Politecnico di Milano
Corso di Laurea in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

CATTURA DEL MOVIMENTO SENZA MARCATORI MEDIANTE ITERATIVE CLOSEST POINT E PARTICLE FILTER

Gruppo di ricerca ispg
del Politecnico di Milano

Relatore: Prof. Marco MARCON
Correlatore: Ing. Eliana FRIGERIO

Tesi di Laurea di:
Emanuele PLEBANI, Matr. 642044

Anno Accademico 2010-2011

Alla mia famiglia

Sommario

La cattura e l'interpretazione del movimento umano senza marcatori da una o più telecamere è un campo di ricerca, emerso recentemente, che ha come obiettivo il riconoscimento della posa di una persona in un dato istante da una sequenza video, senza l'ausilio di speciali attrezzature.

L'inerente difficoltà del compito richiede tecniche specifiche. In questa tesi abbiamo sviluppato un algoritmo di inseguimento dei diversi arti nell'esecuzione di un'azione basato su Iterative Closest Point, al quale sono stati aggiunti vincoli sulla posizione e l'orientamento degli elementi del modello, e un passo di filtraggio mediante particle filter allo scopo di migliorare la stima della posa. In particolare, abbiamo valutato l'impatto dell'introduzione dei vincoli e dei particle filter sull'accuratezza di Iterative Closest Point applicato a modelli articolati.

L'algoritmo Iterative Closest Point è una tecnica che permette di allineare due nuvole di punti in tre dimensioni, ed è largamente usato nei casi in cui è necessario combaciare un modello a dati sperimentali. I particle filter sono una classe di filtri stocastici, che descrivono quindi l'evoluzione di un sistema dinamico come una successione di distribuzioni di probabilità, e sono in grado di gestire distribuzioni non gaussiane e sistemi dinamici non lineari.

L'algoritmo finale si è dimostrato sufficientemente accurato nella maggior parte delle sequenze di test. L'algoritmo si è dimostrato inoltre molto veloce e permette di ottenere tempi di elaborazione compatibili con il tempo reale senza perdite apprezzabili di accuratezza.

Ringraziamenti

Desidero ringraziare prima di tutti il prof. Marco Marcon per avermi seguito in questo lungo percorso di tesi, per avere stima delle mie capacità e per la pazienza dimostrata con gli infiniti rinvii. Finalmente ora il lavoro è finto! Ringrazio Eliana Frigerio per essere stata sempre disponibile a darmi consigli e correzioni, nonostante gli innumerevoli impegni di dottorato. Grazie a lei ho potuto sempre contare su qualcuno che potesse rispondere ai miei dubbi e alle mie incertezze. Ringrazio il prof. Stefano Tubaro per i suoi preziosi consigli. Anche se ci siamo visti solo un paio di volte durante la tesi, le sue osservazioni mi hanno sempre indirizzato in direzioni originali e promettenti.

Un grazie a tutte le persone che ho incontrato negli anni dell'università. Ringrazio in particolare Lele per avermi aiutato a orientarmi a Milano nei primi due anni di università, quando non avevo nessun altro amico su cui fare affidamento. Ringrazio Lorenzo D. per aver condiviso con me non solo gli anni di liceo ma anche dell'università e per avermi fatto capire che esiste altro oltre al dovere. Ringrazio Imer per essersi sempre preoccupato per me, anche quando avevo un carattere impossibile. Ringrazio Antonino per i dotti discorsi; Elia, Faifer e le altre teste matte dell'alta Valtellina per aver reso più sopportabili le lunghe ore di lezione. Ringrazio Bruno e Annalisa per avermi fatto conoscere nuovi amici nelle uscite del quinto anno.

Ringrazio in modo particolare Don Bepino, per avermi detto due anni fa le parole giuste che nessun altro era riuscito a dire, perché senza di lui sicuramente non sarei riuscito a completare questo percorso universitario e a scrivere questa tesi. Ringrazio tutte le persone che ho conosciuto in questi anni, soprattutto Matteo per avere creduto sempre in me, Paolo per aver sempre mostrato interesse in quello che facevo anche quando non poteva essere più lontano dai suoi interessi, e Mosè per le sue inesauribili freddure e per avermi fatto conoscere tanti suoi amici. Ringrazio Max per aver condiviso con me la sua passione per l'informatica, Francesco per aver condiviso la

sua passione per la fisica, e Paolo C. per avermi fatto conoscere il suo modo di vedere il mondo. Ringrazio anche Roberto per la (troppa) ammirazione che nutre nei miei confronti, Alfredo, Luigi, Daniela e gli altri di Bonate per le gite in montagna e per avermi tenuto un po' lontano dagli esami e dalla tesi.

Ringrazio la mia famiglia per avermi supportato in questi anni e per l'aiuto che mi hanno dato. In particolare ringrazio mio padre e mia madre per la pazienza che hanno avuto, anche quando non andavo avanti con gli studi, e per essere stati sempre disposti ad aiutarmi. Ringrazio mio fratello Simone per le corse, i giri in bici e per avermi incoraggiato a modo suo. Ringrazio mio fratello Damiano per tutti i discorsi sul design, sull'università, sulla musica e sulla vita in generale e per aver sempre creduto che io possa ottenere qualcosa nella vita.

Ringrazio infine tutte le persone e le esperienze che mi hanno influenzato in tutti questi anni, e che mi hanno permesso di arrivare a questo traguardo.

Milano, 23 Giugno 2011

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

La cattura e l'analisi del movimento, senza particolari ausili o sensori ma solo tramite l'impiego di telecamere, è uno dei settori più attivi nel campo della visione artificiale. Spinte dall'importanza dei possibili ambiti applicativi, iniziano ad essere sviluppate le prime applicazioni su scala commerciale, come Microsoft Kinect per il controllo di una console di gioco o Organic Motion per la cattura del movimento nell'ambito cinematografico.

Il campo della cattura del movimento riguarda i metodi che permettono di ricostruire il movimento delle principali articolazioni umane, allo scopo di usare i valori così ottenuti per animare un attore virtuale o per analizzare le caratteristiche del movimento. L'analisi del movimento può essere utile in campo medico per valutare i risultati di una terapia di riabilitazione o nella videosorveglianza di anziani e disabili, per segnalare tempestivamente situazioni di potenziale pericolo. La capacità di interpretare la gestualità umana da parte di un robot permette di gestire in maniera più naturale l'interazione uomo-macchina, e in un dispositivo dotato di telecamera permette di interagire senza alcun contatto o dispositivo ausiliario. La classificazione delle azioni umane rende possibile l'analisi automatica dei filmati di sorveglianza alla ricerca di comportamenti potenzialmente pericolosi, oppure l'etichettatura automatica di video per l'indicizzazione basata sul contenuto. In campo videoludico, la cattura del movimento umano può sostituire le tradizionali periferiche di controllo permettendo un'esperienza più dinamica e immersiva.

Nella quasi totalità delle applicazioni commerciali la cattura del movimento avviene attraverso l'uso di un qualche tipo di marcatore (ad esempio, palline colorate, accelerometri o giroscopi) che devono essere indossati dall'attore e che segnalano in modo non ambiguo la posizione delle principali articolazioni del corpo umano. Se la presenza di marcatori è accettabile

negli ambienti controllati che si incontrano in campo medico e soprattutto nel campo cinematografico, nell'ambito della sorveglianza e dell'interazione uomo-macchina è un requisito del tutto irrealistico e impraticabile.

Per questo motivo esistono forti motivazioni per lo sviluppo di metodi capaci di riconoscere i movimenti delle persone senza marcatori e in ambienti dove non è possibile controllare illuminazione e punto di vista. Questo compito è immediato per la mente umana, plasmata da un lungo processo di evoluzione e dalle interazioni con altre persone, mentre per un computer l'interpretazione dei dati è molto più difficile. Differenze nell'aspetto delle persone, differenze di posa, di punto di vista e occlusioni rendono la forma del corpo umano molto differente da situazione a situazione, rendendo difficile il compito di riconoscimento. Ad ora non esiste un sistema di visone artificiale capace di superare in maniera affidabile tutte queste difficoltà, e quindi nel laboratorio abbiamo scelto di lavorare nell'ambito meno realistico ma più semplice da affrontare della cattura del movimento con diverse telecamere sincronizzate e calibrate: in questo modo la dipendenza dal punto di vista e le occlusioni possono essere in gran parte rimosse.

In questa tesi è stato sviluppato un sistema di cattura del movimento umano che, a partire da un volume che rappresenta il corpo di una persona in un dato istante, cerca di estrarre nel modo più fedele possibile la posa della persona. Per raggiungere questo obiettivo è stato definito un modello articolato del corpo umano che rappresenta la forma delle varie parti del corpo e le caratteristiche delle varie articolazioni. Per il calcolo della posa abbiamo scelto l'algoritmo *Iterative Closest Point*, già usato con successo nell'ambito della cattura del movimento. La posa ottenuta non è sempre affidabile, e per migliorare la qualità della cattura abbiamo deciso di usare la tecnica dei *particle filter*, che permette simultaneamente di integrare un modello cinematico del movimento e di gestire l'inevitabile rumore nei dati.

Nella scelta delle tecniche abbiamo posto particolare enfasi nella velocità di esecuzione dell'algoritmo, indispensabile per applicazioni in tempo reale e interattive. Il metodo sviluppato quindi è molto veloce e richiede un tempo dell'ordine del decimo di secondo per elaborare la maggior parte dei fotogrammi su un moderno computer di fascia bassa (AMD M320 Dual Core).

La tesi è strutturata nel modo seguente.

Nel secondo capitolo è presentata una panoramica dello stato dell'arte del campo, con una particolare enfasi sui lavori che si basano su tecniche simili a quelle scelte per questa tesi, anche se applicate in contesti diversi.

Nel terzo capitolo è descritto sinteticamente il sistema di acquisizione dei dati.

Nel quarto capitolo è offerta una trattazione teorica delle tecniche usate nella tesi.

Nel quinto capitolo è descritta nel dettaglio la strategia adottata.

Nel sesto capitolo sono riportati i risultati sperimentali e le prestazioni dell'algoritmo sviluppato.

Il settimo capitolo infine conclude il lavoro e ne discute i possibili sviluppi.

L'appendice A è un breve excursus sui metodi statistici a integrazione dell'algoritmo dei particle filter.

L'appendice B contiene alcuni dettagli implementativi importanti per il lavoro di tesi.

Capitolo 2

Stato dell'arte

Il campo della cattura e analisi del movimento umano ha visto uno sviluppo notevole negli ultimi anni, sotto la spinta degli avanzamenti nel campo della visione artificiale e dei possibili ambiti d'uso. In questo capitolo offriamo una discussione delle categorie di applicazioni possibili e una breve panoramica della letteratura, con particolare attenzione ai metodi relativi all'ambito di questa tesi.

2.1 Applicazioni

Il campo delle applicazioni è molto vasto. Seguendo la classificazione di Moeslund[48] si possono individuare tre principali categorie:

Controllo: Il riconoscimento di gesti simbolici permette il controllo di un computer o altri dispositivi mediante il semplice movimento del corpo umano. Il tempo di risposta deve essere molto breve, perché è richiesta un'interazione in tempo reale. I comandi possono essere simbolici, come un saluto, o parametrici, come un dito che indica un punto preciso e può essere usato come sostituto del mouse di un computer. In questa categoria rientrano anche le interfacce emozionali, che permettono a un computer di reagire al linguaggio del corpo dell'utente e non richiedono il riconoscimento di sequenze specifiche.

Analisi: Riguarda tutte le applicazioni in cui è necessaria un'analisi dettagliata del movimento. In questa categoria rientrano quindi la cattura del movimento nel campo del cinema, l'analisi dei movimenti a scopo medico o sportivo, la codifica compressa di video di persone in movimento e più in generale la costruzione di modelli per l'analisi

del contenuto di un video. In queste applicazioni è più importante l'accuratezza, a scapito della velocità.

Sorveglianza: In questa categoria rientrano tutte le applicazioni che riguardano il monitoraggio automatico di ambienti affollati o obbiettivi importanti, alla ricerca di comportamenti anomali o potenzialmente pericolosi. É molto importante la capacità di seguire indipendentemente un gran numero di persone, la gestione di video a bassa risoluzione come quelli delle telecamere di sicurezza, e la capacità di funzionare in ambienti nei quali solitamente non è possibile controllare le condizioni di acquisizione dei dati. La precisione nel rilevamento dei movimenti è invece secondaria.

Da questa classificazione sono escluse le applicazioni di visione robotica e di *learn-by-example*, che possono essere considerate come casi molto particolari della categoria del controllo. Nel caso del learn-by-example, l'obiettivo è insegnare a un robot un movimento mostrando la stessa azione svolta da un attore umano, e quindi la fase di cattura e interpretazione del movimento è fondamentale. Un caso di notevole interesse nella categoria del controllo riguarda le applicazioni di *augmented reality*, nelle quali è possibile vedere e interagire con oggetti virtuali sovrapposti a una scena reale, e dove quindi un'interfaccia gestuale è molto comoda. Un esempio importante in questa direzione è il progetto Sixth Sense del Media Lab del MIT (`http://www.pranavmistry.com/projects/sixthsense/`).

Come è evidente dai requisiti delle diverse categorie, a seconda dell'applicazione devono essere scelti diversi compromessi, e questo, a sua volta, influenza pesantemente le caratteristiche degli algoritmi e delle strategie scelte. Nel campo della sorveglianza, ad esempio, si considera frequentemente il caso in cui si devono estrarre più informazioni possibili da una singola telecamera, mentre nel caso dell'analisi del movimento si assume spesso la presenza di più telecamere calibrate e sfondo neutro. Negli anni quindi si sono sviluppati filoni di ricerca separati per le varie categorie di applicazioni.

In questo campo la ricerca scientifica costituisce la maggior parte del lavoro svolto nel campo, ma negli anni sono state sviluppate alcune applicazioni commerciali. Il pioniere nel campo è sicuramente Gesturetek (`http://www.gesturetek.com/`), che propone soluzioni di riconoscimento dei gesti nell'ambito del controllo dagli anni '80. Tuttavia solo negli ultimi anni è esploso l'interesse per il settore, spinto dal continuo miglioramento dell'accuratezza e della robustezza degli algoritmi proposti. Gran parte delle applicazioni commerciali attuali riguardano l'interazione uomo-macchina, nelle quali è permessa una maggiore tolleranza agli errori e dove è possibile un mag-

giore controllo sulle condizioni dell'ambiente di acquisizione. In quest'ambito si trovano ad esempio Softkinetic (`http://www.softkinetic.net/`) e la periferica di controllo Microsoft Kinect (`http://www.xbox.com/it-IT/kinect`). Al di fuori della categoria del controllo un notevole esempio è invece OrganicMotion (`http://www.organicmotion.com/`), che offre soluzioni di cattura del movimento senza marcatori nel campo biomedico e cinematografico.

2.2 Rassegna della letteratura

Il consistente numero di articoli pubblicati e la varietà delle strategie esplorate rende impossibile una trattazione esaustiva del campo in questa tesi. Per un quadro più completo dello stato dell'arte si rimanda ai numerosi articoli di rassegna. In particolare, l'articolo di Moeslund[48] e il suo aggiornamento scritto a distanza di cinque anni[49] offrono la rassegna più completa e approfondita delle varie strategie usate nei vari campi di applicazione. Wang, Hu e Tan[70] offrono una panoramica orientata maggiormente alle applicazioni di sorveglianza, con un'enfasi sull riconoscimento delle azioni umane e sull'implementabilità in situazioni reali dei metodi descritti. Aggarwal e Cai[1] si concentrano sulle tecniche usate nei problemi di estrazione di un modello, inseguimento delle traiettorie e riconoscimento delle azioni. Gavrila[26] offre uno sguardo generale sulle tecniche che sfruttano un modello, in due o tre dimensioni, per semplificare il processo di riconoscimento. Turaga, Chellappa *et al.*[68] infine si concentrano sul problema di riconoscere azioni o attività da un video.

Le varie tecniche presentate in letteratura sono classificabili a livello molto generale in due categorie principali, in base all'uso o meno di un modello esplicito della struttura del corpo e del movimento umano. Nelle prossime sezioni esaminiamo le caratteristiche principali dei metodi che fanno uso di un modello e dei metodi che invece costruiscono una rappresentazione implicita. Per ciascuna categoria offriamo alcuni esempi significativi presi dalla letteratura. Infine esaminiamo più in dettaglio i metodi che fanno uso dell'algoritmo Iterative Closest Point, per la loro rilevanza in questa tesi.

2.2.1 Metodi con modello esplicito

I metodi basati su un modello esplicito costruiscono una rappresentazione della struttura del corpo umano: ad esempio, ogni parte del corpo può essere rappresentata da ellissoidi, vincolati a muoversi l'uno rispetto all'altro rispettando limiti biomeccanici. Il modello è tipicamente costruito a partire

da dati ad alta definizione (scanner laser, acquisizione del movimento con marcatori) oppure sintetizzato a partire da informazioni antropometriche. Se da una parte la scelta di un modello rende più complesso l'algoritmo e ne riduce il campo di applicabilità, dall'altra riduce la dimensionalità dello spazio della configurazione a decine di parametri del modello, dalle migliaia di pixel o voxel (l'analogo tridimensionale del pixel) dei dati acquisiti.
Spesso, per semplificare il problema della ricerca della posa ad un dato istante, viene fatta l'ipotesi che l'evoluzione tra la posa ad un dato istante e quella ad un istante successivo non presenti variazioni repentine ma sia caratterizzata da un'evoluzione fluida, ipotesi quasi sempre valida purché la frequenza dei fotogrammi in acquisizione sia sufficientemente alta. Si assume inoltre che, all'istante iniziale, siano disponibili i parametri del modello e i valori della posa iniziale. Quindi, nella maggior parte dei casi, l'algoritmo di riconoscimento richiede due fasi principali:

- inizializzazione del modello
- calcolo dei parametri della posa ad ogni passo

Le caratteristiche del modello possono essere definite prima dell'analisi del movimento, oppure il modello può essere costruito con le informazioni raccolte durante l'analisi. Nelle successive sezioni esaminiamo queste due possibilità separatamente.

Modello a priori

In questa categoria rientrano i metodo che costruiscono un modello a partire da conoscenze sulla cinematica e sulla dinamica del corpo umano. Anche se il modello può avere dei parametri liberi, che corrispondono spesso a dimensioni e lunghezza delle parti del corpo, la struttura del modello è fissa: proprietà come il numero di elementi o topologia delle connessioni fra elementi non può essere variata.

Nella fase di inizializzazione vengono stimati parametri, che possono essere grandezze intuitive come la lunghezza degli arti e la posa iniziale, oppure possono definire in casi più avanzati un modello tridimensionale e deformabile del corpo umano. La fase successiva consiste nella stima delle variabili che definiscono la posa, spesso ottenuta attraverso la minimizzazione di una opportuna funzione errore. La funzione errore è scelta in modo da codificare al meglio la differenza fra la posa del modello e la posa dell'individuo reale. La procedura di inizializzazione può richiedere l'acquisizione di una singola posa predefinita (ad esempio, la posizione in piedi con braccia alzate) oppure richiedere una serie di movimenti complessi.

Mikić, Trivedi *et al.*[46] hanno costruito un modello tridimensionale semplificato del corpo umano con cilindri ed ellissoidi Il modello è applicato a un volume tridimensionale, costruito a partire da quattro telecamere con il metodo della *Visual Hull* (vedi Sezione 3.3). La posa e i parametri iniziali sono ottenuti mediante euristiche, e la posa in ogni fotogramma è calcolata con l'aiuto di un filtro di Kalman esteso.

Deutscher, Blake e Reid[20] hanno applicato la tecnica dei particle filter all'inseguimento della posa in immagini monoculari e hanno elaborato una variante dell'algoritmo, chiamato *annealed particle filter*, per superarne la sensibilità all'alto numero di dimensioni dello spazio dei parametri.

I metodi statistici in questo campo sono molto popolari. Un esempio è il lavoro di Kehl *et al.*[35], nel quale è stata applicata la tecnica della *stochastic meta descent* per rendere più veloce e robusta la convergenza del modello a un volume tridimensionale. La fase di inizializzazione avviene mediante l'acquisizione di una posa predefinita.

I modelli non sono necessariamente tridimensionali. Sigal e Black[64] hanno costruito un modello articolato formato da trapezi, al quale sono stati aggiunti vincoli cinematici. L'errore nel calcolo della posa è reso da una funzione di verosimiglianza e la posa che spiega meglio i dati è ottenuta con una variante dell'algoritmo di *belief propagation*.

É possibile definire il modello unicamente in termini di parti del corpo, e ridurre il problema della cattura del movimento alla localizzazione nello spazio della posizione dei giunti corrispondenti ad ogni parte. Questo approccio è stato usato da Shotton *et al.*[62] e implementato nella periferica Kinect. I pixel dell'immagine sono assegnati a una parte del corpo mediante l'uso di una foresta di alberi di decisione[9], costruita a partire da un grande numero di esempi, e la posizione finale è data da uno stimatore con parametri ottimizzati per ogni parte del corpo.

Spesso tuttavia si ricorre a un modello tridimensionale, anche nel caso di analisi di sequenze monoculari, perché i vincoli spaziali sui movimenti delle articolazioni umane permettono di risolvere almeno parzialmente le ambiguità dovute a un singolo punto di vista. Balan, Sigal *et al.*[3] ricorrono a un modello dettagliato del corpo umano basato sulla parametrizzazione SCAPE[2] delle variazioni di corporatura. La posizione del modello è inizializzata facendo uso di marcatori; quindi ad ogni passo viene eseguita prima una predizione della posa, mediante annealed particle filter, e la posa reale viene poi calcolata con un metodo di ottimizzazione stocastica. Nonostante l'errore sia calcolato solo sulla silhouette, il sistema dimostra una buona accuratezza nella ricostruzione della posa tridimensionale.

Modello derivato dai dati

Il modello può essere estratto direttamente dai dati acquisiti: in questo caso la fase di inizializzazione consiste nella stima dei parametri e nella stima della struttura del modello. Per ridurre significativamente lo spazio molto vasto dei possibili modelli si ricorre spesso ad assunzioni specifiche. Ad esempio, si possono prendere in considerazione solo i modelli che rappresentano il corpo con una rete di blocchi collegati da articolazioni.

Chu, Jenkins e Mataric[14] estraggono un modello ad albero del corpo umano a partire dai dati volumetrici. Per ridurre le ambiguità dovute alle variazioni di posa e di corporatura, il volume è riparametrizzato in uno spazio indipendente dalla posa.

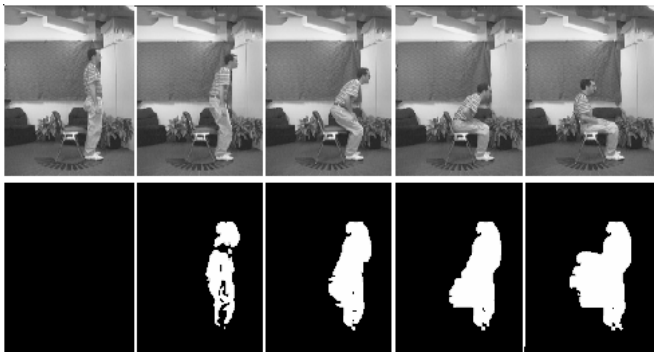
Theobalt, de Aguiar *et al.*[17] adattano un numero elevato di ellissoidi al volume acquisito e con un algoritmo di *split-and-merge*, sviluppato in un lavoro precedente[67], calcolano dei cluster dai quali è possibile estrarre un modello scheletrico. La ricerca di corrispondenze fra gli ellissoidi in fotogrammi consecutivi permette di seguirne il movimento sull'intera sequenza e di etichettare in modo consistente ogni parte del corpo, inoltre questo tipo di analisi globale permette di raffinare il modello scheletrico risolvendo i fotogrammi ambigui, come ad esempio i casi in cui le gambe dell'attore sono unite.

Spesso il modello scheletrico è definito a priori, mentre la forma tridimensionale è adattata caso per caso con lo scopo di aumentare l'accuratezza della cattura. Ad esempio Plänkers e Fua[58] hanno costruito un modello volumetrico attraverso un numero di metaball di diversa dimensione (le metaball sono superfici che si deformano in prossimità di altre metaball e sono adatte a rappresentare forme organiche).

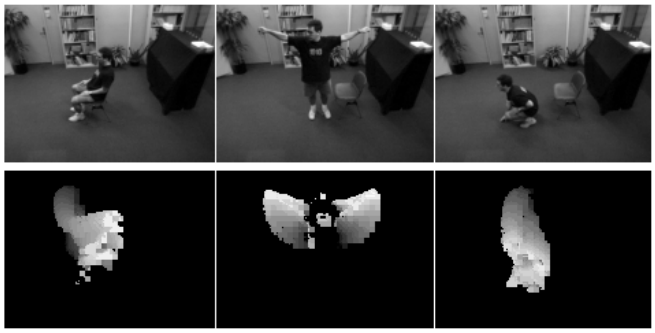
2.2.2 Metodi senza modello

Il secondo possibile approccio al riconoscimento non fa uso di un modello esplicito. Dai dati viene estratto direttamente un vettore di caratteristiche (*feature vector*) che poi viene elaborato da un algoritmo di riconoscimento automatico. Il vettore può essere ricavato dal modello tridimensionale, oppure direttamente dalle immagini.

In questa classe uno dei metodi storicamente più importanti è il metodo delle *Motion Energy Image* (MEI) e *Motion History Image* (MHI) proposto da Bobick e Davis[6]. Le MEI e le MHI sono costruite a partire dalle silhouette della persona che esegue il movimento (vedi Figura 2.1). Le MEI sono ottenute costruendo su un certo numero di fotogrammi l'unione insiemistica delle silhouette, viste come insiemi di punti. Le MHI sono ancora una unione



(a) Motion Energy Image



(b) Motion History Image

Figura 2.1: **MHI e MEI.** *Motion Energy Image per diversi intervalli nell'azione "sedersi"* **(a)** *e Motion History Image per tre differenti azioni* **(b)**. *Immagini prese dalla pagina personale di James W. Davis (*`http://www.cse.ohio-state.edu/~jwdavis/`*).*

di silhouette su un intervallo di tempo, con la differenza che ora l'intensità del pixel dipende dalla "età" della silhouette più recente che lo contiene: i pixel interessati dal movimento sono quindi progressivamente più chiari, creando una caratteristica "scia". Questi descrittori incorporano informazioni sul carattere del movimento e sulle sue caratteristiche dinamiche. Una rappresentazione compatta, in termini di momenti di Hu, è estratta dalle MEI e MHI e confrontata con template per classificare il gesto. Il metodo è stato successivamente esteso con i *Motion History Gradient*[7], applicato a dati tridimensionali nella forma dei *Motion History Volume*[72], e usato per costruire una firma del movimento denominata *Action Signature*[10].

Un altro approccio che ha ricevuto molto interesse negli ultimi anni è dato dai modelli che rappresentano il movimento con un insieme di caratteristiche locali derivate da ogni fotogramma o dalla sequenza. Niebles *et al.*[53] costruiscono il vettore di caratteristiche sovrapponendo i fotogrammi

di un video uno sopra l'altro, e applicando un filtro al volume così ottenuto. Nella fase di riconoscimento è poi applicata la tecnica della *Latent Semantic Analysis* per classificare sequenze di pattinaggio di figura. Liu e Sha[42] hanno estratto elementi tridimensionali da un video, denominati *cuboid*, che sono stati poi classificati con metodi statistici e confrontati con un template per il riconoscimento. Recentemente, è stato proposto da Yao e Fei-Fei il metodo delle *grouplets*[73] per l'analisi di interazioni fra persone ed oggetti, con buone prestazioni nella classificazione di persone che suonano uno strumento.

2.2.3 Iterative Closest Point nella cattura del movimento

Nell'ambito della cattura del movimento a partire da dati volumetrici, l'algoritmo *Iterative Closest Point* (ICP)[4] si è dimostrato particolarmente efficace, ed è stato scelto da alcuni gruppi di ricerca nel loro sistema di analisi del movimento. ICP calcola la trasformazione rigida che meglio sovrappone due nuvole di punti in tre dimensioni, e quindi è adatto al compito di allineare un modello con i dati acquisiti; tuttavia richiede corpi rigidi, e nel caso di corpi articolati richiede delle opportune modifiche. Ogni gruppo ha elaborato una sua originale soluzione a questo problema.

Demirdjian e Darrell[18] hanno sviluppato un sistema di acquisizione del movimento del torso e degli arti basato su un modello semplificato del corpo. La testa è rappresentata da una sfera, il torso, le braccia e le gambe da cilindri. ICP è applicato ad ogni parte del corpo indipendentemente, e le variabili di giunto ottenute sono poi proiettate in uno spazio di valori ammessi, rappresentato da un vincolo lineare. Il vincolo è stato ulteriormente migliorato con l'apprendimento dello spazio dei possibili movimenti[19]. Il lavoro è diventato poi la base di successive indagini sul riconoscimento, come ad esempio Wang, Quattoni *et al.*[71].

Knoop *et al.*[38], partendo dal lavoro di Demirdjian, hanno integrato un metodo più sofisticato per la codifica dei vincoli, permettendo giunti con differenti gradi di libertà[37]. Questi vincoli, definiti da Knoop *et al.* "morbidi" in contrasto con i vincoli "duri" di Demirdjian, lasciano un margine di errore nel soddisfacimento dei vincoli. Questo, in unione ai limiti del movimento dati dai diversi tipi di giunto, garantisce una migliore convergenza di ICP alla posa corretta. Oltre a dati volumetrici ottenuti da una telecamera stereoscopica, il sistema è in grado di gestire punti di interesse (come il volto) ottenuti dalle immagini acquisite. I parametri del modello sono ottenuti in una fase di inizializzazione da euristiche applicate alla mappa di

profondità, e non è quindi necessaria una posa predefinita[43]. Il sistema è in grado di funzionare in real-time, essendo pensato per la visione robotica.

Mundermann, Corazza *et al.*[15] hanno costruito un modello articolato accurato del corpo umano basandosi sulla già citata parametrizzazione SCAPE. I parametri del modello sono ricavati da una posa predefinita e, diversamente dagli approcci precedenti, nella fase di allineamento il termine di errore dell'algoritmo ICP non viene minimizzato separatamente per ogni parte, ma minimizzato globalmente con il metodo di Levenberg-Marquardt. I vincoli fra le varie parti del corpo sono stati modellati originariamente da un termine di errore, che calcola la discrepanza fra le posizioni delle estremità in corrispondenza del giunto[50], e nelle versioni successive da un vincolo sui valori di traslazione e rotazione dei giunti applicato ad ogni passo del metodo di ottimizzazione[16]. In questo approccio è stata posta maggiore enfasi sull'accuratezza. Con un errore dell'ordine di 1,5 cm nella posizione dei giunti sul dataset HumanEva[63], è stato indicato come stato dell'arte nel campo della cattura del movimento in ambiente controllato e con più telecamere calibrate[65].

2.3 Considerazioni finali

Da questa breve trattazione emerge che non esiste un metodo univoco e definitivo per eseguire l'analisi del movimento umano, neppure all'interno di un ambito specifico. Nel campo sono state applicate le tecniche più disparate di apprendimento automatico, sono state elaborate diverse strategie originali e ogni anno ne vengono aggiunte di nuove all'arsenale dei ricercatori. Tuttavia nessuna strategia è emersa chiaramente al di sopra delle altre, a differenza di altri campi della visione artificiale come il riconoscimento dei volti (Viola *et. al.*[69]). Questo significa che è necessario scegliere con cura il metodo in base al tipo di dati disponibili e agli obiettivi che si vogliono raggiungere.

Per la tesi, sviluppata nell'ambito della cattura del movimento con dati tridimensionali, abbiamo scelto l'algoritmo ICP non solo perché è stato applicato con successo in diverse occasioni, ma perché si è dimostrato adattabile a diversi ambiti applicativi (dalle interfacce uomo–macchina alle analisi biometriche) e scalabile in termini di velocità di esecuzione. Questo ci permette di sviluppare un sistema che può essere successivamente esteso e migliorato anche al di là degli obiettivi inizialmente fissati.

La scelta dei particle filter non solo è in linea con il gran numero di esempi in letteratura, ma continua il lavoro di un precedente tesista[11] applicandolo in un ambito significativamente diverso.

Capitolo 3

Sistema di acquisizione

In questo capitolo è descritto il sistema di acquisizione del movimento di una persona da viste multiple messo a punto nel gruppo ISPG del Politecnico di Milano. Il sistema è progettato in modo da acquisire un'azione compiuta da una persona da diversi punti di vista e costruirne una rappresentazione tridimensionale adatta a successive analisi. L'attuale configurazione e i dati volumetrici sono stati costruiti da Miorelli[47], basandosi sul lavoro precedente di Caravello[12].

I dati di acquisizione sono stati salvati in un formato specifico, discusso nella prima sezione dell'Appendice B.

3.1 Sistema di acquisizione da viste multiple

Otto telecamere sincronizzate e calibrate sono disposte in una stanza opportunamente preparata, denominata *Smart Space* (Figura 3.1), di dimensioni $6,8 \times 5 \times 2,5m$. Quattro telecamere sono disposte a un'altezza di un metro sui quattro spigoli laterali della stanza, e quattro sono disposte a metà dei quattro spigoli superiori. La risoluzione è 1024×768, la lunghezza focale 4,8 mm, e la velocità è 30 fotogrammi al secondo[11]. Le riprese avvengono su sfondo uniforme costituito da tende di colore blu, e la stanza ha illuminazione uniforme. Questi accorgimenti servono a ridurre l'impatto delle ombre nella fase di elaborazione delle immagini.

3.1.1 Calibrazione

La telecamera è rappresentata da un classico modello pin-hole con termini di distorsione radiale e tangenziale. La calibrazione della telecamera avviene in due fasi.

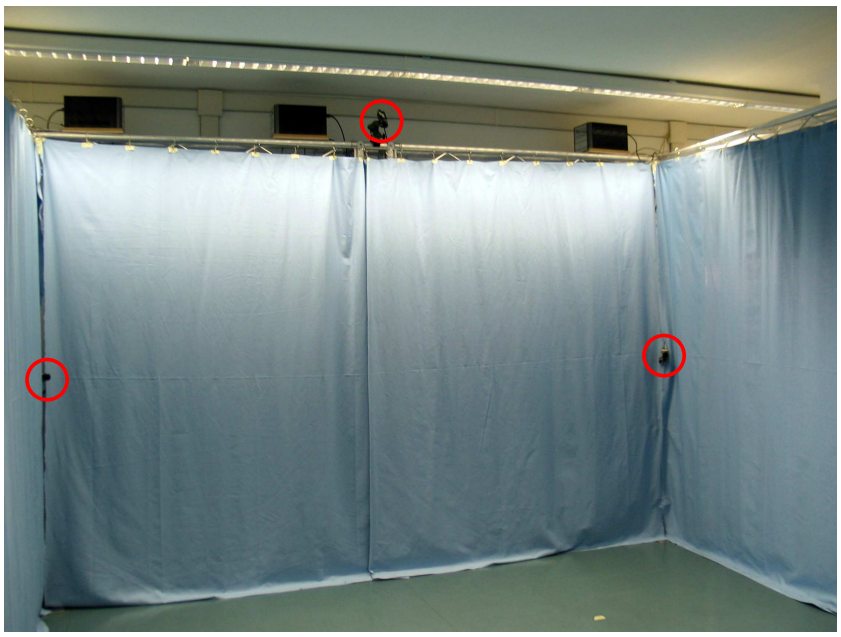


Figura 3.1: **L'ambiente Smart Space.** *Nell'immagine è mostrata una parte del sistema realizzato dall laboratorio ISPG, dove sono evidenziate tre delle otto telecamere. Sono visibili le tende azzurre usate per creare uno sfondo uniforme.*

La misura dei parametri intrinseci (lunghezza focale, dimensione dei pixel, punto principale) e di distorsione avviene mediante l'acquisizione di più immagini di una scacchiera, mostrata in Figura 3.2, da differenti punti di vista.

