Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Informatica

Trasformazione a posa standard di point cloud di persone per re-identificazione

Candidato:
Alberto Basso

Relatore:
Emanuele Menegatti

Correlatore:
Matteo Munaro

Anno Accademico 2012-2013

Alla mia famiglia, Danilo, Giuseppina e Andrea,
per avermi sostenuto e permesso di giungere fino a questo traguardo.

Ad Alessandro, Luca, Stefano ed Umberto,
per questi cinque impegnativi anni passati a studiare assieme.

Alberto Basso

Sommario

Questa tesi si propone di analizzare e implementare tecniche per la creazione di modelli 3D da impiegare per la re-identificazione nell'ambito del tracking di persone. In particolare, viene proposta una tecnica per la trasformazione di point cloud di persone a posa standard in tempo reale, impiegando le informazioni fornite dal tracking dello scheletro, la quale permette di ignorare il problema delle differenti pose che una persona assume. Il problema della re-identificazione viene quindi affrontato con un approccio derivato dalla 3D object recognition, dove si suppone che gli oggetti subiscano solamente trasformazioni rigide. Viene spiegato come utilizzare queste point cloud trasformate per comporre modelli 3D di persone in movimento che possono essere impiegate per la re-identificazione tramite ICP matching con nuove cloud, comparando la forma globale del corpo.

Abstract

This thesis targets the analysis and implementation of techniques for creating people 3D models for re-identification in people tracking. A novel technique is proposed for exploiting skeleton information to transform persons' point clouds to a standard pose in real-time, which allows to disregard the problem of the different poses a person can assume. This way, we can tackle the re-identification problem with an approach borrowed from the 3D object recognition domain, where objects are supposed to undergo rigid transformations only. We explain how to use these transformed point clouds for composing 3D models of moving people which can be used for re-identification by means of an ICP matching with new test clouds, that is by comparing their global body shape.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Questa tesi si propone di analizzare e implementare tecniche per la creazione di modelli 3D da impiegare per la re-identificazione nell'ambito del tracking di persone.
Il problema della re-identificazione di persone consiste nel riconoscere se una persona attualmente visibile da una telecamera è già stata osservata in precedenza. Questo problema rappresenta un'attività fondamentale in molte applicazioni pratiche: controllo degli accessi, video sorveglianza, tracking di persone sono solamente alcuni esempi.
In questa tesi viene presentata una nuova tecnica per la re-identificazione basata sulla comparazione di features *soft biometric* non collaborative per la re-identificazione a lungo termine. In particolare viene comparata la *global body shape*, ovvero la forma globale del corpo della persona. La tecnica proposta deriva dalla *3D object recognition* ed impiega i dati di profondità ottenuti da sensori RGB-D come la *Microsoft Kinect* e la *Asus Xtion PRO*. Dato che la forma del corpo cambia anche a causa delle differenti pose che un soggetto può assumere durante il movimento è stata sviluppata una nuova tecnica che sfrutta le informazioni fornite dagli skeletal tracker Microsoft e OpenNI per trasformare le point cloud di persone a una *posa standard* prima di effettuare la comparazione.

1.1 Stato dell'arte

La maggior parte dei sistemi di identificazione basati sulla visione impiega descrittori *soft biometric*, ovvero un insieme di caratteristiche della persona che forniscono delle informazioni biometriche, ma non sono individualmente sufficienti ad autenticare un soggetto, principalmente a causa della mancanza di carattere distintivo o permanenza nel tempo. Queste informazioni biometriche si definiscono *collaborative* se richiedono l'intervento dell'utente per essere raccolte, come ad esempio il riconoscimento dell'iride o l'analisi delle impronte digitali, oppure

non-collaborative come il riconoscimento facciale [18]. Un altro tipo di metodo basato sulla visione che è stato impiegato per la re-identificazione di persone è il riconoscimento dell'andatura, come proposto in [16, 12].
Gli approcci applicati alla re-identificazione impiegano solitamente dati provenienti da telecamere 2D e si basano su tecniche che prendono in considerazione solamente l'aspetto delle persone [17, 4, 3]. Queste tecniche assumono che l'abbigliamento degli individui rimanga invariato durante il periodo di osservazione, vincolando l'applicazione di questi metodi in un intervallo temporale limitato. Inoltre cambiamenti di illuminazione e punto di vista aggiungono ulteriore complessità al problema.
I sensori RGB-D economici come la *Microsoft Kinect* e la *Asus Xtion PRO*, entrambe prodotte impiegando tecniche sviluppate da Primesense [5], permettono di acquisire informazioni di profondità in maniera veloce e conveniente. Questo ha spinto i ricercatori a impiegare sensori RGB-D in differenti campi applicativi, come ad esempio per la stima della posa [13] e il riconoscimento di oggetti [9]. Nonostante questo impulso alla ricerca tali sensori sono apparsi sul mercato solo recentemente per cui la letteratura in questo campo è abbastanza limitata. L'adozione di informazioni sulla struttura 3D del corpo umano per la re-identificazione è stata introdotta per la prima volta in [1] dove viene proposto un modello 3D rigido, simile ad un sarcofago, per il riconoscimento di pedoni. Questo modello viene adattato in base all'altezza del soggetto, ma non viene considerata nessuna informazione riguardante la forma del corpo e viene utilizzato solamente per mappare l'apparenza della persona. Similarmente in [11] viene proposto un descrittore cilindrico tridimensionale creato a partire dai dati forniti da una depth camera. Tale descrittore si basa su una griglia cilindrica 3D che memorizza variazioni di colore con angolo e altezza. In [2] vengono proposti due differenti insiemi di feature calcolate solamente a partire dai dati di profondità. Il primo comprende feature estratte dallo scheletro, in particolare la lunghezza degli arti e l'altezza del soggetto, mentre il secondo comprende le distanze geodesiche calcolate sulla superficie della forma del corpo tra delle coppie di giunti (per esempio dalla spalla all'anca destra), che sono indici della curvatura e per approssimazione delle dimensioni di specifiche regioni del corpo. Infine, il software *Kinect Identity* [8], eseguito dalla Kinect per Xbox360, usa tre tipi di descrittori, l'altezza del soggetto, un descrittore facciale e un descrittore di colore dei vestiti dell'utente per re-identificare il giocatore durante la sessione di gioco. In questo caso, però il problema è semplificato in quanto questa applicazione copre intervalli di tempo limitati ed un numero esiguo di soggetti.

1.2 Struttura della tesi

Nel Capitolo 2 viene descritto il framework ROS (*Robot Operating System*) e le principali librerie impiegate nel lavoro di tesi: PCL (*Point Cloud Library*) e OpenCV (*Open Computer Vision*).
Nel Capitolo 3 vengono presentate le principali caratteristiche del sensore RGB-D Microsoft Kinect e introdotti il Microsoft Kinect for Windows SDK e il framework OpenNI.
Nel Capitolo 4 vengono presentati e comparati i principali software per il tracking dello scheletro compatibili con il sensore Kinect.
Nel Capitolo 5 viene presentato l'algoritmo di trasformazione a posa standard di point cloud di persone e viene descritta una tecnica che sfrutta questo algoritmo per la creazione di modelli 3D di persone.
Nel Capitolo 6 viene presentato un approccio alla re-identificazione di persone basato sulle caratteristiche di forma del corpo che impiega un matching ICP di point cloud di persone in posa standard con un modello 3D. Vengono inoltre illustrati i risultati ottenuti su due differenti dataset di persone.
Il Capitolo 7 è dedicato alle conclusioni tratte da questo lavoro, in Appendice A sono riportati i dettagli riguardati l'implementazione delle tecniche sviluppate e in Appendice B viene descritta la configurazione dell'ambiente software impiegato nello sviluppo.

Capitolo 2

ROS - Robot Operating System

ROS[1] (*Robot Operating System*) è un framework per lo sviluppo di software per robot, che fornisce funzionalità simili ad un sistema operativo su piattaforme software eterogenee. Il progetto ROS è nato nel 2007 presso lo *Stanford Artificial Intelligence Laboratory* per il controllo del robot *STAIR*. Dal 2008 lo sviluppo è portato avanti da Willow Garage[2], un istituto di ricerca per la robotica, formato da più di venti istituzioni che collaborano mediante un modello di sviluppo federale. ROS fornisce alcuni dei servizi standard per un sistema operativo come astrazione hardware, controllo di periferiche a basso livello, gestione di pacchetti. ROS è basato su un'architettura a grafo dove l'elaborazione ha luogo in nodi che possono pubblicare e ricevere messaggi di controllo, stato o provenienti da sensori e attuatori. Il framework attualmente supporta pienamente solamente Ubuntu Linux mentre il funzionamento su altre distribuzioni o piattaforme software (Windows, MacOS) non è garantito in quanto il supporto è ancora in fase sperimentale. ROS viene rilasciato con licenza BSD, gratuito per uso commerciale e accademico.

2.1 Architettura a grafo

ROS si basa su un'architettura a grafo, simile ad una rete peer-to-peer nella quale i processi ROS (*nodes*) condividono i dati sotto forma di messaggi. I componenti pricipali di questa architettura sono:

Nodes Un nodo è sostanzialmente un processo che effettua delle elaborazioni, nei nodi vengono eseguite le istruzioni e per questo sono il principale componente del grafo. I nodi comunicano

[1] http://www.ros.org
[2] http://www.willowgarage.com

tra di loro pubblicando **messaggi** su *topic*, *servizi RPC* o il *parameter server*.

Messages Un messaggio ROS è una semplice struttura dati contenente campi di tipo primitivo (integer, floating point, boolean, etc.) oppure array e struct.

Topics Un topic è un bus identificato da un nome mediante il quale i nodi possono scambiare messaggi. I topic implementano una semantica *publish/subscribe* che disaccoppia la produzione dell'informazione dal suo consumo. In generale, i nodi non sono a conoscenza del loro interlocutore, i nodi che sono interessati ai dati di un topic lo sottoscrivono, mentre i nodi che generano i messaggi li pubblicano nel topic in questione. Ogni topic può avere più nodi che pubblicano o sottoscrittori.

Services Il modello publish/subscribe costituisce un metodo di comunicazione molto flessibile, ma il suo paradigma *molti-a-molti* non è appropriato per interazioni di tipo RPC, le quasi sono molto frequenti in un sistema distribuito. Il paradigma RPC *request/reply* viene implementato tramite servizi, i quali sono definiti da un paio di messaggi, uno per la richiesta e uno per la risposta. Ogni servizio viene identificato da un nome, mediante il quale un nodo client può invocare il servizio.

Master Il master fornisce servizi di naming e registrazione per gli altri nodi nel sistema ROS. Tiene traccia di publishers e subscribers ai topic e dei servizi offerti. Il master mette a disposizione degli altri nodi anche il parameter server, un dizionario condiviso utilizzato dai nodi per archiviare e recuperare i parametri in fase di esecuzione.

Bags Le bag sono un formato per il salvataggio e la riproduzione di messaggi ROS. Sono un importante meccanismo per il salvataggio dei dati provenienti ad esempio dai sensori, che possono essere utilizzati in un secondo momento per la creazione di un ambiente di test per gli algoritmi sviluppati.

2.2 PCL - Point Cloud Library

PCL[3] (*Point Cloud Library*) è un progetto open source che contiene numerosi algoritmi allo stato dell'arte per l'elaborazione di immagini 2D/3D e point cloud, i quali includono filtering, feature estimation, surface reconstruction, registrazione, model fitting e segmentatione. Questi algoritmi possono essere usati, per esempio, per filtrare valori anomali da dati rumorosi, effettuare lo stiching di point cloud 3D, segmentare parti rilevanti da una scena, individuare keypoint e calcolare descrittori per riconoscere oggetti nel mondo in base alla loro forma geometrica, creare superfici da point cloud e visualizzarle. PCL è ben integrata in ROS il quale fornisce

[3] http://www.pointclouds.org

anche alcune funzionalità come nodi pronti all'uso. PCL viene rilasciato secondo i termini della licenza BSD, gratuito sia per un utilizzo nell'ambito della ricerca che commerciale. Il progetto viene supportato finanziariamente da grandi aziende quali Open Perception, Willow Garage, NVIDIA, Google, Toyota, Trimble, Urban Robotics, Honda Research Institute, Sandia, Dinast, Optronic, Ocular Robotics, Velodyne, Spectrolab, Fotonic, Leica Geosystems, National Institute of Standards and Technology, Southwest Research Institute, e MKE. PCL è cross-platform ed attualmente è disponibile per sistemi Linux, Windows, MacOS e Android nella versione 1.6. La libreria si compone di diversi moduli:

- **common** contiene le strutture di dati comuni e i metodi usati dalla maggior parte delle librerie PCL. La struttura dei dati di base include la classe Point Cloud e una moltitudine di tipi di punti che sono utilizzati per rappresentare punti semplici, le normali delle superfici, i valori di colore RGB e i descrittori delle features. Contiene inoltre numerose funzione per calcolare distanze/norme, medie e covarianze, conversioni angolari e trasformazioni geometriche.
- **filters** contiene algoritmi di rimozione di outlier e rumore per il filtraggio di point cloud 3D. Contiene inoltre filtri generici usati per estrarre sottoinsiemi di point cloud, per escluderne una parte o effettuare il downsampling.
- **features** contiene le strutture di dati e i meccanismi per stimare *features* 3D a partire dalle point cloud. Le features 3D descrivono pattern geometrici basati sulle informazioni disponibili attorno al punto. Lo spazio di dati selezionato attorno al punto considerato è solitamente conosciuto sotto il nome di k-neighborhood.
- **keypoints** contiene l'implementazione di diversi algoritmi per l'individuazione di *keypoint* nelle point cloud. Questi keypoint (chiamati anche punti di interesse) sono punti in un immagine o point cloud che sono stabili, distintivi e possono essere identificati usando dei criteri ben definiti. In genere, il numero di punti di interesse in una point cloud è molto inferiore rispetto al numero di punti totali, e quando usato in combinazione con descrittori di feature locali per ogni keypoint, i keypoint e i descrittori possono essere usati per formare una rappresentazione compatta ma distintiva dei dati originali.
- **registration** contiene molti algoritmi per la registrazione di point cloud organizzate e non organizzate. La registrazione tra due point cloud si occupa di trovare la trasformazione che minimizza la distanza (errore di allineamento) tra punti corrispondenti.
- **kdtree** fornisce la struttura dati kd-tree, che usa l'implementazione FLANN per effettuare velocemente ricerche nearest neighbor. Un Kd-tree (albero a k dimensioni) è una struttura di dati per partizionare lo spazio che memorizza un set di punti a k-dimensioni in una struttura ad albero che permette efficienti range search e nearest neighbor search. Nearest neighbor searches sono operazioni fondamentali quando si lavora con dati point cloud e

può essere utilizzata per trovare corrispondenze tra gruppi di punti, descrittori di features o per definire un neighborhood locale attorno a un punto o ad un insieme di punti.

