



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE - DIN

**CORSO DI LAUREA MAGISTRALE IN
INGEGNERIA MECCANICA**

MODELLAZIONE VIRTUALE DEI CENTRI DI ROTAZIONE E DEL RANGE OF MOTION ARTICOLARE NEI PAZIENTI ORTOPEDICI: STUDIO E APPLICAZIONE IN AMBIENTE OPEN SOURCE

Tesi laurea magistrale in

Computer Aided Design Lab M

Relatore

Prof. Gian Maria Santi

Presentata da

Francesco Olivieri

Correlatori

Dott.ssa Grazia Chiara Menozzi

Dott. Alessandro Depaoli

Sessione Dicembre 2023

Anno Accademico 2022/2023

Indice

ABSTRACT	1
Introduzione e Diagramma di flusso	2
DIAGRAMMA DI FLUSSO	4
1. Il contesto	5
1.1. Stato dell'Arte.....	5
1.2. Scelta del software: Blender	7
1.3. Data Fusion	8
2. Scelta degli Arti Inferiori e Scalabilità	9
2.1. Collaborazione con l'Istituto Ortopedico Rizzoli	9
2.2. Rispondere a Esigenze Cliniche Specifiche	9
2.3. Prospettiva di Scalabilità	10
2.4. Vantaggi della Scelta di Blender	10
2.5. Adattabilità a Modelli Anatomici Patient Specific	10
2.6. Adattabilità a Modelli Anatomici Diversificati.....	11
3. Metodologia di Modellazione Virtuale	11
3.1. Conversione e preparazione dei dati	11
3.1.1. Conversione dei file .c3d	12
3.1.2. Verifica e Sovrapposizione con la TAC	13
3.1.3. Caricamento delle Geometrie su Blender	14
3.2. Automatizzazione del modello scheletrico tramite scripting in Blender	14
4. Scalatura e preparazione del modello scheletrico.	15
5. Definizione dei centri di rotazione articolare.....	16
5.1. Centro dell'Anca	17
5.1.1. Metodo distanza massima fra due punti di una porzione di <i>.stl</i>	19
5.2. Centro del Ginocchio	24
5.3. Centro della Caviglia	28
6. Selezione delle pose per i test fisiatrici.....	32
6.1. ANCA	32
6.1.1. Test1: ROM ADD/ABDUZIONE ANCA	32
6.1.2. Test2: ROM INTRA/EXTRA ANCA	34
6.1.3. Test3: ROM FLEX/EXT ANCA	35
7.2. GINOCCHIO.....	37

7.2.1	Test4: ROM FLEX/EXT GINOCCHIO (Anca 0°)	37
7.2.2	Test5: ROM FLEX/EXT GINOCCHIO (Anca 90°)	38
7.3.	CAVIGLIA	39
7.3.1.	Test6-7: FLESSIONE PLANTARE T-T Silverskiold Gin 0°/Gin 90°	39
7.4.	Scripting Test 1-7	41
7.4.1.	Creazione Text_1-7	42
7.4.2.	Creazione copie modello Test1-7	43
7.4.3.	Parenting Test O-7	44
7.4.4.	Messa in posa Test 1-3 ANCA	47
7.4.5.	Messa in posa Test 4-5 GINOCCHIO	50
7.4.6.	Messa in posa Test 6-7 CAVIGLIA	52
7.4.7.	Simulazione Test 1-7	53
8.	Cartella Clinica Virtuale Integrata (CCVI): Comprehensive Health Data	56
8.1.	Caratteristiche Principali della CCVI	57
8.2.	Impostazione Manuale dei Parametri Clinici	59
8.3.	Personalizzazione del Range of Motion (ROM) Articolare	60
8.4.	Presentazione Dinamica dei Dati Clinici	60
9.	Sviluppi Futuri: Automazione del Monitoraggio del Movimento Articolare	61
9.2.	Evoluzione del Rigging	61
9.3.	Interfaccia Utente Intuitiva per i Clinici	63
9.4.	Estensione Modello al Movimento degli Arti Superiori e Miglioramenti in Blender	63
9.5.	Integrazione Parte Geometrica e Clinica-Funzionale	63
9.6.	Potenziati Vantaggi: Applicazione Presso l'Istituto Ortopedico Rizzoli	63
9.7.	Affrontare Sfide e Proporre Soluzioni	64
9.8.	Rilevanza Futura nell'Ambito Ortopedico	64
	Conclusioni	65
	Appendice	66
7.4.3.a.	Parenting Test O-7	66
7.4.7.a.	Simulazione Test 1-7	73
	Ringraziamenti	75

Indice delle figure

Figura 1 File .trc importato su Blender	12
Figura 2 Verifica e Sovrapposizione con la TAC.....	13
Figura 3 Upload .stl su Blender.....	14
Figura 4 Script scalatura e unione busto	15
Figura 5 Metodo fitting sfera a testa femorale e ricerca centroidi	17
Figura 6 Script 1 Anca e visualizzazione operazioni.....	19
Figura 7 Script 2 Anca e visualizzazione operazioni.....	20
Figura 8 Metodo 3 centro dell'anca.....	21
Figura 9 Script 3 Anca.....	21
Figura 10 Script 4 Anca.....	22
Figura 11 Risultato centro di rotazione dell'Anca	23
Figura 12 Script 1 Ginocchio	24
Figura 13 Script 2 Ginocchio	25
Figura 14 Script 3-6 Ginocchio.....	26
Figura 15 Script 7 Ginocchio	27
Figura 16 Risultato centro di rotazione del Ginocchio	27
Figura 17 Script 1 Caviglia	28
Figura 18 Script 2 Caviglia	29
Figura 19 Script 3-6 Caviglia	30
Figura 20 Script 7 Caviglia	31
Figura 21 Risultato centro di rotazione della Caviglia	31
Figura 22 ADD/ABD anca.....	32
Figura 23 ROM ADD/ABD ANCA	33
Figura 24 INTRA/ EXTRA Anca	34
Figura 25 ROM INTRA/EXTRA ANCA.....	34
Figura 26 FLEX/EST anca	35
Figura 27 ROM FLEX/EXT ANCA.....	36
Figura 28 FLEX/EST ginocchio.....	37
Figura 29 ROM FLEX/EXT GINOCCHIO	38
Figura 30 ROM FLEX/EXT GINOCCHIO (anca 90°).....	38
Figura 31 FLEX PLANT T-T Silverskiold	39
Figura 32 ROM FLEX PLANTARE T-T	40
Figura 33 Script creazione copie modello Test1-7.....	42
Figura 34 Script creazione copie modello Test1_7	43
Figura 35 Risultato del Parenting	45
Figura 36 Script Parenting Modello originale	46
Figura 37 Messa in posa Test1-3	47
Figura 38 script messa in posa Test1	47
Figura 39 script messa in posa Test2	48
Figura 40script messa in posa Test3	49
Figura 41 Messa in posa Test4-5	50
Figura 42 script messa in posa Test4	50
Figura 43 script messa in posa Test5	51
Figura 44 Messa in posa Test6-7	52
Figura 45 script messa in posa Test7	52
Figura 46 Script simulazione A.....	53

<i>Figura 47 Script simulazione B</i>	54
<i>Figura 48 Script simulazione C</i>	54
<i>Figura 49 Script simulazione D</i>	54
<i>Figura 50 Script simulazione E</i>	55
<i>Figura 51 Visualizzazione CCVI</i>	56
<i>Figura 52 Impostazione parametri dei test</i>	59
<i>Figura 53 Visualizzazione angoli durante il movimento</i>	60
<i>Figura 54 Armature per Rigging</i>	61
<i>Figura 55 Dati marker .trc</i>	61
<i>Figura 56 Implementazione Cinematica inversa e gait</i>	62
<i>Figura 57 Script Parenting Test1</i>	66
<i>Figura 58 Script Parenting Test2</i>	67
<i>Figura 59 Script Parenting Test3</i>	68
<i>Figura 60 Script Parenting Test4</i>	69
<i>Figura 61 Script Parenting Test5</i>	70
<i>Figura 62 Script Parenting Test6</i>	71
<i>Figura 63 Script Parenting Test7</i>	72
<i>Figura 64 Script Simulazione 2-7 A</i>	73
<i>Figura 65 Script Simulazione 2-7 B</i>	74

ABSTRACT

Questa ricerca pionieristica nell'ambito dell'ortopedia propone un'innovativa metodologia di modellazione virtuale dei centri di rotazione articolari mediante l'integrazione di scripting in Blender. Utilizzando un ricco set di dati anonimizzati forniti dall'Istituto Ortopedico Rizzoli, la metodologia automatizza il processo dalla conversione dei dati alla creazione di modelli scheletrici virtuali, rivoluzionando l'approccio ortopedico. L'automazione riduce la dipendenza da interventi manuali, garantendo coerenza e precisione nella creazione di rappresentazioni virtuali personalizzate. La metodologia, focalizzata sull'adattabilità dinamica a nuovi dati, si propone di creare una cartella clinica 3D esaustiva e sempre aggiornabile, fornendo un supporto visivo fondamentale per decisioni ortopediche informate. Oltre a rispondere alle attuali esigenze della pratica ortopedica, questa ricerca anticipa le sfide future, posizionando la modellazione virtuale come un elemento essenziale nella guida diagnostica e nella pianificazione dei trattamenti ortopedici.

Introduzione e Diagramma di flusso

Nel panorama dell'ortopedia contemporanea, la modellazione virtuale dei centri di rotazione articolari emerge come un elemento cruciale, destinato a rivoluzionare l'approccio alla pratica ortopedica. Questo studio si propone di automatizzare il complesso processo di creazione di modelli scheletrici virtuali attraverso l'impiego di avanzate tecniche di modellazione tridimensionale. Dal trattamento iniziale dei dati fino all'analisi dettagliata dei parametri clinici, questa metodologia, basata sull'integrazione di scripting in Blender, mira a instaurare coerenza, efficienza, precisione e ripetibilità in tutte le fasi del processo di modellazione.

L'Istituto Ortopedico Rizzoli, in collaborazione con Unibo dal 2018, rappresenta la fonte primaria, fornendo un set ricco di dati ANONIMI o PSEUDOANONIMIZZATI, tra cui file .stl con le geometrie delle ossa, immagini DICOM della TAC, e dati raccolti tramite telemetria e padana dinamometrica (.C3D). Un caso clinico specifico funge da tramite, aprendo la strada verso una possibile standardizzazione più ampia.

Parallelamente, nel contesto dell'Istituto Ortopedico Rizzoli, si è sviluppata la Cartella Clinica Virtuale Integrata (CCVI). Questo avanzato sistema di gestione dati rappresenta un punto centrale in cui convergono una vasta gamma di informazioni cliniche di un paziente. La CCVI integra dati attuali e passati provenienti da esami clinici, test fisiatrici, esami di imaging (come TAC) e dati storici del paziente, offrendo un ambiente 3D per la visualizzazione e l'analisi dettagliata.

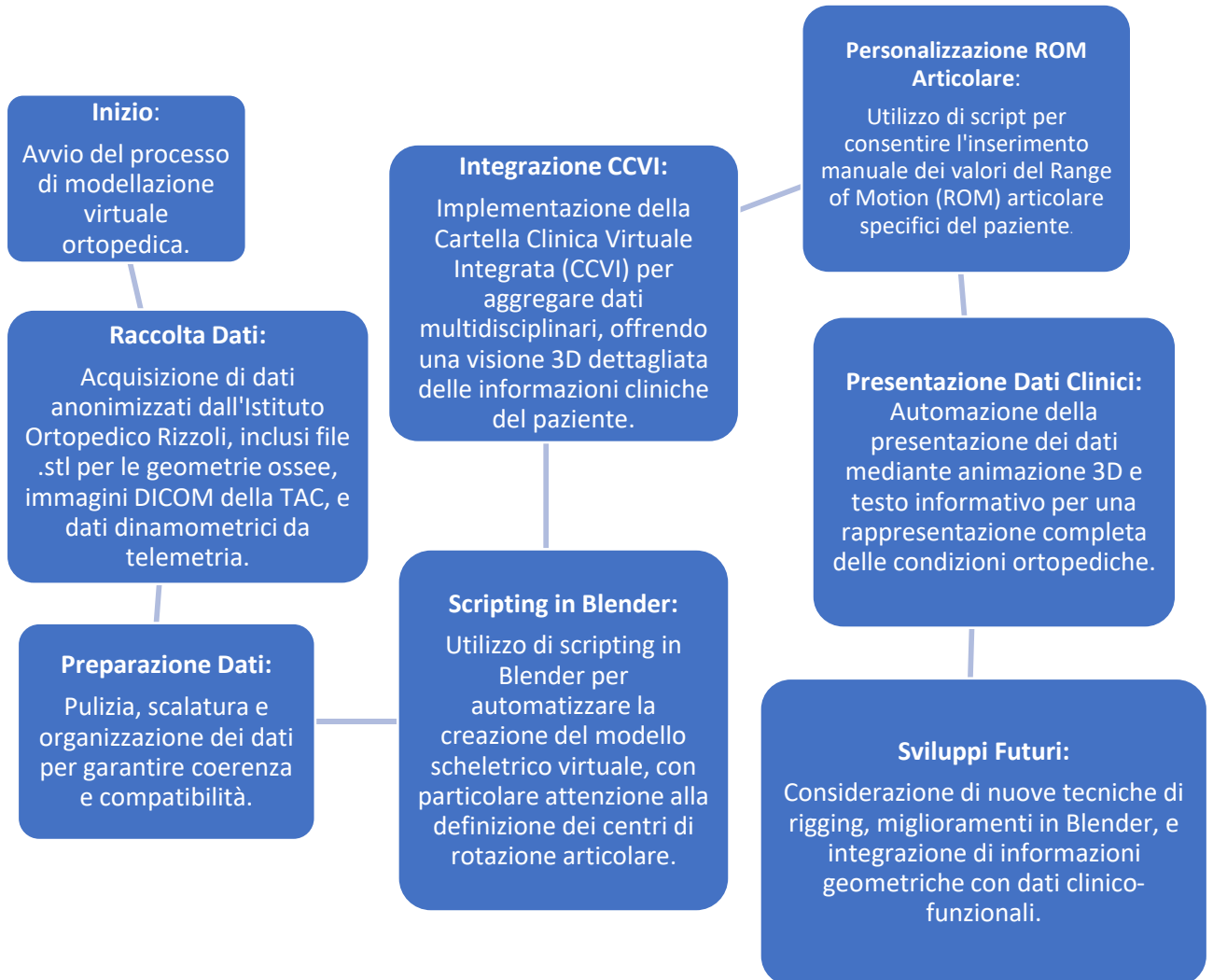
L'implementazione di scripting in Blender non solo semplifica le operazioni di pulizia, scalatura e definizione dei centri articolari ma rappresenta un autentico passo avanti verso la creazione di modelli scheletrici virtuali altamente accurati e personalizzati. L'automazione di tali processi diventa essenziale data la crescente complessità dei dati ortopedici e l'ampio utilizzo di modelli virtuali.

Questo lavoro si inserisce come un contributo all'espansione di un progetto in corso presso l'Istituto Ortopedico Rizzoli. L'obiettivo è raggiungere una modellazione virtuale in grado di adattarsi dinamicamente a nuove informazioni e integrare fluidamente dati aggiuntivi emersi durante il trattamento del paziente. La creazione di una cartella clinica 3D esaustiva e facilmente aggiornabile costituisce la finalità ultima, fornendo un supporto visivo e informativo fondamentale per le decisioni ortopediche.

In questo contesto, l'implementazione di nuove informazioni diventa un tratto distintivo, conferendo alla metodologia una natura in continua evoluzione. L'approccio di scripting in Blender è concepito per accogliere e integrare nuovi dati nella struttura del modello scheletrico esistente, rendendo la metodologia non solo attuale ma pronta a rispondere alle esigenze in costante mutamento della pratica ortopedica.

In sintesi, questa introduzione sottolinea l'importanza di una modellazione virtuale avanzata e automatizzata, offrendo una prospettiva chiara e ambiziosa delle potenzialità della ricerca. La metodologia proposta non solo risponde alle attuali esigenze di precisione e personalizzazione nell'ambito ortopedico, ma prospetta una visione futura in cui la modellazione virtuale diventa un pilastro di riferimento in continua evoluzione.

DIAGRAMMA DI FLUSSO



1. Il contesto

1.1. Stato dell'Arte

Attualmente, il panorama delle metodologie e dei software dedicati alla gestione dei movimenti nei modelli scheletrici è vasto e variegato. Numerosi lavori e applicazioni sono stati sviluppati per analizzare e simulare i movimenti del corpo umano. Tuttavia, emerge una tendenza comune: molte di queste soluzioni, pur efficaci in determinati contesti, presentano limitazioni significative in termini di flessibilità e adattabilità.

Uno degli ostacoli principali riscontrati nella letteratura scientifica è la rigidità di molte piattaforme software. Spesso, tali sistemi sono progettati con una struttura sequenziale e rigidamente collegati a specifici passaggi o procedure. Questo approccio può rivelarsi limitante quando si affrontano scenari complessi o quando è necessaria un'adattabilità dinamica alle esigenze di ricerca.

Inoltre, molte soluzioni esistenti sono focalizzate su determinati aspetti del movimento o su particolari articolazioni, sacrificando la completezza e la flessibilità. Questo approccio frammentato può risultare poco pratico per coloro che cercano di affrontare questioni multidimensionali o desiderano esplorare diverse parti del corpo umano in modo integrato.

L'evoluzione delle tecnologie e delle metodologie di ricerca richiede una soluzione che vada oltre le limitazioni attuali. La necessità di una piattaforma flessibile, in grado di adattarsi dinamicamente ai requisiti specifici della ricerca ortopedica e biomeccanica, è sempre più evidente.

In questo contesto, emerge la rilevanza di un approccio che integri la potenza del software open-source con una metodologia altamente flessibile e modulare. Blender si posiziona in modo unico in questo scenario, offrendo non solo un ambiente accessibile e collaborativo, ma anche una vasta gamma di funzionalità di scripting che consentono di modellare e gestire le diverse tipologie di dati provenienti da fuori Blender, che essendo Open Source, permette un'implementazione personalizzata e adattabile tramite Data Fusion.

In conclusione, sebbene esistano numerose soluzioni per la gestione modelli multicorpo, la ricerca di un approccio completo, flessibile e integrato è ancora in corso. L'adozione di strumenti come Blender rappresenta una risposta innovativa a questa sfida, offrendo una piattaforma aperta e flessibile che si adatta alle mutevoli esigenze della ricerca biomeccanica e ortopedica.

1.2. Scelta del software: Blender

Alcuni software noti per l'analisi del movimento (motion analysis) includono:

1. Vicon

- Utilizzato per la motion capture e l'analisi del movimento in vari settori, come biomeccanica e animazione.

2. OptiTrack

- Offre soluzioni di motion capture per una vasta gamma di applicazioni, inclusa l'analisi del movimento umano e animale.

3. MotionAnalysis

- Specializzato in sistemi di motion capture per applicazioni biomeccaniche e sportive.

4. Qualisys

- Fornisce sistemi di motion capture 3D per l'analisi del movimento, ampiamente utilizzati in ricerca e industria.

5. SIMI Reality Motion Systems

- Si concentra su soluzioni di motion capture per applicazioni mediche, sportive e industriali.

6. C-Motion (Visual3D)

- Utilizzato per l'analisi biomeccanica, offre strumenti per elaborare dati di motion capture e generare risultati dettagliati.

7. Kinovea

- Più leggero, è un software open-source per l'analisi del movimento che può essere utilizzato per scopi educativi e sportivi.

8. OpenSim

- Piattaforma open-source utilizzata per modellare e simulare il movimento umano. Può essere integrato con dati di motion capture.

9. Blender

- Oltre a essere un software di modellazione 3D, Blender offre funzionalità di animazione e scripting che possono essere utilizzate per l'analisi del movimento.

1.3. Data Fusion

La scelta di utilizzare Blender come piattaforma principale per l'analisi del movimento è stata motivata dalla sua flessibilità derivante dalla potente combinazione di funzionalità di scripting e strumenti avanzati di animazione. Mentre altri software offrono soluzioni valide che spaziano dall'gait analysis alla biomeccanica e fino all'animazione, Blender si distingue per la sua capacità di adattarsi alle esigenze specifiche del progetto e tiene una porta aperta verso tutte le soluzioni offerte dai diversi programmi sopra citati. La possibilità di implementare script personalizzati consente una maggiore personalizzazione, consentendo di sviluppare funzioni uniche e visualizzazioni personalizzate, come la rappresentazione delle forze attraverso vettori dinamici che variano in direzione e modulo nel tempo, l'imposizione di forze esterne nell'ambiente, l'analisi di variazioni di dati e visualizzazione tramite grafici.

Questa flessibilità apre nuove opportunità per l'analisi dettagliata e la rappresentazione accurata dei dati biomeccanici, supportando l'approccio innovativo proposto in questa ricerca.

Inoltre, la scelta di Blender è motivata dalla sua capacità di semplificare la modifica delle geometrie e l'implementazione di operazioni aggiuntive. La flessibilità offerta consente di adattare facilmente le geometrie o di introdurre nuove operazioni, garantendo un approccio agile alle necessità del progetto. La struttura aperta di Blender facilita ulteriori miglioramenti e ottimizzazioni dello scripting nel tempo, rendendo possibile standardizzare il processo e consentendo una maggiore adattabilità per future operazioni simili.

2. Scelta degli Arti Inferiori e Scalabilità

La scelta di concentrarsi sugli arti inferiori come focus principale di questo studio è stata guidata dalla collaborazione stretta con l'Istituto Ortopedico Rizzoli, un'eccellenza nel campo ortopedico. Questa decisione è stata influenzata dal desiderio di allineare la ricerca agli obiettivi clinici specifici dell'Istituto e di affrontare le problematiche cliniche associate agli arti inferiori.