La misura dei parametri estrinseci (centro ottico e matrice di rototraslazione) avviene mediante l'acquisizione, da parte di tutte le telecamere dello spazio, del movimento di una sbarra di dimensione nota. Due palline di colore diverso poste agli estremi facilitano il rilevamento della sbarra, e la sincronizzazione fra le telecamere aiutano ad identificare le corrispondenze fra i fotogrammi delle sequenze acquisite. Il movimento offre un numero sufficiente di punti per la calibrazione, mentre la correlazione fra immagini prese da telecamere diverse offre informazioni sulla loro mutua posizione, aumentando la robustezza dell'algoritmo.

Il calcolo dei parametri avviene mediante l'uso di un algoritmo basato sul *bundle adjustement* e sviluppato per lo Smart Space in una precedente tesi[12].

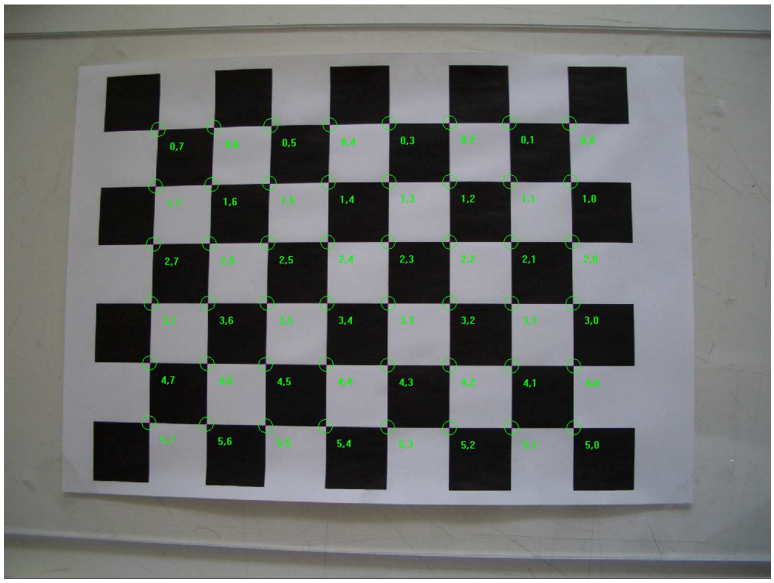


Figura 3.2: **Scacchiera di calibrazione.** *In figura si vede la scacchiera usata in fase di calibrazione. Gli angoli interni dei quadrati sono stati correttamente identificati e ad ogni angolo è stata assegnata una numerazione riga-colonna, in basso a destra rispetto all'angolo.*

3.2 Segmentazione

L'algoritmo di ricostruzione del volume richiede l'estrazione della silhouette della persona dalle immagini acquisite. L'ambiente controllato dello Smart Space e la presenza di un solo attore per sequenza permette di calcolare le silhouette con un semplice algoritmo di segmentazione figura–sfondo. L'algoritmo, sviluppato da Miorelli[47] è ispirato alla tecnica della *Running Gaussian Average* (RGA) ed è applicato allo spazio di colore *Hue-Saturation-Value* (HSV). Lo sfondo è descritto da un semplice modello gaussiano con media e varianza, e questi parametri sono appresi da una sequenza di 20 fotogrammi a stanza vuota situati all'inizio della sequenza. Allo scopo di ridurre l'uso di memoria, media e varianza sono calcolate incrementalmente usando due immagini come accumulatori; dopo la fase di apprendimento, nei successivi fotogrammi i valori ottenuti sono mantenuti fissi e servono a definire il modello di sfondo per la sequenza. La segmentazione avviene attraverso un test binario: un pixel è considerato parte della silhouette se la differenza fra il suo valore di tonalità (*Hue*) e la sua media locale supera la sua varianza locale scalata di un valore α solitamente fissato a 3 o 4. Chiamata I_n la tonalità del pixel, μ_n e σ_n rispettivamente la media e la varianza nella posizione del pixel, la formula per determinare se un pixel appartiene

al foreground è:

$$|I_n - \mu_n| \geq \alpha\sigma_n \tag{3.1}$$

La media rappresenta il colore dello sfondo, e potendo variare nell'immagine permette di descrivere variazioni di illuminazione o di ombreggiatura delle tende. La varianza rappresenta il livello di rumore, e permette di tollerare lievi deviazioni casuali nel colore dei pixel.

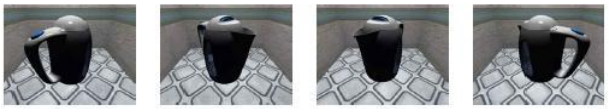
3.2.1 Post-processing

Il modello gaussiano usato nella segmentazione è in grado di eliminare gran parte degli errori dovuti a rumore o a ombre, ma nella silhouette rimane molto spesso del rumore residuo o delle regioni spurie. Il risultato della fase precedente è sottoposto quindi a ulteriori elaborazioni. Un filtro mediano 5x5 elimina il rumore di tipo "sale e pepe", che consiste in pixel isolati di colore differente dai pixel circostanti. Errori di scala superiore, come i buchi all'interno della silhouette, sono rimossi applicando all'immagine l'operatore di apertura morfologica. L'operatore tende tuttavia ad eliminare piccoli anelli di pixel correttamente identificati, nel caso la silhouette non sia semplicemente connessa, e per questa ragione l'operatore è applicato solo nelle regioni in cui il colore dei pixel è significativamente differente dal colore dello sfondo. La massima regione connessa è infine selezionata ricorrendo ancora ad operatori morfologici[47] e identificata come la silhouette dell'attore.

3.3 Ricostruzione tridimensionale

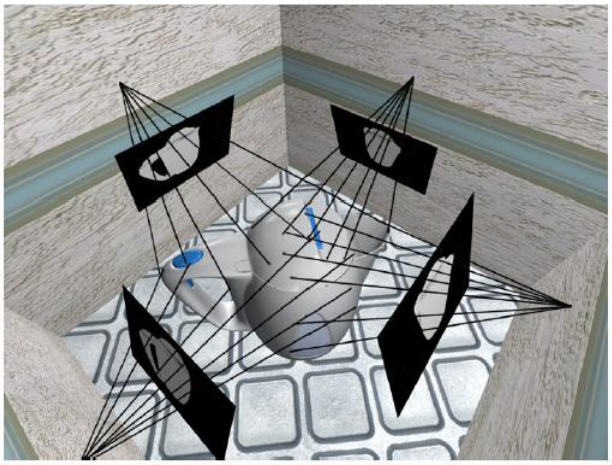
Il volume è ottenuto mediante la tecnica della *Visual Hull*[41], i cui passi principali sono schematizzati nella Figura 3.3. A partire da ogni silhouette ottenuta dal passo di segmentazione viene costruito un cono generalizzato, il vertice del quale coincide con l'origine del sistema di riferimento della telecamera corrispondente alla silhouette. Tutti i coni così ottenuti sono intersecati fra di loro per ottenere il volume finale.

É opportuno osservare che il problema di costruire un volume tridimensionale a partire da un insieme di immagini è generalmente malposto e quindi la tecnica della Visual Hull non è in grado di ricostruire perfettamente tutti gli oggetti, anche nell'ipotesi di un numero infinito di telecamere[41]. In particolare le regioni dell'oggetto che presentano concavità possono essere ricostruite erroneamente, perché la tecnica è in grado al più di recuperare l'inviluppo convesso (*convex hull*) degli oggetti.

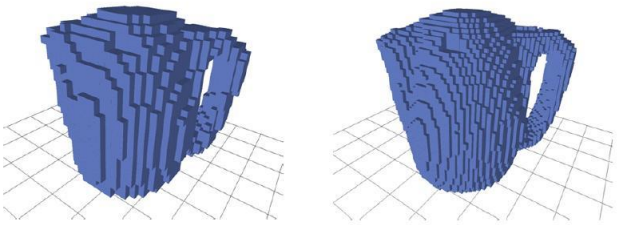


(a) Oggetto

(b) Silhouette



(c) Coni generalizzati



(d) Voxelset finale

Figura 3.3: **Voxel Carving.** *Sono illustrati i vari necessari a generare un voxelset con l'algoritmo del Voxel Carving. Un oggetto viene ripreso da diversi punti di vista* **(a)** *e con un algoritmo di segmentazione si ottiene la sua sagoma nelle varie immagini* **(b)**. *L'intersezione dei coni ottenuti riproiettando le sagome nello spazio dell'oggetto* **(c)** *è infine approssimata con un insieme di voxel, la cui dimensione determina l'accuratezza nella definizione del volume* **(d)**.

Anche se esistono metodi per recuperate il volume di intersezione esatto a partire dall'informazione presente all'interno di ciascuna silhouette[24], nella pratica la Visual Hull è approssimata da un insieme di voxel (l'estensione tridimensionale dei pixel) campionati su un reticolo cubico con passo uniforme. Il *voxelset* così ottenuto è un'approssimazione del reale volume della Visual Hull. Il voxelset è costruito sfruttando una versione semplificata della tecnica del *Voxel Carving*[40], nella quale ogni voxel è proiettato nel piano immagine di ogni telecamera, ed è assegnato al volume se per ogni vista la proiezione del voxel sul piano immagine della telecamera ricade nella silhouette. L'algoritmo per calcolare la Visual Hull (VH) è descritto sinteticamente dallo pseudocodice nel riquadro 1.

Algoritmo 1 Voxel Carving

```
dividi lo spazio in N1 × N2 × N3 voxel v ∈ V
for all v ∈ V do
  value(v) ← 1 (occupato)
end for
for all v ∈ V do
  for n = 1 to N_cam do
    Proietta il voxel v sul piano immagine della n-esima telecamera
    if area proiettata esterna alla silhouette then
      value(v) ← 0 (vuoto)
    end if
  end for
end for
VH = {v ∈ V t.c. value(v) = 1}
```

In condizioni ideali, questo algoritmo trova tutti i voxel interni alla Visual Hull; tuttavia, a causa di errori in fase di segmentazione, alcuni voxel possono risultare esterni alla silhouette di una telecamera, e quindi non sono accettati. Per evitare questo problema, un voxel è considerato interno anche se non passa il test in una singola vista. La probabilità invece che voxel non appartenenti alla Visual Hull siano accettati è molto bassa, perché richiede errori di segmentazione simultanei su tutte le telecamere.

Capitolo 4

Cattura del movimento umano

Posso misurare il moto dei corpi, non l'umana follia.

Isaac Newton

Nell'algoritmo di inseguimento sviluppato per questa tesi abbiamo scelto delle tecniche specifiche per modellare adeguatamente ed efficientemente il corpo umano e la varietà dei movimenti che è in grado di eseguire. In questo capitolo descriveremo brevemente i fondamenti teorici di ogni tecnica, mentre nel prossimo capitolo mostreremo come ciascuno di questi elementi si integra nell'algoritmo finale.

Data l'importanza della rappresentazione delle rotazioni nei modelli tridimensionali, abbiamo deciso di presentare nella prima sezione una rassegna relativamente approfondita, anche se nel seguito saranno impiegate unicamente le rappresentazioni a matrici e a quaternioni. I vincoli sugli angoli sono un altro argomento importante, e nella seconda sezione presentiamo un approccio specifico al problema.

L'algoritmo di inseguimento si basa sull'algoritmo *Iterative Closest Point* (ICP), del quale diamo una breve descrizione nella terza sezione. In un modello articolato ICP deve essere modificato per gestire adeguatamente le relazioni fra le varie parti del corpo; esistono vari approcci possibili al problema, e in questa sezione descriviamo la strategia scelta.

Il metodo dei *particle filter* infine merita particolare attenzione. Nonostante nelle forme più semplici sia di facile implementazione, la teoria che ne motiva l'uso è sofisticata e richiede una solida conoscenza della teoria della probabilità. In questo capitolo quindi diamo una breve descrizione teorica dei particle filter.

4.1 Rappresentazione delle rotazioni

In un modello articolato è necessario rappresentare il sistema di riferimento locale di ciascuna parte, in modo da avere una codifica della posizione e dell'orientamento. Mentre le traslazioni possono essere sempre rappresentate da un vettore a tre componenti, esistono diverse rappresentazioni possibili per le rotazioni.

Ogni rappresentazione ha vantaggi e svantaggi, e la scelta dipende in larga parte dal tipo di operazioni che devono essere svolte. Nelle seguenti sezioni illustreremo le rappresentazioni più note.

4.1.1 Matrici di rotazione

Una rotazione è definita come un'operazione lineare che preserva le lunghezze dei vettori, gli angoli fra due vettori, e non altera l'orientamento degli assi del sistema di riferimento. In termini di applicazioni da $\mathbb{R}^3$ a $\mathbb{R}^3$ una rotazione è quindi una matrice *ortogonale* 3×3 con determinante uguale a 1.

Per ruotare un vettore $\mathbf{v}$ si applica semplicemente la trasformazione:

$$\mathbf{v}' = R\mathbf{v} \tag{4.1}$$

mentre la composizione di rotazioni ($\circ$) si ottiene dalla *pre*moltiplicazione delle rispettive matrici:

$$R \circ S(\mathbf{v}) = SR\mathbf{v} \tag{4.2}$$

Le matrici di rotazione sono il metodo più efficiente per calcolare la rotazione di un vettore, e sono di fatto lo standard nella computer graphics dove, tramite l'impiego di coordinate omogenee, è possibile integrare nella medesima matrice anche le informazioni sulla traslazione del vettore.

La notazione matriciale nasconde il fatto che, pur essendo la matrice definita da nove differenti parametri, i gradi di libertà sono solo tre: infatti le condizioni di ortogonalità e norma unitaria dei vettori colonna, date dalla definizione di matrice ortogonale, eliminano sei gradi di libertà. La relazione fra i gradi di libertà reali e gli elementi della matrice di rotazione non è però immediata. Inoltre, nei casi pratici, errori di calcolo possono portare al non soddisfacimento delle condizioni di ortogonalità, e quindi alla necessità di ortogonalizzare la matrice, in modo che rappresenti ancora una rotazione. Nel caso delle matrici di rotazione questa procedura richiede un certo numero di calcoli.

Questi problemi sono in parte risolti dalle altre rappresentazioni. In particolare il problema della ridondanza dei parametri è risolto dagli angoli di Eulero e dai twist, mentre è presente in misura minore nei quaternioni (quattro parametri invece di tre).

4.1.2 Angoli di Eulero

Gli angoli di Eulero risolvono il problema della mappatura dei gradi di libertà esprimendo la rotazione cercata come la composizione di tre diverse rotazioni intorno ad assi predefiniti. Differenti convenzioni nella scelta degli assi e del verso di rotazione portano a diverse parametrizzazioni possibili delle rotazioni.

Chiamata $R(\hat{\mathbf{n}}, \theta)$ la matrice di rotazione di un angolo θ intorno all'asse $\hat{\mathbf{n}}$, la convenzione indicata con il nome di *angoli propri* di Eulero prevede la seguente decomposizione:

$$R = R(Z, \gamma)R(X, \beta)R(Z, \alpha) \tag{4.3}$$

Si osservi che la prima rotazione di un angolo α cambia l'orientamento degli assi, che quindi nelle successive due rotazioni non corrispondono agli assi X e Z del sistema di riferimento originale. Per questo motivo la convenzione è talvolta indicata con $Z - X' - Z''$. Una notazione alternativa spesso usata in robotica è quella degli *angoli di Tait-Bryan* ($Z - Y' - X''$), nella quale gli angoli di rotazione sono comunemente chiamati *imbardata*, *beccheggio* e *rollio* (*yaw*, *pitch* e *roll* in inglese) e indicati rispettivamente con ψ, θ e ϕ.

Gli angoli di Eulero presentano il problema del *blocco cardanico* (*gimbal lock* in inglese). Nel caso $\beta = 0$ gli assi Z e Z'' coincidono e R può essere rappresentata da una semplice rotazione di $\alpha + \gamma$ intorno a Z. La perdita di un grado di libertà significa che in determinate configurazioni i parametri non riflettono fedelmente i gradi di libertà reali del sistema, e nelle animazioni si manifesta con artefatti nel movimento.

4.1.3 Quaternioni

La rappresentazione mediante quaternioni risolve il problema del blocco cardanico mantenendo nel contempo una parametrizzazione compatta delle rotazioni. I quaternioni sono una generalizzazione dei numeri complessi. Sono definite tre unità immaginarie $\mathbf{i}$, $\mathbf{j}$ e $\mathbf{k}$:

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{4.4}$$

La componente w è spesso chiamata *parte scalare* del quaternione, mentre i restanti tre termini, interpretati come un vettore $\mathbf{v}$ avente versori $\mathbf{i}$, $\mathbf{j}$ e $\mathbf{k}$, è chiamata *parte vettoriale* del quaternione. Le unità immaginarie hanno le seguenti proprietà:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1 \qquad \mathbf{ij} = -\mathbf{ji} = \mathbf{k} \qquad \mathbf{jk} = -\mathbf{kj} = \mathbf{i} \qquad \mathbf{ki} = -\mathbf{ik} = \mathbf{j}$$

che permettono di derivare le espressioni della somma e prodotto di due quaternioni. Per una trattazione approfondita dell'argomento si rimanda agli innumerevoli articoli online e ai vari libri pubblicati sull'argomento[39][30]; in questa sezione riportiamo solo alcuni risultati importanti.

É possibile stabilire una relazione fra quaternioni di norma unitaria e rotazioni nello spazio tridimensionale. Una rotazione intorno all'asse $\mathbf{u}$ di un angolo θ è rappresentata da:

$$q = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2} \tag{4.5}$$

La composizione di rotazioni è data dal prodotto di quaternioni, e l'inverso di una rotazione dal reciproco del rispettivo quaternione, che nel caso di quaternioni unitari equivale al quaternione coniugato:

$$\bar{q} = w - \mathbf{v} = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k} \tag{4.6}$$

L'esponenziale di un quaternione è definito dall'usuale serie di potenze, e il logaritmo come l'inverso dell'esponenziale. Nel caso di quaternioni unitari, espressi nella forma (4.5), il logaritmo è puramente immaginario e ha la forma notevole:

$$\log q = \mathbf{u}\frac{\theta}{2} \tag{4.7}$$

L'esponenziale di un quaternione puramente immaginario è quindi sempre unitario:

$$e^{\mathbf{u}\frac{\theta}{2}} = \cos\frac{\theta}{2} + \mathbf{u}\sin\frac{\theta}{2} \tag{4.8}$$

L'elevamento a potenza è definito come l'esponenziale del logaritmo moltiplicato per l'esponente e corrisponde ad applicare un fattore di scala all'angolo θ:

$$q^\alpha = e^{\alpha \log q} = \cos\frac{\alpha\theta}{2} + \mathbf{u}\sin\frac{\alpha\theta}{2} \tag{4.9}$$

Questo risultato è molto utile quando è necessario interpolare le rotazioni o definire un fattore di scala per i quaternioni, come faremo nel prossimo capitolo.

Un vettore ruotato può essere calcolato direttamente nello spazio dei quaternioni, ma l'operazione richiede due moltiplicazioni fra quaternioni e consuma più calcoli rispetto alle matrici di rotazione. Abbiamo preferito usare quest'ultime nei calcoli delle trasformazioni geometriche, mentre nella rappresentazione delle rotazioni abbiamo deciso di mantenere una doppia codifica con matrici e quaternioni.

4.1.4 Coordinate esponenziali e twist

Benché i quaternioni siano una rappresentazione adeguata per molte applicazioni, la condizione di norma unitaria è difficile da incorporare in problemi di ottimizzazione. La notazione in coordinate esponenziali risolve questo problema mantenendo una notazione compatta e assenza di blocco cardanico, ed è quindi la scelta migliore nei casi in cui è necessario minimizzare una funzione errore sull'intero scheletro (per esempi si vedano Bregler[8] e Mündermann[50]).

Una rotazione descritta da un asse di rotazione $\boldsymbol{\omega} = (\omega_1, \omega_2, \omega_3)$ e da un angolo θ può essere rappresentata in modo compatto dalle *coordinate esponenziali* $\boldsymbol{\omega}\theta$, sfruttando il fatto che il vettore che definisce l'asse è unitario. Definito l'operatore cappuccio:

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{4.10}$$

La matrice di rotazione può essere ottenuta con la formula:

$$R = \exp(\hat{\boldsymbol{\omega}}\theta) = \sum_{k=0}^{\infty} \frac{(\hat{\boldsymbol{\omega}}\theta)^k}{k!} = I + \hat{\boldsymbol{\omega}}\theta + \dots \tag{4.11}$$

che, troncata ai primi due termini, costituisce un'approssimazione lineare della rotazione per piccoli valori di θ. Spesso viene usata la *formula di Rodrigues*, che permette di calcolare esattamente la matrice di rotazione:

$$R = I + \hat{\boldsymbol{\omega}} \sin(\theta) + \hat{\boldsymbol{\omega}}^2 (1 - \cos(\theta)) \tag{4.12}$$

Quando la mappa esponenziale è estesa alle trasformazioni rigide nello spazio si ottiene la notazione dei *twist*, che codifica con sei parametri i

sei gradi di libertà di un corpo rigido. Analogamente alle coordinate esponenziali, detti G la matrice di trasformazione in coordinate omogenee e $\mathbf{v}$ i parametri della traslazione, si ha:

$$G = \exp(\hat{\boldsymbol{\xi}}\theta) \qquad \text{con} \qquad \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} \qquad \text{e} \qquad \hat{\boldsymbol{\xi}} = \begin{bmatrix} \hat{\boldsymbol{\omega}} & \mathbf{v} \\ 0 & 0 \end{bmatrix} \tag{4.13}$$

Le coordinate esponenziali e i twist possiedono una ricca struttura matematica. In particolare $\hat{\boldsymbol{\omega}}$ e $\hat{\boldsymbol{\xi}}$ sono rispettivamente elementi delle algebre di Lie $so(3)$ e $se(3)$ e rappresentano la derivata della trasformazione rispetto al parametro θ.[51]

4.2 Vincoli sulle rotazioni

Nella costruzione di un modello articolato si presenta spesso il problema di limitare il movimento di un elemento rispetto all'altro, sia per vincoli dovuti all'articolazione, sia per escludere configurazioni che portano alla compenetrazione fra due elementi.

Mentre i vincoli sull'ampiezza della traslazione possono essere facilmente codificati attraverso un riquadro di delimitazione (*bounding box*) che rappresenta l'intervallo ammesso per lo spostamento, i vincoli sugli angoli sono più difficili da definire, a causa della complessa relazione fra proprietà intuitive della rotazione e la rappresentazione scelta. Ad esempio, data una notazione asse-angolo, è difficile stabilire quanto sarà l'ampiezza di movimento di un particolare vettore. A questa difficoltà si aggiunge il fatto che, in modelli realistici, i vincoli relativi a diverse articolazioni non sono indipendenti, e quindi l'insieme delle posizioni possibili può essere un sottoinsieme complicato nello spazio dei parametri delle rotazioni. Questo insieme ha dimensione $3n$, dove n è il numero di articolazioni del modello. Pur esistendo in letteratura risultati in questa direzione[31], una soluzione dettagliata del problema esula dagli obiettivi di questa tesi.

Esistono diverse strategie per costruire vincoli sulle rotazioni: una rassegna approfondita è disponibile nella tesi di David Kelly (2008) [36]. Per non complicare eccessivamente il sistema di riconoscimento abbiamo scelto un metodo relativamente semplice, ispirato in larga parte da un articolo di Jonathan Blow[5]. I vincoli sono applicati nello spazio dei quaternioni.

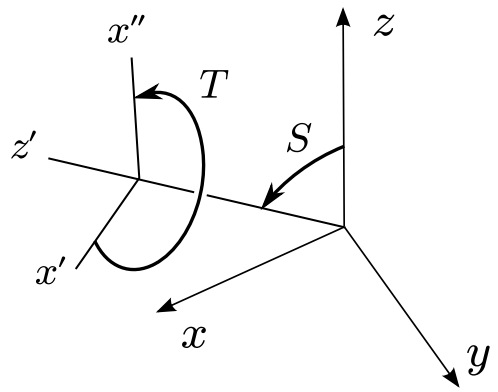


Figura 4.1: **Oscillazione e torsione.** *Decomposizione di una rotazione nelle componenti di oscillazione S e di torsione T. Nell'esempio in figura, S è una rotazione attorno all'asse y. L'oscillazione trasforma l'asse z nell'asse z' e l'asse x in x'. La torsione mantiene il nuovo asse z inalterato e trasforma x' in x''.*

4.2.1 Decomposizione della rotazione

Abbiamo diviso per comodità la rotazione in due componenti (vedi Figura 4.1). La componente di **oscillazione**, o *swing*, descrive di quanto è stata modificata la direzione di un vettore dato. La componente di **torsione**, o *twist*, è una rotazione attorno alla direzione finale del vettore, e quindi non ne altera l'orientamento. Questa rappresentazione è molto utile per descrivere articolazioni come la spalla, dove si vogliono gestire separatamente i movimenti in ampiezza e la rotazione sul proprio asse del braccio.

Indicata con S l'oscillazione e T la torsione, una rotazione R può essere decomposta secondo la formula $R = TS$. Se $\mathbf{w} = R\mathbf{v}$, si ha $S\mathbf{v} = \mathbf{w}$ e $T\mathbf{w} = \mathbf{w}$, vale a dire che solo S influenza la direzione finale del vettore. É sempre possibile trovare il quaternione che descrive l'oscillazione, data la direzione iniziale $\mathbf{v}$ e la direzione finale $\mathbf{w}$, calcolando la radice quadrata del *prodotto di Clifford* fra i due vettori $\mathbf{v}$ e $\mathbf{w}$[21]. Il prodotto di Clifford è calcolabile con la formula:

$$w_s = \mathbf{v} \cdot \mathbf{w} \qquad \mathbf{s} = \mathbf{v} \times \mathbf{w} \tag{4.14}$$

dove w_s e $\mathbf{s}$ rappresentano rispettivamente la parte scalare e vettoriale del quaternione s, mentre per la radice quadrata di un quaternione esiste una formula efficiente che richiede solo interpolazione lineare e normalizzazione[5]. La torsione è ricavata facilmente da $t = rs^{-1}$.

La torsione e l'oscillazione sono vincolate separatamente. La torsione è una rotazione intorno alla direzione finale del vettore trasformato e quindi

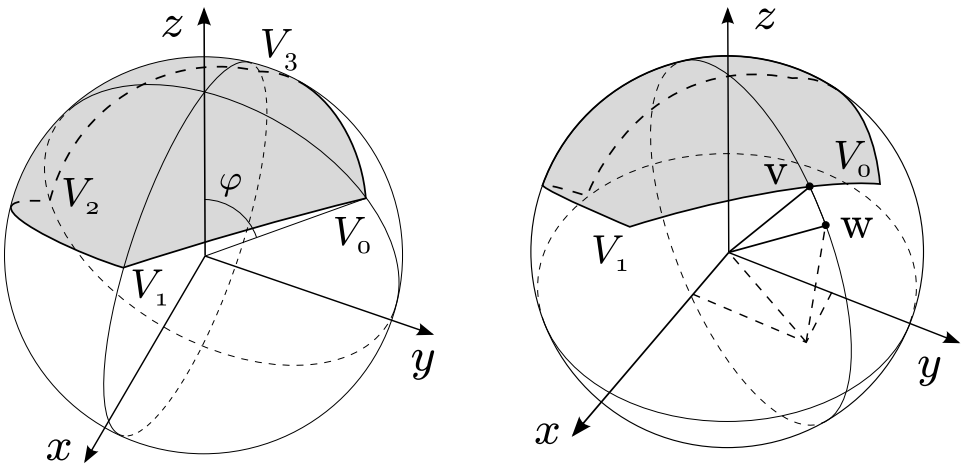


Figura 4.2: **Vincolo sull'oscillazione.** *A sinistra, regione di vincolo dell'oscillazione, individuata dai punti V_0, V_1, V_2, V_3; in figura è mostrato anche l'angolo ϕ relativo a V_0 e i due cerchi massimi nei piani xz e yz su cui giacciono i vertici. A destra, illustrazione geometrica del calcolo del valore vincolato* $\mathbf{v}$ *per l'oscillazione* $\mathbf{w}$*; in figura sono riportati i vertici V_0 e V_1 dell'arco interessato e il cerchio massimo sul quale in vincolo è calcolato.*

può essere descritta da un singolo angolo θ. Il vincolo che abbiamo scelto è un semplice intervallo di valori contenente l'origine, e gli angoli che eccedono il valore massimo o minimo ammessi assumono il valore massimo o minimo, rispettivamente. L'angolo della rotazione è facilmente estraibile dalla rappresentazione in quaternioni, e la nuova rotazione può essere calcolata agevolmente a partire dall'angolo modificato. Per l'oscillazione il procedimento è invece più complesso.

4.2.2 Vincolo sull'oscillazione

Fissato un vettore unitario $\mathbf{v}$, l'oscillazione è descritta univocamente dalla direzione finale del vettore trasformato $\mathbf{w}$, che corrisponde a un punto sulla superficie della sfera di raggio unitario centrata nell'origine. Il vincolo sull'oscillazione è quindi ridotto alla definizione di una regione sulla superficie di una sfera. Abbiamo scelto come $\mathbf{v}$ l'asse z, e per semplicità abbiamo considerato solo le regioni che contengono il versore z, e che quindi ammettono come valida un'oscillazione nulla.

La regione del vincolo è quadrilatero curvilineo definito dai quattro punti (vedi Figura 4.2) che corrispondono ai limiti di una pura rotazione attorno all'asse x (V_0, V_2) o di una pura rotazione attorno all'asse y (V_1, V_3). I punti sono individuati univocamente dagli angoli formati dal raggio individuato

dai punti con l'asse z nel piano yz (V_0, V_2) e xz (V_1, V_3), e gli angoli così definiti rappresentano i limiti angolari per le rotazioni attorno all'asse x e all'asse y rispettivamente. Sono quindi sufficienti due intervalli di valori per gli angoli, uno per l'asse x e uno per l'asse y, per definire completamente il vincolo.

Quando l'oscillazione non giace sul piano yz o xz il bordo della regione del vincolo è calcolato effettuando l'interpolazione sferica (*Slerp*) fra i due vertici del quadrante corrispondente. Ad esempio, se $\mathbf{w}$ ricade nel terzo quadrante, i vertici saranno V_2 e V_3. Indicate con i vettori $\mathbf{v}_0$ e $\mathbf{v}_1$ le coordinate dei vertici da interpolare, se i vettori sono perpendicolari e unitari l'interpolazione sferica si riduce semplicemente a:

$$\mathbf{v} = \mathbf{v}_0 \cos(\theta t) + \mathbf{v}_1 \sin(\theta t) \qquad t \in [0, 1] \qquad \text{e} \qquad \theta = \frac{\pi}{2} \tag{4.15}$$

Nel caso la coppia non sia ortogonale, si fissa $\theta = \arccos(\mathbf{v}_0 \cdot \mathbf{v}_1)$ e a $\mathbf{v}_1$ si sostituisce la componente ortogonale a $\mathbf{v}_0$ normalizzata, che indicheremo con $\mathbf{v}_2$:

$$\mathbf{v} = \mathbf{v}_0 \cos(\theta t) + \mathbf{v}_2 \sin(\theta t) \qquad t \in [0, 1] \tag{4.16}$$

Il problema è ora ridotto alla verifica dell'appartenenza di $\mathbf{w}$ alla regione del vincolo. La regione è centrata sull'asse z, ed è quindi possibile costruire un test di appartenenza usando solo una soglia variabile per la coordinata z. La soglia corrisponde al bordo della regione e dipende quindi dalla direzione di $\mathbf{w}$ sul piano xy; in particolare, dipende dal quadrante in cui ricade, che determina i due estremi V_i e V_j del lato corrispondente. Tutti i vettori $\mathbf{w}$ che, proiettati sul piano xy, hanno la stessa direzione e verso, hanno la stessa soglia. L'insieme dei vettori che hanno la stessa soglia individua un semicerchio sulla sfera, e l'intersezione fra questo semicerchio e il bordo della regione individua il punto di soglia $\mathbf{v}$ (vedi Figura 4.2). $\mathbf{w}$ appartiene alla regione se la sua coordinata z è maggiore della coordinata z del punto di soglia.

Una volta noto il quadrante, è possibile trovare il punto di soglia scegliendo, fra tutti i punti dell'interpolazione sferica, quello con proiezione sul piano xy parallela alla proiezione di $\mathbf{w}$. In formule:

$$\begin{aligned} v_{0x}\alpha \cos(\theta t) + v_{2x}\alpha \sin(\theta t) &= w_x \\ v_{0y}\alpha \cos(\theta t) + v_{2y}\alpha \sin(\theta t) &= w_y \end{aligned} \tag{4.17}$$

dove la costante di proporzionalità incognita α esprime la condizione di parallelismo. Il sistema lineare (4.17) nelle incognite $\alpha\cos(\theta t)$ e $\alpha\sin(\theta t)$, unito all'identità trigonometrica fondamentale, permette di recuperare i parametri di l'interpolazione dell'equazione (4.16). Per la soglia serve solo la coordinata z, e il test si riduce finalmente a:

$$w_z > v_z = v_{0z}\cos(\theta t) + v_{2z}\sin(\theta t) \tag{4.18}$$

I vettori ortogonalizzati possono essere precalcolati per ridurre il carico computazionale. Inoltre, poiché i parametri $\cos(\theta t)$ e $\sin(\theta t)$ dell'interpolazione sferica sono noti a meno di una costante di proporzionalità, non è necessario calcolare il determinante del sistema (4.17). Per limitare ulteriormente il numero di calcoli effettuiamo un test preliminare su una soglia pari alla massima coordinata z dei vertici V_0, V_1, V_2 e $V3$, che corrisponde a testare l'appartenenza a un cerchio curvilineo inscritto nella regione del vincolo[5]. Solo se questo test fallisce eseguiamo il più oneroso calcolo esatto.

Se $\mathbf{w}$ non rispetta il vincolo, la sua coordinata z assume il valore della soglia, e le coordinate x e y sono scalate per mantenere la norma unitaria del vettore. Alla fine di tutto il processo il prodotto di Clifford (4.14) permette di recuperare l'oscillazione espressa in quaternioni.

