporto visivo preciso e immediato durante l'intervento.

Questo tipo di feedback clinico, informale ma autorevole, ha avuto un ruolo fondamentale nell'orientare lo sviluppo del prototipo descritto in questa tesi. L'interazione con i chirurghi ha permesso di definire requisiti realistici, progettare interfacce coerenti con le esigenze operative e valutare con maggiore consapevolezza l'impatto potenziale delle soluzioni proposte nel contesto clinico reale.

1.5 Scenario d'uso

Il sistema integrato è progettato per supportare **chirurghi ortopedici** nella fase preoperatoria e intraoperatoria degli interventi di *TKA*. Gli utenti target sono professionisti medici con esperienza in chirurgia ortopedica che necessitano di strumenti avanzati per migliorare la precisione degli interventi.

Personas: Dr. Alessandro Rivetti
Dr. Alessandro Rivetti è un chirurgo ortopedico presso un ospedale pubblico con esperienza ventennale negli interventi di protesi al ginocchio. La sua priorità è migliorare la precisione degli interventi riducendo i rischi. Attualmente utilizza radiografie e TAC per la pianificazione ma trova difficoltoso trasporre le informazioni bidimensionali nella realtà tridimensionale dell'intervento. È interessato a tecnologie innovative ma ha budget limitati per l'acquisto di sistemi robotici.

Fase preoperatoria (applicazione per macOS):

1. Dr. Rivetti importa le TAC del paziente nel sistema attraverso l'interfaccia
2. Il sistema genera automaticamente un modello 3D del ginocchio
3. Il Dr. Rivetti analizza l'anatomia del ginocchio utilizzando gli strumenti di visualizzazione multiplanare, che permettono di osservare l'immagine nei tre piani principali: assiale (dall'alto verso il basso), coronale (fronte-retro) e sagittale (lato-lato)
4. Attraverso l'interfaccia, posiziona i marker per definire i piani di taglio
5. Il sistema visualizza in tempo reale i piani di taglio tridimensionali sovrapposti al modello osseo
6. Dr. Rivetti esporta il modello 3D con la pianificazione per l'intervento

Fase intraoperatoria (applicazione per visionOS):

1. Dr. Rivetti indossa il Vision Pro e carica il modello 3D
2. Il sistema visualizza il modello 3D del ginocchio con i piani di taglio in realtà mista e virtuale
3. Utilizzando il tracking spaziale, il chirurgo allinea il modello virtuale con l'anatomia reale del paziente
4. Durante l'intervento, i piani di taglio virtuali guidano visivamente l'esecuzione dei tagli ossei
5. Dr. Rivetti imposta i marker attraverso il pinch della mano, creando il piano di taglio ancorato spazialmente e sovrapposto al ginocchio reale del paziente

Capitolo 2

Sistema software integrato per il supporto decisionale

Prima del processo di pianificazione e realizzazione dell'intervento chirurgico analizzato nel capitolo precedente, il chirurgo deve affrontare una decisione fondamentale: stabilire se l'intervento sia effettivamente necessario. Per supportare questa fase critica è stato sviluppato un sistema software integrato che, pur non costituendo l'oggetto diretto della tesi, rappresenta una componente dell'ecosistema complessivo. Il presente capitolo descrive l'applicazione web per medici e pazienti e i due modelli di intelligenza artificiale sviluppati per la classificazione dell'osteoartrite e la generazione di radiografie sintetiche.

2.1 Visione d'insieme del sistema

L'osteoartrite viene comunemente classificata secondo la scala di *Kellgren-Lawrence*, che suddivide la patologia in cinque gradi di severità, da 0 (nessuna evidenza di artrite) a 4 (artrite grave), come illustrato in Figura 2.1. La presenza dell'osteoartrite si può notare dal restringimento dello spazio articolare e, nei casi più avanzati, dalla deformità dell'osso.

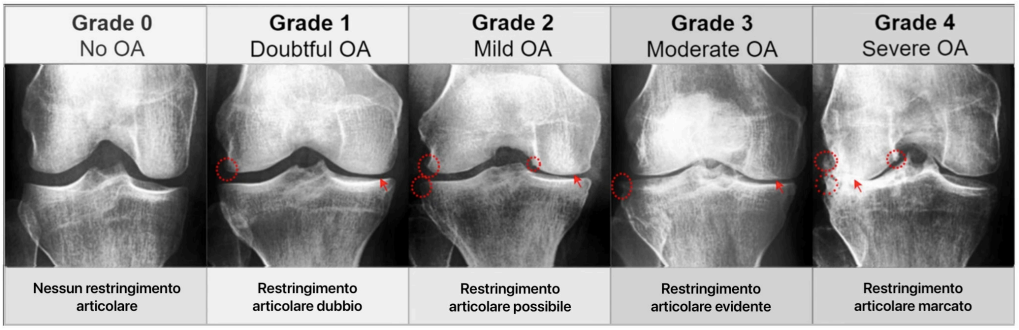

Figura 2.1: Gradi di severità dell'artrite secondo la classificazione di Kellgren-Lawrence.

Il processo decisionale che porta alla scelta di eseguire una *protesi totale del ginocchio (TKA)* non è sempre immediato o intuitivo. In particolare, nei casi intermedi della scala (grado 2 e 3), anche un chirurgo esperto può

trovarsi di fronte a un quadro clinico incerto, in cui la scelta di operare non è scontata e deve essere attentamente ponderata. L'assenza di strumenti digitali di supporto rende tale valutazione fortemente soggettiva, basata principalmente sull'esperienza del medico.

In risposta a questa esigenza, è stato progettato e sviluppato il sistema software integrato descritto in questo capitolo, con l'obiettivo di affiancare il chirurgo ortopedico nella fase di **valutazione clinica** e nel processo decisionale pre-intervento.

Esso si articola in due componenti principali:

- un'**applicazione web** accessibile da medici e pazienti, progettata per facilitare la condivisione di informazioni cliniche e l'organizzazione del percorso diagnostico.
- due **modelli di rete neurale**: uno classificatore per stimare la gravità dell'osteoartrite da radiografie del ginocchio (secondo la scala di *Kellgren–Lawrence*) e uno generativo, in grado di produrre immagini sintetiche coerenti con i diversi gradi della malattia.

Il sistema fornisce così un **supporto oggettivo e automatizzato** al chirurgo nella valutazione iniziale, migliorando la comunicazione medico-paziente e tracciando in modo sistematico tutte le fasi del percorso clinico. Il sistema software integrato si colloca in una fase **anteriore alla pianificazione preoperatoria** vera e propria: ha l'obiettivo di supportare la diagnosi e la decisione sull'opportunità dell'intervento. Nel caso in cui l'esito di tale processo porti alla scelta di operare, entra in gioco il contributo centrale di questa tesi (Figura 2.2).

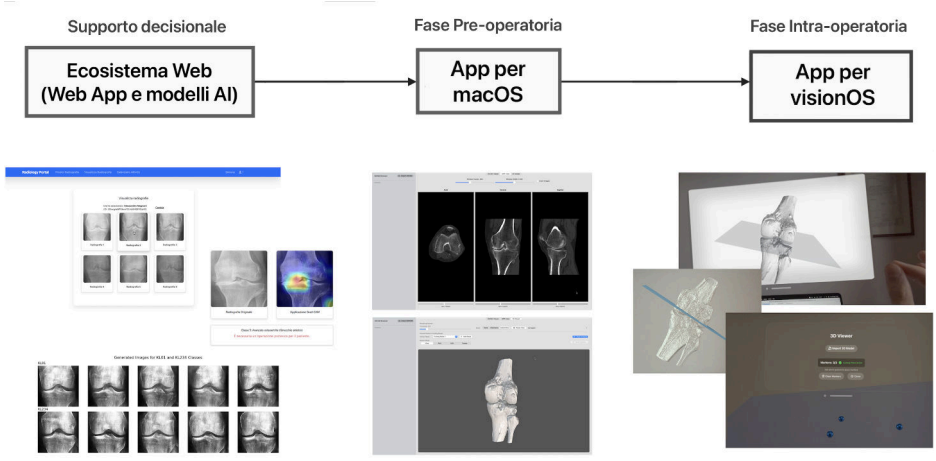

Figura 2.2: Flusso completo del sistema: dal sistema web per il supporto decisionale alla pianificazione preoperatoria fino al supporto intraoperatorio in realtà mista.

Sebbene questa piattaforma non rappresenti il focus diretto della presente tesi, ne costituisce una **componente abilitante e integrata**. Ha infatti influenzato la definizione dei requisiti, il contesto d'uso e la visione della

soluzione proposta, contribuendo a costruire un percorso digitale coerente che accompagna il paziente dalla diagnosi fino all'intervento chirurgico.

2.2 Applicazione web per medici e pazienti

La componente centrale del sistema software sviluppato è rappresentata da un'**applicazione web** accessibile da browser, progettata per facilitare l'interazione tra medico e paziente durante il processo di valutazione clinica. L'applicazione si propone come un **punto di accesso unico e centralizzato** per la gestione delle informazioni, la condivisione di documenti sanitari e il supporto alle decisioni terapeutiche.

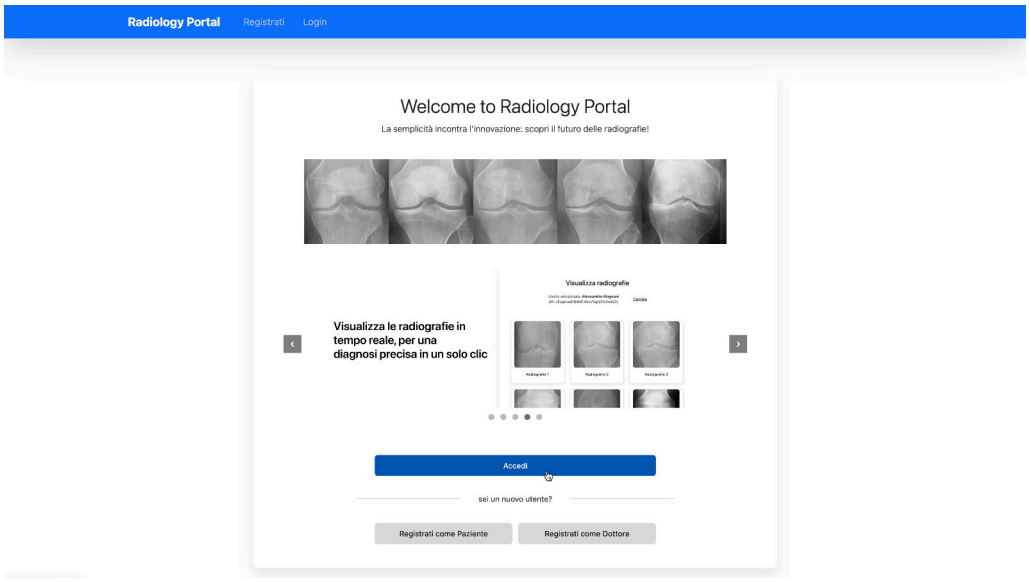

Figura 2.3: Interfaccia iniziale.

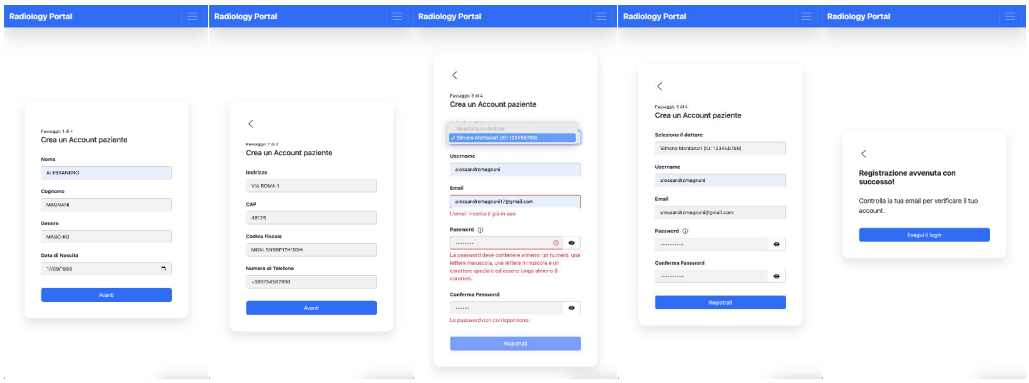

Figura 2.4: Interfacce di registrazione.

L'interfaccia dedicata ai medici è stata progettata per essere semplice, efficiente e centrata sulle necessità reali emerse durante il confronto con i chirurghi coinvolti.

Le principali funzionalità includono:

- **Caricamento radiografie**: il medico può caricare radiografie del ginocchio direttamente all'interno del sistema. Le immagini vengono automaticamente salvate e associate al paziente corretto.
- **Consultazione della cartella clinica**: è possibile visualizzare lo storico delle visite, le annotazioni precedenti, le diagnosi formulate e piani di trattamento in corso.

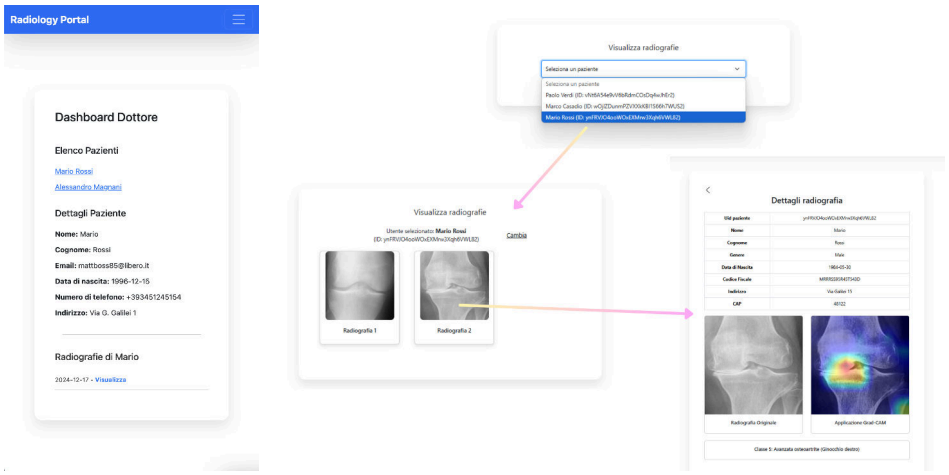

Figura 2.5: Dashboard per la consultazione delle cartelle cliniche e visualizzazione radiografie.

- **Visualizzazione dei risultati del modello AI**: per ogni immagine caricata, il sistema restituisce una valutazione automatica del grado di osteoartrite secondo la scala di *Kellgren–Lawrence*, accompagnata da una *heatmap* che evidenzia le aree più indicative della patologia.
- **Pianificazione dell'intervento**: qualora l'intelligenza artificiale suggerisca un livello avanzato della patologia, il medico può indicare l'intenzione di procedere con la preparazione di un intervento chirurgico e comunicarlo al paziente tramite il sistema.

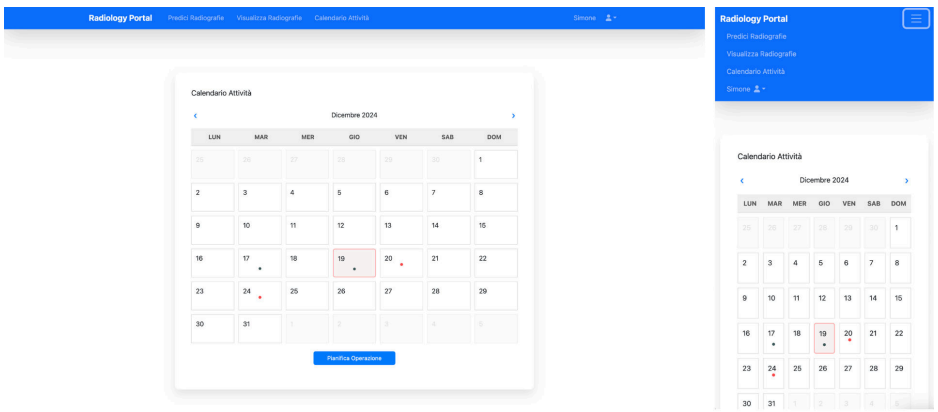

Figura 2.6: Calendario attività per la pianificazione degli interventi.

L'interfaccia lato paziente risulta **intuitiva** anche per utenti non esperti, con un linguaggio semplice e informazioni essenziali. Comprende le seguenti funzionalità:

- **Comunicazioni e notifiche**: il sistema notifica in automatico quando vengono caricati nuovi esami, pianificate visite o inserite comunicazioni dal medico. Il paziente può leggere i messaggi in una sezione dedicata e ricevere avvisi via notifica push.

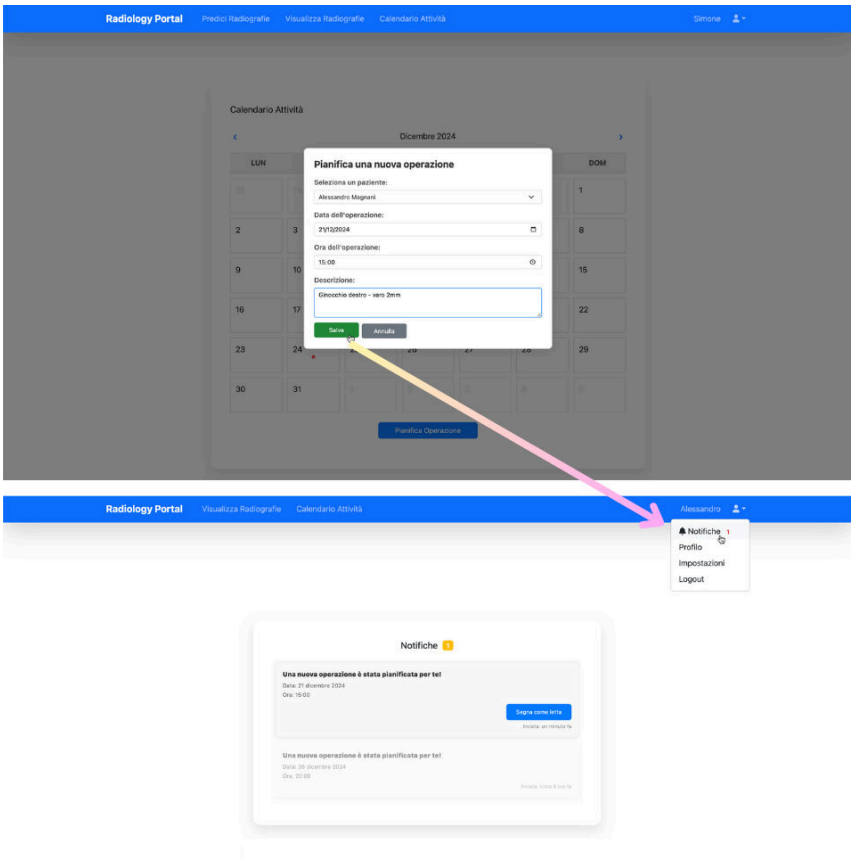

Figura 2.7: Interfaccia per la pianificazione di nuove operazioni e sistema di notifiche automatiche.

- **Visualizzazione dello stato clinico**: il paziente può consultare l'esito delle valutazioni ricevute, accedere ai documenti clinici caricati e seguire lo stato del proprio percorso (diagnosi → follow-up → eventuale indicazione chirurgica).

2.3 Modello AI per la classificazione dell'osteoartrite

Uno degli strumenti chiave del sistema è il modello di intelligenza artificiale basato su reti neurali convoluzionali (CNN), progettato per supportare il medico nella diagnosi dell'**osteoartrite del ginocchio**.

L'input del modello è costituito da una **radiografia del ginocchio** in proiezione antero-posteriore. Il sistema produce tre tipi di output:

- Il grado stimato di osteoartrite (intero da 0 a 4);
- Una decisione binaria che suggerisce se l'intervento chirurgico sia potenzialmente necessario o meno;
- Una **heatmap visiva** che evidenzia le regioni dell'immagine che hanno contribuito maggiormente alla decisione, aumentando la trasparenza e l'affidabilità percepita.

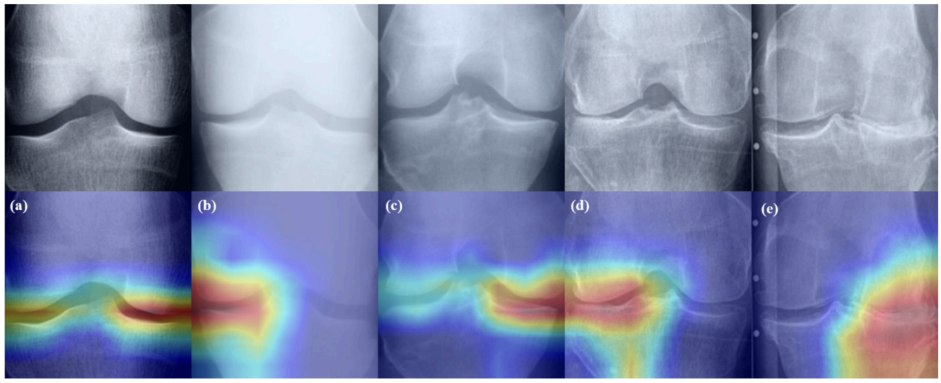

Figura 2.8: Esempi di heatmap.

Il modello è basato su una **ResNet-50**, una rete convoluzionale profonda pre-addestrata su ImageNet e successivamente *fine-tuned* su un dataset medico. In particolare, è stato utilizzato il dataset *OAI (Osteoarthritis Initiative)*, che contiene migliaia di radiografie etichettate secondo il grading Kellgren–Lawrence. Le immagini sono state preprocessate tramite normalizzazione, ridimensionamento e data augmentation per migliorare la generalizzazione del modello.

Il modello è stato valutato mediante **accuratezza**, **F1-score** e **confusion matrix**. I risultati ottenuti indicano una buona capacità del modello di distinguere tra le diverse classi, con performance particolarmente robuste nel discriminare tra i gradi estremi (Figure 2.9 e 2.10).

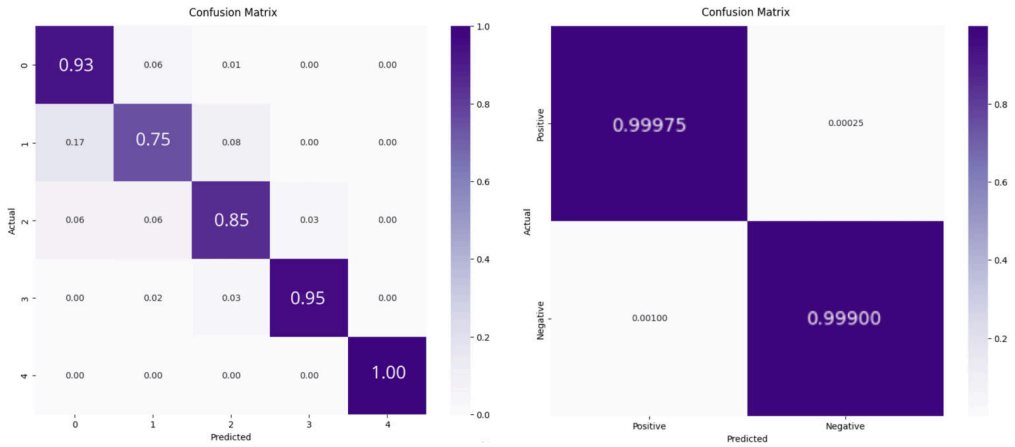

Figura 2.9: Confusion matrix per classificazione multiclasse (gradi KL 0-4) e binaria (necessità intervento chirurgico).

	Precision	Recall	F1
Grado 0	0.886	0.933	0.909
Grado 1	0.772	0.747	0.759
Grado 2	0.907	0.853	0.879
Grado 3	0.945	0.954	0.950
Grado 4	1.000	1.000	1.000
Media	**0.883**	**0.883**	**0.882**

	Precision	Recall	F1
Positive	0.99900	0.99975	0.99938
Negative	0.99975	0.99900	0.99938
Media	**0.99938**	**0.99938**	**0.99937**

Figura 2.10: Metriche di valutazione (precision, recall, F1-score) per classificazione multiclasse e binaria.

Il modello non si sostituisce al medico ma agisce come uno **strumento di supporto decisionale**, in grado di fornire una seconda opinione quantitativa e visivamente interpretabile. Il risultato dell'analisi viene mostrato direttamente nell'interfaccia web del medico (come descritto nel Capitolo 2.2), integrandosi perfettamente nel flusso clinico e favorendo una diagnosi più rapida e basata su evidenza algoritmica.

2.4 Modello AI generativo per la creazione di radiografie sintetiche

Uno degli ostacoli principali nello sviluppo di modelli AI in ambito medico è rappresentato dalla **limitata disponibilità di dati eitichettati** e dalla **necessità di proteggere la privacy dei pazienti**. Questo è particolarmente rilevante nel caso delle radiografie che sono considerate dati sensibili e soggette a rigorose normative di trattamento e conservazione.

Per affrontare questa problematica, all'interno del sistema software è stato sviluppato un modello di rete neurale *generativa*, capace di creare **radiografie sintetiche del ginocchio** realistiche a partire da un grado desiderato di osteoartrite sempre secondo la scala *Kellgren–Lawrence*. Queste immagini possono essere utilizzate per **evitare la distribuzione di dati reali**,

specialmente in contesti demo, pubblicazioni o formazione medica.

Prima dell'addestramento, tutte le immagini sono state sottoposte ad una fase di **preprocessing**, comprensiva di *equalizzazione*, *flip orizzontale*, *normalizzazione* e *focus filtering*. Il modello impiegato è una **Wasserstein GAN con Penalizzazione del Gradiente (WGAN-GP)** [12], un'architettura avanzata di *Generative Adversarial Network* che migliora la stabilità e la qualità delle immagini prodotte. L'intera pipeline, dal preprocessing al training della rete neurale, è illustrata in Figura 2.11.

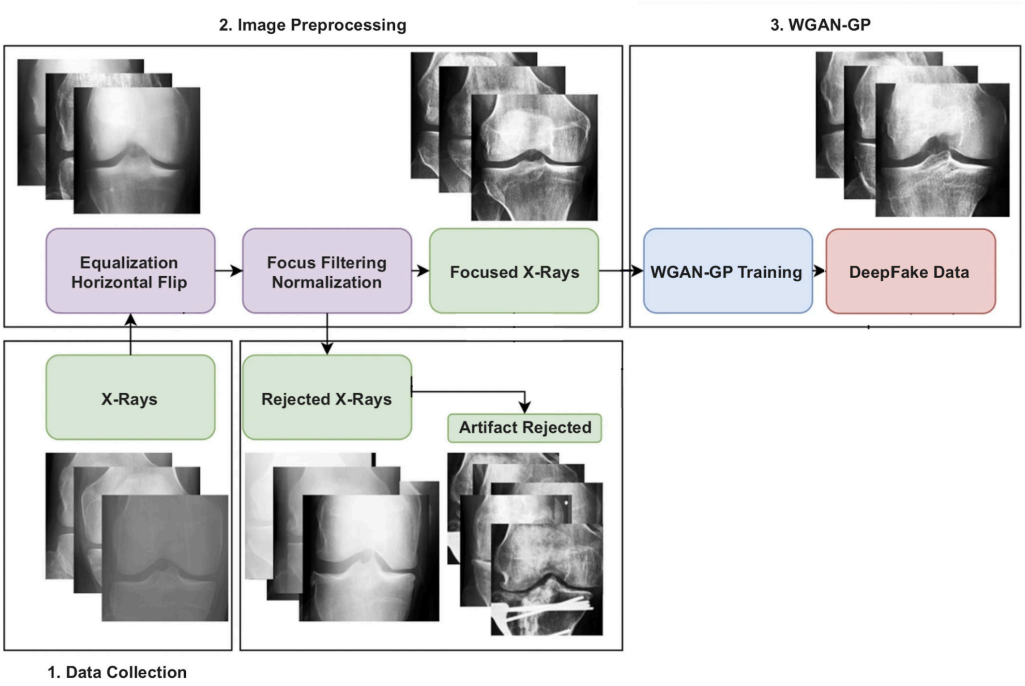

Figura 2.11: Pipeline di preprocessing e training della WGAN-GP per la generazione di radiografie sintetiche.