- **octree** fornisce un modo efficiente per creare una struttura ad albero gerarchica dei dati a partire da una point cloud. Questo permette il partizionamento spaziale, il downsampling e le operazioni di ricerca sul data set di punti. Ogni nodo octree ha otto figli oppure nessuno. Il nodo radice descrive un bounding box cubico che racchiude tutti i punti. A ogni livello dell'albero questo spazio vien suddiviso da un fattore 2 che corrisponde a un incremento della voxel resolution.
- **segmentation** contiene algoritmi per la segmentazione di point cloud in cluster distinti. Questi algoritmi sono maggiormente indicati per elaborare una point cloud che è composta da un numero di regioni spazialmente isolate. In questi casi, clustering è spesso usato per suddividere la cloud nelle sue parti che la costituiscono, che possono poi essere elaborate in modo indipendente.
- **sample consensus** contiene metodi di SAmple Consensus (SAC) come RANSAC e modelli come piani e cilindri. Questi possono essere combinati liberamente in modo da identificare modelli specifici e i loro parametri nelle point clouds. Alcuni dei modelli implementati in questa libreria includono: linee, piani, e sfere. L'adattamento ai piani è solitamente applicato al compito di identificare superfici interne comuni, come muri, pavimenti o piani di tavoli. Altri modelli possono essere usati per identificare e segmentare oggetti dalle strutture geometriche comuni.
- *surface* si occupa della ricostruzione della superficie originale da scansioni 3D. in base al compito assegnato, che può essere ad esempio un convex/concave hull, una rappresentazione mesh o una superficie smoothed/resampled con normali. Creare un guscio convesso o concavo è utile per esempio quando c'è la necessità di una rappresentazione semplificata di una superficie o quando devono esserne estratti i confini. Meshing è un metodo utilizzato in generale per creare una superficie a partire da dei punti. Smoothing e resampling possono essere importanti se la cloud è rumorosa o se è composta da scansioni multiple che non sono allineate perfettamente. La complessità della surface estimation può essere regolata, e le normali possono essere stimate nello stesso step se necessario.
- **range image** contiene classi per rappresentare ed elaborare *range image*. Una range image (o depth map) è un'immagine nella quale i valori di ogni pixel rappresentano la distanza dal sensore e sono rappresentazioni 3D generate comunemente da telecamere stereo o a tempo di volo. Conoscendo i parametri intrinseci di calibrazione del sensore è possibile convertire una range image in una point cloud.
- **io** contiene classi e funzioni per la lettura e la scrittura di file di dati point cloud e mesh (PCD e PLY), e per catturare point clouds da una varietà di sensori RGB-D compatibili,

quali la Microsoft Kinect e la Asus XTionPro.

- **visualization** questa libreria è stata creata con lo scopo di riuscire a prototipare velocemente e visualizzare i risultati degli algoritmi operanti su point cloud 3D. Simile alle routines `highhui` di OpenCV per mostrare immagini in 2D e per disegnare forme in 2D sullo schermo questo modulo permette di visualizzare point cloud 3D e disegnare forme 3D.

2.3 OpenCV

OpenCV[4] (*Open Source Compure Vision*) è una libreria di funzioni orientata allo sviluppo di applicazioni real time nell'ambito della computer vision. Originariamente sviluppata da Intel, viene ora supportata da Willow Garage e Itseez. OpenCV è cross-platform e viene rilasciata con licenza BSD, gratuita sia per un uso accademico che commerciale, per molte piattaforme tra cui Linux, Windows, MacOS, Android e iOS. La libreria comprende più di 2500 algoritmi e offre interfacce in C++, C, e Python. Esempi di applicazioni di OpenCV sono: Human-Computer Interaction (HCI), object identification, segmentation e recognition, face recognition, gesture recognition, motion tracking, ego motion, motion understanding, structure from motion (SFM), stereo e multi-camera calibration e depth computation, mobile robotics. Come PCL, OpenCV è completamente integrata in ROS il quale fornisce funzioni di conversione tra formato immagini nativo della libreria e messaggi ROS.

[4] http://http://opencv.willowgarage.com

Capitolo 3

Microsoft Kinect

Figura 3.1: Microsoft Kinect

Microsoft Kinect (Figura 3.1) si basa su una tecnologia software sviluppata internamente da Rare, una sussidiaria di Microsoft Game Studios, e su una range camera sviluppata dall'israeliana PrimeSense, che interpreta le informazioni provenienti da un proiettore a luce infrarossa strutturata per ricostruire la scena 3D. Questo sistema di scansione 3D è chiamato *Light Coding* ed impiega una variante delle ben note tecnologie di ricostruzione di immagini 3D [5]. Per Microsoft il progetto Kinect aveva l'obiettivo di allargare l'utenza della console Microsoft Xbox 360 oltre la tradizionale utenza di giocatori, permettendo un'esperienza di intrattenimento e di gioco libera dall'impiego di sistemi di input tradizionali, attraverso un'interfaccia utente naturale (NUI) che fa uso di gesture e comandi vocali. Il sensore Kinect permette al computer di percepire direttamente la terza dimensione, ovvero la profondità dell'ambiente in cui è immerso il giocatore; inoltre è in grado di riconoscere quando l'utente sta parlando, sa identificare gli utenti, e sa interpretare i movimenti e i gesti compiuti da ciascuno di essi, traducendoli in un formato che gli sviluppatori possono utilizzare per costruire una nuova esperienza utente. L'impatto del sensore Kinect in breve tempo si è espanso ben oltre l'industria videoludica. Complice la sua grande disponibilità ed il prezzo contenuto, molti ricercatori e professionisti in informatica,

ingegneria elettronica, e robotica stanno sfruttando la tecnologia di rilevamento per sviluppare nuovi modi creativi per interagire con le macchine e per eseguire altre attività, dai sistemi informativi, al controllo robotico.

3.1 Caratteristiche hardware

Il sensore Kinect incorpora dell'hardware di percezione molto avanzato. In particolare, presenta un sensore di profondità, una telecamera RGB, e un array di quattro microfoni che abilitano funzionalità di motion capture 3D, riconoscimento facciale e riconoscimento vocale. Il sensore di profondità consiste di un proiettore laser ad infrarossi combinato con un sensore CMOS monocromatico che interpreta la luce infrarossa catturata creando una *depth map* dell'ambiente. Il range del sensore di profondità è regolabile e il software Kinect è in grado di calibrarlo in modo automatico sulla base del gioco o dell'ambiente fisico del giocatore, in modo da adeguarsi alla presenza di mobili o altri tipi di ostacoli. L'output video di default del sensore ha un rate

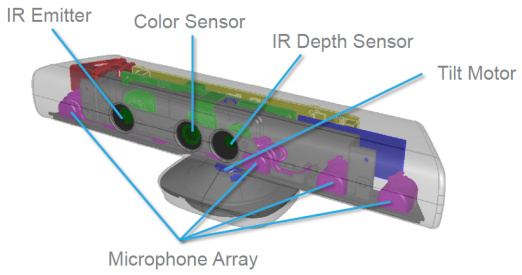


Figura 3.2: Componenti sensore Microsoft Kinect

di 30Hz, ed è composto da due differenti stream. Il primo proveniente dalla camera RGB ha risoluzione VGA a 8bit (640 x 480 pixel), con un filtro colore di Bayer, mentre il secondo è prodotto dal sensore di profondità monocromatico sempre con risoluzione VGA, con profondità a 11bit, cioè 2048 livelli di sensibilità. Per quanto riguarda lo stream RGB è possibile configurare il sensore per funzionare a risoluzione 1280 x 960 pixel a 12Hz, mentre per lo stream depth sono disponibili anche le risoluzioni 160 x 120 e 320 x 240 pixel sempre a 30 Hz. Il range di utilizzo della Kinect, quando utilizzata con software Xbox, è compreso tra 1.2 e 3.5 metri di distanza. L'area richiesta per giocare con la Kinect è all'incirca di 6 metri quadri, un volume di 12 metri cubi, nonostante il sensore sia in grado di effettuare il tracciamento in un range più ampio, tra 0.7 e 6 metri. Il sensore ha un angolo di visione di 57° orizzontalmente e 43° verticalmente, mentre il pivot motorizzato può garantire ulteriori 27° sia in alto che in basso. Il campo orizzontale della Kinect alla distanza minima di 0.8 metri è pari a circa 87cm, e il campo verticale a 63cm,

che comporta una risoluzione di 1.3mm per pixel. Come ampiamente descritto in precedenza, la Kinect è formata essenzialmente da 3 tipi di input, una normale telecamera, un sensore di profondità e un array di microfoni. L'output del sensore è costituito da due matrici, una per la camera RGB e la seconda per la depth camera. La prima viene mostrata direttamente in modo grafico, in quanto si tratta di una normale immagine RGB; la seconda è di solito utilizzata nel riconoscimento dello scheletro, può essere mappata come una range image, con una colorazione che esprime la profondità degli oggetti.

3.2 Microsoft Kinect for Windows SDK

Kinect for Windows SDK fornisce un insieme di librerie software e strumenti per lo sviluppo di applicazioni con il sensore Kinect. La versione attuale dell'SDK è la 1.6, rilasciata nell'ottobre 2012, ed è compatibile solamente con Windows 7 e Windows 8. Kinect for Windows SDK si articola nell'architettura mostrata in Figura 3.4.

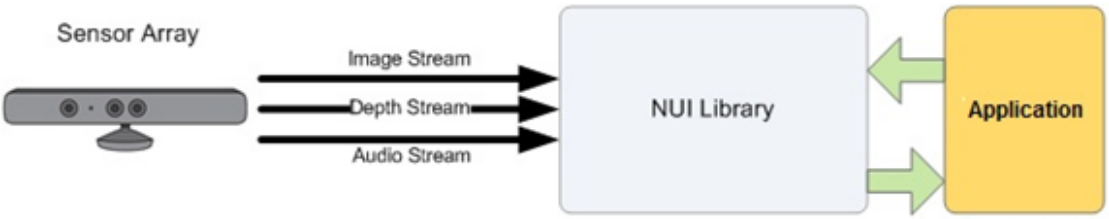


Figura 3.3: Interazione tra hardware Kinect, SDK e applicazioni

1. **Hardware** comprende i componenti del sensore Kinect e l'hub USB attraverso il quale è connesso al computer.
2. **Microsoft Kinect driver Kinect** i driver per Windows hanno le seguenti funzionalità:
   - accesso all'array di microfoni con API Windows.
   - accesso agli stream video RGB e depth.
   - impiego di più periferiche Kinect contemporaneamente.
3. **NUI API** Insieme di API che permettono di controllare le funzionalità più avanzate di Kinect, quali
   - skeleton tracking
   - face recognition
   - voice recognition
4. **Kinect Audio DMO** estende le funzionalità dell'array di microfoni per fornire le funzionalità di *beamforming* (mappatura sonora dell'ambiente) e localizzazione della sorgente sonora.

5. **Windows standard API** le API audio, speech e media presenti in Windows e Microsoft Speech.

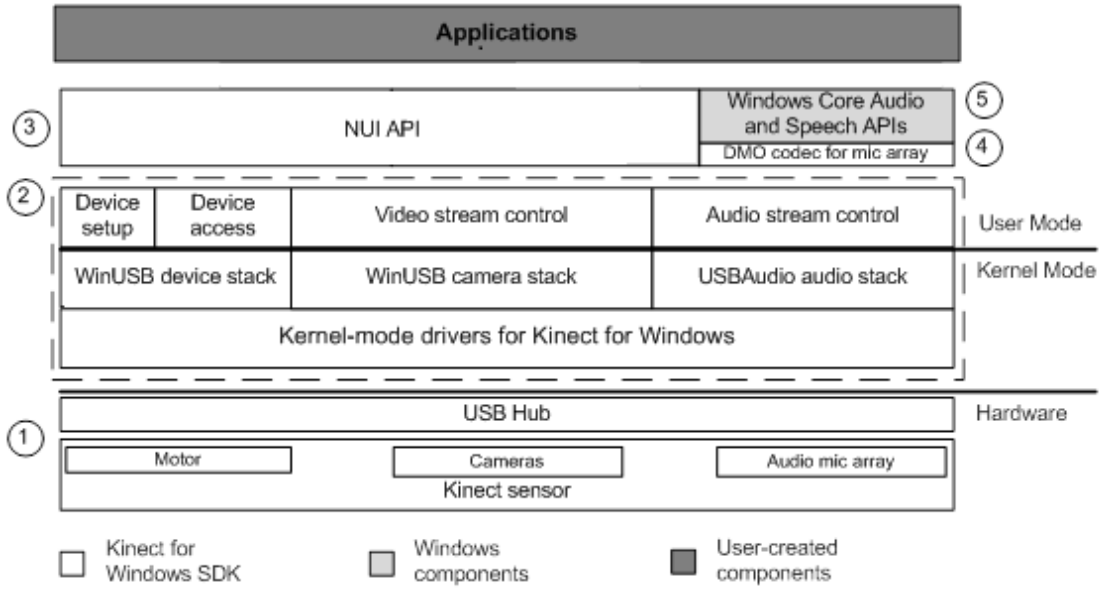


Figura 3.4: Architettura Kinect for Windows SDK

3.3 OpenNI

OpenNI[1] (*Open Natural Interaction*) è un framework multi-piattaforma che mette a disposizione API per sviluppare applicazioni utilizzando la *Natural Interaction*. Il termine *Natural Interaction* si riferisce ai sistemi che impiegano i sensi umani, in particolare, visione e udito per l'interazione uomo-macchina. Il principale obiettivo di OpenNI è quello di definire un insieme di API standard in grado di gestire sensori audio e video, e *middleware* di percezione (componenti software che analizzano l'audio e il video di una scena), permettendo la comunicazione tra questi due componenti, pur mantenendoli chiaramente distinti. Grazie a questa architettura, OpenNI permette di sviluppare algoritmi che funzionano con i dati raw, indipendentemente dal sensore che li ha generati. Il framework OpenNI è quindi un layer di astrazione (Figura 3.5) che fornisce interfacce sia verso i sensori che verso i componenti middleware. I sensori attualmente supportati sono: sensori video 3D (RGB camera, IR camera come Microsoft Kinect o Asus XTion Pro) e dispositivi audio (microfoni o array di microfoni). Mentre le componenti middleware messe a disposizione da OpenNI sono:

[1] http://http://www.openni.org/

- full body analysis e skeleton tracking
- hand point analysis
- gesture detection
- scene analyzer