4.3 Iterative Closest Point

L'algoritmo *Iterative Closest Point*, proposto per la prima volta da Besl e McKay[4], è stato sviluppato per risolvere il problema di allineare due nuvole di punti attraverso una trasformazione rigida. La trasformazione è ottenuta cercando iterativamente le coppie di punti più vicini fra loro e calcolando la trasformazione geometrica che minimizza una funzione errore sulle coppie.
In questa sezione offriamo una panoramica sulle basi teoriche dell'algoritmo. La trattazione che segue è basata essenzialmente sull'approccio originale di Besl e McKay[4].

Dati due insiemi di N punti $X = \{\mathbf{x}_i\}$ e $P = \{\mathbf{p}_i\}$, chiamati rispettivamente *insieme modello* e *insieme dei dati*, per ogni punto modello $\mathbf{x}_i$ si cerca il punto a distanza minima $\mathbf{p}_j$, e si riordinano di conseguenza i punti P. In generale questa operazione richiede $O(N^2)$ operazioni, che possono essere ridotte a $O(N\log(N))$ con l'aiuto di strutture gerarchiche come gli alberi *k-d*[66]. Alternativamente, i punti del modello possono essere calcolati dai punti dei dati usando un modello geometrico in $O(n)$ operazioni, come è stato fatto in questa tesi.

A partire dalle coppie di punti si definisce una misura di errore della trasformazione rigida data da una rotazione R e da una traslazione $\mathbf{t}$:

$$E(R, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} \|R(\mathbf{x}_i) + \mathbf{t} - \mathbf{p}_i\|^2 \tag{4.19}$$

e lo scopo è trovare la trasformazione $(R, \mathbf{t})$ che minimizza questo errore. Nota la rotazione ottima R_q, si dimostra[33] che la traslazione ottima è data da:

$$\mathbf{t} = \boldsymbol{\mu}_p - R_q(\boldsymbol{\mu}_x) \tag{4.20}$$

Il calcolo della rotazione ottima richiede un procedimento più complesso. Indicati con $\boldsymbol{\mu}_x$ e $\boldsymbol{\mu}_p$ i centri di massa dei due insiemi di punti:

$$\boldsymbol{\mu}_x = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i \qquad \text{e} \qquad \boldsymbol{\mu}_p = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i \tag{4.21}$$

si costruisce la matrice di covarianza Σ_{xp}:

$$\Sigma_{xp} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu}_x)(\mathbf{p}_i - \boldsymbol{\mu}_p)^T = \frac{1}{N} \sum_{i=1}^{N} [\mathbf{x}_i \mathbf{p}_i^T] - \boldsymbol{\mu}_x \boldsymbol{\mu}_p^T \tag{4.22}$$

Costruita la matrice $A = \Sigma_{xp} - \Sigma_{xp}^T$ e il vettore colonna $\Delta = [A_{23}, A_{31}, A_{12}]^T$ è possibile finalmente definire la matrice Q:

$$Q(\Sigma_{xp}) = \begin{bmatrix} \mathrm{tr}(\Sigma_{xp}) & \Delta^T \\ \Delta & \Sigma_{xp} + \Sigma_{xp}^T - \mathrm{tr}(\Sigma_{xp})\mathbf{I_3} \end{bmatrix} \tag{4.23}$$

dove $\mathbf{I_3}$ rappresenta la matrice identità di dimensioni 3×3. Si dimostra[33] che l'autovettore a norma unitaria $\mathbf{q}$ di Q corrispondente al massimo autovalore, interpretato come quaternione q, è la rotazione che minimizza l'errore $E(R, \mathbf{t})$.

La trasformazione calcolata con questo procedimento non offre generalmente una buona sovrapposizione fra le nuvole di punti, perché le coppie di punti ottenute dipendono dall'orientamento reciproco dei due insiemi e non codificano informazioni di similarità locale fra le superfici. Tuttavia, se l'orientamento iniziale è abbastanza vicino all'orientamento ottimale, è più probabile che le corrispondenze fra punti calcolate in base alla minima distanza corrispondano alle coppie di punti vicini calcolate nel caso di sovrapposizione ottimale e che la trasformazione $(R, \mathbf{t})$ trovata possa portare a un'approssimazione migliore. Queste considerazioni portano a formulare il seguente algoritmo iterativo:

1. Costruisci le coppie $(\mathbf{x}_i, \mathbf{p}_i)$ a minima distanza all'iterazione k
2. Calcola la trasformazione $(R, \mathbf{t})$ che minimizza l'errore di sovrapposizione E_k
3. Applica la trasformazione ai punti $\{\mathbf{x}_i\}$
4. se $|E_k - E_{k-1}| \leq \varepsilon$ oppure massimo numero di iterazioni stop; altrimenti, torna a 1.

Come criterio di convergenza è generalmente scelta la variazione dell'errore invece del valore assoluto dell'errore per non introdurre nell'algoritmo una dipendenza dalla qualità di approssimazione del modello, che vincola l'errore minimo raggiungibile. Besl e McKay hanno dimostrato[4] la convergenza dell'algoritmo a un minimo locale dell'errore di sovrapposizione E, interpretato come funzione di R, $\mathbf{t}$ e della particolare scelta di coppie $(\mathbf{x}_i, \mathbf{p}_i)$. Nei casi pratici questa funzione presenta vari minimi locali e l'algoritmo converge alla soluzione "esatta" solo se l'orientamento iniziale è sufficientemente vicino al minimo globale: è quindi di vitale importanza disporre di una buona stima iniziale.

Esistono diverse varianti dell'algoritmo originale: la scelta delle coppie può essere basata su un criterio diverso dalla distanza euclidea, l'errore può integrare informazioni sulle normali alla superficie e così via [59]. Di particolare rilevanza per questa tesi sono le varianti che permettono di applicare l'algoritmo a modelli articolati.

4.3.1 Estensione ai modelli articolati

In letteratura sono stati proposti diversi metodi per integrare i vincoli di un modello articolato nell'algoritmo Iterative Closest Point. Citiamo in particolare gli approcci di Demirdjian *et al.*[18] e di Knoop *et al.*[37], sviluppati con particolare enfasi sulla figura umana. Per la sua semplicità e facilità di implementazione abbiamo scelto il metodo di Knoop, indicato con il nome di *corrispondenze artificiali*. Un metodo sostanzialmente equivalente è stato presentato da Mündermann *et al.*[50].

Date due parti contigue M_j e M_k di un modello articolato, il metodo delle corrispondenze artificiali consiste nel creare, in corrispondenza dell'articolazione fra le due parti, alcune coppie di punti modello e di punti dati

fittizie. Questi punti sono usati per generare delle corrispondenze artificiali dati–modello, da aggiungere rispettivamente agli insiemi P e X nell'algoritmo ICP. Per ogni coppia di punti, un punto è associato a M_j e l'altro a M_k, ciascuno con posizione fissa rispetto alla parte alla quale è associato. Quando si calcola l'orientamento di M_j, per ogni coppia di punti quelli di M_j sono aggiunti all'insieme modello, mentre i corrispondenti su M_k ai punti dati. Quando si calcola l'orientamento di M_k, i suoi punti sono aggiunti al modello, e i punti di M_j ai dati.

Con le corrispondenze artificiali ICP deve minimizzare non solo la distanza fra la superficie del modello e i punti dei dati, ma anche la distanza fra i punti appartenenti a parti diverse. L'effetto finale è una penalizzazione dell'allontanamento di una parte rispetto all'altra e, a seconda del numero e della posizione dei punti, una limitazione dei movimenti angolari. Intuitivamente l'effetto è analogo alla presenza, fra le varie parti del modello, di bande elastiche che permettono una libertà di movimento tanto limitata quanto maggiore è la discrepanza dalla posizione di riposo.

Quando un passo dell'algoritmo ICP è eseguito sulla parte M_i, le coppie relative ad ogni articolazione di M_i sono aggiunte agli insiemi P e X. Poiché il numero di punti delle corrispondenze è spesso molto minore del numero di punti originari di ICP, per ogni coppia di punti sono aggiunte copie multiple agli insiemi, in modo da rendere significativo il loro contributo. Il numero di copie è scelto in modo che la proporzione dei punti derivati dalle corrispondenze sul totale abbia un valore r compreso fra 0.3 e 0.7. Il valore r influenza l'effetto delle corrispondenze sulla posa calcolata da ICP: tanto più alto è il suo valore, tanto più i vincoli influenzano la posa e tanto più quindi l'articolazione è rigida.

Per rendere più efficiente l'algoritmo le copie di ogni corrispondenza non sono aggiunte direttamente agli insiemi X e P originali, ma baricentri e covarianza finali sono calcolati a partire dai baricentri e dalla covarianza degli insiemi X e P originali, che indicheremo con X_1 e P_1 rispettivamente, e degli insiemi X e P delle corrispondenze, che indicheremo con X_2, P_2. I valori finali sono calcolati supponendo che negli insiemi X e P finali la proporzione dei punti in X_2 e in P_2 sul totale sia r, simulando quindi l'aggiunta di copie multiple.

Essendo X l'unione disgiunta degli insiemi X_1 e X_2, la formula del baricentro di X può essere espressa in funzione dei baricentri di X_1 e X_2. Infatti, chiamati N_1 e N_2 rispettivamente il numero di punti nei due insiemi e supponendo che i primi N_1 punti appartengano a X_1, il baricentro di X è esprimibile nel seguente modo:

$$\boldsymbol{\mu}_x = \frac{\sum_{i=1}^{N_1} \mathbf{x}_i + \sum_{i=N_1+1}^{N_1+N_2} \mathbf{x}_i}{N_1 + N_2} = \frac{N_1}{N_1 + N_2}\boldsymbol{\mu}_{x1} + \frac{N_2}{N_1 + N_2}\boldsymbol{\mu}_{x2} \tag{4.24}$$

e scegliendo come r la quantità $\frac{N_2}{N_1+N_2}$, si ottiene la formula finale:

$$\boldsymbol{\mu}_x = (1 - r)\,\boldsymbol{\mu}_{x1} + r\boldsymbol{\mu}_{x2} \tag{4.25}$$

Le stesse considerazioni valgono nel caso dell'insieme P. Per calcolare la covarianza è utile definire prima la covarianza non centrata S_{xp}, data dalla formula:

$$S_{xp} = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{p}_i^T \tag{4.26}$$

La covarianza Σ_{xp} si ottiene semplicemente sottraendo il prodotto tensoriale dei baricentri:

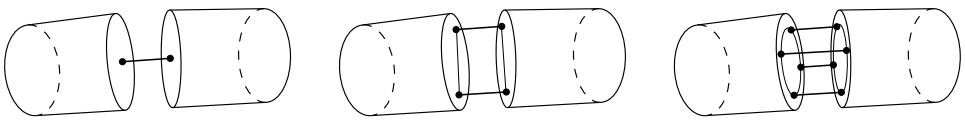
$$\Sigma_{xp} = S_{xp} - \boldsymbol{\mu}_x \boldsymbol{\mu}_p^T \tag{4.27}$$

S_{xp} può essere facilmente spezzata in due termini seguendo lo stesso procedimento di (4.24), e la covarianza Σ_{xp} è recuperabile con la formula (4.27), noti i baricentri dalla formula (4.25). I baricentri e la covarianza finali sono espressi quindi dalle seguenti formule:

$$\begin{aligned} \boldsymbol{\mu}_x &= (1 - r)\,\boldsymbol{\mu}_{x1} + r\boldsymbol{\mu}_{x2} \\ \boldsymbol{\mu}_p &= (1 - r)\,\boldsymbol{\mu}_{p1} + r\boldsymbol{\mu}_{p2} \\ \Sigma_{xp} &= (1 - r)\,S_{xp1} + rS_{xp2} - \boldsymbol{\mu}_x \boldsymbol{\mu}_p^T \end{aligned} \tag{4.28}$$

La scelta del numero e della posizione delle corrispondenze permette di influenzare la libertà di movimento del giunto nelle varie direzioni. Poiché l'algoritmo ICP cerca di ridurre la distanza fra i punti fittizi, tanto più una coppia di corrispondenze sono distanti, tanto più la rotazione lungo la loro congiungente sarà limitata. Benché i punti delle corrispondenze possano essere scelti arbitrariamente, abbiamo limitato le possibilità a tre differenti classi di giunti (vedi Figura 4.3), che corrispondono a tre diversi tipi di articolazione del corpo umano. La classificazione segue lo schema di Knoop *et al.*.

Universale: solo una corrispondenza artificiale è definita per l'articolazione. La libertà di movimento non è vincolata significativamente in nessuna direzione, e quindi questo tipo di giunto è utile a modellare articolazioni come la spalla o l'anca.



(a) universale (b) a cerniera (c) ellittica

Figura 4.3: **Tipi di articolazioni.** *Tipi di articolazioni modellate dal metodo di Knoop* et al.*. I tipi possibili sono* universale **(a)**, a cerniera **(b)** *ed* ellittica **(c)**.

A cerniera: sono definite due corrispondenze artificiali poste agli estremi di un segmento. Il giunto può ruotare solo sul segmento contenente le corrispondenze, e quindi questa classe è utile a modellare articolazioni a un grado di libertà, come i gomiti e le ginocchia.

Ellittico: sono definite quattro corrispondenze poste agli estremi degli assi di un ellisse, ottenendo un giunto che può muoversi con tre gradi di libertà, ma con un'escursione limitata. Questa classe è quindi adatta ad articolazioni come collo e polso.

4.4 Induzione bayesiana e particle filter

I valori di giunto ottenuti dall'adattamento del modello al volume acquisito non necessariamente corrispondono ai valori della posa reale, a causa di errori causati da artefatti della Visual Hull, rumore di quantizzazione dei voxel o convergenza dell'algoritmo di inseguimento a un minimo locale. I valori misurati quindi possono essere interpretati come una versione "corrotta" di valori reali che si vogliono recuperare con un'operazione di filtraggio. In questa sezione si analizzeranno in termini molto generali le tecniche sviluppate per risolvere questo problema e le motivazioni teoriche della soluzione scelta.

4.4.1 Filtraggio stocastico

Il problema del filtraggio stocastico nel caso di tempo discreto e sistemi privi di ingresso può essere formalizzata dalle seguenti equazioni:

$$\begin{aligned} x_{n+1} &= f(x_n, d_n) \\ y_n &= g(x_n, v_n) \end{aligned} \tag{4.29}$$

dove i termini d_n e v_n sono sequenze casuali che rappresentano rumore aggiunto al sistema e hanno una distribuzione generalmente non nota. La

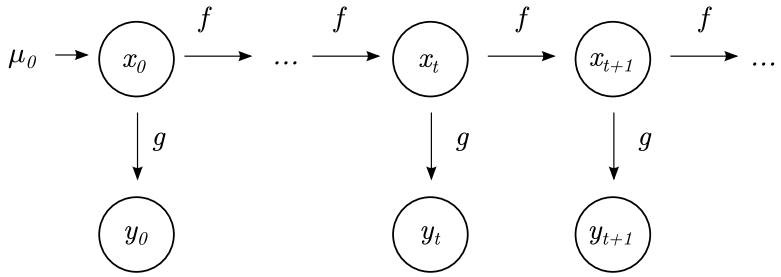


Figura 4.4: **Sistema dinamico stocastico.** *Schema del sistema dinamico descritto dalle equazioni (4.29). Lo stato iniziale x_0 è campionato da una distribuzione iniziale μ_0. Gli stati successivi x_n e le osservazioni y_n sono generati rispettivamente dalla probabilità di transizione di stato $f(x_{n+1}|x_n)$ e dalla probabilità di osservazione $g(y_n|x_n)$.*

prima equazione è chiamata *equazione di stato* e caratterizza la probabilità di transizione di stato $p(x_{n+1}|x_n)$ del sistema, mentre la seconda è chiamata *equazione della misurazione* e definisce la probabilità di osservazione $p(y_n|x_n)$[13]. Una rappresentazione grafica del sistema è riportata in Figura 4.4.

Spesso è necessario calcolare una stima dello stato x_n partendo dalla distribuzione $p(x_n|y_n)$ e da una osservazione y_n. Una delle scelte più frequenti è lo stimatore del *Minimo errore quadratico medio* (*Minimum Mean Square Error* o MMSE in inglese), definito come:

$$\min_{\hat{x}_k} E[\|x_k - \hat{x}_k\|^2 | y_k] \tag{4.30}$$

dove $\hat{x}_k$ rappresenta la stima cercata. L'espressione è soddisfatta dalla media condizionata di x_k, dato y_k:

$$\hat{x}_k = \mu_{y_k} = E[x_k|y_k] = \int x_k p(x_k|y_k) dx_k \tag{4.31}$$

4.4.2 Filtro bayesiano ricorsivo

Prima di calcolare una stima, è necessario avere un'espressione per $p(x_n|y_n)$. In un sistema dinamico lo stato x_n e l'osservazione y_n all'istante n dipendono in generale da tutta la sequenza precedente di stati $x_{0:n} := \{x_0, x_1, \cdots, x_n\}$ e di osservazioni $y_{0:n} := \{y_0, y_1, \cdots, y_n\}$, che quindi saranno gli eventi presi in considerazione. La formula delle probabilità condizionate di Bayes, che in questo contesto viene spesso chiamata *formula di aggiornamento bayesiano*, permette di derivare la seguente relazione:

$$p(x_{0:n}|y_{0:n}) = \frac{p(x_{0:n}, y_{0:n})}{p(y_{0:n})} = \frac{p(x_{0:n})p(y_{0:n}|x_{0:n})}{p(y_{0:n})} \tag{4.32}$$

Il termine $p(x_{0:n}|y_{0:n})$ è chiamato **probabilità a posteriori** (*a posteriori density*), $p(x_{0:n})$ è la **probabilità a priori** (*prior*), $p(y_{0:n}|x_{0:n})$ è la **funzione di verosimiglianza** (*likelihood*) e $p(y_{0:n})$ è l'**evidenza** (*evidence*).

In letteratura spesso è riportata una differente formulazione, che richiede due ulteriori ipotesi per essere valida. Lo stato all'istante n deve dipendere solo dallo stato all'istante precedente, vale a dire che deve essere valida l'*ipotesi di Markov* al primo ordine: $p(x_n|x_{0:n-1}) = p(x_n|x_{n-1})$. L'osservazione all'istante n deve dipendere solo dallo stato in quell'istante: $p(y_n|x_{0:n}) = p(y_n|x_n)$. Con quest'ultimo vincolo è possibile riscrivere la probabilità congiunta in funzione dello stato e osservazione correnti:

$$p(x_{0:n}, y_{0:n}) = p(y_n|y_{0:n-1}, x_{0:n})p(x_{0:n}|y_{0:n-1})p(y_{0:n-1}) \tag{4.33}$$

Poiché y_n dipende solo dallo stato x_n, il primo termine del secondo membro è $p(y_n|x_n)$. La formula (4.32) può essere riscritta come:

$$p(x_{0:n}|y_{0:n}) = \frac{p(y_n|x_n)p(x_{0:n}|y_{0:n-1})}{p(y_n|y_{0:n-1})} \tag{4.34}$$

dove l'ultimo termine di (4.33) è stato incorporato nella probabilità condizionata al denominatore. Marginalizzando l'equazione rispetto a $x_{0:n-1}$ è possibile finalmente ottenere una versione ricorsiva della formula (4.32):

$$p(x_n|y_{0:n}) = \frac{p(y_n|x_n)p(x_n|y_{0:n-1})}{p(y_n|y_{0:n-1})} \tag{4.35}$$

Questa equazione rappresenta il *passo di aggiornamento* di un filtro bayesiano ricorsivo. Dei termini nella formula, solo la funzione di verosimiglianza può essere calcolata agevolmente a partire dal sistema in (4.29), perché coincide con la funzione g. La probabilità a priori è calcolata nel *passo di predizione*:

$$p(x_n|y_{0:n-1}) = \int p(x_n|x_{n-1})p(x_{n-1}|y_{0:n-1})dx_{n-1} \tag{4.36}$$

dove la probabilità di transizione $p(x_n|x_{n-1})$ è derivata dalla f in (4.29) e la probabilità a posteriori è disponibile dal passo precedente. É possibile calcolare l'evidenza marginalizzando la funzione di verosimiglianza sullo stato x_n, ma generalmente non è necessario, perché una volta noto il numeratore

nelle formule (4.35) e (4.32) si può calcolare il denominatore come fattore di normalizzazione, imponendo che le probabilità a posteriori abbiano somma unitaria.

Il problema è ora usare (4.32) per stimare la distribuzione di probabilità a posteriori, che in genere ha una forma molto complessa.

4.4.3 Particle filter

I particle filter approssimano la distribuzione a posteriori dello stato $x_{0:n}$ con un certo numero di campioni, chiamati particelle, opportunamente pesati; tuttavia, prima di addentrarci nei dettagli teorici, dobbiamo introdurre alcune nozioni sui metodi di Monte Carlo.

Metodi di Monte Carlo

La famiglia dei *metodi di Monte Carlo*, alla quale i particle filter appartengono, comprende le strategie di risoluzione di un problema mediante l'estrazione ripetuta di campioni casuali da una distribuzione, con l'idea che un numero sufficiente grande di campioni sia in grado di "catturare la complessità" degli oggetti che si vogliono manipolare.

In questa sezione descriviamo brevemente l'idea che caratterizza i metodi di Monte Carlo con un esempio sull'integrazione di una funzione. Nelle successive sezioni mostreremo come questi metodi possono essere efficacemente applicati nell'approssimazione di una distribuzione di probabilità.

Nel caso dell'integrazione di Monte Carlo, l'obiettivo è calcolare un integrale usando campioni estratti da una variabile casuale X di distribuzione p. Data una funzione $f : \mathbb{R}^n \mapsto \mathbb{R}$ e una densità di probabilità nota a valori in $\mathbb{R}^n$ non nulla nel supporto di f, si definisce $g := \frac{f}{p}$. Calcolare l'integrale di f equivale a calcolare la media di g rispetto a p:

$$E_p[g] = \int g(x)p(x)dx = \int f(x)dx \tag{4.37}$$

Dati N campioni estratti da p: $x^i \sim p(x)$ è possibile approssimare la reale distribuzione con la formula:

$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N} x^i \tag{4.38}$$

e la media può essere quindi approssimata dalla media campionaria di $g(x)$:

$$E_p \approx \hat{g}_N = \frac{1}{N}\sum_{i=1}^{N} g(x^i) \tag{4.39}$$

Il teorema del limite centrale assicura per $\hat{g}_N$ la convergenza al valore esatto con un errore che decade con ordine $O(N^{-\frac{1}{2}})$, rispetto all'ordine $O(N^{-\frac{1}{n}})$ dei metodi basati su n-intervalli in $\mathbb{R}^n$. I metodi di Monte Carlo sono quindi molto utili quando si manipolano funzioni in molte variabili.

Solitamente non è facile estrarre campioni da $p(x)$, e quindi si ricorre alla tecnica dell'*importance sampling*. Definita una distribuzione *proposta* π sufficientemente simile a p, si riscrive il problema come:

$$\int g(x)p(x)dx = \int g(x)\frac{p(x)}{\pi(x)}\pi(x)dx \approx \frac{1}{N}\sum g_i w_i \tag{4.40}$$

dove:

$$w_i = \frac{p(x^i)}{\pi(x^i)} \qquad \text{e} \qquad x^i \sim \pi(x) \tag{4.41}$$

e quindi il problema si riduce all'estrazione dei campioni dalla funzione proposta e al calcolo dei pesi w_i.

Il metodo dell'importance sampling non è limitato alla risoluzione degli integrali e può essere usato per approssimare una distribuzione di probabilità mediante le coppie di campioni e pesi $\left(x^i, w_i\right)$. Questo metodo, applicato alla stima della distribuzione degli stati in una sequenza, permette di definire i metodi di Monte Carlo sequenziali.

Metodi di Monte Carlo sequenziali

In un sistema dinamico stocastico l'obiettivo è spesso ottenere una stima dello stato a partire dalla distribuzione a posteriori $p(x_{0:n}|y_{0:n})$. La distribuzione nei metodi di Monte Carlo sequenziali è approssimata da un certo numero di campioni estratti da una distribuzione proposta $\pi(x_{0:n}|y_{0:n})$:

$$p(x_{0:n}|y_{0:n}) \approx \sum_{i=1}^{N} \tilde{\omega}_n^i \delta(x_{0:n} - x_{0:n}^i) \tag{4.42}$$

dove δ è la delta di Dirac. La distribuzione è approssimata da N masse (particelle) discrete, pesate in base all'importanza con la quale contribuiscono all'approssimazione. I pesi sono dati dalla formula per l'importance sampling presentata nella precedente sezione:

$$\omega_i^* = \frac{p(x_{0:n}^i|y_{0:n})}{\pi(x_{0:n}^i|y_{0:n})} = \frac{p(y_{0:n}|x_{0:n}^i)p(x_{0:n}^i)}{p(y_{0:n})\pi(x_{0:n}^i|y_{0:n})} \tag{4.43}$$

e in genere il termine di normalizzazione $p(y_{0:n})$ è sostituito da una normalizzazione dei pesi:

$$\omega_i = \frac{p(y_{0:n}|x_{0:n}^i)p(x_{0:n}^i)}{\pi(x_{0:n}^i|y_{0:n})} \qquad \tilde{\omega_i} = \frac{\omega_i}{\sum \omega_i} \tag{4.44}$$

e i pesi così ottenuti sono usati nella formula (4.42).

Una distribuzione proposta definita su tutta la sequenza è difficile da campionare, ed è preferibile calcolare i pesi con una formula ricorsiva. Si sceglie quindi π in modo che soddisfi la proprietà:

$$\pi(x_{0:n}|y_{0:n}) = \pi(x_n|x_{0:n-1}, y_{0:n})\pi(x_{0:n-1}|y_{0:n-1}) \tag{4.45}$$

che unita ai vincoli del filtro bayesiano ricorsivo (ipotesi di Markov e dipendenza delle osservazioni solo dallo stato corrente) permette di derivare una formula di calcolo ricorsivo dei pesi:

$$\omega_n^i = \omega_{n-1}^i \frac{g(y_n|x_n^i)f(x_n^i|x_{n-1}^i)}{\pi(x_n^i|x_{0:n-1}^i, y_{0:n})} \qquad \omega_0^i = \frac{g(y_0|x_0^i)\mu(x_0^i)}{\pi(x_0^i|y_0)} \tag{4.46}$$

dove $\mu(x)$ rappresenta la distribuzione iniziale degli stati. É finalmente possibile descrivere l'algoritmo *Sequential Importance Sampling* (SIS):

1. campiona $x_0^i \sim \pi(x_0|y_0)$, calcola i pesi ω_0^i e normalizza
2. campiona $x_n^i \sim \pi(x_n|x_{0:n-1}^i, y_{0:n})$ e aggiungi alla sequenza generata $x_{0:n}^i := (x_{0:n-1}^i, x_n^i)$
3. calcola ω_n^i con la formula ricorsiva e normalizza
4. incrementa il tempo a $n+1$ e torna a 2.

L'algoritmo tuttavia non è usato nei casi pratici perché la varianza dei pesi rispetto alla distribuzione proposta tende a crescere esponenzialmente con n, rendendo necessario un numero irrealistico di particelle. Questo fenomeno, chiamato *degenerazione*, avviene perché la massa tende a concentrarsi in poche particelle con peso molto alto, mentre le rimanenti particelle danno un contributo trascurabile all'approssimazione della distribuzione. Una misura della degenerazione è data dal *numero efficace* di particelle:

$$N_{eff} = \frac{1}{\sum (\tilde{\omega}_k^i)^2} \tag{4.47}$$

Sequential Importance Resampling

Il problema della degenerazione è risolto ricorrendo a un processo di ricampionamento: ogni particella genera un numero di "figli" proporzionale al peso, e i pesi sono fissati tutti allo stesso valore per conservare la massa. L'algoritmo così ottenuto è chiamato *Sequential Importance Resampling* (SIR):

1. campiona $x_n^i \sim \pi(x_n|x_{0:n-1}^i, y_{0:n})$
2. calcola i pesi ω_n^i
3. ricampiona x_n^i usando i pesi ω_n^i come densità di probabilità
4. resetta i pesi: $w_n^i = \frac{1}{N}$
5. incrementa il tempo a $n+1$ e torna a 2.

La scelta canonica per la distribuzione proposta è la funzione di transizione di stato $f(x_n|x_{n-1})$. Le formule in questo caso diventano:

$$\begin{aligned} &\pi(x_0|y_0) = \mu(x_0) \\ &\pi(x_n|x_{0:n-1}^i, y_{0:n}) = f(x_n|x_{n-1}) \\ &\omega_n^i = \omega_{n-1}^i g(y_n|x_n^i) \end{aligned} \tag{4.48}$$

Questa è la formulazione originale del particle filter, presentata da Gordon *et al.* sotto il nome di *boostrap filter*[27]. Nel campo dell'inseguimento di oggetti in un'immagine è noto anche come *condensation algorithm*[34]. Un'interpretazione visuale dei passaggi dell'algoritmo è riportata in Figura 4.5.

Una distribuzione di transizione $f(x_n^i|x_{n-1}^i)$ molto stretta tende a creare traiettorie troppo simili fra di loro nella fase di ricampionamento, portando a un impoverimento dei campioni. Per ovviare a questo problema spesso si esegue il ricampionamento solo se il numero efficace di campioni N_{eff} è inferiore a una soglia fissata.

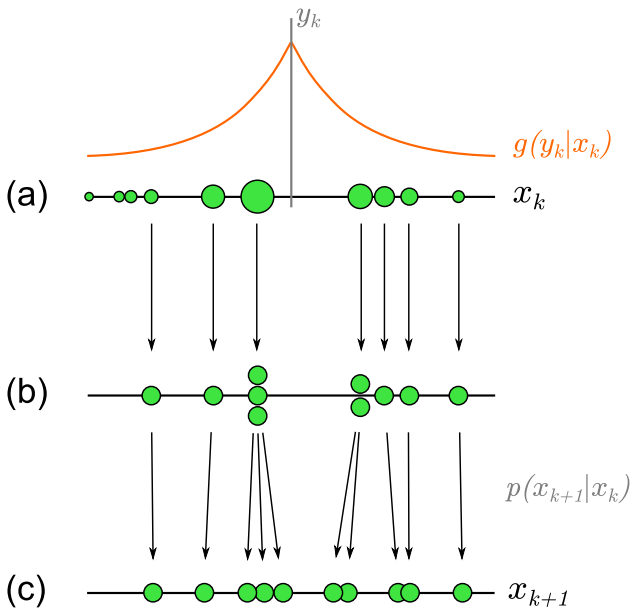


Figura 4.5: **Sequential Importance Resampling.** *Schema dell'algoritmo* Sequential Importance Resampling *nel caso del* bootstrap filter. *In* **(a)** *i campioni all'istante k sono pesati in base alla verosimiglianza $g(y_k|x_k)$. In* **(b)** *viene effettuato il passo di ricampionamento, e in* **(c)** *sono calcolati i nuovi campioni x_{k+1}. L'area dei cerchi è proporzionale al peso.*

Modelli di Markov di ordine superiore

Se lo stato corrente dipende non solo dallo stato precedente, ma da p stati precedenti (in altre parole, è dato da un processo markoviano di ordine p), è possibile applicare ugualmente le formule derivate nelle sezioni precedenti, definendo opportunamente un processo markoviano di primo ordine equivalente. Se $x_n = f(x_{n-1}, \ldots, x_{n-p}, d_n)$, si definisce il nuovo stato $\bar{x}_n = (\bar{x}_{i,n})$:

$$\begin{aligned} \bar{x}_{1,n} &= f(\bar{x}_{1,n-1}, \cdots \bar{x}_{p,n-1}) &&= x_n \\ \bar{x}_{2,n} &= \bar{x}_{1,n-1} &&= x_{n-1} \\ \vdots \quad &\vdots &&\vdots \\ \bar{x}_{p,n} &= \bar{x}_{p-1,n-1} &&= x_{n-p+1} \end{aligned} \tag{4.49}$$

Si osserva immediatamente che:

$$p(\bar{x}_n|\bar{x}_{n-1}) = p(x_n, \ldots, x_{n-p+1}|x_{n-1}, \ldots, x_{n-p}) = p(x_n|x_{n-1}, \ldots, x_{n-p})$$

e quindi la distribuzione di transizione di stato è uguale per i due sistemi. La funzione di verosimiglianza può dipendere in generale dagli ultimi p stati e, se le osservazioni sono influenzate solo dallo stato corrente, sarà non nulla solo in corrispondenza del primo componente di $\bar{x}_n$: $g(y_n|\bar{x}_n) := g(y_n|\bar{x}_{1,n}) = g(y_n|x_n)$. L'algoritmo Sequential Importance Resampling si applica come nel caso di sistemi del primo ordine, con una importante differenza: poiché lo stato $\bar{x}_n$ è definito non solo su uno stato precedente x_{n-1}, ma su p stati $x_{n-1}, \ldots, x_{n-p}$, è necessario ricampionare le particelle fino a profondità p ad ogni passo.

Si può motivare intuitivamente questo fatto osservando che, a causa del ricampionamento, le traiettorie delle particelle con peso alto tendono a ramificarsi in più traiettorie, mentre le traiettorie delle particelle con peso basso possono interrompersi. Questo significa che quando devono essere generate le particelle x_n^i esiste una ambiguità sulla scelta delle particelle degli istanti precedenti (per sistemi di primo ordine questo problema non esiste perché il ricampionamento è eseguito dopo la generazione dei campioni). Con un ricampionamento a profondità p sono generate N traiettorie distinte per gli ultimi p passi e quindi l'ambiguità viene rimossa.

4.4.4 Auxiliary particle filter

Il Sequential Importance Resampling con scelta canonica della distribuzione proposta tende ad essere sensibile agli outlier nelle osservazioni, perché concentra i pesi sulle poche particelle che si sono trovate casualmente nelle regioni a più alta verosimiglianza, ma che non sono necessariamente rappresentative della distribuzione a posteriori. L'*Auxiliary Particle Filter* (APF)[57] cerca di superare questo problema favorendo la scelta delle particelle più predittive in fase di ricampionamento. Prima di addentrarsi nella descrizione teorica tuttavia è utile dare una formula per l'approssimazione di $p(x_n|y_{0:n})$ analoga a quella già vista in (4.42).

Definito w_n^i il termine di aggiornamento dei pesi:

$$w_n^i = \frac{g(y_n|x_n^i) f(x_n^i|x_{n-1}^i)}{\pi(x_n^i|x_{0:n-1}^i, y_{0:n})} \tag{4.50}$$

e indicando con $\tilde{w}_n^i$ la sua versione normalizzata su tutti i campioni all'istante n, si ottiene:

$$p(x_n|y_{0:n}) \approx \sum_{i=1}^{N} \tilde{w}_n^i \delta(x_n - x_n^i) \tag{4.51}$$

Questa espressione si ricava facilmente da (4.42) sfruttando la formula ricorsiva dei pesi e le proprietà di δ:

$$p(x_{0:n}|y_{0:n}) \approx \sum_{i=1}^{N} \tilde{w}_n^i \delta(x_n - x_n^i) \tilde{\omega}_{n-1}^i \delta(x_{0:n-1} - x_{0:n-1}^i) \tag{4.52}$$

Marginalizzando su $x_{0:n-1}$ si ottiene subito la (4.51). I pesi normalizzati delle due distribuzioni a posteriori sono differenti solo se è stato applicato il SIS; nel caso del SIR il passo di ricampionamento rende uguali i pesi al passo $n-1$, e quindi a parte una costante moltiplicativa $\frac{1}{N}$ i pesi ω_n^i e w_n^i sono uguali.