2.1. Collaborazione con l'Istituto Ortopedico Rizzoli

La partnership con l'Istituto Ortopedico Rizzoli ha rappresentato un punto chiave nel delineare il focus sugli arti inferiori. Grazie alla vasta esperienza e ai dati clinici ANONIMI raccolti ed elaborati dall'Istituto, che esegue l'esame della gait, TAC ed elabora le geometrie, questa scelta è stata plasmata dalla volontà di affrontare direttamente le esigenze e le sfide ortopediche presenti nella pratica clinica quotidiana.

2.2. Rispondere a Esigenze Cliniche Specifiche

Le patologie ortopediche associate agli arti inferiori costituiscono una parte significativa delle condizioni affrontate dall'Istituto Ortopedico Rizzoli. La scelta di concentrarsi su quest'area permette di adattare direttamente la ricerca alle esigenze cliniche esistenti, garantendo che l'analisi sia strettamente allineata agli obiettivi medici specifici.

2.3. Prospettiva di Scalabilità

La metodologia sviluppata è stata concepita con la scalabilità in mente. La focalizzazione sugli arti inferiori prevede una futura espansione per includere il modello scheletrico completo. Questa prospettiva di crescita riflette la volontà di adattare la metodologia agli arti superiori, mantenendo un approccio modulare e flessibile.

2.4. Vantaggi della Scelta di Blender

La scelta di utilizzare Blender come ambiente principale per lo sviluppo è stata motivata dal fatto che era già usato presso l'Istituto Ortopedico Rizzoli per la sua natura open-source, dalla vasta comunità di sviluppatori e dalle potenti capacità di scripting. La flessibilità offerta da Blender consente di integrare dati provenienti da diverse fonti, sfruttando appieno la metodologia di data fusion, e di adattare il workflow in base alle specifiche esigenze della ricerca ortopedica.

2.5. Adattabilità a Modelli Anatomici Patient Specific

L'adozione di Blender come piattaforma per la creazione di modelli specifici del paziente va oltre la convenienza finanziaria, rappresentando una scelta strategica mirata a massimizzare la flessibilità e la personalizzazione nel processo di modellazione virtuale. I modelli patient specific non sono semplici rappresentazioni geometriche, ma integrano anche aspetti clinico-funzionali, consentendo modifiche immediate e l'implementazione di funzioni personalizzate. Blender si distingue per la sua versatilità, permettendo di adattarsi alle mutevoli esigenze della ricerca e di incorporare progressi futuri. La sua utilità si estende

oltre la mera creazione di modelli, posizionandosi come una risorsa fondamentale per esplorare in profondità le complessità delle condizioni ortopediche attraverso una modellazione virtuale avanzata e altamente personalizzabile.

2.6. Adattabilità a Modelli Anatomici Diversificati

Nonostante la focalizzazione sugli arti inferiori, la metodologia è intrinsecamente adattabile per affrontare una vasta gamma di modelli anatomici. L'approccio modulare dello scripting consente l'implementazione su articolazioni e segmenti superiori, aprendo le porte a un'applicazione più ampia nei contesti ortopedici e biomeccanici.

3. Metodologia di Modellazione Virtuale

3.1. Conversione e preparazione dei dati

Il processo di acquisizione e preparazione dei dati dall'Istituto Ortopedico Rizzoli ha seguito una serie di fasi cruciali. Dopo aver ricevuto i dati, compresi i file *.stl* delle geometrie ossee, gli *.stl* dei reperi anatomici, le immagini DICOM della TAC e i file *.c3d* contenenti le posizioni temporali dei markers e altre informazioni, il processo è stato orchestrato come segue:

3.1.1. Conversione dei file .c3d

I dati provenienti dai file .c3d sono stati convertiti in due formati chiave:

- Il formato .mot, che registra le informazioni sul movimento.
- Il formato .trc, contenente le traiettorie spaziali dei markers.

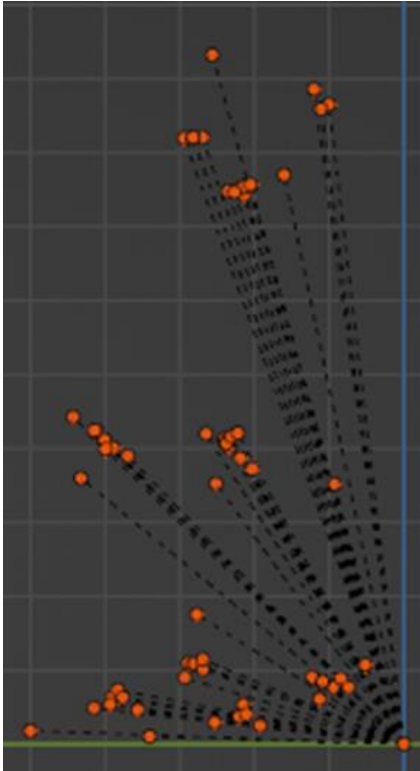


Figura 1 File .trc importato su Blender

Per effettuare la conversione dei file .c3d nei formati .trc e .mot, è stato utilizzato uno script preimpostato su MATLAB denominato "c3dExport". Questo script ha consentito di estrarre e organizzare le informazioni cruciali dai dati di movimento, rendendoli pronti per le fasi successive del processo.

3.1.2. Verifica e Sovrapposizione con la TAC

Un'accurata verifica è stata eseguita per garantire la coesione e l'acquisizione corretta delle informazioni durante il caricamento. In particolare, è stata eseguita una sovrapposizione delle geometrie scheletriche e dei reperi anatomici su Blender con le immagini TAC del paziente, ottenute a partire dai file DICOM. Questo controllo è stato cruciale per assicurare l'allineamento preciso delle strutture virtuali con l'anatomia reale del paziente.

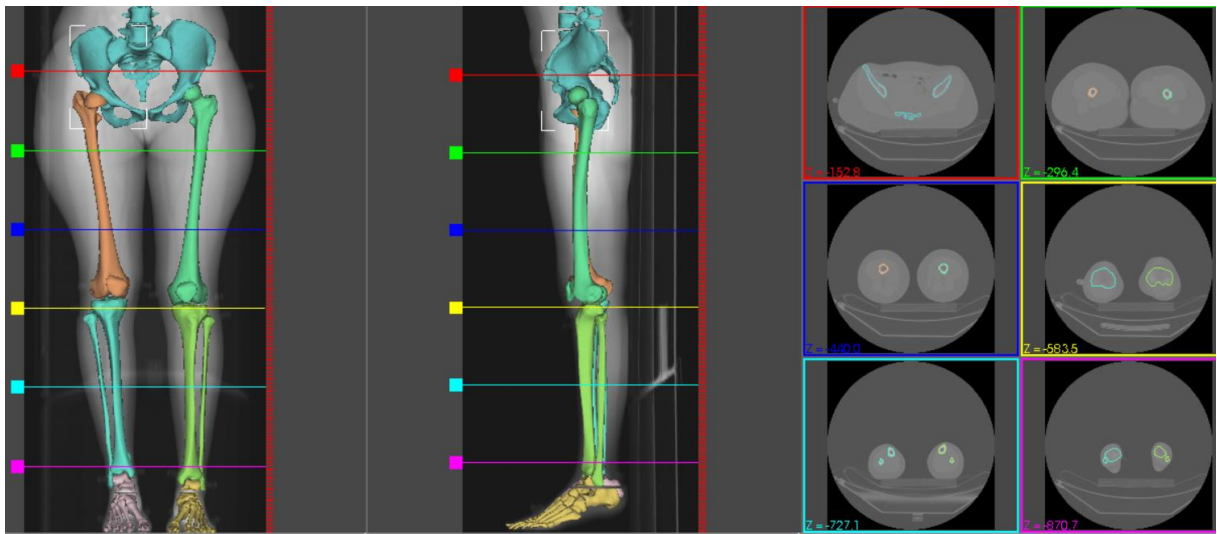


Figura 2 Verifica e Sovrapposizione con la TAC

Questo approccio dettagliato e metodico ha contribuito a garantire la qualità e l'integrità dei dati, preparandoli per le fasi successive del processo di modellazione virtuale.

3.1.3. Caricamento delle Geometrie su Blender



Le geometrie scheletriche e dei reperi anatomici in formato .stl sono state caricate in ambiente Blender. Questa fase è stata fondamentale per garantire la corretta rappresentazione virtuale delle strutture anatomiche del paziente.

Figura 3 Upload .stl su Blender

3.2. Automatizzazione del modello scheletrico tramite scripting in Blender

Dopo la fase di caricamento e preparazione dei dati, l'attenzione si è focalizzata sull'automatizzazione del modello scheletrico mediante l'implementazione di script in Blender. Questo approccio mira a standardizzare il più accuratamente possibile il processo di pulizia, scalatura, definizione dei centri articolari (RoM) e parenting dei segmenti ossei tramite relazioni genitore-figlio.

L'adozione di script, soprattutto nella definizione automatica dei centri di rotazione basati sulle porzioni di geometrie ossee, per automatizzare queste fasi cruciali della modellazione virtuale, considerando anche la scala iniziale dei dati, non solo aumenta l'efficienza del processo, ma riduce anche la dipendenza da interventi manuali, garantendo una maggiore coerenza e precisione nei risultati finali. Questo approccio rappresenta un passo significativo verso la realizzazione di modelli scheletrici virtuali altamente accurati e personalizzati e facilita il lavoro di visualizzazione desiderato.

4. Scalatura e preparazione del modello scheletrico.

L'automatizzazione della scalatura dello scheletro è stata implementata attraverso uno script dedicato, considerando che i file *.stl* forniti erano in scala 1000:1.

Questo script, basandosi su parametri specifici dei dati forniti in ingresso, garantisce una scalatura accurata delle geometrie scheletriche, senza che vi siano alterazioni dell'insieme di geometrie presenti.

Dopo la scalatura, lo script analizza attentamente i dati iniziali per identificare eventuali suddivisioni del bacino in segmenti separati (cerca "BUSTO" per vedere se ce n'è solo uno o è diviso in rachidi + colonna).

Una volta rilevata la suddivisione del bacino, lo script procede automaticamente con l'unione di questi segmenti in un unico busto che verrà salvato con gli stessi prefissi e suffissi dei file in ingresso.

L'automazione di questo passaggio non solo semplifica il processo complessivo, ma migliora anche la precisione della modellazione scheletrica, assicurando che il busto risultante rifletta accuratamente l'anatomia del paziente. Questo approccio automatizzato si adatta dinamicamente alle variazioni nei dati iniziali, garantendo una rappresentazione virtuale coesa e fedele dell'apparato scheletrico, visto che nei test implementati la posizione del bacino era mantenuta ferma.

```
1 import bpy
2
3 # Unisci gli oggetti con "BUSTO" nel nome in un unico oggetto "BUSTO"
4 busto_objects = [obj for obj in bpy.context.scene.objects if "BUSTO" in obj.name and obj.type == 'MESH']
5 if busto_objects:
6     bpy.context.view_layer.objects.active = busto_objects[0]
7     bpy.ops.object.select_all(action='DESELECT')
8     for obj in busto_objects:
9         obj.select_set(True)
10    bpy.ops.object.join()
11    bpy.context.object.name = "SA_BUSTO"
12
13 # Definisci il fattore di scala e la percentuale di semplificazione
14 scale_factor = 0.1 # Modifica il fattore di scala a tuo piacimento
15 decimate_ratio = 0.2 # Modifica la percentuale di semplificazione a tuo piacimento
16
17 # Applica la scala a tutti gli oggetti nella scena
18 for obj in bpy.context.scene.objects:
19     if obj.type == 'MESH':
20         obj.scale *= scale_factor
21
22 # Applica la decimazione a tutti gli oggetti nella scena
23 for obj in bpy.context.scene.objects:
24     if obj.type == 'MESH':
25         bpy.context.view_layer.objects.active = obj
26         bpy.ops.object.modifier_add(type='DECIMATE')
27         bpy.context.object.modifiers["Decimate"].ratio = decimate_ratio
28         bpy.ops.object.modifier_apply(modifier="Decimate")
29
30 print("Operazioni completate: unione, scala e decimazione.")
```

Figura 4 Script scalatura e unione busto

5. Definizione dei centri di rotazione articolare.

I centri di rotazione articolare sono punti virtuali intorno ai quali avviene il movimento di rotazione di un'articolazione. Ogni articolazione del corpo umano ha il suo centro di rotazione specifico, determinato dalle caratteristiche anatomiche e strutturali di quell'articolazione. Questi centri sono fondamentali per comprendere il movimento delle articolazioni e sono utilizzati in biomeccanica per analizzare il funzionamento del sistema muscolo-scheletrico.

Il centro di rotazione di un'articolazione può variare a seconda del tipo di movimento che si sta considerando. Ad esempio, per l'articolazione dell'anca, il centro di rotazione può essere differente per i movimenti di flessione, estensione, adduzione o abduzione. Inoltre, questi centri possono variare da persona a persona a causa delle differenze anatomiche individuali.

Nell'ambito della modellazione virtuale del movimento articolare, individuare con precisione i centri di rotazione è cruciale per ottenere rappresentazioni accurate e realistiche. Questi centri influenzano direttamente il comportamento dei modelli virtuali durante la simulazione dei movimenti, e la loro corretta definizione è essenziale per garantire risultati biomeccanicamente validi.

È stata implementata un'approfondita metodologia basata sull'uso di script specifici. Questi script sono stati sviluppati per interagire con porzioni altamente specifiche delle geometrie ossee, garantendo un posizionamento estremamente accurato dei centri articolari.

Il processo automatico si è distinto per la sua capacità di analizzare in modo dettagliato le caratteristiche anatomiche delle ossa coinvolte. Attraverso un'analisi approfondita, gli script sono stati in grado di identificare con precisione i punti centrali di rotazione, evitando la necessità di interventi manuali, soprattutto per utenti che non hanno dimestichezza con la modellazione.

L'utilizzo di questi script ha rappresentato un passo significativo verso l'automatizzazione del processo, garantendo una coerenza e una precisione notevolmente elevate nella determinazione dei centri di rotazione articolari. Questo approccio avanzato ha non solo ottimizzato il tempo necessario per eseguire questa della modellazione virtuale, ma è anche riproducibile per diversi modelli.

5.1. Centro dell'Anca

Nel corso della ricerca volta alla modellazione virtuale dei centri di rotazione articolari, sono stati esplorati diversi approcci al fine di ottenere risultati precisi e ripetibili. Tra i vari metodi sperimentati, tre si sono distinti per la loro efficacia e rilevanza nell'identificazione accurata del centro dell'articolazione dell'anca.

- **Il primo metodo** ha coinvolto l'adattamento di una sfera alla porzione superiore del 15% dell'.*stl*, offrendo un approccio iniziale per definire il centro dell'articolazione.
- La scelta del 15% è stata possibile grazie alla evidente similarità in scala dei diversi segmenti ossei di diversi soggetti generici.

In particolare, permette di rilevare la distanza desiderata, ovvero dal punto sulla superficie della testa femorale più vicino al baricentro del busto fino al punto più esterno del grande trocantere femorale.

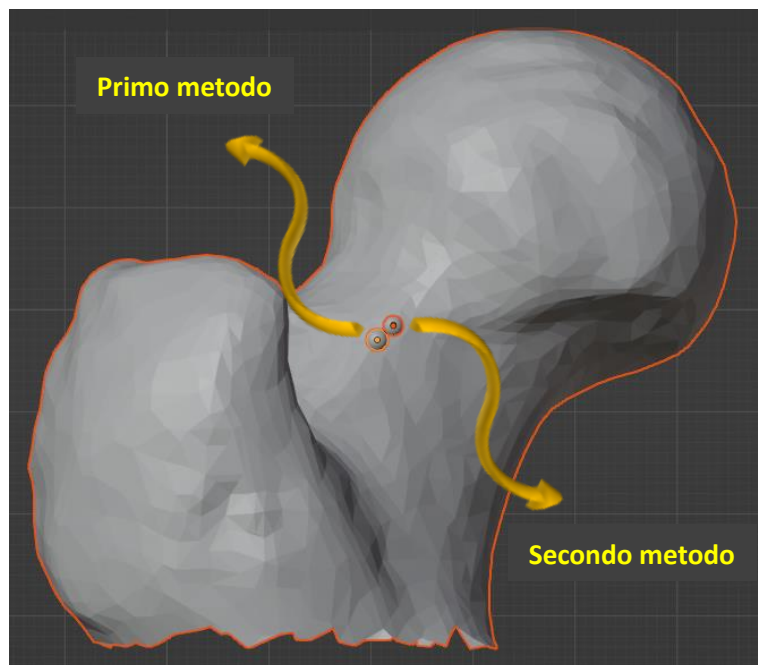


Figura 5 Metodo fitting sfera a testa femorale e ricerca centroidi

- **Il secondo metodo** ha abbracciato la ricerca dei centroidi.

I centri di rotazione articolare sono punti virtuali intorno ai quali avviene il movimento di rotazione di un'articolazione. Ogni articolazione del corpo umano ha il suo centro di rotazione specifico, determinato dalle caratteristiche anatomiche e strutturali di quell'articolazione. Questi centri sono fondamentali per comprendere il movimento delle articolazioni e sono utilizzati in biomeccanica per analizzare il funzionamento del sistema muscolo-scheletrico.

Il centro di rotazione di un'articolazione può variare a seconda del tipo di movimento che si sta considerando. Ad esempio, per l'articolazione dell'anca, il centro di rotazione può essere

differente per i movimenti di flessione, estensione, adduzione o abduzione. Inoltre, questi centri possono variare da persona a persona a causa delle differenze anatomiche individuali.

Nell'ambito della modellazione virtuale del movimento articolare, individuare con precisione i centri di rotazione è cruciale per ottenere rappresentazioni accurate e realistiche. Questi centri influenzano direttamente il comportamento dei modelli virtuali durante la simulazione dei movimenti, e la loro corretta definizione è essenziale per garantire risultati biomeccanicamente validi.

Purtroppo, il risultato che si è riusciti ad ottenere era anche solo visivamente inaccettabile.

- È stato il **terzo metodo**, basato sulla ricerca dei due punti a distanza massima (sempre nella porzione superiore del 15% dell'.*stl*) per individuare la retta sulla quale si trova il centro dell'anca, a emergere come il più preciso e ripetibile, soprattutto andando a visualizzare visivamente presenza più o meno importante di compenetrazioni con il segmento osseo adiacente durante il suo movimento. Questo approccio si è dimostrato notevolmente superiore nella sua capacità di definire con precisione il centro di rotazione articolare, fornendo risultati coerenti con la fisiologia umana.

Nel contesto di questa tesi, ci concentreremo principalmente su questo terzo metodo, presentandolo come il cuore della nostra metodologia. L'approfondita analisi di questa tecnica avanzata rivelerà non solo la sua superiorità in termini di precisione ma anche la sua applicabilità pratica e la sua efficacia nel contesto della modellazione virtuale delle articolazioni.

5.1.1. Metodo distanza massima fra due punti di una porzione di *.stl*

La precisione nella definizione del centro dell'articolazione dell'anca è cruciale per una modellazione virtuale accurata.

- **Script 1: Duplicazione e Rimozione di Vertici Inferiori**

Questo script è progettato per trattare modelli STL di femori. Inizialmente, esegue la duplicazione degli oggetti STL originali ("SA_FEMORE SX" e "SA_FEMORE DX") creando copie distinte, indicate dai nomi "_copia". Successivamente, calcola la soglia Z per la rimozione dei vertici, basandosi sulla parte inferiore del 15% dell'*.stl* originale. Vengono selezionati ed eliminati i vertici che si trovano al di sotto di questa soglia per ottimizzare la geometria:

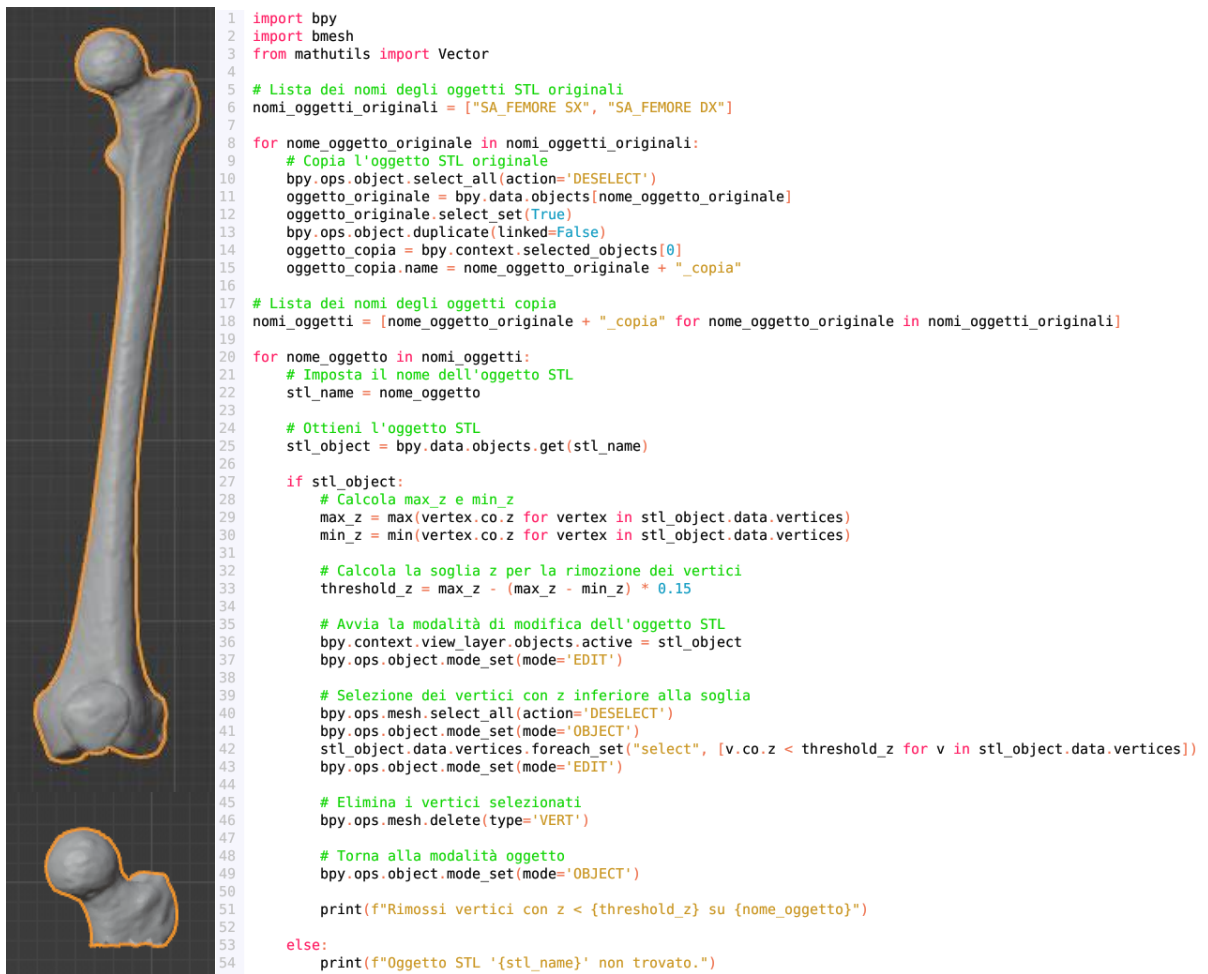


Figura 6 Script 1 Anca e visualizzazione operazioni

- **Script 2: Ricerca di Punti a Distanza Massima e Creazione di Sfere**

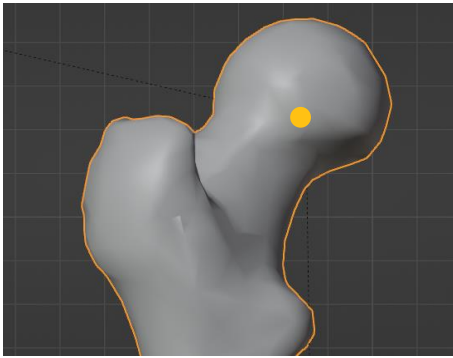
Questo script opera sugli oggetti STL duplicati ("SA_FEMORE DX_copia" e "SA_FEMORE SX_copia"). Inizialmente, calcola i punti all'interno dell'oggetto che presentano la massima distanza tra loro. Successivamente, calcola i punti intermedi lungo questa distanza e crea sfere su questi punti. Infine, mantiene solo la seconda sfera creata, offrendo un punto di riferimento centrale.