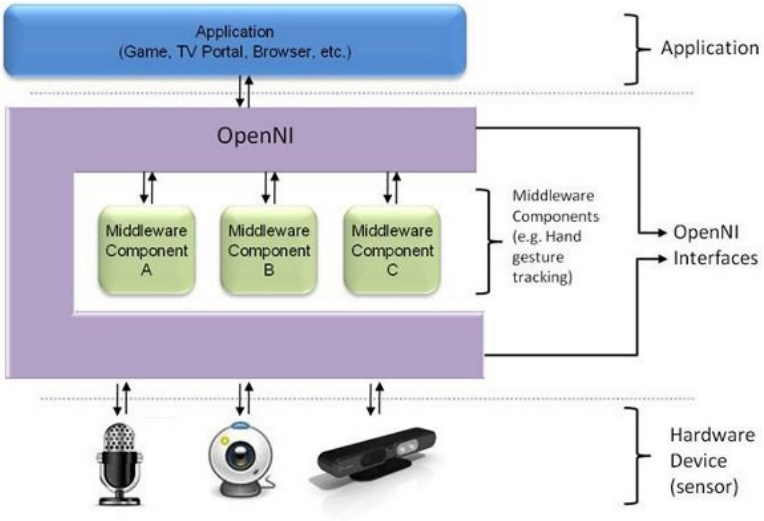


Figura 3.5: Architettura framework OpenNI

Capitolo 4

Tracking dello scheletro

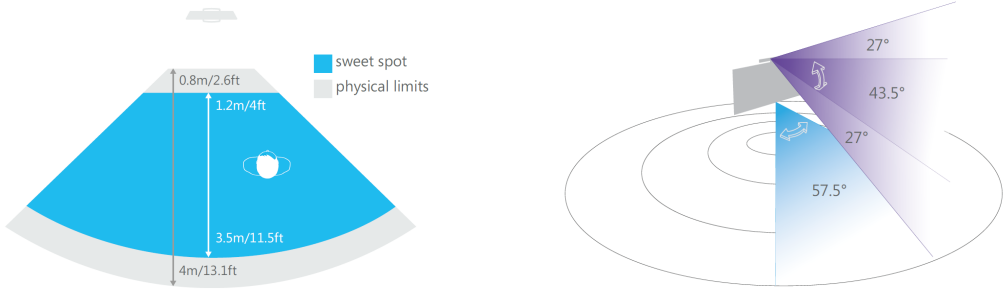
La tecnica di trasformazione a posa standard impiega le informazioni ottenute dal tracking dello scheletro per ricondurre le point cloud di persone ad una posa comune. Lo scheletro fornito dal tracker comprende un insieme di punti (*joints*), le cui coordinate 3D corrispondono a punti caratteristici del corpo umano (testa, mani, spalle, piedi, ecc.). La concatenazione di due joint da origine a segmenti, detti *link* o *bone*, che rappresentano le ossa dello scheletro e la cui orientazione corrisponde all'orientazione spaziale della relativa parte del corpo. In questo capitolo vengono introdotti gli algoritmi utilizzati per il tracking dello scheletro e vengono analizzati e comparati tre framework compatibili con il sensore Microsoft Kinect impiegati durante il progetto di tesi:

- Kinect for Windows SDK Skeletal Tracker
- OpenNI Skeletal Tracker
- PCL Skeletal Tracker

4.1 Kinect for Windows SDK Skeletal Tracker

Questo skeletal tracker è compreso all'interno del Kineck for Windows SDK e fornisce un insieme di API che permettono un facile accesso alle articolazioni scheletro. Lo skeletal tracker è in grado per riconoscere persone in piedi o sedute di fronte alla Kinect, ad una distanza compresa tra 0.8 e 4.0 metri, sebbene il range di visione ottimale si colloca tra 1.2 e 3.5 metri, oltre i quali le informazioni di profodità del sensore IR tendono ad essere più rumorose e quindi meno affidabili. Il tracker non è però in grado di riconoscere se la persona è rivolta verso il sensore oppure volge le spalle alla Kinect. Kinect for Windows SDK permette di riconoscere fino a sei utenti, solo due utenti possono essere monitorati in modo dettagliato, il che significa

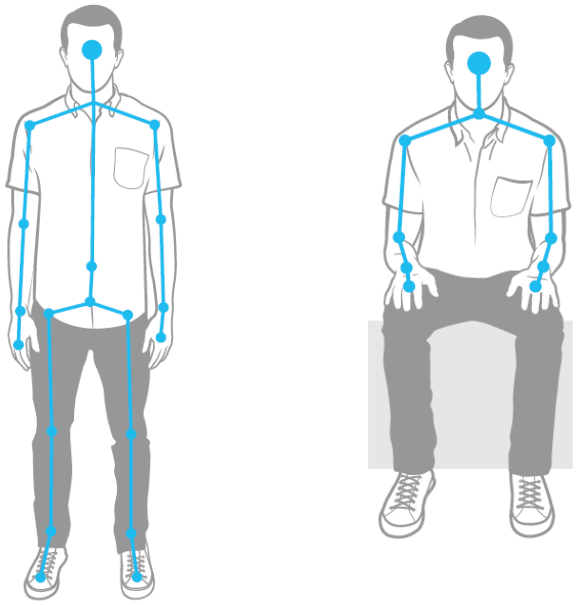
che il sensore restituisce le posizioni di tutti e 20 i joint, mentre per gli altri viene fornita solamente la posizione complessiva. Questo perchè il tracking completo di tutti gli utenti risulta computazionalmente troppo oneroso per applicazioni real-time.



(a) Range in default mode

(b) Angoli di visione (depth e RGB).

Figura 4.1: Campo di visione.



(a) Full skeleton mode

(b) Seated skeleton mode

Figura 4.2: Modalità di tracking: Kinect for Windows SDK permette di riconoscere 20 joint per scheletro in modalità *default* e 10 joint in modalità *seated*.

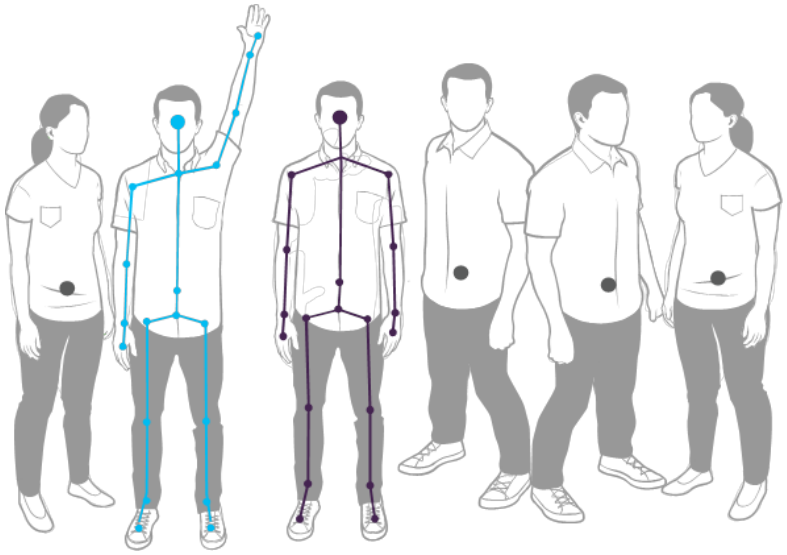


Figura 4.3: Tracking multi-utente

4.1.1 Algoritmo di tracking

Il tracking dello scheletro in tempo reale risulta difficile a causa delle differenti pose che l'utente può assumere (ogni parte del corpo si può muovere in migliaia di modi e direzioni differenti), altezza, corporatura, vestiti e così via. Per superare questi problemi e riconoscere i joint indipendentemente dalla posa della persona *Shotton et al.* in [13, 14], propongono un approccio basato sul *machine learning*. In particolare viene impiegato un classificatore basato su *random decision forest* per predire accuratamente la posizione dei joint dello scheletro. Questo classificatore viene allenato mediante un dataset di training appositamente creato composto da molti modelli di persone generati sinteticamente (500k frame e 100k pose) che variano per altezza, corporatura, vestiti, orientazioni e punti di vista, nei quali ogni punto è etichettato in base alla parte del corpo. Questa segmentazione definisce 31 *body part*: LU/RU/LW/RW head, neck, L/R shoulder, LU/RU/LW/RW arm, L/R elbow, L/R wrist, L/R hand, LU/RU/LW/RW torso, LU/RU/LW/RW leg, L/R knee, L/R ankle, and L/R foot (Left, Right, Upper, loWer).

La rendering pipeline utilizzata dallo skeletal tracker si compone di diversi passaggi che permettono di riconoscere le parti del corpo a partire dai dati di profondità:

1. Il sensore Kinect restituisce i dati grezzi di profondità, in cui ogni pixel contiene un valore che rappresenta la distanza tra il sensore e la persona. A partire dalla depth image viene rimosso il background identificando solamente i pixel corrispondenti all'utente (*user map*).
2. L'algoritmo *Body Part Classification* (BPC) si occupa di assegnare indipendentemente

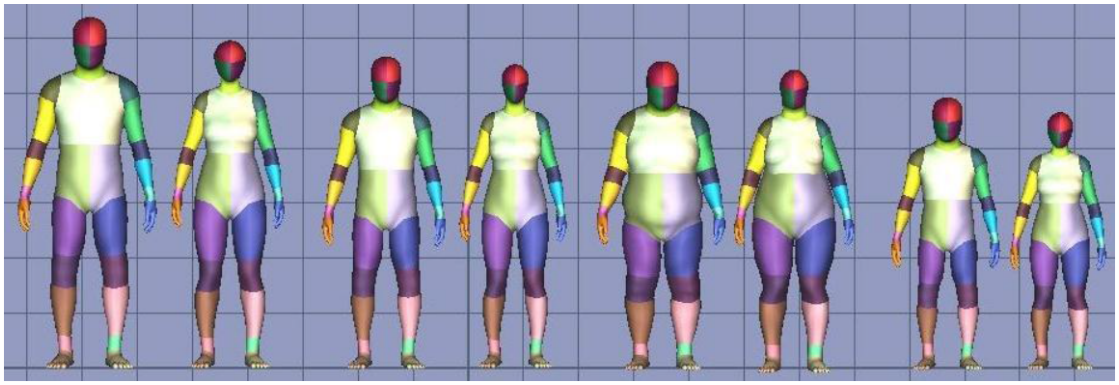


Figura 4.4: Esempi di modelli del training set generati sinteticamente.

Figura 4.5: Esempi di *depth image* e corrispondente *body part label image* del training set dello skeletal tracker.

ad ogni pixel dell'utente la label della parte del corpo a cui appartiene, classificandolo mediante decision forest. Le decision forest sono delle strutture ad albero nelle quali ogni nodo rappresenta un modello con etichettatura delle parti del corpo.

3. La segmentazione del corpo tramite labelling viene utilizzata per localizzare i joint del corpo.
4. La posizione dei joint viene espressa mediante coordinate 3D (X, Y e Z) dove X e Y definiscono la posizione del joint e Z rappresenta la distanza dal sensore. Per ottenere le coordinate corrette vengono calcolate tre viste della stessa immagine: frontale, superiore e laterale come mostrato in Figura 4.6

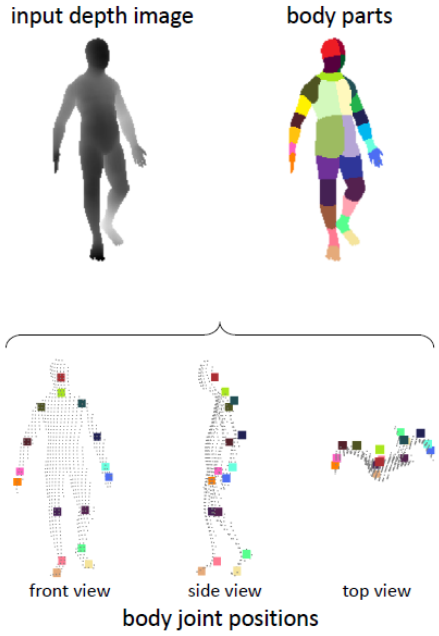


Figura 4.6: BPC calcola la body part label per ogni pixel e poi usa questa segmentazione per localizzare i joint dello scheletro.

4.1.2 Caratteristiche dello scheletro

Lo scheletro ottenuto dal tracker si compone di 20 joint e 19 bone. Per ogni joint vengono restituite le coordinate 3D e la confidenza del tracking: `Tracked` (il joint è riconosciuto con la massima confidenza), `Inferred` (il joint non è riconosciuto perchè occluso ma la sua posizione viene ricavata dalla posizione degli altri joint) o `Not Tracked` (il joint non è riconosciuto dal tracker). Lo schema in Figura 4.7 rappresenta uno scheletro umano completo rivolto verso il sensore Kinect. I joint dello scheletro sono organizzati in una struttura gerarchica ad albero che

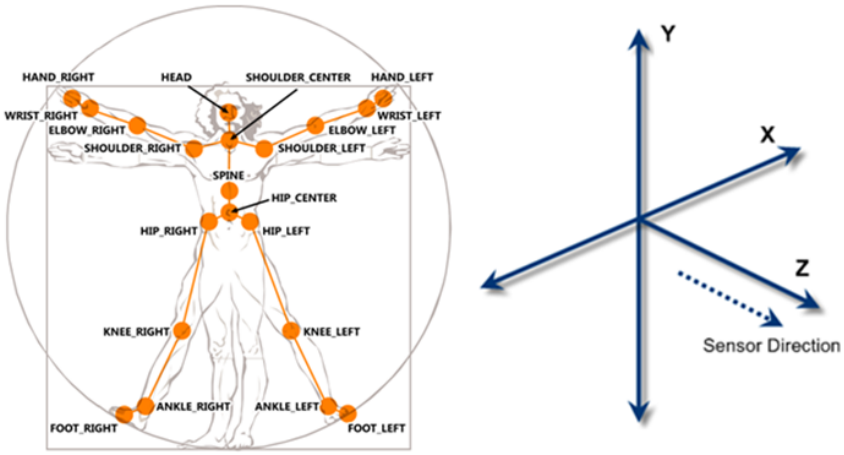


Figura 4.7: Schema scheletro Kinect SDK

ha come nodo radice l'*hip center* e ogni bone ha un joint padre e un joint figlio. Questo implica che ogni joint ad esclusione dei joint foglia (testa, mani e piedi) può essere sia padre che figlio in base al bone considerato.

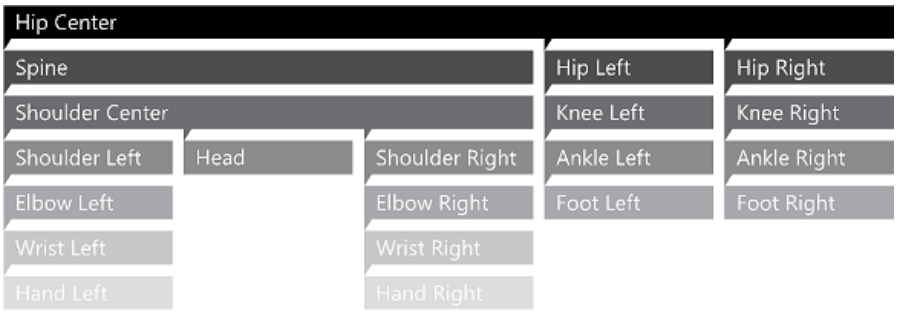


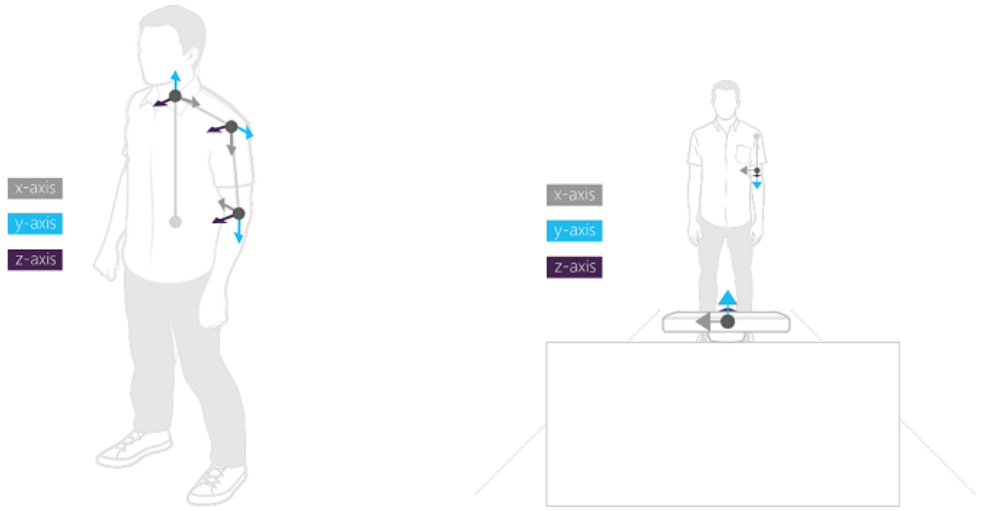
Figura 4.8: Organizzazione gerarchica dello scheletro

Per ogni bone viene restituito un quaternione che ne definisce l'orientazione spaziale assoluta oppure relativa al bone precedente nella gerarchia:

- **Hierarchical orientation** Il quaternione associato al bone rappresenta la rotazione necessaria per passare dall'orientazione del bone padre a quella del bone corrente. Questo equivale a considerare la rotazione del sistema di riferimento cartesiano dallo spazio del bone padre a quello del bone figlio considerando il bone parallelo all'asse y. Nella definizione gerarchica, la rotazione dell'hip center fornisce l'orientamento assoluto dell'utente respetto alle coordinate spaziali del sensore. Ciò presuppone che lo spazio oggetto giocatore ha l'origine nell'hip center, l'asse y è verticale, l'asse x è a sinistra, e l'asse z verso il

sensore.