Nell'Auxiliary particle filter viene definita una variabile casuale ausiliaria j che rappresenta l'indice di un campione scelto nell'istante $n-1$. Una formula per la distribuzione a posteriori di x_n e j si ottiene dalla distribuzione a posteriori dell'intera sequenza, marginalizzando su $x_{0:n-2}$:

$$p(x_n, x_{n-1}|y_{0:n}) = g(y_n|x_n) f(x_n|x_{n-1}) p(x_{n-1}|y_{0:n-1}) \tag{4.53}$$

Alla distribuzione a posteriori di x_{n-1} si sostituisce la sua approssimazione (4.51) e per ottenere la distribuzione di x_n e j dati i campioni all'istante $n-1$ si fissa un campione x_{n-1}^j. Si ottiene quindi:

$$p(x_n, j) := p(x_n, x_{n-1}^j|y_{0:n}) \propto f(x_n|x_{n-1}) g(y_n|x_n) w_{n-1}^j \tag{4.54}$$

L'approssimazione (4.51) valutata in x_{n-1}^j è pari al peso del campione moltiplicato per una delta di Dirac centrata sul campione, ma poiché si è passati da una variabile casuale continua x_{n-1} a una discreta j la delta è stata tralasciata; è stato tralasciato inoltre il fattore di normalizzazione dei pesi. Analogamente al particle filter classico, si ricorre all'importance sampling per campionare dalla distribuzione. La distribuzione proposta è:

$$\pi(x_n, j|y_{0:n}) = p(x_n|x_{n-1}^j) p(y_n|\mu_n^j) w_{n-1}^j \tag{4.55}$$

dove μ_n^j è un parametro della distribuzione $p(x_n|x_{n-1}^j)$, come media o moda, e ha lo scopo di rompere la dipendenza ricorsiva della formula (4.54) da x_n.

La distribuzione proposta viene adesso divisa in due termini, corrispondenti al campionamento rispettivamente di j e x_n:

$$j^{(i)} \sim \quad \pi(j|y_{0:n}) = \quad w_{n-1}^j p(y_n|\mu_n^j) \approx \quad w_{n-1}^j p(y_n|x_n) \tag{4.56}$$

$$x_n^i \sim \quad \pi(x_n|j, y_{0:n}) = \quad p(x_n|x_{n-1}) \tag{4.57}$$

e invece di campionare direttamente la coppia $(x_n^i, j^{(i)})$ si campiona prima j e poi, fissato $j^{(i)}$, si campiona x_n. I pesi sono:

$$w_n^i = \frac{p(y_n|x_n^i)}{p(y_n|\mu_n^{j^{(i)}})} \tag{4.58}$$

e permettono di approssimare la distribuzione a posteriori di (x_n, j):

$$p(x_n, j|y_{0:n}) \approx \sum_{i=1}^{N} \tilde{w}_n^i \delta(x_n - x_{n-1}^i)\delta(j - j^{(i)}) \tag{4.59}$$

Gli stessi pesi possono essere usati per approssimare la distribuzione a posteriori di x_n. Infatti per ogni i esiste un solo campione $j^{(i)}$, e quindi marginalizzando su j si ottiene:

$$p(x_n, y_{0:n}) \approx \sum_{i=1}^{N} \tilde{w}_n^i \delta(x_n - x_{n-1}^i) \tag{4.60}$$

Ora è possibile dare lo schema completo dell'algoritmo:

1. ricampiona le particelle scegliendo j: $j^{(i)} \sim w_n^j p(y_n|\mu_n^j)$
2. genera le nuove particelle a $j^{(i)}$ fissato: $x_n^i \sim p(x_n|x_{n-1}^{j^{(i)}})$
3. calcola i pesi w_n^i e normalizza
4. incrementa il tempo a $n+1$ e torna a 1.

L'Auxiliary particle filter può essere interpretato come una variante del boostrap filter nel quale il passo di ricampionamento è stato spostato all'iterazione successiva ed è stato introdotto un termine correttivo per i pesi $p(y_n|\mu_n^j)$ dipendente dall'osservazione y_n. Poiché questo termine influenza la frequenza con cui è scelto un campione, può distorcere la distribuzione approssimata, e quindi deve comparire al denominatore nella formula dei pesi. Il termine correttivo favorisce i campioni al tempo $t-1$ che sono in grado

di generare campioni al tempo t con più alta verosimiglianza e quindi pesi maggiori, mentre penalizza campioni che hanno più probabilità di avere peso basso, e quindi di essere scartati alla successiva iterazione. L'effetto finale è un insieme di particelle più rappresentativo della distribuzione a posteriori rispetto al SIR.

4.5 Conclusioni

Abbiamo qui descritto le varie tecniche necessarie a costruire un sistema di cattura e inseguimento del movimento umano. Nel successivo capitolo spiegheremo come queste tecniche sono combinate fra loro a formare il sistema sviluppato per questa tesi.

Capitolo 5

Algoritmo di inseguimento

In questo capitolo descriveremo nel dettaglio l'algoritmo di inseguimento sviluppato per questa tesi. Abbiamo scelto un modello a priori esplicito del corpo umano, come definito nella Sezione 2.2.1, per il quale abbiamo quindi fissato il numero di parti del corpo, i tipi di articolazioni che le collegano e il grado di libertà dei movimenti. Questa scelta è stata motivata dalla notevole riduzione del numero di parametri rispetto alle possibili configurazioni di voxel, che nelle nostre intenzioni dovrebbe portare a una migliore gestione degli errori di ricostruzione e di quantizzazione dei voxel.

Nella prima parte del capitolo (Sezione 5.1) descriviamo il modello che, essendo composto da molti elementi, merita una discussione approfondita. In particolare discutiamo la rappresentazione della forma del corpo (Sezione 5.1.1), la codifica della posa nei sistemi di riferimento (5.1.4) e i vincoli (5.1.5).

Nella seconda parte del capitolo (Sezione 5.2) descriviamo in dettaglio l'algoritmo di inseguimento. Per l'inseguimento abbiamo scelto di applicare Iterative Closest Point ad ogni parte del corpo separatamente, e abbiamo introdotto dei vincoli sulle articolazioni sfruttando il metodo delle corrispondenze artificiali di Knoop *et al.* (Sezione 4.3.1). Per le fasi di inizializzazione del modello e di classificazione dei voxel abbiamo sviluppato strategie specifiche, descritte nelle Sezioni 5.2.1 e 5.2.2. Al termine dell'elaborazione di ogni fotogramma applichiamo infine una serie di particle filter.

5.1 Modello del corpo umano

Il corpo di una persona è rappresentato da un modello volumetrico articolato, nel quale le parti del corpo sono rappresentate da solidi geometrici semplici collegati fra loro da articolazioni in una struttura ad albero (vedi

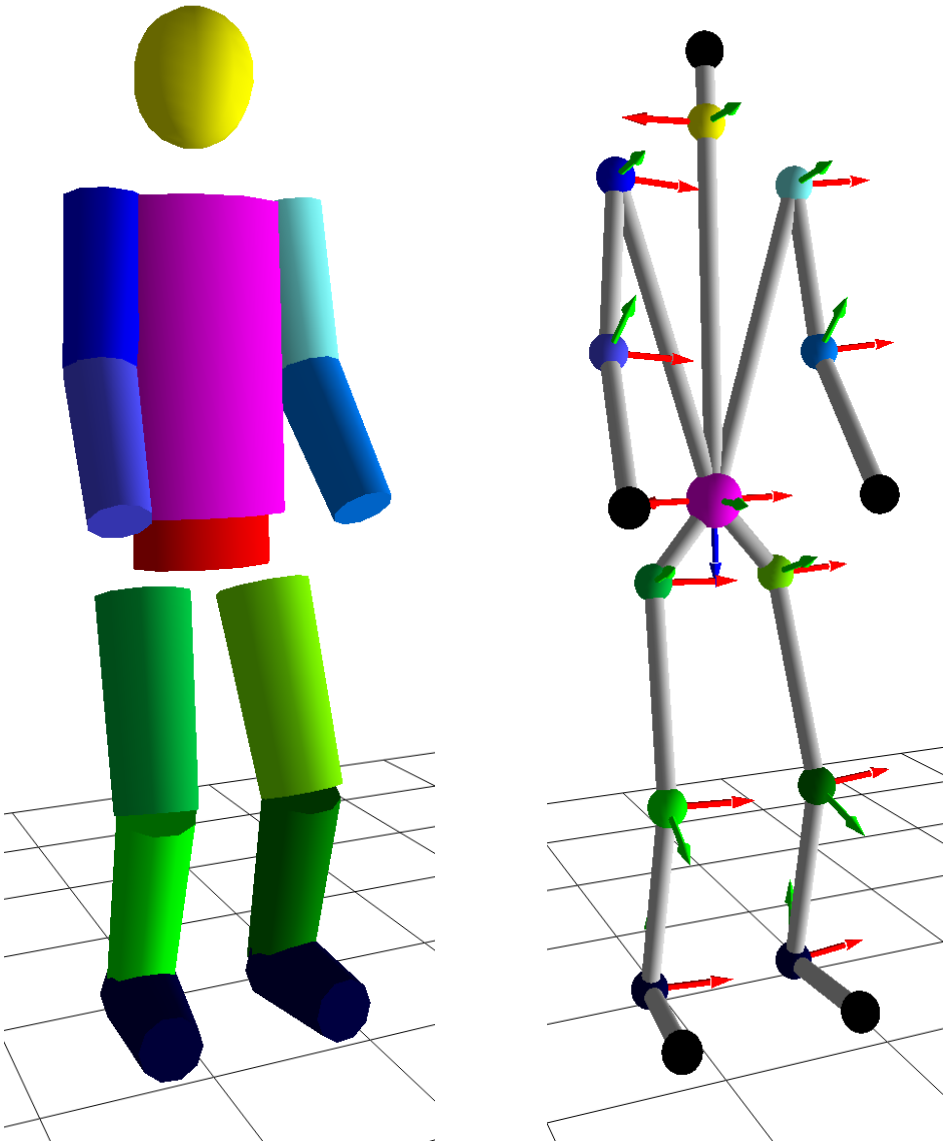
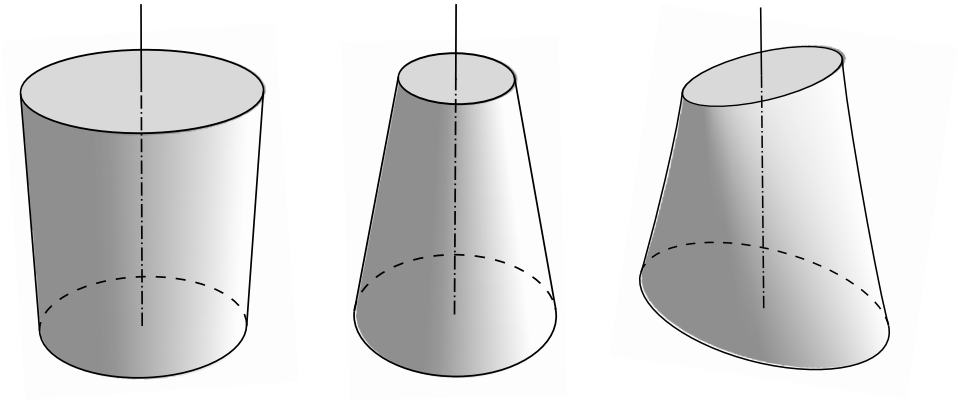


Figura 5.1: **Modello volumetrico e scheletrico.** *In figura è mostrato il modello volumetrico usato dall'algoritmo di inseguimento (a sinistra), e un modello scheletrico dei collegamenti fra i nodi (a destra), con i sistemi di riferimento relativi ad ogni parte riportati sul nodo corrispondente. L'asse x è disegnato in rosso, l'asse y in verde e l'asse z in blu.*



(a) (b) (c)

Figura 5.2: **Cilindri generalizzati.** *In figura sono riportate alcune forme geometriche ottenibili dalla equazione parametrica del cilindro generalizzato. Un cilindro ordinario* **(a)** *si ottiene quando i semiassi delle due basi sono uguali, un cono* **(b)** *quando i semiassi delle basi hanno lo stesso rapporto e una superficie complessa* **(c)** *quando i semiassi sono scelti arbitrariamente. In tutte le figure le due basi sono parallele.*

Figura 5.1). Ogni nodo dell'albero, partendo dal torso, corrisponde una parte del corpo, mentre le articolazioni sono definite dalle relazioni padre–figlio fra i nodi.

Poiché per ogni figlio esiste un solo padre, la coppia padre–figlio individua una articolazione che può essere associata univocamente al nodo figlio. Le informazioni sulle articolazioni sono quindi memorizzate nei nodi figli, fra le quali il tipo (vedi Sezione 4.3.1), i vincoli sugli angoli (vedi Sezione 4.2) e il sistema di riferimento locale del figlio rispetto al padre (vedi Sezione 5.1.4). Il sistema di riferimento è particolarmente importante perché, descrivendo posizione e orientamento di una parte del corpo rispetto all'altra, contiene tutte le informazioni necessarie a ricostruire la posa della persona in un dato istante.

Tutte le lunghezze in questo capitolo sono espresse in metri e calcolate a partire dalla dimensione dei voxel (2,5 cm). In accordo con Losh, Knoop *et al.*[43] abbiamo osservato che il modello ha prestazioni migliori quando è leggermente più piccolo del soggetto. Le lunghezze in questo capitolo non sono da considerarsi accurate dal punto di vista antropometrico, e in particolare la le reali dimensioni del soggetto sono sottostimate nell'ordine del 10–15%.

5.1.1 Elementi del modello

Le parti del corpo, ad eccezione della testa, sono rappresentate da cilindri generalizzati, cioè cilindri a base ellittica le cui basi possono avere differenti parametri ai due estremi. A seconda della scelta dei parametri si possono ottenere cilindri a base circolare o ellittica, tronchi di cono, oppure forme più complesse, come mostrato in Figura 5.2. La testa è rappresentata da un ellissoide. Il modello è dinamicamente configurabile, vale a dire che diverse configurazioni di ellissoidi e cilindri generalizzati permettono di definire diversi modelli di scheletro, anche durante l'esecuzione del programma. Nell'algoritmo i diversi tipi di solido sono gestiti trasparentemente, a patto supportino alcune operazioni basilari: in particolare, deve essere definita una funzione che restituisce il punto sulla superficie più vicino a un punto dato (vedi Sezione 5.1.3) e deve essere definita una *bounding box* per il solido (vedi Sezione 5.2.2). Ulteriori dettagli sull'implementazione sono riportati nell'Appendice B.2.2.

La superficie degli ellissoidi e dei cilindri sono definite da equazioni parametriche. Per il cilindro, scelte le coordinate $t \in [-\pi, \pi]$ e $u \in [0, 1]$, le equazioni parametriche della superficie del cilindro avente come asse l'asse z hanno, nel caso più generale, la forma:

$$\begin{aligned} x &= (a_1 u + (1-u)\, a_2) \cos t \\ y &= (b_1 u + (1-u)\, b_2) \sin t \\ z &= Lu \end{aligned} \tag{5.1}$$

dove a_1 e b_1 sono le lunghezze dei semiassi della prima base del cilindro, a_2 e b_2 i semiassi della seconda base e L l'altezza. Nel caso particolare in cui $a_1 = b_1$ e $a_2 = b_2$, le equazioni rappresentano un tronco di cono, mentre nel caso in cui $a_1 = a_2$ e $b_1 = b_2$, rappresentano un cilindro a base ellittica e con sezione costante.

Analogamente, chiamati a, b e c i tre semiassi, le equazioni parametriche per l'ellissoide hanno la forma:

$$\begin{aligned} x &= a \cos t \cos u \\ y &= b \sin t \cos u \\ z &= c \sin u \end{aligned} \tag{5.2}$$

dove $t \in [-\pi, \pi]$ e $u \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

Co	Nome	Id	Padre	Articolazione	GdL
	Torso	To	–	–	6
	Testa	Te	To	Ellittica	3
	Addome	Ad	To	Ellittica	3
	Coscia Dx	CoDx	Ad	Ellittica	3
	Gamba Dx	GaDx	CoDx	Cerniera	1
	Piede Dx	PdDx	GaDx	Cerniera	1
	Coscia Sx	CoSx	Ad	Ellittica	3
	Gamba Sx	GaSx	CoSx	Cerniera	1
	Piede Sx	PdSx	CoSx	Cerniera	1
	Braccio Dx	BrDx	To	Ellittica	3
	Avambraccio Dx	AvDx	BrDx	Cerniera	1
	Braccio Sx	BrSx	To	Ellittica	3
	Avambraccio Sx	AvSx	BrSx	Cerniera	1

Tabella 5.1: **Struttura del modello volumetrico.** *Nella tabella è riportato il colore e il nome di ogni nodo, l'abbreviazione del nome, l'abbreviazione del nodo padre (se esistente), il tipo di articolazione che collega il figlio al padre e il numero di gradi di libertà.*

Param.	Default	Asse
R_{To}	0,15	x
L_{To}	0,50	z
R_{Te}	0,1	z
L_{Ga}	0,34	z
R_{Ga}	0,08	x
L_{Br}	0,27	z
R_{Br}	0,07	x
L_{Pd}	0,2	y

Tabella 5.2: **Dimensioni predefinite del modello.** *Dimensioni predefinite del modello per l'attore* `Fede` *e le direzioni di scala associata ad ogni dimensione.*

Id	Dimensioni [a]				
	L	a_1	b_1	a_2	b_2
To	L_{To}	R_{To}	$\frac{2}{3} \cdot R_{To}$	R_{To}	$\frac{2}{3} \cdot R_{To}$
Te	$1,1 \cdot R_{Te}$	R_{Te}		R_{Te}	
Ad	$\frac{1}{6} L_{To}$	$0,8 \cdot R_{To}$	$\frac{1}{2} \cdot R_{To}$	$0,8 \cdot R_{To}$	$\frac{1}{2} \cdot R_{To}$
Co	L_{Ga}	R_{Ga}		$0,85 \cdot R_{Ga}$	
Ga	L_{Ga}	$0,8 \cdot R_{Ga}$		$0,7 \cdot R_{Ga}$	
Pd	L_{Pd}	$\frac{1}{3} \cdot L_{Pd}$		$\frac{1}{4} \cdot L_{Pd}$	
Br	L_{Br}	R_{Br}		$0,85 \cdot R_{Br}$	
Av	L_{Br}	$0,85 \cdot R_{Br}$		$0,7 \cdot R_{Br}$	

[a] Per l'ellissoide, $c = L$, $a_1 = b_1 = a$ e $a_2 = b_2 = b$

Tabella 5.3: **Dimensioni del modello volumetrico.** *Dimensioni del modello espresse in funzione dei parametri principali.*

5.1.2 Struttura del modello

Il modello del corpo umano è costituito da 13 parti: *Torso*, *Addome*, *Testa*, *Coscia Destra*, *Gamba Destra*, *Piede Destro*, *Coscia Sinistra*, *Gamba Sinistra*, *Piede Sinistro*, *Braccio Destro*, *Avambraccio Destro*, *Braccio Sinistro* e *Avambraccio Sinistro*. Il nodo radice dell'albero, che corrisponde al torso, è posizionato nella zona dell'addome, in modo che sia vicino al baricentro del voxelset e renda quindi più semplice il posizionamento del modello. Nella Tabella 5.1 è riportato uno schema riassuntivo del modello volumetrico con la lista delle varie parti del corpo e delle loro relazioni.

Il modello volumetrico è definito da una serie di parametri fondamentali che rappresentano alcune dimensioni importanti del corpo umano: il raggio del tronco R_{Tr}, l'altezza del tronco L_{Tr}, il raggio della testa R_{Te}, la lunghezza delle gambe L_{Ga}, il raggio delle gambe R_{Ga}, la lunghezza delle braccia L_{Br}, il raggio delle braccia R_{Br} e la lunghezza dei piedi L_{Pi}. Ai parametri sono assegnati inizialmente dei valori predefiniti, scelti adattandoli al meglio ad un attore di riferimento. Per gestire diversi tipi di attori abbiamo usato un fattore di scala distinto per i tre assi e abbiamo applicato tale fattore di scala in una direzione predefinita, scelta in base al contributo del parametro sulla dimensione finale del modello in quella direzione. Ad esempio la lunghezza del tronco influenza l'altezza (z) del modello ed è quindi scalata di un fattore s_z. I fattori di scala sono calcolati nella procedura di inizializzazione (vedi 5.2.1).

Le formule per calcolare le dimensioni del modello a partire dai parametri fondamentali sono riportate nella Tabella 5.3. I valori predefiniti e la direzione di scala per ogni parametro sono riportati nella Tabella 5.2.

5.1.3 Distanza di un punto

Per ogni parte del corpo è necessario calcolare la distanza di un punto dalla superficie di un elemento del modello e calcolare il punto sulla superficie più vicino al punto dato. La prima è necessaria ad assegnare un voxel a una specifica parte (vedi Sezione 5.2.2) e il secondo a generare i punti modello nell'algoritmo ICP (Sezione 5.2.3). Essendo le superfici dei solidi definite da equazioni parametriche, abbiamo scelto di calcolare queste quantità a partire da considerazioni geometriche.

La distanza di un punto dalla superficie è definita come la distanza dal punto più vicino, o in formule:

$$d = \|\mathbf{p} - \mathbf{v}_{min}\| = \min_{\mathbf{v} \in S} \|\mathbf{p} - \mathbf{v}\| \tag{5.3}$$

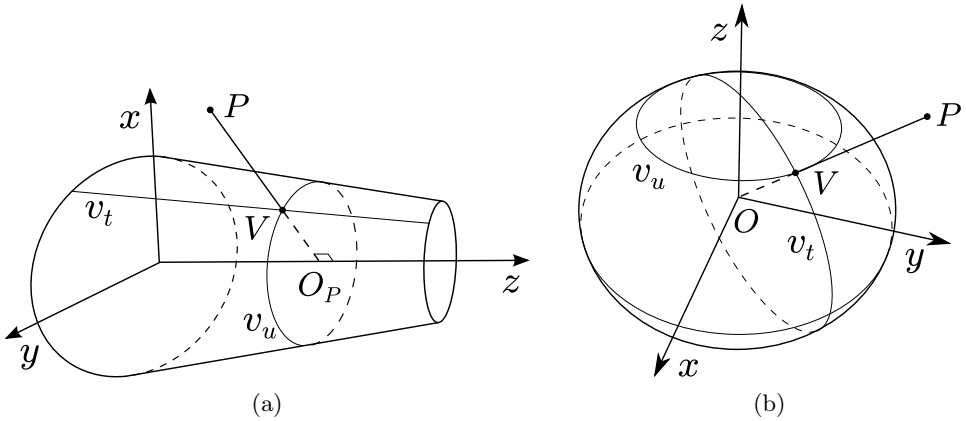


Figura 5.3: **Punto più vicino sulla superficie.** *Nel cilindro* **(a)** *il punto più vicino a P è calcolato costruendo la perpendicolare PO_V all'asse del cilindro passante per P e calcolando l'intersezione V fra la superficie e la superficie del cilindro. Il problema è ridotto al calcolo dell'intersezione fra la perpendicolare e l'ellisse a coordinata u costante individuato dalla coordinata v_u di V. Nell'ellisse* **(b)**, *il punto più vicino è individuato dall'intersezione fra la superficie e la congiungente OP di P all'origine O.*

dove S è la superficie in esame e $\mathbf{p}$ è il punto dato. L'unico problema che deve essere risolto è quindi il calcolo del punto più vicino sulla superficie di un ellissoide o di un cilindro generalizzato.

Una soluzione esatta in forma analitica è molto difficile da ottenere per i solidi scelti, in quanto già il sottoproblema della distanza minima da un ellisse sul piano è arduo[23]. Abbiamo scelto quindi delle approssimazioni basate su figure più semplici; in particolare, abbiamo approssimato il cilindro generalizzato con un cilindro a sezione costante, e l'ellissoide con una sfera. La scelta è giustificata dal fatto che, quando il punto $\mathbf{p}$ è molto vicino alla superficie, l'errore di approssimazione per il punto $\mathbf{v}_{min}$ è molto piccolo.

In un cilindro a sezione costante, $\mathbf{v}_{min}$ si trova, per ragioni di simmetria, sul piano passante per $\mathbf{p}$ e perpendicolare all'asse del cilindro. $\mathbf{v}_{min}$ ha quindi la stessa coordinata z di $\mathbf{p}$ e il problema si riduce al calcolo del punto più vicino a $\mathbf{p}$ sull'ellisse individuato dalla sezione del cilindro con il piano $z = p_z$. Il problema è ulteriormente approssimato usando come punto più vicino sull'ellisse l'intersezione $\mathbf{v}$ dell'ellisse con la congiungente fra $\mathbf{p}$ e l'origine sul piano (vedi Figura 5.3a). Indicati con a e b i parametri dell'ellisse, che sono definiti univocamente dalla coordinata u e quindi da p_z, si ottiene la formula finale:

Id	Padre	Posizione			Orientamento	
		x	y	z	θ	r
To	–	0	0	0	0	–
Te	To	0	0	$L_{To} + \frac{1}{2}R_{Te}$	0	–
Ad	To	0	0	0	180	y
CoDx	Ad	$-0,7 \cdot R_{To}$	0	$\frac{1}{3}L_{To}$	0	–
GaDx	CoDx	0	0	L_{Ga}	0	–
PdDx	GaDx	0	0	L_{Ga}	-90	x
CoSx	Ad	$0,7 \cdot R_{To}$	0	$\frac{1}{3}L_{To}$	0	–
GaSx	CoSx	0	0	L_{Ga}	0	–
PdDx	GaSx	0	0	L_{Ga}	-90	x
BrDx	To	R_{To}	0	L_{To}	180	y
AvDx	BrDx	0	0	L_{Br}	0	–
BrSx	To	R_{To}	0	L_{To}	180	y
AvSx	BrSx	0	0	L_{Br}	0	–

Tabella 5.4: **Sistemi di riferimento iniziali.** *La tabella riporta i sistemi di riferimento locali del modello nella posa iniziale. Il sistema di riferimento del torso è stato scelto coincidente con il sistema di riferimento globale per semplicità. L'asse di rotazione è specificato solo quando l'angolo di rotazione non è nullo.*

$$\begin{aligned} v_x &= \frac{p_x}{\|\mathbf{r}\|} \\ v_y &= \frac{p_y}{\|\mathbf{r}\|} \qquad \text{con} \qquad \mathbf{r} = \left(\frac{p_x}{a}, \frac{p_y}{b}\right) \\ v_z &= p_z \end{aligned} \tag{5.4}$$

Poiché il cilindro ha un'estensione finita lungo l'asse z, la formula è valida solo se $0 \leq p_z \leq L$ (vedi eq. 5.1). Quando questa condizione non è rispettata, p_z assume il valore dell'estremo più vicino.

Il punto più vicino su un ellissoide è calcolato, come nel caso della sfera, cercando l'intersezione $\mathbf{v}$ della superficie dell'ellissoide con la congiungente fra $\mathbf{p}$ e l'origine (Figura 5.3b). La formula finale è:

$$\mathbf{v} = \frac{\mathbf{p}}{\|\mathbf{r}\|} \qquad \text{con} \qquad \mathbf{r} = \left(\frac{p_x}{a}, \frac{p_y}{b}, \frac{p_z}{c}\right) \tag{5.5}$$

ed è l'analogo tridimensionale dell'approssimazione usata per l'ellisse nella precedente equazione, come è osservabile dalla similarità fra le due formule.

5.1.4 Sistemi di riferimento

Per codificare la posa della persona in ogni istante abbiamo associato un sistema di riferimento ad ogni parte del corpo. Il sistema di riferimento è

definito localmente e per ogni parte esprime posizione e orientamento rispetto al nodo padre. Ad esempio, posizione e orientamento della testa sono definite rispetto al tronco, quelle dell'avambraccio rispetto al braccio e così via. Il sistema di riferimento del nodo radice, non avendo padre, definisce la posizione e l'orientamento di tutto il modello rispetto a un sistema di riferimento globale.

Abbiamo scelto gli assi in modo da codificare naturalmente i movimenti più frequenti di ogni parte del corpo. Questo significa che nel caso del torso e dell'addome l'asse z è diretto lungo la spina dorsale, nel caso della testa lungo il collo e nel caso degli arti lungo la direzione dell'osso. Per gli elementi con un grado di libertà l'asse x è coincidente con l'asse di rotazione, mentre per i restanti elementi è sul piano frontale della posa iniziale (vedi Sezione 5.2.1). Abbiamo calcolato infine l'asse y come prodotto vettoriale fra l'asse z e l'asse x. Nella Tabella 5.4 sono riportati i parametri di ciascun sistema di riferimento nella posa predefinita.

L'orientamento è definito da un quaternione e la posizione da un vettore a tre componenti. L'orientamento è memorizzato anche come matrice di rotazione per rendere più efficienti le trasformazioni geometriche, che costituiscono una parte consistente del carico computazionale dell'algoritmo di inseguimento. Il sistema di riferimento globale di ogni parte può essere ricavato facilmente a partire dai sistemi di riferimento locali della catena cinematica che collega la parte corrente alla radice del modello. Indicati con $G_0^1, G_1^2, \ldots G_{n-1}^n$ i riferimenti locali sulla catena cinematica, ordinati dalla radice alla parte corrente, il sistema di riferimento globale cercato G è dato dalla formula:

$$G = G_0^1 \cdot G_1^2 \cdot \ldots \cdot G_{n-1}^n \tag{5.6}$$

Essendo i sistemi di riferimento definiti rispetto al sistema di riferimento della parte precedente nella catena, le matrici che li descrivono devono essere *post*moltiplicate fra loro. Per efficienza e semplicità di implementazione il riferimento globale è memorizzato fra i dati della parte del corpo corrispondente, e calcolato su richiesta con *lazy evaluation*. Nella tecnica della lazy evaluation il valore di un'espressione è calcolato solo quando è effettivamente richiesto e memorizzato in una variabile: in questo modo il calcolo del valore, solitamente oneroso, avviene solo quando è richiesto la prima volta o le variabili dell'espressione hanno cambiato valore.

5.1.5 Vincoli

Nel corpo umano le varie parti degli arti e del torso non possono muoversi liberamente le une rispetto alle altre, ma hanno limiti molto stringenti sui tipi di movimento e sulle ampiezze possibili. In un modello articolato è quindi necessario inserire dei vincoli sulla posizione e orientamento di ogni parte per escludere a priori le configurazioni non ammesse.

Un primo livello di vincoli è dato dalle corrispondenze artificiali per l'algoritmo ICP descritte nella Sezione 4.3.1, che penalizzano l'allontanamento di una parte rispetto all'altra e, a seconda del tipo di articolazione, limitano alcuni tipi di rotazione. Nelle nostre prove, riportate nella Sezione 6.2.3, abbiamo visto che questo tipo di vincolo, pur migliorando la stabilità di convergenza dell'algoritmo, non è sufficiente ad escludere pose palesemente irrealistiche. Per questo motivo abbiamo deciso di aggiungere un secondo livello di vincoli sulla posizione e sull'orientamento.

I vincoli sono stati definiti come regioni di ammissibilità nello spazio delle posizioni e orientamenti possibili. Nel caso della posizione, i vincoli sono dati da una *bounding box* definita per l'origine del sistema di riferimento locale. La bounding box è data un intervallo tridimensionale per le coordinate dell'origine e il centro della bounding box è dato dalla posizione predefinita dell'elemento corrente, calcolata in fase di inizializzazione. I vincoli sull'orientamento sono definiti separatamente per la componente di oscillazione e di torsione, seguendo il metodo sviluppato nel precedente capitolo (vedi Sezione 4.2).

I vincoli sui movimenti delle articolazioni sono stati ricavati dai dati antropometrici e biomeccanici della NASA[52]. Nella rappresentazione dei vincoli che abbiamo sviluppato non è possibile codificare ampiezze di oscillazione superiori a 180° e quindi, nei casi in cui questo limite viene superato, abbiamo limitato l'ampiezza a 180°. Esistono differenze significative nella mobilità delle articolazioni fra maschi e femmine, ma poiché il sistema sviluppato in questa tesi non è in grado di gestire differenze di genere, abbiamo costruito un insieme di vincoli unico per i due sessi. Nei casi in cui, fra i due sessi, si presentano differenze significative nell'escursione del movimento, abbiamo scelto l'intervallo di angoli più ampio, per garantire una maggiore tolleranza nei confronti dei movimenti estremi. I limiti per le rotazioni usati nel programma sono riassunti nella Tabella 5.5, dove ogni articolazione è associata al nodo figlio corrispondente.

I limiti sono applicati ogni volta che un sistema di riferimento è modificato, per assicurare che la posa sia sempre compatibile con i vincoli.

Elemento	Oscillazione X		Y		Torsione Z	
	min.	max.	min.	max.	min.	max.
Testa	-64	64	-70	100	-99	99
Addome[a]	-40	40	-20	20	-20	20
Coscia	-53	53	0	170	0	0
Gamba	0	0	-145	0	0	0
Piede	0	0	-80	17	0	0
Braccio destro	-173	60	-83	178	-96	126
Braccio sinistro	-60	173	-83	178	-126	96
Avambraccio	0	0	0	159	0	0

[a]Dati non NASA.

Tabella 5.5: **Limiti sugli angoli.** *Limiti per gli angoli usati nel modello cinematico, in gradi, associati alla parte del corpo corrispondente. Quando una direzione del movimento non è ammessa, gli estremi dell'intervallo sono posti a* 0*. Ad eccezione dell'addome, i limiti provengono dai dati biometrici raccolti dalla* NASA*.*

5.2 Algoritmo di inseguimento

L'algoritmo di inseguimento è la parte del sistema che si occupa di combinare il modello definito nella precedente sezione con una sequenza di voxelset, in modo da ottenere una sequenza di pose che costituiscono il movimento in termini di parametri delle articolazioni. Per questo lavoro di tesi abbiamo scelto un metodo che, data una stima sufficientemente vicina, cerca di calcolare la posa reale della persona ad ogni istante. Lo schema dell'algoritmo è riportato in Figura 5.4.

La stima è tipicamente data, come nel nostro caso, dalla posa calcolata all'istante precedente, per sfruttare a proprio vantaggio la coerenza temporale del movimento. La posa al momento precedente è disponibile per ogni fotogramma della sequenza a parte il primo, che quindi rende necessaria una una procedura di inizializzazione della posa. Nella fase di inizializzazione eseguiamo anche il calcolo dei parametri fondamentali del modello (Sezione 5.1.2) per accomodare le variazioni di corporatura fra i diversi attori. La procedura di inizializzazione è descritta nella Sezione 5.2.1.