La generazione è *condizionata* sul grado di osteoartrite: al generatore viene fornito come input un valore intero da 0 a 4 e viene prodotto un output coerente con la morfologia tipica di una radiografia corrispondente a quel grado. Le immagini sintetiche prodotte dal modello trovano diverse applicazioni:

- **Training e validazione** del modello classificatore, permettendo di bilanciare il dataset o testare la robustezza della rete.
- **Dimostrazioni cliniche** e presentazioni pubbliche.
- **Formazione e simulazione medica**, offrendo a studenti e specializzandi un dataset vario e controllabile su cui esercitarsi.

Un esempio di immagini sintetiche generate dalla WGAN-GP è riportato in Figura 2.12, in cui si osserva la progressione visiva tra i diversi stadi di degenerazione articolare.

Figura 2.12: Radiografie sintetiche generate dal modello WGAN-GP.

Nel corso dell'addestramento, il modello ha mostrato un miglioramento progressivo nella qualità e coerenza delle immagini, raggiungendo un risultato soddisfacente dopo circa **50 epoche** di training (Figura 2.13). L'integrazione di questo modello rappresenta un esempio concreto di come l'AI generativa possa contribuire allo sviluppo di strumenti **più etici, flessibili e accessibili**.

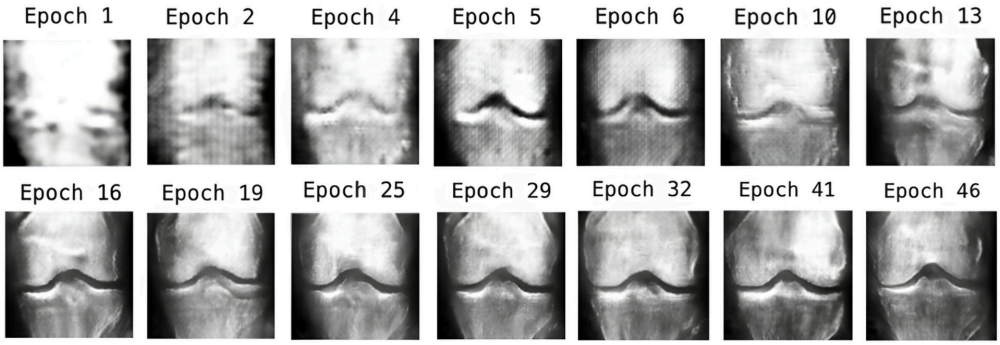

Figura 2.13: Evoluzione qualitativa delle immagini sintetiche durante le epoche di training.

Entrambi i modelli AI sviluppati hanno ottenuto risultati tali da **superare lo stato dell'arte** su benchmark interni e pubblici. In Figura 2.14 è infatti possibile osservare un confronto tra radiografie sintetiche ottenute con il modello proposto e quelle prodotte da modelli esistenti: **la maggiore fedeltà visiva e qualità strutturale delle immagini generate** è evidente.

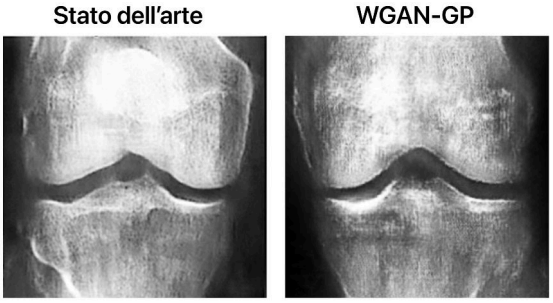

Figura 2.14: Confronto qualitativo tra immagini sintetiche generate dal modello proposto e da altri modelli presenti in letteratura.

Capitolo 3

Analisi dei requisiti

Dopo aver delineato il contesto clinico e presentato la piattaforma di supporto decisionale, il presente capitolo affronta il nucleo centrale della tesi: l'analisi dei requisiti per il sistema di realtà mista dedicato alle fasi di pianificazione preoperatoria e supporto intraoperatorio. Vengono definiti i requisiti funzionali che guidano la progettazione del sistema bimodale *macOS-visionOS*, i vincoli non funzionali per l'adozione clinica e le scelte tecnologiche alla base dello sviluppo del prototipo. L'analisi tiene conto delle criticità emerse dal confronto con i chirurghi e delle limitazioni tecniche e normative che condizionano l'applicabilità del sistema nel contesto reale.

3.1 Requisiti funzionali

La scelta progettuale di distinguere il sistema in due applicazioni — una per la *fase preoperatoria* su **macOS** e una per la *fase intraoperatoria* su **visionOS** — è frutto di precise considerazioni cliniche, tecniche e progettuali.

L'idea iniziale prevedeva lo sviluppo di un *modello di intelligenza artificiale* in grado di segmentare automaticamente il ginocchio a partire da radiografie e di tracciare in autonomia uno o più piani di taglio osseo. Questo approccio avrebbe richiesto l'annotazione manuale di decine di migliaia di immagini da parte di un chirurgo esperto, operazione non solo **impraticabile**, ma anche potenzialmente fuorviante. Ogni chirurgo, infatti, ha il proprio *stile operatorio* e le proprie preferenze, difficilmente generalizzabili in un modello univoco.

Sebbene esistano linee guida condivise a livello internazionale, la pratica chirurgica è fortemente influenzata da variabili individuali: ogni professionista sviluppa, nel tempo, una propria *"scuola di pensiero"*. Alcuni chirurghi preferiscono determinati angoli o tecniche per ragioni legate all'esperienza personale, mentre altri — spesso appartenenti a generazioni più giovani — adottano approcci più recenti, frutto dell'evoluzione tecnologica e della letteratura scientifica più attuale. Di conseguenza, un sistema di AI allenato su uno specifico insieme di decisioni soggettive (es. quelle del dott. Colombelli) rischierebbe di fornire *risultati non universali* o addirittura **fuorvianti**

per altri chirurghi.

A questo si aggiungeva una criticità tecnica legata alla discrepanza tra i dati di addestramento e il contesto reale d'uso. Un modello di intelligenza artificiale verrebbe tipicamente allenato su radiografie pulite, standardizzate e in bianco e nero, prive di elementi di disturbo. Tuttavia, durante un intervento chirurgico reale, le immagini disponibili sono ben diverse: a colori, con la presenza di sangue, tessuti molli e grasso. Questa distanza tra i due domini visivi renderebbe estremamente difficile l'applicazione diretta del modello in sala operatoria, compromettendone l'affidabilità e l'efficacia.

Per queste ragioni, si è deciso di mantenere **il completo controllo operativo in mano al chirurgo**, proponendo un approccio che ne *rispetti stile e giudizio clinico*. Da qui nasce l'idea di progettare un **workflow suddiviso in due fasi complementari**:

- una fase di **pianificazione preoperatoria**, supportata da un'applicazione per *macOS*, che deve permettere al chirurgo di:
  - importare immagini **TAC**;
  - generare e visualizzare un *modello 3D* del ginocchio;
  - creare e modificare i **piani di taglio osseo** direttamente sul modello;
  - esportare tutti i dati in un formato compatibile con il visore.
- una fase **intraoperatoria**, gestita da un'applicazione nativa per *visionOS*, che dovrà consentire al chirurgo di:
  - importare e visualizzare il modello 3D del paziente in **realtà mista**;
  - manipolare il modello (traslazione, rotazione, zoom) per analizzarlo da qualsiasi angolazione;
  - visualizzare i piani di taglio;
  - sovrapporre manualmente il modello al ginocchio reale tramite gesti (es. `pinch`), al fine di massimizzare la precisione di posizionamento.

Tale suddivisione consente una netta separazione tra la fase di analisi e pianificazione (a desktop, comoda e ad alta precisione) e quella intraoperatoria (immersiva e a mani libere), mantenendo **continuità semantica e operativa** tra i due momenti. Questo approccio garantisce il massimo della **flessibilità** e dell'**adattabilità clinica**, in linea con le esigenze espresse direttamente dai chirurghi coinvolti nel progetto.

Per illustrare concretamente le interazioni tra chirurgo e sistema, sono stati sviluppati diagrammi dei casi d'uso che rappresentano il flusso di lavoro clinico completo. Questi diagrammi traducono i requisiti funzionali in scenari

operativi specifici, evidenziando la sequenza logica delle operazioni dalla pianificazione preoperatoria al supporto intraoperatorio.

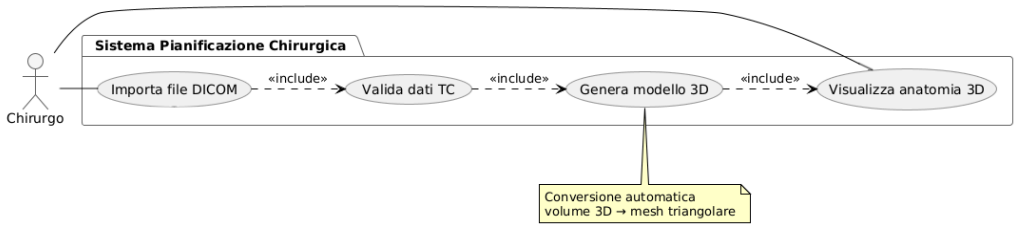

Figura 3.1: Diagramma dei casi d'uso: importazione e elaborazione TAC.

Il primo diagramma illustra la fase iniziale del workflow, dove il chirurgo importa le TAC del paziente e il sistema procede alla loro elaborazione per generare il modello tridimensionale.

In Figura 3.2 vengono mostrate le funzionalità di pianificazione, dove il chirurgo utilizza gli strumenti di visualizzazione e posiziona i marker per definire i piani di taglio.

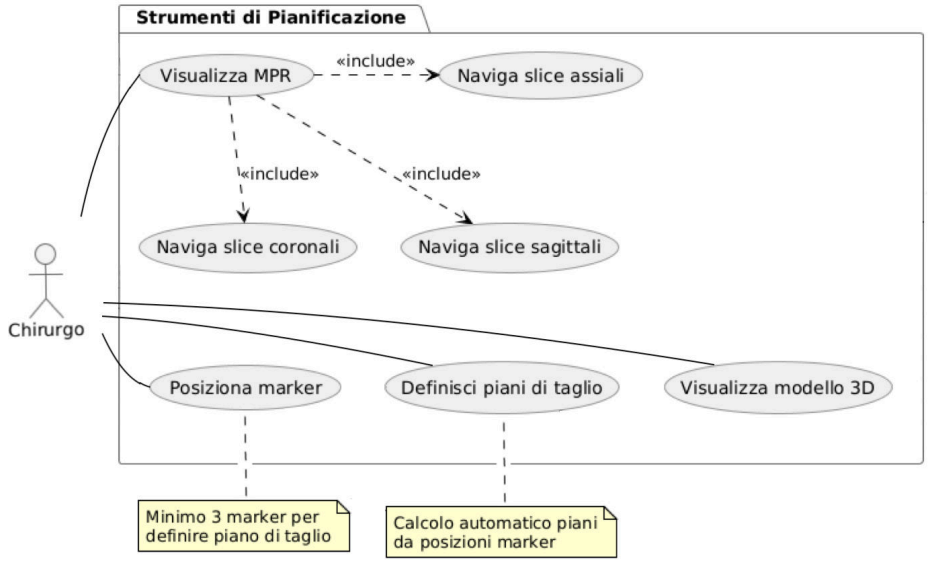

Figura 3.2: Diagramma dei casi d'uso: pianificazione preoperatoria.

Il terzo diagramma rappresenta la fase di transizione tra pianificazione ed esecuzione, con l'esportazione dei dati nell'applicazione per visore.

Figura 3.3: Diagramma dei casi d'uso: trasferimento pianificazione.

L'ultimo diagramma illustra l'utilizzo intraoperatorio del sistema, dove il chirurgo interagisce con il modello virtuale sovrapposto alla realtà durante l'intervento. Questo rappresenta il culmine del processo, dove la pianificazione digitale si traduce in supporto visivo diretto durante l'operazione.

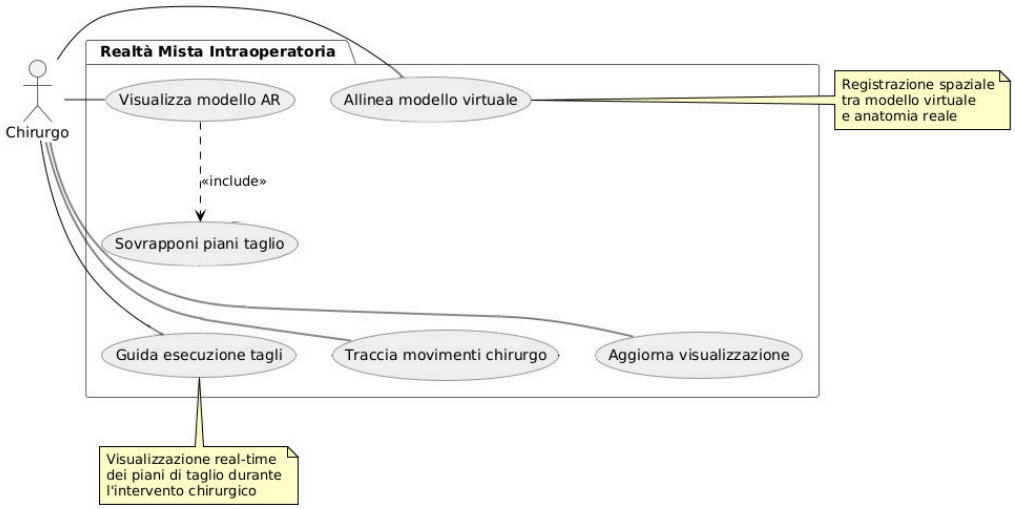

Figura 3.4: Diagramma dei casi d'uso: supporto intraoperatorio in realtà mista.

3.2 Requisiti non funzionali

Oltre alle funzionalità principali, lo sviluppo delle due applicazioni dovrà rispettare una serie di **requisiti non funzionali**, ovvero aspetti progettuali che incidono sull'esperienza d'uso, sulla sicurezza dei dati e sull'effettiva adozione clinica del sistema.
I principali requisiti non funzionali individuati sono:

- **Usabilità:** l'interfaccia dell'applicazione *visionOS* dovrà essere estremamente semplice e diretta, compatibile con l'uso in sala operatoria. L'interazione dovrà avvenire tramite gesture naturali, come il pinch, senza necessità di controller esterni, rendendo possibile l'utilizzo anche con guanti chirurgici. L'applicazione *macOS* dovrà risultare intuitiva, guidando il chirurgo nella creazione e modifica dei piani di taglio in pochi passaggi.

- **Prestazioni:** entrambe le applicazioni dovranno garantire tempi di caricamento rapidi e un *rendering* fluido e stabile del modello 3D del ginocchio, anche durante la manipolazione in tempo reale o il movimento della testa. In particolare, sull'applicazione *visionOS*, questo requisito sarà essenziale per evitare disorientamento visivo o affaticamento.

- **Privacy e sicurezza:** i dati clinici dovranno essere trattati secondo principi di minimizzazione e anonimizzazione, con l'obiettivo di ridurre al minimo il rischio di associazione diretta tra dati sensibili e

identità del paziente. In particolare, i file destinati alla fase intraoperatoria dovranno essere pseudonimizzati, in modo che non contengano informazioni personali identificabili. L'intero processo dovrà essere progettato tenendo conto degli standard di riservatezza richiesti in ambito sanitario.

- **Compatibilità:** il sistema dovrà essere progettato per garantire operatività anche in ambienti clinici con connettività limitata. In particolare, l'applicazione *visionOS* dovrà essere in grado di funzionare completamente offline durante la fase intraoperatoria, utilizzando dati preconfigurati ottenuti in fase preoperatoria. La procedura di preparazione dei dati dovrà essere semplice e affidabile, così da non dipendere da infrastrutture complesse o non disponibili in contesti ospedalieri standard.

- **Accessibilità clinica:** il sistema dovrà poter essere utilizzato anche da chirurghi con scarsa familiarità tecnologica. L'interfaccia dovrà privilegiare elementi visivi chiari, evitare tecnicismi superflui e riprodurre un flusso coerente con la pratica clinica consolidata, configurandosi come un'estensione naturale dell'attività chirurgica.

3.3 Criteri decisionali: scelta del visore e delle tecnologie

Dal punto di vista tecnologico, la scelta è ricaduta su **Apple Vision Pro** per diversi motivi. Innanzitutto, le sue prestazioni in termini di *rendering 3D*, tracciamento spaziale e interazione naturale (basata su occhi e mani) ssono risultate nettamente superiori rispetto a dispositivi concorrenti testati durante il processo di valutazione, come *HoloLens 2* e *Magic Leap 2*.

Come evidenziato in precedenza, **Apple Vision Pro non è attualmente certificato** come dispositivo medico e il prototipo descritto in questa tesi non è destinato all'uso reale in sala operatoria. Attualmente infatti, il dispositivo non è registrato né come *Classe I* né in altre classi più avanzate, poiché Apple non ha esplicitamente dichiarato intenti terapeutici per il visore. Tuttavia, si è scelto comunque di utilizzare questo visore in quanto ritenuto la soluzione con maggiore **futuribilità** in contesti clinici, anche grazie alla possibilità di utilizzare gli *occhi come puntatore*, una modalità unica e potenzialmente molto promettente in ambito chirurgico. Resta il dubbio legittimo che *Apple*, essendo un'azienda focalizzata su target generalisti, non abbia reale interesse a ottenere una certificazione medica per il *Vision Pro*. Tuttavia, anche qualora questo scenario si confermasse, il lavoro presentato può costituire un **punto di partenza** per futuri sviluppi clinici, magari su visori certificabili o sviluppati ad hoc.

3.4 Vincoli tecnici, operativi e clinici

Durante lo sviluppo del progetto è stato fondamentale mantenere una visione lucida dei vincoli e delle limitazioni che ne condizionano l'applicabilità nel contesto reale. Alcuni di questi limiti sono di natura tecnica, altri derivano da normative cliniche o da esigenze operative proprie dell'ambiente chirurgico.

Dal punto di vista tecnico, uno dei principali vincoli è rappresentato dal supporto ancora parziale che *visionOS* offre per operazioni complesse di condivisione e sincronizzazione di contenuti 3D tra dispositivi. Questo ha richiesto di progettare un sistema basato su *importazione manuale* dei modelli da *macOS* al visore, attraverso file intermedi. Inoltre, la giovane età del sistema operativo e la relativa scarsità di documentazione tecnica aggiornata hanno comportato un tempo aggiuntivo di sperimentazione e adattamento [13].

Dal punto di vista clinico, come già accennato, il limite più importante è legato all'**assenza di certificazione** del visore come dispositivo medico secondo il Regolamento (UE) 2017/745 (MDR) [14, 15].

Sul piano operativo, l'introduzione di nuovi strumenti in sala operatoria richiede particolare attenzione all'ergonomia, alla sterilità e alla fluidità dei flussi. Sebbene il visore possa essere indossato sopra cuffie e copricapi chirurgici, rimane ancora poco chiaro quanto possa risultare pratico durante un'operazione lunga, sia per comfort fisico sia per interferenze visive. Inoltre, la necessità di mantenere interfacce semplici e reattive è ancora più marcata in contesti clinici, dove ogni interazione deve essere immediata, affidabile e facilmente interpretabile anche sotto stress.

Infine, va ricordato che il personale sanitario ha competenze molto diverse sul piano tecnologico: un sistema utile in sala operatoria deve poter essere utilizzato con **minima formazione**, senza dover fare affidamento su figure tecniche esterne o su configurazioni complesse.

Tutti questi vincoli sono stati tenuti in considerazione nella progettazione del prototipo, con l'obiettivo di realizzare una soluzione coerente con lo stato attuale della tecnologia e allo stesso tempo pronta ad accogliere evoluzioni future in termini di dispositivi, certificazioni e pratiche cliniche.

Capitolo 4

Design

Dopo aver definito i requisiti funzionali e non funzionali del sistema, il presente capitolo si concentra sulla progettazione architetturale e sul design delle interfacce utente. Viene presentata l'architettura complessiva del sistema bimodale e le architetture specifiche di entrambe le applicazioni, analizzando la struttura modulare, i pattern comportamentali e le interazioni tra componenti. Il capitolo conclude con il design *UX/UI* ottimizzato per i diversi contesti d'uso: pianificazione preoperatoria su desktop e supporto intraoperatorio in realtà mista.

4.1 Architettura complessiva

Come spiegato precedentemente, il sistema si compone di due applicazioni **distinte e complementari**, sviluppate rispettivamente per *macOS* e *visionOS*. Infatti, come illustrato in Figura 4.1, il modello di utilizzo prevede tre fasi sequenziali ben definite:

1. **Fase preoperatoria** su *macOS*: il chirurgo importa immagini TAC in formato `DICOM`[1], ricostruisce il modello 3D del ginocchio e definisce i piani di taglio desiderati in modo interattivo.

2. **Esportazione** del modello arricchito in formato `.usdz`[2] o in formato `.scn`[3], compatibili con i visori Apple.

3. **Fase intraoperatoria** su *visionOS*: il chirurgo importa il modello 3D sul visore *Apple Vision Pro*, lo posiziona manualmente nello spazio reale e può manipolarlo (rotazione, scala, posizione) in sovrapposizione al ginocchio del paziente.

---

[1]Digital Imaging and Communications in Medicine: standard internazionale per la gestione, archiviazione e trasmissione di immagini medicali, ampiamente adottato nei sistemi ospedalieri.

[2]Universal Scene Description: formato 3D sviluppato da Pixar per la descrizione di scene tridimensionali, ottimizzato per dispositivi *Apple* e supporto nativo in `RealityKit`.

[3]SceneKit Scene: formato nativo Apple per scene 3D, utilizzato come formato intermedio per l'elaborazione e il post-processing dei modelli prima della conversione finale.

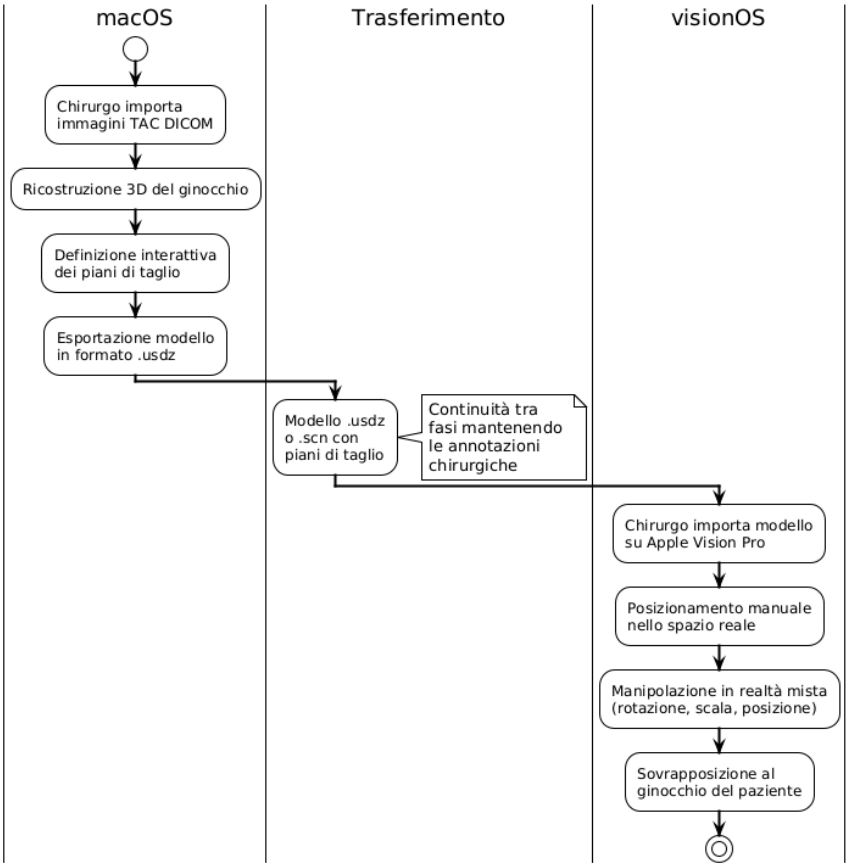

Figura 4.1: Flusso del sistema completo: dalla pianificazione preoperatoria all'assistenza intraoperatoria.

L'architettura distribuita del sistema, illustrata in Figura 4.2, mostra come le due applicazioni mantengano compiti specifici, pur condividendo un formato comune di interoperabilità tramite i file `.usdz` che agiscono come un **ponte tecnologico** tra le due piattaforme, mantenendo sia la forma del modello tridimensionale che i piani di taglio definiti durante la pianificazione.

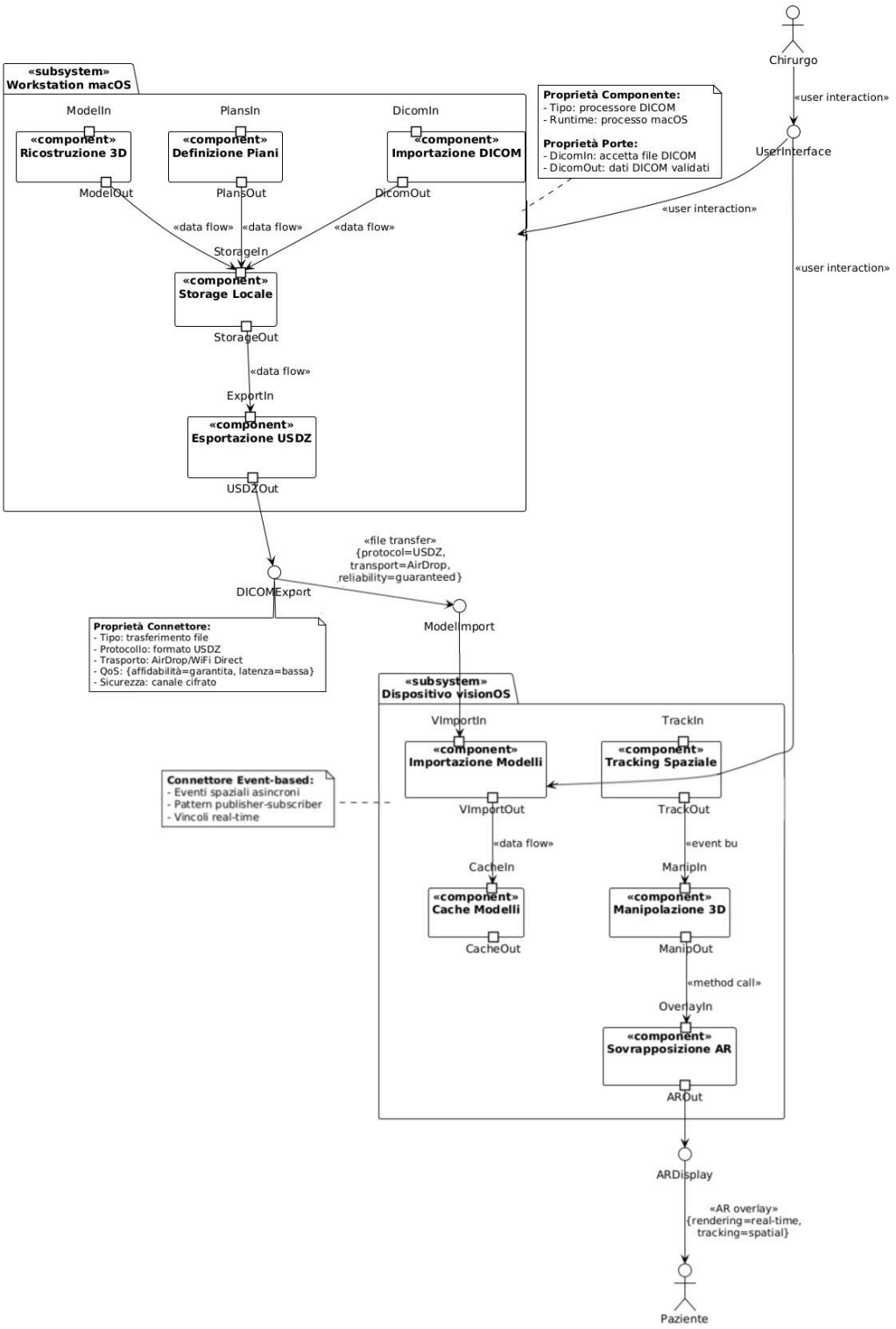

Figura 4.2: Architettura del sistema distribuito.