In entrambi gli script, viene fornito un riscontro costante attraverso stampe a console, indicando quali azioni vengono eseguite e su quali oggetti.

Questi passaggi preparano gli oggetti STL per ulteriori fasi di modellazione virtuale, posizionando punti chiave e rimuovendo elementi non essenziali., tramite l'impiego di script dedicati, individuamo due punti nella porzione di .stl che si trovano alla massima distanza tra loro. Utilizzando questo segmento, generiamo cinque sfere equidistanti tra loro:



Figura 7 Script 2 Anca e visualizzazione operazioni



Di particolare interesse è la seconda sfera, selezionata a partire da destra o sinistra, a seconda del femore considerato.

Figura 8 Metodo 3 centro dell'anca

- **Script 3-4: Posizionamento del Corsore 3D, Impostazione dell'Origine e Rimozione**

Il terzo script si occupa dello spostamento del cursore 3D al centro delle sfere precedentemente create, nonché dell'impostazione del nuovo origine degli oggetti STL "SA_FEMORE DX" e "SA_FEMORE SX" su questo cursore. Ciò consente di stabilire un nuovo punto di riferimento per gli oggetti STL, facilitando il successivo processo di modellazione e analisi.

```

103 import bpy
104
105 def sposta_cursore_3d_al_centro(nomeoggetto):
106     oggetto = bpy.data.objects.get(nomeoggetto)
107     if oggetto:
108         bpy.context.scene.cursor.location = oggetto.location
109         print(f"Cursore 3D spostato al centro di {nomeoggetto}")
110     else:
111         print(f"Oggetto '{nomeoggetto}' non trovato.")
112
113 def imposta_origine_al_cursore_3d(nomeoggetto):
114     oggetto = bpy.data.objects.get(nomeoggetto)
115     if oggetto:
116         bpy.context.view_layer.objects.active = oggetto
117         oggetto.select_set(True)
118         bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
119         oggetto.select_set(False) # Deseleziona l'oggetto dopo l'operazione
120         print(f"Origine di {nomeoggetto} impostata al cursore 3D")
121     else:
122         print(f"Oggetto '{nomeoggetto}' non trovato.")
123
124 # Esegui le operazioni per "Sphere.001" e "SA_FEMORE DX"
125 sposta_cursore_3d_al_centro("Sphere.001")
126 imposta_origine_al_cursore_3d("SA_FEMORE DX")
127
128 # Esegui le operazioni per "Sphere.005" e "SA_FEMORE SX"
129 sposta_cursore_3d_al_centro("Sphere.005")
130 imposta_origine_al_cursore_3d("SA_FEMORE SX")

```

Figura 9 Script 3 Anca

- **Pulizia ambiente di lavoro**

L'ultimo script elimina gli oggetti intermedi creati durante il processo, tra cui le copie duplicate degli STL e le sfere. Questo passo è importante per mantenere un ambiente di lavoro pulito e concentrarsi solo sugli oggetti essenziali per la modellazione virtuale.

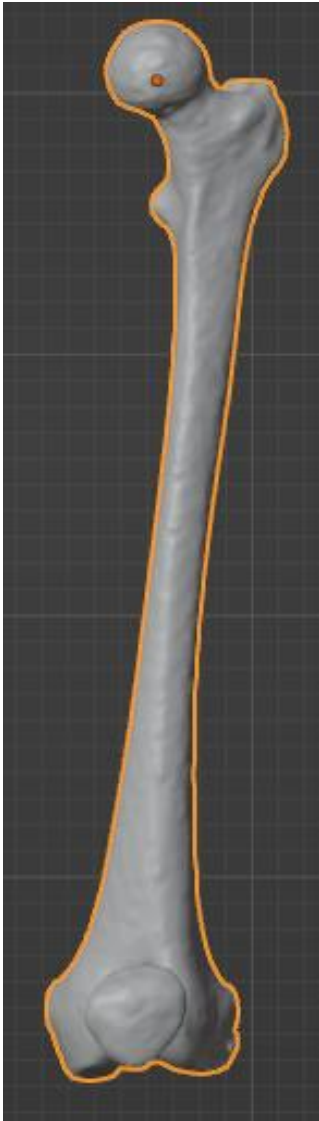
```
135 import bpy
136
137 oggetti_da_cancellare = [
138     "SA_FEMORE DX_copia", "SA_FEMORE SX_copia",
139     "Sphere.001",
140     "Sphere.002",
141     "Sphere.003",
142     "Sphere.004",
143     "Sphere.005",
144     "Sphere.006",
145     "Sphere.007",
146     "Sphere.008"
147 ]
148
149 # Deseleziona tutti gli oggetti
150 bpy.ops.object.select_all(action='DESELECT')
151
152 for nomeoggetto in oggetti_da_cancellare:
153     oggetto = bpy.data.objects.get(nomeoggetto)
154     if oggetto:
155         # Seleziona l'oggetto prima di eliminarlo
156         oggetto.select_set(True)
157     else:
158         print(f"Oggetto '{nomeoggetto}' non trovato.")
159
160 # Elimina gli oggetti selezionati
161 bpy.ops.object.delete()
162
163 print("Oggetti eliminati con successo.")
```

Figura 10 Script 4 Anca

Questi script rappresentano una sequenza di azioni mirate a preparare gli oggetti STL per un'analisi accurata, definendo il centro di rotazione e ottimizzando il modello per ulteriori elaborazioni.

Questa operazione ci restituisce il femore, ora con il suo centro accuratamente definito nella testa femorale .

La precisa selezione e posizionamento del centro dell'articolazione dell'anca nel femore fornisce una base solida e affidabile per ulteriori analisi e rappresentazioni virtuali.



Questo processo meticoloso *assicura che il centro dell'articolazione dell'anca sia accuratamente definito nella testa femorale, consentendo di ottenere dati e misurazioni precise per valutare il movimento articolare, la biomeccanica e altre caratteristiche cliniche.

Una volta stabilito il centro dell'articolazione dell'anca, è possibile utilizzarlo come punto di riferimento per una serie di applicazioni nel campo dell'ortopedia virtuale.

Ad esempio, consente di definire e quantificare l'angolo di movimento articolare (ROM) dell'anca in diversi piani di movimento, identificare eventuali asimmetrie o limitazioni nella gamma di movimento e valutare l'efficacia dei programmi di riabilitazione o interventi chirurgici.

Inoltre, la definizione accurata del centro dell'articolazione dell'anca è essenziale per la creazione di modelli anatomici virtuali realistici e personalizzati.

Figura 11 Risultato centro di rotazione dell'Anca

Questi modelli possono essere utilizzati per scopi educativi, ad esempio per illustrare agli studenti o ai pazienti le caratteristiche anatomiche dell'articolazione dell'anca e i relativi movimenti. Inoltre, possono essere impiegati per simulazioni virtuali, training chirurgico virtuale o pianificazione preoperatoria, consentendo agli ortopedici di visualizzare, manipolare e valutare il femore virtuale in modo dettagliato e preciso.

Complessivamente, l'accurata definizione del centro dell'articolazione dell'anca nel femore rappresenta un passo fondamentale nell'ambito della modellazione virtuale. Fornisce una base solida per un'analisi approfondita, una valutazione precisa e una rappresentazione virtuale dettagliata dell'articolazione dell'anca, contribuendo a migliorare la comprensione e il trattamento delle condizioni ortopediche.

5.2. Centro del Ginocchio

- **Script 1: Duplicazione e Rimozione di Vertici Superiori**

```
1 import bpy
2 import bmesh
3 from mathutils import Vector
4
5 # Lista dei nomi degli oggetti STL originali
6 nomi_objetti_originali = ["SA_FEMORE SX", "SA_FEMORE DX"]
7
8 for nome_objetto_originale in nomi_objetti_originali:
9     # Copia l'oggetto STL originale
10    bpy.ops.object.select_all(action='DESELECT')
11    oggetto_originale = bpy.data.objects[nome_objetto_originale]
12    oggetto_originale.select_set(True)
13    bpy.ops.object.duplicate(linked=False)
14    oggetto_copia = bpy.context.selected_objects[0]
15    oggetto_copia.name = nome_objetto_originale + "_copia"
16
17 # Lista dei nomi degli oggetti copia
18 nomi_objetti = [nome_objetto_originale + "_copia" for nome_objetto_originale in nomi_objetti_originali]
19
20 for nome_objetto in nomi_objetti:
21     # Imposta il nome dell'oggetto STL
22     stl_name = nome_objetto
23
24     # Ottieni l'oggetto STL
25     stl_object = bpy.data.objects.get(stl_name)
26
27     if stl_object:
28         # Calcola max_z e min_z
29         max_z = max(vertex.co.z for vertex in stl_object.data.vertices)
30         min_z = min(vertex.co.z for vertex in stl_object.data.vertices)
31
32         # Calcola la soglia z per la rimozione dei vertici
33         threshold_z = max_z - (max_z - min_z) * 0.85
34
35         # Avvia la modalità di modifica dell'oggetto STL
36         bpy.context.view_layer.objects.active = stl_object
37         bpy.ops.object.mode_set(mode='EDIT')
38
39         # Selezione dei vertici con z superiore alla soglia
40         bpy.ops.mesh.select_all(action='DESELECT')
41         bpy.ops.object.mode_set(mode='OBJECT')
42         stl_object.data.vertices.foreach_set("select", [v.co.z > threshold_z for v in stl_object.data.vertices])
43         bpy.ops.object.mode_set(mode='EDIT')
44
45         # Elimina i vertici selezionati
46         bpy.ops.mesh.delete(type='VERT')
47
48         # Torna alla modalità oggetto
49         bpy.ops.object.mode_set(mode='OBJECT')
50
51         print(f"Rimossi vertici con z < {threshold_z} su {nome_objetto}")
52
53     else:
54         print(f"Oggetto STL '{stl_name}' non trovato.")
```

Figura 12 Script 1 Ginocchio

Questo script duplica gli oggetti STL originali relativi ai femori ("SA_FEMORE SX" e "SA_FEMORE DX"), creando copie distinte con nomi modificati aggiungendo "_copia". Successivamente, calcola la soglia Z per la rimozione dei vertici, basandosi sulla parte superiore dell'85% dell'.stl originale. Vengono selezionati ed eliminati i vertici che si trovano al di sopra di questa soglia per ottimizzare la geometria.

- **Script 2: Calcolo del Punto con Massima Densità e Creazione di Sfere**

Questo script opera sugli oggetti STL duplicati ("SA_FEMORE DX_copia" e "SA_FEMORE SX_copia"). Calcola la dimensione massima dell'oggetto STL e definisce una distanza massima da considerare (20% della dimensione massima). Per ogni vertice dell'oggetto, calcola la densità dei punti sulla superficie entro la distanza specificata. Trova il punto con la massima densità e calcola un punto interno dell'oggetto STL. Infine, crea una sfera nel punto interno con un raggio di 0.1 e la rinomina in base all'oggetto originale:

```

59 import bpy
60 import bmesh
61 from mathutils import Vector
62
63 # Lista dei nomi degli oggetti STL
64 nomi_oggetti = ["SA_FEMORE SX_copia", "SA_FEMORE DX_copia"]
65
66 for nome_oggetto in nomi_oggetti:
67     # Ottieni un riferimento all'oggetto STL
68     oggetto = bpy.data.objects.get(nome_oggetto)
69
70     if oggetto:
71         # Ottieni il mesh dell'oggetto
72         mesh = oggetto.data
73
74         # Crea un oggetto BMesh
75         bm = bmesh.new()
76         bm.from_mesh(mesh)
77
78         # Calcola la dimensione massima dell'oggetto STL
79         dimensione_massima = max(oggetto.dimensions)
80
81         # Calcola la distanza massima da considerare (20% della dimensione massima)
82         distanza_massima = 0.2 * dimensione_massima
83
84         # Variabili per memorizzare il punto con la massima densità
85         massima_densita = 0
86         punto_massima_densita = None
87
88         # Itera attraverso i vertici dell'oggetto
89         for vertice in bm.verts:
90             punto = oggetto.matrix_world @ vertice.co
91             densita = 0
92
93             # Calcola la densità dei punti sulla superficie
94             for altro_vertice in bm.verts:
95                 altro_punto = oggetto.matrix_world @ altro_vertice.co
96                 distanza = (punto - altro_punto).length
97
98                 if distanza < distanza_massima:
99                     densita += 1
100
101             if densita > massima_densita:
102                 massima_densita = densita
103                 punto_massima_densita = punto
104
105         # Calcola il punto all'interno dell'STL
106         punto_interno = Vector()
107         for vertice in bm.verts:
108             punto = oggetto.matrix_world @ vertice.co
109             if punto != punto_massima_densita:
110                 punto_interno += punto
111
112         punto_interno /= len(bm.verts) - 1
113
114         # Crea una sfera nel punto interno con il nome appropriato
115         if nome_oggetto == "SA_FEMORE SX_copia":
116             nome_sfera = "Sfera_SX"
117         elif nome_oggetto == "SA_FEMORE DX_copia":
118             nome_sfera = "Sfera_DX"
119
120         bpy.ops.mesh.primitive_uv_sphere_add(radius=0.1, location=punto_interno)
121         bpy.context.active_object.name = nome_sfera
122

```

Figura 13 Script 2 Ginocchio

- **Script 3-6: Posizionamento del 3D Cursor e Impostazione dell'Origine**

```

137 import bpy
138
139 # Deseleziona tutti gli oggetti
140 bpy.ops.object.select_all(action='DESELECT')
141
142 # Imposta il nome dell'oggetto "Sfera_DX"
143 nome_sfera_dx = "Sfera_DX"
144
145 # Ottieni l'oggetto "Sfera_DX"
146 sfera_dx = bpy.data.objects.get(nome_sfera_dx)
147
148 if sfera_dx:
149     # Sposta il cursore 3D al centro di "Sfera_DX"
150     bpy.context.scene.cursor.location = sfera_dx.location
151
152     print("Il cursore 3D è stato spostato al centro di 'Sfera_DX'.")
153
154 else:
155     print("Oggetto 'Sfera_DX' non trovato.")
156
157 import bpy
158
159 # Imposta il nome dell'oggetto "SA_FEMORE DX"
160 nome_sa_femore_dx = "SA_GAMBA DX"
161
162 # Ottieni l'oggetto "SA_FEMORE DX"
163 sa_femore_dx = bpy.data.objects.get(nome_sa_femore_dx)
164
165 if sa_femore_dx:
166     # Seleziona "SA_FEMORE DX"
167     sa_femore_dx.select_set(True)
168
169     # Imposta il nuovo origine di "SA_FEMORE DX" sul cursore 3D
170     bpy.context.view_layer.objects.active = sa_femore_dx
171     bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
172
173     print("Il nuovo origine di 'SA_GAMBA DX' è stato impostato sul 3D cursor.")
174 else:
175     print("Oggetto 'SA_FEMORE DX' non trovato.")
176
177
178 |
179 import bpy
180
181 # Deseleziona tutti gli oggetti
182 bpy.ops.object.select_all(action='DESELECT')
183
184 # Imposta il nome dell'oggetto "Sfera_SX"
185 nome_sfera_sx = "Sfera_SX"
186
187 # Ottieni l'oggetto "Sfera_SX"
188 sfera_sx = bpy.data.objects.get(nome_sfera_sx)
189
190 if sfera_sx:
191     # Sposta il cursore 3D al centro di "Sfera_SX"
192     bpy.context.scene.cursor.location = sfera_sx.location
193
194     print("Il cursore 3D è stato spostato al centro di 'Sfera_SX'.")
195
196 else:
197     print("Oggetto 'Sfera_SX' non trovato.")
198
199 import bpy
200
201 # Imposta il nome dell'oggetto "SA_FEMORE SX"
202 nome_sa_femore_sx = "SA_GAMBA SX"
203
204 # Ottieni l'oggetto "SA_FEMORE SX"
205 sa_femore_sx = bpy.data.objects.get(nome_sa_femore_sx)
206
207 if sa_femore_sx:
208     # Seleziona "SA_FEMORE SX"
209     sa_femore_sx.select_set(True)
210
211     # Imposta il nuovo origine di "SA_FEMORE SX" sul cursore 3D
212     bpy.context.view_layer.objects.active = sa_femore_sx
213     bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
214
215     print("Il nuovo origine di 'SA_GAMBA SX' è stato impostato sul 3D cursor.")
216 else:
217     print("Oggetto 'SA_FEMORE SX' non trovato.")
218
219

```

Figura 14 Script 3-6 Ginocchio

Questi script si occupano di spostare il cursore 3D al centro delle sfere create ("Sfera_DX" e "Sfera_SX") e quindi impostare l'origine degli oggetti delle gambe ("SA_GAMBA DX" e "SA_GAMBA SX") sul cursore 3D.

- **Script 7: Eliminazione degli Oggetti Temporanei e Salvataggio**

Questo script deseleziona tutti gli oggetti, elimina gli oggetti temporanei (copia degli STL e sfere) e salva il file principale.

```
229 import bpy
230
231 # Deseleziona tutti gli oggetti
232 bpy.ops.object.select_all(action='DESELECT')
233
234 oggetti_da_cancellare = ["SA_FEMORE DX_copia", "SA_FEMORE SX_copia", "Sfera_DX", "Sfera_SX"]
235
236 for nome_oggetto in oggetti_da_cancellare:
237     oggetto = bpy.data.objects.get(nome_oggetto)
238     if oggetto:
239         # Seleziona l'oggetto prima di eliminarlo
240         oggetto.select_set(True)
241         bpy.ops.object.delete()
242         print(f"Eliminato '{nome_oggetto}'.")
243     else:
244         print(f"Oggetto '{nome_oggetto}' non trovato o già eliminato.")
245
246 # Per confermare l'eliminazione
247 bpy.ops.wm.save_mainfile()
```

Figura 15 Script 7 Ginocchio

In ogni script, le stampe a console forniscono riscontro costante, indicando le azioni eseguite e i progressi. Questi script sono progettati per preparare gli oggetti STL per le fasi successive della modellazione virtuale.