Nell'algoritmo di inseguimento abbiamo applicato ICP separatamente ad ogni parte del corpo: dobbiamo quindi definire quali sono i voxel assegnati ad ogni parte, che costituiranno l'inseme dei dati per ICP. Questo compito è svolto dall'algoritmo di classificazione dei voxel, descritto nella Sezione 5.2.2. Poiché ICP lavora su nuvole di punti che rappresentano una superficie, sono

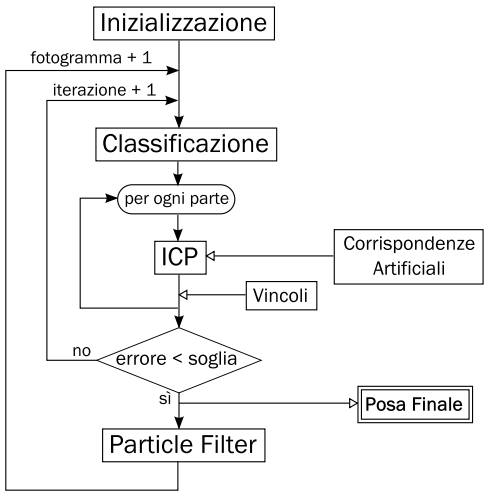


Figura 5.4: **Schema dell'algoritmo.** *Schema dell'algoritmo di inseguimento con i vari stadi di elaborazione, che saranno discussi nelle prossime sezioni. L'inizializzazione è eseguita solo una volta all'inizio dell'esecuzione, i vincoli sono applicati ad ogni parte al termine di ogni allineamento di Iterative Closest Point, e la posa è estratta prima che siano applicati i particle filter.*

classificati solo i voxel della superficie del voxelset, calcolata con un metodo sviluppato in una precedente tesi [29].

La classificazione deve essere eseguita almeno una volta all'inizio dell'elaborazione di un nuovo fotogramma, quando il cambiamento del voxelset invalida la precedente classificazione; tuttavia i nostri esperimenti hanno mostrato che si ottengono risultati migliori quando la classificazione è eseguita prima di ogni iterazione di ICP, in accordo con il risultato di altri gruppi di ricerca come Knoop *et al.*[38]. Una discussione sull'effetto della frequenza di classificazione sulle prestazioni è svolta nel prossimo capitolo (Sezione 6.2.2).

La posa finale trovata dall'algoritmo è data dall'ultima iterazione di ICP. Alla posa finale abbiamo applicato due particle filter per ogni parte del corpo, uno per la posizione e uno per l'orientamento, ottenendo una posa che costituisce la stima iniziale per il fotogramma successivo. I motivi di questa scelta e i dettagli dei filtri usati sono discussi in dettaglio nella Sezione 5.3.

5.2.1 Inizializzazione del modello

All'inizio dell'esecuzione l'algoritmo di inseguimento richiede una stima della posizione e orientamento iniziali del modello, una stima della posa iniziale e una stima delle dimensioni delle parti del corpo: tutte queste informazioni

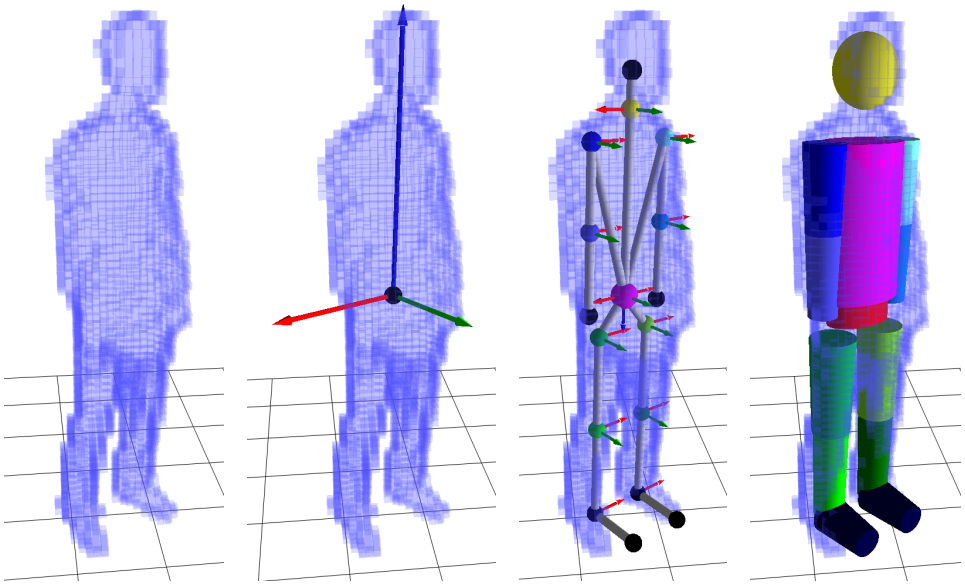


Figura 5.5: **Inizializzazione del modello.** *Nella procedura di inizializzazione a partire da un voxelset* **(a)** *si calcolano il baricentro, le direzioni principali e la dispersione lungo le direzioni principali* **(b)**. *Il valore della dispersione è dato dalla lunghezza delle frecce. Il baricentro e le direzioni principali definiscono rispettivamente la posizione e l'orientamento del sistema di riferimento globale del modello, a partire dal quale sono calcolati i sistemi di riferimento di ogni parte del corpo* **(c)**. *La dispersione definisce invece i fattori di scala del modello volumetrico* **(d)**.

sono fornite da una procedura di inizializzazione. Durante l'inizializzazione vengono create anche le corrispondenze artificiali per ICP, descritte nella Sezione 5.2.3. Le sequenze usate, prodotte dal lavoro di un precedente tesista[47], iniziano sempre con una posa predefinita, nella quale la persona è in posizione eretta, le gambe leggermente divaricate e le braccia lungo il corpo. Abbiamo scelto questa posa come posa iniziale per l'algoritmo.

Nella prima fase dell'inizializzazione calcoliamo la posizione e l'orientamento iniziali. La posizione è il baricentro del voxelset del primo fotogramma:

$$\mathbf{t} = \frac{1}{n} \sum_{i=0}^{n} \mathbf{x}_i \tag{5.7}$$

dove $\mathbf{x}_i$ sono le coordinate dei voxel, e n è il numero totale di voxel. Per calco-

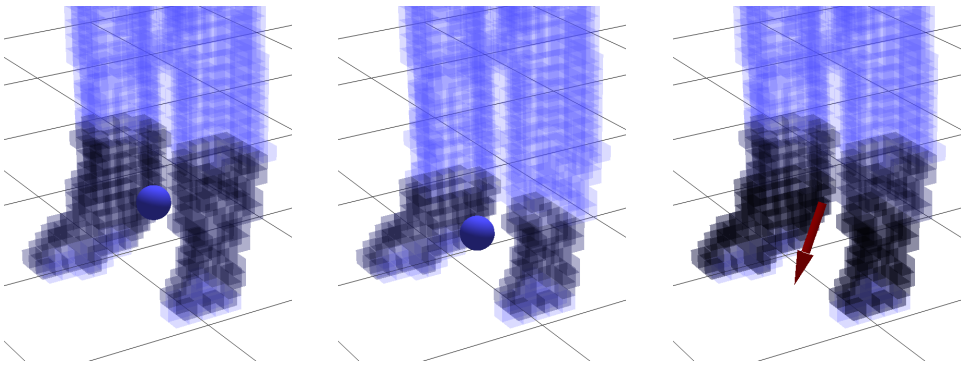


Figura 5.6: **Direzione frontale del modello.** *Tre passaggi nel calcolo della direzione frontale del modello. Inizialmente si individua la regione inferiore delle gambe* **(a)**, *la regione dei piedi* **(b)**, *e i relativi baricentri. Successivamente si calcola la direzione frontale come differenza fra i due baricentri* **(c)**.

lare l'orientamento iniziale abbiamo scelto di usare l'analisi delle componenti principali (*Principal Component Analysis*, o PCA) sul voxelset, interpretato come insieme di punti a tre dimensioni. La PCA permette di calcolare le direzioni principali di dispersione del voxelset, che nel caso di una figura eretta corrispondono all'asse longitudinale, trasversale e sagittale del corpo umano e permettono quindi di definire un sistema di riferimento naturale per il voxelset. Le direzioni principali sono calcolate applicando la decomposizione ai valori singolari (*Singular Value Decomposition*, o SVD) a una particolare matrice M di dimensioni $n \times 3$ costruita in modo che le righe contengano le coordinate dei voxel rispetto al baricentro. In formule:

$$M = U\Sigma V^T \tag{5.8}$$

dove $\mathrm{tr}(\Sigma) = (\sigma_x, \sigma_y, \sigma_z)$ sono i valori singolari di M, mentre le colonne della matrice unitaria V sono i vettori singolari destri di M e rappresentano le direzioni principali del voxelset. L'orientamento è definito dai versori **i**, **j** e **k** degli assi, le cui coordinate sono date dai vettori singolari, e quindi V può essere interpretata anche come la matrice di rotazione corrispondente all'orientamento cercato. I vettori singolari tuttavia non sono sufficienti a individuare univocamente la matrice di rotazione, perché una permutazione dell'ordine degli assi o inversioni di segno corrispondono a rotazioni compatibili con la decomposizione a valori singolari, ma corrispondenti a diversi

orientamenti. Abbiamo quindi deciso l'ordine di scelta e il verso degli assi in base ad euristiche.

Come asse z abbiamo scelto il vettore singolare unitario associato al valore singolare massimo che, nel caso di posa eretta, corrisponde alla direzione di massima estensione del voxelset; il verso è sempre nella direzione delle coordinate z positive. Come asse x abbiamo scelto il vettore associato al secondo valore singolare in ordine di grandezza, e l'asse y al valore singolare minimo. Abbiamo scelto l'asse y in modo che sia sempre diretto frontalmente rispetto al voxelset, mentre abbiamo scelto il verso di x in modo che gli assi x, y e z formino una terna destrorsa.

Possiamo calcolare la direzione frontale con una semplice euristica: scelto il nuovo asse z e indicata con z_f la minima coordinata z dei voxel rispetto al baricentro, abbiamo definito due regioni che corrispondono approssimativamente ai piedi (R_f) e alla regione inferiore delle gambe (R_l). Le regioni sono individuate dalle seguenti condizioni:

$$R_f = \{v | z_f < v_z < 0,9 z_f\} \qquad e \qquad R_l = \{v | z_f < v_z < 0,8 z_f\} \tag{5.9}$$

La direzione frontale è approssimata dalla differenza fra i baricentri delle due regioni $\mathbf{t}_f - \mathbf{t}_l$, e scegliamo il verso di $\mathbf{j}$ in modo che il prodotto scalare $\mathbf{j} \cdot (\mathbf{t}_f - \mathbf{t}_l)$ sia positivo.

Le dimensioni del modello volumetrico sono definite da otto parametri, il cui valore dipende dai fattori di scala nelle direzioni x, y, e z (vedi 5.1.2). I fattori di scala sono ottenuti a partire dai valori singolari calcolati in (5.8), e in particolare come rapporto fra i valori singolari del voxelset corrente e i valori singolari del voxelset di riferimento, indicati con σ_{0x}, σ_{0y} e σ_{0z}. Scalando i valori della matrice M di un fattore s i valori singolari variano di un fattore s^2, e quindi per ottenere un fattore di scala lineare è necessario calcolare la radice quadrata dei rapporti. Otteniamo infine le formule:

$$s_x = \sqrt{\frac{\sigma_x}{\sigma_{0x}}} \qquad s_y = \sqrt{\frac{\sigma_y}{\sigma_{0y}}} \qquad s_z = \sqrt{\frac{\sigma_z}{\sigma_{0z}}} \tag{5.10}$$

5.2.2 Classificazione dei voxel

Prima di ogni iterazione di ICP dobbiamo assegnare ad ogni parte del corpo un insieme di punti, che costituiscono i dati di ingresso all'algoritmo. Questo compito è assolto dalla procedura di classificazione dei voxel.

Un voxel è assegnato a una parte del corpo se rispetta almeno una delle seguenti condizioni:

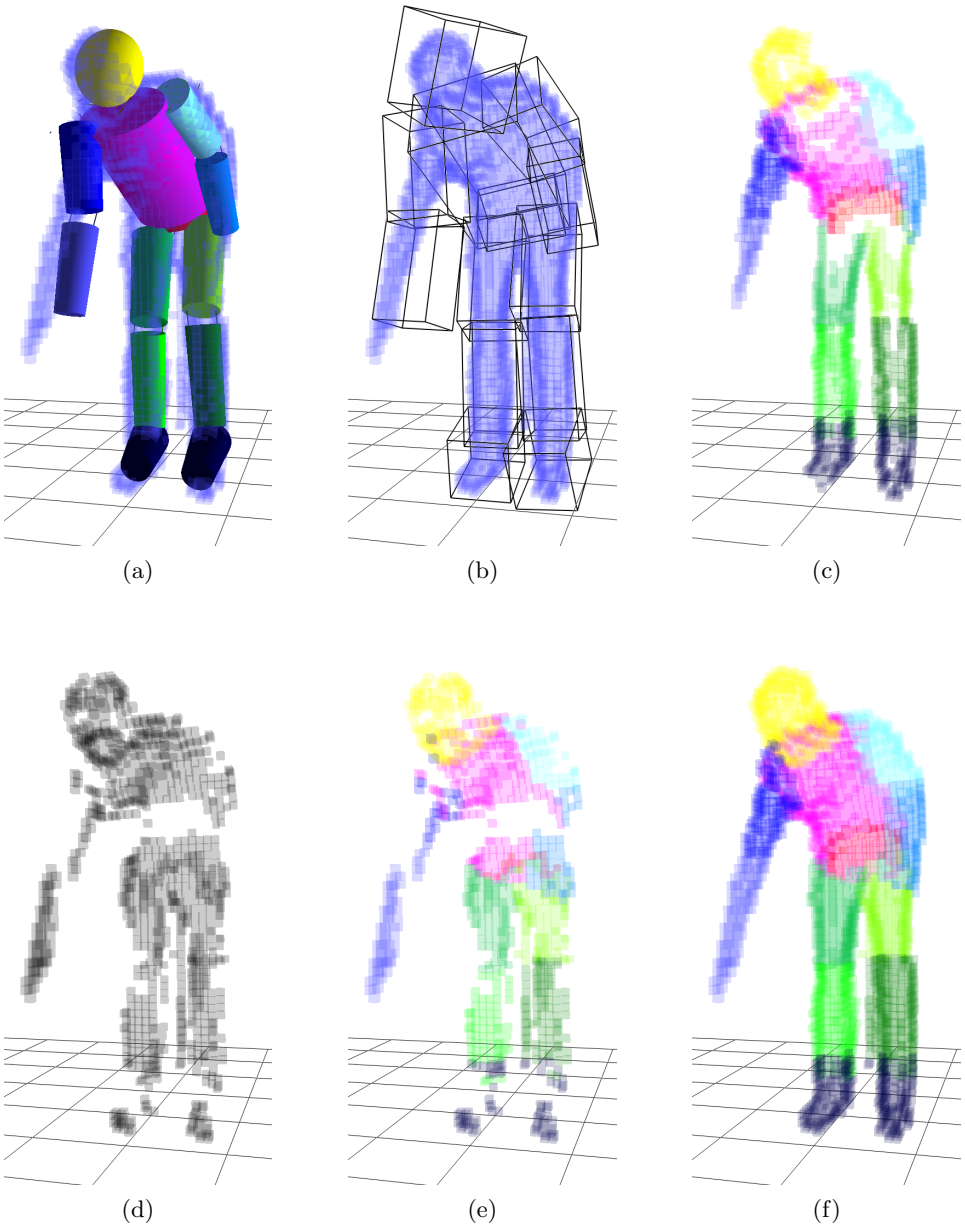


Figura 5.7: **Procedura di classificazione dei voxel.** *In figura sono riportate varie fasi della procedura di classificazione dei voxel. A partire dal modello volumetrico* **(a)** *si procede a calcolare le bounding box relative ad ogni parte del corpo* **(b)**. *All'interno di ogni bounding box sono classificati tutti i voxel con distanza inferiore alla soglia* **(c)** *e i voxel rimanenti formano l'insieme dei voxel residui* **(d)**. *I voxel residui sono assegnati alla parte del corpo più vicina* **(e)** *e, uniti ai precedenti voxel classificati, portano alla classificazione finale* **(f)**.

- la distanza dalla parte scelta è minore di una soglia prefissata h
- la distanza dalla parte scelta è minore delle distanze da tutte le altre parti

Un voxel può essere assegnato a più parti del corpo solo se la distanza da ciascuna di esse è minore della soglia; è quindi possibile una certa sovrapposizione fra diversi insiemi di voxel solo in corrispondenza delle articolazioni. Una verifica diretta delle condizioni richiede di calcolare un numero di distanze pari a *numero di parti* $\times$ *numero di voxel*, ma poiché il calcolo delle distanze è un'operazione computazionalmente onerosa, abbiamo elaborato un metodo per ridurre il numero di distanze che è necessario calcolare.

Ad ogni parte del corpo abbiamo associato una *bounding box* con una tolleranza sufficiente ad includere tutti i voxel con distanza minore o uguale alla soglia h. In questo modo è possibile verificare la prima condizione solo sui voxel contenuti nella bounding box, ed evitare la più difficile verifica della seconda condizione, perché è sufficiente che solo una delle due sia verificata.

Ogni bounding box è un intervallo simmetrico nello spazio tridimensionale individuato da un centro $\mathbf{c}_{BB}$ e dai raggi per le tre coordinate $\mathbf{s}_{BB}$. Per l'ellissoide il centro e i raggi della bounding box sono rispettivamente:

$$\mathbf{c}_{BB} = (0, 0, 0) \quad \mathbf{s}_{BB} = (a + h, b + h, c + h)$$

mentre per il cilindro generalizzato:

$$\mathbf{c}_{BB} = \left(0, 0, \frac{L}{2}\right) \quad \mathbf{s}_{BB} = \left(\max(a_1, a_2) + h, \max(b_1, b_2) + h, \frac{L}{2} + h\right)$$

Per ogni parte scegliamo i voxel che rientrano nella bounding box, e fra questi scegliamo quelli che si trovano a una distanza inferiore ad h; i voxel che non appartengono a nessuna bounding box, o troppo lontani dalla parte del corpo corrente sono aggiunti a una lista di voxel residui. Per ogni voxel residuo infine calcoliamo la distanza da ogni parte del corpo e assegniamo il voxel alla parte con distanza minore. L'algoritmo completo è descritto dallo pseudocodice nel riquadro 2. Se il numero di voxel residui è significativamente minore del numero di voxel totale, questo algoritmo garantisce una maggiore velocità di calcolo senza perdita di accuratezza.

5.2.3 Iterative Closest Point

L'algoritmo ICP (vedi Sezione 4.3) si occupa di generare, data una lista di voxel opportunamente classificati, la nuova posa e orientamento per ogni

Algoritmo 2 Classificazione dei voxel

```
for all parte ∈ M do
  for all v ∈ Voxelset do
    if v dentro BB(parte) then
      if d(v, parte) < d then
        Voxel(parte) ← Voxel(parte) ∪ {v}
      else
        residui ← residui ∪ {v}
      end if
    else
      residui ← residui ∪ {v}
    end if
  end for
end for
for all v ∈ residui do
  parte ← p  t.c.  d(v, p) = min_{p'∈M} d(v, p')
  Voxel(parte) ← Voxel(parte) ∪ {v}
end for
```

parte del corpo. L'allineamento avviene nel sistema di riferimento locale della parte del corpo corrente.

Per ogni parte l'algoritmo richiede due insiemi di punti, l'insieme dei dati $\{\mathbf{p}_i\}$ e l'insieme dei punti del modello geometrico $\{\mathbf{x}_i\}$. L'insieme dei dati è costituito dai punti generati dalla procedura di classificazione (vedi precedente sezione) e trasformati nelle coordinate locali della parte corrente. I punti del modello sono calcolati trovando per ogni punto dei dati $\mathbf{p}_i$ il punto più vicino $\mathbf{x}_i$ sulla superficie della parte corrente, seguendo la falsariga di Knoop *et al.*[37].

Le corrispondenze artificiali fra punti appartenenti ad elementi contigui del modello (vedi Sezione 4.3.1) sono generati in fase di inizializzazione. In ogni articolazione, i punti dell'elemento figlio sono creati sul piano xy del sistema di riferimento locale. Nel caso di articolazione universale, è creato un solo punto nell'origine. Nel caso di articolazione a cerniera, sono generati due punti con le coordinate:

$$P_1 = (-\sigma, 0, 0) \qquad P_2 = (\sigma, 0, 0) \tag{5.11}$$

Nel caso di articolazione ellittica infine si aggiungono ai due punti in (5.11) altri due punti:

$$P_3 = \left(0, -\frac{\sigma}{2}, 0\right) \qquad P_4 = \left(0, \frac{\sigma}{2}, 0\right) \tag{5.12}$$

Nel caso dell'articolazione a cerniera i punti sono disposti su una retta, e nel caso dell'articolazione ellittica su un ellisse. σ è un fattore di scala calcolato come prodotto della dimensione della parte del corpo lungo la direzione x (d_x) e da un termine che esprime la *cedevolezza* (*compliance*) c dell'articolazione. d_x per l'ellissoide è la lunghezza a del semiasse x, mentre per il cilindro generalizzato è la media dei semiassi delle due basi:

$$s_x = \frac{a_1 + a_2}{2} \tag{5.13}$$

Le coordinate di questi punti sono salvate sia nel padre sia nel figlio, nei rispettivi sistemi di riferimento locali, creando due insiemi di punti inizialmente coincidenti. Con l'evoluzione della posa il sistema di riferimento del figlio rispetto al padre cambia gradualmente, creando una discrepanza fra i due insiemi. L'algoritmo ICP, cercando di minimizzare la distanza fra punti corrispondenti, assicura che la distanza fra parti del corpo rimanga contenuta, e nel caso di articolazioni a cerniera o ellittiche, che alcune direzioni di rotazione siano penalizzate.

L'allineamento è eseguito seguendo l'ordine gerarchico delle parti nel modello. Un passo dell'algoritmo ICP è eseguito per ogni nodo, partendo dalla radice e imponendo le corrispondenze artificiali con le parti collegate. L'algoritmo procede finché il massimo errore su tutti le parti del corpo, come definito nella Sezione 4.3, è minore di una soglia prefissata, oppure quando è stato raggiunto il limite massimo di iterazioni.

I tipi di articolazione scelti per il modello sono riportati in Tabella 5.1 a inizio capitolo. Per le articolazioni della spalla e dell'anca abbiamo deciso di usare il tipo di articolazione ellittica, benché le loro caratteristiche siano meglio rappresentate dal tipo di articolazione universale, perché è necessario introdurre vincoli sulla torsione. Infatti quando, come nel nostro caso, il modello per la parte del corpo ha una simmetria radiale, esiste un'ambiguità sulla rotazione intorno all'asse di simmetria, che tende ad introdurre torsioni spurie nella posa. Questo fenomeno, a sua volta, porta negli arti ad errori di allineamento delle successive articolazioni (ginocchia e gomiti). Il disallineamento, in combinazione con i vincoli sugli angoli, può portare ad errori nella posa quando la rotazione calcolata da ICP ricade all'esterno dei vincoli, come nell'esempio riportato in Figura 5.8.

Poiché l'allineamento segue l'ordine gerarchico dal torso alle estremità degli arti, non è possibile propagare all'indietro l'errore e prevenire la torsione. Abbiamo osservato invece che un'articolazione ellittica con cedevolezza

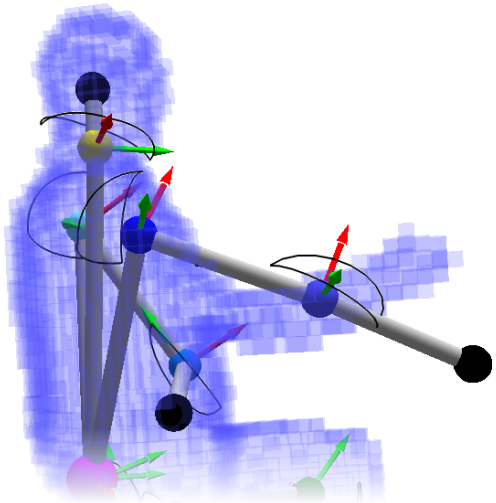
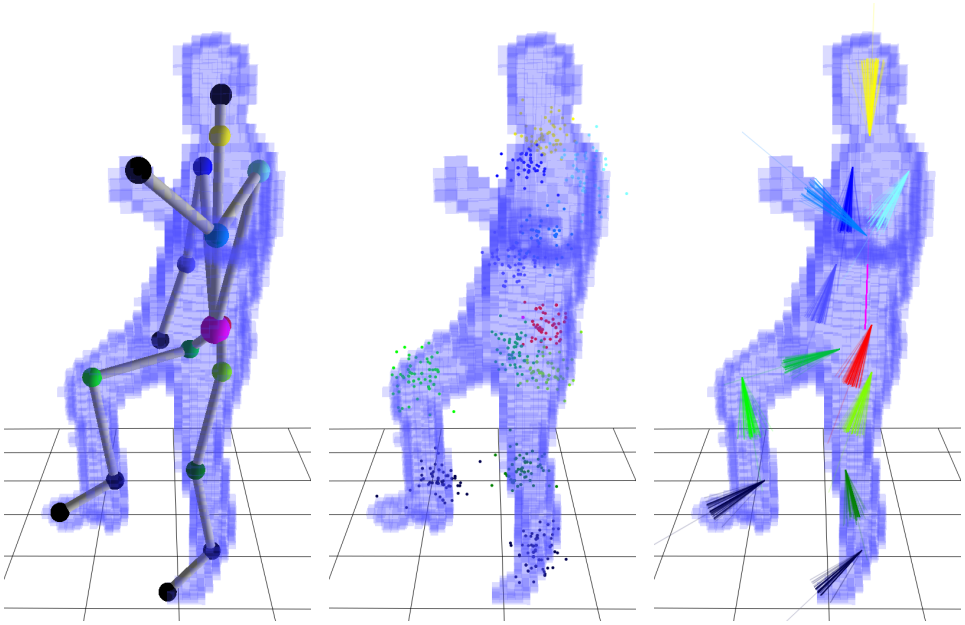


Figura 5.8: **Errore nelle articolazioni universali.** *In figura è mostrato il risultato dell'inseguimento con articolazioni universali per anche e spalle. Per ogni articolazione sono riportati in nero i limiti angolari. Il gomito destro ha un orientamento visibilmente errato, impedendo al braccio di piegarsi e di raggiungere quindi la posizione corretta.*

predefinita non limita significativamente l'ampiezza di movimento, mentre inibisce la torsione di un fattore sufficiente ad evitare, in gran parte dei casi, questo tipo di errori.

5.3 Particle filter

La posa calcolata dall'algoritmo di inseguimento può essere affetta da una serie di errori. Ad esempio, la classificazione di alcuni voxel può essere errata; oppure, dato che il modello del corpo umano è un'approssimazione della forma reale, le caratteristiche non modellate — come i particolari del viso o le dita della mano — possono introdurre rumore nella posa calcolata. La posa può essere inaccurata perché è stato superato il limite massimo di iterazioni, oppure perché la nuova configurazione è troppo diversa dalla precedente e l'algoritmo ICP è rimasto intrappolato in un minimo locale. La sequenza di pose in un movimento tuttavia presenta una significativa coerenza temporale, a causa di limiti nelle velocità e nelle accelerazioni dei movimenti umani. Questo fatto suggerisce di sfruttare la somiglianza fra pose consecutive per penalizzare le pose che si discostano troppo dall'andamento generale della sequenza, e che con maggiore probabilità sono affette da errori. A questo scopo abbiamo scelto di utilizzare *particle filter*.



(a) Scheletro (b) Posizione (c) Orientamento

Figura 5.9: **Particle filter per il modello.** *Ad ogni giunto del modello* **(a)** *sono associati due particle filter, uno per la posizione* **(b)** *e uno per l'orientamento* **(c)**. *Le particelle per la posizione sono rappresentati come punti, e le particelle per l'orientamento sono rappresentate dalla direzione dell'asse z ruotato dalla particella.*

Ad ogni articolazione abbiamo associato due particle filter, uno per la posizione e uno per l'orientamento di ogni parte nel sistema di riferimento globale, come illustrato in Figura 5.9. Le prestazioni dei particle filter tendono a degradare con l'aumentare del numero di dimensioni delle variabili, e poiché posizione e orientamento si possono considerare in larga parte indipendenti, possono essere gestiti separatamente. Entrambi i filtri sono di tipo SIR (vedi Sezione 4.4.3) con ricampionamento residuale dei pesi (vedi appendice A).

Per descrivere completamente i filtri è necessaria la definizione di una distribuzione iniziale delle particelle, di un modello di transizione di stato $f(x_{t+1}|x_t)$, di un modello di osservazione $g(y_t|x_t)$ e di uno stimatore di stato (vedi Sezione 4.4.1). Deve essere deciso inoltre il numero di particelle n. Come stimatore è stato scelto il *minimo errore quadratico medio* e come distribuzione iniziale è stata scelta la delta di Dirac, che corrisponde alla scelta di n valori uguali per le particelle. I modelli di transizione e di osservazione sono descritti nelle prossime sezioni.

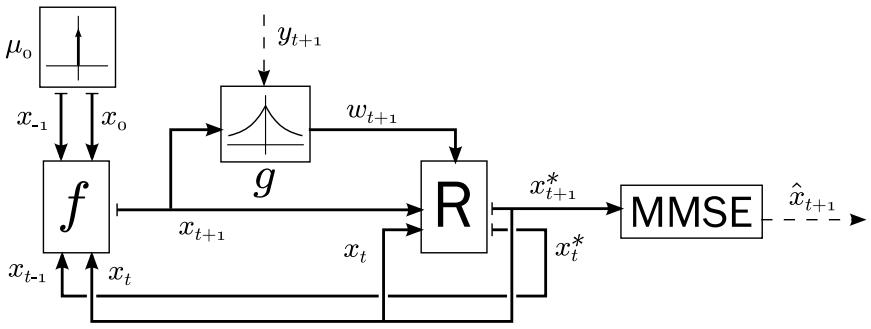


Figura 5.10: **Schema dell'algoritmo dei particle filter.** *Nella figura è riportato uno schema dell'algoritmo* SIR *usato per i particle filter. Le frecce in grassetto indicano un insieme di particelle, le frecce tratteggiate un valore singolo e una barra all'inizio di una freccia indica campionamento.*

Uno schema del funzionamento dei particle filter usati nell'algoritmo di inseguimento è riportato in Figura 5.10. Nel primo passo di *predizione* un nuovo insieme di particelle x_{t+1} è generato dalla funzione di transizione di stato f a partire dalle particelle ai due istanti precedenti x_{t-1} e x_t. Negli istanti iniziali, quando gli insiemi di particelle non sono stati ancora generati, le particelle sono estratte dalla distribuzione iniziale μ_0. Nel passo di *valutazione* sono generati i pesi w per ogni particella applicando la funzione di osservazione g alle particelle x_{t+1} e all'osservazione y_{t+1} generata dal risultato di ICP. Nel passo di *ricampionamento* R le particelle all'istante corrente x_{t+1}, all'istante precedente x_t e i pesi w permettono di generare due nuovi insiemi di particelle per gli istanti t e $t+1$. Dalle particelle ricampionate x^*_{t+1} infine si può calcolare il valore filtrato dello stato come minimo scarto quadratico medio (MMSE) delle particelle. Gli insiemi delle particelle agli istanti t e $t+1$ servono poi a calcolare le nuove particelle all'istante successivo.

I valori filtrati, diversamente dall'uso dei particle filter in letteratura, non costituiscono la posa finale del fotogramma corrente, ma la posizione iniziale del successivo. Abbiamo scelto questa strategia principalmente per due ragioni: la posa finale deve rispettare i vincoli di posizione e orientamento di ogni parte del corpo, ma applicare questi vincoli direttamente sull'insieme dei campioni ne altera le proprietà statistiche e degrada in modo inaccettabile le prestazioni del filtro; applicando invece i vincoli dopo il filtraggio, abbiamo osservato che i particle filter hanno difficoltà a seguire cambiamenti di velocità molto rapidi, a differenza di ICP che si è dimostrato molto adattabile. D'altro canto ICP è notoriamente sensibile ai minimi

locali, che la perturbazione stocastica dei particle filter aiuta ad evitare.

5.3.1 Modello per la posizione

Il modello di transizione di stato per la posizione è un modello MA (*Moving Average*) di ordine due costruito sull'ipotesi di moto a velocità costante. Lo stato $\mathbf{x}_{t+1}$ al tempo $t+1$ a partire dallo stato precedente $\mathbf{x}_t$ è ottenuto dalla seguente formula:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha\Delta\mathbf{x}_t + \mathbf{d}_t \qquad \text{con} \qquad \Delta\mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_{t-1} \tag{5.14}$$

La differenza finita $\Delta\mathbf{x}_t$ è una stima della velocità al tempo t e garantisce al modello adattabilità a velocità diverse, mentre $\mathbf{d}_t$ è un vettore gaussiano in $\mathbb{R}^3$ a media nulla e varianza $\boldsymbol{\sigma}_x$ che rappresenta il rumore inerente alla dinamica del modello. La stima della velocità è scalata da un parametro di attenuazione $\alpha < 1$ che rende stabile il sistema dinamico del modello MA ed evita quindi che le piccole perturbazioni date dal rumore siano arbitrariamente amplificate.

Il modello di osservazione è una funzione esponenziale che dipende da un unico parametro di diffusione σ_y:

$$w_t^{(i)} = e^{\sigma_y \|\mathbf{y}_t - \mathbf{x}_t^{(i)}\|} \tag{5.15}$$

e quindi ha valore massimo quando il campione coincide con l'osservazione, mentre decade rapidamente con l'aumentare dell'errore di predizione.

Essendo il modello di ordine due (per la presenza della differenza finita) nell'applicare il particle filter è necessario ricampionare non solo lo stato corrente $\mathbf{x}_t$, ma anche lo stato precedente $\mathbf{x}_{t-1}$, come spiegato nella Sezione 4.4.3.

5.3.2 Modello per l'orientamento

L'orientamento nel modello è descritto da quaternioni unitari che, diversamente dai vettori, non costituiscono uno spazio chiuso rispetto alla somma per componente e al prodotto per uno scalare. Questo comporta una problema nella definizione del modello di transizione di stato, del modello di osservazione e dello stimatore, perché la differenza per componente non restituisce, nel caso generale, un quaternione unitario. L'insieme dei quaternioni unitari tuttavia è chiuso rispetto alla moltiplicazione fra quaternioni, ed è quindi possibile riformulare le operazioni usate per la posizione in operazioni per l'orientamento.