La distribuzione fisica del sistema su hardware eterogeneo, mostrata in Figura 4.3, riflette proprio questa separazione funzionale: la workstation *macOS* garantisce la potenza di calcolo necessaria per le elaborazioni intensive, mentre il dispositivo *Apple Vision Pro* si occupa dell'esperienza immersiva intraoperatoria, sfruttando le sue capacità di tracking spaziale e *rendering* a bassa latenza.

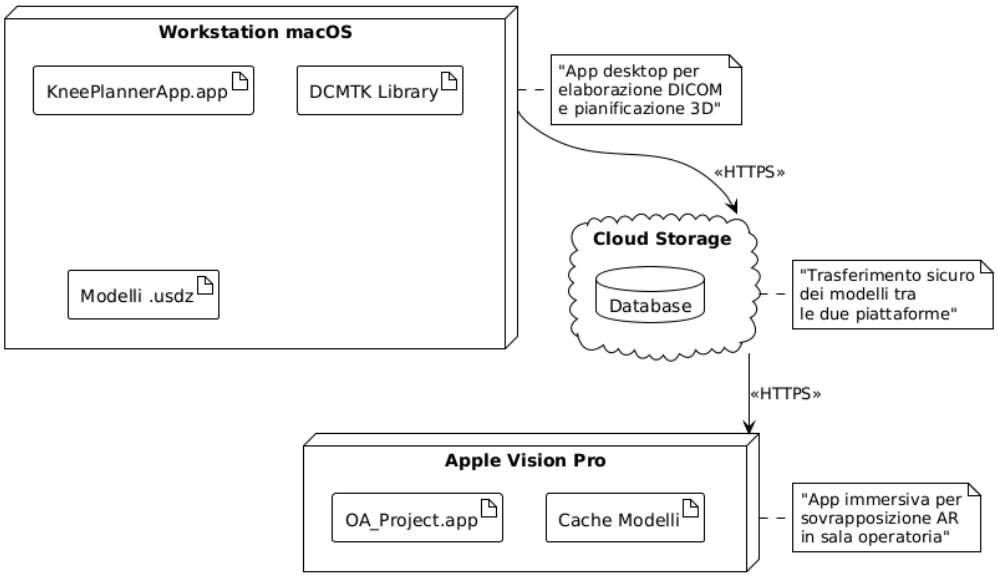

Figura 4.3: Deployment del sistema completo su piattaforme eterogenee.

Un aspetto fondamentale dell'architettura è la **gestione del trasferimento dati** tra le piattaforme. Attualmente, il sistema si affida esclusivamente al *cloud storage* per la condivisione dei file, una soluzione che offre praticità operativa ma richiede particolare attenzione sotto il profilo della sicurezza. È quindi fondamentale che l'utilizzo di tale modalità avvenga nel rispetto delle politiche di protezione dei dati dell'istituzione sanitaria, specialmente considerando la natura sensibile delle informazioni trattate. Questo approccio impone al team medico e tecnico di **adottare misure adeguate per garantire la riservatezza e l'integrità dei dati** durante tutto il flusso di lavoro.

4.2 Design dell'app per macOS

4.2.1 Struttura

L'applicazione per *macOS* adotta un'architettura modulare a quattro componenti principali, progettata per separare le responsabilità secondo il principio di *separation of concerns* e facilitare manutenibilità ed estensibilità:

- **Core**: responsabile della gestione del dominio medico e dei dati `DICOM`. Espone un'interfaccia di alto livello per l'accesso a pazienti, serie di immagini e volumi 3D, nascondendo la complessità dei formati medicali.
- **Rendering**: trasforma i dati volumetrici in rappresentazioni visuali 3D. Fornisce servizi di ricostruzione mesh, gestione della scena 3D e supporto per annotazioni chirurgiche.
- **UI**: orchestra l'interazione utente seguendo il pattern *MVVM*. Coordina la comunicazione tra i moduli sottostanti per fornire un'esperienza utente coerente.

- **Bridge**: adatta le librerie native *C++* all'ecosistema `Swift`, fornendo un'interfaccia pulita per l'accesso ai dati `DICOM`.

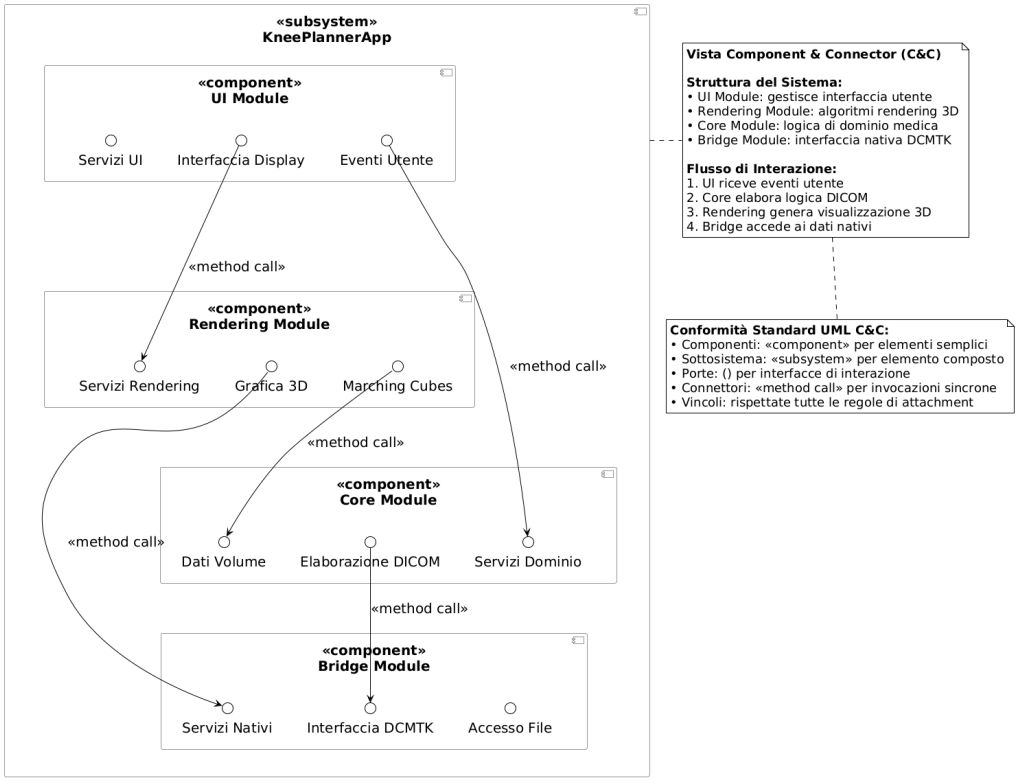

Figura 4.4: Struttura generale del sistema con architettura a livelli.

L'architettura adotta un approccio stratificato che implementa tre principi fondamentali:

- **Dipendenze unidirezionali**: ogni modulo dipende esclusivamente da quelli dei livelli inferiori, evitando dipendenze circolari e facilitando la comprensione e la manutenibilità dell'architettura (Figura 4.4).
- **Incapsulamento delle responsabilità**: ciascun modulo espone interfacce ben definite che nascondono i dettagli implementativi, consentendo modifiche interne senza impatti sui moduli che li utilizzano.
- **Separazione tecnologica**: le dipendenze da tecnologie esterne (come librerie native o *framework di rendering*) sono isolate in moduli specifici.

La Figura 4.5 evidenzia il flusso delle dipendenze: *Bridge* fornisce servizi primitivi, *Core* costruisce l'astrazione di dominio, *Rendering* specializza la visualizzazione 3D, mentre *UI* coordina l'interazione utente. L'applicazione principale (`KneePlannerApp`) gestisce l'inizializzazione e la configurazione del sistema.

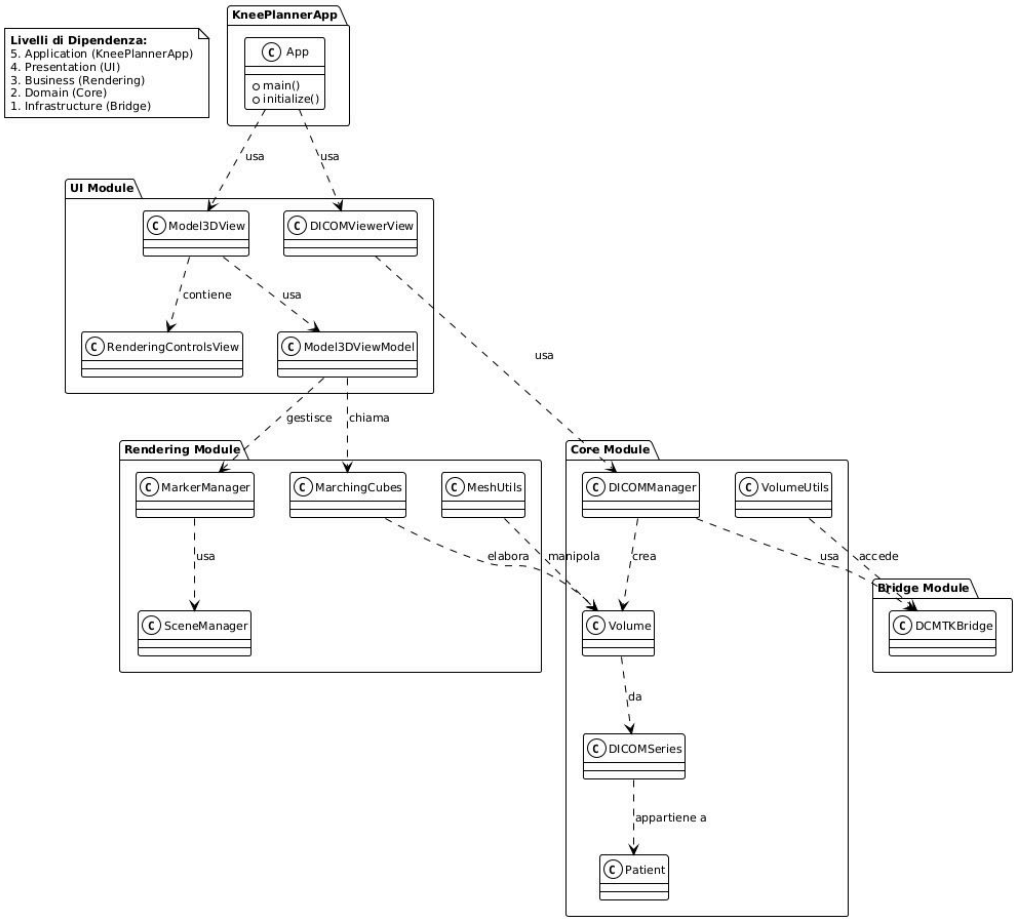

Figura 4.5: Diagramma delle dipendenze dettagliato tra i moduli.

4.2.1.1 Modulo Core

Il modulo *Core* implementa il **dominio applicativo**, definendo le entità e le regole di business per la gestione di dati medicali. La progettazione segue principi *Domain-Driven Design* per garantire che il codice rifletta il linguaggio e i concetti del dominio medico.

Il modulo si concentra sulla definizione delle entità di dominio (`Patient`, `DICOMSeries`, `Volume`) e sull'orchestrazione dell'importazione e validazione dei dati `DICOM`. Una delle funzionalità centrali è la conversione da serie 2D a volumi 3D con il calcolo automatico dei parametri spaziali necessari per la ricostruzione tridimensionale.

Il `DICOMManager` funge da *facade* principale per l'accesso ai servizi del modulo, mentre i servizi specializzati come il `DICOMService` forniscono operazioni specifiche per l'elaborazione dei dati medicali. La Figura 4.6 evidenzia la separazione tra entità di dominio e servizi applicativi, garantendo una struttura pulita e manutenibile.

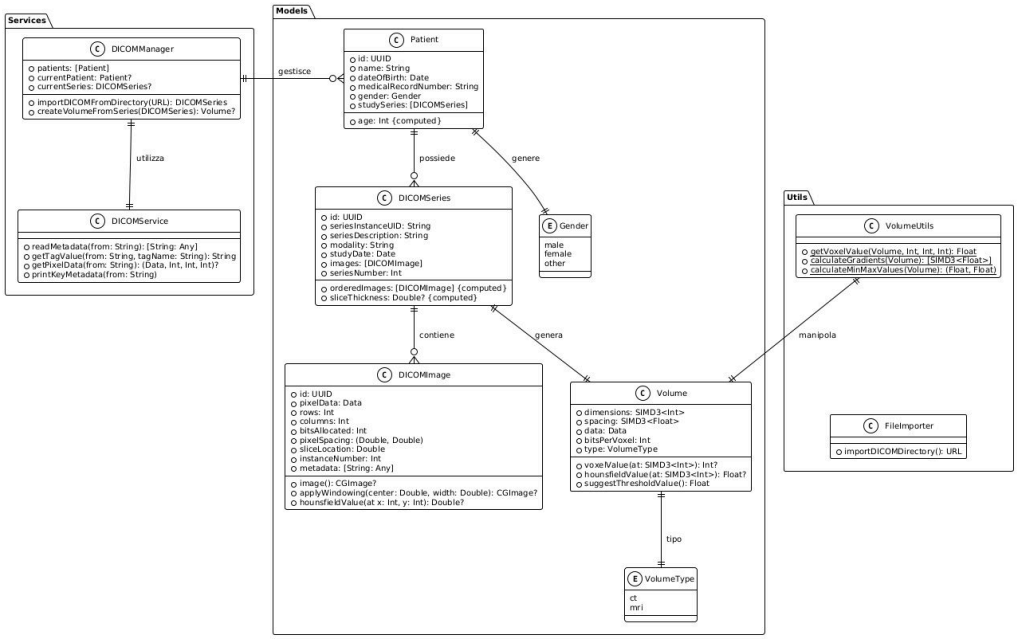

Figura 4.6: Diagramma delle classi del modulo Core.

Il modulo mantiene una dipendenza esclusiva dal *Bridge* per l'accesso ai dati nativi, preservando l'indipendenza da dettagli implementativi delle librerie esterne e facilitando l'evoluzione del sistema.

4.2.1.2 Modulo Rendering

Il modulo *Rendering* si specializza nella **trasformazione visuale** dei dati volumetrici, fornendo capacità di ricostruzione 3D e gestione della scena per supportare la pianificazione chirurgica.

Il modulo si specializza nella ricostruzione di mesh 3D da dati volumetrici e nella gestione di annotazioni chirurgiche come marker e piani di taglio. Coordina inoltre la scena 3D attraverso diverse modalità di visualizzazione e gestisce l'esportazione dei modelli.

Come si può notare in Figura 4.7, l'organizzazione interna del modulo di *rendering* è articolata in quattro *package* distinti: `Models`, che raccoglie le strutture dati principali; `Algorithms`, dedicato ai processi di ricostruzione 3D; `Scene`, responsabile della gestione degli elementi chirurgici interattivi; e `Utils`, che fornisce funzioni di supporto comuni agli altri componenti. Questa suddivisione permette l'evoluzione indipendente delle diverse componenti del *rendering*. Il `SceneManager` coordina la visualizzazione complessiva del modello 3D, mentre il `MarkerManager` si occupa specificamente delle annotazioni chirurgiche.

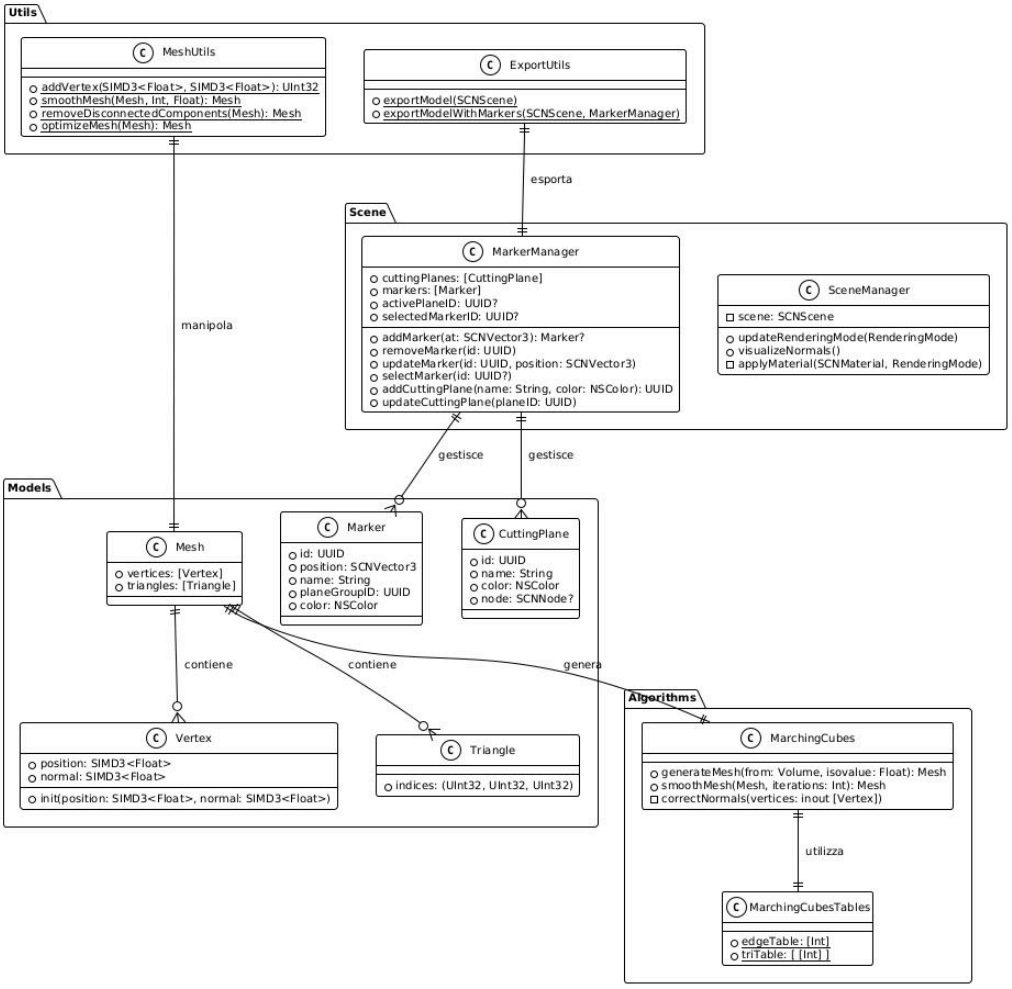

Figura 4.7: Diagramma delle classi del modulo *Rendering*.

4.2.1.3 Modulo UI

Il modulo *UI* implementa il **livello di presentazione**, orchestrando l'interazione tra utente e sistema secondo il pattern architetturale *MVVM* per garantire separazione tra logica di presentazione e logica di business.

Il modulo orchestra l'interazione tra utente e sistema, gestendo gli stati dell'interfaccia e coordinando la comunicazione tra visualizzazione 2D delle immagini `DICOM` e visualizzazione 3D dei modelli ricostruiti. Implementa inoltre i workflow clinici specifici per la pianificazione chirurgica, adattando i dati del dominio per la presentazione visuale.

L'organizzazione delle view segue una logica funzionale con due categorie principali: view specializzate per l'analisi delle immagini medicali bidimensionali e view dedicate all'interazione con i modelli tridimensionali. Entrambe le tipologie integrano controlli specifici per la manipolazione dei parametri di visualizzazione, mantenendo coerenza nell'esperienza utente.

4.2.1.4 Modulo Bridge

Il modulo *Bridge* risolve il **problema dell'integrazione tecnologica**, fungendo da adattatore tra l'ecosistema `Swift` e librerie native scritte in *C++* utilizzate per l'elaborazione dei dati `DICOM`. Si occupa della traduzione delle API native in interfacce accessibili da `Swift`, gestendo la conversione dei tipi di dati e isolando le dipendenze da codice esterno per semplificare la manutenzione del sistema.

L'utilizzo di *Objective-C++* rappresenta una scelta tecnologica strategica, in quanto consente l'interoperabilità diretta con il codice *C++* mantenendo la piena compatibilità con l'ambiente `Swift`. Questa architettura concentra tutta la complessità dell'integrazione in un modulo dedicato, schermando il resto del sistema dalle specificità delle librerie native.

Il modulo espone un'API semplificata che astrae completamente i dettagli dell'implementazione nativa, fornendo solo le operazioni realmente necessarie alla logica applicativa. Questo approccio facilita anche l'eventuale sostituzione delle librerie sottostanti, senza ripercussioni sui moduli superiori.

4.2.1.5 Architettura di Deployment

L'architettura di deployment è ottimizzata per sfruttare le capacità hardware specifiche della piattaforma *macOS*, garantendo performance adeguate per l'elaborazione di dataset medicali di grandi dimensioni.

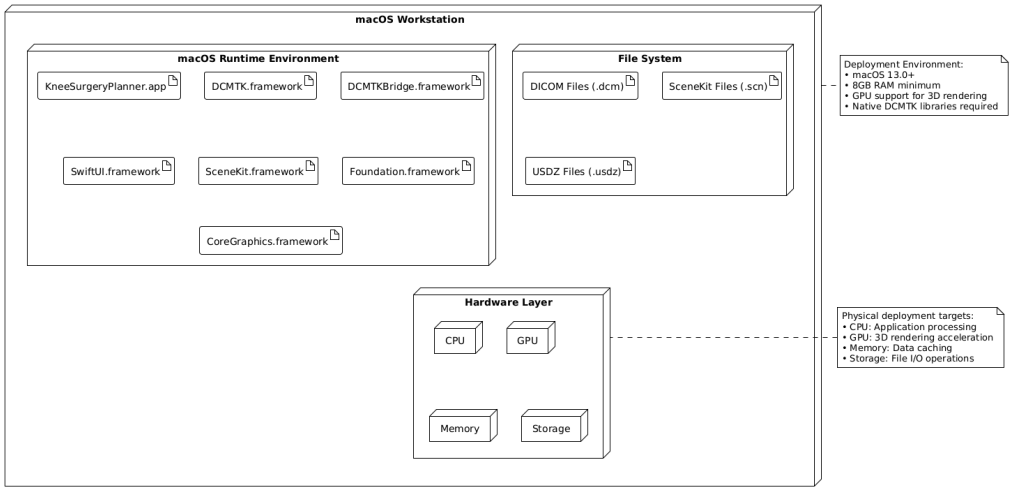

Figura 4.8: Architettura di deployment su *macOS*.

L'applicazione è distribuita come bundle nativo *macOS*, includendo tutte le dipendenze necessarie e sfruttando l'accelerazione hardware per le operazioni intensive. Come mostrato in Figura 4.8, l'architettura di deployment prevede l'uso della parallelizzazione della CPU per l'elaborazione dei volumi e della GPU per il *rendering* 3D, garantendo elevate prestazioni e responsività anche con dataset complessi.

4.2.2 Comportamento

Il comportamento del sistema illustra come i moduli collaborino attraverso flussi operativi specifici per trasformare dati grezzi in informazioni utilizzabili dal dominio applicativo. Due processi principali caratterizzano l'operatività del sistema: l'importazione di dati `DICOM` e la ricostruzione di modelli tridimensionali.

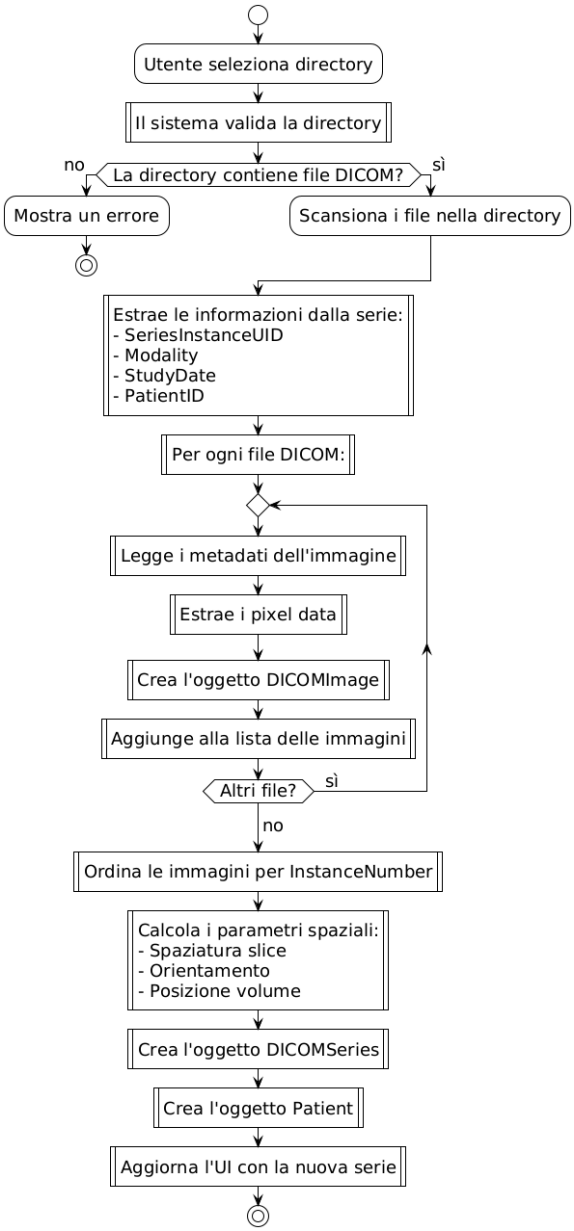

Figura 4.9: Diagramma delle attività per l'importazione `DICOM`.

Il processo di importazione dei dati `DICOM` (Figura 4.9) si articola in più fasi interconnesse. Inizialmente, il sistema valida l'integrità e la conformità dei file `DICOM`. A seguire, estrae dei metadati, ovvero informazioni che descrivono ogni immagine, come l'orientamento nello spazio, la distanza tra i frame e il tempo di acquisizione: elementi fondamentali per ricostruire correttamente la geometria del volume.Le immagini vengono quindi ordinate in base alla loro posizione spaziale all'interno del corpo del paziente,

utilizzando le informazioni di orientamento e distanza tra le slice. Questo permette di ricostruire la sequenza corretta e generare la serie volumetrica completa. Infine, il sistema costruisce il volume tridimensionale applicando tecniche di interpolazione, utili a compensare eventuali dati mancanti.

Ogni fase del processo implementa controlli specifici per garantire la qualità dei dati e fornire feedback appropriato all'utente in caso di anomalie o file non conformi agli standard `DICOM`. La gestione degli errori è progettata per essere robusta e informativa, permettendo al sistema di continuare l'elaborazione anche in presenza di file parzialmente corrotti.