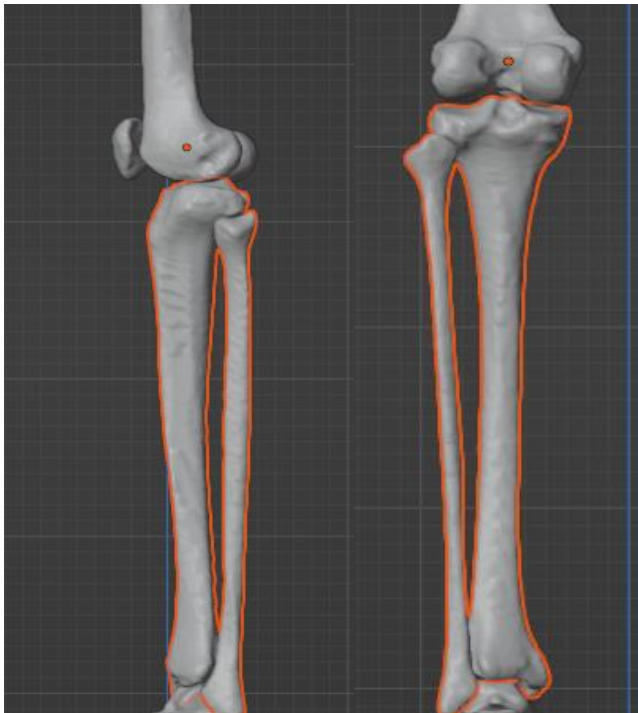


Figura 16 Risultato centro di rotazione del Ginocchio

5.3. Centro della Caviglia

- **Script 1: Duplicazione e Rimozione di Vertici Superiori**

Questo script è simile al primo set di script che hai fornito, ma agisce sugli oggetti STL relativi alle gambe ("SA_GAMBA SX" e "SA_GAMBA DX"). Duplica gli oggetti originali, calcola la soglia Z per la rimozione dei vertici, e rimuove i vertici che si trovano al di sopra di questa soglia.

```
1 import bpy
2 import bmesh
3 from mathutils import Vector
4
5 # Lista dei nomi degli oggetti STL originali
6 nomi_oggetti_originali = ["SA_GAMBA SX", "SA_GAMBA DX"]
7
8 for nome_oggetto_originale in nomi_oggetti_originali:
9     # Copia l'oggetto STL originale
10    bpy.ops.object.select_all(action='DESELECT')
11    oggetto_originale = bpy.data.objects[nome_oggetto_originale]
12    oggetto_originale.select_set(True)
13    bpy.ops.object.duplicate(linked=False)
14    oggetto_copia = bpy.context.selected_objects[0]
15    oggetto_copia.name = nome_oggetto_originale + "_copia"
16
17 # Lista dei nomi degli oggetti copia
18 nomi_oggetti = [nome_oggetto_originale + "_copia" for nome_oggetto_originale in nomi_oggetti_originali]
19
20 for nome_oggetto in nomi_oggetti:
21     # Imposta il nome dell'oggetto STL
22     stl_name = nome_oggetto
23
24     # Ottieni l'oggetto STL
25     stl_object = bpy.data.objects.get(stl_name)
26
27     if stl_object:
28         # Calcola max_z e min_z
29         max_z = max(vertex.co.z for vertex in stl_object.data.vertices)
30         min_z = min(vertex.co.z for vertex in stl_object.data.vertices)
31
32         # Calcola la soglia z per la rimozione dei vertici
33         threshold_z = max_z - (max_z - min_z) * 0.85
34
35         # Avvia la modalità di modifica dell'oggetto STL
36         bpy.context.view_layer.objects.active = stl_object
37         bpy.ops.object.mode_set(mode='EDIT')
38
39         # Selezione dei vertici con z superiore alla soglia
40         bpy.ops.mesh.select_all(action='DESELECT')
41         bpy.ops.object.mode_set(mode='OBJECT')
42         stl_object.data.vertices.foreach_set("select", [v.co.z > threshold_z for v in stl_object.data.vertices])
43         bpy.ops.object.mode_set(mode='EDIT')
44
45         # Elimina i vertici selezionati
46         bpy.ops.mesh.delete(type='VERT')
47
48         # Torna alla modalità oggetto
49         bpy.ops.object.mode_set(mode='OBJECT')
50
51         print(f"Rimossi vertici con z < {threshold_z} su {nome_oggetto}")
52
53     else:
54         print(f"Oggetto STL '{stl_name}' non trovato.")
55
56
```

Figura 17 Script 1 Caviglia

- **Script 2: Calcolo del Punto con Massima Densità e Creazione di Sfere**

Anche questo script è simile al secondo set di script precedente ma agisce sugli oggetti STL duplicati relativi alle gambe ("SA_GAMBA DX_copia" e "SA_GAMBA SX_copia"). Calcola il punto con la massima densità sulla superficie dell'oggetto, determina un punto interno dell'oggetto e crea una sfera nel punto interno con un raggio di 0.1. La sfera viene quindi rinominata in base all'oggetto originale.

```

59 import bpy
60 import bmesh
61 from mathutils import Vector
62
63 # Lista dei nomi degli oggetti STL
64 nomi_oggetti = ["SA_GAMBA SX_copia", "SA_GAMBA DX_copia"]
65
66 for nome_oggetto in nomi_oggetti:
67     # Ottieni un riferimento all'oggetto STL
68     oggetto = bpy.data.objects.get(nome_oggetto)
69
70     if oggetto:
71         # Ottieni il mesh dell'oggetto
72         mesh = oggetto.data
73
74         # Crea un oggetto BMesh
75         bm = bmesh.new()
76         bm.from_mesh(mesh)
77
78         # Calcola la dimensione massima dell'oggetto STL
79         dimensione_massima = max(oggetto.dimensions)
80
81         # Calcola la distanza massima da considerare (20% della dimensione massima)
82         distanza_massima = 0.2 * dimensione_massima
83
84         # Variabili per memorizzare il punto con la massima densità
85         massima_densita = 0
86         punto_massima_densita = None
87
88         # Itera attraverso i vertici dell'oggetto
89         for vertice in bm.verts:
90             punto = oggetto.matrix_world @ vertice.co
91             densita = 0
92
93             # Calcola la densità dei punti sulla superficie
94             for altro_vertice in bm.verts:
95                 altro_punto = oggetto.matrix_world @ altro_vertice.co
96                 distanza = (punto - altro_punto).length
97
98                 if distanza < distanza_massima:
99                     densita += 1
100
101                 if densita > massima_densita:
102                     massima_densita = densita
103                     punto_massima_densita = punto
104
105             # Calcola il punto all'interno dell'STL
106             punto_interno = Vector()
107             for vertice in bm.verts:
108                 punto = oggetto.matrix_world @ vertice.co
109                 if punto != punto_massima_densita:
110                     punto_interno += punto
111
112             punto_interno /= len(bm.verts) - 1
113
114             # Crea una sfera nel punto interno con il nome appropriato
115             if nome_oggetto == "SA_GAMBA SX_copia":
116                 nome_sfera = "Sfera_SX"
117             elif nome_oggetto == "SA_GAMBA DX_copia":
118                 nome_sfera = "Sfera_DX"
119
120             bpy.ops.mesh.primitive_uv_sphere_add(radius=0.1, location=punto_interno)
121             bpy.context.active_object.name = nome_sfera
122

```

Figura 18 Script 2 Caviglia

- **Script 3-6: Posizionamento del 3D Cursor e Impostazione dell'Origine**

```

137 import bpy
138
139 # Deseleziona tutti gli oggetti
140 bpy.ops.object.select_all(action='DESELECT')
141
142 # Imposta il nome dell'oggetto "Sfera_DX"
143 nome_sfera_dx = "Sfera_DX"
144
145 # Ottieni l'oggetto "Sfera_DX"
146 sfera_dx = bpy.data.objects.get(nome_sfera_dx)
147
148 if sfera_dx:
149     # Sposta il cursore 3D al centro di "Sfera_DX"
150     bpy.context.scene.cursor.location = sfera_dx.location
151
152     print("Il cursore 3D è stato spostato al centro di 'Sfera_DX'.")
153
154 else:
155     print("Oggetto 'Sfera_DX' non trovato.")
156
157 import bpy
158
159 # Imposta il nome dell'oggetto "SA_FEMORE DX"
160 nome_sa_femore_dx = "SA_PIEDE DX"
161
162 # Ottieni l'oggetto "SA_FEMORE DX"
163 sa_femore_dx = bpy.data.objects.get(nome_sa_femore_dx)
164
165 if sa_femore_dx:
166     # Seleziona "SA_FEMORE DX"
167     sa_femore_dx.select_set(True)
168
169     # Imposta il nuovo origine di "SA_FEMORE DX" sul cursore 3D
170     bpy.context.view_layer.objects.active = sa_femore_dx
171     bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
172
173     print("Il nuovo origine di 'SA_PIEDE DX' è stato impostato sul 3D cursor.")
174 else:
175     print("Oggetto 'SA_PIEDE DX' non trovato.")
176
177 import bpy
178
179 # Deseleziona tutti gli oggetti
180 bpy.ops.object.select_all(action='DESELECT')
181
182
183 # Imposta il nome dell'oggetto "Sfera_SX"
184 nome_sfera_sx = "Sfera_SX"
185
186 # Ottieni l'oggetto "Sfera_SX"
187 sfera_sx = bpy.data.objects.get(nome_sfera_sx)
188
189 if sfera_sx:
190     # Sposta il cursore 3D al centro di "Sfera_SX"
191     bpy.context.scene.cursor.location = sfera_sx.location
192
193     print("Il cursore 3D è stato spostato al centro di 'Sfera_SX'.")
194
195 else:
196     print("Oggetto 'Sfera_SX' non trovato.")
197
198 import bpy
199
200 # Imposta il nome dell'oggetto "SA_FEMORE SX"
201 nome_sa_femore_sx = "SA_PIEDE SX"
202
203 # Ottieni l'oggetto "SA_FEMORE SX"
204 sa_femore_sx = bpy.data.objects.get(nome_sa_femore_sx)
205
206 if sa_femore_sx:
207     # Seleziona "SA_FEMORE SX"
208     sa_femore_sx.select_set(True)
209
210     # Imposta il nuovo origine di "SA_FEMORE SX" sul cursore 3D
211     bpy.context.view_layer.objects.active = sa_femore_sx
212     bpy.ops.object.origin_set(type='ORIGIN_CURSOR')
213
214     print("Il nuovo origine di 'SA_PIEDE SX' è stato impostato sul 3D cursor.")
215 else:
216     print("Oggetto 'SA_PIEDE SX' non trovato.")

```

Figura 19 Script 3-6 Caviglia

Questi script gestiscono il posizionamento del cursore 3D al centro delle sfere create ("Sfera_DX" e "Sfera_SX") e l'impostazione dell'origine degli oggetti dei piedi ("SA_PIEDE DX" e "SA_PIEDE SX") sul cursore 3D

- **Script 7: Eliminazione degli Oggetti Temporanei e Salvataggio**

Deseleziona tutti gli oggetti, elimina gli oggetti temporanei (copia degli STL e sfere) e salva il file principale. In generale, questi script sembrano duplicare, modificare la geometria, creare sfere, posizionare il 3D Cursor e impostare origini per oggetti STL specifici. Gli oggetti temporanei vengono eliminati alla fine, e il file principale viene salvato.

```
226 import bpy
227
228 # Deseleziona tutti gli oggetti
229 bpy.ops.object.select_all(action='DESELECT')
230
231 oggetti_da_cancellare = ["SA_GAMBA_DX_copia", "SA_GAMBA_SX_copia", "Sfera_DX", "Sfera_SX"]
232
233 for nome_oggetto in oggetti_da_cancellare:
234     oggetto = bpy.data.objects.get(nome_oggetto)
235     if oggetto:
236         # Seleziona l'oggetto prima di eliminarlo
237         oggetto.select_set(True)
238         bpy.ops.object.delete()
239         print(f"Eliminato '{nome_oggetto}'.")
240     else:
241         print(f"Oggetto '{nome_oggetto}' non trovato o già eliminato.")
242
```

Figura 20 Script 7 Caviglia



Figura 21 Risultato centro di rotazione della Caviglia

6. Selezione delle pose per i test fisiatrici.

L'utilizzo di script è implementato per posizionare automaticamente il modello scheletrico nella specifica posa del test fisiatrico di interesse. Questo approccio automatizzato non solo risparmia tempo, ma garantisce una precisione anatomica nella rappresentazione della posa, facilitando ulteriori analisi ortopediche.

Le pose virtualizzate sono quelle in cui la posizione del bacino viene mantenuta ferma dal fisiatra.

6.1. ANCA

6.1.1. Test1: ROM ADD/ABDUZIONE ANCA

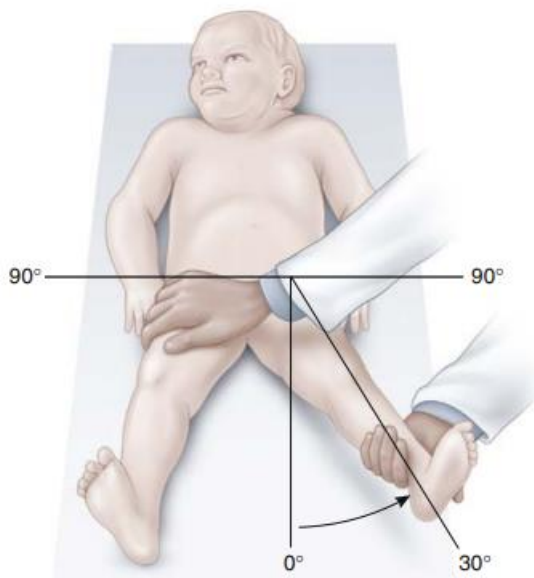


Figura 22 ADD/ABD anca

Dal "Texas Scottish Rite Hospital for Children", 6th edition

L'obiettivo dell'esame è misurare l'ampiezza del movimento di adduzione e abduzione dell'anca per valutare la mobilità dell'articolazione. Questo test è spesso parte di una valutazione più ampia della funzione muscolo-scheletrica e può fornire indicazioni sulla flessibilità e la stabilità dell'anca.

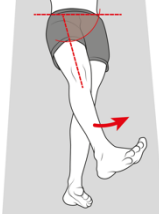
ROM e manovre per la valutazione				
ROM				
ADDUZ/ABDUZ	20°	25°	Pz supino, angolo tra la linea inter-SIAS e l'asse della coscia.	
ANCA				

Figura 23 ROM ADD/ABD ANCA

Il paziente è solitamente posizionato supino su un lettino o una superficie piana con entrambe le gambe distese.

L'operatore stabilisce un riferimento anatomico, ad esempio, la linea tra le spine iliache anteriori superiori (SIAS) che rappresentano i punti ossei sopra l'inguine.

Per l'ADDUZIONE, il ginocchio viene gradualmente portato verso l'altro ginocchio, riducendo l'angolo tra la linea inter-SIAS e l'asse della coscia. La misurazione viene eseguita in gradi.

Per l'ABDUZIONE, il ginocchio viene portato nella direzione opposta, aumentando l'angolo tra la linea inter-SIAS e l'asse della coscia. Anche in questo caso, la misurazione viene eseguita in gradi.

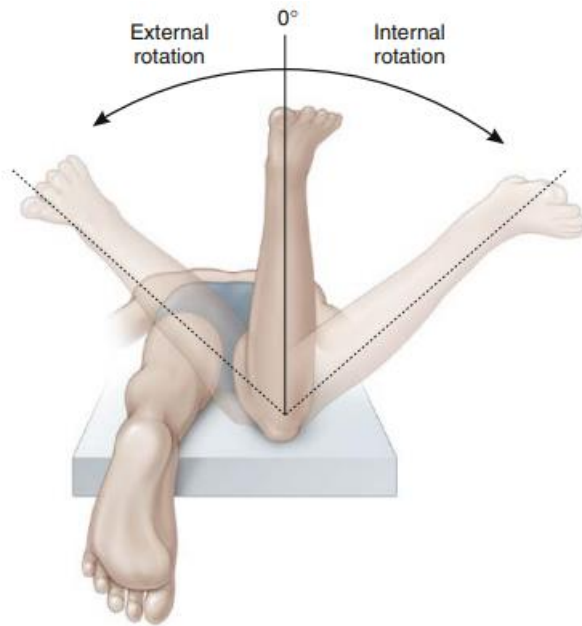
L'esame fornisce informazioni sull'ampiezza del movimento di adduzione e abduzione dell'anca. Utile nella valutazione di condizioni come limitazioni di movimento, instabilità dell'anca o possibili tensioni muscolari.

L'esame dovrebbe essere eseguito con attenzione per evitare lesioni o disagio per il paziente.

I valori di riferimento possono variare in base all'età, al sesso e ad altri fattori individuali.

Come sempre, la conduzione di questo tipo di esami dovrebbe essere effettuata da professionisti medici qualificati.

6.1.2. Test2: ROM INTRA/EXTRA ANCA



L'obiettivo dell'esame è misurare l'ampiezza del movimento di intra-rotazione ed extra-rotazione dell'anca per valutare la mobilità dell'articolazione. Questo test è significativo nella valutazione della funzione muscolo-scheletrica e può fornire indicazioni sulla stabilità e la flessibilità dell'anca.

Figura 24 INTRA/ EXTRA Anca

Dal "Texas Scottish Rite Hospital for Children", 6th edition

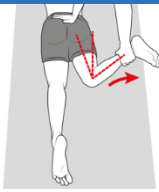
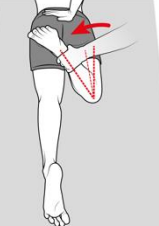
ROM e manovre per la valutazione					
ROM ANCA	INTRA	35°	50°	Pz prono, bacino simmetrico, ginocchio flesso 90°.	
ROM ANCA	EXTRA	20°	45°	Pz prono, bacino simmetrico, ginocchio flesso 90°.	

Figura 25 ROM INTRA/EXTRA ANCA

Il paziente è posizionato prono (a faccia in giù) con il bacino in posizione simmetrica e il ginocchio flesso a 90°.

Il paziente è prono con il bacino in una posizione simmetrica e il ginocchio flesso a un angolo di 90°.

Per la ****INTRA-ROTAZIONE****, il ginocchio viene gradualmente ruotato verso l'interno, riducendo l'angolo di rotazione dell'anca.

Per l'****EXTRA-ROTAZIONE****, il ginocchio viene ruotato nella direzione opposta, aumentando l'angolo di rotazione dell'anca. Entrambe le misurazioni vengono eseguite in gradi.

L'esame fornisce informazioni sull'ampiezza del movimento di intra-rotazione ed extra-rotazione dell'anca.

Utile nella valutazione di condizioni come limitazioni di movimento, instabilità dell'anca o possibili tensioni muscolari.

L'esame dovrebbe essere eseguito con attenzione per evitare lesioni o disagio per il paziente.

I valori di riferimento possono variare in base all'età, al sesso e ad altri fattori individuali.

6.1.3. Test3: ROM FLEX/EXT ANCA



Figura 26 FLEX/EST anca

Dal "Texas Scottish Rite Hospital for Children", 6th edition

L'esame di flessione ed estensione dell'anca è un test utilizzato per valutare l'ampiezza del movimento articolare nell'articolazione dell'anca. Questo test è spesso parte di una valutazione più ampia della funzione muscolo-scheletrica e può essere eseguito in diverse posizioni e condizioni.

L'obiettivo è misurare l'ampiezza del movimento di flessione ed estensione dell'anca per valutare la funzionalità e l'elasticità dell'articolazione.

Il paziente è solitamente posizionato supino su un lettino o una superficie piana.

Il paziente viene fatto sdraiare sulla schiena con entrambe le gambe distese.

L'operatore può stabilizzare l'anca opposta per evitare movimenti indesiderati durante il test.

Per la flessione dell'anca, il ginocchio viene gradualmente portato verso il torace, portando l'anca in flessione. La misurazione viene eseguita in gradi.

Per l'estensione dell'anca, il ginocchio viene portato nella direzione opposta, estendendo l'anca nella direzione posteriore. Anche in questo caso, la misurazione viene eseguita in gradi.

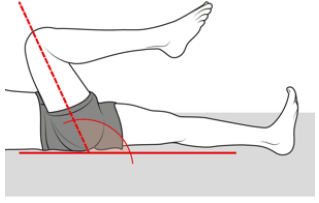
ROM e manovre per la valutazione				
Anca	DX	SN	Note	Immagini
ROM FLEX/EXT	100°	140°	Pz supino	

Figura 27 ROM FLEX/EXT ANCA

L'esame fornisce informazioni sull'ampiezza del movimento articolare nell'articolazione dell'anca.

Può essere utilizzato per valutare la presenza di limitazioni o dolore durante la flessione o l'estensione dell'anca.

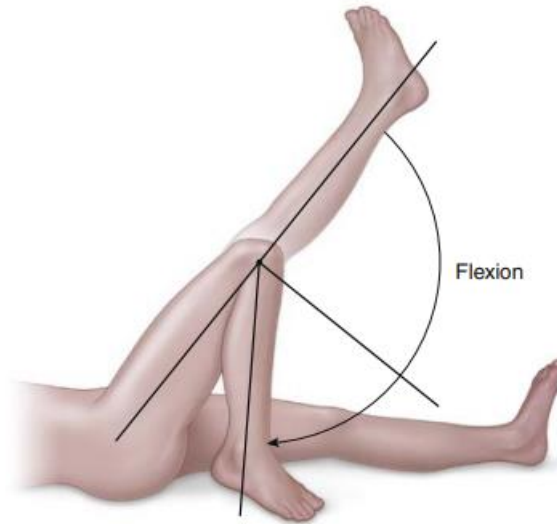
Utile nella valutazione di condizioni muscolo-scheletriche, post-operatorie o post-traumatiche.

L'esame dovrebbe essere eseguito con attenzione per evitare lesioni o disagio per il paziente.

La comparazione dei dati ottenuti durante l'esame può essere fatta con i valori di riferimento normali.

Si noti che la procedura specifica può variare in base alle esigenze del paziente e alla pratica clinica. È sempre consigliabile eseguire questo tipo di esami sotto la supervisione di professionisti medici qualificati.

7.2. GINOCCHIO



L'obiettivo è misurare l'angolo di flessione ed estensione del ginocchio per valutare la mobilità dell'articolazione. Questo test è essenziale nella valutazione della funzione muscolo-scheletrica e può fornire indicazioni sulla flessibilità e la stabilità del ginocchio.

Figura 28 FLEX/EST ginocchio

Dal "Texas Scottish Rite Hospital for Children", 6th edition

7.2.1 Test4: ROM FLEX/EXT GINOCCHIO (Anca 0°)

Il paziente è prono (a faccia in giù), e l'angolo è misurato tra l'asse del femore e l'asse del ginocchio.

Il paziente è in posizione prona.

L'angolo di flessione del ginocchio è misurato considerando l'asse del femore e l'asse del ginocchio.

Valuta l'ampiezza del movimento di flessione del ginocchio.

Utile nella valutazione di condizioni come limitazioni di movimento, instabilità del ginocchio o possibili tensioni muscolari.

L'esame dovrebbe essere eseguito con attenzione per evitare lesioni o disagio per il paziente.

I valori di riferimento possono variare in base all'età, al sesso e ad altri fattori individuali.

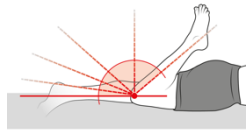
ROM e manovre per la valutazione					
Ginocchio	DX	SN	Note	Immagini	
ROM FLEX/EXT GINOCCHIO (Anca 0°)	OK	OK	Pz prono; angolo asse del femore – asse del ginocchio.		