L'analogo della somma è semplicemente il prodotto di quaternioni, con l'importante differenza che il prodotto fra quaternioni non è commutativo, ed è necessario specificare l'ordine dei fattori. La "differenza" fra i quaternioni q_1 e q_0, definita come operazione inversa della "somma", è ottenuta moltiplicando q_1 per l'inverso di q_0 che, nel caso dei quaternioni unitari, è il coniugato:

$$\Delta q = q_1 \cdot \bar{q}_0 \tag{5.16}$$

La differenza così definita può essere interpretata come approssimazione della velocità di variazione dell'orientamento, o *momento angolare*. L'analogo della moltiplicazione è l'elevamento a potenza, definito come l'esponenziale del logaritmo (vedi Sezione 4.1.3):

$$q^\alpha = \exp(\alpha \log(q)) \tag{5.17}$$

L'esponenziale di un quaternione puramente immaginario ($q_w = 0$) è sempre unitario e può quindi essere sfruttato per generare il rumore dell'orientamento a partire a un vettore casuale in $\mathbb{R}^3$:

$$d_t = \exp(n) \qquad n = n_1\mathbf{i} + n_2\mathbf{j}, n_3\mathbf{k} \qquad n_1, n_2, n_3 \sim \mathcal{N}(0, \sigma_x) \tag{5.18}$$

Ora che sono stati definiti tutti gli elementi necessari, è possibile scrivere un modello di transizione di stato analogo a (5.14) come:

$$q_{t+1} = d_t \cdot \exp\left(\alpha \log(\Delta q_t)\right) \cdot q_t \qquad \text{con} \qquad \Delta q_t = q_t \cdot \bar{q}_{t-1} \tag{5.19}$$

Il modello dell'osservazione è definito da un esponenziale dello scarto fra lo stato x_t e l'osservazione y_t come in (5.15), e quindi è necessario definire una distanza fra quaternioni unitari. Una possibile definizione è l'angolo fra quaternioni, che a sua volta è il modulo del logaritmo dello scarto $y_t\bar{q}_t$:

$$w_t^{(i)} = e^{\sigma_y \| \log y_t \cdot \bar{q}_t^{(i)} \|} \tag{5.20}$$

Lo stimatore ai minimi quadrati non può essere calcolato sulla base di queste considerazioni, perché la non commutatività del prodotto di quaternioni rende ambiguo l'analogo della somma di tre o più quaternioni. Esiste tuttavia una formula iterativa per il calcolo corretto della somma pesata di quaternioni[61][56]. A partire da una stima iniziale $\hat{q}^{(0)}$, l'errore $e^{(j)}$ al passo j è dato dalla formula:

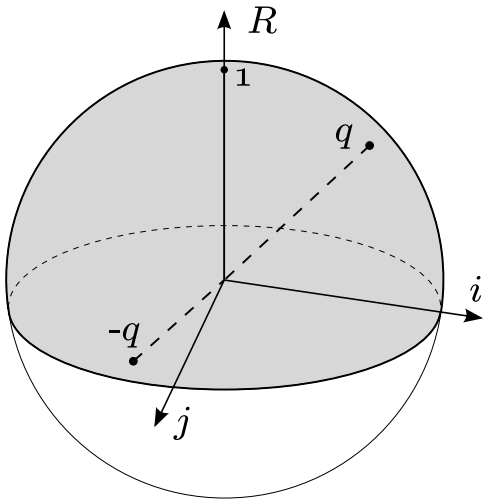


Figura 5.11: **Quaternioni equivalenti.** *Nella figura è riportata la sfera $\mathbb{S}^3$ dei quaternioni unitari senza la componente k, per poter essere rappresentata nello spazio tridimensionale. L'asse $\mathbb{R}$ è la retta reale e l'intersezione 1 con la sfera è la rotazione nulla. Due quaternioni opposti sulla sfera rappresentano la stessa rotazione, e quindi un solo emisfero (area ombreggiata) è sufficiente a rappresentare tutte le rotazioni.*

$$e^{(j)} = \sum_i w_i \log\left((\hat{q}^{(j)})^{-1} \cdot q_i\right) \qquad \sum_i w_i = 1 \tag{5.21}$$

e la stima al passo $(j+1)$-esimo è:

$$\hat{q}^{(j+1)} = \hat{q}^{(j)} \cdot \exp(e^{(j)}) \tag{5.22}$$

La formula non è strettamente necessaria nel caso particolare di pesi uniformi: in quel caso si dimostra infatti che il baricentro normalizzato corrisponde alla soluzione corretta[28].

Nell'applicare queste formule è importante ricordare che un quaternione q e il suo opposto $-q$ rappresentano la stessa rotazione, perché la mappa fra le rotazioni e i quaternioni è suriettiva ma non biunivoca; in particolare, i quaternioni contenuti in un emisfero della sfera $\mathbb{S}^3$ nello spazio $\mathbb{R}^4$ possono rappresentare tutte le rotazioni possibili (Figura 5.11). Nella somma pesata possono esserci artefatti quando rotazioni molto simili sono rappresentate da quaternioni in emisferi diversi, e diventa allora necessario verificare che tutti i quaternioni appartengano allo stesso emisfero. Il modo più semplice per verificare questa condizione è confrontare l'orientamento fra il quaternione da sommare q_k e la somma parziale s_k, usando il prodotto scalare in $\mathbb{R}^4$:

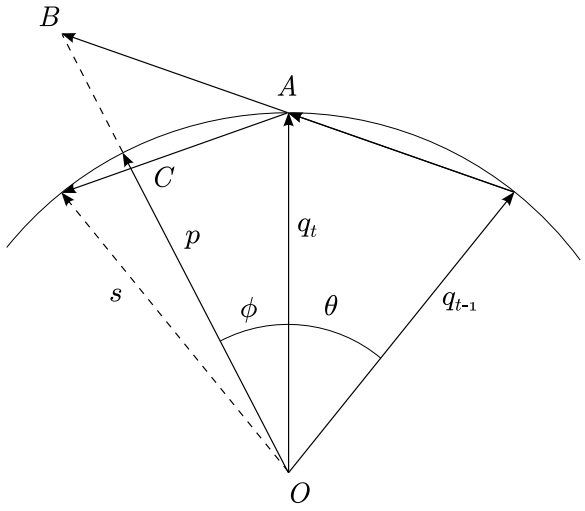


Figura 5.12: **Scarto fra quaternioni.** *Approssimazione del prodotto di $\Delta q_t \cdot q_t$ con la somma $q_t + (q_t - q_{t-1})$. Il quaternione s è la soluzione esatta, p è l'approssimazione. L'angolo $C\widehat{A}B$ è pari a θ, e tanto più è piccolo tanto più il segmento BA è vicino a CA.*

$$\langle q_k, s_k \rangle = w_q w_s + \mathbf{v}_q \cdot \mathbf{v}_s \geq 0 \tag{5.23}$$

dove w e $\mathbf{v}$ sono rispettivamente la parte scalare e vettoriale dei quaternioni.

Approssimazione per piccole rotazioni

Pur essendo le considerazioni della precedente sezione valide in generale, quando il momento angolare è molto piccolo è possibile usare per i quaternioni un modello lineare analogo a (5.14), senza incorrere in errori significativi. In questa sezione presentiamo una breve giustificazione.

Il nostro obiettivo è esprimere la soluzione esatta:

$$q_{t+1} = d_t \cdot \exp\left(\alpha \log(q_t \cdot \bar{q}_{t-1})\right) \cdot q_t \tag{5.24}$$

con una soluzione approssimata:

$$q_{t+1} = q_t + \alpha\,(q_t - q_{t-1}) + \log(d_t) \tag{5.25}$$

nella quale il risultato finale è normalizzato per ottenere un quaternione unitario.

Se esprimiamo il quaternione $\Delta q_t = q_t \cdot \bar{q}_{t-1}$ come una rotazione di 2θ attorno all'asse $\mathbf{u}$, i quaternioni q_t e q_{t-1} formano un angolo θ nello spazio

$\mathbb{R}^4$ e quindi moltiplicare un quaternione q per Δq_t corrisponde a ruotarlo di un angolo θ nello stesso piano che contiene q_t e q_{t+1}.

Una buona approssimazione della moltiplicazione per Δq_t è data dalla somma della quantità $q_t - q_{t-1}$ seguita dalla normalizzazione: infatti il risultato p appartiene ancora al piano di q_t e q_{t+1}, e per θ piccolo è molto vicino al risultato s ottenuto moltiplicando un quaternione per Δq_t. Una giustificazione è riportata in Figura 5.12. Per angoli piccoli l'angolo ϕ fra s e q_t è approssimato dalla corda CA, che a sua volta è ben approssimata dal segmento BA; d'altra parte la lunghezza di BA è il modulo della differenza $q_t - q_{t-1}$ ed è un'approssimazione dell'angolo θ. Quindi sommare $q_t - q_{t-1}$ a un quaternione equivale approssimativamente a ruotarlo di un angolo θ sullo stesso piano, e quindi a moltiplicarlo per Δq_t.

L'elevamento a potenza di Δq_t, implementato come l'esponenziale del logaritmo scalato di una costante α, equivale a costruire un quaternione che ruota di $\alpha\theta$ nello stesso piano di Δq_t. Seguendo lo stesso ragionamento del caso precedente, si può mostrare che la somma di $\alpha\,(q_t - q_{t-1})$ è una buona approssimazione della moltiplicazione per Δq_t^{α}.

Resta da dimostrate la validità dell'approssimazione per il modello dell'osservazione (5.20), vale a dire:

$$e^{\sigma_y \| \log y_t \cdot \bar{q}_t^{(i)} \|} \approx e^{\sigma_y \| y_t - q_t^{(i)} \|} \tag{5.26}$$

Abbiamo già osservato che per angoli molto piccoli il modulo della differenza, dato dall'esponente nel secondo membro, approssima l'angolo fra i quaternioni, dato dall'esponente nel primo membro, e quindi le due formule sono approssimativamente equivalenti.

Capitolo 6

Analisi e risultati sperimentali

Dopo aver descritto nel dettaglio il funzionamento dell'algoritmo di inseguimento, è arrivato il momento di valutarne le prestazioni. I risultati dell'algoritmo saranno valutati sia attraverso un'ispezione visuale di alcune sequenze caratteristiche, sia attraverso la valutazione dell'errore di inseguimento. L'algoritmo è costituito da diversi stadi di elaborazione e dipende da molti parametri. In questo capitolo saranno analizzati gli effetti in termini di prestazioni della scelta di diversi parametri o dell'esclusione di alcuni stadi di elaborazione.
Una delle caratteristiche più importanti dell'algoritmo sviluppato è la velocità di esecuzione. In aggiunta alle considerazioni sull'accuratezza, saranno valutate le prestazioni in termini di velocità e i compromessi possibili fra qualità della posa e velocità di elaborazione.

6.1 Risultati sperimentali

L'algoritmo di inseguimento dipende da un certo numero di parametri, fra i quali i più importanti per l'accuratezza sono:

- la soglia dell'algoritmo di classificazione dei voxel h
- il massimo errore tollerato dall'algoritmo ICP ε_{Thr}
- il numero massimo di iterazioni di ICP N_{max}
- la cedevolezza delle corrispondenze artificiali c
- il numero di particelle nel particle filter per la posizione e l'orientamento n

Param.	Default
h	$0,03$
ε_{Thr}	10^{-5}
N_{max}	15
c	$0,7$
n	100
σ_t	$0,03$
σ_R	$0,05$

Tabella 6.1: **Parametri predefiniti dell'algoritmo.** *Nella tabella sono riportati i principali parametri dell'algoritmo e i loro valori predefiniti. L'errore massimo ε_{Thr} rappresenta uno scarto quadratico medio e corrisponde ad un errore per voxel pari a circa 3 cm, comparabile quindi con il valore h della soglia nell'algoritmo di classificazione.*

- la varianza nel particle filter per la posizione σ_t
- la varianza nel particle filter per l'orientamento σ_R

In questa sezione analizzeremo le prestazioni dell'algoritmo per i valori predefiniti, riportati nella tabella 6.1. Nella successiva sezione invece analizzeremo le prestazioni quando uno o più parametri sono variati, in modo da apprezzarne l'importanza relativa sull'accuratezza dell'inseguimento. Tuttavia, prima di discutere i risultati, descriveremo brevemente i dati che abbiamo usato per testare l'algoritmo e i criteri scelti per valutare l'accuratezza dell'inseguimento.

6.1.1 Dati sperimentali

I dati consistono in una serie di dieci azioni svolte da cinque attori in cinque diverse ripetizioni ciascuno, per un totale di 250 differenti sequenze. Tutte le sequenze iniziano e finiscono con l'attore fermo, in posizione eretta e con le braccia lungo i fianchi. Dei cinque attori, quattro sono di sesso maschile (`Fede`, `Gabbo`, `Massi` e `Teo`) e una di sesso femminile (`Vale`). Nella Figura 6.1 abbiamo riportato i voxelset di ogni attore all'inizio della sequenza `Apri-002`, in modo da apprezzarne la differenza di corporatura. Le dieci azioni sono: `Apri`, `Calcio`, `Camminata`, `Crouch` (rannicchirsi), `Grasp` (afferrare un oggetto), `Marcia`, `Puntamento`, `Spinta`, `SpostaDx` e `Tira`. Per ciascuna di esse riportiamo una breve descrizione.

Nella sequenza `Apri` l'attore, in posizione eretta, mima l'azione di aprire un armadietto all'altezza della testa.
Nella sequenza `Calcio` l'attore esegue un calcio ampio con la gamba destra.
Nella sequenza `Camminata` l'attore inizia a camminare, compie due o tre passi, e poi si ferma.

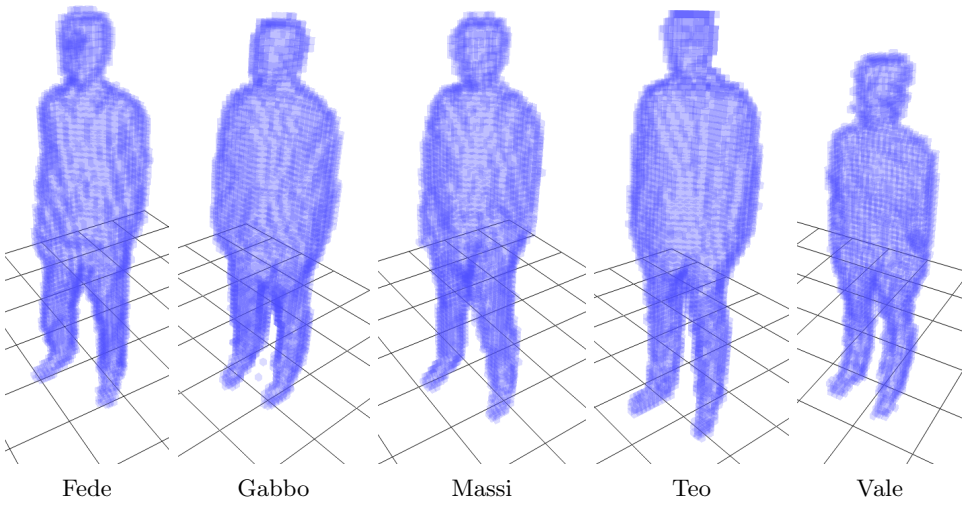


Figura 6.1: **Attori nella posa predefinita.** *Voxelset degli attori all'inizio della sequenza `Apri-002`. É possibile vedere la differenza di corporatura fra i diversi attori, in particolare `Teo` e `Vale`, che rende indispensabile una procedura di inizializzazione del modello.*

Nella sequenza `Crouch` l'attore compie il movimento di rannicchiarsi a terra, mantiene la posizione accovacciata per qualche istante e poi compie il movimento inverso, tornando in posizione eretta.
Nella sequenza `Grasp` l'attore mima l'azione di raccogliere con la mano destra un oggetto da terra.
Nella sequenza `Marcia` l'attore mima una tipica marcia, con movimenti ampi delle gambe e delle braccia.
Nella sequenza `Puntamento` l'attore indica con un dito un punto nel cielo, usando il braccio destro.
Nella sequenza `Spinta` l'attore solleva le braccia vicino al corpo e poi le allontana, come se dovesse farsi spazio in uno spazio angusto; infine le riporta accanto al corpo seguendo il movimento inverso.
Nella sequenza `SpostaDx` l'attore mima il gesto di allontanamento con il braccio destro.
Nella sequenza `Tira` l'attore piega le braccia sul petto e poi le riporta nella posizione di riposo, mantenendole sempre vicino al corpo.

6.1.2 Criteri di valutazione

I risultati sono stati valutati sia mediante ispezione visuale, sia attraverso indicatori più accurati estratti dall'algoritmo. Gli indicatori che abbiamo scelto sono l'errore sulla posa finale, l'errore di convergenza di ICP, il numero di iterazioni e il tempo di esecuzione.

L'indicatore più importante è l'errore di inseguimento, che abbiamo scelto come la radice quadrata dell'errore ICP:

$$e = \left(\frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{p}_i\|^2 \right)^{\frac{1}{2}} \tag{6.1}$$

dove $\{\mathbf{x}_i\}$ e $\{\mathbf{p}_i\}$ rappresentano i punti del modello e i punti dati rispettivamente. L'uso della radice quadrata garantisce che l'errore sia commensurabile con una lunghezza e in particolare possa essere interpretato come una distanza media fra i punti del modello e i dati sperimentali. L'errore per ogni fotogramma è calcolato come il massimo errore su tutte le parti del corpo, calcolato sull'ultima iterazione, in modo da penalizzare gli errori significativi anche quando sono localizzati.

Lo stimatore può sia sovrastimare che sottostimare l'errore reale: gli artefatti del voxelset o imprecisioni dei parametri antropometrici incrementano l'errore anche in caso di posa corretta, mentre quando la forma del voxelset è ambigua la procedura di classificazione può assegnare i voxel alla parte sbagliata e portare ad un errore apparente minore di quello effettivo. Nonostante questi limiti abbiamo scelto questo stimatore perché è facilmente calcolabile a partire dai dati dell'algoritmo e perché, come vedremo nelle prossime sezioni, in gran parte dei casi è un buon indicatore dell'accuratezza della posa.

L'errore di convergenza è il valore assoluto della differenza fra gli errori di ICP fra due iterazioni consecutive ed è usato come criterio di arresto dell'algoritmo (vedi Sezione 4.3). Questa informazione, così come il numero di iterazioni, è facilmente ricavabile dall'algoritmo.

Nel tempo di esecuzione abbiamo incluso il tempo impiegato dall'algoritmo ad eseguire la classificazione dei voxel, l'allineamento del modello e il filtraggio della posa, mentre abbiamo escluso l'estrazione della superficie del voxelset, che ha un contributo trascurabile sul tempo totale. Il programma di test è stato realizzato in `C++` con singolo thread e compilato in `gcc` con ottimizzazioni moderate (`-O2`). Il tempo è stato misurato su una macchina dotata di processore AMD M320 Dual Core.

6.1.3 Analisi delle sequenze

In questa sezione analizziamo in dettaglio tre sequenze, in modo da valutare approfonditamente i punti di forza e di debolezza dell'algoritmo.

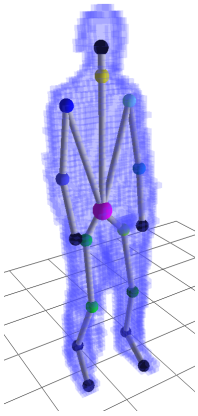
La sequenza `Apri` è stata scelta perché per una parte consistente del movimento le braccia sono molto vicine fra loro e i voxel presentano spesso degli artefatti che rendono difficile separarle chiaramente. La sequenza `Marcia` è stata scelta perché l'attore mette alla prova la capacità dell'algoritmo di seguire variazioni significative e relativamente veloci della posa. La sequenza `Crouch` è stata scelta perché nella fase cruciale del movimento il voxelset presenta un gran numero di occlusioni ed è quindi difficile distinguere le varie parti del corpo. Quest'ultima sequenza rappresenta un caso nel quale l'algoritmo non è in grado di seguire correttamente tutto il movimento; tuttavia permette di valutare le capacità di recupero della posa.

Nella prossima sezione invece analizzeremo alcune sequenze che presentano particolari difficoltà di inseguimento, che saranno il punto di partenza di una discussione sui limiti dell'algoritmo.

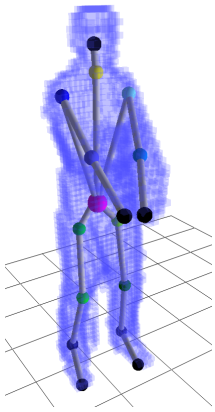
Apri

Nella sequenza `Apri` il movimento è limitato agli arti superiori: le gambe e la testa sono essenzialmente fermi in tutta la sequenza. Alcuni fotogrammi significativi sono riportati in Figura 6.2. Verso la metà della sequenza le due mani entrano quasi in contatto, introducendo degli artefatti nella regione di spazio compresa fra le due braccia e creando nel voxelset punti di fusione alle estremità delle braccia. I voxel spuri possono causare confusione nell'algoritmo di inseguimento e portare a un'errata identificazione della posizione degli arti. Le braccia inoltre iniziano il movimento vicino al corpo, e quindi l'algoritmo deve correttamente interpretare la deformazione lungo i fianchi dei voxelset durante i primi fotogrammi come il movimento delle due braccia. Data la scelta della posa iniziale, questo è un problema comune a tutte le sequenze elaborate.

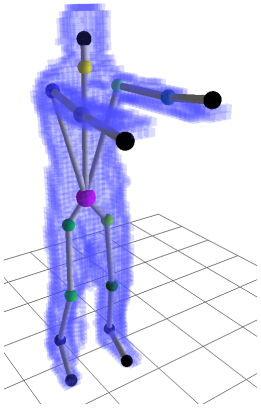
Dal primo grafico in Figura 6.2 osserviamo che l'errore è relativamente contenuto, fra i 3 e i 4 cm per voxel, con l'eccezione dei primi fotogrammi della sequenza. La tendenza ad avere un errore maggiore nei primi fotogrammi è una caratteristica comune a tutte le sequenze, ed è dovuta all'imperfetta corrispondenza fra la posa iniziale reale e la posa usata per l'inizializzazione. L'errore scende man mano che ICP migliora la stima della posa nei primi istanti del movimento, durante i quali l'attore rimane fermo. Dal secondo grafico vediamo che gli errori di convergenza per ogni parte del corpo diminuiscono rapidamente durante le prime 5 iterazioni ed entro le 10 iterazioni



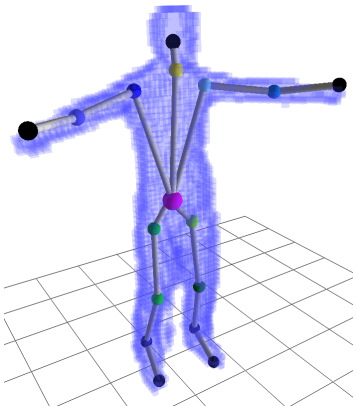
Fotogramma 9



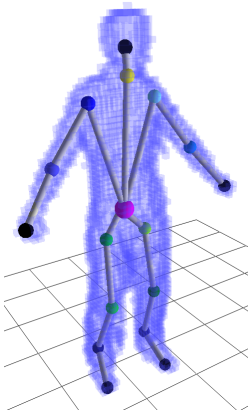
Fotogramma 21



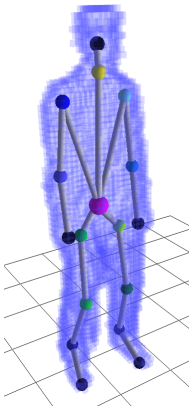
Fotogramma 31



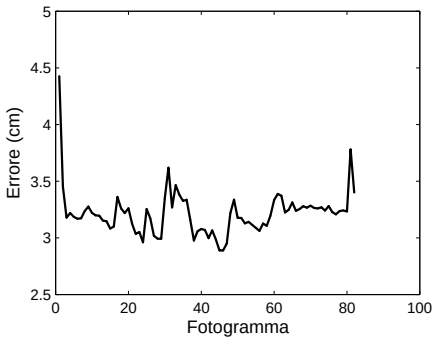
Fotogramma 42



Fotogramma 56



Fotogramma 66



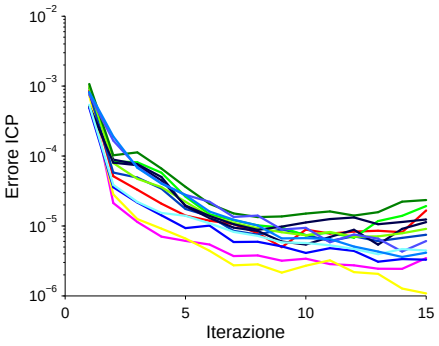


Figura 6.2: **Sequenza Apri.** *Nella figura sono riportati alcuni fotogrammi significativi della sequenza* `Apri-Fede-001` *e due grafici, che riportano l'andamento dell'errore di inseguimento su tutta la sequenza e l'errore di convergenza per ogni parte del corpo e per iterazione. La convenzione dei colori è la stessa usata nel Capitolo 5.*

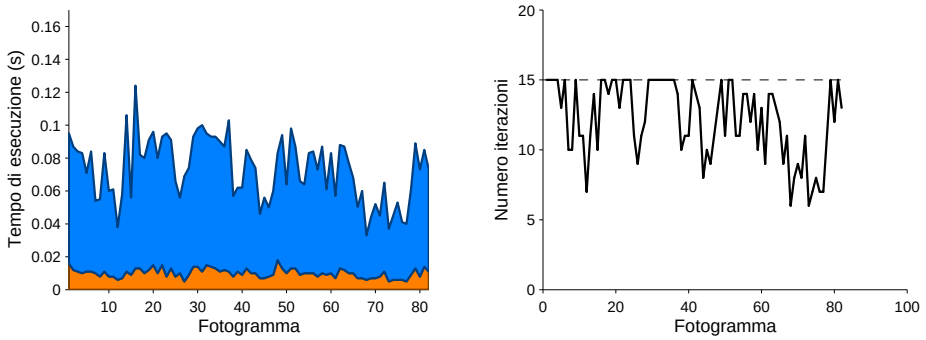


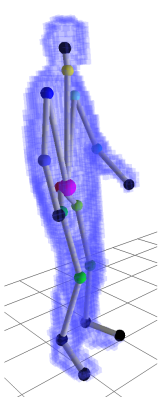
Figura 6.3: **Prestazioni della sequenza Apri.** *Grafici dei tempi di esecuzione dell'algoritmo per ogni fotogramma e il numero di iterazioni corrispondenti. L'area blu nel primo grafico è il tempo impiegato dalla procedura di classificazione dei voxel, l'area arancione il tempo di tutte le altre operazioni.*

si stabilizzano in un intervallo fra 10^{-5} e 10^{-4}. I valori sono di poco superiori alla soglia ε_{Thr}, e quindi l'algoritmo può raggiungere la convergenza per ogni parte del corpo.

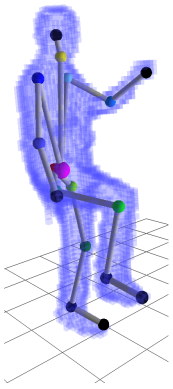
La vicinanza della soglia agli errori di convergenza finali è visibile anche dal grafico a destra della Figura 6.3, dove si può vedere che in molti punti della sequenza viene raggiunto il limite massimo di iterazioni (linea tratteggiata nel grafico). Il tempo di esecuzione dell'algoritmo (grafico a sinistra) ha una forte dipendenza dal numero di iterazioni, come è evidente dalla correlazione fra i due grafici, e quindi la scelta della soglia ha un'influenza critica sulle prestazioni.

Marcia

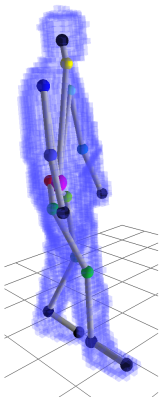
Nella sequenza `Marcia` le braccia e le gambe sono piegate ritmicamente fino a formare un angolo retto, nel tipico movimento stilizzato di una marcia. Il movimento presenta due possibili fonti di errore per l'algoritmo di inseguimento. Un angolo ridotto fra le due sezioni di un arto possono confondere l'algoritmo di classificazione e portare a configurazioni nelle quali una sezione "invade" un'altra, compromettendo la qualità della posa. Il problema è esacerbato dalla sensibilità di ICP ai minimi locali, che porta a bloccare una parte del corpo in una configurazione non ottimale anche quando l'errore è molto alto. Un altro fattore di difficoltà è la vicinanza delle gambe fra loro durante tutto il movimento, che può portare alla confusione fra i due arti, analogamente alla sequenza `Apri` precedentemente discussa.



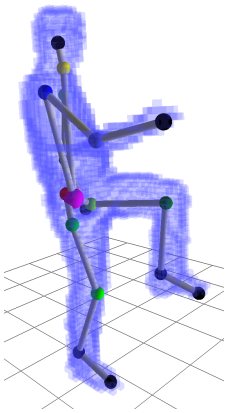
Fotogramma 7



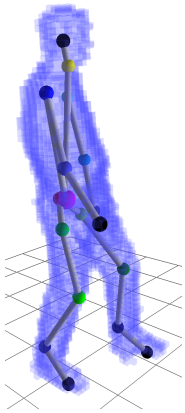
Fotogramma 17



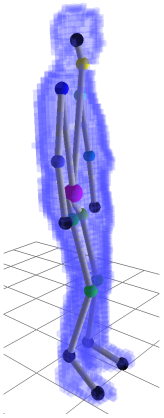
Fotogramma 28



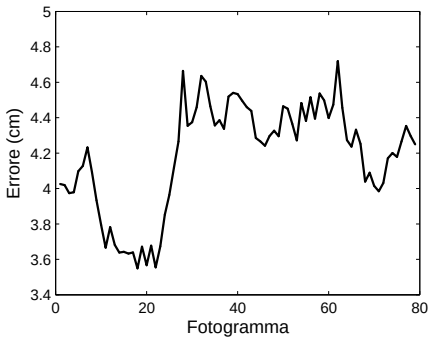
Fotogramma 45



Fotogramma 59



Fotogramma 71



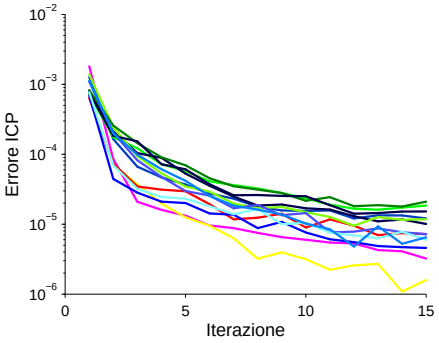


Figura 6.4: **Sequenza Marcia.** *Analisi della sequenza* `Marcia-Massi-002`*, con alcuni fotogrammi significativi e con i grafici dell'errore sulla posa calcolata e del decadimento dell'errore di* ICP *con il numero di iterazioni.*

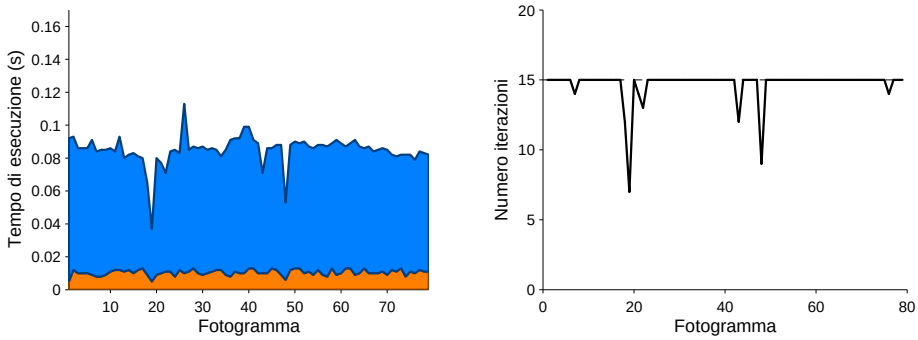


Figura 6.5: **Prestazioni della sequenza Marcia.** *Tempi di esecuzione dell'algoritmo e numero di iterazioni di* ICP *sulla sequenza.*

Come è possibile vedere dalla Figura 6.4, l'errore sulla posa è maggiore rispetto alla sequenza `Apri`, perché in questa sequenza sono coinvolti un maggiore numero di arti ed è quindi più probabile che errori di una parte del corpo siano trasmessi ad altre parti attraverso i vincoli. L'errore di convergenza segue invece un profilo analogo alla precedente sequenza. Come nel caso della sequenza `Apri` le braccia e le gambe (verde e blu rispettivamente nel grafico) tendono ad avere un errore maggiore, perché sono coinvolte più frequentemente in un movimento, mentre testa e torso, essenzialmente ferme, hanno un errore minore. Il maggiore errore sugli arti è riflesso nella velocità di convergenza (Figura 6.5): l'algoritmo ora raggiunge quasi sempre il limite massimo di iterazioni, e questo si riflette su un tempo di esecuzione uniforme su tutta la sequenza.

Crouch

Nella sequenza `Crouch` l'attore si rannicchia per qualche istante e poi si rialza. Il movimento quindi presenta un gran numero di occlusioni e punti di contatto fra gli arti: nel momento centrale della sequenza gli unici particolari riconoscibili del corpo sono la testa e le punte dei piedi, mentre tutte le altre parti del corpo sono fuse in un'unica forma. Gli avambracci sono completamente in contatto con le gambe, le gambe sono ripiegate su sé stesse, e la zona fra il torso e le gambe è nascosta alle telecamere e quindi non visibile nel voxelset. Come nelle precedenti sequenze, i numerosi punti di contatto fra le parti del corpo e le occlusioni possono confondere l'algoritmo di classificazione e portare gradualmente a una posa errata.