Il processo di generazione dei modelli 3D (Figura 4.10) coinvolge la collaborazione tra i moduli *Core* e *Rendering*, con l'obiettivo di trasformare i dati volumetrici ottenuti dalle immagini `DICOM` delle TAC in modelli 3D visualizzabili. La pipeline inizia con un'analisi del volume per estrarre informazioni che guidano i parametri di ricostruzione. Successivamente, viene applicato un algoritmo di estrazione delle superfici (come il *Marching Cubes*) per generare la geometria 3D del modello. Seguono fasi di post-elaborazione, come il filtraggio e la semplificazione della mesh, per migliorarne l'aspetto visivo e l'efficienza computazionale.

L'intero flusso sfrutta tecniche di elaborazione parallela per ridurre significativamente i tempi di calcolo, mantenendo la responsività dell'interfaccia utente anche durante l'elaborazione di modelli complessi. La parallelizzazione è implementata a diversi livelli per ottimizzare l'utilizzo delle risorse hardware disponibili.

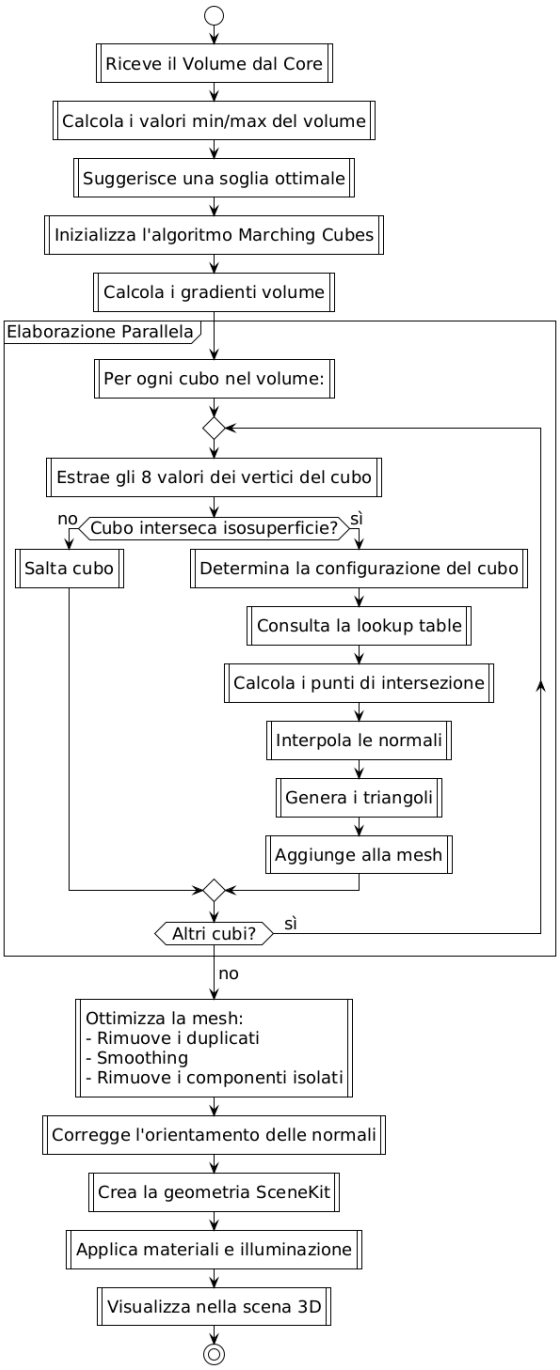

Figura 4.10: Processo di generazione mesh 3D tramite algoritmo Marching Cubes.

4.2.3 Interazione

L'architettura del sistema è progettata per garantire **comunicazioni efficienti** tra i moduli, seguendo pattern consolidati per la gestione delle dipendenze e la propagazione degli eventi.

La Figura 4.11 illustra come un caso d'uso completo attraversi tutti i livelli architetturali, evidenziando la distribuzione delle responsabilità secondo la gerarchia stabilita. Il flusso di controllo ha origine dal livello *UI* in ri-

sposta all'azione dell'utente, si propaga attraverso i livelli intermedi per l'elaborazione dei dati e ritorna con i risultati per la visualizzazione finale.

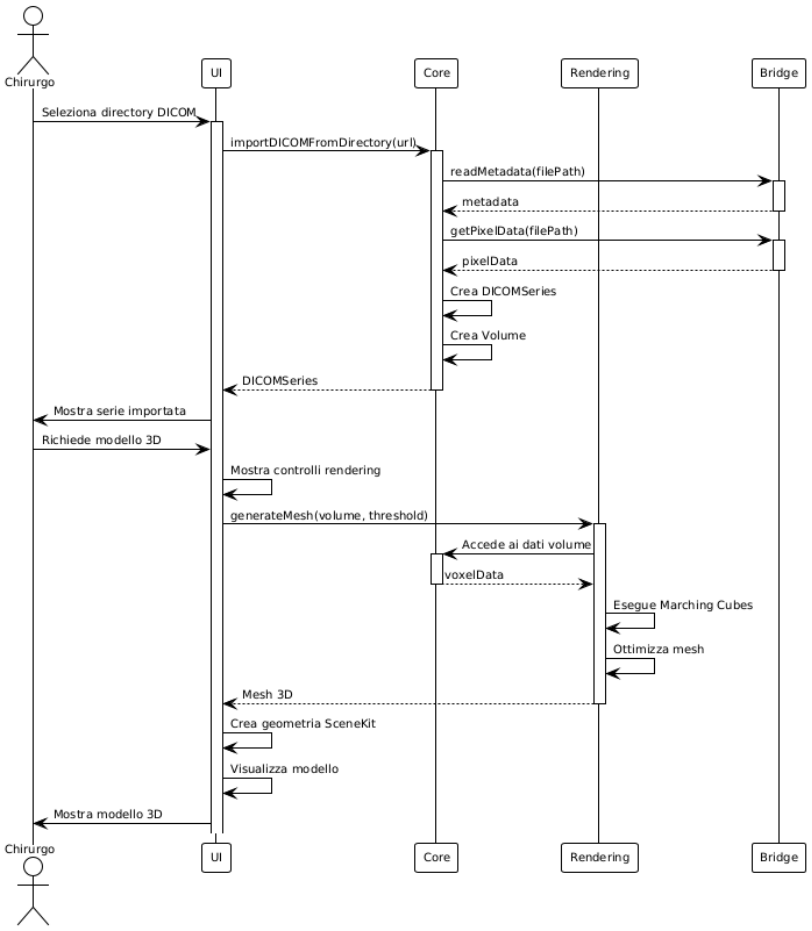

Figura 4.11: Diagramma di sequenza per la generazione di un modello 3D.

La gestione delle interfacce tra moduli è progettata per minimizzare l'accoppiamento, con ogni componente che espone interfacce essenziali che **nascondono la complessità interna**. Questo approccio permette l'evoluzione indipendente delle implementazioni *senza generare impatti sui moduli dipendenti*.

Il sistema implementa strategie di **gestione degli errori** appropriate per ogni livello architetturale, garantendo robustezza anche in presenza di dati non conformi o condizioni anomale. La propagazione degli errori segue percorsi ben definiti che permettono una gestione granulare delle diverse tipologie di problemi.

Il sistema utilizza il pattern *Dependency Injection* per gestire le dipendenze tra moduli, in modo da facilitare le attività di testing e permettere una configurazione flessibile delle implementazioni. Questo approccio migliora significativamente la testabilità del sistema e semplifica l'integrazione di nuove funzionalità. Inoltre un sistema di notifiche asincrone basato su eventi permette di comunicare informazioni significative tra moduli senza creare

accoppiamenti diretti.

L'architettura supporta operazioni a lunga durata attraverso meccanismi di elaborazione asincrona. Questi meccanismi sono particolarmente importanti per mantenere un'esperienza utente fluida durante le operazioni computazionalmente intensive come la ricostruzione 3D e l'elaborazione di volumi di grandi dimensioni.

4.3 Design dell'app per visionOS

4.3.1 Struttura

Anche il sistema *visionOS* si basa su un'architettura modulare a quattro componenti, pensata per garantire chiarezza strutturale, isolamento delle responsabilità e adattamento efficiente all'ambiente immersivo:

- **Core**: gestisce lo stato centralizzato dell'app e coordina il ciclo di vita dello spazio immersivo, fungendo da punto di interconnessione tra i moduli.
- **UI**: definisce le viste e le modalità di interazione in 2D e 3D, mantenendo separazione tra logica dell'interfaccia e logica applicativa.
- **Utility**: fornisce servizi trasversali legati al caricamento e alla preparazione dei modelli 3D, garantendo astrazione rispetto ai dettagli tecnologici.
- **Service**: incapsula la logica specializzata per l'interazione immersiva e l'analisi geometrica, mantenendola separata dalla gestione dello stato e dalla *UI*.

La Figura 4.12 evidenzia come il modulo *Core* funga da coordinatore centrale, con *UI* che dipende da esso per lo stato dell'applicazione e utilizza servizi da *Service* per le funzionalità immersive. Il modulo *Utility* fornisce servizi di supporto trasversali, particolarmente critici per la conversione dei modelli 3D necessaria all'esperienza immersiva.

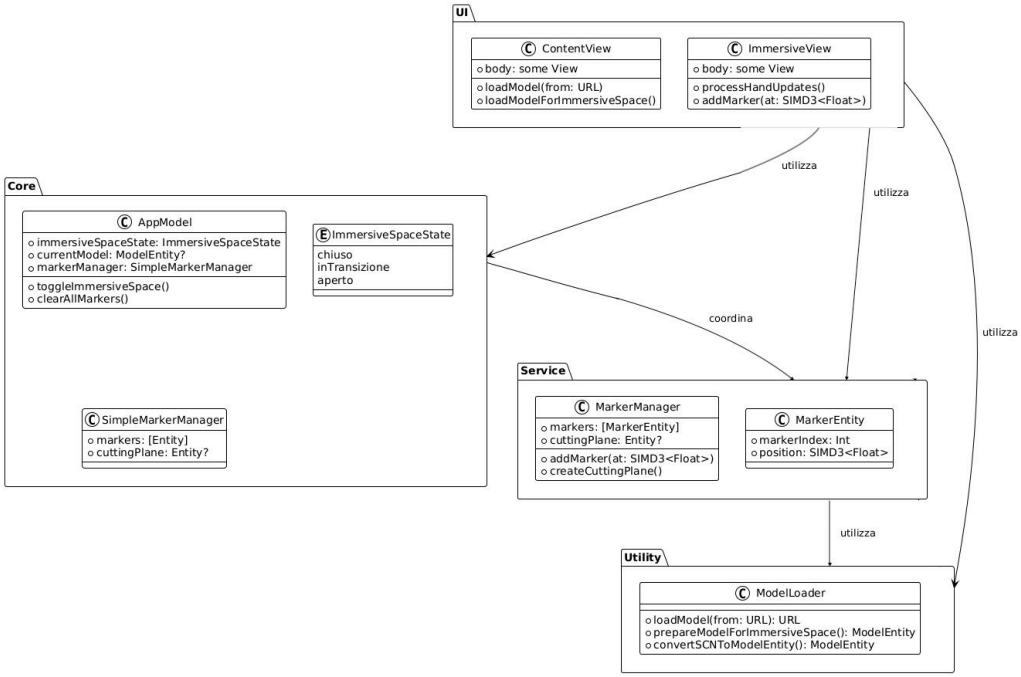

Figura 4.12: Diagramma delle dipendenze tra i moduli *visionOS*.

L'architettura è ottimizzata per le caratteristiche uniche della piattaforma *visionOS* e si basa su principi fondamentali pensati appositamente per la realtà mista:

- **Gestione dello stato reattivo**: il sistema utilizza il pattern `@Observable` di *SwiftUI* per garantire la sincronizzazione automatica tra lo stato dell'applicazione e il *rendering* immersivo.
- **Separazione delle modalità di interazione**: l'architettura distingue chiaramente tra logica per l'interfaccia tradizionale 2D e interazione spaziale 3D, consentendo ottimizzazioni specifiche per ciascuna modalità senza interferenze.
- **Gestione efficiente delle risorse**: tenendo conto delle limitazioni hardware di *Apple Vision Pro*, ogni modulo adotta strategie mirate per ottimizzare memoria e prestazioni durante operazioni immersive ad alta intensità.

4.3.1.1 Modulo Core

Il modulo *Core* rappresenta il coordinatore centrale del sistema *visionOS*, responsabile della gestione dello stato dell'applicazione e dell'orchestrazione delle operazioni immersive. La progettazione segue i principi di *reactive programming* di *SwiftUI* con particolare attenzione all'integrazione nativa con *visionOS*.

Il modulo si concentra sulla gestione centralizzata dello stato attraverso l'`AppModel`, che coordina tutte le operazioni dell'applicazione e mantiene la coerenza durante le transizioni tra modalità 2D e immersiva.

`ImmersiveSpaceState` regola il ciclo di vita dello spazio immersivo, con particolare attenzione alla fluidità delle transizioni e all'allocazione ottimale delle risorse tra le modalità operative.

La Figura 4.13 mostra la separazione tra gestione dello stato globale e coordinamento dei servizi specializzati. Il `SimpleMarkerManager` fornisce un'interfaccia coordinata per le operazioni base sui marker, delegando la logica complessa al modulo *Service* e mantenendo l'architettura pulita.

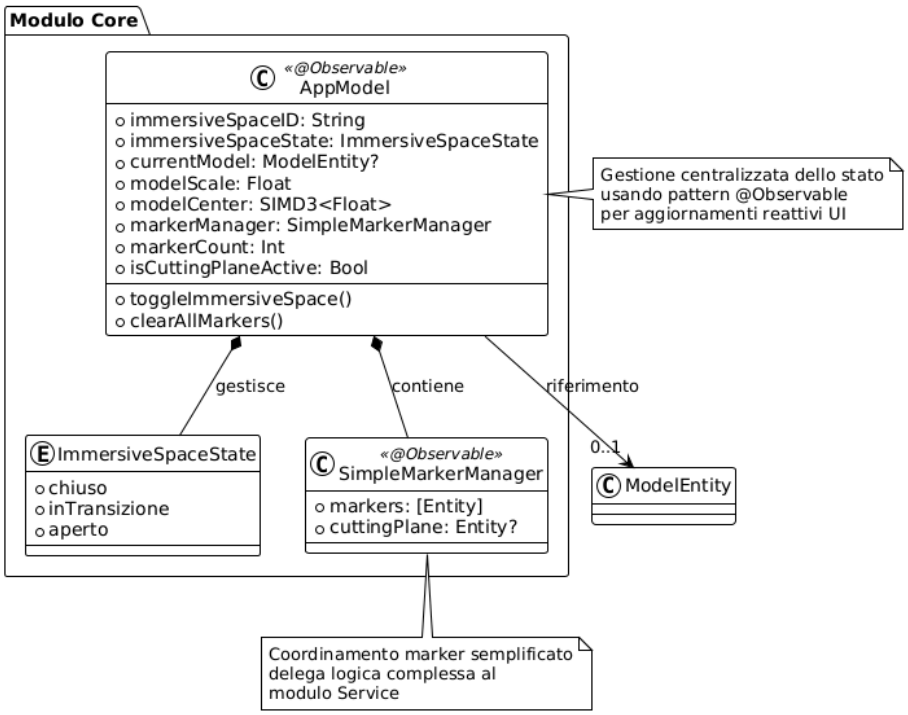

Figura 4.13: Diagramma delle classi del modulo Core per *visionOS*.

Un aspetto centrale del design è la gestione efficiente delle risorse specifiche per *visionOS*. Il modulo implementa strategie per l'ottimizzazione della memoria, tenendo conto delle limitazioni hardware di *Apple Vision Pro*, e gestisce automaticamente il ciclo di vita delle entity 3D nelle transizioni tra modalità operative.

4.3.1.2 Modulo UI

Il modulo *UI* coordina l'interazione tra l'interfaccia 2D tradizionale e l'ambiente immersivo. Al suo interno, `ContentView` gestisce la visualizzazione convenzionale, mentre `ImmersiveView` costituisce il fulcro della modalità mista, sfruttando *RealityView* per il *rendering* di modelli 3D con materiali olografici ottimizzati per *visionOS*.

La gestione dei controlli gestuali è una responsabilità centrale del modulo. L'`ImmersiveView` sfrutta l'*hand tracking* nativo per riconoscere i gesti *pinch*, con un sistema di validazione temporale che evita attivazioni accidentali. Il rilevamento distingue in modo affidabile tra interazioni con

l'interfaccia e movimenti nello spazio 3D.

La Figura 4.14 mostra la suddivisione in componenti dedicati, ciascuno responsabile della gestione delle transizioni tra modalità 2D e immersive. I controlli interattivi si interfacciano direttamente con il modulo *Core*, coordinando l'apertura dello spazio immersivo e aggiornando in tempo reale lo stato dei marker e dei piani di taglio.

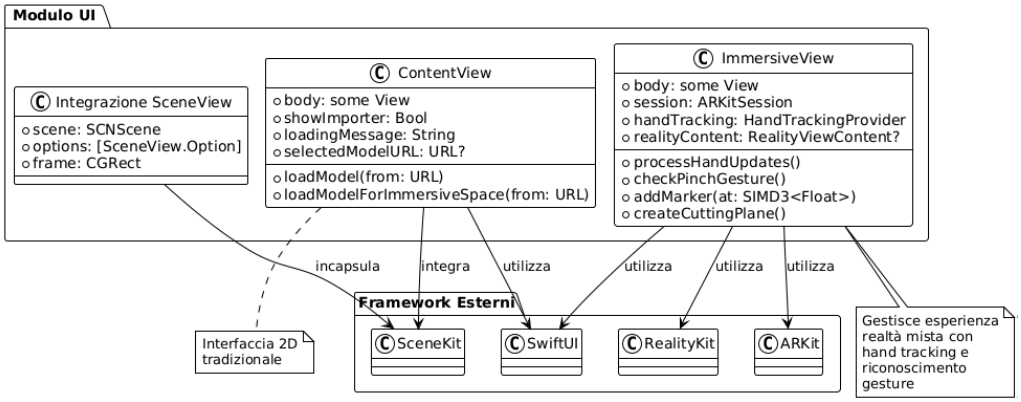

Figura 4.14: Diagramma delle classi del modulo *UI* per *visionOS*.

4.3.1.3 Modulo Utility

Il modulo *Utility* si specializza nel caricamento e conversione di modelli 3D, fornendo servizi essenziali ottimizzati per *visionOS* e i formati supportati da `RealityKit`.

Tra le responsabilità principali del modulo *Utility* vi è la gestione del caricamento e della preparazione dei modelli 3D per l'ambiente immersivo. Il sistema accede in modo sicuro ai file tramite *security-scoped resources* e converte automaticamente i formati compatibili, adattando scala e posizionamento per una visualizzazione corretta in realtà mista.

La conversione dei formati rappresenta un aspetto critico: i file `.usdz` sono supportati nativamente, mentre i file `.scn` vengono processati tramite un convertitore dedicato, in grado di mantenere geometrie, materiali e trasformazioni con precisione.

Il modulo fornisce strumenti dedicati alla gestione del file system, con particolare attenzione all'accesso sicuro ai file esterni. Supporta completamente le *security-scoped resources*, un meccanismo che consente all'applicazione di accedere, in modo controllato, solo ai file esplicitamente selezionati dall'utente. Questa esigenza nasce dal funzionamento dell'ambiente sandboxato, in cui ogni applicazione opera in uno spazio isolato per motivi di sicurezza. La logica di gestione dei permessi è confinata in questo modulo, così da mantenere semplice e sicura l'interazione con il file system nel resto dell'architettura.

4.3.1.4 Modulo Service

Il modulo *Service* centralizza la logica di interazione immersiva e l'elaborazione geometrica, separandola dalle responsabilità di interfaccia e *rendering*. La sua progettazione riflette un'architettura orientata ai servizi, in cui ogni componente espone funzionalità ben definite e facilmente componibili.

Elemento centrale è il `MarkerManager`, progettato per gestire in modo indipendente il posizionamento dei marker nello spazio 3D e la definizione dei piani di taglio. Il suo comportamento segue un **modello stateless a eventi**, in cui ogni azione dell'utente (come un gesto riconosciuto o un'interazione spaziale) viene trattata come un **evento discreto**, ovvero come *un'unità indipendente*, con un inizio e una fine chiari. Questo favorisce la prevedibilità del sistema, facilita il testing e rende più semplice integrare o sostituire componenti.

Il modulo include inoltre logiche di validazione geometrica: i punti vengono controllati per evitare configurazioni non valide, come la *collinearità*, ovvero il caso in cui i tre marker si trovano allineati lungo una stessa retta. In queste condizioni non è possibile definire un piano univoco. Una volta validata la configurazione, il sistema calcola automaticamente la normale del piano e genera una rappresentazione visuale nello spazio. Per queste operazioni geometriche, il sistema sfrutta librerie basate su **SIMD (Single Instruction, Multiple Data)**, che permettono di eseguire calcoli vettoriali in parallelo. Questo approccio migliora significativamente le prestazioni durante operazioni intensive come la computazione di normali o distanze tra punti, mantenendo la reattività dell'interfaccia anche in ambienti immersivi.

4.3.2 Comportamento

Il comportamento dell'applicazione riflette come i diversi moduli collaborino in fase di esecuzione. Due scenari rappresentano in modo chiaro questa cooperazione: il caricamento dei modelli 3D e l'interazione immersiva per la definizione dei piani di taglio.

Nel primo caso, l'utente avvia il processo di **importazione** tramite l'interfaccia, sfruttando un componente esposto dal modulo *UI*. A seguito della selezione di un file, il modulo *Core* si occupa del coordinamento, delegando l'intera gestione tecnica al modulo *Utility*. Quest'ultimo provvede alla validazione del formato, al trasferimento del file in una locazione sicura (in linea con le restrizioni imposte dall'ambiente *sandboxato*) e infine alla preparazione del contenuto per l'ambiente immersivo. A seconda del tipo di file, il sistema attiva un caricamento diretto oppure una conversione verso un formato compatibile con il motore di *rendering*.

Durante questa fase vengono calcolate anche informazioni come la scala e il posizionamento iniziale del modello nello spazio 3D. Una volta terminata l'elaborazione, il modello viene reso disponibile nell'ambiente immersivo e **la scena si aggiorna automaticamente**. Questo flusso mostra una netta

separazione delle responsabilità tra moduli e una coordinazione basata su eventi che semplifica l'estensione del sistema. Il comportamento è illustrato in Figura 4.15.

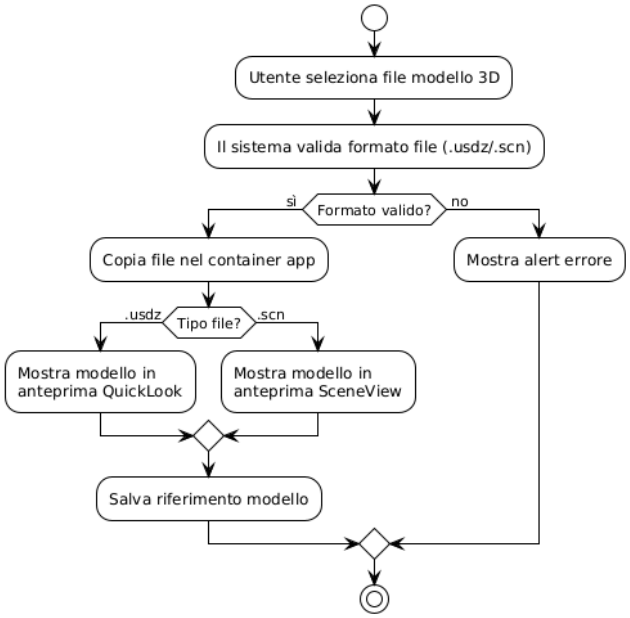

Figura 4.15: Diagramma delle attività per il caricamento dei modelli 3D.

Il secondo scenario riguarda l'interazione immersiva per la **definizione dei piani di taglio**. In questo contesto, l'utente può posizionare marker nello spazio tridimensionale tramite un gesto naturale come il pinch tra pollice e indice. Il modulo *UI* rileva il gesto attraverso l'*hand tracking* e lo trasmette come evento al *Core*, che aggiorna lo stato globale e invia la richiesta al modulo *Service*, responsabile della logica spaziale.

All'interno di *Service*, il componente `MarkerManager` gestisce la creazione, la rotazione e la validazione dei marker. Quando vengono posizionati tre marker, il sistema verifica che **non siano collineari** — ovvero disposti lungo una stessa retta — poiché questa configurazione non permetterebbe di definire un piano. Se la configurazione è valida, viene calcolata la normale al piano tramite operazioni vettoriali ad alta precisione, e viene generata una rappresentazione visiva nella scena immersiva.

Questo comportamento segue un modello a **eventi discreti**, dove ogni interazione dell'utente attiva una sequenza ben delimitata di reazioni nei moduli coinvolti. L'intero processo è rappresentato in Figura 4.16.

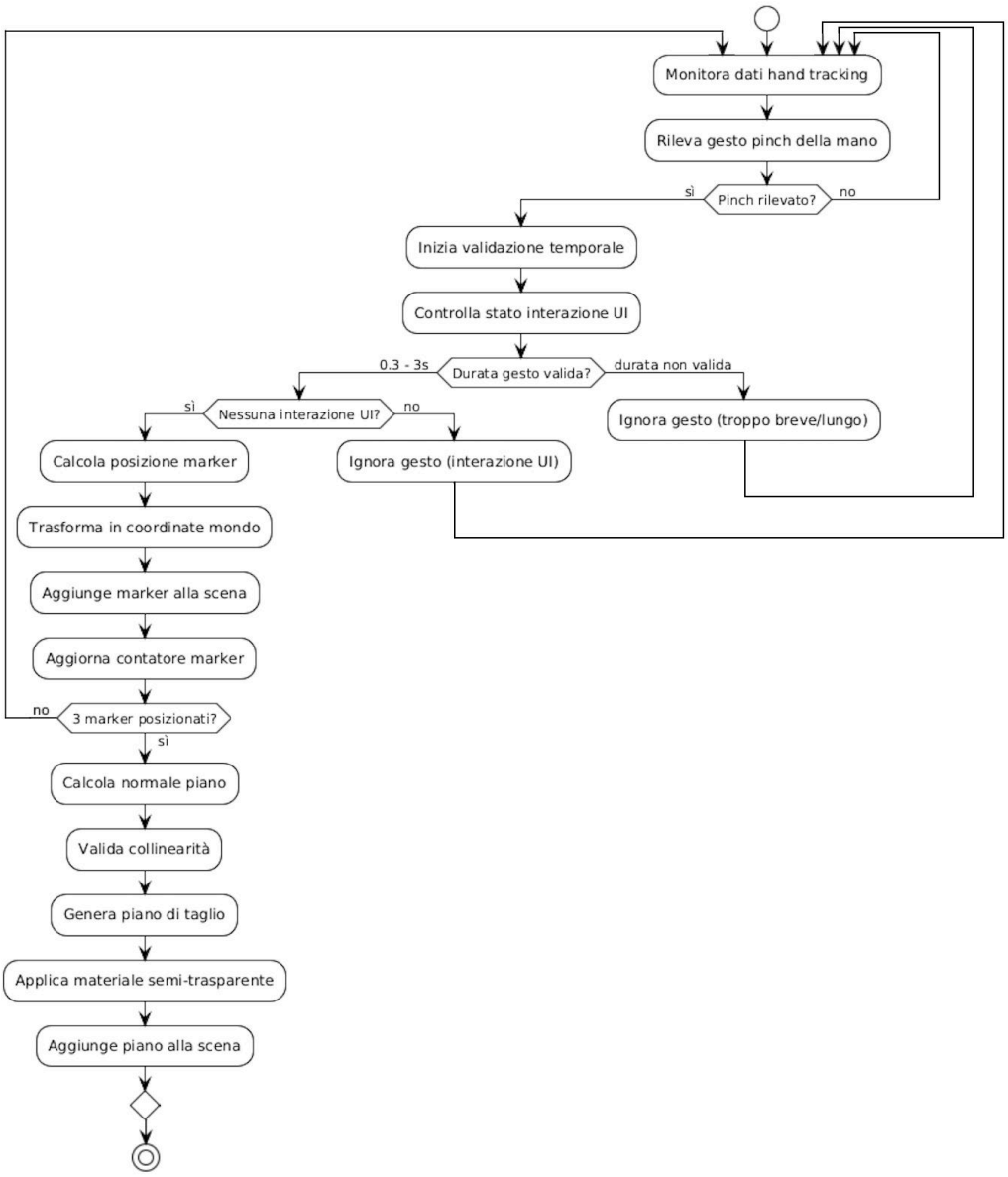

Figura 4.16: Processo di gestione marker e generazione piano di taglio.