Figura 29 ROM FLEX/EXT GINOCCHIO

7.2.2. Test5: ROM FLEX/EXT GINOCCHIO (Anca 90°)

Il paziente è supino con l'anca flessa a 90°. L'arto controlaterale è esteso, e l'angolo di estensione del ginocchio è misurato considerando l'asse della gamba e la perpendicolare al letto.

Valuta l'ampiezza del movimento di estensione del ginocchio.

Utile nella valutazione di condizioni come limitazioni di movimento, instabilità del ginocchio o possibili tensioni muscolari.

L'esame dovrebbe essere eseguito con attenzione per evitare lesioni o disagio per il paziente.

I valori di riferimento possono variare in base all'età, al sesso e ad altri fattori individuali.

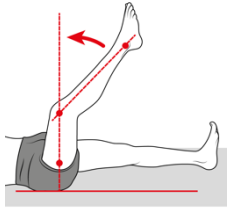
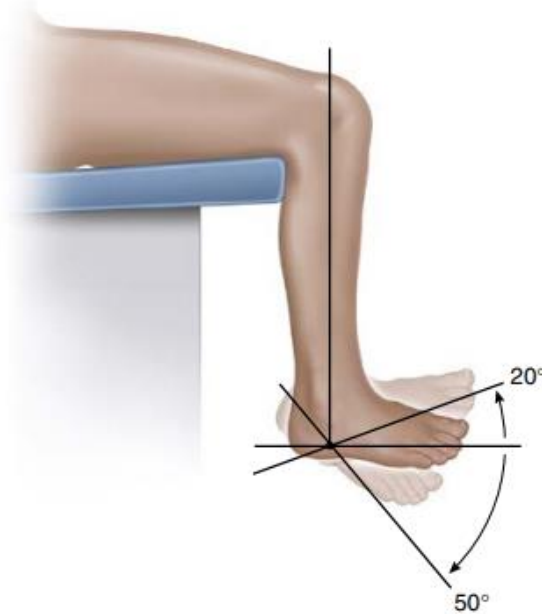
ROM e manovre per la valutazione					
FLEX/EXT GINOCCHIO (anca 90°)	-10	-10	Pz supino, anca flex a 90°, ginocchio inizialmente flex, arto controlaterale esteso. Angolo asse della gamba – perpendicolare al letto.		

Figura 30 ROM FLEX/EXT GINOCCHIO (anca 90°)

7.3. CAVIGLIA

7.3.1. Test6-7: FLESSIONE PLANTARE T-T Silverskiöld Gin 0°/Gin 90°



L'obiettivo dell'esame è valutare la flessione plantare dell'articolazione tibio-tarsica (T-T) attraverso la manovra di Silverskjöld. Questo test è utile per identificare eventuali restrizioni o tensioni nella zona dorsale del piede.

Il paziente è supino (a faccia in su) con la sottoastraglica in posizione neutra. Il ginocchio è flesso, e la misurazione viene effettuata durante la dorsiflessione della T-T.

La sottoastraglica deve essere mantenuta in posizione neutra. Il ginocchio è flesso.

Figura 31 FLEX PLANT T-T Silverskiöld

Dal "Texas Scottish Rite Hospital for Children", 6th edition

Si misura l'angolo di flessione plantare della T-T.

Rileva eventuali restrizioni o tensioni nella zona dorsale del piede.

Utile nella valutazione di condizioni come contratture muscolari, tendinopatie o altre patologie legate alla parte posteriore del piede.

L'esame deve essere eseguito attentamente per evitare disagio o lesioni al paziente.

I valori di riferimento possono variare in base all'età, al sesso e ad altri fattori individuali.

Varianti della Manovra:

- Ginocchio a 90°: La manovra viene eseguita con il ginocchio flesso a 90°.
- Ginocchio a 0° (R2 in correzione): La stessa manovra viene eseguita con il ginocchio esteso.

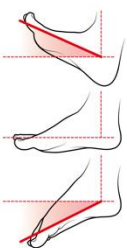
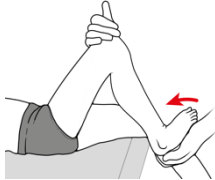
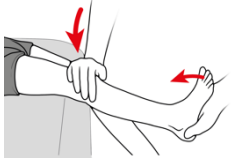
T-T / piede	DX	SN	Note	Immagini
FLESSIONE PLANTARE T-T Silverskiold*	R L 45°	R L 45°	Pz supino, assicurarsi che la sottoastraglica sia in posizione neutra, ginocchio flesso, ROM in dorsiflessione della T-T.	
Gin 90°	30°	30°		
Gin 0° (R2 in correzione)	R L 10°	R L 10°	Stessa manovra a ginocchio esteso.	

Figura 32 ROM FLEX PLANTARE T-T

7.4. Scripting Test 1-7

In questa sezione, verranno presentati gli script di automazione sviluppati per i Test 1-7. Gli script sono stati progettati per facilitare l'esecuzione dei test di movimento, consentendo una precisa riproduzione degli stessi e garantendo una maggiore coerenza nei risultati ottenuti. L'automazione attraverso gli script mira a semplificare il processo di valutazione, riducendo al contempo la possibilità di errori manuali.

Ciascuno degli script è stato creato considerando le specifiche esigenze di ogni test e implementa le corrette procedure di movimento lungo gli assi desiderati. Nel dettaglio, gli script coprono aspetti come la rotazione degli oggetti lungo gli assi X, Y, e Z, l'applicazione di vincoli di movimento, nonché la generazione di animazioni che consentono una visualizzazione chiara e dettagliata dei risultati.

L'introduzione degli script di automazione rappresenta un passo cruciale nel processo di sviluppo, in quanto ne garantisce l'affidabilità e la ripetibilità. La loro implementazione consente anche di risparmiare tempo prezioso durante la fase di valutazione e analisi dei dati, contribuendo così a una più efficiente gestione del progetto.

7.4.1. Creazione Text_1-7

Questo script Blender crea sette oggetti di testo posizionati in diverse coordinate nello spazio 3D. Ogni oggetto di testo è ruotato di -90 gradi lungo l'asse X e ingrandito dieci volte rispetto alle dimensioni predefinite. La funzione `crea_testo` è definita per semplificare il processo di creazione di testo. Le posizioni dei sette testi sono definite da vettori tridimensionali e indicati nello script.

```
1 import bpy
2 from mathutils import Vector
3
4 # Funzione per creare il testo
5 def crea_testo(nome, posizione):
6     bpy.ops.object.text_add(location=posizione)
7     testo = bpy.context.active_object
8     testo.name = nome
9     testo.data.body = f"Testo {nome[-1]} su Blender"
10    bpy.ops.transform.rotate(value=-1.5708, orient_axis='X', orient_type='LOCAL')
11    bpy.ops.transform.resize(value=(10, 10, 10))
12    print(f"Oggetto testo '{nome}' creato, ruotato di -90 gradi lungo X e ingrandito per 10.")
13
14 # Definizione delle posizioni
15 traslazione_Text_1 = Vector((80, 0, -20))
16 traslazione_Text_2 = Vector((280, 0, -20))
17 traslazione_Text_3 = Vector((480, 0, -20))
18 traslazione_Text_4 = Vector((80, 0, -220))
19 traslazione_Text_5 = Vector((280, 0, -220))
20 traslazione_Text_6 = Vector((80, 0, -420))
21 traslazione_Text_7 = Vector((280, 0, -420))
22
23 # Creazione dei testi
24 crea_testo("Text_1", traslazione_Text_1)
25 crea_testo("Text_2", traslazione_Text_2)
26 crea_testo("Text_3", traslazione_Text_3)
27 crea_testo("Text_4", traslazione_Text_4)
28 crea_testo("Text_5", traslazione_Text_5)
29 crea_testo("Text_6", traslazione_Text_6)
30 crea_testo("Text_7", traslazione_Text_7)
```

Figura 33 Script creazione copie modello Test1-7

Inoltre, la stringa di testo all'interno di ciascun oggetto è impostata come "Testo X su Blender", dove X è il numero corrispondente all'oggetto di testo (da 1 a 7). Questo script è utile per creare annotazioni o etichette all'interno di uno spazio 3D in Blender.

7.4.2. Creazione copie modello Test1-7

Questo script Blender è progettato per creare sette copie di tutti gli oggetti presenti nella scena, ciascuna spostata a una posizione diversa. Ogni copia ha un nome univoco ottenuto aggiungendo un suffisso numerico all'originale, seguito da "_test1", "_test2", ..., "_test7".

```
1 import bpy
2 from mathutils import Vector
3
4 # Imposta il vettore di traslazione per le tre copie
5 traslazione_test1 = Vector((200, 0, 0))
6 traslazione_test2 = Vector((400, 0, 0))
7 traslazione_test3 = Vector((600, 0, 0))
8 traslazione_test4 = Vector((200, 0, -200))
9 traslazione_test5 = Vector((400, 0, -200))
10 traslazione_test6 = Vector((200, 0, -400))
11 traslazione_test7 = Vector((400, 0, -400))
12
13 # Itera attraverso tutti gli oggetti nella scena
14 for oggetto in bpy.context.scene.objects:
15     # Crea e posiziona la copia _test1
16     nuova_copia_test1 = oggetto.copy()
17     nuova_copia_test1.name = oggetto.name + "_test1"
18     nuova_copia_test1.location += traslazione_test1
19     bpy.context.scene.collection.objects.link(nuova_copia_test1)
20
21     # Crea e posiziona la copia _test2
22     nuova_copia_test2 = oggetto.copy()
23     nuova_copia_test2.name = oggetto.name + "_test2"
24     nuova_copia_test2.location += traslazione_test2
25     bpy.context.scene.collection.objects.link(nuova_copia_test2)
26
27     # Crea e posiziona la copia _test3
28     nuova_copia_test3 = oggetto.copy()
29     nuova_copia_test3.name = oggetto.name + "_test3"
30     nuova_copia_test3.location += traslazione_test3
31     bpy.context.scene.collection.objects.link(nuova_copia_test3)
32
33     # Crea e posiziona la copia _test4
34     nuova_copia_test4 = oggetto.copy()
35     nuova_copia_test4.name = oggetto.name + "_test4"
36     nuova_copia_test4.location += traslazione_test4
37     bpy.context.scene.collection.objects.link(nuova_copia_test4)
38
39     # Crea e posiziona la copia _test5
40     nuova_copia_test5 = oggetto.copy()
41     nuova_copia_test5.name = oggetto.name + "_test5"
42     nuova_copia_test5.location += traslazione_test5
43     bpy.context.scene.collection.objects.link(nuova_copia_test5)
44
45     # Crea e posiziona la copia _test6
46     nuova_copia_test6 = oggetto.copy()
47     nuova_copia_test6.name = oggetto.name + "_test6"
48     nuova_copia_test6.location += traslazione_test6
49     bpy.context.scene.collection.objects.link(nuova_copia_test6)
50
51     # Crea e posiziona la copia _test7
52     nuova_copia_test7 = oggetto.copy()
53     nuova_copia_test7.name = oggetto.name + "_test7"
54     nuova_copia_test7.location += traslazione_test7
55     bpy.context.scene.collection.objects.link(nuova_copia_test7)
56 print("Copie create con successo.")
```

Figura 34 Script creazione copie modello Test1_7

Le traslazioni per posizionare le copie sono definite dai vettori tridimensionali `traslazione_test1``, ``traslazione_test2``, ..., ``traslazione_test7``. Le posizioni sono specificate in modo tale che le copie siano posizionate in colonne e righe, creando così una griglia di copie.

Questo script è utile quando si desidera eseguire esperimenti o test su diverse configurazioni di oggetti all'interno della scena, mantenendo l'originale e le sue copie organizzate in una griglia.

7.4.3. Parenting Test O-7

Il "parenting" nella modellazione virtuale anatomica si riferisce al processo di stabilire relazioni gerarchiche tra gli elementi del modello, creando una struttura genitore-figlio. In questo contesto specifico, viene utilizzato per definire la gerarchia degli elementi del modello scheletrico, riflettendo accuratamente la struttura anatomica e funzionale del corpo.

Nella pratica, il bacino viene designato come il genitore principale, rappresentando il livello più alto della gerarchia. I segmenti femorali destro e sinistro sono configurati come "figli" del bacino, poiché sono direttamente collegati ad esso. In modo analogo, i segmenti tibiali sono associati come "figli" dei segmenti femorali, e le geometrie dei piedi diventano "figli" dei segmenti tibiali. Questa struttura gerarchica rispecchia fedelmente la connessione anatomica tra le diverse parti dello scheletro.

L'implementazione di questo approccio di "parenting" è fondamentale per garantire coerenza nei movimenti articolari. Quando il genitore viene mosso o ruotato, tutti i figli connessi a esso seguiranno il suo movimento in modo proporzionale, rispettando le relazioni anatomiche predefinite. Ciò facilita una gestione più intuitiva ed efficace della struttura scheletrica nell'ambiente 3D, consentendo una rappresentazione accurata e dinamica dei movimenti articolari.

L'automatizzazione di questo processo attraverso l'implementazione di uno script elimina la necessità di intervento manuale, garantendo una rapida e precisa definizione delle relazioni gerarchiche in base ai dati anatomici forniti.

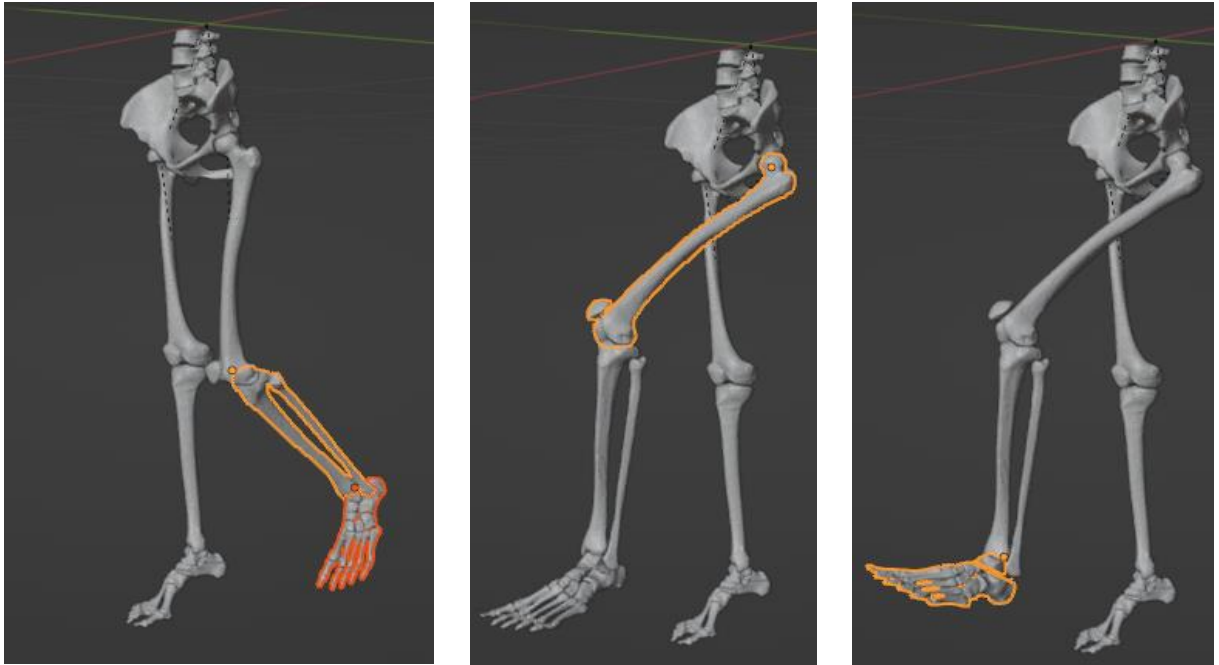


Figura 35 Risultato del Parenting

Come anche nei prossimi capitoli, le parti di scripting simili o ripetute per i diversi test vengono omesse e viene mostrato solo il primo caso per alleggerire il discorso e non perdere concentrazione dal work-flow.

Le restanti parte di codice vengono inserite in Appendice.

Qui vediamo il codice per il modello originale:

```
1 import bpy
2
3 # Seleziona l'oggetto A (Genitore principale)
4 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO']
5 genitore_principale = bpy.context.active_object
6
7 # Seleziona l'oggetto B (Figlio di A)
8 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE DX']
9 figlio_B = bpy.context.active_object
10
11 # Collega B a A
12 bpy.ops.object.select_all(action='DESELECT')
13 genitore_principale.select_set(True)
14 figlio_B.select_set(True)
15 bpy.context.view_layer.objects.active = genitore_principale
16 bpy.ops.object.parent_set(type='OBJECT')
17
18 # Seleziona l'oggetto C (Figlio di A)
19 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE SX']
20 figlio_C = bpy.context.active_object
21
22 # Collega C a A
23 bpy.ops.object.select_all(action='DESELECT')
24 genitore_principale.select_set(True)
25 figlio_C.select_set(True)
26 bpy.context.view_layer.objects.active = genitore_principale
27 bpy.ops.object.parent_set(type='OBJECT')
28
29 # Seleziona l'oggetto D (Figlio di C)
30 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA SX']
31 figlio_D = bpy.context.active_object
32
33 # Collega D a C
34 bpy.ops.object.select_all(action='DESELECT')
35 figlio_C.select_set(True)
36 figlio_D.select_set(True)
37 bpy.context.view_layer.objects.active = figlio_C
38 bpy.ops.object.parent_set(type='OBJECT')
39
40 # Seleziona l'oggetto E (Figlio di B)
41 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA DX']
42 figlio_E = bpy.context.active_object
43
44 # Collega E a B
45 bpy.ops.object.select_all(action='DESELECT')
46 figlio_B.select_set(True)
47 figlio_E.select_set(True)
48 bpy.context.view_layer.objects.active = figlio_B
49 bpy.ops.object.parent_set(type='OBJECT')
50
51 # Seleziona l'oggetto F (Figlio di E)
52 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE DX']
53 figlio_F = bpy.context.active_object
54
55 # Collega F a E
56 bpy.ops.object.select_all(action='DESELECT')
57 figlio_E.select_set(True)
58 figlio_F.select_set(True)
59 bpy.context.view_layer.objects.active = figlio_E
60 bpy.ops.object.parent_set(type='OBJECT')
61
62 # Seleziona l'oggetto G (Figlio di D)
63 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE SX']
64 figlio_G = bpy.context.active_object
65
66 # Collega G a D
67 bpy.ops.object.select_all(action='DESELECT')
68 figlio_D.select_set(True)
69 figlio_G.select_set(True)
70 bpy.context.view_layer.objects.active = figlio_D
71 bpy.ops.object.parent_set(type='OBJECT')
```

Figura 36 Script Parenting Modello originale

7.4.4. Messa in posa Test 1-3 ANCA

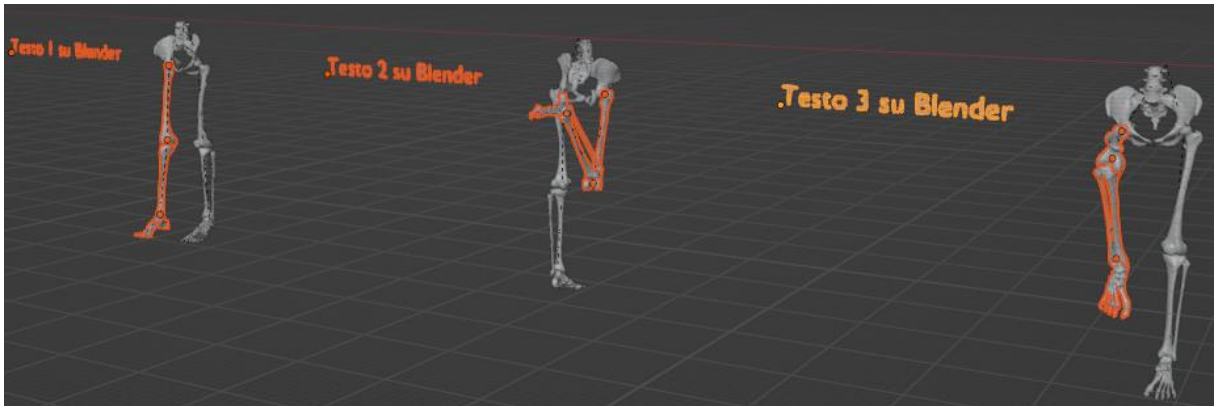


Figura 37 Messa in posa Test1-3

- Lo script per il Test 1 esegue la rotazione dell'oggetto denominato "SA_FEMORE DX_test1" di -15 gradi lungo l'asse X. Questa operazione simula una specifica posizione dell'osso femorale durante il processo di valutazione dei movimenti articolari.

```
1 # POSA TEST1
2 import bpy
3 import math
4
5 # Nome dell'oggetto da ruotare
6 object_name = "SA_FEMORE DX_test1"
7
8 # Trova l'oggetto nella scena
9 selected_object = bpy.data.objects.get(object_name)
10
11 if selected_object is not None:
12     # Specifica l'angolo di rotazione in gradi
13     angle_x = -15
14
15     # Ruota l'oggetto lungo l'asse X
16     selected_object.rotation_euler.x = math.radians(angle_x)
17
18     # Aggiorna la visualizzazione nella viewport
19     bpy.context.view_layer.update()
20
21     print(f"Oggetto '{object_name}' ruotato di {angle_x}° lungo l'asse X.")
22 else:
23     print(f"Oggetto con nome '{object_name}' non trovato. Assicurati di specificare un nome valido.")
24 ..
```