Una difficoltà peculiare di questa sequenza è la presenza di molti movimenti che raggiungono i limiti articolari, come le gambe chiuse sul corpo

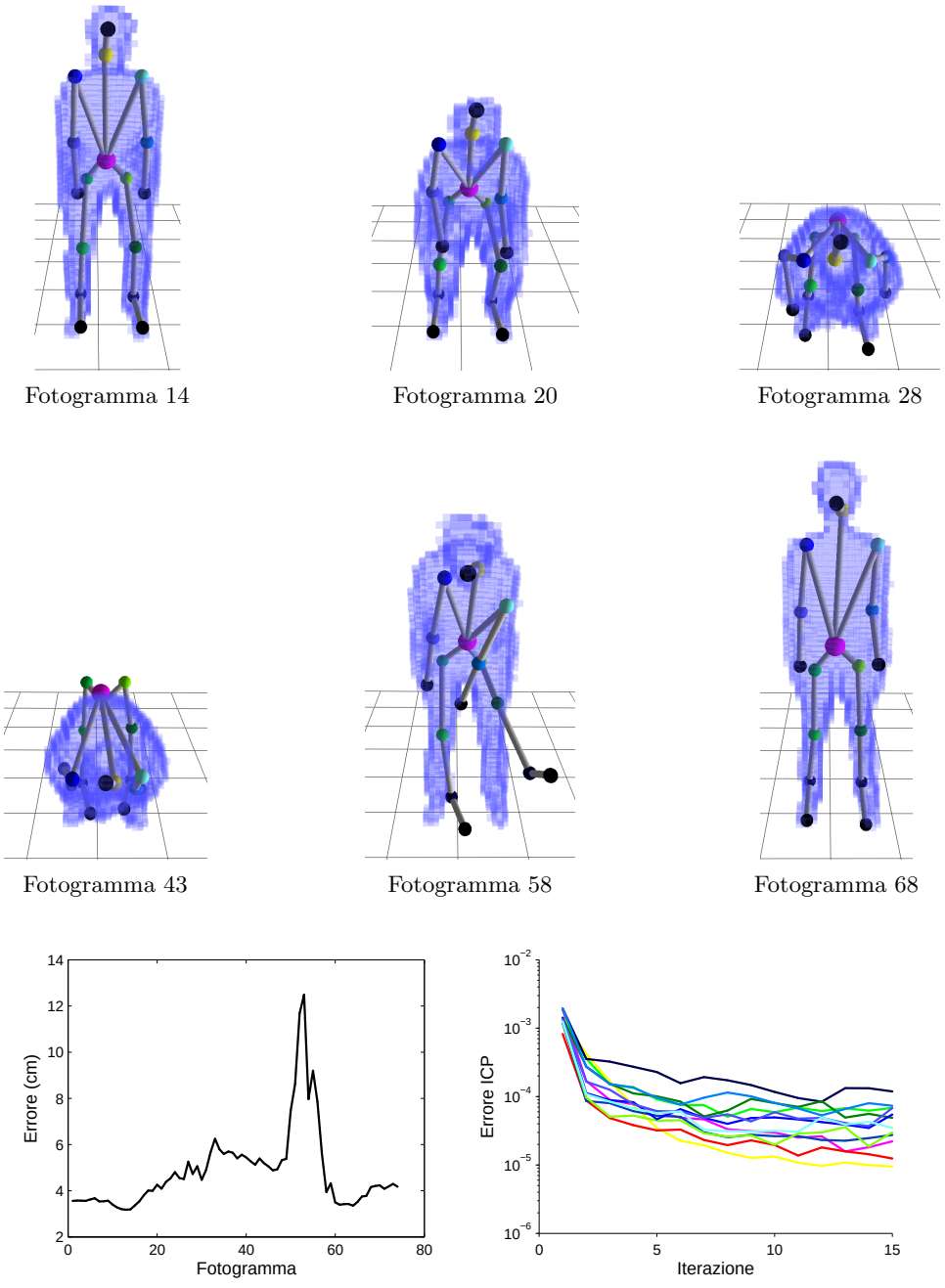


Figura 6.6: **Sequenza Crouch.** *Analisi della sequenza* `Crouch-Vale-001`*, con alcuni fotogrammi significativi e con i grafici dell'errore sulla posa calcolata e del decadimento dell'errore di* ICP *con il numero di iterazioni.*

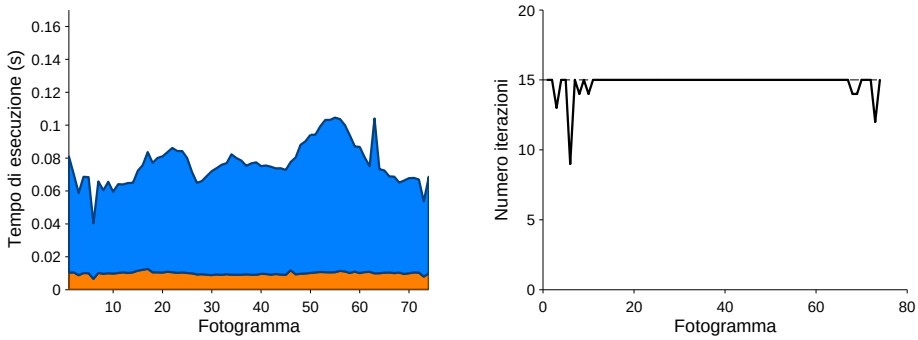
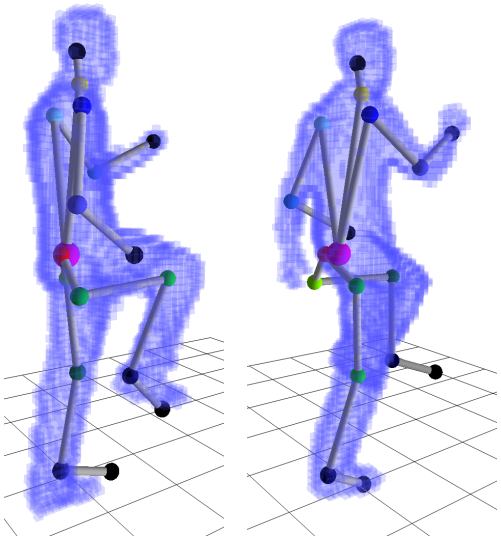


Figura 6.7: **Prestazioni della sequenza Crouch.** *Tempi di esecuzione dell'algoritmo e numero di iterazioni di* ICP *sulla sequenza.*

o la colonna vertebrale completamente ripiegata, che mostrano i limiti della modellazione delle parti del corpo come elementi rigidi. Il problema è particolarmente evidente nel fotogramma 43 in Figura 6.6: benché la testa sia correttamente riconosciuta, il nodo del torso è posizionato all'esterno del voxelset perché il cilindro che rappresenta il torso nel modello non può seguire la deformazione della colonna vertebrale. Le tolleranze scelte per i vincoli sulla posizione non sono sufficienti a compensare queste deformazioni, e tolleranze maggiori danneggerebbero l'accuratezza dell'algoritmo in altre sequenze. Una possibile soluzione a questo problema potrebbe essere l'aggiornamento dinamico delle dimensioni del corpo durante l'inseguimento, come è stato fatto ad esempio in Mikić *et al.*[46].

Gli errori hanno un forte impatto sull'accuratezza della posa, che in questa sequenza è significativamente inferiore rispetto ai casi precedenti. Sorprendentemente, l'errore ha un valore maggiore quando l'attore si sta rialzando, ma questo fenomeno è un artefatto dovuto alla forma del voxelset nella parte centrale del movimento. Infatti, quando il voxelset ha una forma compatta, anche una classificazione sbagliata dei voxel porta a un errore relativamente basso, perché tutti i voxel sono vicini fra loro. Quando l'attore si rialza aumentano le distanze fra gli arti ed è più facile che si manifestino errori, come è visibile nel fotogramma 58.

L'errore di convergenza risente pesantemente della minore accuratezza, e ha valori finali molto maggiori (fra 10^{-4} e 10^{-3}) rispetto alle precedenti sequenze. Invece il tempo di esecuzione, in Figura 6.7, è sensibile solo al numero di iterazioni che, come nel caso della sequenza `Marcia`, raggiunge quasi sempre il limite massimo ed è quindi essenzialmente uguale al caso precedente.



Fotogramma 17 Fotogramma 35

Figura 6.8: **Problema di inizializzazione.** *In figura sono riportati due fotogrammi della sequenza `Marcia-Fede-005`. Nel fotogramma 35 è possibile vedere che il nodo dell'anca sinistra (in verde chiaro) è troppo basso rispetto alla posizione reale dell'articolazione.*

6.1.4 Errori dell'algoritmo

L'azione `Crouch` discussa nella precedente sezione non è l'unica a creare problemi all'algoritmo di inseguimento: altre sequenze, illustrate in questa sezione, portano ad errori significativi in almeno una parte del movimento. Il primo è un caso in cui l'inizializzazione, non essendo sufficientemente accurata, porta a un modello errato per l'attore; gli esempi successivi sono invece casi in cui la procedura di classificazione non riesce ad identificare correttamente le parti all'interno del voxelset.

La sequenza `Marcia-Fede-005` ha prestazioni peggiori rispetto alla sequenza `Marcia-Massi-002` precedentemente discussa, benché i movimenti siano molto simili fra loro. Il motivo è visibile nella Figura 6.8: nel fotogramma 35 il baricentro è all'altezza dell'articolazione dell'anca, e l'articolazione dell'anca sinistra è chiaramente visibile all'esterno del voxelset. A causa di questo errore, le sezioni della gamba, e in particolare il ginocchio, non possono raggiungere la posizione corretta; inoltre alcuni voxel della gamba sinistra sono confusi con i voxel dell'avambraccio sinistro, perché sono più vicini al braccio a causa dell'errato posizionamento, portando anche quella parte del corpo ad una posa errata. L'errato posizionamento dell'anca è visibile anche

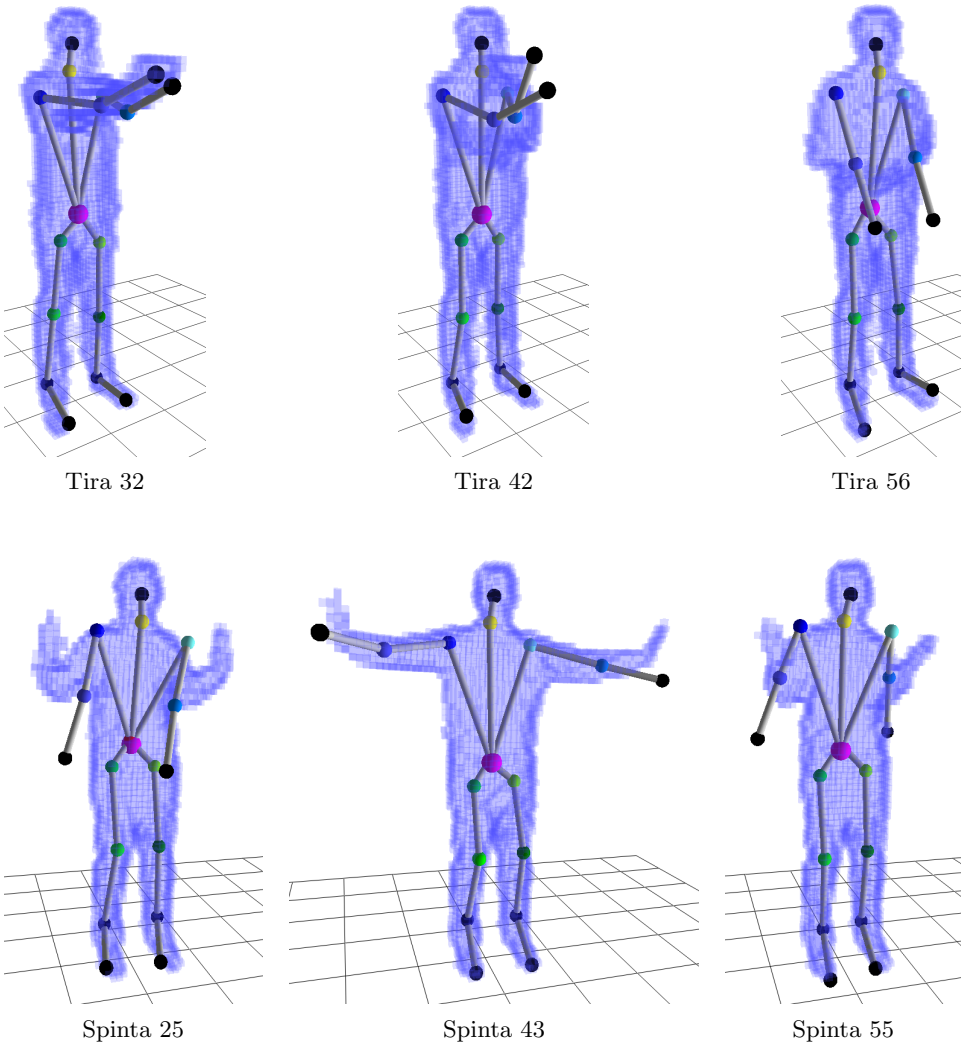


Figura 6.9: **Sequenze problematiche.** *In figura sono riportati alcuni fotogrammi significativi presi dalle sequenze* `Tira-Teo-001` *e* `Spinta-Gabbo-001` *nella parte del movimento più difficile per l'algoritmo.*

nel fotogramma 17, dove però non si traduce in errori di inseguimento.

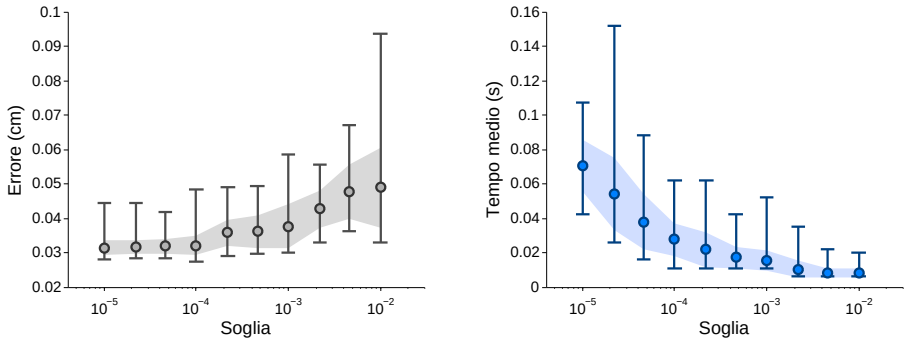
Il problema è causato dalla procedura di inizializzazione. La lunghezza del torso e delle gambe dipendono dal fattore di scala in direzione z, e quindi il loro rapporto è fisso per qualsiasi modello. Tuttavia gli attori `Fede` e `Massi`, pur avendo la stessa altezza, hanno una diversa lunghezza delle gambe e quindi un modello adatto ad un attore porta inevitabilmente ad errori per l'altro. Per evitare questo problema è necessaria una procedura di inizializzazione più sofisticata, capace di estrarre le dimensioni di ogni parte del corpo direttamente dal voxelset, oppure una procedura di aggiornamento delle dimensioni come proposto nella Sezione 6.1.3.

Le azioni `Spinta` e `Tira` portano ad errori per tutte le ripetizioni e per tutti gli attori. Alcuni fotogrammi per le due azioni sono riportati nella Figura 6.9. Nell'azione `Spinta` l'attore solleva le braccia vicino al corpo, le allontana e infine le riporta accanto al corpo. L'algoritmo ha difficoltà a seguire il braccio nella prima parte del movimento, quando le braccia si confondono con il tronco e solo una protuberanza ne segnala posizione e forma. Quando le braccia sono completamente estese, l'algoritmo è in grado di recuperare una posa approssimativamente corretta, ma perde ancora accuratezza quando le braccia si avvicinano al corpo.

Nell'azione `Tira` l'attore piega le braccia sul petto e poi le riporta nella posizione di riposo. L'algoritmo è in grado di calcolare correttamente la posa nella prima parte del movimento, ma analogamente al caso precedente compie degli errori quando le braccia si mantengono in contatto con il corpo. In entrambe le azioni le braccia, come nel caso dell'azione `Crouch`, sono appoggiate al corpo per una parte del movimento e rendono difficile alla procedura di classificazione distinguere fra i voxel delle braccia e del torso. Le informazioni sulla posizione delle braccia sono inoltre date prevalentemente dalla curvatura della superficie, che non è presa in considerazione dall'algoritmo. Un modo per superare questo limite potrebbe essere l'introduzione in ICP di una funzione errore che integra le differenze fra le normali della superficie del voxelset e del modello[59].

6.2 Analisi comparative

In questa sezione analizziamo il comportamento dell'algoritmo di inseguimento al variare dei principali parametri. Nella prima sottosezione controlliamo l'accuratezza e la velocità di esecuzione al variare dei due principali parametri di ICP, N_{max} e ε_{Thr}. Nella seconda sottosezione valutiamo le possibili strategie per accelerare la classificazione dei voxel, passo che domina



(a) (b)

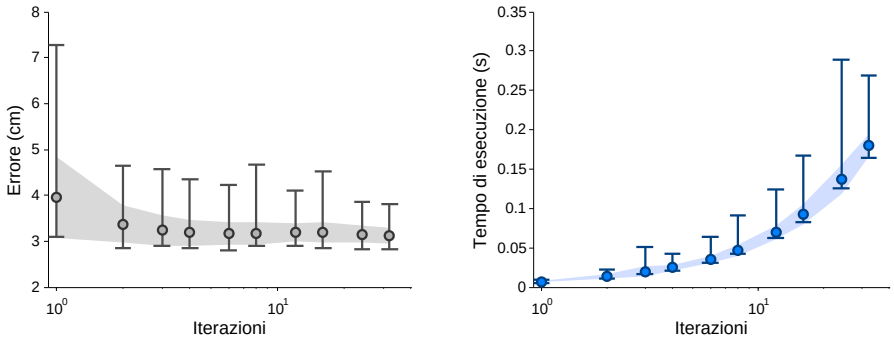
Figura 6.10: **Errore e velocità al variare della soglia.** *In figura sono riportati i diagrammi dell'errore medio* **(a)** *e dei tempi medi* **(b)** *per la sequenza* `Apri-Fede-001` *al variare della soglia di* ICP ε_{Thr}. *Per ogni valore della soglia sono riportati anche i valori massimi e minimi (delimitatori orizzontali) e la varianza (area colorata) sulla sequenza.*

il carico computazionale dell'algoritmo. Nella terza sottosezione osserviamo l'influenza dei vincoli del modello sulla qualità dell'inseguimento. Infine nell'ultima sezione controlliamo l'effetto dei parametri dei particle filter sulla posa calcolata.

6.2.1 Scelta della soglia e del numero di iterazioni

L'algoritmo ICP dipende principalmente da due parametri, la soglia di convergenza ε_{Thr} e il massimo numero di iterazioni consentite N_{max}. La soglia di convergenza, stabilendo quando la posa trovata è soddisfacente, ha un ruolo molto importante nell'accuratezza dell'algoritmo di inseguimento. La scelta della soglia influenza inoltre il numero di iterazioni eseguite dall'algoritmo per ogni fotogramma e, essendo il tempo di esecuzione di un'iterazione all'incirca costante, anche la velocità dell'algoritmo.

Nella Figura 6.10a abbiamo riportato l'andamento dell'errore medio di inseguimento per la sequenza `Apri-Fede-001`, già esaminata nelle precedenti sezioni. Ulteriori informazioni sull'andamento dell'errore nella sequenza sono fornite dai valori estremi e dalla varianza, riportati sul grafico. Nel grafico è visibile un graduale degrado delle prestazioni per soglie superiori a 10^{-4}, che peggiora ulteriormente superata la soglia 10^{-3}, mentre per soglie inferiori a 10^{-4} i risultati sono essenzialmente indipendenti dalla soglia. Il massimo errore per la soglia 10^{-2} è vicino ai 10 cm e segnala che il risultato dell'algoritmo inizia a deviare significativamente dalla posa corretta.



(a) (b)

Figura 6.11: **Errore e velocità al variare del numero di iterazioni.** *In figura sono riportati i diagrammi dell'errore medio* **(a)** *e dei tempi medi* **(b)** *per la sequenza* `Apri-Fede-001` *al variare del numero di iterazioni.*

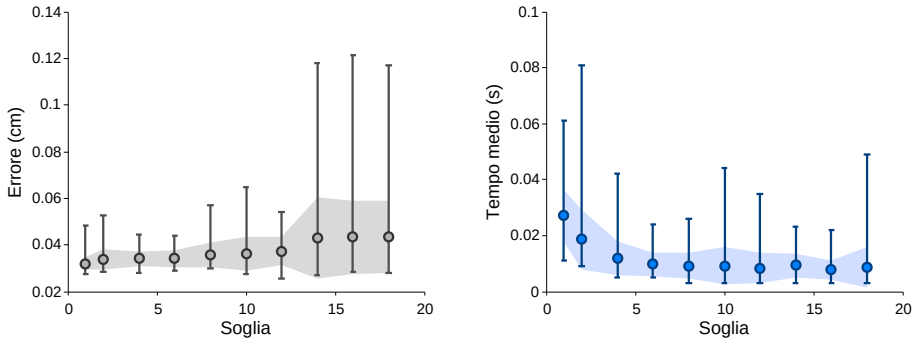
Il valore della soglia, avendo un valore vicino alla differenza del quadrato degli errori e_{t-1} e e_t delle ultime due iterazioni, può essere riscritto nel seguente modo:

$$\varepsilon_{Thr} \approx |e_{t-1}^2 - e_t^2| \approx 2 \cdot e_t \cdot |e_{t-1} - e_t| \tag{6.2}$$

e quindi dalla soglia e dall'errore sul fotogramma è possibile estrarre una stima dell'ultima correzione apportata da ICP sulla posa. Per il valore di soglia 10^{-3}, che segna l'inizio di un degrado significativo delle prestazioni, la correzione ha valori nell'ordine dei centimetri, e indica che ICP non ha la possibilità di raggiungere la convergenza. L'insensibilità per valori molto piccoli della soglia suggerisce invece che in quel caso l'errore di convergenza è dominato dal rumore e dallo scarto fra la forma del modello e la forma reale della persona e che quindi una accuratezza maggiore è raggiungibile solo aumentando la qualità del modello o la precisione dei dati acquisiti.

La velocità di esecuzione, in Figura 6.10b, segue prevedibilmente un andamento inverso all'errore e suggerisce un compromesso fra qualità dell'inseguimento e velocità di elaborazione, costruito sulla scelta della soglia. Una soglia più bassa sarà scelta quando è importante avere una buona qualità della posa, mentre una soglia più alta sarà scelta nei casi in cui è importante la reattività del sistema. La soglia può essere anche decisa di fotogramma in fotogramma, usando un metodo adattativo che potrebbe costituire un'estensione a questo lavoro di tesi.

Per valutare l'impatto del numero di iterazioni indipendentemente dall'effetto della soglia, abbiamo eseguito un test con soglia nulla, in modo che



(a) (b)

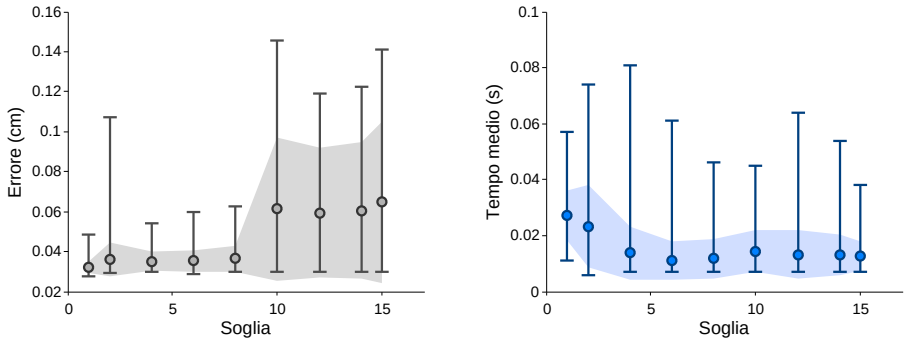
Figura 6.12: **Errore e velocità con sottocampionamento del voxelset.** *In figura sono riportati i diagrammi dell'errore medio* **(a)** *e dei tempi medi* **(b)** *per la sequenza* `Apri-Fede-001` *per diversi fattori di sottocampionamento del voxelset. La classificazione è applicata ad ogni iterazione. Il valore di soglia di* ICP *è* $\varepsilon = 10^{-4}$.

l'algoritmo possa fermarsi solo quando ha raggiunto il massimo numero di iterazioni. I risultati sono riportati in Figura 6.11. Dalla Figura 6.11a è chiaro che con un numero di iterazioni massimo inferiore a 10 è possibile ottenere l'errore massimo raggiungibile. Questo risultato è in accordo con l'osservazione fatta nella Sezione 6.1.3 che l'algoritmo raggiunge essenzialmente la convergenza entro una decina di iterazioni. Aumentare ulteriormente il numero di iterazioni porta solo a un aumento del tempo di elaborazione, che cresce linearmente con il numero di iterazioni, come è evidente dalla Figura 6.11b.

6.2.2 Sottocampionamento dei dati

Abbiamo visto nella Sezione 6.1.3 che la maggior parte del tempo di elaborazione dell'algoritmo è occupato dalla procedura di classificazione dei voxel, benché sia un'operazione accessoria alla più importante procedura di allineamento del modello. É quindi giustificata la ricerca di metodi per accelerarne l'esecuzione senza comprometterne significativamente l'accuratezza.

La classificazione è effettuata ad ogni iterazione e, per ogni iterazione, è effettuata su tutti i voxel: questo suggerisce di accelerare l'elaborazione applicando la classificazione solo a un sottoinsieme di voxel, oppure solo a un sottoinsieme di iterazioni. Per semplicità abbiamo scelto di considerare solo il caso di campionamento uniforme degli insiemi, dove quindi un fattore di campionamento n corrisponde alla scelta di un elemento ogni n.



(a) (b)

Figura 6.13: **Errore e velocità con sottocampionamento delle iterazioni.** *In figura sono riportati i diagrammi dell'errore medio* **(a)** *e dei tempi medi* **(b)** *per la sequenza* `Apri-Fede-001` *per diversi fattori di campionamento delle iterazioni. Non è stato invece eseguito nessun campionamento del voxelset. Il valore di soglia di* ICP *è* $\varepsilon = 10^{-4}$.

In Figura 6.12a è mostrato il comportamento dell'algoritmo per diversi fattori di campionamento del voxelset, da $n = 1$ (nessun campionamento) a $n = 18$. Le prestazioni iniziano a degradare significativamente solo per valori di n superiori a 10. La velocità di esecuzione (Figura 6.12b) ha un miglioramento fino a $n = 5$ e poi tende ad appiattirsi su un valore di 0,01 s. Questo comportamento indica che, con la riduzione del tempo di classificazione dei voxel, altre parti dell'algoritmo con tempo di esecuzione fisso iniziano ad occupare una proporzione maggiore del tempo totale. I particle filter, la verifica dei vincoli e il calcolo delle corrispondenze artificiali sono tutte elaborazioni che hanno un tempo di esecuzione dipendente solo dal modello del corpo.

In Figura 6.13a abbiamo eseguito la stessa analisi quando la classificazione è applicata solo ad alcune iterazioni, con fattori che variano da $n = 1$ a $n = 15$. Per $n = 15$ la classificazione è eseguita solo una volta, all'inizio dell'elaborazione del fotogramma, e non viene più cambiata. Rispetto al caso precedente, le prestazioni degradano più rapidamente con l'aumentare del fattore di campionamento. L'algoritmo si dimostra quindi più sensibile alla mancanza di aggiornamento della classificazione tra un'iterazione e l'altra che alla disponibilità di un numero minore di dati. Già per $n = 2$ è osservabile un errore significativo, e superata la soglia di $n = 10$ le prestazioni sono invariabilmente negative. Da questi dati è evidente che la procedura di classificazione non è in grado di assegnare correttamente i voxel in un singolo

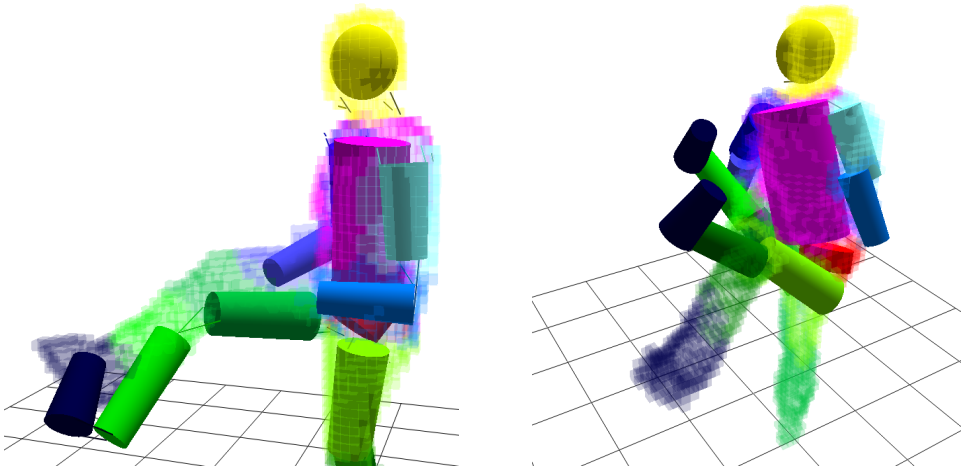


Figura 6.14: **Vincoli e corrispondenze.** *A sinistra, fotogramma 17 della sequenza* `Calcio-Vale-001` *elaborata senza vincoli su posizione e orientamento, e con cedevolezza fissata al valore* 1*. A destra, fotogramma 21 della stessa sequenza, con vincoli ma senza corrispondenze artificiali.*

passo, ma ha la necessità di correggere iterativamente la stima in collaborazione con ICP. La perdita di accuratezza ha un visibile impatto anche sul tempo di esecuzione (Figura 6.13b) che, risentendo della peggiore qualità di convergenza, ha una maggiore variabilità rispetto al caso precedente.

6.2.3 Importanza dei vincoli sugli angoli

Nell'algoritmo abbiamo aggiunto dei vincoli espliciti sulla posizione e sull'orientamento (vedi Sezione 5.1.5) con lo scopo di superare i limiti delle corrispondenze artificiali (Sezione 4.3.1), che sono sufficienti a vincolare l'algoritmo in presenza di movimenti localizzati e lenti, ma che in altri casi mostrano rapidamente i loro limiti.

Nella Figura 6.14 è mostrato il comportamento dell'algoritmo per la sequenza `Calcio`, nella quale un solo arto, la gamba destra, compie un largo movimento. La cedevolezza è stata fissata a un valore superiore al valore predefinito per compensare l'assenza di vincoli espliciti. Il piede ha chiaramente una posa innaturale, mentre le due braccia, avendo più libertà di movimento, hanno coinvolto erroneamente dei voxel del torso e della gamba destra. Le corrispondenze artificiali, mostrate come linee nella figura, si allungano e si accorciano durante l'inseguimento, giustificando il nome "bande elastiche" dato da Knoop[37].

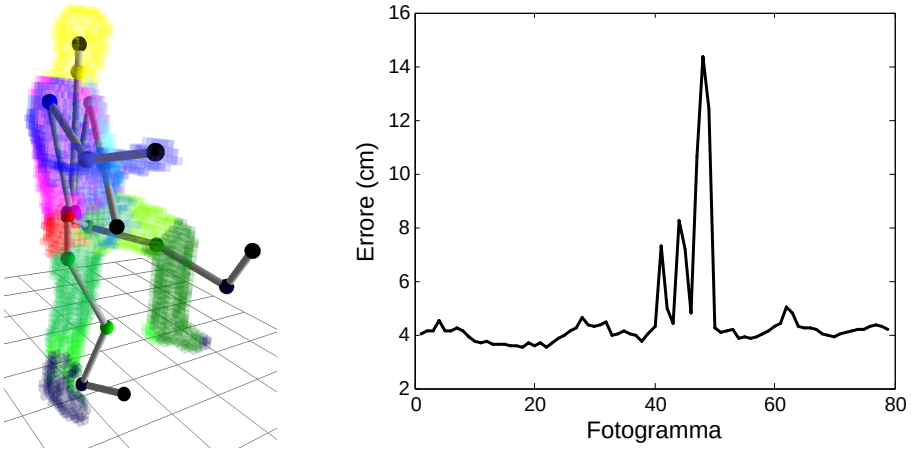


Figura 6.15: **Prestazioni senza particle filter.** *In figura è riportato il fotogramma 49 della sequenza `Marcia-Massi-002` elaborata senza particle filter e il diagramma di errore sulla sequenza. É possibile vedere l'aumento drammatico dell'errore di inseguimento fra il quarantesimo e il cinquantesimo fotogramma.*

Dei due tipi di vincoli espliciti, i vincoli di posizione sono quelli che hanno l'impatto maggiore sull'accuratezza. Non solo impediscono alle parti del corpo di andare alla deriva durante l'inseguimento, ma una regione di ammissibilità troppo ristretta può influenzare negativamente l'accuratezza dell'orientamento a causa della continua proiezione della posa trovata nella regione di ammissibilità. Le dimensioni della regione devono essere quindi scelte con cura.

Le corrispondenze artificiali sono tuttavia vitali per la buona convergenza dell'algoritmo. Come è chiaro dalla Figura 6.14, l'applicazione diretta di ICP e la successiva proiezione della soluzione nello spazio ammesso dai vincoli porta rapidamente il modello ad allontanarsi dalla posa corretta, con effetti disastrosi sull'accuratezza: nell'esempio riportato, già dopo venti fotogrammi il risultato dell'algoritmo è completamente sbagliato, pur rispettando i vincoli articolari.

6.2.4 Ruolo dei particle filter

I particle filter, come già accennato nella Sezione 5.3, non sono usati per calcolare la posizione finale, ma per introdurre perturbazioni stocastiche nell'algoritmo di inseguimento e facilitarne la convergenza al minimo globale. In particolare i particle filter controbilanciano due fattori che influenzano negativamente la qualità della posa finale: la tendenza di ICP a bloccarsi in

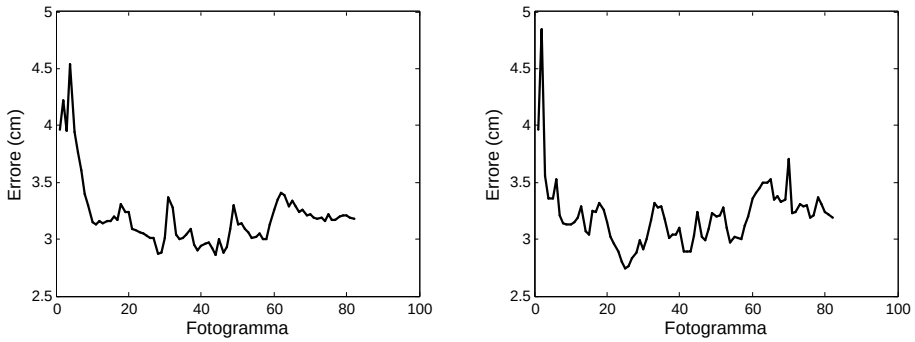


Figura 6.16: **Sequenza Apri con e senza particle filter.** *In figura è riportato il diagramma dell'errore per la sequenza* `Apri-Fede-001` *senza particle filter (a sinistra) e con particle filter (a destra). La soglia di* ICP *è stata fissata a* $\varepsilon = 10^{-4}$.

un minimo locale, e la necessità di orientare correttamente le articolazioni a cerniera (vedi Sezione 5.2.3), perché un'orientamento errato porta a vincolare gli angoli di una rotazione in realtà ammissibile.

Avendo i particle filter un ruolo preventivo, ci possiamo aspettare che la loro importanza dipenderà dal tipo di sequenza. Nelle sequenze caratterizzate da movimenti fluidi e con escursioni angolari limitate avranno un impatto marginale, mentre nelle sequenze con grandi escursioni nei movimenti e ambiguità nel voxelset porteranno un contributo significativo alla qualità della soluzione finale. Questo è esattamente quello che abbiamo osservato.

In Figura 6.15 vediamo il comportamento dell'algoritmo senza particle filter per la sequenza `Marcia-Massi-002`, già esaminata nella Sezione 6.1.3. Fra il quarantesimo e il cinquantesimo fotogramma l'algoritmo perde la posizione del braccio destro e delle due gambe, come è possibile vedere nell'immagine a sinistra. L'errore è chiaramente visibile nella coppia di picchi presente nel diagramma a destra. L'effetto in sequenze meno dinamiche è invece trascurabile, come è evidente dalla coppia di grafici in Figura 6.16.