4.4 Design UX/UI

Il design dell'interfaccia utente ha ricoperto un ruolo cruciale nella progettazione del sistema, data la necessità di integrarsi efficacemente in un flusso di lavoro clinico reale. Sono state adottate soluzioni grafiche e funzionali che puntano alla **semplicità**, alla **chiarezza visiva** e alla **compatibilità con ambienti chirurgici** ad alta criticità. L'interfaccia è stata concepita con l'obiettivo di ridurre al minimo le distrazioni e massimizzare l'efficacia operativa, nel pieno rispetto dei principi *KISS* (Keep It Simple, Stupid), *DRY* (Don't Repeat Yourself) e *Less is More*.

L'applicazione per *macOS* offre strumenti di pianificazione precisi e accessibili, mentre quella per *visionOS* privilegia un'interazione leggera e discreta, pensata per ambienti ad alta intensità come la sala operatoria. In entrambi

i casi, l'attenzione è stata rivolta alla costruzione di un'interfaccia immediata e affidabile, utilizzabile anche da personale con esperienza tecnologica limitata.
Nel contesto operatorio reale, l'uso delle interfacce richiede particolare attenzione:

- **Compatibilità con i guanti chirurgici:** sono stati eliminati tutti i controlli *touch* tradizionali. I gesti supportati dal visore non richiedono contatto fisico e risultano naturali anche con i guanti chirurgici.

- **Nessuna tastiera o input testuale:** per evitare distrazioni e complessità inutili, non esistono campi di testo o menu complessi.

- **Gestione del carico cognitivo:** ogni elemento è progettato per comunicare una singola funzione, con visual feedback chiaro ed essenziale. Questo consente ai chirurghi di mantenere il focus sull'operazione, evitando overload visivi.

4.4.1 UX dell'applicazione per macOS

L'interfaccia grafica dell'applicazione è progettata per offrire un'esperienza visiva chiara e funzionale, con strumenti dedicati alla manipolazione dei modelli 3D e alla definizione dei piani di taglio in modo preciso e intuitivo. Dal punto di vista estetico, lo stile segue le *Human Interface Guidelines* di *Apple*, con una **palette colori** sobria e ad alto contrasto. Toni scuri e neutri dominano l'interfaccia, permettendo alle superfici ossee e ai marker di risaltare in primo piano e favorendo la concentrazione visiva sui contenuti clinici rilevanti.

L'organizzazione e il *layout* dell'interfaccia seguono una struttura **lineare e pulita**, dove sono presenti pochi controlli, tutti essenziali, posizionati lateralmente o in pannelli contestuali. Il focus principale è concentrato sull'area centrale occupata dal **rendering 3D interattivo**, che permette di ruotare, ingrandire e manipolare la scena in modo fluido, garantendo al chirurgo un controllo preciso e intuitivo del modello anatomico.

Particolare attenzione è stata dedicata all'**accessibilità**: l'interfaccia supporta *font* scalabili, contrasti elevati per utenti con deficit visivi e icone semanticamente chiare.

Un aspetto fondamentale è rappresentato dal **feedback in tempo reale**: ogni interazione, come la selezione o la modifica dei piani, genera un aggiornamento immediato nella scena, grazie all'uso di `@Published` e `ObservableObject`. Questo meccanismo consente al chirurgo di verificare istantaneamente l'impatto delle proprie azioni e di procedere con maggiore sicurezza nella pianificazione. Infine, la **responsività** dell'interfaccia garantisce un adattamento dinamico alle dimensioni della finestra, assicurando sempre un'esperienza utente ottimale.

4.4.2 UX dell'applicazione per visionOS

L'applicazione intraoperatoria è stata progettata con una filosofia totalmente diversa, focalizzata sulla massima **essenzialità** e **non intrusività** durante l'intervento chirurgico. L'obiettivo principale è fornire uno *strumento di supporto* efficace senza introdurre complessità che potrebbero interferire con la procedura chirurgica.

Il **design olografico** rappresenta il cuore dell'approccio visivo: i modelli 3D e i piani di taglio sono rappresentati come *ologrammi traslucidi* con contorni netti, facilmente distinguibili dal reale ma sufficientemente integrati nell'ambiente per non disturbare la vista del chirurgo. Questa scelta progettuale permette di sovrapporre informazioni digitali critiche al campo operatorio **senza compromettere la percezione dell'anatomia reale**.

I controlli dell'applicazione si basano esclusivamente su *gesture* naturali: il *pinch* per le manipolazioni e lo sguardo per il *targeting* rappresentano modalità di interazione che evitano qualsiasi necessità di tocco fisico. Questo aspetto risulta fondamentale in **ambienti sterili**, dove il mantenimento delle condizioni asettiche è cruciale per il successo dell'intervento.

Il **minimalismo visivo** caratterizza profondamente l'interfaccia, che è completamente priva di pannelli, menu o controlli testuali tradizionali. Ogni elemento è collocato direttamente nello spazio tridimensionale e risponde in modo intuitivo al comportamento dell'utente, creando un'esperienza immersiva che non distrae dalla procedura principale.

Il **contrasto e la leggibilità** sono stati ottimizzati attraverso l'utilizzo di colori ad alta visibilità e bordi evidenziati che rendono immediatamente riconoscibili i piani di taglio rispetto all'anatomia circostante. La scelta cromatica è stata calibrata per risultare visibile sia su fondo chiaro che scuro, garantendo la massima efficacia in diverse condizioni di illuminazione operatoria.

La gestione della **profondità spaziale** rappresenta un elemento distintivo dell'interfaccia: gli elementi *UI* sono disposti con particolare attenzione alla profondità e alla distanza di focalizzazione, evitando sovrapposizioni problematiche con il campo operatorio reale e facilitando l'adattamento visivo del chirurgo tra il mondo digitale e quello fisico.

Capitolo 5

Dettagli implementativi

Dopo aver definito l'architettura e il design del sistema, il presente capitolo si concentra sui dettagli implementativi del prototipo funzionante. Vengono descritte le tecnologie e i *framework* utilizzati per realizzare entrambe le applicazioni. Il capitolo illustra il processo completo dal caricamento delle immagini TAC alla visualizzazione immersiva, includendo le funzionalità di manipolazione gestuale e posizionamento marker in realtà mista. Ogni sezione fornisce esempi di codice e dettagli tecnici necessari per comprendere l'implementazione concreta delle funzionalità progettate.

5.1 Tecnologie utilizzate

L'implementazione del sistema bimodale *macOS-visionOS* ha richiesto l'integrazione di diverse tecnologie. La scelta è stata guidata da criteri di compatibilità con l'ecosistema *Apple*, prestazioni nell'elaborazione di dati medicali e supporto nativo per la realtà mista. Il sistema si basa su un'architettura che privilegia l'uso di *framework* nativi *Apple* per garantire ottimizzazioni hardware e integrazione *seamless* tra le piattaforme.

5.1.1 Tecnologie comuni alle due applicazioni

Le tecnologie condivise tra *macOS* e *visionOS* forniscono la base comune per lo sviluppo del sistema, in modo da garantire coerenza nell'architettura e nel codice.

Swift è il linguaggio principale usato per entrambe le applicazioni. La scelta di `Swift` è stata naturale per sviluppare app native su dispositivi Apple, con buone prestazioni e integrazione con i *framework* di sistema [13].

SwiftUI è il *framework* dichiarativo usato per costruire le interfacce utente. Consente di creare *UI* moderne e reattive con poco codice, supportando funzionalità come *Dark Mode*, localizzazione e accessibilità. Il pattern MVVM è integrato in modo naturale grazie a decoratori come `@Observable` e `@State`, facilitando la separazione tra logica e interfaccia [13].

5.1.2 Stack tecnologico per l'applicazione macOS

L'applicazione desktop richiede tecnologie specifiche per l'elaborazione intensiva di dati `DICOM` e la visualizzazione 3D ad alte prestazioni.

DCMTK (`DICOM` Toolkit) è la libreria C++ usata per leggere e gestire i file . È uno standard ampiamente adottato nel campo medico, noto per la sua affidabilità e conformità [16, 17]. L'integrazione con `Swift` avviene tramite un *bridge Objective-C++* che semplifica l'uso della libreria nel resto del sistema.

Metal Performance Shaders accelerano calcoli complessi, sfruttando la GPU per elaborare più rapidamente grandi volumi [18, 19].

SceneKit è il *framework* 3D usato per visualizzare modelli tridimensionali. Supporta illuminazione, animazioni e formati standard come `.scn`, integrandosi con `Metal` per sfruttare l'hardware *Apple*. È stato scelto per la sua stabilità e ottimizzazione su *macOS* [20].

Core Graphics e **Core Image** vengono usati per elaborare immagini, con operazioni come *windowing*, conversioni e filtri. Offrono buone prestazioni grazie all'accelerazione hardware dei *Mac* [13].

5.1.3 Stack tecnologico per l'applicazione visionOS

L'applicazione per realtà mista utilizza le tecnologie più avanzate di *Apple* per lo *spatial computing* e l'interazione naturale.

RealityKit è il *framework AR* usato come motore di *rendering* principale su *visionOS*. Ottimizzato per *AR/VR*, offre funzioni avanzate come illuminazione realistica, occlusione e ancoraggio spaziale, gestendo in automatico molti aspetti complessi del *rendering* [13].

ARKit fornisce tracciamento delle mani tramite algoritmi di *computer vision* e *machine learning*, consentendo l'uso di gesti naturali come il pinch [11, 13].

SIMD (Single Instruction, Multiple Data) è impiegato per eseguire calcoli vettoriali ad alte prestazioni, come trasformazioni geometriche e operazioni di posizionamento nello spazio 3D, sfruttando l'elaborazione parallela offerta dal *chip M2* [10].

Spatial Computing APIs di *visionOS* permettono di creare e gestire ambienti immersivi, mantenere ancoraggi stabili nello spazio reale e controllare il comportamento dell'app in scenari di realtà mista [11, 13].

5.2 Importazione e visualizzazione delle TAC

Questa sezione descrive il processo di importazione, organizzazione e visualizzazione delle immagini TAC all'interno dell'applicazione *macOS*.

5.2.1 Gestione dei file DICOM

L'importazione dei dati rappresenta il primo passo fondamentale del workflow chirurgico. L'utente seleziona semplicemente una cartella contenente i file della TAC attraverso un *file picker* nativo e l'applicazione si occupa automaticamente di tutto il resto.

L'interfaccia, in coordinamento con i servizi offerti dal *modulo Core dell'app macOS*, fornisce *feedback* visivo immediato durante l'importazione. Il sistema esegue automaticamente una serie di controlli di qualità che verificano l'integrità dei file e la coerenza della serie di acquisizione. Questi controlli sono fondamentali per garantire che la successiva ricostruzione 3D sia accurata e affidabile.

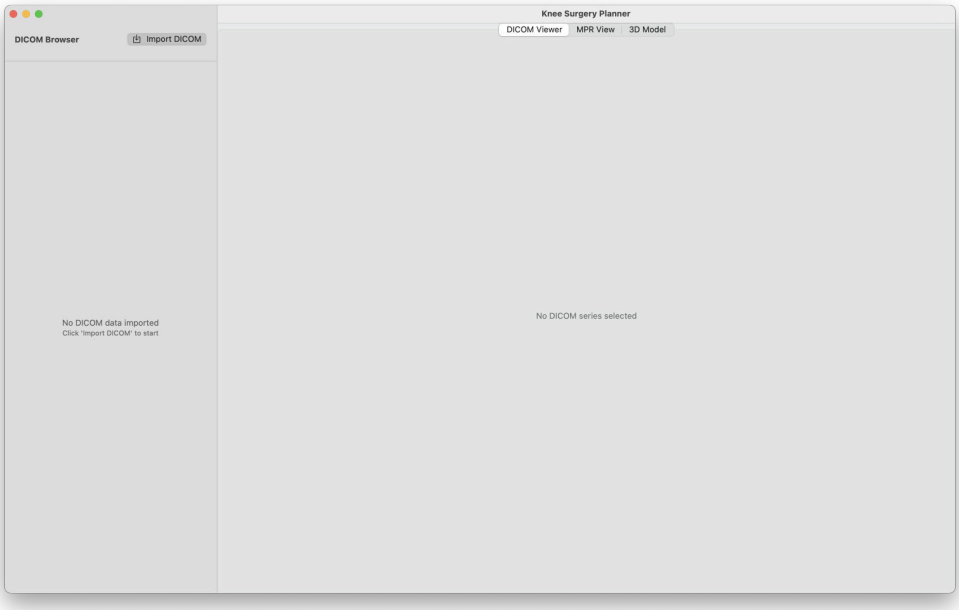

Figura 5.1: Interfaccia iniziale con importazione .

Durante l'importazione, il sistema estrae i **metadati** dai file `DICOM`. Si tratta di informazioni descrittive incorporate nei file, che comprendono ad esempio: il nominativo del paziente (in forma anonimizzata per garantire la privacy), la risoluzione dell'immagine, la distanza tra i piani di scansione, il numero di slice, la posizione spaziale di ciascuna immagine e il tipo di acquisizione effettuata. Questi dati sono fondamentali per ricostruire correttamente il volume 3D e mantenere l'allineamento tra le immagini in base alla loro posizione reale nel corpo.

I dati vengono poi organizzati secondo la gerarchia *standard* dei file `DICOM` (`Patient` → `Study` → `Series` → `Image`), consentendo di gestire anche casi

complessi con acquisizioni multiple o parziali.

L'implementazione utilizza un sistema di caricamento efficiente: i metadati vengono letti e organizzati subito, così da consentire una navigazione veloce e fluida tra le immagini. I dati delle immagini vere e proprie, invece, vengono caricati solo quando servono davvero, evitando sprechi di memoria. Questo approccio riduce i tempi di attesa e rende l'interazione più rapida e piacevole per l'utente.

File: DICOMManager.swift Modulo: Core App: macOS

```swift
func importDICOMFromDirectory(_ url: URL) async throws -> DICOMSeries {
    // Recupera e filtra i file DICOM dalla cartella
    let fileURLs = try FileManager.default.contentsOfDirectory(at: url,
        includingPropertiesForKeys: nil)
    let dicomFiles = fileURLs.filter { $0.pathExtension.lowercased() == "dcm" }
    let sortedDicomFiles = dicomFiles.sorted { $0.lastPathComponent < $1.
        lastPathComponent }

    // Organizza secondo gerarchia Patient     Study     Series     Image
    return try await readDICOMFiles(from: sortedDicomFiles)
}

private func readDICOMFiles(from files: [URL]) async throws -> DICOMSeries {
    // Estrae metadati immediatamente per navigazione fluida
    let metadata = dicomService.readMetadata(from: firstFile.path)

    // Crea struttura organizzata
    var series = DICOMSeries(seriesInstanceUID: seriesInstanceUID,
                             seriesDescription: seriesDescription, ...)

    // Carica dati immagini solo quando necessario (lazy loading)
    for fileURL in files {
        let imageMetadata = dicomService.readMetadata(from: fileURL.path)
        // Pixel data caricati on-demand per ottimizzare memoria
    }

    return series
}
```

Listato 5.1: Importazione e organizzazione intelligente dei file DICOM

5.2.2 Visualizzazione 2D e ricostruzione multiplanare

Una volta importati i file `DICOM` — che contengono i dati acquisiti dalla TAC del paziente — il *modulo UI dell'app macOS* gestisce la visualizzazione, offrendo all'utente un'interfaccia intuitiva per esplorare le singole immagini (slice) della scansione. L'esperienza di navigazione è simile a quella dei software medici professionali utilizzati per l'analisi radiologica. La visualizzazione include il supporto al *windowing*, una tecnica che regola luminosità e contrasto in base al tipo di tessuto da osservare (es. osso, muscolo o aria), così da rendere più evidenti le strutture di interesse.

La Figura 5.2 mostra l'interfaccia dedicata alla visualizzazione delle immagini TAC, con i controlli di *windowing* ben visibili e facilmente accessibili. La navigazione tra le slice è stata progettata per risultare naturale e immediata: l'utente può scorrere utilizzando la rotella del *mouse*, i tasti direzionali della tastiera oppure interagire direttamente con lo *slider* per saltare rapidamente a una slice specifica. A ogni interazione, l'applicazione fornisce

un *feedback* visivo in tempo reale, permettendo un'esplorazione continua e intuitiva del volume anatomico.

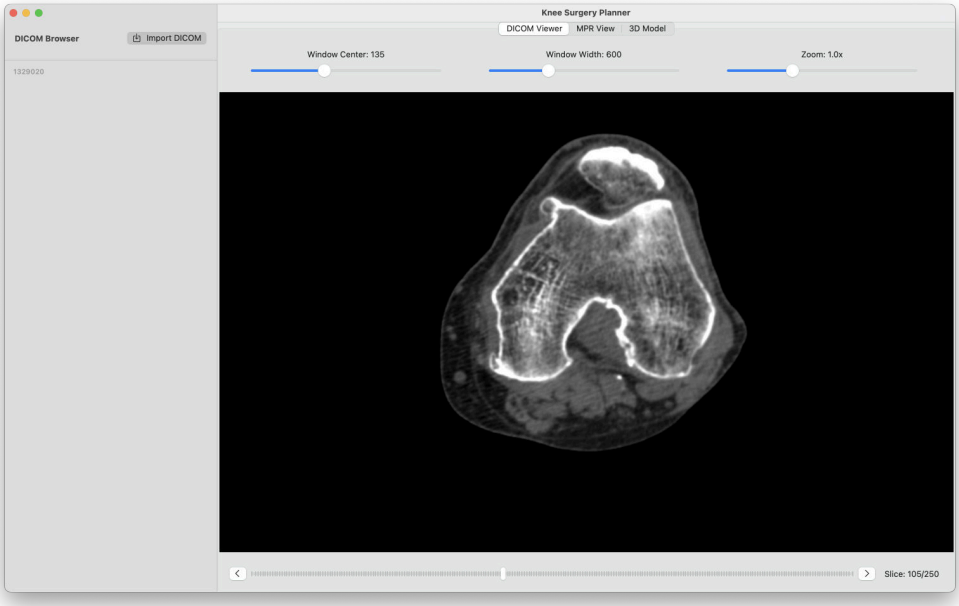

Figura 5.2: Visualizzatore di slice con controlli di windowing e preset anatomici.

Il sistema estende le funzionalità di esplorazione tramite la *ricostruzione multiplanare (MPR)*, una tecnica che consente di osservare il ginocchio da più angolazioni: **assiale** (dall'alto verso il basso), **coronale** (dal fronte verso il retro) e **sagittale** (da un lato all'altro).

Sebbene le immagini originali della TAC siano acquisite solo nel piano assiale, il sistema è in grado di ricostruire viste virtuali lungo gli altri due piani. Questo è possibile accedendo direttamente ai dati delle immagini importate: per ogni *pixel* da visualizzare nelle nuove viste, viene calcolata la posizione corrispondente nelle immagini originali, da cui viene estratto il valore di densità corretta.

File: MultiplanarView.swift Modulo: UI App: macOS

```swift
// Vista coronale (piano xz, y costante)
let y = sliceIndex
for z in 0..<height {
    for x in 0..<width {
        let volumeIndex = z * volDims.x * volDims.y + y * volDims.x + x

        if volumeIndex < int16Buffer.count {
            let pixelValue = volume.type == .ct ?
                Double(int16Buffer[volumeIndex]) * slope + intercept :
                Double(int16Buffer[volumeIndex])

            // Applica windowing per la visualizzazione
            adjustedData[outputIndex] = applyWindowing(value: pixelValue)
        }
    }
}
```

Listato 5.2: Generazione di slice multiplanari tramite accesso diretto

Questo metodo di accesso diretto ai dati consente di mantenere l'accuratezza delle informazioni acquisite dalla TAC e garantisce al tempo stesso alte prestazioni. Il sistema calcola in modo automatico la corrispondenza tra le coordinate delle diverse viste (assiale, coronale e sagittale), permettendo così una navigazione continua e reattiva all'interno del corpo del paziente.

Come si può notare in Figura 5.3, l'interfaccia multiplanare offre al chirurgo una visione dettagliata dell'anatomia **da più angolazioni**, mettendo a disposizione tutti gli strumenti necessari per analizzare a fondo la struttura ossea prima di passare alla ricostruzione 3D e alla pianificazione dei piani di taglio.

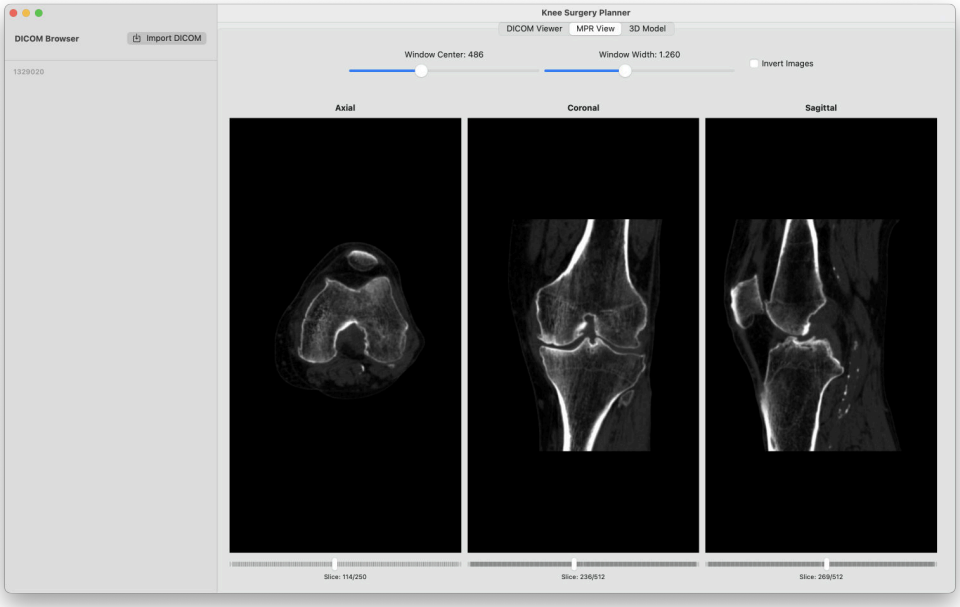

Figura 5.3: Vista multiplanare con sezioni assiale, sagittale e coronale.

5.3 Generazione e rendering del modello 3D

Questa sezione descrive il processo con cui l'applicazione trasforma le immagini TAC bidimensionali in un modello tridimensionale del ginocchio, utilizzando algoritmi di ricostruzione volumetrica e tecniche di *rendering* ottimizzate per l'ambito clinico.

5.3.1 Dal 2D al 3D: l'algoritmo Marching Cubes

Dopo l'importazione e l'organizzazione delle immagini acquisite con la TAC, il sistema ricostruisce un modello 3D del ginocchio utilizzando l'algoritmo *Marching Cubes* [19], implementato all'interno del *modulo Rendering* dell'applicazione *macOS*. Si tratta di una tecnica molto usata nella computer grafica medica per generare superfici tridimensionali a partire da un volume ricostruito internamente. In questo contesto, consente di visualizzare le strutture anatomiche — come ossa, tessuti o organi — sotto forma di

superfici 3D navigabili.

Il volume tridimensionale su cui lavora l'algoritmo è una rappresentazione interna costruita a partire dalle slice bidimensionali ottenute dalla TAC. Ogni slice mostra una sezione del ginocchio e viene posizionata correttamente nello spazio grazie ai metadati contenuti nei file `DICOM`, come l'orientamento e la distanza tra una slice e l'altra. Sulla base di queste informazioni, il sistema ricostruisce automaticamente un volume coerente che rappresenta l'interno del ginocchio.

L'algoritmo suddivide il volume in una struttura regolare tridimensionale, composta da cubi di dimensioni uniformi. Ogni cubo è caratterizzato da:

- **8 vertici (voxel)**: gli angoli del cubo, numerati da 0 a 7 (Figura 5.4);
- **12 spigoli (edge)**: le linee che collegano i vertici (Figura 5.5);
- **Un valore di intensità per ogni vertice**: espresso in *Hounsfield Units (HU)*, un'unità di misura standardizzata utilizzata in ambito medico per distinguere i diversi tessuti del corpo umano in base alla loro densità. Ad esempio, le ossa emergono con valori di intensità intorno a 300–400 HU, i tessuti molli tra 0 e 100 HU, mentre l'aria si evidenzia attorno a –1000 HU.

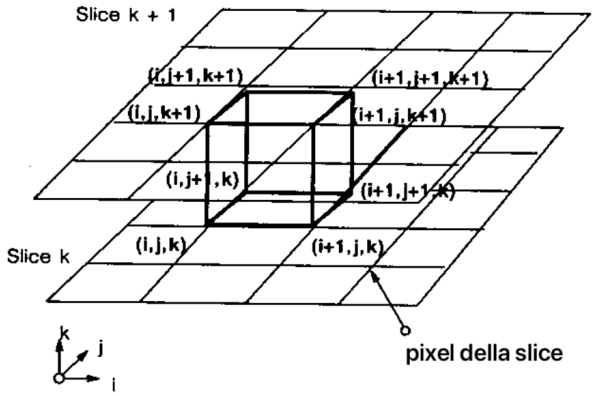

Figura 5.4: Griglia tridimensionale di voxel: ogni cubo è formato da 8 vertici con coordinate (i, j, k).

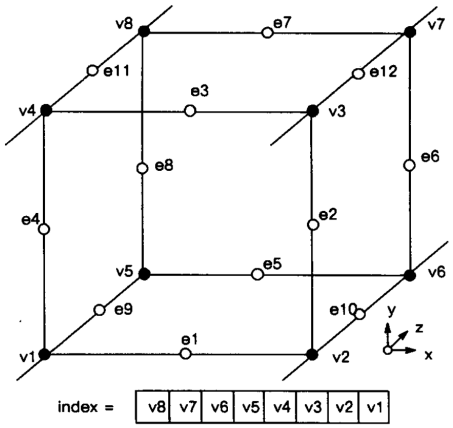

Figura 5.5: Numerazione dei vertici e degli edge di un cubo.

Di seguito sono illustrati i principali step dell'algoritmo:

1. **Scansione del volume**
   L'algoritmo esamina l'intero volume tridimensionale scorrendo un cubo alla volta. Per ciascun cubo, legge i valori di intensità (espressi in HU) associati ai suoi otto vertici.

2. **Classificazione binaria**
   I valori di intensità HU letti dai vertici vengono confrontati con un valore soglia prestabilito, chiamato *isovalue*, che dipende dal tipo di tessuto da visualizzare (ad esempio, l'osso). Per ogni vertice:

   - Se il valore è **maggiore o uguale** alla soglia, il punto è considerato interno alla struttura → **1**;
   - Se è **inferiore** alla soglia, è considerato esterno → **0**.

   Questo confronto produce un codice binario a 8 bit che rappresenta la configurazione del cubo, indicando quali vertici sono interni o esterni rispetto alla superficie.