- **Absolute orientation** Il quaternione associato al bone rappresenta la rotazione assoluta del bone nello spazio 3D. Anche in questo caso la rotazione dell'hip center fornisce l'orientamento assoluto dell'utente.



(a) Hierarchical orientation (b) Absolute orientation

Figura 4.9: Bone orientation

4.2 OpenNI Skeletal Tracker

Le funzionalità di tracking dello scheletro del framework OpenNI vengono fornite dal middleware NiTE (*Natural Interface Technology for End-User*) sviluppato da Primesense. Rispetto al tracker Microsoft, NiTE permette di riconoscere se le persone sono rivolte verso il sensore oppure sono di spalle ed inoltre non presenta nessuna limitazione software sul numero di utenti per i quali è possibile effettuare il tracking completo di tutti i joint; l'unico limite è imposto dalla capacità di elaborazione della CPU del sistema impiegato.

4.2.1 Algoritmo di tracking

Non sono diponibili molte informazioni riguardanti l'algoritmo impiegato da NiTE per il tracking dello scheletro in quanto il middleware non è open source e Primesense non ha pubblicato nessun dettaglio al riguardo. Nella Figura 4.10 è schematizzata la pipeline di tracking dello scheletro implementata da OpenNI.

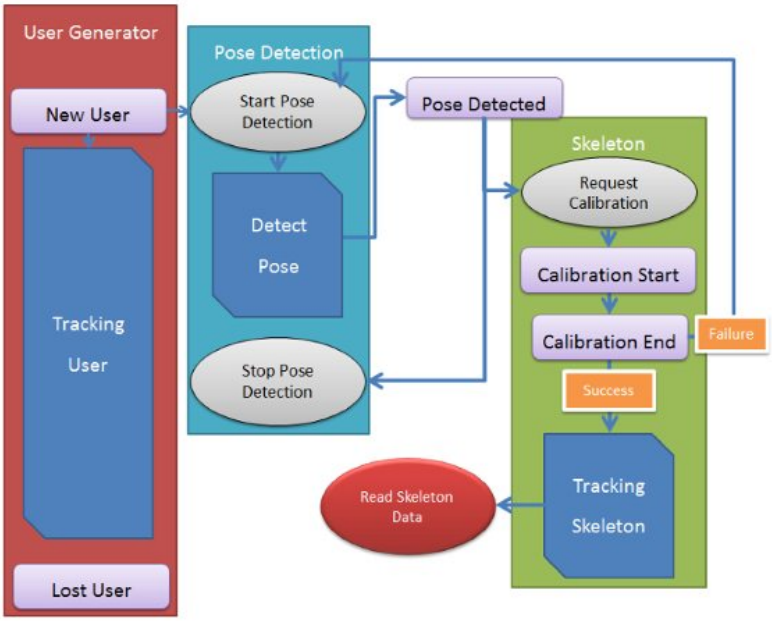


Figura 4.10: Pipeline di tracking di OpenNI

4.2.2 Caratteristiche dello scheletro

Il tracker permette ti riconoscere fino a 15 joint e 14 links. Come per lo skeletal tracker Microsoft per ogni joint vengono restituite le coordinate 3D, l'orientazione e la confidenza del tracking: `Tracked` (il joint è riconosciuto con la massima confidenza), `Inferred` (il joint non è riconosciuto perchè occluso ma la sua posizione viene ricavata dalla posizione degli altri joint) o `Not Tracked` (il joint non è riconosciuto dal tracker). Come si può osservare dalla (Figura 4.11) la posizione e l'orientazione dei joint viene fornita nel sistema di coordinate del mondo reale. L'origine del sistema di coordinate è il sensore, +X punta verso destra, +Y punta verso l'altro e +Z punta verso la direzione di profondità crescente.

4.3 PCL Skeletal Tracker

Questo skeletal tracker si basa sul lavoro di Koen Buys, Cedric Cagniart, Anatoly Bashkeev e Caroline Pantofaru, ed è stato presentato a ICRA2012 e IROS2012. Una prima versione del tracker è disponibile per la compilazione nel trunk di PCL e necessita di una GPU Nvidia con architettura *Fermi* o *Kepler* per essere eseguito in quanto sfrutta CUDA per la parallelizzazione dell'algoritmo di labeling.

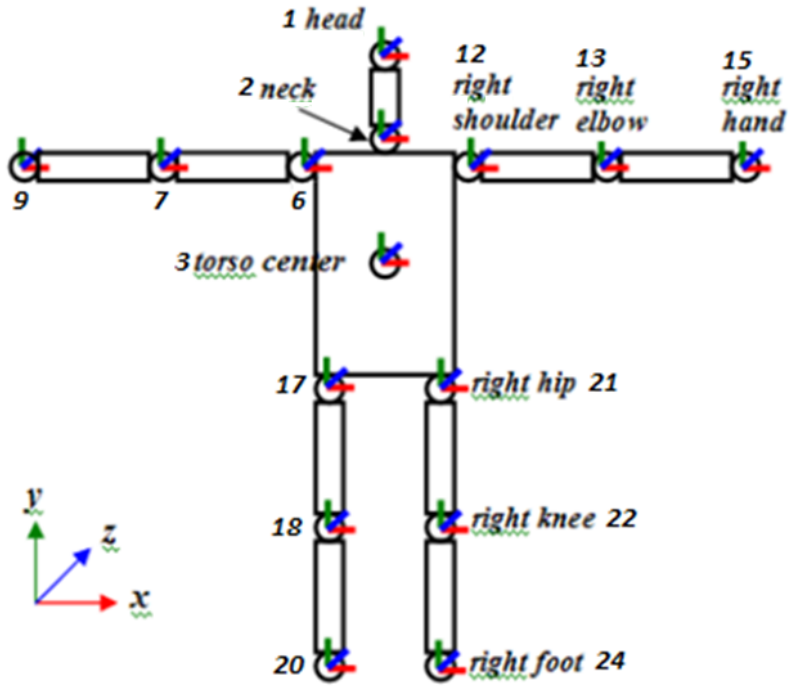


Figura 4.11: Schema scheletro OpenNI

4.3.1 Algoritmo di tracking

L'implementazione si basa su un approccio simile a *Shotton et al.*[13] ed impiega, come illustrato in precedenza per il tracker Microsoft, delle *random decision forest* allenate offline mediante un dataset di modelli di persone già segmentati rispetto alle parti del corpo. A partire dalla classificazione delle parti del corpo mediante le random decision forest (*labeling*) vengono estratte le coordinate dei joint. Nel dettaglio il workflow impiegato dall'algoritmo è il seguente:

1. rimozione del background
2. etichettattura di ogni pixel della persona in base alla parte del corpo a cui appartiene con maggiore probabilità mediante la classificazione con ogni *tree* della *random forest*
3. sintesi dell'output proveniente da ogni tree in una singola etichettatura media
4. *clustering* di ogni parte del corpo per calcolare la posizione del joint corrispondente

4.3.2 Caratteristiche dello scheletro

Il tracker restituisce la segmentazione della persona in 27 parti del corpo (Lfoot,Lleg, Lknee, Lthigh, Rfoot, Rleg, Rknee, Rthigh, Rhips, Lhips, Neck, Rarm, Relbow, Rforearm, Rhand, Larm, Lelbow, Lforearm, Lhand, FaceLB, FaceRB, FaceLT, FaceRT, Rchest, Lchest, Lshoulder,

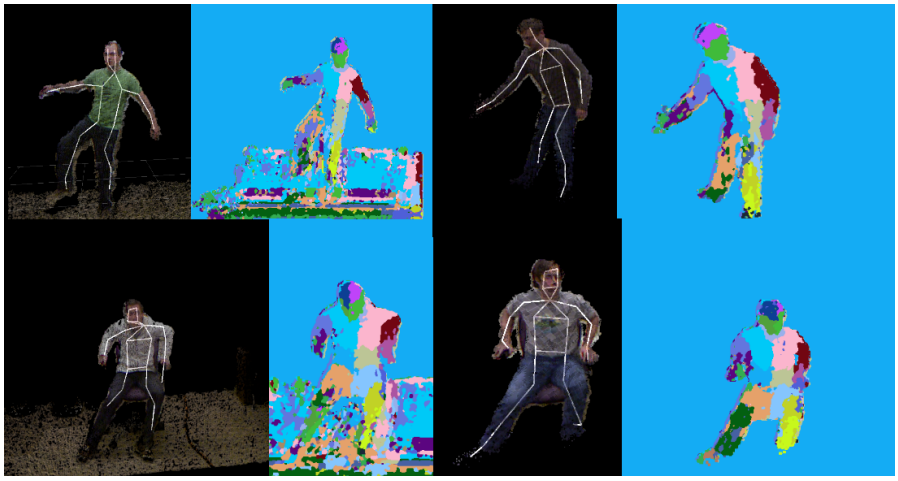


Figura 4.12: Esempi di scheletro e labeling delle parti del corpo.

Rshoulder), al momento della scrittura di questa tesi non vengono invece fornite direttamente né le posizioni dei joints né l'orientazione dei link dello scheletro. Queste limitazioni lo rendono inadatto ai fini del progetto descritto in questa tesi, il tracker è comunque ancora nelle prime fasi di sviluppo e grazie alla sua implementazioni su GPU in futuro potrebbe presentare una valida alternativa.

Figura 4.13: Output del tracker

4.4 Comparazione

Nella tabella seguente vengono riassunte le principali caratteristiche dei tracker Microsoft e OpenNI presentati in questo capitolo.

Skeletal tracker	**Microsoft**	**OpenNI**
N° joint	20	15
N° link	19	14
Risoluzione immagine RGB massima	1280x960	640x480
Risoluzione depth map massima	640x480	640x480
Tracking a 360°	No	Si
N° di scheletri riconosciuti	2	Illimitato

Tabella 4.1: Riepilogo caratteristiche skeletal tracker.

Scheletro Lo skeletal tracker Microsoft riconosce 20 joint, mentre OpenNI solamente 15. In particolare, OpenNI non definisce i joint di *wrist*, *ankle* e *spine* presenti nel tracker Microsoft, per cui l'orientazione di mani e piedi non viene calcolata indipendentemente ma viene impostata uguale all'orientazione del corrispondente avambraccio o gamba. Inoltre OpenNI richiede alcuni secondi di calibrazione per il riconoscimento della persona. La precedente versione del tracker infatti richiedeva all'utente di mantenere per qualche secondo una posa di calibrazione (*psi pose* o *surrender pose*), e sebbene questo vincolo è stato rimosso nella versione corrente le informazioni dello scheletro raccolte nei primi frame potrebbero non risultare accurate. Infine, OpenNI è in grado di effettuare il tracking sia se l'utente è rivolto verso il sensore, sia di spalle, adattando correttamente la rotazione dello scheletro, mentre il tracker Microsoft non distingue nativamente se la persona è ripresa di spalle e quindi in questo caso non è in grado di stimare la posizione corretta dei joint.

Accuratezza Entrambi i tracker permettono di riconoscere i joint con buona accuratezza nel caso in cui tutte le parti del corpo siano completamente visibili, mentre nel caso in cui alcuni joint siano occlusi il tracker Microsoft riesce a predirre con più accuratezza la loro posizione. Per quanto riguarda le orientazioni dei link invece il tracker OpenNI alcune volte non ritorna la rotazione corretta dei link in relazione alla posizione frontale o di spalle dell'utente.

L'algoritmo di trasformazione a posa standard è stato inizialmente implementato impiegando i dati dello scheletro forniti dal Kinect for Windows SDK e poi è stato adattato per lavorare anche

con i dati forniti da OpenNI in modo da permettere una ricostruzione completa a 360 gradi della persona non possibile con il tracker Microsoft.

Capitolo 5

Trasformazione a posa standard

Le tecniche di riconoscimento di oggetti 3D si basano sul presupposto che gli oggetti siano indeformabili e siano sottoposti solamente a trasformazioni rigide. Tuttavia, quando si considerano le persone in movimento non è possibile assumere l'ipotesi di rigidità in quanto le persone sono oggetti articolati e possono assumere una grande varietà di pose differenti. La tecnica che abbiamo sviluppato sfrutta le informazioni di posizione dei joint e l'orientazione dei link dello scheletro forniti dallo skeletal tracker per trasformare le persone ad una nuova posa, definita *posa standard*.

5.1 Segmentazione del corpo in parti

Il primo passo per la trasformazione a posa standard della point cloud consiste nel segmentare il corpo della persona in parti. Come descritto nel capitolo precedente, anche se lo skeletal tracker Microsoft stima la segmentazione del corpo come prima fase e poi da queste deriva la posizione dei joint, non fornisce in output il labeling della depth map in parti. Per questo motivo abbiamo implementato una procedura che permette di ottenere la segmentazione della persona in parti a partire dalle coordinate 3D delle posizioni dei joint. In particolare, per ogni punto della point cloud viene calcolata la distanza da tutti i link dello scheletro e viene assegnato alla parte del corpo corrispondente al link più vicino. Per migliorare la segmentazione di tronco e braccia, soprattutto quando le braccia sono in posizione di riposo lungo in corpo, sono stati aggiunti due link fittizi tra i joint delle anche e quelli delle spalle.