Nelle sequenze in cui i particle filter si sono dimostrati utili, abbiamo osservato che è sufficiente un numero di particelle limitato, inferiore al centinaio per ogni parte del corpo, per avere prestazioni sufficientemente accurate. Come è possibile vedere in Figura 6.17a, quando il numero di particelle è molto basso è presente un errore significativo perché la stima è troppo dipendente dalle fluttuazioni statistiche dei campioni; quando il numero di particelle è molto alto non ci sono invece significativi miglioramenti della stima. L'impatto sul tempo di esecuzione, che non abbiamo riportato, inizia ad essere significativo solo quando il numero delle particelle è estremamen-

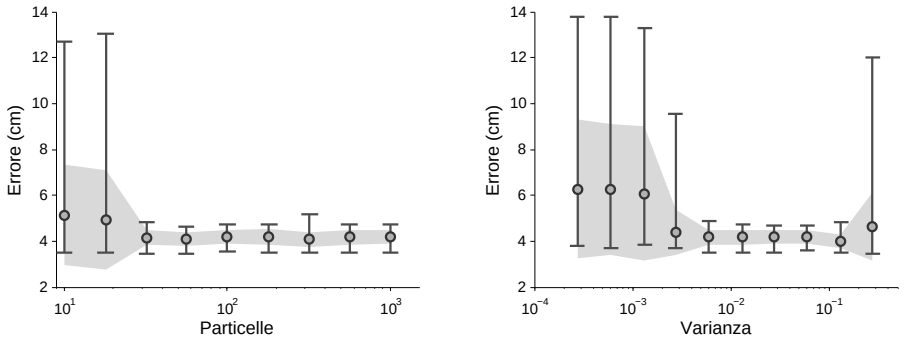


Figura 6.17: **Errori per i particle filter.** *In figura sono riportati i diagrammi dell'errore medio per la sequenza* `Marcia-Massi-002` *in funzione del numero di particelle* **(a)** *e della varianza dei particle filter* **(b)**. *Il rapporto fra le varianze della posizione è fisso e pari al valore predefinito.*

te alto, dell'ordine delle migliaia per ogni parte del corpo. Il valore della varianza dei particle filter ha invece un impatto più significativo sull'accuratezza. In Figura 6.17b si può osservare che valori troppo bassi o troppo alti hanno un effetto negativo, per motivi differenti. Quando la varianza è troppo bassa il rumore stocastico è di fatto assente e l'algoritmo ha un comportamento simile al caso senza particle filter, presentando quindi errori dovuti a minimi locali o al cattivo allineamento delle articolazioni. Quando la varianza è molto alta invece aumenta la variabilità della stima, a parità di numero di particelle, ed aumenta quindi la probabilità che sia selezionata una stima troppo lontana dalla posa corretta. Il problema può essere compensato da un numero maggiore di particelle, al prezzo di un maggiore carico computazionale.

6.3 Considerazioni sulla velocità

Nella scelta dell'algoritmo di questa tesi abbiamo posto particolare attenzione all'impatto sulle prestazioni e, come è possibile vedere dai risultati delle precedenti sezioni, l'algoritmo è relativamente veloce: un fotogramma è elaborato mediamente in un tempo di 0,1 secondi, che corrisponde a una frequenza di 10 fotogrammi al secondo e quindi vicina al tempo reale. Come abbiamo discusso in precedenza (Sezioni 6.2.1 e 6.2.2) queste prestazioni possono essere significativamente migliorate nei casi in cui è tollerabile un decremento della qualità dell'inseguimento.

Il programma sviluppato non fa uso di concorrenza e quindi solo metà del processore di test (AMD M320) è usato durante l'esecuzione. La quasi totalità del carico computazione dell'algoritmo è concentrato nella procedura di classificazione e nell'esecuzione di ICP e in particolare in operazioni che coinvolgono il calcolo di distanze su liste di punti, procedure facilmente ed efficientemente parallelizzabili. Le prestazioni dell'algoritmo possono essere quindi raddoppiate a parità di accuratezza e di processore usato e, tenuto in considerazione che il processore usato è di fascia bassa, probabilmente raggiungono il tempo reale nella maggior parte dei processori recenti.

Le operazioni vettoriali coinvolte nel calcolo delle distanze sono ottime candidate per l'implementazione in processori paralleli e in particolare nei processori grafici (GPU) moderni. In molte applicazioni un'implementazione parallela di un algoritmo sulle GPU più potenti può guadagnare fino ad un ordine di grandezza in velocità rispetto all'implementazione sui migliori processori classici (CPU). Un guadagno di queste proporzioni per il nostro algoritmo porterebbe la velocità finale nell'ordine di almeno 200 fotogrammi al secondo, più che sufficienti a liberare tempo di processore per altre elaborazioni e favorire quindi l'integrazione in sistemi interattivi più complessi.

Capitolo 7

Conclusioni e direzioni future di ricerca

Abbiamo sviluppato un algoritmo di cattura del movimento umano che, a partire da un volume calcolato elaborando le immagini di otto telecamere, cerca di ricostruire il movimento del corpo con l'ausilio di un modello. A questo scopo abbiamo scelto l'algoritmo Iterative Closest Point, applicato ad ogni parte del modello articolato e opportunamente modificato per rispettare vincoli antropometrici sui movimenti. Per migliorare la qualità finale del risultato abbiamo aggiunto all'algoritmo una fase di filtraggio realizzata con particle filter.

Rispetto a quanto già fatto in letteratura, questo lavoro indaga sull'interazione fra i vincoli creati da corrispondenze artificiali in ICP e vincoli di posizione e orientamento esplicitamente definiti e ispirati a valori antropometrici. In aggiunta, il lavoro valuta l'impatto dell'introduzione di particle filter in un sistema di questo tipo.

L'algoritmo sviluppato ha mostrato una buona accuratezza di inseguimento in gran parte delle sequenze esaminate e una capacità di recupero, anche nel caso di deviazioni significative dalla posa corretta. L'accuratezza non è sufficiente per applicazioni di precisione nel campo biomedico e cinematografico, ma lo è sufficiente per applicazioni nel campo della sorveglianza e di interazione uomo–macchina. L'algoritmo si è dimostrato inoltre molto veloce, con tempi inferiori al decimo di secondo, e con ampi margini di miglioramento.

L'attuale lavoro può essere esteso in molte direzioni. Una migliore procedura di inizializzazione è sicuramente l'estensione più importante. In questa tesi è stata scelto un metodo molto semplice per estrarre le dimensioni del

modello e la posa iniziale perché il punto centrale del lavoro è l'algoritmo di inseguimento. Un metodo di inizializzazione flessibile permetterebbe di iniziare l'inseguimento da una posa arbitraria, oppure di recuperare la posa quando l'errore di inseguimento è eccessivo. L'algoritmo di inizializzazione non dovrebbe essere necessariamente veloce, perché l'algoritmo di inseguimento sarebbe usato nella maggior parte dei casi. Una possibile scelta è data dal metodo sviluppato da un precedente tesista[29] e pubblicato dal gruppo ISPG[44], nel quale sono estratti alcuni punti di interesse da un voxelset usando funzioni che valutano la distanza fra voxel, il raggio di curvatura della superficie e così via. A partire da questi punti sono successivamente calcolati la posizione degli arti e le lunghezze delle parti del corpo dell'attore.

Fra i possibili algoritmi di inizializzazione sono particolarmente interessanti i metodi che non assumono un modello predefinito. La rappresentazione usata in questo lavoro può accomodare qualsiasi modello con una struttura ad albero, fra cui quelli generati dinamicamente da un algoritmo. Schaefer e Yuksel[60] hanno ottenuto risultati promettenti nell'estrazione di modello scheletrico da superfici tridimensionali. Il metodo di Pekelny e Gotsman[54] cerca gli elementi di una superficie approssimativamente rigidi usando una funzione di errore analoga all'errore in ICP ed è particolarmente adatto ad essere integrato nel nostro algoritmo.

Inoltre di un modello accurato della superficie potrebbe beneficiare non solo l'inizializzazione, ma anche l'inseguimento. Molti risultati, in particolare Balan *et al.*[3] e Corazza *et al.*[15], mostrano che l'uso di un modello fedele del corpo umano permette di ottenere risultati molto accurati anche a partire da dati ambigui. La rappresentazione della forma reale del corpo, invece di una sua approssimazione, potrebbe rendere più preciso l'algoritmo e aprirebbe la possibilità di identificare la persona che sta svolgendo il movimento.

Il nostro metodo usa solo le informazioni di occupazione del volume. Il lavoro di Knoop *et al.*[38] mostra che l'integrazione di informazioni di colore può aiutare a risolvere situazioni ambigue. In particolare la posizione delle mani e del viso, a causa delle loro caratteristiche facilmente identificabili, può evitare errori di inseguimento quando le braccia sono molto vicine al corpo.

Per gestire modelli articolati in ICP abbiamo introdotto dei vincoli. Non è l'unica soluzione possibile: Pellegrini *et al.*[55] hanno esplorato un metodo di minimizzazione specifico per modelli articolati che, pur richiedendo un numero di operazioni maggiore rispetto all'algoritmo originale, mostra risultati promettenti.

Infine, la rappresentazione della posa in termini di parametri di un mo-

dello non dipende da variabili come corporatura o orientamento del corpo, ed è più robusta al rumore di ricostruzione del voxelset. Le informazioni ottenute dal nostro algoritmo sono quindi un buon candidato per un successiva identificazione dei movimenti in classi di gesti, come dimostra ad esempio il lavoro di Wang *et al.*.

Appendice A

Integrazioni teoriche

A.1 Metodi di ricampionamento

Nell'algoritmo SIR dei particle filter (vedi Sezione 4.4.3) è stato data per assodata la capacità di ricampionare un insieme di campioni dato. In questa sezione si descrive in breve il problema e alcune delle tecniche più note per affrontarlo. La trattazione è basata principalmente su [32] e [22].

Dati M campioni $\{x^i\}$ di probabilità nota $p(x^i) = w_i$ definita da M pesi, si vogliono estrarre N campioni con la stessa distribuzione. La distribuzione reale $p(x)$ non è generalmente disponibile e quindi i campioni saranno estratti dalla distribuzione approssimata:

$$p_M(x) = \frac{1}{M} \sum_{i=1}^{M} w_i \delta(x - x^i) \approx p(x) \tag{A.1}$$

I pesi d'ora in avanti saranno sempre considerati normalizzati, e si considera disponibile un metodo per generare campioni da una distribuzione uniforme su $[0, 1)$.

Ricampionamento multinomiale

Dati N valori generati da una distribuzione uniforme su $[0, 1)$: $u_j \sim U[0, 1)$, si estraggono N campioni:

$$x^j = F^{-1}(u_j) \tag{A.2}$$

dove F è la distribuzione cumulativa costruita a partire dai pesi w_i:

$$F(x^i) = \sum_{k=1}^{i} w_k \tag{A.3}$$

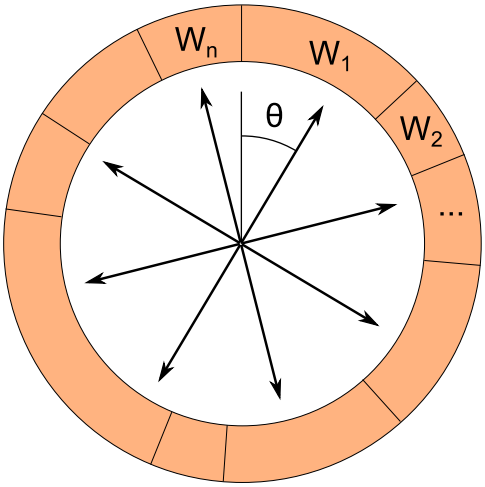


Figura A.1: **La "roulette" del ricampionamento.** *I pesi w_i normalizzati rappresentano porzioni della circonferenza. Il campionamento sistematico è applicato scegliendo N direzioni equidistanti fra loro (che corrispondono ai vertici di un poligono regolare a N lati) e ruotando la struttura di un angolo casuale θ uguale a $\frac{\tilde{u}}{N}$.*

e F^{-1} è l'inversa generalizzata. Poiché F è una funzione a gradini costante fra x^i e x^{i+1}, si ottiene che $F^{-1}(u)$ con $u \in [0,1)$ è x^i tale che $F(x^{i-1}) \leq u < F(x^i)$.

Il calcolo di N campioni da valori non ordinati può richiedere nel caso peggiore $O(MN)$ operazioni. Si possono ordinare i valori prima di procedere al calcolo dei campioni, oppure si può usare il seguente metodo:

$$u_N = \tilde{u}_N^{\frac{1}{N}} \qquad u_k = u_{k+1}\tilde{u}_k^{\frac{1}{k}} \qquad \tilde{u}_k \sim U[0,1) \tag{A.4}$$

che genera N valori preordinati estratti da una distribuzione uniforme. Con questo accorgimento il costo totale di ricampionamento è $O(N)$.

Ricampionamento residuo

Per ogni peso w_i si estraggono $n_i^{'} = \lfloor Mw^i \rfloor$ campioni, dove l'operatore $\lfloor \blacksquare \rfloor$ indica la parte intera dell'argomento. Si definiscono dei nuovi pesi $w_i^{'} = Nw_i - n_i^{'}$ e si estraggono $r = N - \sum n_i^{'}$ campioni usando un altro schema di ricampionamento.

Ricampionamento stratificato

Sono generate N variabili casuali uniformi ordinate $\{u_i\}$, una per ogni intervallo $[\frac{i-1}{N}, \frac{i}{N}]$:

$$u_i = \frac{(i-1) + \tilde{u}_i}{N} \qquad \text{con} \qquad \tilde{u}_i \sim U[0,1) \tag{A.5}$$

e i valori così ottenuti sono usati per estrarre campioni con il ricampionamento multinomiale.

Ricampionamento sistematico

Applica lo stesso procedimento del ricampionamento stratificato, usando una singola estrazione per tutti gli intervalli:

$$u_i = \frac{(i-1) + \tilde{u}}{N} \qquad \text{con} \qquad \tilde{u} \sim U[0,1) \tag{A.6}$$

In Figura A.1 è rappresentata una semplice interpretazione geometrica del campionamento sistematico.

Appendice B

Note di implementazione

B.1 Elaborazione dei dati

B.1.1 Origine dei dati

I dati usati in questa tesi sono stati prodotti dal lavoro di Miorelli[47] e sono stati scelti per la migliore qualità della segmentazione e uniformità delle acquisizioni. I dati sono stati codificati come vettori a tre indici in Matlab e salvati nel formato di memorizzazione '`.mat`', un file per fotogramma e una cartella per sequenza. Il formato di Matlab riduce drasticamente lo spazio occupato da un tipico voxelset, da 64 MB a circa 10 kB. Questo materiale costituisce il dataset standard sul quale è stato costruito il lavoro di tesi.

B.1.2 Formato dei file

Per semplificare il processo di lettura dei dati all'esterno di Matlab e per ridurre il numero di file necessari è stato definito un formato di file specifico, di estensione `.seq`, ispirandosi al formato `.zdat` usato in lavori di tesi precedenti, in particolare Gritti[29].

I file codificano solo voxelset di occupazione, e quindi ogni voxel può assumere solo i valori di 0 (non occupato) o 1 occupato. Ogni voxelset è trattato come un vettore a una dimensione, ottenuto concatenando tutte le colonne lungo la coordinata x prima a coordinata y constante, e poi a z costante. I dati sono convertiti in una codifica *Run Length Encoding* (RLE), dove le informazioni di occupazione sono rappresentate da un flusso di 3 byte (h_0, l_0, l_1) così organizzati:

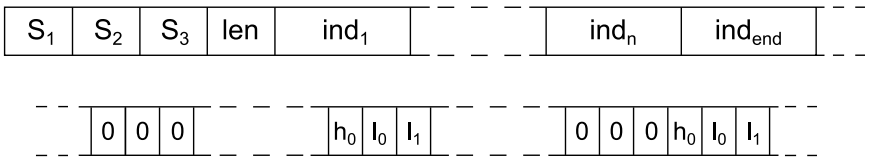


Figura B.1: **Struttura del file sequenza.** *In figura è riportata la struttura dell'header (sopra) e del flusso di dati (sotto) nei file di tipo* `.seq`*. La lunghezza dei blocchi è proporzionale alla dimensione in bit dei dati contenuti.*

- h_0 e l_0 codificano il numero di zeri consecutivi, secondo la formula: $\#0 = 256h_0 + l_0$
- l_1 codifica il numero di uno consecutivi

Ogni tripletta codifica una sequenza di 0 seguita da una sequenza di 1. Se il numero di 0 supera il massimo possibile (65535) viene memorizzato uno 0 nel numero di 1 e il resto della sequenza è memorizzata nelle triplette successive. Analogamente, se il numero di 1 supera 255 il numero di 0 nella tripletta successiva è fissato a 0.

Formato zdat

Nel formato `.zdat` ogni voxelset è salvato in un file separato di estensione `.vox.zdat`. La posizione del file nella sequenza è indicato da una tripletta di cifre alla file del nome del file, e premette di codificare sequenze fino a 999 frame. Le dimensioni x, y e z del voxelset, necessarie per recuperare il volume da un vettore lineare, sono comuni a tutta la sequenza e salvate come stringhe in un file `.vox` con numero di sequenza 001.

Formato .seq

Nel formato di sequenza `.seq` un *header* contiene dati comuni alla sequenza: le dimensioni del voxelset, la lunghezza della sequenza e una serie di indici alla posizione dei frame nel file. I dati dei frame sono salvati nel corpo del file usando la codifica binaria descritta nella precedente sezione. Uno schema dell'header e del corpo dei file è mostrato in Figura B.1. I dati memorizzati sono i seguenti:

- s_1, s_2 e s_3 (16 bit ciascuno) sono le dimensioni del voxelset lungo le direzioni x, y e z rispettivamente
- *len* (16 bit) è la lunghezza della sequenza in frame

- $ind_1, \ldots, ind_n, ind_{end}$ (32 bit ciascuno) sono $len + 1$ indici a posizioni nel file. I primi n rappresentano l'inizio di ciascun frame all'interno del file, mentre ind_{end} è la fine del file, ed è necessario per calcolare la dimensione dell'ultimo frame.

Ogni frame nel corpo del file inizia con una tripletta di byte nulli, poiché il caso $h_0 = l_0 = l_1 = 0$ non può mai accadere e può essere usato come sequenza di controllo. Anche se con questo accorgimento gli indici non sono strettamente necessari, sono stati comunque mantenuti per facilitare l'accesso casuale a singoli frame.

B.2 Struttura del programma

B.2.1 Voxelset

Nella maggior parte dei casi i voxelset sono usati essenzialmente come liste di coordinate. I dati sono quindi rappresentati internamente come un vettore di voxel, e in particolare con un `vector<Voxel>` in `C++`. Tuttavia alcune operazioni sul voxelset, come il calcolo della superficie, richiede informazioni sull'adiacenza dei voxel che non sono facilmente accessibili da un vettore. Il voxelset è quindi rappresentato anche come matrice di occupazione dei voxel. La matrice di occupazione consuma una quantità di memoria molto superiore rispetto alla lista di voxel (100 volte nei casi tipici) e poiché è necessaria solo in casi particolari, abbiamo deciso di calcolarla solo quando è necessario, mediante la tecnica della *lazy evaluation*. Un voxelset inizialmente è rappresentato solo dalla lista di voxel. Quando per la prima volta è richiesto l'accesso di un voxel in una particolare posizione viene generata la matrice di occupazione, e da quel momento la lista e la matrice sono aggiornate in sincrono. La lazy evaluation è eseguita dal seguente codice:

```
float VoxelSet::value(int i, int j, int k) {
  if (voxel_set_ == NULL)
    generate_matrix();
  return voxel_set_[i][j][k];
}
```

`voxel_set_` è la matrice di occupazione, e quando non è stata allocata `generate_matrix` si occupa di crearla, con le opportune dimensioni, a partire dalla lista di voxel.

B.2.2 Classi

L'applicazione sviluppata per questa tesi è stato progettata ponendo particolare enfasi sul principio di *Separation of Concerns* (*SoC*) dell'ingegneria

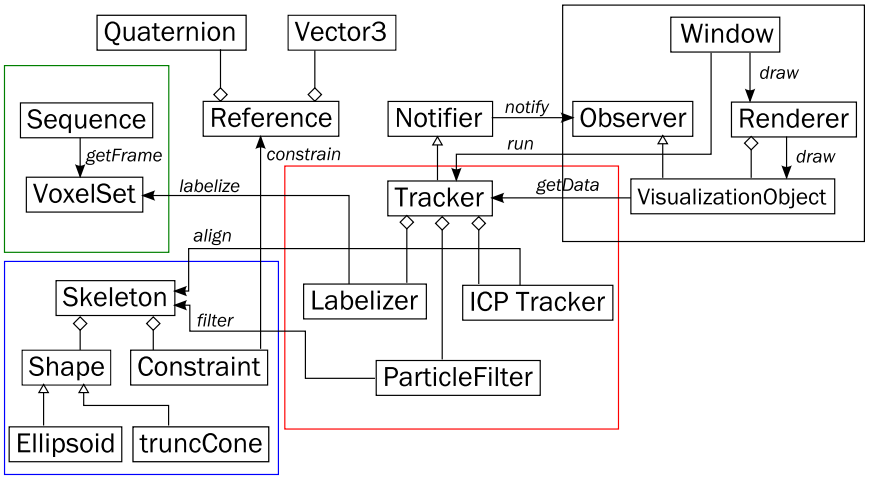


Figura B.2: **Organizzazione del programma in classi.** *Nella figura sono riportate le principali classi dell'applicazione. La freccia bianca indica ereditarietà e il rombo indica composizione, in accordo con la convenzione in Gamma* et al.*[25], mentre le frecce nere mostrano alcune delle azioni principali. I riquadri evidenziano le principali componenti dell'applicazione: lettura dei dati (verde), modello (blu), algoritmo di inseguimento (rosso) e visualizzazione (nero).* `Vector3`*,* `Quaternion` *e* `Reference` *sono classi geometriche di utilità generale.*

del software. Il principio afferma che ogni funzionalità o comportamento specifico deve essere assegnato ad un singolo elemento del programma, che nel caso dei linguaggi ad oggetti corrisponde a una classe. Nella nostra applicazione il principio è stato applicato sia a livello di organizzazione generale, sia a livello delle singole classi.

A livello generale le classi più importanti sono divise in quattro gruppi coesi (vedi Figura B.2): lettura dei dati, che si occupa di leggere le informazioni di sequenza da formato binario e convertirle in un voxelset; modello, che si occupa di gestire il modello di scheletro e i vincoli; algoritmo di inseguimento; parte di visualizzazione dei dati. La parte di lettura dei dati e della gestione del modello sono largamente indipendenti dai restanti gruppi e fra di loro; l'algoritmo di inseguimento dipende da entrambe ma è indipendente dalla modalità di visualizzazione, che può quindi essere cambiata a piacimento. Questo disaccoppiamento si è dimostrato molto comodo nell'implementazione della modalità di test usata per generare i grafici, dove la visualizzazione è stata semplicemente eliminata per ragioni prestazionali.

L'assegnazione di ogni responsabilità nell'applicazione a una singola classe ha l'ulteriore vantaggio di ridurre sensibilmente la duplicazione di codice, di per sé un fattore che aumenta la probabilità di errori e riduce la

manutenibilità del codice sorgente[45].

Progettazione della visualizzazione

La separazione fra le classi di visualizzazione e l'algoritmo di inseguimento è stata particolarmente impegnativa, perché il flusso principale del programma è contenuto all'interno della routine di visualizzazione. Per mostrare in tempo reale il comportamento dell'algoritmo di inseguimento e avere quindi un riscontro visivo siamo costretti a mescolare il codice dell'algoritmo e il codice di visualizzazione, in aperta violazione del principio *SoC* precedentemente introdotto. La soluzione al dilemma che abbiamo scelto è un'architettura di tipo *Model–View–Controller*[25].

Il modello (*Model*) è una rappresentazione in termini di classi del problema in esame, e nella nostra applicazione il ruolo è coperto principalmente dalla classe `Tracker`. Le modalità di visualizzazione (*View*) riguardano la parte di codice devoluta a creare un riscontro visivo delle operazioni, e nell'applicazione sono gli oggetti di interfaccia `VisualizationObject`. Il controllore (*Controller*) infine si occupa di accettare comandi dall'utente e decidere quali operazioni svolgere sul modello; questo ruolo è coperto dalla classe `Window`.

Il disaccoppiamento fra modello e visualizzazione avviene mediante le interfacce `Notifier` e `Observer`, che rappresentano una realizzazione del design pattern *Observer*[25]. Gli oggetti di tipo `Notifier` segnalano, con diversi tipi di evento, che lo stato del modello è cambiato e, in risposta, gli oggetti di tipo `Observer` possono recuperare i dati necessari ad aggiornare la visualizzazione. Il vantaggio di questa configurazione è che il modello è all'oscuro dell'esistenza delle visualizzazioni specifiche e quindi non deve essere modificato quando nuove modalità sono aggiunte o le esistenti modificate.

Modelli geometrici

Le forme geometriche sono rappresentate dalla classe astratta `Shape` e dalle sue due specializzazioni, `Ellipsoid` e `truncCone`. Le classi derivate implementano due metodi principali, `closest` e `distance`, che calcolano rispettivamente il punto più vicino sulla superficie e la distanza di un punto dalla superficie. I metodi sono necessari all'algoritmo di inseguimento, in particolare alla parte di classificazione dei voxel (Sezione 5.2.2) e di Iterative Closest Point (Sezione 5.2.3). Le classi si occupano inoltre di fornire attraverso la stessa interfaccia una triangolazione della superficie, in modo da semplificare la fase di visualizzazione.

Bibliografia

[1] J.K. Aggarwal and Q. Cai. Human motion analysis: a review. In *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*, pages 90–102, San Juan, Puerto Rico, 1997. IEEE Comput. Soc.

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, 2005.

[3] Alexandru O. Balan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed Human Shape and Pose from Images. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[4] Paul J. Besl and Neil D. McKay. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[5] Jonathan Blow. Inverse Kinematics with Quaternion Joint Limits. *The Inner Product*, 2002.

[6] A.F. Bobick and J.W. Davis. The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257–267, August 2002.

[7] Gary R. Bradski and James W. Davis. Motion segmentation and pose recognition with motion history gradients. *Machine Vision and Applications*, 13(3):174–184, July 2002.

[8] Christoph Bregler and Jitendra Malik. Tracking People with Twists and Exponential Maps. In *Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[9] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[10] Simone Calderara, Rita Cucchiara, and Andrea Prati. Action Signature: A Novel Holistic Representation for Action Recognition. In *Advanced Video and Signal Based Surveillance 2008, IEEE Fifth International Conference on*, pages 121–128, September 2008.

[11] Mauro Capetti. *Robust markerless human tracking from volumetric data using particle filtering.* Master thesis, Politecnico di Milano, 2007.

[12] Paolo Caravello. *Una rete di telecamere per l'analisi dei movimenti: calibrazione e primi risultati.* Master thesis, Politecnico di Milano, 2005.

[13] Zhe Chen. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. Technical report, 2003.

[14] Chi Wei Chu, Odest Chadwicke Jenkins, and Maja J Mataric. Markerless Kinematic Model and Motion Capture from Volume Sequences. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:475–482, 2003.

[15] Stefano Corazza, Emiliano Gambaretto, Lars Mundermann, and Thomas P. Andriacchi. Automatic Generation of a Subject Specific Model for Accurate Markerless Motion Capture and Biomechanical Applications. *IEEE transactions on bio-medical engineering*, 57(4):806–812, November 2008.

[16] Stefano Corazza, Lars Mündermann, Emiliano Gambaretto, Giancarlo Ferrigno, and Thomas P. Andriacchi. Markerless Motion Capture through Visual Hull, Articulated ICP and Subject Specific Model Generation. *International Journal of Computer Vision*, 87(1-2):156–169, September 2009.

[17] Edilson de Aguiar, Christian Theobalt, M. Magnor, Holger Theisel, and Hans-Peter Seidel. M3: marker-free model reconstruction and motion tracking from 3D voxel data. In *12th Pacific Conference on Computer Graphics and Applications*, pages 101–110. IEEE, 2004.

[18] David Demirdjian and Trevor Darrell. 3-D Articulated Pose Tracking for Untethered Deictic Reference. *Multimodal Interfaces, IEEE International Conference on*, page 267, 2002.

[19] David Demirdjian, Teresa Ko, and Trevor Darrell. Untethered gesture acquisition and recognition for virtual world manipulation. *Virtual Reality*, 8(4):222–230, July 2005.

[20] Jonathan Deutscher, Andrew Blake, and Ian Reid. Articulated body motion capture by annealed particle filtering. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, pages 126–133, 2000.

[21] Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann, 1st edition, May 2007.

[22] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005*, pages 64–69. Ieee, 2005.

[23] David Eberly. Distance from a Point to an Ellipse in 2D, 2004.

[24] JS Franco and Edmond Boyer. Exact polyhedral visual hulls. *British Machine Vision Conference*, 1:329–338, 2003.

[25] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1st edition, November 1994.

[26] D Gavrila. The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, 73(1):82–98, January 1999.

[27] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[28] Claus Gramkow. On Averaging Rotations. *International Journal of Computer Vision*, 42(1):7–16, April 2001.

[29] Tommaso Gritti. *Markerless Motion Capture System*. Master thesis, Politecnico di Milano, 2005.

[30] Andrew J. Hanson. *Visualizing Quaternions (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann, 1st edition, January 2006.

[31] Lorna Herda, Raquel Urtasun, and Pascal Fua. Hierarchical Implicit Surface Joint Limits to Constrain Video-Based Motion Capture. In *Computer Vision - ECCV 2004*, pages 405–418, 2004.

[32] Jeroen D. Hol, Thomas B. Schon, and Fredrik Gustafsson. On Resampling Algorithms for Particle Filters. *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pages 79–82, September 2006.

[33] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Optical Society of America Journal A*, 4:629–642, 1987.

[34] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[35] R. Kehl, M. Bray, and L. Van Gool. Full body tracking from multiple views using stochastic sampling. In *Computer Vision and Pattern Recognition, 2005. IEEE Computer Society Conference on*, volume 2, pages 129–136, July 2005.

[36] David Kelly. *Texture Encoded Realistic Joint Limits.* Master thesis, 2008.

[37] Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Modeling Joint Constraints for an Articulated 3D Human Body Model with Artificial Correspondences in ICP. In *International Conference on Humanoid Robots*, pages 74–79, 2005.

[38] Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Fusion of 2d and 3d sensor data for articulated body tracking. *Robotics and Autonomous Systems*, 57(3):321–329, 2009.

[39] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality.* Princeton University Press, 2002.

[40] Kiriakos N. Kutulakos and Steven M. Seitz. A Theory of Shape by Space Carving. *International Journal of Computer Vision*, 38(3):199–218, July 2000.

[41] A. Laurentini. The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(2):150–162, February 1994.

[42] Jingen Liu and Mubarak Shah. Learning human actions via information maximization. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[43] M. Losh, S. Gartner, Steffen Knoop, S. R. Schmidt-Rohr, and Rüdiger Dillmann. A human body model initialization approach made real-time capable through heuristic constraints. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6, 2009.

[44] Marco Marcon, Massimiliano Pierobon, Augusto Sarti, and Stefano Tubaro. 3D Markerless Human Limb Localization through Robust Energy Minimization. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008.

[45] Steve McConnell. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press, 2004.

[46] Ivana Mikic, Mohan Trivedi, Edward Hunter, and Pamela Cosman. Articulated Body Posture Estimation from Multi-Camera Voxel Data. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 455–460. IEEE Computer Society, 2001.

[47] Federico Miorelli. *Analisi volumetrica mediante armoniche sferiche applicata al riconoscimento di azioni da viste multiple*. Master thesis, Politecnico di Milano, 2008.

[48] Thomas B. Moeslund. A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding*, 81(3):231–268, March 2001.

[49] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, November 2006.

[50] Lars Mündermann, Stefano Corazza, and Thomas P. Andriacchi. Accurately measuring human movement using articulated ICP with soft-joint constraints and a repository of articulated models. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, June 2007.

[51] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[52] Nasa. Man-systems integration standards. In *NASA-STD-3000, Volume I*. Section 3, 1995.

[53] Juan Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. *International Journal of Computer Vision*, 79(3):299–318, September 2008.

[54] Yuri Pekelny and Craig Gotsman. Articulated Object Reconstruction and Markerless Motion Capture from Depth Video. *Computer Graphics Forum*, 27(2):399–408, April 2008.

[55] Stefano Pellegrini, Konrad Schindler, and Daniele Nardi. A Generalisation of the ICP Algorithm for Articulated Bodies. In *Proceedings of the British Machine Vision Conference*. Citeseer, 2008.

[56] X Pennec. Computing the mean of geometric features Application to the mean rotation. Technical report, 1998.

[57] Michael K. Pitt and Neil Shephard. Filtering via Simulation : Auxiliary Particle Filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[58] Ralf Plänkers and Pascal Fua. Articulated Soft Objects for Multiview Shape and Motion Capture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1182–1187, 2003.

[59] Szymon Rusinkiewicz and Marc Levoy. Efficient Variants of the ICP Algorithm. In *Third International Conference on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[60] S. Schaefer and C. Yuksel. Example-based skeleton extraction. In Alexander Belyaev and Michael Garland, editors, *Eurographics Symposium on Geometry Processing*, pages 153–162, Barcelona, Spain, 2007. Eurographics Association.

[61] Timo Schairer, Benjamin Huhle, and Wolfgang Strasser. Application of particle filters to vision-based orientation estimation using harmonic analysis. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2556–2561, Anchorage, AK, May 2010.

[62] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-Time Human Pose Recognition in Parts from a Single Depth Image. In *Computer Vision and Pattern Recognition*, June 2011.

[63] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *International Journal of Computer Vision*, 87(1-2):4–27, August 2009.

[64] Leonid Sigal and Michael J. Black. Measure Locally, Reason Globally: Occlusion-sensitive Articulated Pose Estimation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2041–2048. IEEE Computer Society, 2006.

[65] Leonid Sigal and Michael J. Black. Guest Editorial: State of the Art in Image- and Video-Based Human Pose and Motion Estimation. *International Journal of Computer Vision*, 87(1-2):1–3, October 2009.

[66] David A. Simon. *Fast and accurate shape-based registration.* Thesis (phd), Carnegie Mellon University, 1996.

[67] Christian Theobalt, Edilson de Aguiar, Marcus A. Magnor, Holger Theisel, and Hans Peter Seidel. Marker-free kinematic skeleton estimation from sequences of volume data. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 57–64, Hong Kong, 2004. ACM.

[68] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea. Machine Recognition of Human Activities: A Survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1473–1488, September 2008.

[69] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 511–518, Kauai, HI, USA, April 2001. IEEE Comput. Soc.

[70] L Wang. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, March 2003.

[71] Sy Bor Wang, Ariadna Quattoni, Louis Philippe Morency, and David Demirdjian. Hidden Conditional Random Fields for Gesture Recognition. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1521–1527. IEEE Computer Society, 2006.

[72] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2-3):249–257, November 2006.

[73] Bangpeng Yao and Li Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. *Vision and Pattern Recognition (CVPR)*, 2010.