   ```
   Soglia = 300 HU (per visualizzare osso)
   Vertice 0: 450 HU → 1 (osso)
   Vertice 1: 200 HU → 0 (tessuto molle)
   Vertice 2: 350 HU → 1 (osso)
   Vertice 3: 180 HU → 0 (tessuto molle)
   ... etc.

   Codice binario risultante: 1010...
   ```

   Questo processo è implementato nel modulo *Rendering* dell'app *macOS* nel seguente modo:

   File: MarchingCubes.swift Modulo: Rendering App: macOS

```swift
var cubeIndex = 0
for i in 0..<8 {
    if cubeValues[i] > isovalue {
        cubeIndex |= (1 << i)
    }
}

if MarchingCubesTables.edgeTable[cubeIndex] == 0 {
    continue
}

let triangleIndices = MarchingCubesTables.triTable[cubeIndex]
```

Listato 5.3: Classificazione binaria e lookup del cubo

3. **Lookup table - le 256 configurazioni**
   Ogni configurazione binaria può assumere $2^8 = 256$ possibili combinazioni. Tuttavia, molte di queste configurazioni sono **geometricamente equivalenti**: possono essere trasformate l'una nell'altra attraverso semplici rotazioni o riflessioni del cubo.

Per comprendere meglio questo concetto, si consideri un esempio illustrativo. La configurazione `00000001` (corrispondente al solo vertice 1 interno alla superficie) e la configurazione `01000000` (relativa al solo vertice 7 interno) producono triangolazioni geometricamente identiche, differenti unicamente per una rotazione spaziale (Figura 5.6). Dal punto di vista algoritmico, entrambe le configurazioni rappresentano il medesimo caso topologico: l'intersezione della superficie con un singolo vertice del cubo.

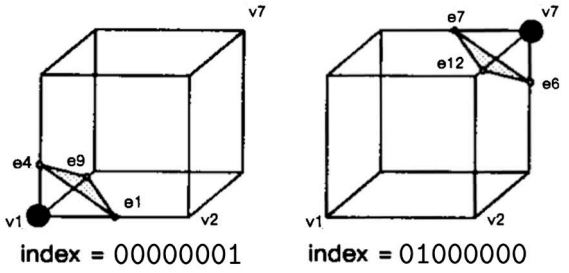

Figura 5.6: Esempi di configurazioni geometricamente equivalenti: ruotando il cubo, la configurazione A (`00000001`) si trasforma nella configurazione B (`01000000`), producendo la stessa topologia di triangoli.

Applicando sistematicamente tutte le possibili rotazioni e riflessioni del cubo, le 256 configurazioni teoriche si riducono a sole **15 configurazioni topologicamente uniche**. Questo significa che l'algoritmo deve gestire effettivamente solo 15 casi diversi, rendendo molto più efficiente l'implementazione della lookup table. (Figura 5.7).

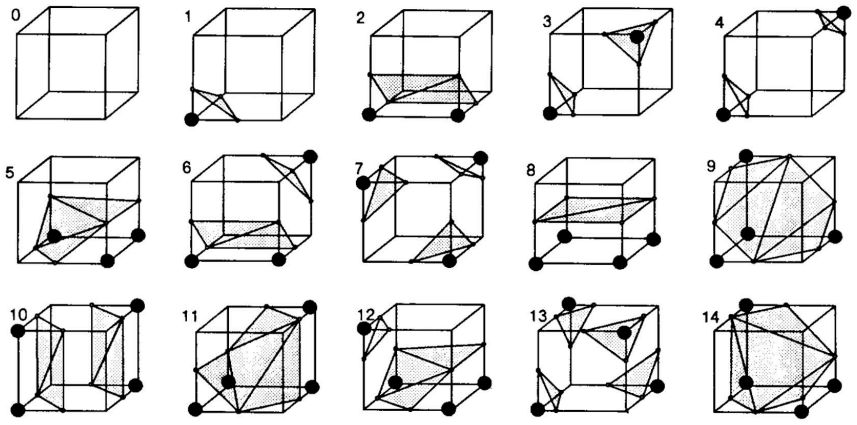

Figura 5.7: Le 15 configurazioni base della lookup table con triangoli generati nel cubo.

Per ciascuna di queste configurazioni, la lookup table fornisce due informazioni fondamentali: specifica quali spigoli del cubo sono attraversati dalla superficie e indica in che modo connettere i punti di intersezione per costruire i triangoli. Uno spigolo si considera attraversato quando i due vertici che lo definiscono si trovano da lati opposti rispetto alla soglia (uno sopra e uno sotto). In questo caso, si assume che la superficie passi in un punto intermedio dell'edge.

4. **Identificazione degli edge attraversati**
   Con la configurazione nota, l'algoritmo individua quali spigoli del cubo sono attraversati. Ogni edge è formato da due vertici, e se i rispettivi valori HU si trovano ai lati opposti della soglia, si considera attraversato dalla superficie.

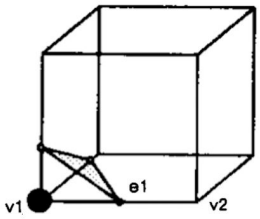

Figura 5.8: Identificazione degli edge attraversati: viene attraversato l'edge `e1`.

5. **Interpolazione lineare**
   Per ogni edge attraversato, viene calcolato il punto esatto in cui la superficie interseca l'edge tramite interpolazione lineare:

$$\mathbf{p} = \mathbf{p_1} + \frac{(iso - v_1)}{(v_2 - v_1)} \times (\mathbf{p_2} - \mathbf{p_1}) \tag{5.1}$$

   Dove:

   - $\mathbf{p_1}, \mathbf{p_2}$ = posizioni dei due vertici
   - v_1, v_2 = valori HU dei vertici
   - iso = soglia
   - $\mathbf{p}$ = punto di intersezione calcolato

6. **Generazione dei triangoli**
   I punti di intersezione vengono collegati secondo lo schema fornito dalla lookup table, formando da 0 a 5 triangoli per cubo. Ogni triangolo rappresenta una piccola porzione della superficie da ricostruire.

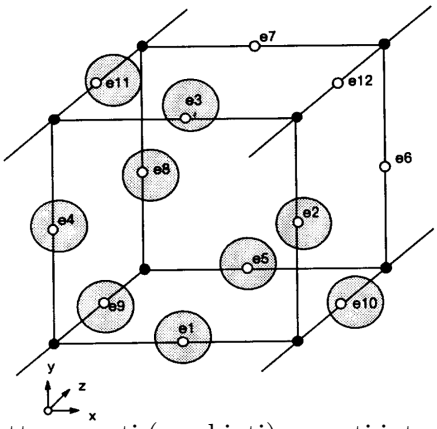

Figura 5.9: Edge attraversati (cerchiati) e punti interpolati uniti in triangoli.

7. **Costruzione della mesh completa**
   Ripetendo questi passaggi su tutti i cubi del volume, l'algoritmo costruisce una mesh triangolare continua che approssima fedelmente la superficie dell'anatomia (Figura 5.10). Il risultato è un modello 3D pronto per la visualizzazione e l'analisi clinica.

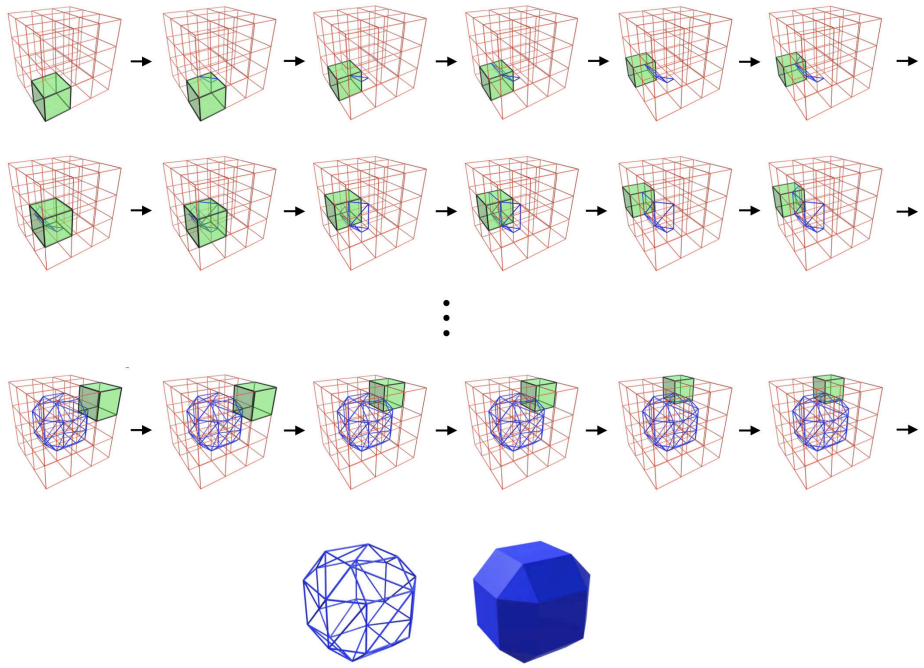

Figura 5.10: Costruzione della mesh: la ripetizione dell'algoritmo su tutti i cubi genera una mesh triangolare continua che approssima la superficie anatomica.

Non tutti gli algoritmi *Marching Cubes* sono uguali. La versione implementata nell'applicazione include diverse ottimizzazioni specifiche per il contesto chirurgico, progettate per migliorare significativamente la qualità dei modelli 3D ottenuti da scansioni TAC del ginocchio:

- **Analisi automatica dell'istogramma**: Il sistema analizza la distribuzione dei valori HU nel volume e **suggerisce automaticamente la soglia ottimale** per evidenziare le strutture ossee, eliminando la necessità di regolazioni manuali da parte dell'operatore.

- **Elaborazione intelligente delle regioni di interesse**: Invece di processare uniformemente tutto il volume — operazione computazionalmente costosa — l'algoritmo **identifica automaticamente le aree che contengono strutture anatomiche rilevanti**, concentrando la potenza di calcolo dove è effettivamente necessaria.

- **Correzione automatica degli artefatti**: Il sistema **rimuove automaticamente piccoli frammenti isolati e componenti disconnessi** che potrebbero essere generati da rumore nei dati o da imperfezioni nell'acquisizione, mantenendo solo le strutture anatomiche principali.

- **Preservazione intelligente dei dettagli**: Utilizza tecniche di **smoothing selettivo** che mantengono le caratteristiche anatomiche importanti (come bordi ossei e superfici articolari) mentre eliminano imperfezioni geometriche irrilevanti per la pianificazione chirurgica.
- **Ottimizzazione delle normali**: Corregge automaticamente l'orientamento delle superfici per *garantire un'illuminazione corretta* e una visualizzazione ottimale del modello 3D durante l'analisi chirurgica.

5.3.2 Rendering e visualizzazione del modello

Il modello tridimensionale generato viene visualizzato tramite `SceneKit` con l'obiettivo di fornire una rappresentazione chiara e intuitiva del ginocchio ricostruito.

Per garantire una resa visiva efficace, come si può notare in Figura 5.11, l'illuminazione della scena è stata ispirata alle *condizioni tipiche di una sala operatoria*. Una luce direzionale principale simula l'effetto di un faretto chirurgico, mentre una luce ambientale diffusa contribuisce a ridurre le ombre nette, migliorando la leggibilità delle superfici anatomiche.

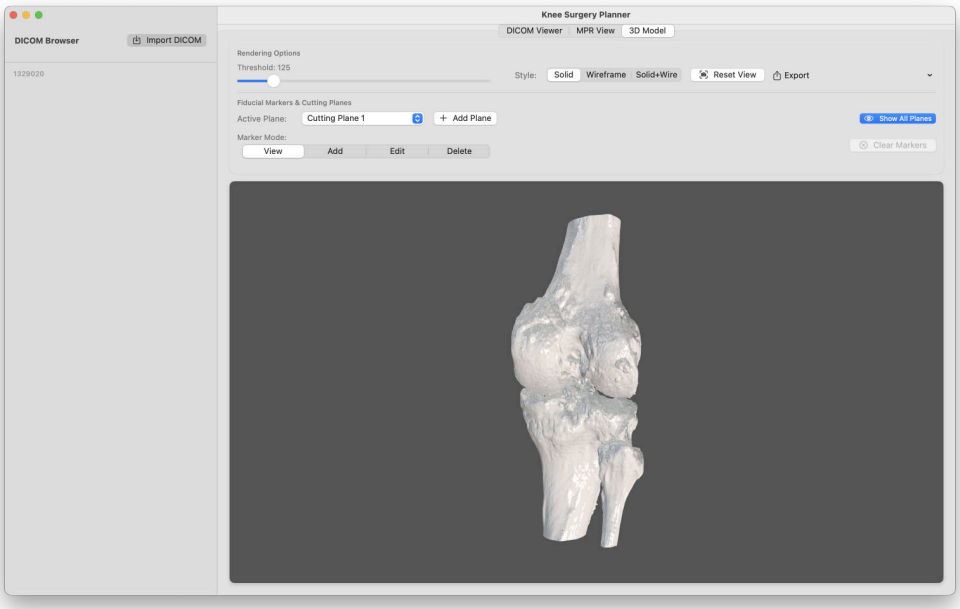

Figura 5.11: Visualizzatore 3D con controlli per rotazione e zoom.

Questo setup, riportato nel Listato 5.4, è pensato per mettere in risalto la geometria senza introdurre distrazioni visive.

File: Model3DViewModel.swift Modulo: UI App: macOS

```
1 private func setupLighting() {
2     // Luce ambientale - illuminazione generale diffusa
3     addLight(type: .ambient, intensity: 70,
4               color: NSColor(calibratedRed: 0.9, green: 0.9, blue: 1.0, alpha: 1.0)
                    )
5
6     // Luce principale direzionale - simula faretto chirurgico
```

```
    let mainLight = addLight(type: .directional, intensity: 1000,
                             color: NSColor(calibratedRed: 1.0, green: 0.95, blue:
                                 0.9, alpha: 1.0),
                             position: SCNVector3(100, 150, 100))
    mainLight.light?.castsShadow = true
    mainLight.look(at: SCNVector3(0, 0, 0))

    // Luce di riempimento - riduce ombre nette
    let fillLight = addLight(type: .directional, intensity: 400,
                             position: SCNVector3(-100, 50, -100))
    fillLight.look(at: SCNVector3(0, 0, 0))
}
```

Listato 5.4: Configurazione dell'illuminazione ispirata alla sala operatoria

Anche la configurazione dei materiali svolge un ruolo importante nella comprensione visiva del modello. Il sistema impiega materiali basati su *rendering* fisico, che simulano realisticamente l'interazione della luce con le superfici ossee. Questo approccio consente di evidenziare le irregolarità anatomiche e migliorare la percezione tridimensionale.

Come evidenziato nella Figura 5.12, il risultato finale è un modello 3D dettagliato e visivamente pulito, che conserva le caratteristiche anatomiche essenziali del distretto osseo. L'intero processo è automatico e richiede solo pochi minuti, senza necessità di interventi manuali da parte dell'utente. Il chirurgo può così disporre rapidamente di una rappresentazione tridimensionale accurata e pronta per essere consultata, ruotata, ingrandita o illuminata a seconda delle esigenze.

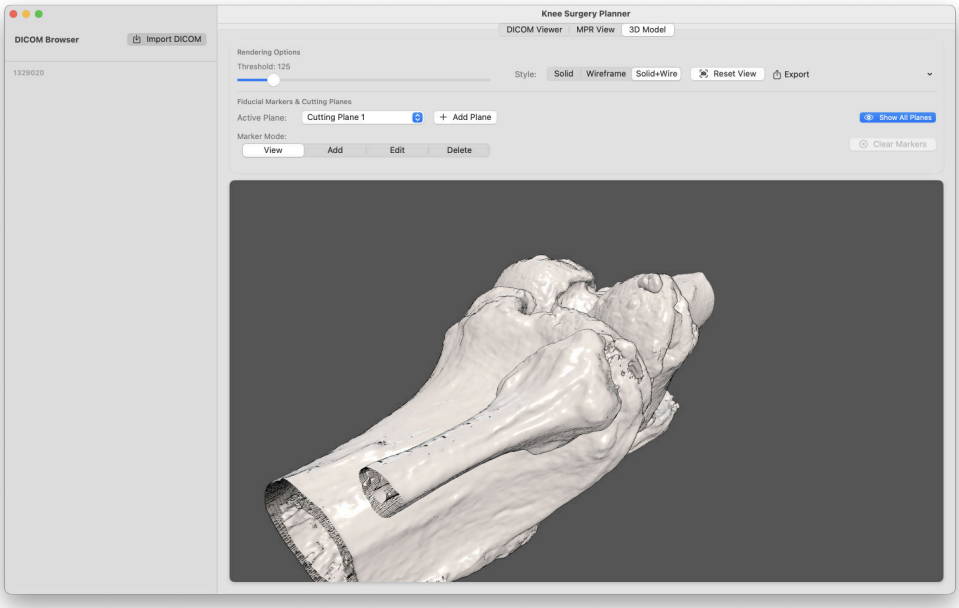

Figura 5.12: Modello 3D del ginocchio generato dall'algoritmo Marching Cubes ottimizzato.

5.4 Creazione e modifica dei piani di taglio

Una volta ottenuto il modello 3D del ginocchio, il chirurgo può iniziare a pianificare l'intervento definendo i piani di taglio.

5.4.1 Modalità di interazione per il posizionamento dei marker

Il sistema, tramite il *modulo UI*, offre diversi approcci per posizionare i marker chirurgici, adattandosi alle preferenze dell'operatore e al livello di precisione richiesto. La modalità principale utilizza il **click placement**: il chirurgo può semplicemente cliccare sulla superficie del modello 3D per posizionare un marker nel punto desiderato (Figura 5.13).

Il meccanismo tecnico che rende possibile questa interazione si basa sul *raycasting*. Quando il chirurgo clicca su un punto dello schermo, il sistema "lancia" un raggio virtuale dalla posizione del cursore verso il modello 3D, calcolando matematicamente dove questo raggio interseca la superficie:

$$\vec{P}(t) = \vec{O} + t \cdot \vec{D}$$

dove $\vec{O}$ rappresenta l'origine del raggio (la posizione della camera), $\vec{D}$ la direzione del raggio e t il parametro che determina la distanza lungo il raggio.

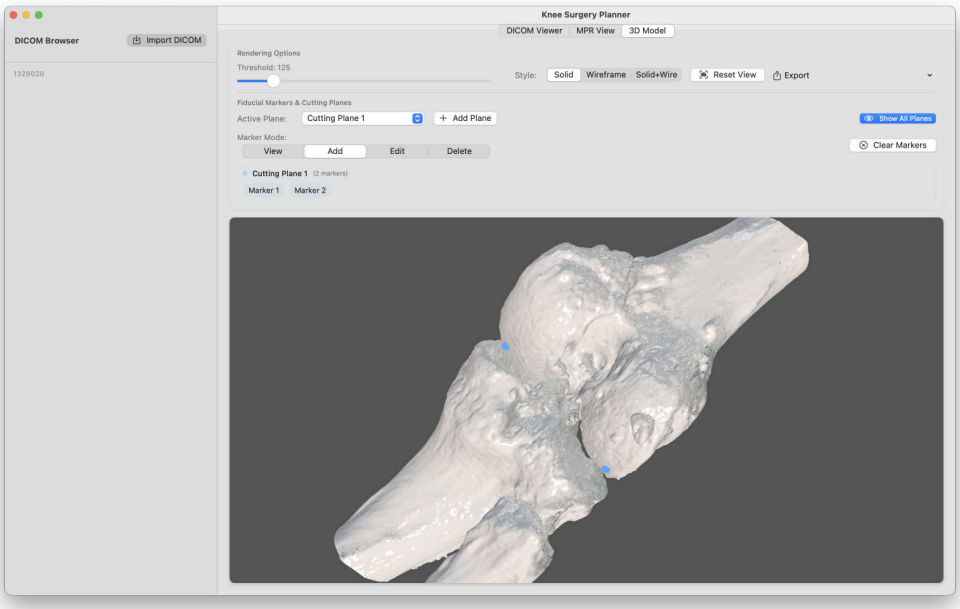

Figura 5.13: Posizionamento di marker sulla superficie del modello 3D tramite interazione diretta.

L'implementazione di questo sistema nel codice gestisce diversi scenari di interazione:

File: SceneKitMarkerView.swift Modulo: UI App: macOS

```swift
@objc func handleClick(_ gesture: NSClickGestureRecognizer) {
    guard let scnView = gesture.view as? SCNView else { return }

    let location = gesture.location(in: scnView)
    let hitResults = scnView.hitTest(location, options: hitTestOptions)

    // Filtra i risultati per escludere i piani di taglio
    let filteredResults = hitResults.filter { result in
        let nodeName = result.node.name ?? ""
        return !nodeName.starts(with: "cuttingPlane_")
    }

    guard let result = filteredResults.first else { return }

    switch markerMode {
    case .add:
        let added = markerManager.addMarker(at: result.worldCoordinates)
        if added == nil {
            // Gestisce il limite massimo di marker per piano
        }
    case .edit:
        // Gestisce la modifica della posizione
    case .delete:
        // Gestisce la rimozione
    }
}
```

Listato 5.5: Gestione dell'interazione per il posizionamento dei marker

Il sistema implementa anche controlli di validazione per garantire che i marker vengano posizionati correttamente. Ad esempio, ogni piano di taglio può contenere al massimo tre marker e il sistema previene automaticamente il posizionamento di marker aggiuntivi una volta raggiunto questo limite.

5.4.2 Calcolo automatico dei piani di taglio

Quando il chirurgo posiziona il terzo marker, il sistema ha informazioni sufficienti per calcolare automaticamente il piano di taglio. Dal punto di vista matematico, tre punti non collineari definiscono univocamente un piano nello spazio tridimensionale.

Il calcolo utilizza il **prodotto vettoriale** per determinare il vettore normale al piano:

$$\vec{n} = \frac{(\vec{P_2} - \vec{P_1}) \times (\vec{P_3} - \vec{P_1})}{|(\vec{P_2} - \vec{P_1}) \times (\vec{P_3} - \vec{P_1})|}$$

Una volta ottenuto il vettore normale $\vec{n}$, l'equazione del piano risulta:

$$\vec{n} \cdot (\vec{P} - \vec{P_1}) = 0$$

L'implementazione pratica di questo calcolo gestisce anche i casi limite e fornisce feedback visivo immediato:

File: MarkerManager.swift Modulo: Rendering App: macOS

```swift
private func calculatePlaneNormal(from points: [simd_float3]) -> simd_float3? {
    guard points.count >= 3 else { return nil }

    // Usa i primi tre punti per definire due vettori sul piano
    let v1 = points[1] - points[0]
    let v2 = points[2] - points[0]

    // Il prodotto vettoriale fornisce il vettore normale
    let normal = simd_normalize(simd_cross(v1, v2))
    return normal
}

func updateCuttingPlane(planeID: UUID) {
    let planeMarkers = markers(forPlane: planeID)

    // Necessari almeno 3 marker per definire un piano
    guard planeMarkers.count >= 3 else { return }

    // Calcola il centro del piano
    let center = calculateCenter(from: planeMarkers)

    // Determina l'orientamento
    let points = planeMarkers.map { simd_float3($0.position) }
    guard let normalVector = calculatePlaneNormal(from: points) else { return }

    // Crea e visualizza il piano
    createPlaneVisualization(center: center, normal: normalVector)
}
```

Listato 5.6: Calcolo automatico del piano di taglio dai marker

Il sistema implementa anche controlli di validazione geometrica per assicurare la correttezza del piano calcolato, funzionalità realizzata dal *modulo Rendering*, con supporto al *livello UI* per il trigger e la presentazione visiva.

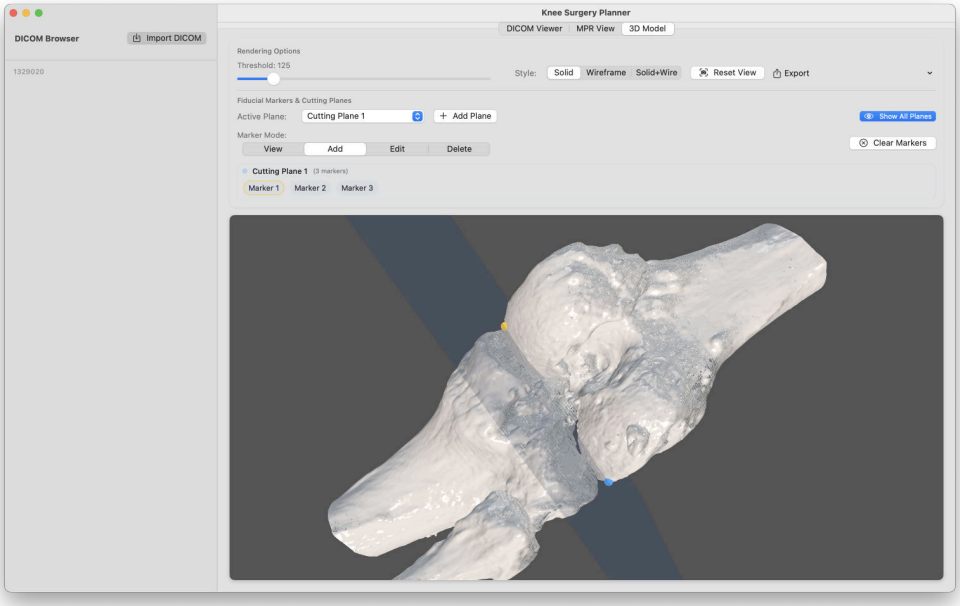

Figura 5.14: Processo di definizione di un piano di taglio.

Verifica che i marker non siano collineari (situazione che non definirebbe un piano unico) e che rispettino una distanza minima tra loro per evitare imprecisioni numeriche.

Negli interventi di chirurgia ortopedica, spesso è necessario definire più piani di taglio. L'applicazione gestisce questa complessità attraverso un sistema di organizzazione intuitivo. Ogni piano viene identificato con un **colore distintivo e un nome descrittivo**, facilitando il riconoscimento durante la pianificazione — come illustrato in Figura 5.15.

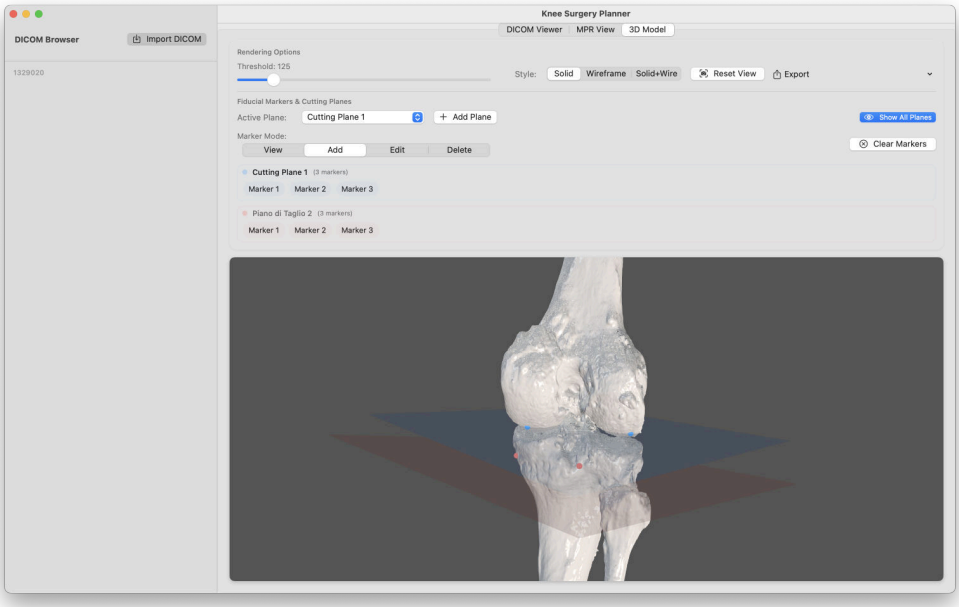

Figura 5.15: Gestione di piani multipli.