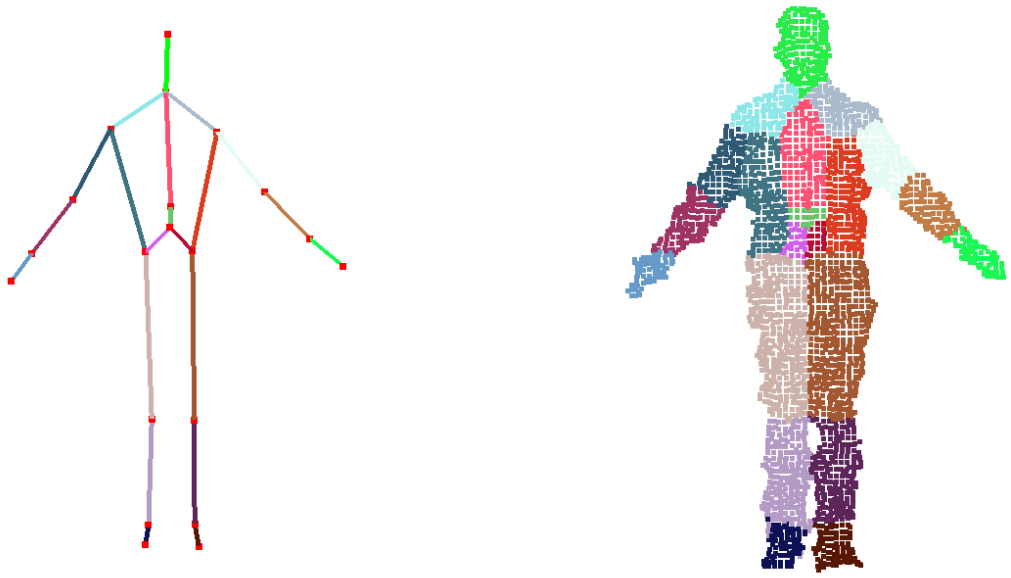


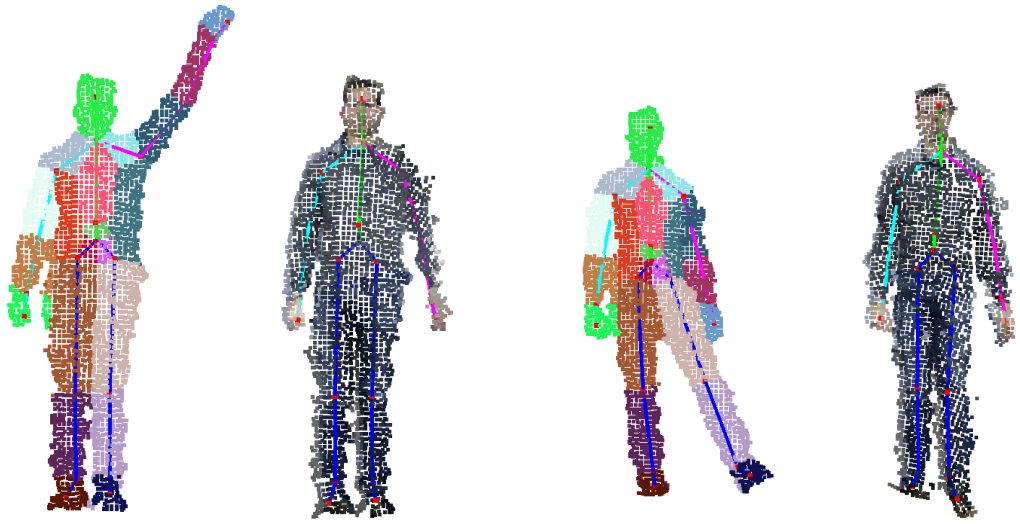
Figura 5.1: Segmentazione della cloud in parti in base ai link dello scheletro.

5.2 Algoritmo di trasformazione a posa standard

Una volta effettuata la segmentazione del corpo in parti, è possibile trasformare la posa assunta dalla persona in posa standard. Come descritto in precedenza la posa standard permette di confrontare direttamente le point cloud ottenute da diversi soggetti e pose, imponendo in tutte la stessa orientazione tra i link. Come posa standard è stata scelta la tipica posa frontale di una persona a riposo, con le braccia lungo i fianchi, come mostrato in Figura 5.2. Questa posizione è stata scelta in quanto naturale per una persona che sta ad esempio camminando, situazione molto comune nell'ambito del tracking di persone. Ogni parte del corpo segmentata viene ruotata in base all'orientazione del corrispondente link dello scheletro e traslata in base alle coordinate dei propri joint. L'orientazione dei link dello scheletro viene espressa dallo skeletal tracker mediante quaternioni per cui, per calcolare la rotazione relativa tra l'orientazione del link della parte del corpo del frame corrente e quello in posa standard, è sufficiente effettuare la seguente operazione:

$$R = Q_S \times Q_C^{-1} \tag{5.1}$$

dove R è la rotazione cercata, Q_S è il quaternione che esprime l'orientazione del link in posa standard mentre Q_C esprime l'orientazione dello stesso link nel frame corrente. Secondo l'algebra dei quaternioni questo equivale ad applicare ad ogni parte del corpo prima una rotazione che la riporta ad una orientazione neutra per poi applicare la rotazione che la porta ad avere



(a) (b)

Figura 5.2: Esempi di trasformazione a posa standard. A sinistra viene mostrata la segmentazione delle parti del corpo, a destra la cloud trasformata con texture RGB.

la corrispondente orientazione in posa standard. Visto che i quaternioni esprimono solamente l'informazione sull'orientazione del link dello scheletro ma non danno nessuna informazione riguardo alle sue coordinate nello spazio, prima di poter applicare la rotazione R è necessario traslare la parte del corpo nell'origine delle coordinate mediante una traslazione di vettore $V_C = \begin{bmatrix} -x_C & -y_C & -z_C \end{bmatrix}$, dove x_C, y_C, z_C sono le coordinate del joint padre del link corrente. A questo punto è possibile applicare la rotazione e successivamente traslare la parte del corpo di $V_S = \begin{bmatrix} x_S & y_S & z_S \end{bmatrix}$ dove x_S, y_S, z_S sono le coordinate del joint in posa standard. Queste coordinate vengono calcolate in base al link precedente come spiegato in dettaglio nel prossimo paragrafo.
Riassumendo la trasformazione applicata ad ogni punto della parte del corpo corrente è la seguente:

$$P' = T_{V_S}(R(T_{V_C}(P))) \tag{5.2}$$

L'algoritmo ricalcola le coordinate dei joint del link corrente a partire dalle coordinate del link padre secondo la struttura ad albero descritta in Figura 5.2. A partire dall'hip center ogni link viene concatenato con il precedente facendo coincidere il secondo joint del link padre con il primo joint del link figlio. Questo permette allo scheletro standard di essere indipendente dalla lunghezza dei link e di adattarsi alla struttura scheletrica del soggetto ripreso, evitando situazioni in cui la point cloud ricostruita è *spezzata* o *compressa* perché i link del soggetto

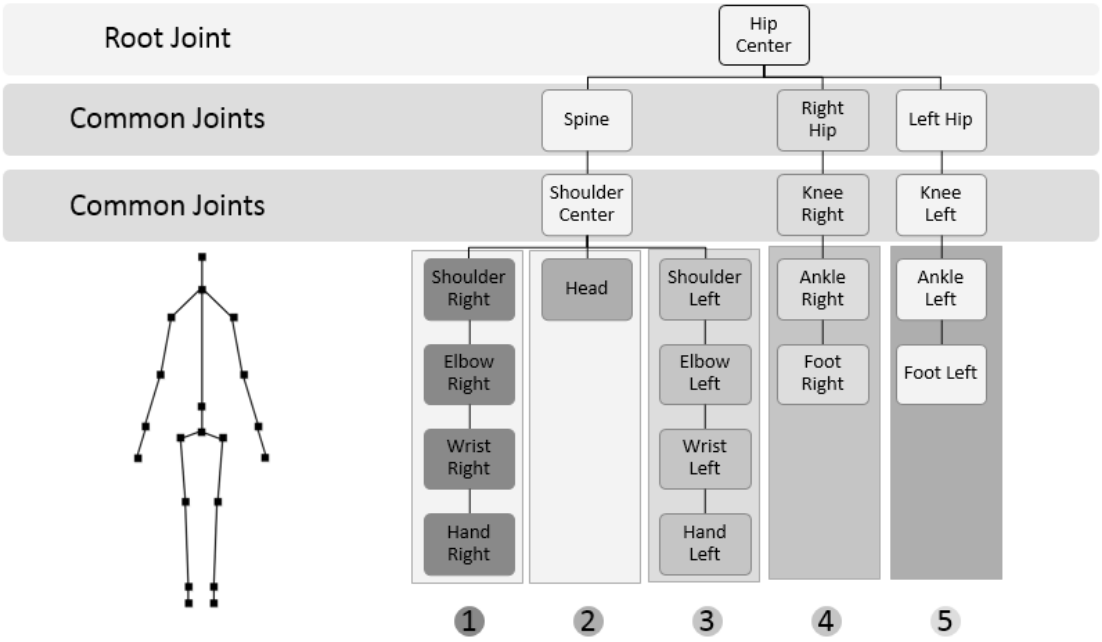


Figura 5.3: Organizzazione gerarchica ad albero dei joint.

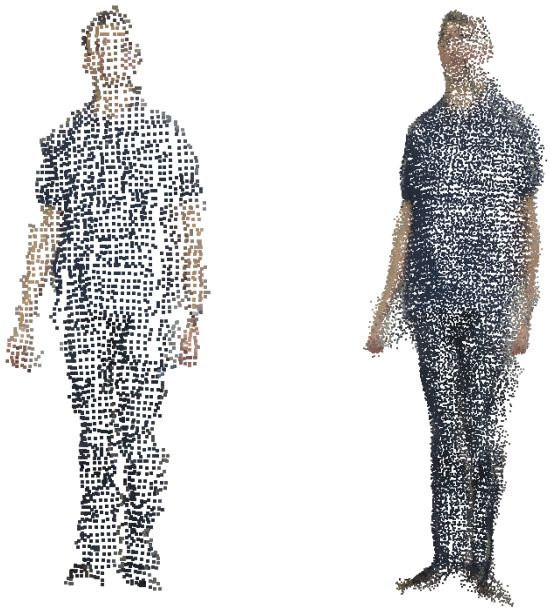
ripreso sono rispettivamente più corti o più lunghi rispetto allo scheletro standard. Nel dettaglio, al primo link ($hip - center \rightarrow spine$) viene applicata la rototraslazione in base all'orientazione e alle coordinate della posa standard e ad esso vengono concatenati gli altri link. Per ogni link successivo indicando con $P(P_0, P_1)$ e $F(F_0, F_1)$ rispettivamente il link padre in posa standard e il link figlio, il vettore ${V_S}^F = \begin{bmatrix} x_{P_1} & y_{P_1} & z_{P_1} \end{bmatrix}$ esprime la traslazione da applicare al link figlio (dopo la rotazione) per portarlo in posa standard.

5.3 Generazione modello 3D

In seguito alla trasformazione a posa standard le point cloud appartenenti alla stessa persona in movimento possono essere facilmente unite per comporre un modello più completo della persona. La Figura 5.4, mostra una point cloud (a) derivata da un singolo frame, mentre in (b) viene mostrato il modello ottenuto fondendo alcune point cloud acquisite da diversi punti di vista e trasformate in posa standard. Si può notare come la point cloud incrementale sia più densa e completa rispetto alla cloud singola. Questo modello può essere utilizzato come riferimento per il matching di nuove point cloud di test.
La creazione del modello si compone delle seguenti fasi:

1. Pre-processing

   (a) Controllo sulla validità dei dati dello scheletro
   (b) Face detection
2. Trasformazione a posa standard
3. Aggiunta della cloud corrente al modello e allineamento ICP
4. Voxeling
5. SOR filtering
6. MLS filtering



(a) (b)

Figura 5.4: (a) Point cloud in posa standard ottenuta da un singolo frame e (b) modello ottenuto sommando insieme point cloud in posa standard ottenute da diversi frame.

5.3.1 Pre-processing

Validazione scheletro

Per ottenere dei modelli visivamente di buona qualità ed utilizzabili per il matching è stato scelto di considerare solamente i frame in cui lo skeletal tracker restituisce lo stato tracked per tutti i joint dello scheletro. In questo modo solamente i frame in cui tutte le parti del corpo sono visibili vengono presi in considerazione per la creazione del modello. Inoltre viene effettuato un ulteriore controllo sulla posizione dei joint tramite un bounding box, centrato nel centroide della

cloud della persona e di dimensioni 2m x 3m x 1m rispettivamente nelle coordinate X, Y, Z. Se le coordinate di un joint risultano fuori dallo spazio delimitato dal box a causa di errori di stima dello skeletal tracker o imprecisioni in fase di registrazione il frame viene scartato.

Face detection

Come descritto in precedenza l'algoritmo di skeletal tracking Microsoft si basa su un random forest classifier che è stato allenato con esempi di persone viste solamente di fronte, per cui non fornisce la stima corretta dello scheletro quando una persona viene vista di spalle. Per questa ragione nella creazione del modello a partire dai dati forniti dal tracker Microsoft vengono mantenuti solamente i frame nei quali viene riconosciuta la presenza di una faccia. Per la face detection viene impiegato l'algoritmo proposto da Viola e Jones in [15]. Per migliorare il tempo di elaborazione e diminuire il numero di falsi positivi, la regione di ricerca della faccia è stata limitata ad un intorno della posizione 2D del joint della testa, ottenuto dallo skeletal tracker.

5.3.2 Voxel filtering

Per ridurre il numero di punti da processare e limitare il numero di punti nella cloud incrementale viene applicato un filtro voxel grid alla cloud di ogni frame. Il filtro voxel grid permette di effettuare il down-sample di una point cloud utilizzando un reticolo di punti. In dettaglio:

1. viene costruito un reticolo di punti dove il lato di ogni cubo viene impostato dal leafsize
2. per ogni cubo viene calcolato il centroide dei punti presenti al suo interno e impostato come voxel del cubo corrente

5.3.3 ICP - Iterative Closest Points

Dopo la trasformazione a posa standard, le point cloud di frame differenti possono essere sommate per formare un modello incrementale della persona. Questo viene fatto prima allineando le cloud secondo l'orientazione globale della persona, fornita dallo skeletal tracker come orientazione dell'hip center e successivamente rifinendo questo allineamento impiegando l'algoritmo di registrazione *Iterative Closest Points*. La trasformazione iniziale viene effettuata in quanto ICP converge molto velocemente se le due cloud non sono molto distanti l'una dall'altra, in caso contrario potrebbe rimanere bloccato in un minimo locale. ICP minimizza iterativamente la distanza tra punti corrispondenti delle point cloud. In ogni iterazione viene calcolata una nuova matrice di trasformazione. Le corrispondenze vengono ricercate solamente tra punti vicini (con un intorno imposto da un treshold) e poi una piccola trasformazione viene calcolata ad ogni iterazione. L'algoritmo, nell'implementazione di PCL, ha tre condizioni di terminazione:

- massimo numero di iterazioni
- il valore di epsilon tra le due ultime iterazione differisce meno di un valore specificato
- una funzione di fitness calcolata internamente raggiunge il threshold impostato

Nel nostro software solamente le prime due condizioni sono state impostate rispettivamente a 10 iterazioni ed epsilon a $10^{-}7$.

5.3.4 Statistical Outlier Removal filtering

La segmentazione fornita dalla *user map*, riesce solitamente ad isolare i pixel corrispondenti all'utente, eliminando il background. Nonostante ciò può capitare che alcuni pixel appartenenti al background siano presenti nei punti della point cloud della persona a causa di parametri di registrazione non perfetti o ritardi nella sincronizzazione. Per limitare l'impatto di queste imperfezioni sulla qualità del modello finale viene impiegato un filtro *Statistical Outlier Removal*. Il filtro SOR si basa sul calcolo della distribuzione dei punti nell'intorno del dataset di input. Per ogni punto viene calcolata la distanza media da esso ai suoi vicini. Assumendo che la distribuzione risultante sia gaussiana, il filtro elimina tutti i punti la cui distanza media risulta fuori dall'intervallo definito dalla media globale delle distanze e dalla deviazione standard.