Figura 38 script messa in posa Test1

- L'insieme di script per il Test 2 opera due rotazioni su due oggetti distinti. Innanzitutto, l'oggetto "SA_BUSTO_test2" viene ruotato di 180 gradi lungo l'asse Z. Successivamente, l'oggetto "SA_GAMBA DX_test2" subisce una rotazione di -145 gradi lungo l'asse X. Queste trasformazioni simulano diverse configurazioni anatomiche per il busto e la gamba durante l'analisi del movimento.

```

25 # POSA TEST2
26
27 import bpy
28 import math
29
30 # Nome degli oggetti da ruotare
31 object_name_busto = "SA_BUSTO_test2"
32 object_name_gamba = "SA_GAMBA_DX_test2"
33
34 # Trova gli oggetti nella scena
35 selected_object_busto = bpy.data.objects.get(object_name_busto)
36 selected_object_gamba = bpy.data.objects.get(object_name_gamba)
37
38 # Ruota l'oggetto "SA_BUSTO_test2" di 180° lungo l'asse Z
39 if selected_object_busto is not None:
40     angle_z_busto = 180
41     selected_object_busto.rotation_euler.z = math.radians(angle_z_busto)
42     bpy.context.view_layer.update()
43     print(f"Oggetto '{object_name_busto}' ruotato di {angle_z_busto}° lungo l'asse Z.")
44 else:
45     print(f"Oggetto con nome '{object_name_busto}' non trovato. Assicurati di specificare un nome valido.")
46 |
47 # Ruota l'oggetto "SA_GAMBA_DX_test2" di -145° lungo l'asse X
48 if selected_object_gamba is not None:
49     angle_x_gamba = 145
50     selected_object_gamba.rotation_euler.x = math.radians(angle_x_gamba)
51     bpy.context.view_layer.update()
52     print(f"Oggetto '{object_name_gamba}' ruotato di {angle_x_gamba}° lungo l'asse X.")
53 else:
54     print(f"Oggetto con nome '{object_name_gamba}' non trovato. Assicurati di specificare un nome valido.")
55

```

Figura 39 script messa in posa Test2

- Gli script relativi al Test 3 coinvolgono due oggetti STL: "SA_FEMORE DX_test3" e "SA_GAMBA DX_test3". Per il primo oggetto, viene eseguita una rotazione di 90 gradi lungo l'asse X, mentre per il secondo oggetto, si applica una rotazione di -110 gradi lungo lo stesso asse. Queste operazioni modellano specifiche orientazioni per l'osso femorale e la gamba, contribuendo così alla varietà di configurazioni articolari considerate durante l'esame.

```

57 # POSA TEST3
58
59 import bpy
60
61 # Nome degli oggetti STL
62 nome_femore = "SA_FEMORE DX_test3"
63 nome_gamba = "SA_GAMBA DX_test3"
64
65 # Funzione per ruotare l'oggetto lungo l'asse X
66 def ruota_oggetto(oggetto, angolo):
67     bpy.ops.object.select_all(action='DESELECT')
68     bpy.context.view_layer.objects.active = oggetto
69     oggetto.select_set(True)
70     bpy.ops.transform.rotate(value=angolo, orient_axis='X', orient_type='LOCAL')
71
72 # Ruota di 90 gradi lungo l'asse X l'oggetto SA_FEMORE DX_test3
73 oggetto_femore = bpy.data.objects.get(nome_femore)
74 if oggetto_femore:
75     ruota_oggetto(oggetto_femore, 1.5708)
76     print(f"Oggetto STL '{nome_femore}' ruotato di 90 gradi lungo X.")
77 else:
78     print(f"Oggetto STL '{nome_femore}' non trovato.")
79
80 # Ruota di -110 gradi lungo l'asse X l'oggetto SA_GAMBA DX_test3
81 oggetto_gamba = bpy.data.objects.get(nome_gamba)
82 if oggetto_gamba:
83     ruota_oggetto(oggetto_gamba, -1.91986) # -110 gradi in radianti
84     print(f"Oggetto STL '{nome_gamba}' ruotato di -110 gradi lungo X.")
85 else:
86     print(f"Oggetto STL '{nome_gamba}' non trovato.")

```

Figura 40script messa in posa Test3

7.4.5. Messa in posa Test 4-5 GINOCCHIO



Figura 41 Messa in posa Test4-5

La messa in posa dei test 4 e 5 è stata effettuata tramite i seguenti script:

```
1 import bpy
2 import math
3
4 # Nome degli oggetti da ruotare
5 object_name_busto = "SA_BUSTO_test4"
6 object_name_gamba = "SA_GAMBA_DX_test4"
7
8 # Trova gli oggetti nella scena
9 selected_object_busto = bpy.data.objects.get(object_name_busto)
10 selected_object_gamba = bpy.data.objects.get(object_name_gamba)
11
12 # Ruota l'oggetto "SA_BUSTO_test4" di 180° lungo l'asse Z
13 if selected_object_busto is not None:
14     angle_z_busto = 180
15     selected_object_busto.rotation_euler.z = math.radians(angle_z_busto)
16     bpy.context.view_layer.update()
17     print(f"Oggetto '{object_name_busto}' ruotato di {angle_z_busto}° lungo l'asse Z.")
18 else:
19     print(f"Oggetto con nome '{object_name_busto}' non trovato. Assicurati di specificare un nome valido.")
20
21 # Ruota l'oggetto "SA_GAMBA_DX_test4" di 90° lungo l'asse X
22 if selected_object_gamba is not None:
23     angle_x_gamba = 90
24     selected_object_gamba.rotation_euler.x = math.radians(angle_x_gamba)
25     bpy.context.view_layer.update()
26     print(f"Oggetto '{object_name_gamba}' ruotato di {angle_x_gamba}° lungo l'asse X.")
27 else:
28     print(f"Oggetto con nome '{object_name_gamba}' non trovato. Assicurati di specificare un nome valido.")
```

Figura 42 script messa in posa Test4

Questo script ruota l'oggetto "SA_BUSTO_test4" di 180° lungo l'asse Z e "SA_GAMBA DX_test4" di 90° lungo l'asse X. La rotazione avviene in Blender, aggiornando la visualizzazione nella viewport.

```
31 import bpy
32 import math
33
34 # Nome degli oggetti da ruotare
35 object_name_femore = "SA_FEMORE DX_test5"
36 object_name_gamba = "SA_GAMBA DX_test5"
37
38 # Trova gli oggetti nella scena
39 selected_object_femore = bpy.data.objects.get(object_name_femore)
40 selected_object_gamba = bpy.data.objects.get(object_name_gamba)
41
42 # Ruota l'oggetto "SA_FEMORE DX_test5" di 90° lungo l'asse X
43 if selected_object_femore is not None:
44     angle_x_femore = -90
45     selected_object_femore.rotation_euler.x = math.radians(angle_x_femore)
46     bpy.context.view_layer.update()
47     print(f"Oggetto '{object_name_femore}' ruotato di {angle_x_femore}° lungo l'asse X.")
48 else:
49     print(f"Oggetto con nome '{object_name_femore}' non trovato. Assicurati di specificare un nome valido.")
50
51 # Ruota l'oggetto "SA_GAMBA DX_test5" di 45° lungo l'asse X
52 if selected_object_gamba is not None:
53     angle_x_gamba = 45
54     selected_object_gamba.rotation_euler.x = math.radians(angle_x_gamba)
55     bpy.context.view_layer.update()
56     print(f"Oggetto '{object_name_gamba}' ruotato di {angle_x_gamba}° lungo l'asse X.")
57 else:
58     print(f"Oggetto con nome '{object_name_gamba}' non trovato. Assicurati di specificare un nome valido.")
```

Figura 43 script messa in posa Test5

Questo script ruota l'oggetto "SA_FEMORE DX_test5" di 90° lungo l'asse X e "SA_GAMBA DX_test5" di 45° lungo l'asse X. La rotazione avviene in Blender, aggiornando la visualizzazione nella viewport.

7.4.6. Messa in posa Test 6-7 CAVIGLIA



Figura 44 Messa in posa Test6-7

Per il test 6, non è richiesta una specifica messa in posa.

```
1 import bpy
2 import math
3
4 # Nome degli oggetti da ruotare
5 object_name_femore = "SA_FEMORE DX_test7"
6 object_name_gamba = "SA_GAMBA DX_test7"
7
8 # Trova gli oggetti nella scena
9 selected_object_femore = bpy.data.objects.get(object_name_femore)
10 selected_object_gamba = bpy.data.objects.get(object_name_gamba)
11
12 # Ruota l'oggetto "SA_FEMORE DX_test7" di 90° lungo l'asse X
13 if selected_object_femore is not None:
14     angle_x_femore = -90
15     selected_object_femore.rotation_euler.x = math.radians(angle_x_femore)
16     bpy.context.view_layer.update()
17     print(f"Oggetto '{object_name_femore}' ruotato di {angle_x_femore}° lungo l'asse X.")
18 else:
19     print(f"Oggetto con nome '{object_name_femore}' non trovato. Assicurati di specificare un nome valido.")
20
21 # Ruota l'oggetto "SA_GAMBA DX_test7" di 90° lungo l'asse X
22 if selected_object_gamba is not None:
23     angle_x_gamba = 90
24     selected_object_gamba.rotation_euler.x = math.radians(angle_x_gamba)
25     bpy.context.view_layer.update()
26     print(f"Oggetto '{object_name_gamba}' ruotato di {angle_x_gamba}° lungo l'asse X.")
27 else:
28     print(f"Oggetto con nome '{object_name_gamba}' non trovato. Assicurati di specificare un nome valido.")
29 ,
```

Figura 45 script messa in posa Test7

Per il test 7, lo script in Blender ruota l'oggetto "SA_FEMORE DX_test7" di -90° lungo l'asse X e l'oggetto "SA_GAMBA DX_test7" di 90° lungo l'asse X, aggiornando la visualizzazione nella viewport.

7.4.7. Simulazione Test 1-7

Il seguente script Python è stato sviluppato per la realizzazione di una Computer-Aided Clinical Virtual Intervention (CCVI), parte integrante di un approccio innovativo nell'ambito delle simulazioni cliniche assistite da computer. L'obiettivo principale è simulare movimenti articolari specifici attraverso animazioni 3D, fornendo un ambiente virtuale per l'analisi di diverse pose articolari.

```
1 import bpy
2 import math
3
4 # Funzione per ruotare l'oggetto solo sull'asse X
5 def rotate_object_x(obj, angle):
6     obj.rotation_euler.x = math.radians(angle)
7
8 # Funzione per ruotare l'oggetto solo sull'asse Z
9 def rotate_object_z(obj, angle):
10    obj.rotation_euler.z = math.radians(angle)
11
12 # Funzione per ruotare l'oggetto solo sull'asse Y
13 def rotate_object_y(obj, angle):
14    obj.rotation_euler.y = math.radians(angle)
15
16 preset = 90
17 # Parametri dei test da impostare
18 max_ADD_test1 = 20
19 max_ABD_test1 = 15
20
21 max_EXTRA_test2 = 20
22 max_INTRA_test2 = 35
23
24 max_EST_test3 = 100
25 max_FLEX_test3 = 0
26
27 max_EST_test4 = 150
28 max_FLEX_test4 = 45
29
30 max_EST_test5 = 10
31 max_FLEX_test5 = 30
32
33 max_EST_test6 = -10
34 max_FLEX_test6 = 45
35
36 max_EST_test7 = -30
37 max_FLEX_test7 = 45
38
39 # Velocità animazione (impostabile)
40 n = 3
41 v = round(1 / n, 2) # Velocità
42 max_EST_test3 = preset - 100
43 # Nome dell'oggetto da animare
44 object_name_test1 = "SA_FEMORE DX_test1" # Nome dell'oggetto per il primo script
45 object_name_test2 = "SA_FEMORE DX_test2" # Nome dell'oggetto per il secondo script
46 object_name_test3 = "SA_FEMORE DX_test3" # Nome dell'oggetto per il terzo script
47 object_name_test4 = "SA_GAMBA DX_test4" # Nome dell'oggetto per il quarto script
48 object_name_test5 = "SA_GAMBA DX_test5" # Nome dell'oggetto per il quinto script
49 object_name_test6 = "SA_PIEDE DX_test6" # Nome dell'oggetto per il sesto script
50 object_name_test7 = "SA_PIEDE DX_test7" # Nome dell'oggetto per il settimo script
51
52 text_object_name_test1 = "Text_1" # Nome dell'oggetto testo per il primo script
53 text_object_name_test2 = "Text_2" # Nome dell'oggetto testo per il secondo script
54 text_object_name_test3 = "Text_3" # Nome dell'oggetto testo per il terzo script
55 text_object_name_test4 = "Text_4" # Nome dell'oggetto testo per il quarto script
56 text_object_name_test5 = "Text_5" # Nome dell'oggetto testo per il quinto script
57 text_object_name_test6 = "Text_6" # Nome dell'oggetto testo per il sesto script
58 text_object_name_test7 = "Text_7" # Nome dell'oggetto testo per il settimo script
59
60 en
```

Figura 46 Script simulazione A

calcolata in modo tale da garantire una riproduzione fluida, basandosi sul numero desiderato di fotogrammi (n).

- **Definizione di Funzioni di Rotazione**

Le funzioni `rotate_object_x`, `rotate_object_y`, e `rotate_object_z` sono state progettate per facilitare la rotazione controllata degli oggetti 3D lungo gli assi X, Y e Z. Questa capacità di manipolare le rotazioni è essenziale per simulare i movimenti articolari all'interno della CCVI.

- **Parametri dei Test e Velocità di Animazione**

Parametri specifici per ciascun test, come i gradi massimi di adduzione, abduzione, flessione ed estensione, sono stati definiti per rappresentare le varie pose articolari. La velocità dell'animazione (v) è stata

- **Nomi degli Oggetti e Oggetti di Testo**

```
60 # Trova gli oggetti di testo
61 text_object_test1 = bpy.data.objects.get(text_object_name_test1)
62 text_object_test2 = bpy.data.objects.get(text_object_name_test2)
63 text_object_test3 = bpy.data.objects.get(text_object_name_test3)
64 text_object_test4 = bpy.data.objects.get(text_object_name_test4)
65 text_object_test5 = bpy.data.objects.get(text_object_name_test5)
66 text_object_test6 = bpy.data.objects.get(text_object_name_test6)
67 text_object_test7 = bpy.data.objects.get(text_object_name_test7)
68
69 selected_object_test1 = bpy.data.objects.get(object_name_test1)
70 selected_object_test2 = bpy.data.objects.get(object_name_test2)
71 selected_object_test3 = bpy.data.objects.get(object_name_test3)
72 selected_object_test4 = bpy.data.objects.get(object_name_test4)
73 selected_object_test5 = bpy.data.objects.get(object_name_test5)
74 selected_object_test6 = bpy.data.objects.get(object_name_test6)
75 selected_object_test7 = bpy.data.objects.get(object_name_test7)
76
```

Figura 47 Script simulazione B

Gli oggetti 3D coinvolti nelle simulazioni sono stati denominati in base alla loro rappresentazione anatomica, come femore, gamba e piede. Inoltre, sono stati associati oggetti di testo per fornire informazioni dettagliate sui singoli test.

- **Creazione simulazione**

Per ciascun test, sono stati definiti intervalli di simulazione durante i quali gli oggetti 3D si muovono seguendo un modello sinusoidale. Questo approccio consente di ottenere una rappresentazione realistica delle diverse pose articolari. Le animazioni sono state keyframe-ate, garantendo la cattura precisa delle pose desiderate. Nello script che segue viene visualizzato solo la parte di script che regola la simulazione del test1 per non appesantire la visualizzazione del processo di lavoro (in appendice è riportata la parte inerente agli altri test).

```
77 if selected_object_test1 is not None and selected_object_test2 is not None and selected_object_test3 is not
None and selected_object_test4 is not None and selected_object_test5 is not None and selected_object_test6 is
not None and selected_object_test7 is not None:
78 # Primo script
79 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse Y
80 min_angle_test1 = -max_ADD_test1
81 max_angle_test1 = +max_ABD_test1
82
83 anim_test1 = selected_object_test1.animation_data_create()
84 anim_test1.action = bpy.data.actions.new(name="Rotate Animation Test1")
85
86 frame_start_test1 = 1
87 frame_end_test1 = (max_angle_test1 - min_angle_test1) * n
```

Figura 48 Script simulazione C

```
155 for frame in range(frame_start_test1, frame_end_test1 + 1):
156 # Calcola l'angolo per il primo oggetto
157 t_test1 = (frame - frame_start_test1) / (frame_end_test1 - frame_start_test1)
158 angle_test1 = min_angle_test1 + (max_angle_test1 - min_angle_test1) * (1 + math.sin(2 * math.pi *
t_test1 - math.pi)) / 2
159 rotate_object_y(selected_object_test1, angle_test1)
160 selected_object_test1.keyframe_insert(data_path="rotation_euler", frame=frame)
161
```

Figura 49 Script simulazione D

- **Aggiornamento dei Testi e Velocità**

Dopo ogni animazione, gli oggetti di testo associati vengono aggiornati con informazioni cruciali relative alle pose articolare specifica. L'aggiornamento degli oggetti di testo fornisce un riscontro immediato sulle caratteristiche di ciascun test. Inoltre, è stato incluso un oggetto di testo Text_v per visualizzare la velocità dell'animazione, fornendo ulteriori dettagli sulla dinamica dell'esercizio.

```
201 # Aggiorna il testo negli oggetti di testo
202 if text_object_test1 is not None and text_object_test2 is not None and text_object_test3 is not None and
text_object_test4 is not None and text_object_test5 is not None and text_object_test6 is not None and
text_object_test7 is not None:
203     text_object_test1.data.body = f"Test 1:\nmax_ABD = {max_ABD_test1}°\nmax_ADD = {max_ADD_test1}°"
204     text_object_test2.data.body = f"Test 2:\nmax_INTRA = {max_EXTRA_test2}°\nmax_EXTRA =
{max_INTRA_test2}°"
205     text_object_test3.data.body = f"Test 3:\nmax_EST = {max_FLEX_test3}°\nmax_FLEX = {max_EST_test3}°"
206     text_object_test4.data.body = f"Test 4:\nmax_FLEX = {max_FLEX_test4}°\nmax_EST = {max_EST_test4}°"
207     text_object_test5.data.body = f"Test 5:\nmax_FLEX = {max_FLEX_test5}°\nmax_EST = {max_EST_test5}°"
208     text_object_test6.data.body = f"Test 6:\nmax_EST = {max_FLEX_test6}°\nmax_FLEX = {-max_EST_test6}°"
209     text_object_test7.data.body = f"Test 7:\nmax_EST = {max_FLEX_test7}°\nmax_FLEX = {-max_EST_test7}°"
210
211 # Aggiorna il testo nell'oggetto Text_v
212 text_object_v = bpy.data.objects.get("Text_v") # Assicurati che il nome sia corretto
213 if text_object_v is not None:
214     text_object_v.data.body = f"Velocità animazione (v=1/n): {v}x"
215 else:
216     print(f"Oggetto con nome '{object_name_test1}' o '{object_name_test2}' o '{object_name_test3}' o
'{object_name_test4}' o '{object_name_test5}' o '{object_name_test6}' o '{object_name_test7}' non trovato.
Assicurati di specificare nomi validi.")
```

Figura 50 Script simulazione E

- **Riproduzione dell'Animazione**

La fase finale dello script coinvolge la riproduzione dell'animazione nella vista 3D di Blender. Questo passaggio consente una visualizzazione dinamica delle pose articolari, fornendo un'esperienza visiva essenziale per l'analisi dei risultati ottenuti.

L'implementazione di questo script all'interno della CCVI è cruciale per simulare e analizzare in modo dettagliato le diverse pose articolari. La combinazione di movimenti realistici, informazioni testuali e la possibilità di variare la velocità dell'animazione contribuiscono a creare un ambiente virtuale completo e interattivo per l'utente.

8. Cartella Clinica Virtuale Integrata (CCVI): Comprehensive Health Data

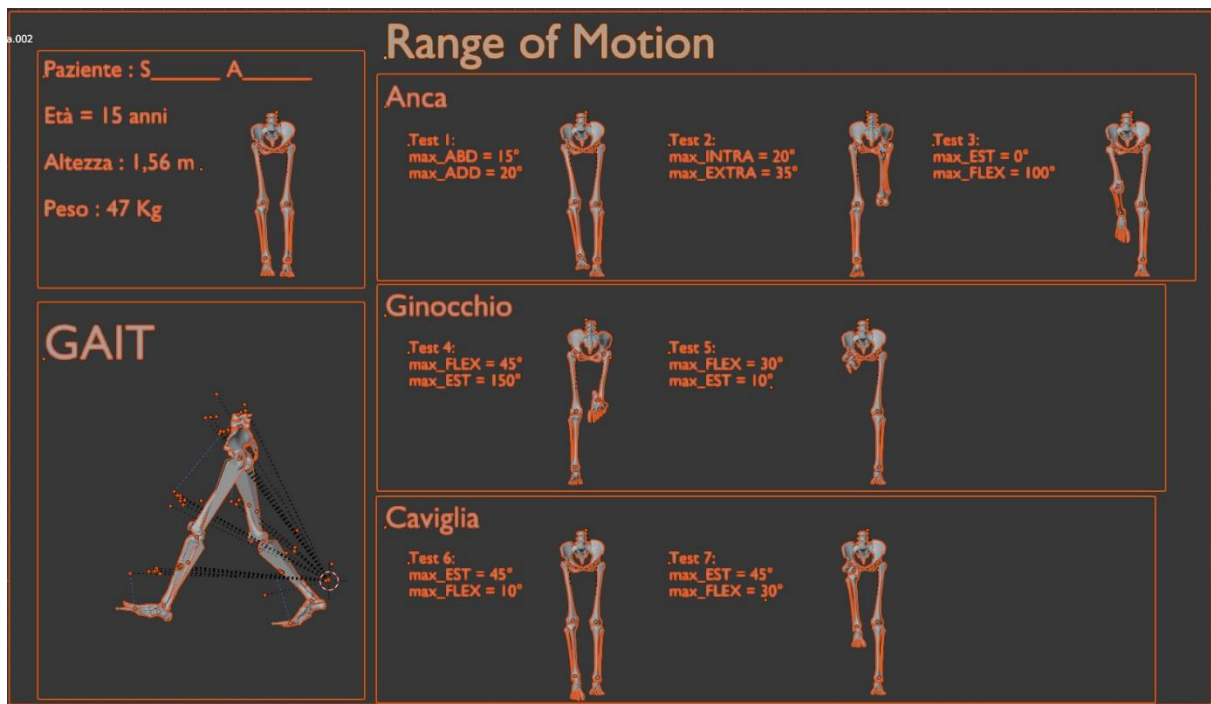


Figura 51 Visualizzazione CCVI

La Cartella Clinica Virtuale Integrata (CCVI) rappresenta un avanzato sistema di gestione dati progettato per raccogliere, organizzare e visualizzare in modo completo tutte le informazioni cliniche di un paziente, integrando dati attuali e passati in un unico ambiente digitale. Nel contesto dell'Istituto Ortopedico Rizzoli, la CCVI è progettata per essere un punto centrale in cui convergono una vasta gamma di dati provenienti da diversi esami e analisi, offrendo un ambiente 3D per la visualizzazione e l'analisi dettagliata.