5.5 Esportazione dei dati per l'app visionOS

Dopo aver completato la pianificazione chirurgica sul desktop, il passo successivo consiste nel trasferire tutto il lavoro svolto verso l'*Apple Vision Pro*. Questo processo non è semplicemente una copia di file: richiede una serie di trasformazioni e ottimizzazioni specifiche per garantire che il modello 3D e le annotazioni chirurgiche funzionino perfettamente in realtà mista.

La conversione dei modelli segue un percorso ben definito che inizia dall'esportazione in formato `SceneKit` e culmina nella creazione di un file `.usdz` ottimizzato per *visionOS*, che preserva tutti gli elementi della pianificazione chirurgica:

File: ExportUtils.swift Modulo: Rendering App: macOS

```
static func exportModelWithMarkers(scene: SCNScene, markerManager: MarkerManager)
    {
   let exportScene = SCNScene()

   // Clona il modello 3D principale
   let clonedNode = meshNode.clone()
   let clonedGeometry = originalGeometry.copy() as! SCNGeometry
```

```
    clonedNode.geometry = clonedGeometry
    exportScene.rootNode.addChildNode(clonedNode)

    // Aggiunge i marker chirurgici
    for marker in markerManager.markers {
        let sphere = SCNSphere(radius: 3.0)
        let sphereMaterial = SCNMaterial()
        sphereMaterial.diffuse.contents = marker.color
        sphere.firstMaterial = sphereMaterial

        let markerNode = SCNNode(geometry: sphere)
        markerNode.position = marker.position
        markerNode.name = "Marker_\(marker.id.uuidString)"
        exportScene.rootNode.addChildNode(markerNode)
    }

    // Aggiunge i piani di taglio
    for plane in markerManager.cuttingPlanes {
        // Crea rappresentazione visiva del piano
        exportScene.rootNode.addChildNode(planeNode)
    }
}
```

Listato 5.7: Esportazione del modello con annotazioni chirurgiche

Durante questa fase, il sistema mantiene una struttura gerarchica precisa che **preserva le relazioni spaziali tra il modello anatomico, i marker e i piani di taglio**. Ogni elemento viene etichettato con convenzioni di nomenclatura specifiche che permetteranno all'app *visionOS* di riconoscere e gestire correttamente ogni componente.

5.6 Importazione e visualizzazione del modello in realtà mista

Questa sezione esplora come l'applicazione gestisce l'importazione e la configurazione dei modelli 3D in *visionOS*. È un processo che richiede particolare attenzione, soprattutto quando si tratta di ottimizzare la visualizzazione per l'ambiente immersivo.

L'importazione dei file, gestita dal *modulo UI dell'app visionOS*, avviene attraverso il meccanismo di `fileImporter` di *SwiftUI*, che implementa automaticamente le *security-scoped resources* per garantire un accesso sicuro ai file selezionati dall'utente:

File: ContentView.swift Modulo: UI App: visionOS

```
.fileImporter(
    isPresented: $showImporter,
    allowedContentTypes: [UTType.usdz, .sceneKitScene],
    allowsMultipleSelection: false
) { result in
    // Gestione dell'importazione
}
```

Listato 5.8: Configurazione dell'importatore di file

5.6.1 Conversione automatica da SCN a RealityKit

Quando l'utente importa un file `.scn`, l'applicazione esegue automaticamente una conversione verso un formato compatibile con `RealityKit`, necessario per la visualizzazione e manipolazione del modello in ambienti immersivi. Questo processo è gestito dal *modulo Utility* dell'app visionOS e si articola in più fasi:

1. **Parsing della scena**: al momento dell'importazione di un file `.scn`, il sistema analizza l'intera struttura gerarchica della scena per **identificare tutti gli elementi che la compongono**. Questo include, ad esempio, i vari componenti del ginocchio (femore, tibia, rotula) ed eventuali luci. Il parsing avviene in modo ricorsivo, così da non trascurare eventuali sottolivelli o nodi figli all'interno della scena originale.

2. **Estrazione delle geometrie**: ogni parte anatomica è rappresentata da una primitiva geometrica (come le sfere per i marker) o da una mesh dettagliata (come nel caso del modello 3D vero e proprio). Durante la conversione, queste geometrie vengono tradotte nel formato `MeshResource`, utilizzato da `RealityKit` per il *rendering*. Ad esempio, la superficie articolare del femore, originariamente definita in `SceneKit`, viene trasformata in una mesh compatibile con il motore 3D della nuova piattaforma.

3. **Migrazione dei materiali**: per mantenere un aspetto visivo coerente, il sistema converte i materiali `SceneKit` (come quelli utilizzati per evidenziare l'osso o i tessuti molli) in materiali `RealityKit`. Viene solitamente impiegato `SimpleMaterial`, che permette di conservare proprietà essenziali come colore, opacità e riflessione, garantendo una buona resa visiva anche in ambienti di realtà mista.

4. **Preservazione delle trasformazioni**: durante la conversione vengono **mantenute posizione, orientamento e scala** di ciascun componente. Questo è fondamentale per assicurare che le varie parti del ginocchio risultino correttamente allineate e posizionate nella scena finale, evitando alterazioni della geometria o errori di sovrapposizione tra le strutture ossee.

Le primitive geometriche più comuni (come `SCNBox`, `SCNSphere`, ecc.) vengono mappate automaticamente in forme equivalenti di `RealityKit`. Il codice riportato nel Listato 5.9 mostra un esempio di conversione per una geometria di tipo `SCNBox` (relativa al modello 3D): vengono estratte le dimensioni del box e utilizzate per generare una mesh compatibile con `RealityKit`. A questa mesh viene poi applicato un materiale semplice, prima di creare l'oggetto finale `ModelEntity` che sarà visualizzato nella scena immersiva.

File: ModelLoader.swift Modulo: Utility App: visionOS

```
if let box = geometry as? SCNBox {
let boxDimensions = SIMD3<Float>(
Float(box.width),
Float(box.height),
Float(box.length)
)
let boxMesh = MeshResource.generateBox(size: boxDimensions)
let material = SimpleMaterial(
color: .cyan,
roughness: 0.2,
isMetallic: false
)
modelEntity = ModelEntity(mesh: boxMesh, materials: [material])
}
```

Listato 5.9: Conversione di primitive geometriche `SCN` in RealityKit

5.6.2 Calcolo automatico di scala e posizionamento

Uno degli aspetti più critici per una buona esperienza utente è la corretta visualizzazione del modello. I modelli 3D possono arrivare con dimensioni molto diverse tra loro — alcuni potrebbero essere microscopici, altri enormi. Per questo, l'applicazione implementa un sistema di normalizzazione automatica.

Il processo funziona attraverso i seguenti passaggi:

- **Calcolo del bounding box**: l'algoritmo determina le dimensioni complessive del modello analizzando tutti i suoi componenti.
- **Auto-scaling**: il modello viene scalato automaticamente per raggiungere dimensioni appropriate (target di 0.5 metri per i modelli anatomici).
- **Centramento**: il modello viene posizionato al centro dell'origine per una visualizzazione ottimale.
- **Correzione dell'orientamento**: se necessario, viene applicata una rotazione per allineare il modello secondo standard predefiniti.

Questo approccio garantisce che indipendentemente dalle dimensioni originali, il modello risulti **sempre visualizzabile correttamente** nell'ambiente virtuale.

Una volta completato il processo di importazione e ottimizzazione, il modello anatomico viene visualizzato nell'interfaccia principale dell'applicazione, come mostrato nella Figura 5.16. Il sistema di *rendering* utilizza tecniche avanzate di *shading* per evidenziare i dettagli anatomici, mentre il piano di taglio semi-trasparente permette di visualizzare la sezione interna del modello.

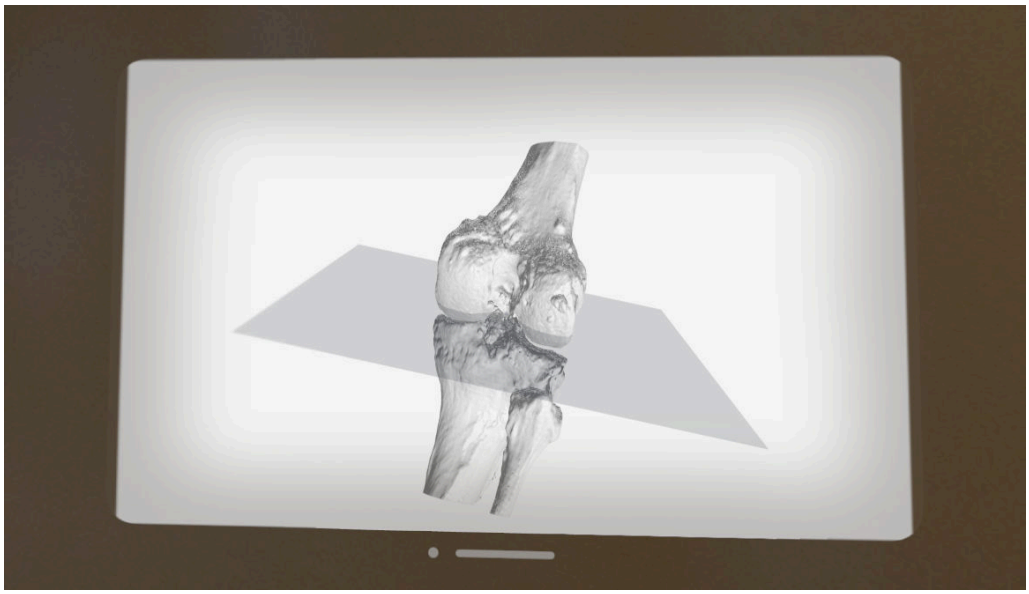

Figura 5.16: Visualizzazione del modello 3D del ginocchio.

5.6.3 Configurazione dello spazio immersivo

Il passaggio dalla visualizzazione tradizionale a quella immersiva richiede una configurazione specifica. L'applicazione utilizza il sistema `ImmersiveSpace` di *visionOS*:

File: OA_ProjectApp.swift Modulo: Entry Point App: visionOS

```
ImmersiveSpace(id: appModel.immersiveSpaceID) {
    ImmersiveView()
        .environment(appModel)
}
.immersionStyle(selection: .constant(.mixed), in: .mixed)
```

Listato 5.10: Configurazione dello spazio immersivo

La gestione dello stato dello spazio immersivo segue un pattern ben definito. L'applicazione traccia tre stati principali:

- `.closed`: lo spazio immersivo non è attivo, l'utente si trova nell'interfaccia tradizionale.
- `.inTransition`: fase temporanea durante l'apertura o la chiusura dello spazio.
- `.open`: lo spazio immersivo è completamente attivo e l'utente può interagire con il modello.

Questa gestione degli stati è fondamentale per evitare problemi di sincronizzazione e per offrire un'esperienza fluida all'utente durante le transizioni.

5.6.3.1 Ottimizzazioni per la realtà mista

Nel contesto della realtà mista, l'applicazione implementa diversi accorgimenti per migliorare l'integrazione con l'ambiente reale:

- **Materiali olografici**: i modelli sono renderizzati con materiali semitrasparenti che consentono di mantenere la percezione dell'ambiente circostante anche durante l'osservazione del contenuto virtuale.
- **Illuminazione adattiva**: l'illuminazione della scena si adatta dinamicamente alle condizioni ambientali reali, migliorando il realismo e la coerenza visiva tra oggetti virtuali e spazio fisico.
- **Ancoraggio spaziale**: i modelli virtuali vengono posizionati su superfici reali (es. pavimento o tavoli) sfruttando il sistema di rilevamento dei piani offerto da `ARKit` all'interno di *visionOS*.

Questo approccio multi-livello garantisce che l'esperienza di visualizzazione sia non solo tecnicamente corretta, ma anche intuitiva e coinvolgente per l'utente finale.

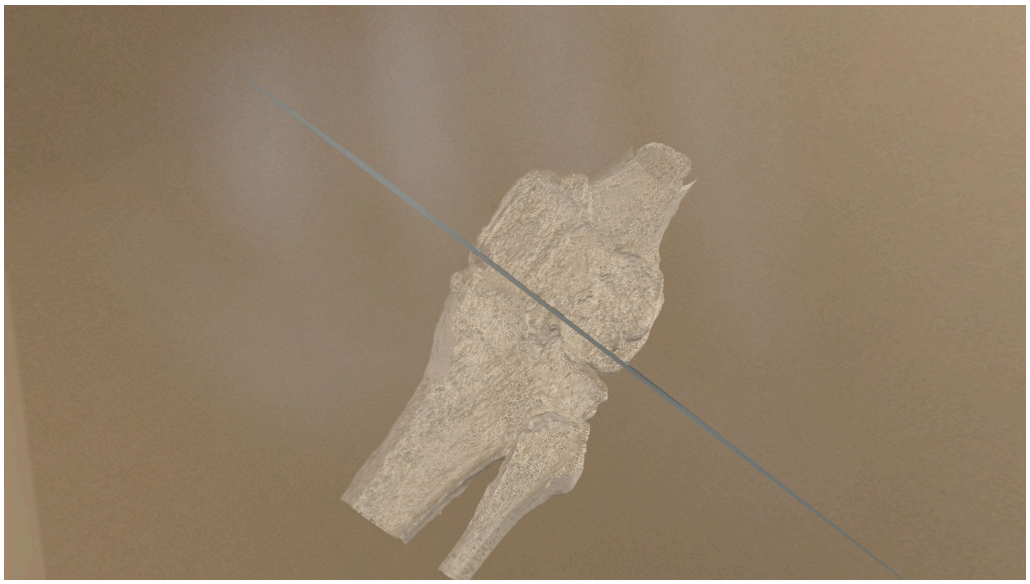

Figura 5.17: Modello 3D del ginocchio nello spazio immersivo con piano di taglio attivo.

5.7 Interazione con il modello: manipolazione, zoom, rotazione

Una volta che il modello 3D è stato caricato nello spazio immersivo, l'utente deve poter interagire con esso in modo naturale e intuitivo. Questa sezione descrive come l'applicazione sfrutta le capacità avanzate di *visionOS* per offrire un'esperienza di manipolazione coinvolgente e precisa.

5.7.1 Sistema di controlli gestuali

La vera forza di *visionOS* risiede nella sua capacità di tracciare con precisione i movimenti delle mani dell'utente. L'applicazione sfrutta il sistema di hand tracking nativo integrato con `ARKit` per riconoscere diversi tipi di gesti, ognuno associato a specifiche funzionalità.

Il sistema riconosce principalmente quattro categorie di gesti:

- **Pinch gesture**: rappresenta il gesto fondamentale per la selezione. L'utente unisce pollice e indice per "afferrare" virtualmente un elemento del modello o per posizionare marker di annotazione.
- **Drag gesture**: consente di ruotare il modello nello spazio. Dopo averlo selezionato tramite pinch, l'utente può trascinarlo per modificarne l'orientamento lungo le tre dimensioni.
- **Two-hand pinch**: coinvolge entrambe le mani per eseguire operazioni di ridimensionamento. Il modello viene ingrandito o ridotto in modo uniforme, preservando le proporzioni.
- **Palm orientation**: permette un controllo fine dell'orientamento. La posizione e la rotazione del palmo influenzano la rotazione precisa del modello, utile per piccoli aggiustamenti.

Il riconoscimento del pinch gesture si basa su un calcolo relativamente semplice ma efficace della distanza euclidea tra pollice e indice:

File: ImmersiveView.swift Modulo: UI App: visionOS

```swift
let distance = simd_distance(
    SIMD3<Float>(thumbPos.x, thumbPos.y, thumbPos.z),
    SIMD3<Float>(indexPos.x, indexPos.y, indexPos.z)
)
let pinchThreshold: Float = 0.03 // 3cm
let isPinching = distance < pinchThreshold
```

Listato 5.11: Riconoscimento del pinch gesture

La soglia di 3 centimetri si è rivelata ottimale attraverso test empirici, offrendo un buon compromesso tra sensibilità e stabilità del riconoscimento.

5.7.2 Manipolazione spaziale del modello

L'interazione con modelli 3D richiede un controllo completo nello spazio tridimensionale. L'applicazione implementa un sistema che supporta tutti e **sei i gradi di libertà (6DOF)**, permettendo movimento e rotazione lungo tutti gli assi principali.

I sei gradi di libertà si dividono in:

1. **Traslazione lungo X, Y, Z**: Movimento lineare del modello in qualsiasi direzione dello spazio.

2. **Rotazione attorno a X, Y, Z**: Rotazione del modello attorno ai tre assi principali.

Per offrire diverse modalità di interazione, l'applicazione implementa un sistema flessibile che si adatta alle esigenze dell'utente. La modalità principale è la **direct manipulation**, in cui l'utente controlla il modello tridimensionale in modo diretto, semplicemente muovendo le mani nello spazio: i **movimenti vengono tracciati dal sistema e applicati in tempo reale all'oggetto virtuale**, in modo da offrire un'interazione naturale e immediata. In alternativa, è disponibile anche la **constraint-based manipulation**, una modalità in cui i movimenti sono vincolati a determinati assi o piani. Questa opzione si rivela particolarmente utile in situazioni che richiedono un alto grado di precisione, come l'allineamento di impianti o la definizione di traiettorie chirurgiche.

A livello implementativo, queste interazioni si basano sul sistema di riconoscimento gesture offerto da *SwiftUI*, opportunamente adattato per gestire operazioni in ambiente tridimensionale. Ad esempio, la rotazione del modello mediante *drag gesture* è stata personalizzata per agire in modo coerente con la prospettiva e l'orientamento della scena 3D:

File: ImmersiveView.swift Modulo: UI App: visionOS

```
.gesture(
    DragGesture()
        .onChanged { value in
            let rotationAmount = Float(value.translation.width - dragAmount.width)
                / 200.0
            if let model = appModel.currentModel {
                let rotationY = simd_quatf(angle: rotationAmount, axis: [0, 1, 0])
                model.transform.rotation = model.transform.rotation * rotationY
                dragAmount = value.translation
            }
        }
        .onEnded { _ in
            dragAmount = .zero
        }
)
```

Listato 5.12: Implementazione della rotazione tramite drag

5.7.3 Controlli di zoom e scala

Il sistema di *zoom* rappresenta un elemento cruciale per l'esplorazione dettagliata dei modelli anatomici. L'applicazione implementa diverse tecniche complementari per offrire all'utente il **massimo controllo**:

- **Pinch-to-zoom**: utilizza la distanza tra le due mani per determinare il fattore di scala. Aumentando la distanza, il modello si ingrandisce proporzionalmente.

- **Focal zoom**: consente di ingrandire una specifica area del modello, mantenendo il punto di interesse centrato nella vista.
- **Uniform scaling**: assicura che tutte le operazioni di zoom mantengano inalterate le proporzioni del modello, evitando qualsiasi distorsione.

La matematica dietro al pinch-to-zoom è elegante nella sua semplicità:

$$scale_{new} = scale_{current} \times \frac{distance_{current}}{distance_{initial}} \tag{5.2}$$

Questa formula garantisce che il fattore di scala sia direttamente proporzionale alla variazione di distanza tra le mani dell'utente.

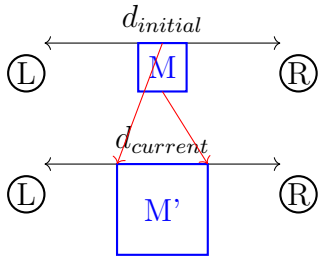

Figura 5.18: Schema del pinch-to-zoom: l'aumento della distanza tra le mani produce un ingrandimento proporzionale del modello.

5.7.4 Feedback aptico e visivo

Un'interfaccia di qualità non può limitarsi alla sola funzionalità, ma deve offrire un feedback chiaro e immediato all'utente. L'applicazione implementa un sistema di feedback multimodale che sfrutta diversi canali sensoriali:

- **Visual feedback**: evidenzia le aree del modello con cui l'utente può interagire, utilizzando effetti di *highlighting* e cambi di colore per guidare l'interazione.
- **Spatial audio**: fornisce un feedback sonoro spazializzato che conferma le azioni dell'utente. I suoni sono posizionati nello spazio 3D in corrispondenza dell'interazione, aumentando il senso di immersione.
- **Constraint visualization**: rappresenta visivamente i vincoli e i limiti di movimento, supportando la comprensione delle azioni possibili in ogni contesto.

L'implementazione del feedback visivo per le aree interattive utilizza la modifica dinamica dei materiali:

File: ImmersiveView.swift Modulo: UI App: visionOS

```
private func highlightInteractiveArea(entity: ModelEntity, isHighlighted: Bool) {
    if isHighlighted {
        var highlightMaterial = SimpleMaterial()
        highlightMaterial.color = .init(tint: .yellow.withAlphaComponent(0.8))
        highlightMaterial.roughness = 0.1
        highlightMaterial.metallic = 0.9
        entity.model?.materials = [highlightMaterial]
```

```
    } else {
        // Ripristina il materiale originale
        applyHolographicMaterial(to: entity)
    }
}
```

Listato 5.13: Implementazione del feedback visivo interattivo

Questo sistema di feedback permette all'utente di comprendere immediatamente quali elementi sono interattivi e quali azioni sono possibili in ogni momento, riducendo la curva di apprendimento e migliorando l'esperienza complessiva.

Il sistema di *gesture recognition* deve funzionare in modo affidabile anche quando i chirurghi indossano guanti sterili, che possono alterare la precisione dell'*hand tracking*. Per questo motivo, l'applicazione implementa **algoritmi di compensazione che tengono conto dello spessore aggiuntivo dei guanti** e delle possibili interferenze che possono causare.

File: ImmersiveView.swift Modulo: UI App: visionOS

```
// Compensazione per guanti chirurgici
let gloveCompensation: Float = 0.005 // 5mm di spessore aggiuntivo
let adjustedPinchThreshold = standardPinchThreshold + gloveCompensation

// Filtro per ridurre jitter causato dai guanti
let handPositionFilter = LowPassFilter(cutoffFrequency: 10.0)
let filteredHandPosition = handPositionFilter.apply(to: rawHandPosition)
```

Listato 5.14: Configurazione del gesture recognition per ambiente sterile

L'*auto-positioning* del modello 3D è un'altra caratteristica essenziale. Il sistema deve essere in grado di **posizionare automaticamente il modello anatomico in base alla posizione del paziente e all'orientamento del campo chirurgico**, minimizzando la necessità di aggiustamenti manuali durante l'intervento.

5.8 Inserimento marker con pinch su ginocchio reale

Una delle funzionalità più innovative del prototipo è la capacità di posizionare marker virtuali direttamente nell'ambiente reale utilizzando gesti naturali delle mani.

5.8.1 Rilevamento del pinch

Il cuore del sistema di posizionamento marker, gestito dal *modulo Service* e coordinato via *UI*, risiede nella capacità di riconoscere con precisione il *pinch* della mano. `ARKit` fornisce un tracking dettagliato delle articolazioni delle mani, ma trasformare questi dati grezzi in un'interazione significativa richiede algoritmi sofisticati.

Il sistema monitora continuamente la distanza tra pollice e indice della mano dominante dell'utente. Quando questa distanza scende sotto una soglia

predefinita, viene registrato un potenziale gesto di pinch.

L'interfaccia principale del sistema di posizionamento marker è mostrata nella Figura 5.19. In questa vista iniziale, il contatore mostra *"Markers: 0/3"*, indicando che non sono ancora stati posizionati marker nell'ambiente. L'interfaccia fornisce istruzioni chiare all'utente su come utilizzare il gesto pinch per posizionare i marker.

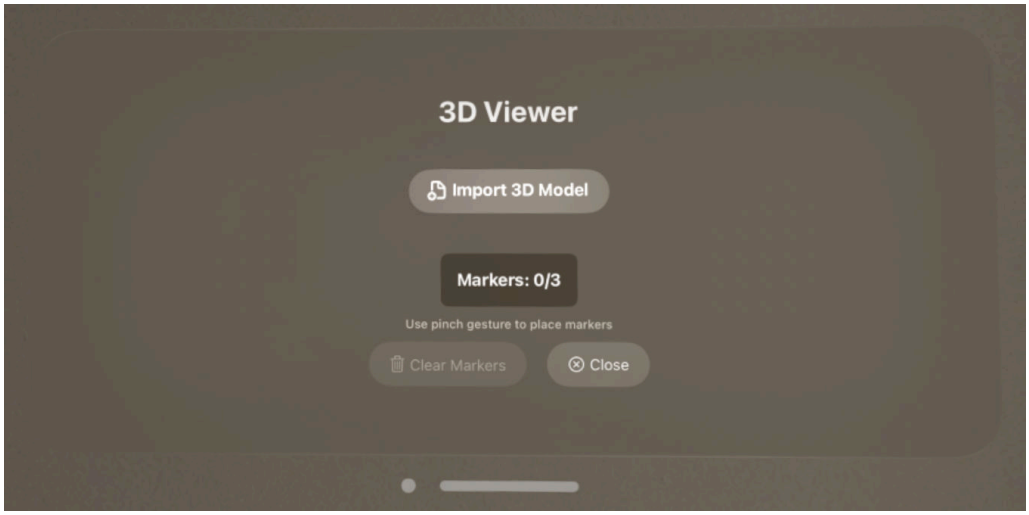

Figura 5.19: Interfaccia iniziale del sistema di posizionamento marker prima dell'inserimento.

Il frammento di codice che segue mostra in che modo l'applicazione converte i dati grezzi di `ARKit` in un segnale binario *"pinch / no-pinch"*. In particolare, viene calcolata in tempo reale la distanza euclidea tra le punte di pollice e indice e la si confronta con una soglia di sicurezza di 3 cm, generando così l'evento che attiva il posizionamento del marker.

File: ImmersiveView.swift Modulo: UI App: visionOS

```swift
private func checkPinchGesture(handAnchor: HandAnchor) async {
    guard let handSkeleton = handAnchor.handSkeleton else { return }

    let thumbTip = handSkeleton.joint(.thumbTip)
    let indexTip = handSkeleton.joint(.indexFingerTip)

    guard thumbTip.isTracked && indexTip.isTracked else { return }

    let thumbPos = thumbTip.anchorFromJointTransform.columns.3
    let indexPos = indexTip.anchorFromJointTransform.columns.3

    let distance = simd_distance(
        SIMD3<Float>(thumbPos.x, thumbPos.y, thumbPos.z),
        SIMD3<Float>(indexPos.x, indexPos.y, indexPos.z)
    )

    let pinchThreshold: Float = 0.03 // Soglia di 3cm
    let isPinching = distance < pinchThreshold
}
```

Rilevamento del gesto pinch tramite ARKit

Un semplice rilevamento della distanza non è sufficiente in un contesto chirurgico avanzato. Il sistema deve essere in grado di distinguere tra gesti

intenzionali e movimenti involontari, evitando al contempo interferenze con l'interfaccia utente tradizionale.

Il processo di validazione opera su più livelli temporali. Prima di tutto, il gesto deve persistere per almeno 300 millisecondi per essere considerato intenzionale. Questa durata minima elimina la maggior parte dei falsi positivi causati da movimenti involontari.

Dall'altro lato, gesti che durano più di 3 secondi vengono considerati probabilmente accidentali. Un chirurgo che intende posizionare un marker eseguirà tipicamente un gesto deciso e relativamente breve. La logica descritta è implementata direttamente nel modulo `UI`, come mostrato nel Listato 5.16. Il metodo `shouldCreateMarker()` controlla che non ci siano state interazioni recenti con l'interfaccia utente e verifica che il gesto abbia una durata compresa tra i 300 millisecondi e i 3 secondi, rispettando i criteri di validazione descritti.