5.3.5 Moving Least Squares smoothing

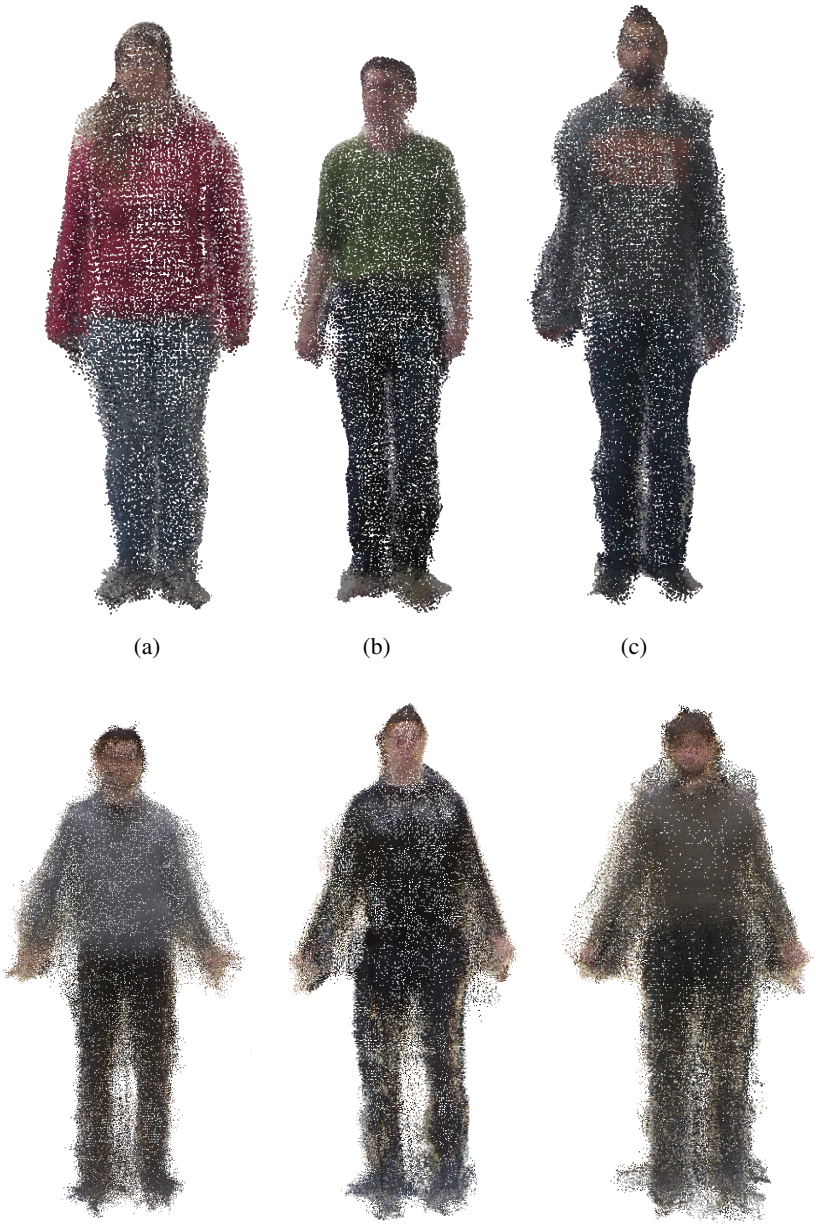
Le point cloud ottenute con il sensore Kinect hanno una buona risoluzione ma il passo di quantizzazione della *depth map* aumenta in maniera quadratica con la distanza e non permette di ottenere point cloud regolari a più di due metri di distanza dal sensore. In Figura 5.5(a) viene riportata una point cloud di una persona a circa tre metri di distanza dal sensore. Come si può notare la superficie della point cloud risulta suddivisa in fette prodotte dai passi di quantizzazione. Per mitigare questo problema è stato impiegato il metodo *Moving Least Squares* (MLS) per la ricostruzione di superfici, il risultato è riportato in Figura 5.5(b). MLS è un metodo di ricostruzione di funzioni continue da un insieme di campioni di punti non organizzati tramite il calcolo dei minimi quadrati in un intorno del punto in cui viene richiesto il valore. In dettaglio, si consideri la funzione $f : \mathbb{R}^n \to \mathbb{R}$ e un insieme di punti $S = (x_i, f_i)|f(x_i) = f_i$ dove $x_i \in \mathbb{R}^n$ e gli f_i sono numeri reali. L'approssimazione moving least squares di grado m nel punto x è $\tilde{p}(x)$ dove $\tilde{p}$ minimizza l'errore dei minimi quadrati

$$\sum_i (p(x) - f_i)^2 \theta(||x - x_i||) \tag{5.3}$$

su tutti i polinomi di grado m in $\mathbb{R}^n$ e $\theta(s)$ tende a zero per $s \to \infty$.

(a) (b)

Figura 5.5: Modello prima e dopo lo smoothing MLS



(a) (b) (c)

(d) (e) (f)

Figura 5.6: Esempi di modelli. (a), (b) e (c) sono stati creati a partire da dati raccolti con il Kinect for Windows SDK mentre (d), (e) e (f) sono stati raccolti con OpenNI.

Capitolo 6

Re-identificazione di persone

La trasformazione a posa standard non è utile solamente perchè permette di sommare point cloud di persone indipendentemente dalla loro posa iniziale per la creazione di modelli, ma anche perchè permette di impiegare questi modelli per la comparazione a scopo di re-identificazione con nuove point cloud di test. Il framework sviluppato permette di identificare un soggetto in piedi di fronte ad una depth camera, prendendo in considerazione un singolo fotogramma di input. Per ottenere questo obiettivo vengono comparate point cloud di persone in posa standard con un modello di riferimento della persona generato con la tecnica descritta in Sezione 5.3.

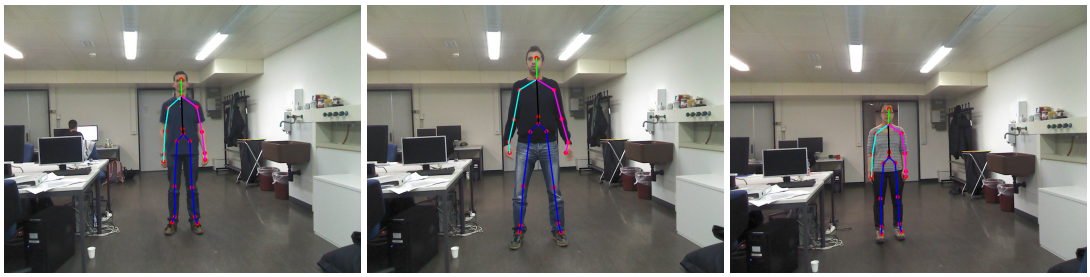
6.1 RGB-D people re-identification dataset

La tecnica di re-identificazione descritta nella Sezione 6.2 è stata testata con due dataset, il primo *BIWI RGBD-ID* dataset è stato raccolto con il Kinect for Windows SDK, mentre il secondo *IAS-Lab RGBD-ID* dataset con il framework OpenNI.

6.1.1 Dataset BIWI RGBD-ID

Il dataset *BIWI RGBD-ID* [10] è composto da 50 sequenze video di persone diverse registrate impiegando il Kinect for Windows SDK. Il dataset include immagini RGB (registrate a risoluzione 1280x960), depth map, user map, dati sullo scheletro e coordinate del piano del pavimento. Questi video sono stati acquisiti a circa 8-10fps e sono della durata di circa un minuto per ogni persona. Inoltre il dataset comprende anche 56 sequenze di test con 28 persone già presenti nel dataset. Queste sequenze sono state registrate in una stanza diversa rispetto alle prime 50 sequenze ed inoltre le persone sono vestite in maniera differente. Per ogni persona sono presenti una prima sequenza, *Still*, in cui la persona è ferma in piedi davanti alla telecamera e una

seconda sequenza, *Walking*, in cui la persona effettua due camminate frontali e due camminate diagonali verso la Kinect.



(a) (b) (c)

Figura 6.1: Esempi di immagini RGB con la stima dello scheletro del dataset *BIWI RGBD-ID*.

(a) (b) (c)

Figura 6.2: Esempi di immagini RGB con la stima dello scheletro del dataset *OpenNI*.

6.1.2 Dataset IAS-Lab RGBD-ID

Il dataset I*AS-Lab RGBD-ID*, registrato con OpenNI, comprende tre sequenze video di 11 persone che si muovono di fronte al sensore Kinect. La prima sequenza (*Training*) e la seconda sequenza (*Room*) sono state registrate in due stanze diverse, mentre la terza (*Outfit*) è stata registrata nella stessa stanza della prima sequenza ma le persone indossano vestiti diversi. Ogni sequenza è composta dalle seguenti azioni:

- rotazione della persona a 360°
- due camminate frontali avanti/indietro
- una camminata diagonale
- una camminata libera

Le sequenze sono della durata di circa un minuto e mezzo e includono immagini RGB (registrate a risoluzione 640x480), depth map, user map, dati dello scheletro.

6.2 Person Point Cloud Matching

L'approccio alla re-identificazione proposto prende in considerazione la forma dell'intera point cloud della persona per il compito di re-identificazione. Per ogni frame di test viene creata la corrispondente point cloud in posa standard, filtrata mediante voxel grid e smoothing MLS. A questo punto la point cloud viene allineata con ICP e confrontata con tutti i modelli di persone generati a partire dal training set. Ad ogni iterazione viene calcolata la fitness score tra la cloud corrente e i modelli. La fitness score impiegata come metrica di similarità è definita come la distanza media dei punti della prima cloud al più vicino punto della seconda cloud. In dettaglio, siano P_1 e P_2 le due point cloud, il fitness score di P_2 rispetto a P_1 è definito come:

$$f_{2\to1} = \sum_{p_1 \in P_2} ||p_i - q_i^*|| \tag{6.1}$$

dove q_i^* viene definito come:

$$q_i^* = arg \min_{p_1 \in P_1} ||p_i - q_j|| \tag{6.2}$$

Questa funzione permette quindi di confrontare la forma del corpo dei soggetti rappresentati dalle cloud definendo una metrica sulla distribuzione dei punti della cloud nello spazio.

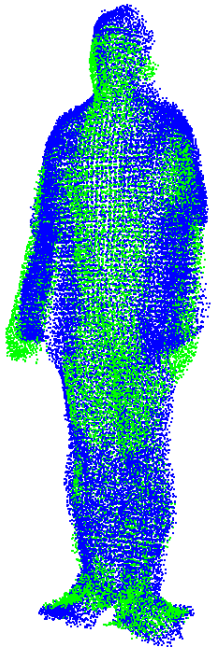


Figura 6.3: Point cloud matching con allineamento ICP.

6.3 Test e risultati

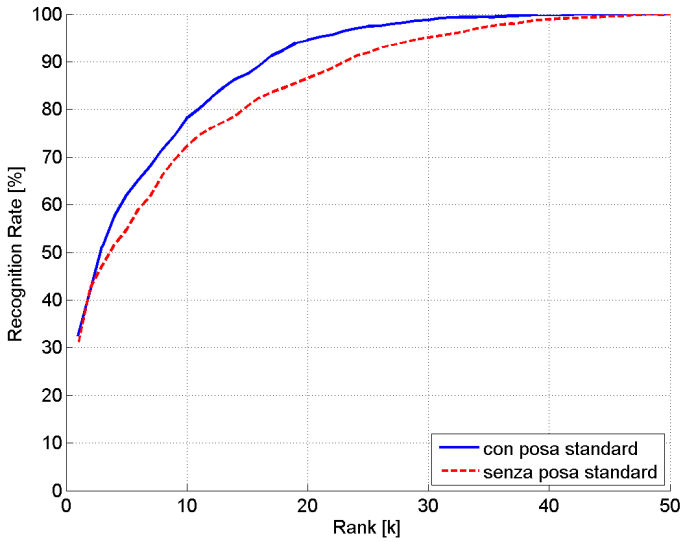
Per la valutazione dei risultati di re-identificazione, sono state calcolate le *Cumulative Matching Curves* (CMC) [7]. Per ogni k da 1 al numero di soggetti di training, queste curve esprimono l'indice di riconoscimento medio calcolato considerando la classificazione corretta se la persona da riconoscere appare tra i soggetti che hanno ottenuto i k migliori score di classificazione. I parametri tipici di valutazione per queste curve sono l'indice di riconoscimento *rank-1* e la *normalized Area Under Curve* (nAUC) che corrisponde all'integrale della CMC. Gli indici di riconoscimento vengono calcolati separatamente per ogni soggetto nel dataset e poi mediati per ottenere l'indice di riconoscimento globale. Vengono inoltre riportati gli istogrammi del rank medio per ogni persona delle sequenze di test, ovvero il rank medio al quale la persona viene correttamente classificata. I valori mancanti nell'asse delle ascisse dipendono dal fatto che non tutti i soggetti di training sono presenti nel dataset di test.

6.3.1 Risultati - BIWI RGBD-ID dataset

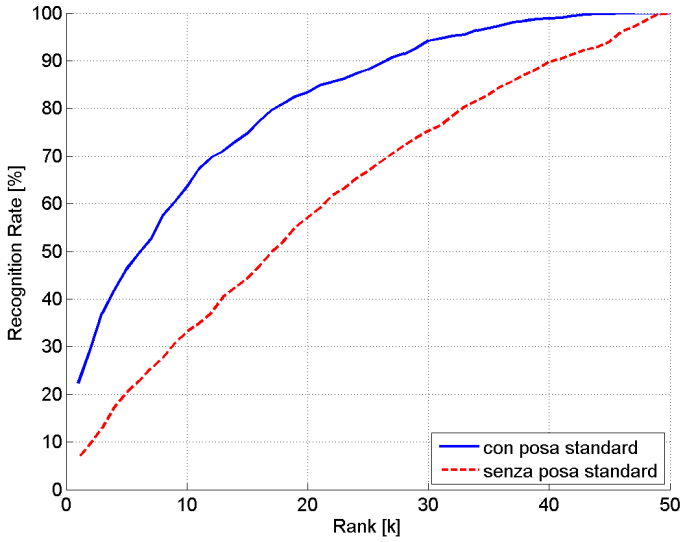
La Tabella 6.1 riporta sinteticamente i risultati ottenuti dall'algoritmo di point cloud matching sul dataset *BIWI RGBD-ID*. In Figura 6.4 viene comparato il metodo proposto con un metodo per il matching simile che però non sfrutta la trasformazione a posa standard. Per il test set in cui le persone sono ferme in posa frontale (*Still*) la differenza di prestazioni è minima in quanto le persone vengono riprese sempre nella stessa posa. Al contrario per il test set con persone in movimento l'applicazione della trasformazione a posa standard permette di migliorare le prestazioni di riconoscimento raggiungendo un rank-1 del 22.38% e una nAUC di 81.56% contro rispettivamente 6.61% e 62.61%. È importante sottolineare che anche in questa fase di testing vengono considerati validi solamente i frame nei quali viene rilevata la faccia del soggetto ripreso come avviene durante la creazione del modello a causa delle già descritte limitazioni dello skeletal tracker Microsoft.

	rank-1	**nAUC**
Cross validation	93.72%	99.62%
Still	32.49%	89.04%
Walking	22.38%	81.56%

Tabella 6.1: Risultati dataset BIWI RGBD-ID.