8.1. Caratteristiche Principali della CCVI

Integrazione di Dati Multidisciplinari

- La CCVI aggrega dati provenienti da esami clinici, test fisiatrici, esami di imaging (come TAC), note operative, e dati storici del paziente.
- Include risultati di test di gait e range of motion (ROM) articolari per ottenere una comprensione completa della funzionalità muscolo-scheletrica, integrando anche informazioni provenienti da visite e trattamenti passati.

Visualizzazione in Ambiente 3D

- Sfrutta capacità avanzate di visualizzazione 3D per rappresentare in modo dettagliato la struttura anatomica del paziente, permettendo una visione comparativa tra dati attuali e storici.
- Consente la visualizzazione di ROM articolari e altri dati clinici in uno spazio tridimensionale, facilitando la comprensione delle condizioni ortopediche nel corso del tempo.

Intuitiva Interfaccia Utente

- Offre un'interfaccia utente intuitiva, consentendo ai professionisti della salute di esplorare facilmente dati e risultati.
- La navigazione tridimensionale facilita una visione approfondita delle strutture anatomiche coinvolte, migliorando la comprensione delle dinamiche cliniche.

Documentazione Dettagliata e Storico Paziente

- Consente di collegare documenti digitali, come piani di resezione, note operative e approcci chirurgici, fornendo una documentazione completa e accessibile del percorso clinico del paziente.
- Archivia dati storici che consentono una valutazione longitudinale della salute del paziente e delle risposte ai trattamenti.

Flessibilità e Scalabilità

- Progettata per adattarsi a diverse specialità mediche e metodologie diagnostiche.
- Scalabile per supportare l'integrazione di nuovi dati e avanzamenti tecnologici nel campo della diagnostica ortopedica.

Facilita la Pianificazione e il Monitoraggio a Lungo Termine

- Agevola la pianificazione delle procedure chirurgiche mediante la visualizzazione dettagliata delle strutture coinvolte nel corso del tempo.
- Consente il monitoraggio a lungo termine della progressione della patologia e dei risultati del trattamento.

In conclusione, la CCVI si propone come uno strumento all'avanguardia che non solo centralizza i dati clinici ma offre anche una visione storica della salute del paziente, fornendo così un approccio completo e trasparente alla gestione delle informazioni mediche.

8.2. Impostazione Manuale dei Parametri Clinici

```
17 # Parametri dei test da impostare
18 max_ADD_test1 = 20
19 max_ABD_test1 = 15
20
21 max_EXTRA_test2 = 20
22 max_INTRA_test2 = 35
23
24 max_EST_test3 = 100
25 max_FLEX_test3 = 0
26
27 max_EST_test4 = 150
28 max_FLEX_test4 = 45
29
30 max_EST_test5 = 10
31 max_FLEX_test5 = 30
32
33 max_EST_test6 = -10
34 max_FLEX_test6 = 45
35
36 max_EST_test7 = -30
37 max_FLEX_test7 = 45
```

Figura 52 Impostazione parametri dei test

Dopo il completamento della fase di automatizzazione del modello scheletrico, l'attenzione si rivolge all'impostazione manuale dei parametri clinici. Questo processo, agevolato dall'integrazione di script, si propone di garantire efficienza e precisione nella creazione della Cartella Clinica Virtuale Integrata (CCVI). L'utilizzo di script non solo semplifica il procedimento ma assicura anche coerenza nella presentazione dei dati, rendendo le informazioni chiare e facilmente interpretabili. Questo approccio sinergico di automatizzazione e impostazione manuale

consente la realizzazione rapida e accurata della CCVI, rendendola accessibile anche a operatori non specializzati.

8.3. Personalizzazione del Range of Motion (ROM) Articolare

L'integrazione di script offre la possibilità di inserire manualmente i valori del Range of Motion (ROM) articolare specifici del paziente. Questi script consentono la personalizzazione dei parametri clinici, riflettendo con precisione le caratteristiche anatomiche e funzionali individuali del paziente. Ogni script è strutturato in modo chiaro e include un'area dedicata all'input dei dati relativi al ROM articolare, facilitando un'interazione agevole anche per operatori meno esperti.

8.4. Presentazione Dinamica dei Dati Clinici

L'automazione della presentazione dei dati avviene attraverso script che generano un'animazione 3D rappresentante la posa del test fisiatrico e il movimento articolare del paziente. In parallelo, uno script genera automaticamente testi informativi, fornendo valori minimi e massimi del ROM articolare durante il test. Questa sinergia tra animazione visuale e testo informativo fornisce una rappresentazione completa e dettagliata delle condizioni ortopediche del paziente. Tale approccio supera le limitazioni delle tradizionali cartelle cartacee, offrendo una visualizzazione più intuitiva e comprensibile dei dati clinici.

Si può vedere nella figura sottostante, puro esempio di un movimento periodico, è possibile anche visualizzare l'andamento nel tempo degli angoli articolari durante il movimento.

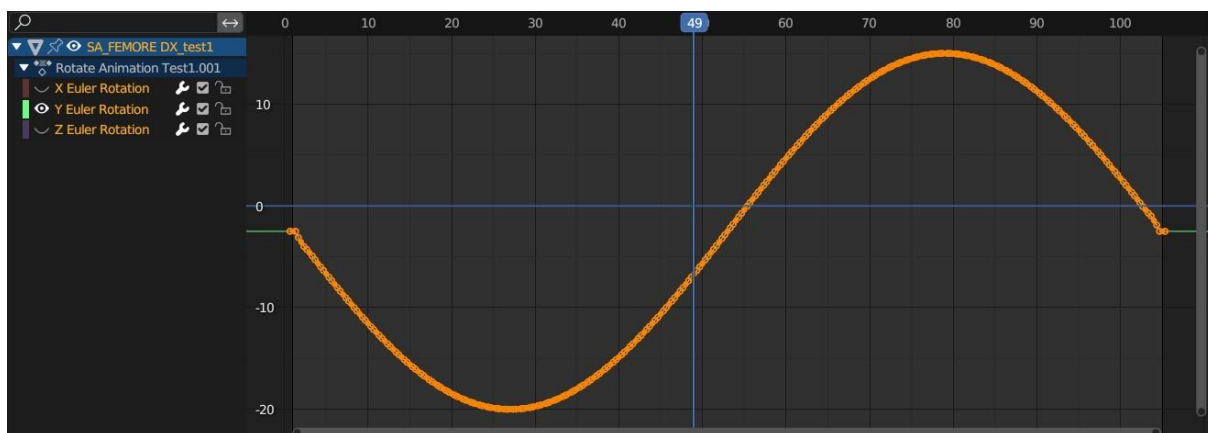


Figura 53 Visualizzazione angoli durante il movimento

9. Sviluppi Futuri: Automazione del Monitoraggio del Movimento Articolare

Nel delineare il futuro della ricerca e sviluppo, sono individuati diversi ambiti di interesse che contribuiranno a perfezionare e ampliare il sistema proposto.

9.2. Evoluzione del Rigging

Nel contesto dell'evoluzione del rigging, che rappresenta una fase cruciale del processo di modellazione virtuale, l'obiettivo principale è migliorare la precisione e la naturalezza dei movimenti articolari nel modello 3D. La metodologia adottata si concentra sull'implementazione di tecniche avanzate di rigging, senza l'utilizzo di script, ponendo l'attenzione sull'analisi delle informazioni temporali provenienti dal file .trc.

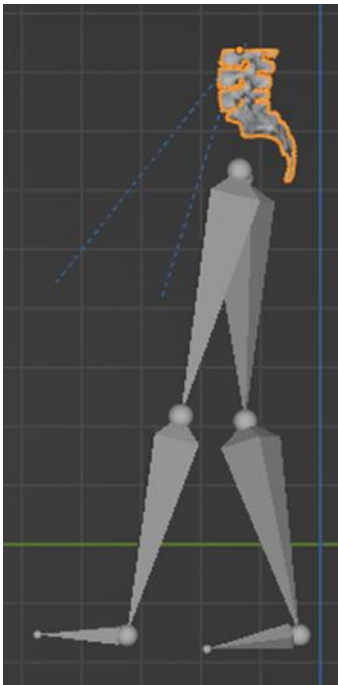


Figura 54 Armature per Rigging

Il processo ha avuto inizio con la creazione di un'armatura, che definisce le ossa di ciascun segmento scheletrico. Successivamente, è stata configurata la cinematica inversa al modello scheletrico, consentendo al modello di articolarsi in modo dinamico e realistico.

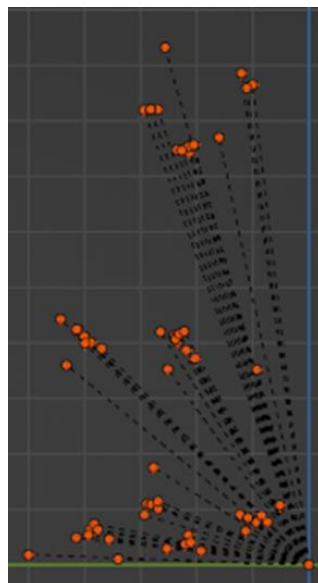


Figura 55 Dati marker .trc

In questa fase, si è posta particolare attenzione all'associazione dei movimenti del modello scheletrico ai marker di interesse principale, presenti nei file .trc esportato e nei file .stl denominati "REPERI".

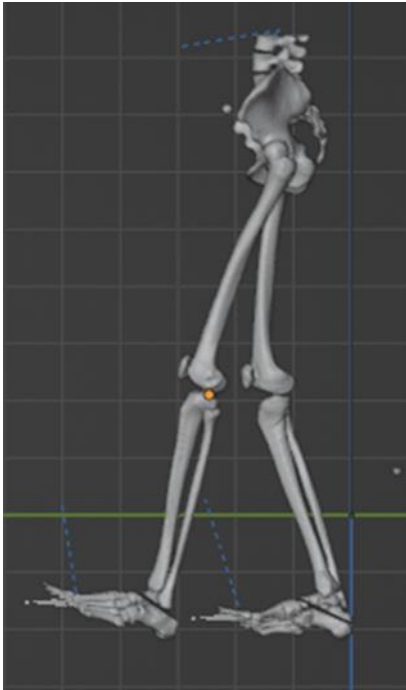


Figura 56 Implementazione Cinematica inversa e gait

L'associazione è stata realizzata attraverso l'utilizzo di vincoli tra i marker corrispondenti e l'impiego di comandi di "follow path". Al fine di garantire una perfetta sincronia tra il modello e i dati provenienti dagli esami di gait, sono stati apportati ulteriori accorgimenti. Il modello, sebbene ancora in fase di sviluppo, evidenzia già il potenziale di implementazione con grande precisione.

Questo approccio integra le informazioni temporali dei marker con il modello scheletrico, sfruttando il rigging per configurare un'armatura dinamica. Questa armatura consente al modello di seguire le configurazioni anatomiche possibili, simile al funzionamento di un braccio meccanico. L'utilizzo di vincoli, come il "follow path", e l'attenzione alla non deformabilità e non compenetrazione delle superfici contribuiscono a creare una rappresentazione virtuale accurata e dinamica dei movimenti articolari durante gli esami di gait, specialmente nelle articolazioni costituite da tessuto osseo.

9.3. Interfaccia Utente Intuitiva per i Clinici

L'introduzione di plugin dotati di finestre user-friendly mira a semplificare l'interazione dei clinici con il sistema. Questi strumenti consentiranno un utilizzo più agevole e immediato delle funzionalità, rendendo il monitoraggio del movimento articolare accessibile anche a utenti non specializzati.

9.4. Estensione Modello al Movimento degli Arti Superiori e Miglioramenti in Blender

Il futuro sviluppo prevede l'estensione del modello anche al movimento dell'arto superiore. Parallelamente, sono previsti continui miglioramenti in Blender per ottimizzare le funzionalità già esistenti e implementare nuove caratteristiche utili all'analisi del movimento articolare.

9.5. Integrazione Parte Geometrica e Clinica-Funzionale

Un'evoluzione significativa consiste nell'integrare in maniera sempre più stretta la parte geometrica del modello con l'aspetto clinico-funzionale. Questa sinergia permetterà di fornire una visione olistica e completa delle condizioni ortopediche dei pazienti.

9.6. Potenziali Vantaggi: Applicazione Presso l'Istituto Ortopedico Rizzoli

Il focus è rivolto all'applicazione pratica presso l'Istituto Ortopedico Rizzoli, sfruttando appieno i potenziali vantaggi del sistema proposto. L'obiettivo è rendere la tecnologia parte integrante delle procedure diagnostiche e terapeutiche, contribuendo alla personalizzazione delle cure.

9.7. Affrontare Sfide e Proporre Soluzioni

L'affrontare delle sfide è una componente essenziale del processo di sviluppo. Il lavoro futuro sarà dedicato all'identificazione e alla risoluzione di eventuali problematiche emergenti, garantendo la robustezza e l'efficacia del sistema.

9.8. Rilevanza Futura nell'Ambito Ortopedico

Infine, si mira a valutare la rilevanza futura di questo approccio nell'ambito ortopedico. L'obiettivo è contribuire all'evoluzione delle pratiche cliniche, promuovendo un utilizzo sempre più diffuso e una maggiore integrazione delle tecnologie innovative nel contesto sanitario.

Conclusion

L'introduzione di scripting nell'ambito della modellazione virtuale ortopedica si rivela come un motore di innovazione, fornendo soluzioni efficaci per superare le sfide legate alla complessità e variabilità dei dati. L'automazione delle fasi chiave del processo, con una particolare enfasi sulla definizione dei centri di rotazione e sulla presentazione visuale dei parametri clinici, costituisce una pietra miliare nell'evoluzione della pratica ortopedica.

La riduzione della dipendenza da interventi manuali si traduce in una maggiore coerenza e precisione nella creazione di modelli scheletrici virtuali. Questa precisione, unita alla capacità di adattare dinamicamente il modello a nuovi dati, contribuisce in modo significativo alla creazione di una rappresentazione virtuale dei pazienti che è accurata, personalizzata e sempre allineata agli sviluppi in corso.

Il valore intrinseco di questa metodologia va oltre la fornitura di una panoramica accurata delle condizioni ortopediche dei pazienti, evidenziando il potenziale innovativo della modellazione virtuale nell'ambito ortopedico. La capacità di integrare nuovi dati e informazioni aggiuntive non solo rende questa metodologia attuale, ma la posiziona all'avanguardia, pronta a rispondere alle sfide diagnostiche e terapeutiche in evoluzione.

Il riassunto dei punti chiave sottolinea l'efficacia di questo approccio, evidenziando come l'automazione e la personalizzazione contribuiscano a elevare la qualità delle rappresentazioni virtuali. La precisione nei dettagli anatomici e funzionali offre un supporto tangibile alle decisioni ortopediche, promuovendo una pratica più informata e mirata.

Nel contesto dell'ortopedia, l'importanza di questa metodologia si estende ben oltre la creazione di modelli scheletrici virtuali, trasformandosi in uno strumento diagnostico e terapeutico avanzato. Questo apre la strada a futuri sviluppi e applicazioni innovative. La continua evoluzione di questa metodologia non solo risponde alle attuali esigenze della pratica ortopedica, ma anticipa anche le sfide future, posizionando la modellazione virtuale come un elemento essenziale nella guida diagnostica e nella pianificazione dei trattamenti ortopedici.

Appendice

7.4.3.a. Parenting Test O-7

```
73 import bpy
74
75 # Seleziona l'oggetto A (Genitore principale)
76 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test1']
77 genitore_principale = bpy.context.active_object
78
79 # Seleziona l'oggetto B (Figlio di A)
80 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test1']
81 figlio_B = bpy.context.active_object
82
83 # Collega B a A
84 bpy.ops.object.select_all(action='DESELECT')
85 genitore_principale.select_set(True)
86 figlio_B.select_set(True)
87 bpy.context.view_layer.objects.active = genitore_principale
88 bpy.ops.object.parent_set(type='OBJECT')
89
90 # Seleziona l'oggetto C (Figlio di A)
91 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test1']
92 figlio_C = bpy.context.active_object
93
94 # Collega C a A
95 bpy.ops.object.select_all(action='DESELECT')
96 genitore_principale.select_set(True)
97 figlio_C.select_set(True)
98 bpy.context.view_layer.objects.active = genitore_principale
99 bpy.ops.object.parent_set(type='OBJECT')
100
101 # Seleziona l'oggetto D (Figlio di C)
102 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test1']
103 figlio_D = bpy.context.active_object
104
105 # Collega D a C
106 bpy.ops.object.select_all(action='DESELECT')
107 figlio_C.select_set(True)
108 figlio_D.select_set(True)
109 bpy.context.view_layer.objects.active = figlio_C
110 bpy.ops.object.parent_set(type='OBJECT')
111
112 # Seleziona l'oggetto E (Figlio di B)
113 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test1']
114 figlio_E = bpy.context.active_object
115
116 # Collega E a B
117 bpy.ops.object.select_all(action='DESELECT')
118 figlio_B.select_set(True)
119 figlio_E.select_set(True)
120 bpy.context.view_layer.objects.active = figlio_B
121 bpy.ops.object.parent_set(type='OBJECT')
122
123 # Seleziona l'oggetto F (Figlio di E)
124 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test1']
125 figlio_F = bpy.context.active_object
126
127 # Collega F a E
128 bpy.ops.object.select_all(action='DESELECT')
129 figlio_E.select_set(True)
130 figlio_F.select_set(True)
131 bpy.context.view_layer.objects.active = figlio_E
132 bpy.ops.object.parent_set(type='OBJECT')
133
134 # Seleziona l'oggetto G (Figlio di D)
135 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test1']
136 figlio_G = bpy.context.active_object
137
138 # Collega G a D
139 bpy.ops.object.select_all(action='DESELECT')
140 figlio_D.select_set(True)
141 figlio_G.select_set(True)
142 bpy.context.view_layer.objects.active = figlio_D
143 bpy.ops.object.parent_set(type='OBJECT')
```

Figura 57 Script Parenting Test1

```

146 import bpy
147
148 # Seleziona l'oggetto A (Genitore principale)
149 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test2']
150 genitore_principale = bpy.context.active_object
151
152 # Seleziona l'oggetto B (Figlio di A)
153 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test2']
154 figlio_B = bpy.context.active_object
155
156 # Collega B a A
157 bpy.ops.object.select_all(action='DESELECT')
158 genitore_principale.select_set(True)
159 figlio_B.select_set(True)
160 bpy.context.view_layer.objects.active = genitore_principale
161 bpy.ops.object.parent_set(type='OBJECT')
162
163 # Seleziona l'oggetto C (Figlio di A)
164 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test2']
165 figlio_C = bpy.context.active_object
166
167 # Collega C a A
168 bpy.ops.object.select_all(action='DESELECT')
169 genitore_principale.select_set(True)
170 figlio_C.select_set(True)
171 bpy.context.view_layer.objects.active = genitore_principale
172 bpy.ops.object.parent_set(type='OBJECT')
173
174 # Seleziona l'oggetto D (Figlio di C)
175 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test2']
176 figlio_D = bpy.context.active_object
177
178 # Collega D a C
179 bpy.ops.object.select_all(action='DESELECT')
180 figlio_C.select_set(True)
181 figlio_D.select_set(True)
182 bpy.context.view_layer.objects.active = figlio_C
183 bpy.ops.object.parent_set(type='OBJECT')
184
185 # Seleziona l'oggetto E (Figlio di B)
186 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test2']
187 figlio_E = bpy.context.active_object
188
189 # Collega E a B
190 bpy.ops.object.select_all(action='DESELECT')
191 figlio_B.select_set(True)
192 figlio_E.select_set(True)
193 bpy.context.view_layer.objects.active = figlio_B
194 bpy.ops.object.parent_set(type='OBJECT')
195
196 # Seleziona l'oggetto F (Figlio di E)
197 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test2']
198 figlio_F = bpy.context.active_object
199
200 # Collega F a E
201 bpy.ops.object.select_all(action='DESELECT')
202 figlio_E.select_set(True)
203 figlio_F.select_set(True)
204 bpy.context.view_layer.objects.active = figlio_E
205 bpy.ops.object.parent_set(type='OBJECT')
206
207 # Seleziona l'oggetto G (Figlio di D)
208 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test2']
209 figlio_G = bpy.context.active_object
210
211 # Collega G a D
212 bpy.ops.object.select_all(action='DESELECT')
213 figlio_D.select_set(True)
214 figlio_G.select_set(True)
215 bpy.context.view_layer.objects.active = figlio_D
216 bpy.ops.object.parent_set(type='OBJECT')
217