File: ImmersiveView.swift Modulo: UI App: visionOS

```swift
private func shouldCreateMarker() -> Bool {
    // Evita conflitti con interazioni UI recenti
    let timeSinceUIInteraction = Date().timeIntervalSince(lastUIInteractionTime)
    if timeSinceUIInteraction < 2.0 {
        return false
    }

    // Verifica durata del pinch
    guard let startTime = pinchStartTime else { return false }
    let pinchDuration = Date().timeIntervalSince(startTime)

    return pinchDuration >= 0.3 && pinchDuration <= 3.0
}
```

Listato 5.16: Sistema di validazione temporale per il pinch

Quando un utente interagisce con elementi dell'interfaccia virtuale, è probabile che nelle immediate vicinanze temporali possa eseguire involontariamente gesti che assomigliano a un *pinch*. Il sistema introduce quindi una *"zona morta"* temporale di 2 secondi dopo ogni interazione *UI*.

5.8.2 Trasformazione delle coordinate spaziali

Una volta validato il gesto, il *modulo Service*, con il supporto del *Core* per le trasformazioni matematiche, si occupa di calcolare la posizione spaziale del marker. Questa viene inizialmente determinata come punto medio tra la punta del pollice e quella dell'indice, ovvero il centro concettuale del gesto di pinch. Tuttavia, poiché le posizioni delle dita sono espresse nel sistema di coordinate locale della mano (che si muove insieme all'utente), è necessario trasformarle in **coordinate mondiali**. Questo sistema di riferimento descrive lo spazio tridimensionale dell'intera scena *AR*, ed è indipendente dalla posizione o orientamento della mano. Convertire la posizione del pinch in coordinate mondiali consente quindi di **posizionare il marker in modo stabile e coerente rispetto all'ambiente circostante**.

Il codice riportato nel Listato 5.17 mostra come l'applicazione esegue la trasformazione delle coordinate. Per prima cosa viene calcolata la posizione media tra la punta del pollice e quella dell'indice, nel sistema di riferimento locale della mano. Successivamente, questa posizione viene trasformata nel sistema di coordinate mondiali applicando la *matrice di trasformazione associata all'ancora della mano (hand anchor)*, che descrive la posizione e l'orientamento della mano all'interno della scena *AR*.

File: ImmersiveView.swift Modulo: UI App: visionOS

```
// Calcola posizione marker (punto medio tra pollice e indice)
let markerLocalPos = (
SIMD3<Float>(thumbPos.x, thumbPos.y, thumbPos.z) +
SIMD3<Float>(indexPos.x, indexPos.y, indexPos.z)
) / 2

// Trasforma in coordinate mondo
let worldTransform = handAnchor.originFromAnchorTransform
let markerWorldPos = worldTransform * SIMD4<Float>(
markerLocalPos.x,
markerLocalPos.y,
markerLocalPos.z,
1
)
let finalPos = SIMD3<Float>(markerWorldPos.x, markerWorldPos.y, markerWorldPos.z)
```

Listato 5.17: Calcolo della posizione mondiale del marker

Il processo è schematizzato anche in Figura 5.20, che mostra il passaggio da coordinate locali (legate al sistema di riferimento della mano) a coordinate mondiali (fisse nello spazio della scena). Come accennato in precedenza, questa trasformazione è fondamentale per garantire un corretto allineamento tra i gesti dell'utente e il posizionamento effettivo dei marker nel mondo virtuale.

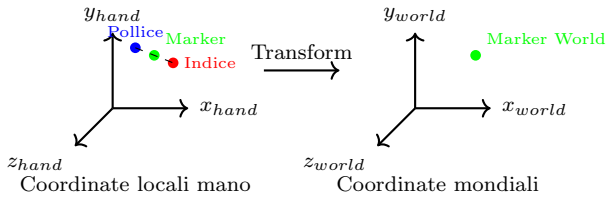

Figura 5.20: Trasformazione delle coordinate dal sistema locale della mano al sistema mondiale.

5.8.3 Generazione automatica del piano di taglio

I marker mantengono la loro persistenza spaziale: rimangono ancorati alla posizione fisica in cui sono stati posizionati anche quando l'utente si muove nell'ambiente. Quando esattamente tre marker sono presenti nella scena, il sistema attiva automaticamente la funzionalità di generazione del piano di taglio.

Come mostrato nella Figura 5.21, una volta posizionati tutti e tre i marker (indicati dalle sfere blu), l'interfaccia aggiorna il contatore a *"Markers: 3/3"* e attiva automaticamente la generazione del piano di taglio. L'in-

dicatore verde *"Cutting Plane Active"* conferma che il sistema ha calcolato e visualizzato il piano basato sui tre punti di riferimento posizionati dall'utente.

Figura 5.21: Interfaccia con tutti e tre i marker posizionati e piano di taglio attivo.

La generazione del piano sfrutta i principi della geometria euclidea adattati al contesto di realtà mista. Una volta che l'utente ha posizionato tre marker nello spazio, l'algoritmo calcola i vettori tra di essi nel sistema di coordinate mondiali fornito da `ARKit`, e determina la normale al piano tramite il prodotto vettoriale. Il risultato è un **piano virtuale ancorato all'ambiente reale**, utile per definire superfici di riferimento come ad esempio piani di taglio chirurgici.

Il Listato 5.18 mostra l'implementazione di questo processo all'interno dell'applicazione. Dopo aver verificato la presenza di tre marker, il sistema calcola due vettori a partire dalle loro posizioni, ne esegue il prodotto vettoriale per ottenere la normale al piano, e infine calcola il centro geometrico tra i tre punti. In caso di collinearità, viene utilizzata una normale di fallback. Il piano risultante viene poi visualizzato nella scena *AR*.

File: ImmersiveView.swift Modulo: UI App: visionOS

```
1
2  private func createCuttingPlane() {
3      guard appModel.markerManager.markers.count == 3 else { return }
4
5      let pos1 = appModel.markerManager.markers[0].position
6      let pos2 = appModel.markerManager.markers[1].position
7      let pos3 = appModel.markerManager.markers[2].position
8
9      // Calcola vettori del piano
10     let v1 = pos2 - pos1
11     let v2 = pos3 - pos1
12
```

```
    // Calcola normale tramite prodotto vettoriale
    let crossProduct = cross(v1, v2)
    let crossLength = simd_length(crossProduct)

    // Gestisce il caso di collinearit
    if crossLength < 0.001 {
        let fallbackNormal = SIMD3<Float>(0, 0, 1)
        createVisiblePlane(at: (pos1 + pos2 + pos3) / 3, normal: fallbackNormal)
        return
    }

    let normal = normalize(crossProduct)
    let center = (pos1 + pos2 + pos3) / 3

    createVisiblePlane(at: center, normal: normal)
}
```

Listato 5.18: Generazione automatica del piano di taglio

Il meccanismo geometrico alla base della costruzione del piano è illustrato schematicamente nella Figura 5.22, dove i tre marker sono utilizzati per definire due vettori, e da questi viene derivata la normale tramite prodotto vettoriale. La normale serve sia per definire l'orientamento del piano, sia per garantire coerenza spaziale rispetto all'ambiente tridimensionale in cui viene proiettato.

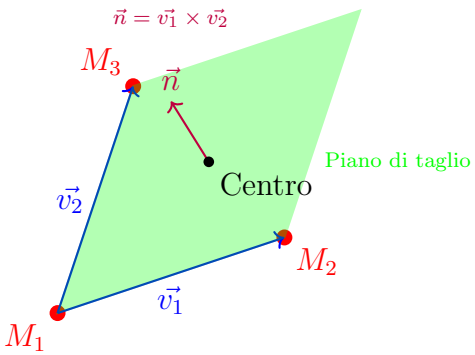

Figura 5.22: Generazione del piano di taglio dai tre marker posizionati.

Il piano di taglio generato necessita di una visualizzazione che sia al tempo stesso informativa e non invasiva. Il sistema utilizza materiali semitrasparenti ottimizzati per l'ambiente di realtà mista, permettendo di vedere attraverso il piano mantenendo comunque una chiara percezione della sua posizione e orientamento.

Capitolo 6

Validazione e riscontri clinici

Completata la fase implementativa, il presente capitolo si concentra sulla valutazione critica del prototipo sviluppato e sulla sua validazione rispetto agli obiettivi iniziali. Viene condotta un'analisi del grado di soddisfacimento dei requisiti funzionali e non funzionali definiti nella fase progettuale, con particolare attenzione alla verifica dell'efficacia del sistema nel contesto clinico reale. Il capitolo presenta inoltre un confronto dettagliato con le metodologie chirurgiche attualmente utilizzate, evidenziando vantaggi, svantaggi e posizionamento competitivo del sistema proposto. Conclude l'analisi una discussione critica delle limitazioni emerse durante la sperimentazione, fornendo una valutazione obiettiva delle potenzialità e dei vincoli che condizionano l'applicabilità del prototipo in contesti operatori reali.

6.1 Verifica del soddisfacimento dei requisiti

L'analisi del sistema implementato conferma il **raggiungimento degli obiettivi principali** stabiliti nella fase di progettazione. L'architettura integrata *macOS-visionOS* ha dimostrato di rispondere efficacemente alle esigenze identificate durante la collaborazione con i chirurghi ortopedici.

Dal punto di vista funzionale, il sistema gestisce correttamente l'intero ciclo operativo, dall'importazione delle TAC fino alla visualizzazione immersiva dei piani di taglio. La compatibilità con gli standard e la qualità dei modelli 3D generati si sono rivelate **adeguate per il contesto clinico**. L'integrazione tra le due piattaforme preserva integralmente le informazioni di pianificazione, garantendo continuità tra fase preoperatoria e intraoperatoria. Il controllo gestuale in ambiente immersivo permette la manipolazione tramite *gesture* naturali con controllo completo sui sei gradi di libertà, mentre il sistema di posizionamento *marker* attraverso *pinch gesture* elimina i falsi positivi tramite validazione temporale e filtri per interazioni *UI*.

L'unico requisito non completamente soddisfatto riguarda l'ancoraggio automatico del modello virtuale al ginocchio reale. Il sistema implementa un posizionamento manuale funzionale, ma l'automazione completa richiederebbe algoritmi avanzati di *computer vision* considerati oltre lo *scope* del

prototipo.

I requisiti non funzionali mostrano un soddisfacimento complessivamente positivo. L'applicazione *macOS* ha ricevuto **riscontri molto favorevoli** dai chirurghi coinvolti, che hanno apprezzato la linearità del processo operativo e l'eliminazione di tecnicismi superflui. L'interfaccia risulta intuitiva e il *workflow* si integra naturalmente con le pratiche cliniche esistenti. L'applicazione *visionOS*, pur dimostrando le potenzialità dell'approccio immersivo, presenta margini di miglioramento nell'usabilità legati principalmente alle **limitazioni hardware attuali** del *Vision Pro* in contesti operatori. Le prestazioni si sono dimostrate adeguate con tempi di ricostruzione 3D di **2-5 minuti** e *rendering* immersivo stabile a bassa latenza.

Requisito Funzionale	Stato	Note Implementative
APPLICAZIONE macOS		
Importazione immagini TAC DICOM	Implementato	*Supporto completo tramite* `DCMTK`
Generazione modello 3D del ginocchio	Implementato	*Algoritmo Marching Cubes ottimizzato*
Visualizzazione multiplanare	Implementato	*Interpolazione trilineare avanzata*
Creazione piani di taglio tramite marker	Implementato	*Calcolo automatico della normale*
Esportazione modelli per visionOS	Implementato	*Formati* `.scn` *e* `.usdz`
APPLICAZIONE visionOS		
Importazione modello 3D in realtà mista	Implementato	*Supporto* `USDZ` *e conversione* `SCN`
Manipolazione modello (6 gradi di libertà)	Implementato	*Gesture naturali e intuitive*
Visualizzazione piani di taglio immersivi	Implementato	*Materiali olografici avanzati*
Posizionamento marker tramite pinch	Implementato	*Hand tracking con validazione temporale*
Generazione automatica piano di taglio	Implementato	*Calcolo geometrico real-time*
Ancoraggio automatico al ginocchio reale	Parzialmente	*Posizionamento manuale funzionale*

91.7% dei requisiti completamente implementati

Figura 6.1: Verifica del soddisfacimento dei requisiti funzionali del sistema.

La verifica dei requisiti funzionali, rappresentata nella Figura 6.1, evidenzia un livello di implementazione elevato per la maggior parte delle funzionalità richieste.

6.2 Confronto con le pratiche attuali

Il confronto con le metodologie attualmente utilizzate nella chirurgia ortopedica del ginocchio permette di identificare i benefici concreti del prototipo e di posizionarlo realisticamente nel panorama delle tecnologie disponibili.

Figura 6.2: Confronto tra *workflow* tradizionale (discontinuo) e sistema proposto (integrato): eliminazione dei punti di interruzione informativa e continuità digitale dalla pianificazione all'esecuzione.

Come mostrato in Figura 6.2, il *workflow* tradizionale presenta **discontinuità nel flusso informativo**: dalla pianificazione bidimensionale su immagini statiche al trasferimento in sala operatoria tramite monitor esterni, fino all'esecuzione manuale dei tagli ossei con guide meccaniche.

Figura 6.3: Analisi comparativa di costi di implementazione e accessibilità.

Il **sistema sviluppato** introduce un paradigma integrato che elimina que-

ste discontinuità. La ricostruzione 3D automatica e la definizione interattiva dei piani garantiscono **precisione matematica**, mentre l'applicazione *visionOS* sovrappone le informazioni di pianificazione direttamente al campo operatorio, eliminando la critica fase di trasposizione mentale.

Come illustrato in Figura 6.3, il confronto economico con i sistemi robotici evidenzia differenze sostanziali: mentre *Mako* richiede investimenti di 1,25 milioni di dollari iniziali, il prototipo adotta una filosofia di *augmentation* delle capacità del chirurgo piuttosto che automazione completa, risultando accessibile a strutture con risorse limitate.

L'investimento principale del prototipo consiste nell'acquisizione di un *Apple Vision Pro* (4.000 dollari) e una *workstation macOS* (2.000 dollari), con **costi operativi praticamente nulli**. Questa differenza economica è sostanziale: mentre i sistemi robotici rimangono appannaggio di centri ospedalieri con *budget* elevati, il sistema proposto presenta costi contenuti che lo rendono accessibile a strutture sanitarie con risorse limitate.

L'analisi dei tempi operativi rivela ulteriori vantaggi. La fase preoperatoria tradizionale **richiede 30-45 minuti, ridotti a 15-20 minuti grazie alla ricostruzione automatica**. I sistemi robotici necessitano di 45-60 minuti per calibrazione e registrazione, mentre il prototipo richiede solo il tempo di caricamento del modello. Durante l'intervento, l'eliminazione della consultazione di monitor esterni può ridurre i tempi operatori di 5-10 minuti per procedure standard.

La **precisione** rappresenta l'aspetto più critico del confronto. I metodi tradizionali presentano margini di errore significativi legati alla soggettività dell'interpretazione ($\pm$3-5° in angolazione), mentre il sistema *Mako* garantisce precisione sub-millimetrica al costo di rigidità operativa. Il prototipo offre un **compromesso interessante**: precisione matematica nella definizione dei piani ($\pm$2-3°) combinata con flessibilità dell'esecuzione manuale guidata visivamente.

Il sistema potrebbe rappresentare una **soluzione intermedia** tra metodo tradizionale e sistemi robotici costosi. Pur non competendo direttamente con *Mako* per i casi più complessi, mostra il potenziale per offrire un'alternativa più accessibile per alcuni interventi di routine. L'approccio si basa sul fornire vantaggi di visualizzazione avanzata con costi significativamente ridotti rispetto ai sistemi robotici.

Questa possibile **democratizzazione della tecnologia chirurgica assistita** potrebbe, in futuro, avere impatti sulla qualità delle cure. Il prototipo esplora la fattibilità di servire il segmento di mercato attualmente non coperto: ospedali pubblici e strutture private che non possono permettersi sistemi robotici ma potrebbero beneficiare di strumenti più avanzati rispetto al metodo tradizionale. Le caratteristiche di accessibilità economica e facilità d'uso, se sviluppate ulteriormente, potrebbero favorire una maggiore

diffusione della chirurgia assistita digitalmente.

6.3 Limitazioni e criticità identificate

L'analisi critica del prototipo sviluppato evidenzia diverse limitazioni che ne condizionano l'applicabilità clinica immediata, fornendo al contempo preziose indicazioni per lo sviluppo futuro del sistema.

Il **vincolo normativo principale** risiede nell'assenza di certificazione medica dell'*Apple Vision Pro*, che limita inevitabilmente l'utilizzo del sistema a contesti sperimentali e di ricerca. Questo aspetto, già discusso nel contesto tecnologico, rappresenta una barriera fondamentale per qualsiasi implementazione clinica reale, indipendentemente dalle prestazioni tecniche raggiunte dal prototipo.

Dal punto di vista **tecnico**, le limitazioni dell'*hardware* attuale condizionano significativamente l'esperienza utente. La precisione del *hand tracking* di `ARKit` ($\pm$2mm) risulta insufficiente per applicazioni chirurgiche che richiedono accuratezza sub-millimetrica. Questa imprecisione rende la funzionalità di posizionamento *marker* tramite *pinch gesture* inadeguata per un uso operatorio reale, nonostante rappresenti un'innovazione concettuale promettente.

La tecnologia *video passthrough* del *Vision Pro* introduce criticità aggiuntive. La mediazione fotografica della realtà comporta non solo latenze percepibili, ma anche una qualità visiva che non raggiunge mai la nitidezza della percezione diretta. Questa caratteristica risulta particolarmente problematica in ambito chirurgico, dove la qualità dell'immagine è cruciale per la sicurezza del paziente.

I **piani di taglio visualizzati in realtà mista** presentano limitazioni significative. Sebbene tecnicamente funzionali, la loro rappresentazione può risultare visivamente invasiva durante l'intervento, creando occasionalmente occlusioni che ostacolano la visuale del chirurgo. Questa problematica evidenzia come la funzionalità, pur rappresentando un'innovazione tecnologica, rimanga attualmente in fase sperimentale e necessiti di sostanziali miglioramenti per risultare clinicamente utilizzabile.

L'**ergonomia prolungata** costituisce un fattore critico spesso sottovalutato. Mentre per interventi di 30-40 minuti il *Vision Pro* risulta tollerabile, l'utilizzo prolungato o multiplo durante la giornata lavorativa può causare affaticamento visivo e disagio fisico. Questo aspetto limita significativamente l'adozione in contesti operatori ad alta intensità.

La **valutazione costo-beneficio** rivela un'asimmetria importante. L'applicazione *macOS* dimostra elevata qualità e usabilità, offrendo ricostruzione 3D precisa e interfaccia intuitiva. Tuttavia, i suoi vantaggi risultano pienamente valorizzabili solo in combinazione con l'applicazione *visionOS*,

che presenta invece le limitazioni sopra descritte. Il vantaggio isolato della visualizzazione 3D rispetto alle radiografie bidimensionali, seppur apprezzabile, non giustifica da solo un cambio di *workflow* chirurgico consolidato.

Infine, l'**ancoraggio automatico** del modello virtuale all'anatomia reale rimane un requisito non implementato, richiedendo intervento manuale dell'operatore. Questa limitazione riduce l'efficienza operativa e introduce ulteriori passaggi nel *workflow* chirurgico.

Le criticità identificate non compromettono il valore del prototipo come *proof of concept*, ma delineano chiaramente le aree di sviluppo prioritarie per future implementazioni cliniche. L'applicazione *macOS* può comunque trovare utilizzo autonomo per scopi didattici, consultazione preoperatoria e comunicazione con i pazienti, mentre l'applicazione *visionOS* necessita di evoluzioni tecnologiche sostanziali prima di raggiungere maturità clinica.

Capitolo 7

Conclusioni e sviluppi futuri

Il presente lavoro di tesi ha esplorato le potenzialità della realtà mista applicata alla chirurgia ortopedica del ginocchio, sviluppando un prototipo innovativo che integra pianificazione preoperatoria e supporto intraoperatorio attraverso un sistema bimodale *macOS-visionOS*. L'obiettivo era quello di proporre una soluzione tecnologicamente avanzata ma economicamente sostenibile per supportare i chirurghi nel processo decisionale e nell'esecuzione degli interventi di protesi totale del ginocchio.

Il sistema sviluppato ha dimostrato la **fattibilità tecnica** di un approccio integrato che elimina le discontinuità tipiche del *workflow* chirurgico tradizionale. L'applicazione *macOS* per la fase preoperatoria si è rivelata particolarmente efficace, ricevendo **riscontri positivi** dai chirurghi coinvolti per la sua capacità di generare automaticamente modelli 3D di qualità clinica e permettere la definizione intuitiva dei piani di taglio.

L'implementazione dell'algoritmo *Marching Cubes* ottimizzato ha consentito la ricostruzione tridimensionale accurata delle strutture anatomiche in tempi compatibili con la pratica clinica, mentre l'interfaccia utente progettata secondo principi di semplicità e usabilità ha facilitato l'adozione da parte di professionisti con diversi livelli di competenza tecnologica.

L'applicazione *visionOS* ha introdotto funzionalità innovative nell'ambito della realtà mista applicata alla chirurgia, in particolare il sistema di posizionamento *marker* tramite *pinch gesture* e la visualizzazione di piani di taglio sovrapposti all'ambiente reale. Questi elementi rappresentano un avanzamento significativo nell'interazione naturale per applicazioni chirurgiche, benché richiedano ulteriori sviluppi per raggiungere la maturità clinica necessaria.

7.1 Contributi originali

Il lavoro presenta diversi elementi di originalità nel panorama della ricerca in chirurgia assistita. L'architettura bimodale rappresenta un **approccio innovativo** che ottimizza l'utilizzo delle risorse computazionali separando le funzionalità di elaborazione intensiva da quelle di visualizzazione immer-

siva. Questa scelta progettuale ha permesso di sfruttare i punti di forza di ciascuna piattaforma, creando un sistema che bilancia potenza computazionale e portabilità.

La filosofia di *augmentation* piuttosto che di automazione completa costituisce un paradigma originale che mantiene il controllo decisionale nelle mani del chirurgo, utilizzando la tecnologia come strumento di supporto e non di sostituzione. Questo approccio potrebbe influenzare lo sviluppo futuro di soluzioni mediche che privilegiano l'**accessibilità economica** senza compromettere l'efficacia clinica.

Dal punto di vista tecnico, la metodologia di conversione automatica tra `SceneKit` e `RealityKit` e l'implementazione del sistema di validazione temporale per i gesti in realtà mista rappresentano contributi specifici che potrebbero trovare applicazione in altri contesti di ricerca.

7.2 Sviluppi futuri e impatto a lungo termine

Le prospettive di evoluzione del sistema seguono direzioni promettenti che potrebbero portare a implementazioni clinicamente utilizzabili. L'integrazione con algoritmi di *computer vision* avanzati per la registrazione automatica modello-anatomia rappresenta la **priorità tecnica principale**, potenzialmente risolvibile attraverso tecniche di *machine learning* specificamente addestrate su *dataset* chirurgici.

Lo sviluppo di *hardware* dedicato certificato medico costituisce un percorso alternativo che richiederebbe partnership strategiche con aziende specializzate in dispositivi medicali. L'evoluzione verso sistemi collaborativi multi-utente potrebbe ampliare significativamente le applicazioni del sistema, includendo formazione remota avanzata, supporto chirurgico distribuito e integrazione diretta con sistemi ospedalieri per facilitare l'adozione clinica.

Particolare importanza riveste la necessità di **validazione clinica rigorosa** attraverso *trial* randomizzati controllati, essenziali per documentare efficacia e sicurezza secondo standard medici consolidati. Lo sviluppo di metriche quantitative per la valutazione dell'*outcome* chirurgico assistito rappresenta un'area di ricerca complementare di grande rilevanza.

Dal punto di vista dell'impatto a lungo termine, il sistema esplora un paradigma che potrebbe trasformare l'accessibilità delle tecnologie di supporto chirurgico. La riduzione drastica dei costi rispetto ai sistemi robotici tradizionali — di due ordini di grandezza — apre la possibilità di estendere i benefici della chirurgia assistita digitalmente a contesti con risorse limitate, contribuendo a una potenziale **democratizzazione della tecnologia chirurgica**.

L'approccio basato su *hardware consumer* e *software* specializzato prefigura lo sviluppo di un ecosistema di applicazioni mediche immersive che potrebbero rivoluzionare non solo la chirurgia ortopedica, ma anche altre specialità mediche. La visione a lungo termine prevede l'evoluzione verso sistemi di supporto chirurgico intelligenti che combinino realtà mista, intelligenza artificiale e robotica *soft*, mantenendo sempre il controllo umano al centro del processo decisionale.

Il contributo di questa ricerca risiede nella **dimostrazione della fattibilità tecnica** di questo approccio ibrido, fornendo una base solida per lo sviluppo di future generazioni di sistemi di supporto chirurgico più accessibili ed efficaci. Sebbene il prototipo presentato non sia ancora pronto per l'implementazione clinica diretta, esso rappresenta un passo significativo verso lo sviluppo di soluzioni innovative che potrebbero ridefinire gli standard di accessibilità e efficacia nella chirurgia ortopedica moderna.

Bibliografia

[1] Orthowood. *DPY PFC Sigma Surgical Steps*, 2014. Manuale procedure chirurgiche.

[2] Dr. Smith Orthopaedic. Total knee replacement procedure, 2022. Documentazione fotografica procedure TKA.

[3] Daniel J Berry and James A Rand. Total knee arthroplasty: clinical and functional outcomes. *The Journal of Bone and Joint Surgery*, 94(19):1839–1850, 2012.

[4] Andrew J Carr, Otto Robertsson, Stephen Graves, and Andrew J Price. Knee replacement. *The Lancet*, 367(9524):1331–1340, 2005.

[5] Kevin J Bozic et al. The epidemiology of revision total knee arthroplasty in the united states. *Clinical Orthopaedics and Related Research*, 470(1):45–51, 2012.

[6] Cost analysis of robotic knee replacement surgery. Medical Economics Research, 2024. Analisi costi sistemi robotici Mako.

[7] UT Southwestern Medical Center. Robotic knee surgery increases costs without improving outcomes. Medical Center Press Release, 2024. Consultato nel giugno 2025.

[8] R Vanzan et al. Mixed reality in surgical education: a review of current applications and future perspectives. *Journal of Surgical Education*, 80(1):123–130, 2023.

[9] Parlamento Europeo. Regolamento (ue) 2017/745 del parlamento europeo e del consiglio, 2017. Normativa europea dispositivi medici.

[10] OptoFidelity. Apple vision pro benchmark test 1: See-through latency, photon to photon. Technical report, OptoFidelity Ltd, 2024. Consultato nel giugno 2025.

[11] UC San Diego Health. Clinical trial evaluates spatial computing app on apple vision pro in operating room, 2024. Consultato nel giugno 2025.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Apple Inc. *visionOS Developer Documentation*, 2024. Consultato nel giugno 2025.

[14] European Parliament. Medical device regulation (mdr) 2017/745, 2017. EU Medical Device Regulation.

[15] Microsoft Corporation. Microsoft hololens 2 industrial edition, 2022. Versione certificata per applicazioni industriali.

[16] OFFIS e.V. *DICOM Toolkit (DCMTK)*, 2023. Consultato nel giugno 2025.

[17] Digital imaging and communications in medicine (dicom) standard. Technical standard, National Electrical Manufacturers Association, 2021. NEMA PS3 Standard.

[18] Evgeni V Chernyaev. Marching cubes 33: construction of topologically correct isosurfaces. Technical Report CN/95-17, CERN, 1995.

[19] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[20] Apple Inc. *Reality Converter Documentation*, 2023. Consultato nel giugno 2025.