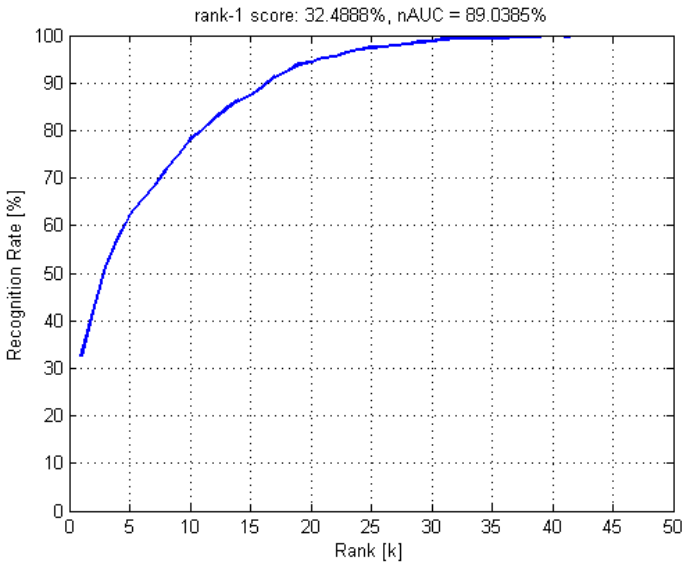


(a) Still

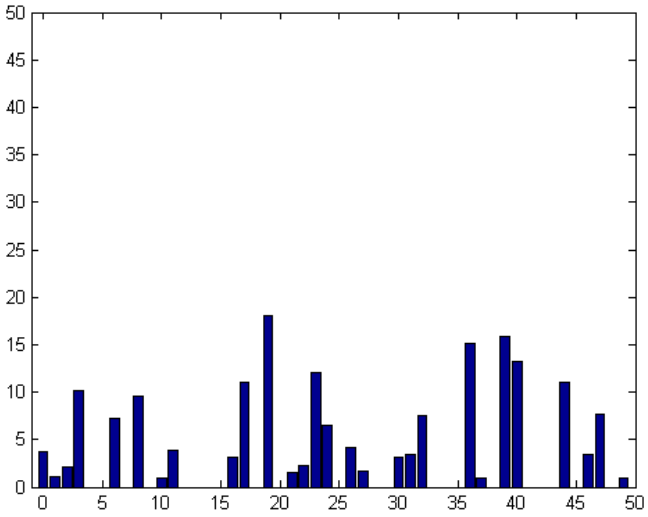


(b) Walking

Figura 6.4: Cumulative Matching Curves ottenute con l'approccio di point cloud matching con e senza trasformazione a posa standard sulle sequenze di test del *BIWI RGBD-ID* dataset.

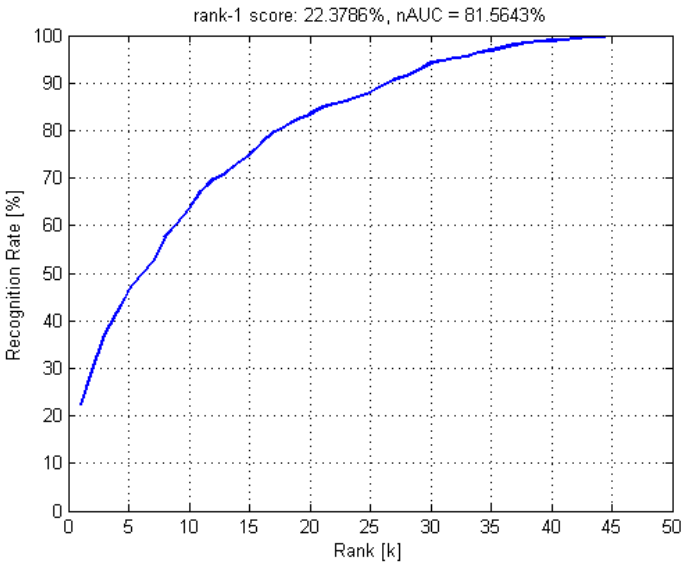


(a)

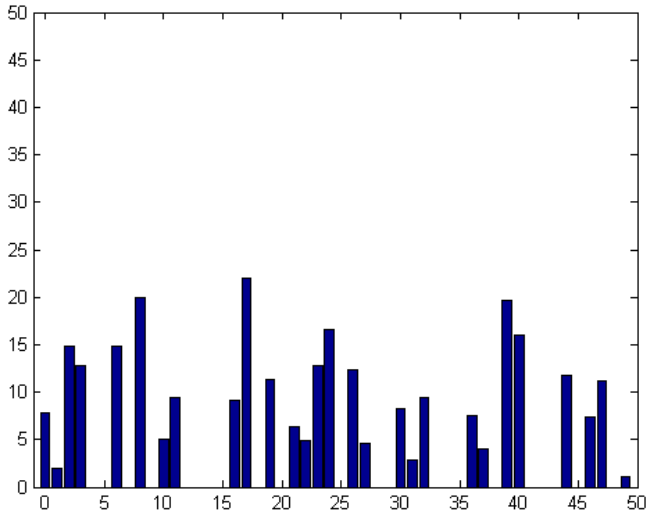


(b)

Figura 6.5: (a) CMC e (b) istogramma rank medio ottenuti dai risultati sulla sequenza *Still* del *BIWI RGBD-ID* dataset.



(a)



(b)

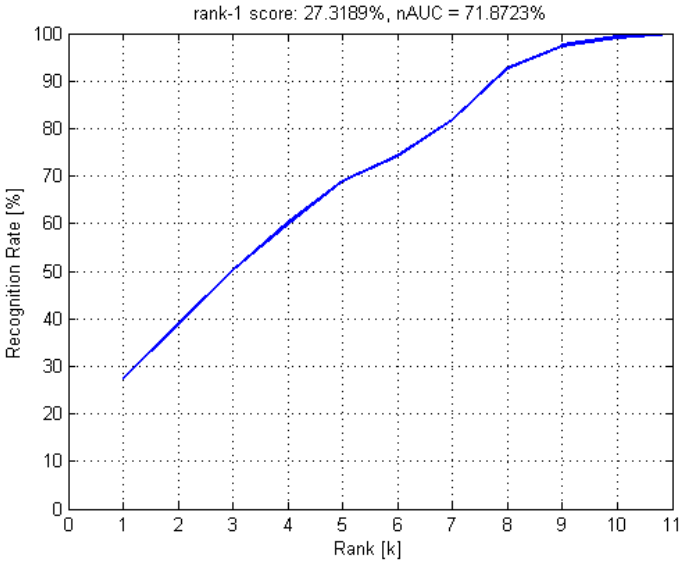
Figura 6.6: (a) CMC e (b) istogramma rank medio ottenuti dai risultati sulla sequenza *Walking* del *BIWI RGBD-ID* dataset.

6.3.2 Risultati - IAS-Lab RGBD-ID dataset

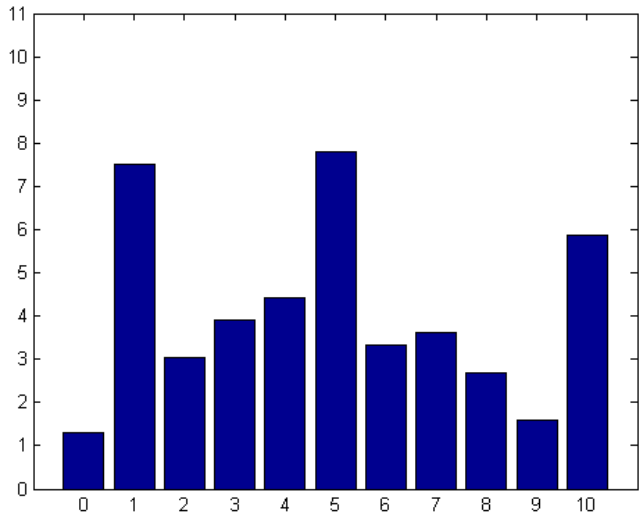
La Tabella 6.2 riporta sinteticamente i risultati ottenuti dall'algoritmo di point cloud matching sul dataset *IAS-Lab RGBD-ID*. Su questo dataset i test sono stati eseguiti dapprima con le modalità descritte per il dataset *BIWI RGBD-ID* impiegando l'algoritmo di face detection per filtrare i frame in cui la persona è di spalle e successivamente su tutti i frame validi, sia frontali che posteriori, sfruttando appieno le potenzialità dello skeletal tracker OpenNI.

	rank-1	**nAUC**
Cross validation	91.99%	98.73%
Outfit con face detection	27.32%	71.87%
Room con face detection	49.39%	83.31%
Outfit	28.61%	73.16%
Room	43.69%	81.65%

Tabella 6.2: Risultati dataset *IAS-Lab RGBD-ID*.

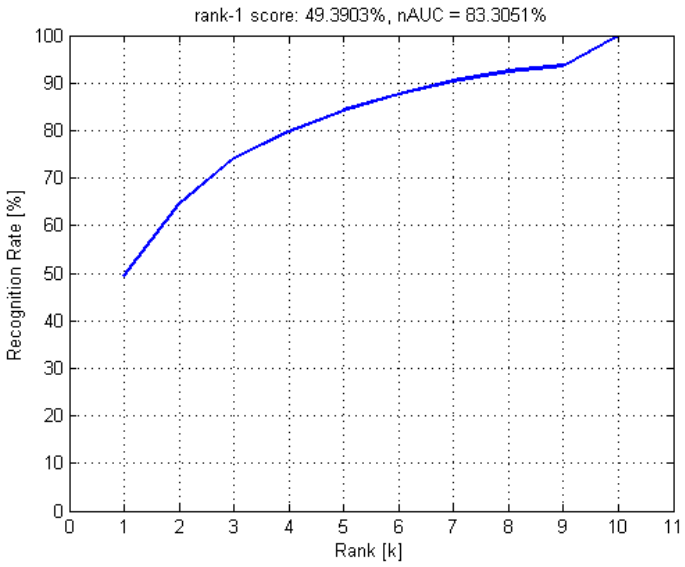


(a)

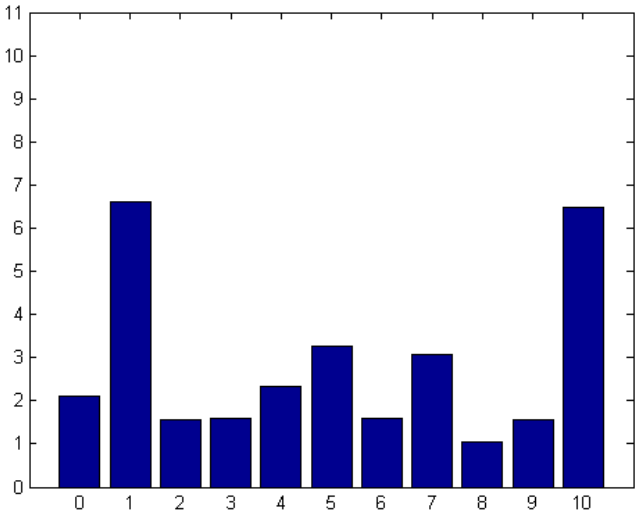


(b)

Figura 6.7: (a) CMC e (b) istogramma rank medio ottenuti dai risultati sulla sequenza *Outfit* del dataset *IAS-Lab RGBD-ID* filtrando i frame con l'algoritmo di face detection.

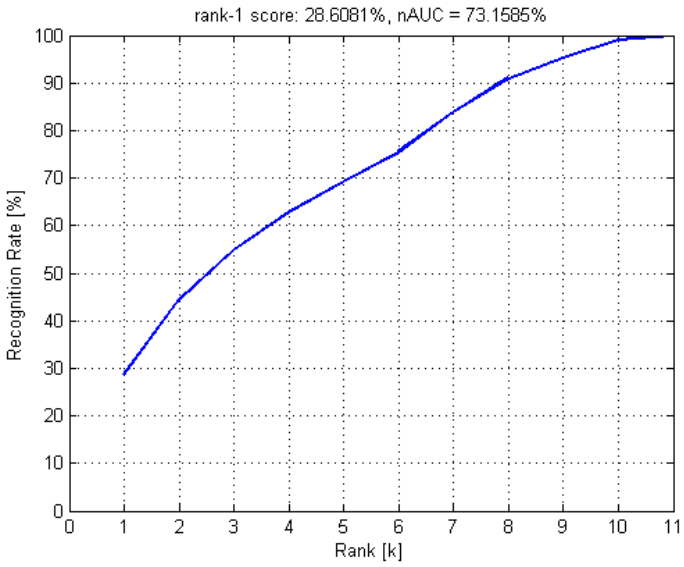


(a)

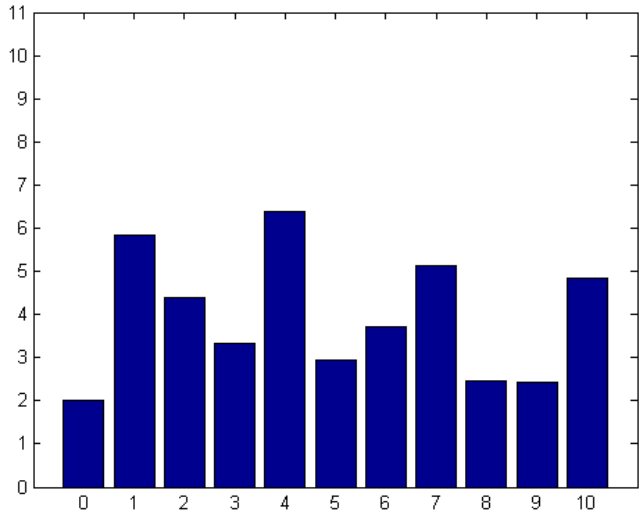


(b)

Figura 6.8: (a) CMC e (b) istogramma rank medio ottenuti dai risultati sulla sequenza *Room* del dataset *IAS-Lab RGBD-ID* filtrando i frame con l'algoritmo di face detection.