```

Figura 58 Script Parenting Test2


```

220 import bpy
221
222 # Seleziona l'oggetto A (Genitore principale)
223 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test3']
224 genitore_principale = bpy.context.active_object
225
226 # Seleziona l'oggetto B (Figlio di A)
227 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test3']
228 figlio_B = bpy.context.active_object
229
230 # Collega B a A
231 bpy.ops.object.select_all(action='DESELECT')
232 genitore_principale.select_set(True)
233 figlio_B.select_set(True)
234 bpy.context.view_layer.objects.active = genitore_principale
235 bpy.ops.object.parent_set(type='OBJECT')
236
237 # Seleziona l'oggetto C (Figlio di A)
238 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test3']
239 figlio_C = bpy.context.active_object
240
241 # Collega C a A
242 bpy.ops.object.select_all(action='DESELECT')
243 genitore_principale.select_set(True)
244 figlio_C.select_set(True)
245 bpy.context.view_layer.objects.active = genitore_principale
246 bpy.ops.object.parent_set(type='OBJECT')
247
248 # Seleziona l'oggetto D (Figlio di C)
249 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test3']
250 figlio_D = bpy.context.active_object
251
252 # Collega D a C
253 bpy.ops.object.select_all(action='DESELECT')
254 figlio_C.select_set(True)
255 figlio_D.select_set(True)
256 bpy.context.view_layer.objects.active = figlio_C
257 bpy.ops.object.parent_set(type='OBJECT')
258
259 # Seleziona l'oggetto E (Figlio di B)
260 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test3']
261 figlio_E = bpy.context.active_object
262
263 # Collega E a B
264 bpy.ops.object.select_all(action='DESELECT')
265 figlio_B.select_set(True)
266 figlio_E.select_set(True)
267 bpy.context.view_layer.objects.active = figlio_B
268 bpy.ops.object.parent_set(type='OBJECT')
269
270 # Seleziona l'oggetto F (Figlio di E)
271 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test3']
272 figlio_F = bpy.context.active_object
273
274 # Collega F a E
275 bpy.ops.object.select_all(action='DESELECT')
276 figlio_E.select_set(True)
277 figlio_F.select_set(True)
278 bpy.context.view_layer.objects.active = figlio_E
279 bpy.ops.object.parent_set(type='OBJECT')
280
281 # Seleziona l'oggetto G (Figlio di D)
282 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test3']
283 figlio_G = bpy.context.active_object
284
285 # Collega G a D
286 bpy.ops.object.select_all(action='DESELECT')
287 figlio_D.select_set(True)
288 figlio_G.select_set(True)
289 bpy.context.view_layer.objects.active = figlio_D
290 bpy.ops.object.parent_set(type='OBJECT')
291

```

Figura 59 Script Parenting Test3

```

295 import bpy
296
297 # Seleziona l'oggetto A (Genitore principale)
298 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test4']
299 genitore_principale = bpy.context.active_object
300
301 # Seleziona l'oggetto B (Figlio di A)
302 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test4']
303 figlio_B = bpy.context.active_object
304
305 # Collega B a A
306 bpy.ops.object.select_all(action='DESELECT')
307 genitore_principale.select_set(True)
308 figlio_B.select_set(True)
309 bpy.context.view_layer.objects.active = genitore_principale
310 bpy.ops.object.parent_set(type='OBJECT')
311
312 # Seleziona l'oggetto C (Figlio di A)
313 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test4']
314 figlio_C = bpy.context.active_object
315
316 # Collega C a A
317 bpy.ops.object.select_all(action='DESELECT')
318 genitore_principale.select_set(True)
319 figlio_C.select_set(True)
320 bpy.context.view_layer.objects.active = genitore_principale
321 bpy.ops.object.parent_set(type='OBJECT')
322
323 # Seleziona l'oggetto D (Figlio di C)
324 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test4']
325 figlio_D = bpy.context.active_object
326
327 # Collega D a C
328 bpy.ops.object.select_all(action='DESELECT')
329 figlio_C.select_set(True)
330 figlio_D.select_set(True)
331 bpy.context.view_layer.objects.active = figlio_C
332 bpy.ops.object.parent_set(type='OBJECT')
333
334 # Seleziona l'oggetto E (Figlio di B)
335 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test4']
336 figlio_E = bpy.context.active_object
337
338 # Collega E a B
339 bpy.ops.object.select_all(action='DESELECT')
340 figlio_B.select_set(True)
341 figlio_E.select_set(True)
342 bpy.context.view_layer.objects.active = figlio_B
343 bpy.ops.object.parent_set(type='OBJECT')
344
345 # Seleziona l'oggetto F (Figlio di E)
346 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test4']
347 figlio_F = bpy.context.active_object
348
349 # Collega F a E
350 bpy.ops.object.select_all(action='DESELECT')
351 figlio_E.select_set(True)
352 figlio_F.select_set(True)
353 bpy.context.view_layer.objects.active = figlio_E
354 bpy.ops.object.parent_set(type='OBJECT')
355
356 # Seleziona l'oggetto G (Figlio di D)
357 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test4']
358 figlio_G = bpy.context.active_object
359
360 # Collega G a D
361 bpy.ops.object.select_all(action='DESELECT')
362 figlio_D.select_set(True)
363 figlio_G.select_set(True)
364 bpy.context.view_layer.objects.active = figlio_D
365 bpy.ops.object.parent_set(type='OBJECT')

```

Figura 60 Script Parenting Test4

```

368 import bpy
369
370 # Seleziona l'oggetto A (Genitore principale)
371 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test5']
372 genitore_principale = bpy.context.active_object
373
374 # Seleziona l'oggetto B (Figlio di A)
375 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test5']
376 figlio_B = bpy.context.active_object
377
378 # Collega B a A
379 bpy.ops.object.select_all(action='DESELECT')
380 genitore_principale.select_set(True)
381 figlio_B.select_set(True)
382 bpy.context.view_layer.objects.active = genitore_principale
383 bpy.ops.object.parent_set(type='OBJECT')
384
385 # Seleziona l'oggetto C (Figlio di A)
386 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test5']
387 figlio_C = bpy.context.active_object
388
389 # Collega C a A
390 bpy.ops.object.select_all(action='DESELECT')
391 genitore_principale.select_set(True)
392 figlio_C.select_set(True)
393 bpy.context.view_layer.objects.active = genitore_principale
394 bpy.ops.object.parent_set(type='OBJECT')
395
396 # Seleziona l'oggetto D (Figlio di C)
397 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test5']
398 figlio_D = bpy.context.active_object
399
400 # Collega D a C
401 bpy.ops.object.select_all(action='DESELECT')
402 figlio_C.select_set(True)
403 figlio_D.select_set(True)
404 bpy.context.view_layer.objects.active = figlio_C
405 bpy.ops.object.parent_set(type='OBJECT')
406
407 # Seleziona l'oggetto E (Figlio di B)
408 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test5']
409 figlio_E = bpy.context.active_object
410
411 # Collega E a B
412 bpy.ops.object.select_all(action='DESELECT')
413 figlio_B.select_set(True)
414 figlio_E.select_set(True)
415 bpy.context.view_layer.objects.active = figlio_B
416 bpy.ops.object.parent_set(type='OBJECT')
417
418 # Seleziona l'oggetto F (Figlio di E)
419 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test5']
420 figlio_F = bpy.context.active_object
421
422 # Collega F a E
423 bpy.ops.object.select_all(action='DESELECT')
424 figlio_E.select_set(True)
425 figlio_F.select_set(True)
426 bpy.context.view_layer.objects.active = figlio_E
427 bpy.ops.object.parent_set(type='OBJECT')
428
429 # Seleziona l'oggetto G (Figlio di D)
430 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test5']
431 figlio_G = bpy.context.active_object
432
433 # Collega G a D
434 bpy.ops.object.select_all(action='DESELECT')
435 figlio_D.select_set(True)
436 figlio_G.select_set(True)
437 bpy.context.view_layer.objects.active = figlio_D
438 bpy.ops.object.parent_set(type='OBJECT')

```

Figura 61 Script Parenting Test5

```

442 import bpy
443
444 # Seleziona l'oggetto A (Genitore principale)
445 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test6']
446 genitore_principale = bpy.context.active_object
447
448 # Seleziona l'oggetto B (Figlio di A)
449 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test6']
450 figlio_B = bpy.context.active_object
451
452 # Collega B a A
453 bpy.ops.object.select_all(action='DESELECT')
454 genitore_principale.select_set(True)
455 figlio_B.select_set(True)
456 bpy.context.view_layer.objects.active = genitore_principale
457 bpy.ops.object.parent_set(type='OBJECT')
458
459 # Seleziona l'oggetto C (Figlio di A)
460 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test6']
461 figlio_C = bpy.context.active_object
462
463 # Collega C a A
464 bpy.ops.object.select_all(action='DESELECT')
465 genitore_principale.select_set(True)
466 figlio_C.select_set(True)
467 bpy.context.view_layer.objects.active = genitore_principale
468 bpy.ops.object.parent_set(type='OBJECT')
469
470 # Seleziona l'oggetto D (Figlio di C)
471 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test6']
472 figlio_D = bpy.context.active_object
473
474 # Collega D a C
475 bpy.ops.object.select_all(action='DESELECT')
476 figlio_C.select_set(True)
477 figlio_D.select_set(True)
478 bpy.context.view_layer.objects.active = figlio_C
479 bpy.ops.object.parent_set(type='OBJECT')
480
481 # Seleziona l'oggetto E (Figlio di B)
482 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test6']
483 figlio_E = bpy.context.active_object
484
485 # Collega E a B
486 bpy.ops.object.select_all(action='DESELECT')
487 figlio_B.select_set(True)
488 figlio_E.select_set(True)
489 bpy.context.view_layer.objects.active = figlio_B
490 bpy.ops.object.parent_set(type='OBJECT')
491
492 # Seleziona l'oggetto F (Figlio di E)
493 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test6']
494 figlio_F = bpy.context.active_object
495
496 # Collega F a E
497 bpy.ops.object.select_all(action='DESELECT')
498 figlio_E.select_set(True)
499 figlio_F.select_set(True)
500 bpy.context.view_layer.objects.active = figlio_E
501 bpy.ops.object.parent_set(type='OBJECT')
502
503 # Seleziona l'oggetto G (Figlio di D)
504 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test6']
505 figlio_G = bpy.context.active_object
506
507 # Collega G a D
508 bpy.ops.object.select_all(action='DESELECT')
509 figlio_D.select_set(True)
510 figlio_G.select_set(True)
511 bpy.context.view_layer.objects.active = figlio_D
512 bpy.ops.object.parent_set(type='OBJECT')

```

Figura 62 Script Parenting Test6

```

516 import bpy
517
518 # Seleziona l'oggetto A (Genitore principale)
519 bpy.context.view_layer.objects.active = bpy.data.objects['SA_BUSTO_test7']
520 genitore_principale = bpy.context.active_object
521
522 # Seleziona l'oggetto B (Figlio di A)
523 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_DX_test7']
524 figlio_B = bpy.context.active_object
525
526 # Collega B a A
527 bpy.ops.object.select_all(action='DESELECT')
528 genitore_principale.select_set(True)
529 figlio_B.select_set(True)
530 bpy.context.view_layer.objects.active = genitore_principale
531 bpy.ops.object.parent_set(type='OBJECT')
532
533 # Seleziona l'oggetto C (Figlio di A)
534 bpy.context.view_layer.objects.active = bpy.data.objects['SA_FEMORE_SX_test7']
535 figlio_C = bpy.context.active_object
536
537 # Collega C a A
538 bpy.ops.object.select_all(action='DESELECT')
539 genitore_principale.select_set(True)
540 figlio_C.select_set(True)
541 bpy.context.view_layer.objects.active = genitore_principale
542 bpy.ops.object.parent_set(type='OBJECT')
543
544 # Seleziona l'oggetto D (Figlio di C)
545 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_SX_test7']
546 figlio_D = bpy.context.active_object
547
548 # Collega D a C
549 bpy.ops.object.select_all(action='DESELECT')
550 figlio_C.select_set(True)
551 figlio_D.select_set(True)
552 bpy.context.view_layer.objects.active = figlio_C
553 bpy.ops.object.parent_set(type='OBJECT')
554
555 # Seleziona l'oggetto E (Figlio di B)
556 bpy.context.view_layer.objects.active = bpy.data.objects['SA_GAMBA_DX_test7']
557 figlio_E = bpy.context.active_object
558
559 # Collega E a B
560 bpy.ops.object.select_all(action='DESELECT')
561 figlio_B.select_set(True)
562 figlio_E.select_set(True)
563 bpy.context.view_layer.objects.active = figlio_B
564 bpy.ops.object.parent_set(type='OBJECT')
565
566 # Seleziona l'oggetto F (Figlio di E)
567 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_DX_test7']
568 figlio_F = bpy.context.active_object
569
570 # Collega F a E
571 bpy.ops.object.select_all(action='DESELECT')
572 figlio_E.select_set(True)
573 figlio_F.select_set(True)
574 bpy.context.view_layer.objects.active = figlio_E
575 bpy.ops.object.parent_set(type='OBJECT')
576
577 # Seleziona l'oggetto G (Figlio di D)
578 bpy.context.view_layer.objects.active = bpy.data.objects['SA_PIEDE_SX_test7']
579 figlio_G = bpy.context.active_object
580
581 # Collega G a D
582 bpy.ops.object.select_all(action='DESELECT')
583 figlio_D.select_set(True)
584 figlio_G.select_set(True)
585 bpy.context.view_layer.objects.active = figlio_D
586 bpy.ops.object.parent_set(type='OBJECT')

```

Figura 63 Script Parenting Test7

7.4.7.a. Simulazione Test 1-7

```
89 # Secondo script
90 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse Z
91 min_angle_test2 = - max_EXTRA_test2
92 max_angle_test2 = + max_INTRA_test2
93
94 anim_test2 = selected_object_test2.animation_data_create()
95 anim_test2.action = bpy.data.actions.new(name="Rotate Animation Test2")
96
97 frame_start_test2 = frame_start_test1
98 frame_end_test2 = frame_end_test1
99
100 # Terzo script
101 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse X
102 min_angle_test3 = - 1.5708 + math.radians(max_EST_test3)
103 max_angle_test3 = math.radians(max_FLEX_test3)
104
105 anim_test3 = selected_object_test3.animation_data_create()
106 anim_test3.action = bpy.data.actions.new(name="Rotate Animation Test3")
107
108 frame_start_test3 = frame_start_test1
109 frame_end_test3 = frame_end_test1
110
111 # Quarto script
112 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse X
113 min_angle_test4 = math.radians(max_EST_test4)
114 max_angle_test4 = math.radians(max_FLEX_test4)
115
116 anim_test4 = selected_object_test4.animation_data_create()
117 anim_test4.action = bpy.data.actions.new(name="Rotate Animation Test4")
118
119 frame_start_test4 = frame_start_test1
120 frame_end_test4 = frame_end_test1
121
122 # Quinto script
123 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse X
124 min_angle_test5 = math.radians(max_EST_test5)
125 max_angle_test5 = math.radians(max_FLEX_test5)
126
127 anim_test5 = selected_object_test5.animation_data_create()
128 anim_test5.action = bpy.data.actions.new(name="Rotate Animation Test5")
129
130 frame_start_test5 = frame_start_test1
131 frame_end_test5 = frame_end_test1
132
133 # Sesto script
134 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse X
135 min_angle_test6 = math.radians(max_EST_test6)
136 max_angle_test6 = math.radians(max_FLEX_test6)
137
138 anim_test6 = selected_object_test6.animation_data_create()
139 anim_test6.action = bpy.data.actions.new(name="Rotate Animation Test6")
140
141 frame_start_test6 = frame_start_test1
142 frame_end_test6 = frame_end_test1
143
144 # Settimo script
145 # Specifica il range ammissibile degli angoli (in gradi) lungo l'asse X
146 min_angle_test7 = math.radians(max_EST_test7)
147 max_angle_test7 = math.radians(max_FLEX_test7)
148
149 anim_test7 = selected_object_test7.animation_data_create()
150 anim_test7.action = bpy.data.actions.new(name="Rotate Animation Test7")
151
152 frame_start_test7 = frame_start_test1
```

Figura 64 Script Simulazione 2-7A

```

155     for frame in range(frame_start_test1, frame_end_test1 + 1):
156         # Calcola l'angolo per il primo oggetto
157         t_test1 = (frame - frame_start_test1) / (frame_end_test1 - frame_start_test1)
158         angle_test1 = min_angle_test1 + (max_angle_test1 - min_angle_test1) * (1 + math.sin(2 * math.pi *
t_test1
159         - math.pi)) / 2
160         rotate_object_y(selected_object_test1, angle_test1)
161         selected_object_test1.keyframe_insert(data_path="rotation_euler", frame=frame)
162
163         # Calcola l'angolo per il secondo oggetto
164         t_test2 = (frame - frame_start_test2) / (frame_end_test2 - frame_start_test2)
165         angle_test2 = min_angle_test2 + (max_angle_test2 - min_angle_test2) * (1 + math.sin(2 * math.pi *
t_test2
166         - math.pi)) / 2
167         rotate_object_z(selected_object_test2, angle_test2)
168         selected_object_test2.keyframe_insert(data_path="rotation_euler", frame=frame)
169
170         # Calcola l'angolo per il terzo oggetto
171         t_test3 = (frame - frame_start_test3) / (frame_end_test3 - frame_start_test3)
172         angle_test3 = min_angle_test3 + (max_angle_test3 - min_angle_test3) * (1 + math.sin(2 * math.pi *
t_test3
173         - math.pi)) / 2
174         rotate_object_x(selected_object_test3, math.degrees(angle_test3))
175         selected_object_test3.keyframe_insert(data_path="rotation_euler", frame=frame)
176
177         # Calcola l'angolo per il quarto oggetto
178         t_test4 = (frame - frame_start_test4) / (frame_end_test4 - frame_start_test4)
179         angle_test4 = min_angle_test4 + (max_angle_test4 - min_angle_test4) * (1 + math.sin(2 * math.pi *
t_test4
180         - math.pi)) / 2
181         rotate_object_x(selected_object_test4, math.degrees(angle_test4))
182         selected_object_test4.keyframe_insert(data_path="rotation_euler", frame=frame)
183
184         # Calcola l'angolo per il quinto oggetto
185         t_test5 = (frame - frame_start_test5) / (frame_end_test5 - frame_start_test5)
186         angle_test5 = min_angle_test5 + (max_angle_test5 - min_angle_test5) * (1 + math.sin(2 * math.pi *
t_test5
187         - math.pi)) / 2
188         rotate_object_x(selected_object_test5, math.degrees(angle_test5))
189         selected_object_test5.keyframe_insert(data_path="rotation_euler", frame=frame)
190
191         # Calcola l'angolo per il sesto oggetto
192         t_test6 = (frame - frame_start_test6) / (frame_end_test6 - frame_start_test6)
193         angle_test6 = min_angle_test6 + (max_angle_test6 - min_angle_test6) * (1 + math.sin(2 * math.pi *
t_test6
194         - math.pi)) / 2
195         rotate_object_x(selected_object_test6, math.degrees(angle_test6))
196         selected_object_test6.keyframe_insert(data_path="rotation_euler", frame=frame)
197
198         # Calcola l'angolo per il settimo oggetto
199         t_test7 = (frame - frame_start_test7) / (frame_end_test7 - frame_start_test7)
200         angle_test7 = min_angle_test7 + (max_angle_test7 - min_angle_test7) * (1 + math.sin(2 * math.pi *
t_test7
201         - math.pi)) / 2
202         rotate_object_x(selected_object_test7, math.degrees(angle_test7))
203         selected_object_test7.keyframe_insert(data_path="rotation_euler", frame=frame)
204
205         frame_end = max(frame_end_test1,
frame_end_test2, frame_end_test3, frame_end_test4, frame_end_test5, frame_end_test6, frame_end_test7)
206         bpy.context.scene.frame_end = frame_end
207         bpy.ops.screen.animation_play()
208
209     # Aggiorna il testo negli oggetti di testo
210     if text_object_test1 is not None and text_object_test2 is not None and text_object_test3 is not None and
text_object_test4 is not None and text_object_test5 is not None and text_object_test6 is not None and
text_object_test7 is not None:
211         text_object_test1.data.body = f"Test 1:\nmax_ABD = {max_ABD_test1}°\nmax_ADD = {max_ADD_test1}°"
212         text_object_test2.data.body = f"Test 2:\nmax_INTRA = {max_EXTRA_test2}°\nmax_EXTRA =
{max_INTRA_test2}°"
213         text_object_test3.data.body = f"Test 3:\nmax_EST = {max_FLEX_test3}°\nmax_FLEX = {max_EST_test3}°"
214         text_object_test4.data.body = f"Test 4:\nmax_FLEX = {max_FLEX_test4}°\nmax_EST = {max_EST_test4}°"
215         text_object_test5.data.body = f"Test 5:\nmax_FLEX = {max_FLEX_test5}°\nmax_EST = {max_EST_test5}°"
216         text_object_test6.data.body = f"Test 6:\nmax_EST = {max_FLEX_test6}°\nmax_FLEX = {-max_EST_test6}°"
217         text_object_test7.data.body = f"Test 7:\nmax_EST = {max_FLEX_test7}°\nmax_FLEX = {-max_EST_test7}°"
218
219     # Aggiorna il testo nell'oggetto Text_v
220     text_object_v = bpy.data.objects.get("Text_v") # Assicurati che il nome sia corretto
221     if text_object_v is not None:
222         text_object_v.data.body = f"Velocità animazione (v=1/n): {v}x"
223
224     else:
225         print(f"Oggetto con nome '{object_name_test1}' o '{object_name_test2}' o '{object_name_test3}' o
'{object_name_test4}' o '{object_name_test5}' o '{object_name_test6}' o '{object_name_test7}' non trovato.
Assicurati di specificare nomi validi.")

```

Figura 65 Script Simulazione 2-7 B

Ringraziamenti

Con profonda gratitudine, desidero ringraziare l'Istituto Ortopedico Rizzoli e l'Università di Bologna per la collaborazione e la preziosa fornitura di dati fondamentali per questo studio. Un sentito ringraziamento a tutti i professionisti della salute e gli esperti che hanno contribuito con il loro know-how.

Un apprezzamento speciale va a coloro che hanno sostenuto lo sviluppo della metodologia innovativa nell'ambito della modellazione virtuale ortopedica. Il contributo di colleghi e mentori è stato fondamentale.

Infine, dedico un caloroso ringraziamento alla mia amata famiglia e alle persone care che mi hanno sostenuto con amore e pazienza. Il vostro supporto è stato la luce che ha illuminato il cammino verso questo traguardo.

Grazie a tutti per aver reso possibile questa straordinaria esperienza di ricerca e scoperta.