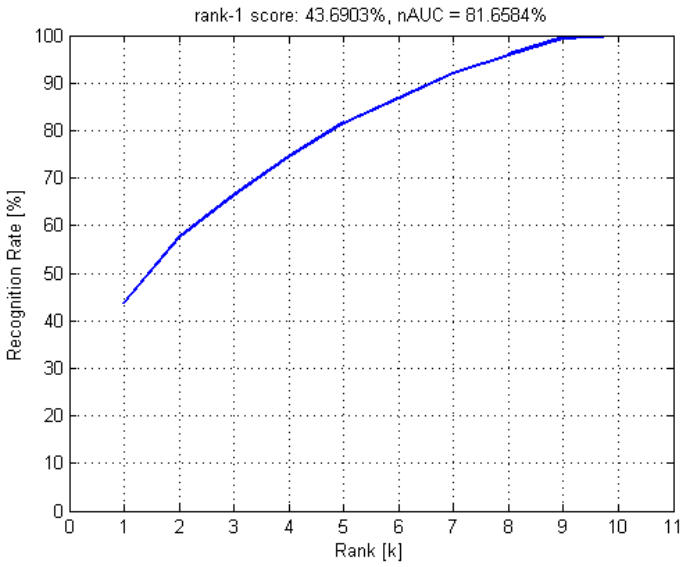


(a)

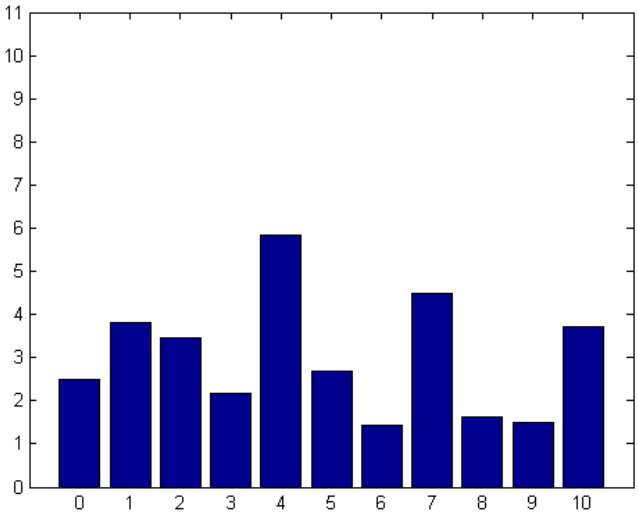


(b)

Figura 6.9: (a) CMC e (b) istogramma rank medio ottenuti dai risultati sulla sequenza *Outfit* del dataset *IAS-Lab RGBD-ID*.



(a)



(b)

Figura 6.10: (a) CMC e (b) istogramma rank medio ottenuti dai risultati sulla sequenza *Room* del dataset *IAS-Lab RGBD-ID*.

6.3.3 Risultati Multi-frame

Il Person Point Cloud Matching implementa un metodo di re-identificazione one-shot a partire dai singoli frame. D'altra parte, se sono disponibili più frame della persona da riconoscere è possibile identificare i soggetti applicando un approccio multi-frame, impiegando uno schema di voto che associa ad ogni sequenza di test la persona che l'algoritmo identifica con il minor fitness score nel maggior numero di frame. In Tabella 6.3 e Tabella 6.4 vengono comparati i risultati ottenuti sulle sequenze di test di entrambi i dataset con l'approccio a frame singolo e multi-frame. Come si può notare l'incremento prestazionale medio è del 10-20%.

BIWI RGBD-ID	**Single**	**Multi**
Cross validation	93.72%	100%
Still	32.49%	42.86%
Walking	22.38%	39.29%

Tabella 6.3: Risultati multi frame dataset BIWI RGBD-ID.

IAS-Lab RGBD-ID	**Single**	**Multi**
Cross validation	91.99%	100%
Outfit con face detection	27.32%	36.36%
Room con face detection	49.39%	72.73%
Outfit	28.61%	63.64%
Room	43.69%	72.73%

Tabella 6.4: Risultati multi frame dataset *IAS-Lab RGBD-ID*.

6.3.4 Comparazione tra PPCM e Skeleton Descriptor

In questa sezione vengono comparati i risultati di re-identificazione dell'algoritmo di Person Point Cloud Matching e di un altro descrittore soft biometric che impiega informazioni sullo scheletro, entrambi ottenuti con il BIWI RGBD-ID dataset. In particolare questo descrittore si basa sul calcolo di alcune lunghezze e rapporti di lunghezza, ottenuti dalla posizione 3D dei joint del corpo forniti dallo skeletal tracker. Le lunghezze prese in considerazione sono le seguenti: altezza testa, altezza collo, distanza tra collo e spalla destra, distanza tra collo e spalla sinistra, distanza da torso e spalla destra, lunghezza braccio destro, lunghezza braccio sinistro, lunghezza gamba destra, lunghezza gamba sinistra, lunghezza torso, distanza tra anca destra e anca sinistra, rapporto tra la lunghezza del torso e lunghezza della gamba destra, rapporto tra la lunghezza

	Cross validation		Test - Still		Test - Walking	
	Rank-1	**nAUC**	**Rank-1**	**nAUC**	**Rank-1**	**nAUC**
Skeleton (NN)	80.5%	98.2%	26.6%	89.7%	21.1%	86.6%
Point cloud matching	93.7%	99.6%	32.5%	89.0%	22.4%	81.6%

Tabella 6.5: Comparazione risultati ottenuti in cross validation e con i testing set del *BIWI RGBD-ID* dataset.

del torso e la lunghezza della gamba sinistra. Tutte queste lunghezze vengono concatenate in un singolo descrittore dello scheletro e viene poi impiegato un classificatore Nearest Neighbor basato sulla distanza euclidea per la comparazione dei descrittori estratti dal training set con quelli dei testing set. Come si può osservare dalle CMC in Figura 6.11 la tecnica di point cloud matching ha performance inferiori rispetto al descrittore dello scheletro, tuttavia la differenza tra le due tecnica risulta minima. Quello che è importante evidenziare è invece che le due tecniche sono in qualche modo complementari in quanto i due differenti approcci portano a errori su soggetti diversi, come mostrato negli istogrammi di rank medio in Figura 6.11

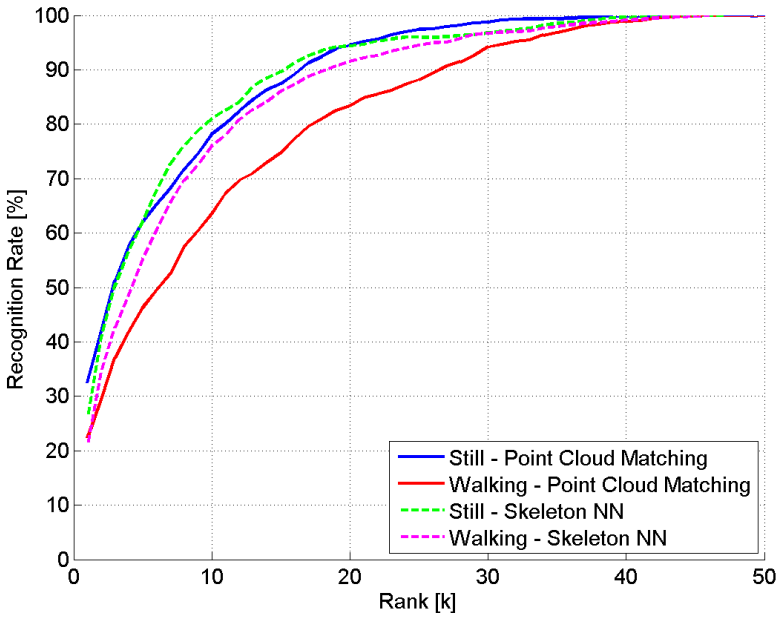
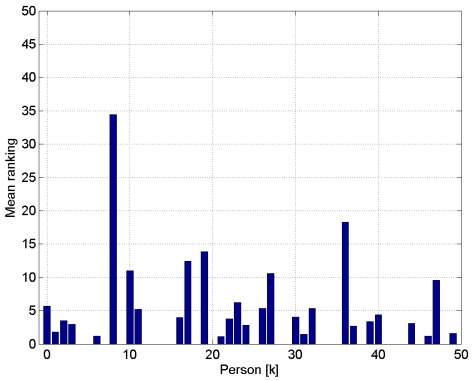
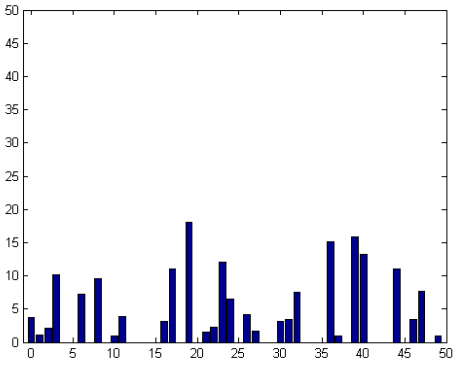


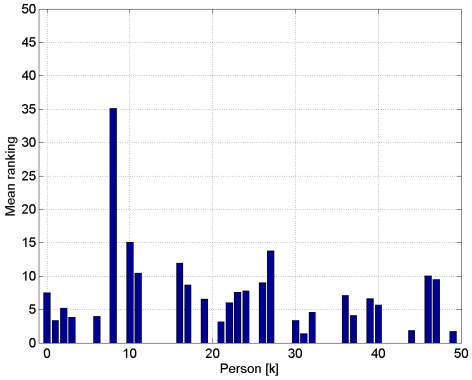
Figura 6.11: Comparazione CMC ottenute con l'algoritmo di Person Point Cloud Matching e il descrittore dello scheletro sui testing set del *BIWI RGBD-ID* dataset.



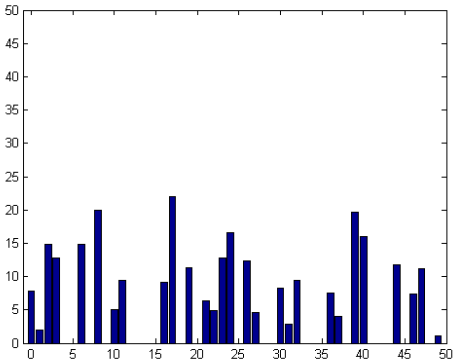
(a) Skeleton (NN)



(b) Point cloud matching



(c) Skeleton(NN)



(d) Point cloud matching

Figura 6.12: Istogrammi di rank medio ottenuti con le due tecniche per ogni persona dei set di test *Still* (a), (b) e *Walking* (c), (d) del dataset *BIWI RGBD-ID*.

6.4 Prestazioni

Nella Tabella 5.2 vengono riportate in dettaglio le prestazioni per l'algoritmo di re-identificazione con la seguente configurazione harware/software ottenute sul BIWI RGB-D dataset:

- CPU: Intel Core i5-3570k @ 3.40Ghz
- RAM: 8 GB DDR3
- SO: Ubuntu 12.04 Precise 64bit
- ROS: Fuerte

L'operazione che richiede più tempo di calcolo è il matching della point cloud trasformata con i modelli di ogni persona del training set, che richiede 250 ms per effettuare 50 comparazioni. Il tempo medio di esecuzione di tutti i passaggi dell'algoritmo è di circa 490ms, che si traduce in un frame rate di 2.08 fps. Questo suggerisce che con ulteriori ottimizzazioni e un numero non troppo elevato di persone nel database questo approccio può essere impiegato in real time.

Operazione	**Tempo di elaborazione (ms)**
Lettura dati scheletro e RGB-D	94.58
Registrazione da depth a RGB	32.01
Segmentatione e labeling	3.03
Face detection	42.19
Trasformazione a posa standard	0.41
Voxel filtering	0.33
SOR filtering	6.40
MLS smoothing	56.35
Allineamento ICP al modello e calcolo fitness score	254.34
Totale	489.63

Tabella 6.6: Tempo medio di elaborazione per le singole operazioni.

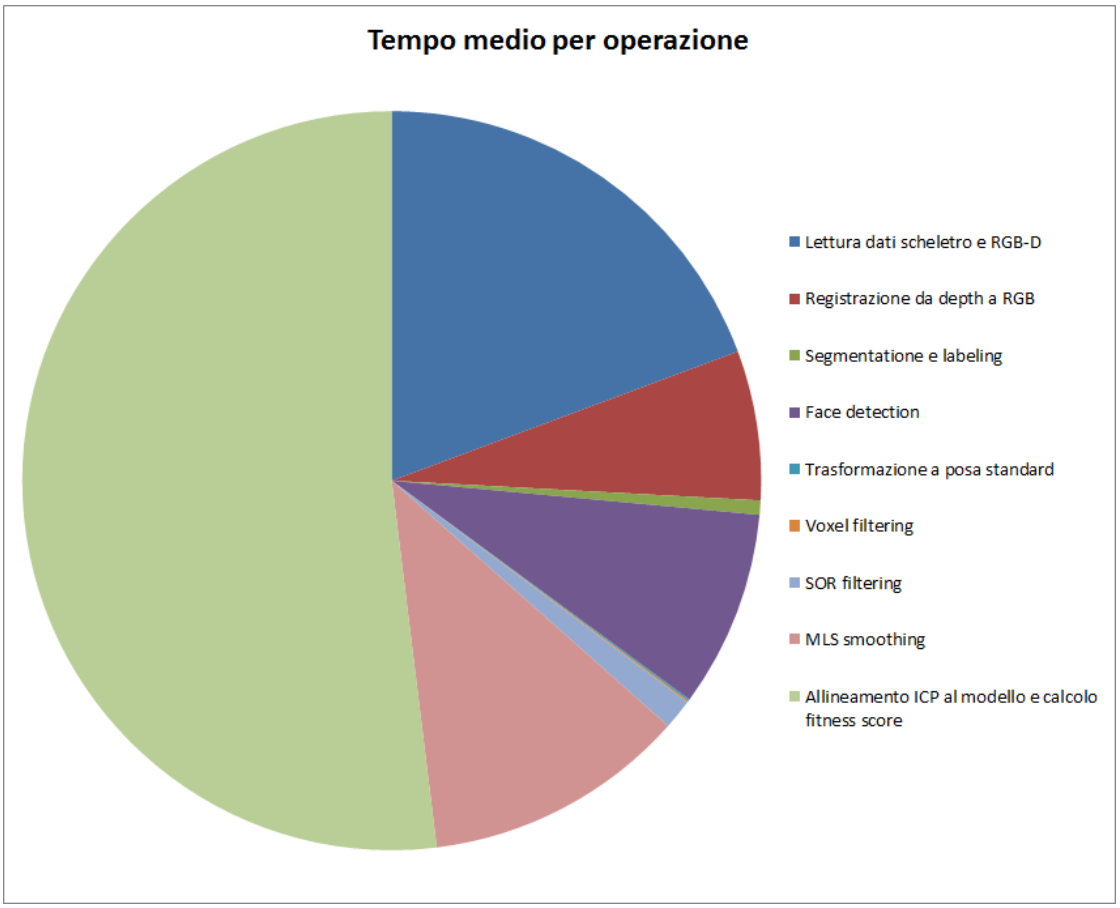


Figura 6.13: Tempo medio di elaborazione per le singole operazioni.

Capitolo 7

Conclusioni

Il contributo di questa tesi è duplice: un algoritmo di re-identificazione e una tecnica per la creazione di modelli 3D di persone che utilizzano i dati RGB-D ottenuti da un sensore Microsoft Kinect. In particolare viene proposto un metodo per la re-identificazione basato sulla forma globale del corpo derivato da tecniche di 3D object recognition. Per questo scopo è stata sviluppata una nuova tecnica, che impiega le informazioni sulla posa del soggetto, per la trasformazione di point cloud di persone ad una nuova posa comune, definita *posa standard*. La posa standard permette una comparazione rigida delle point cloud di soggetti diversi, basata su una tipica fitness score ICP, indipendentemente dalla posa iniziale delle persone. Inoltre, viene illustrato inoltre come impiegare la tecnica di trasformazione a posa standard per ottenere dei modelli 3D di persone da una serie di point cloud di soggetti che si muovono liberamente.
Gli algoritmi proposti sono stati testati su due differenti dataset, *BIWI RGBD-ID* e *IAS-Lab RGBD-ID* dataset. I risultati sperimentali ottenuti mostrano che l'informazione sulla forma del corpo può essere impiegata efficacemente per la re-identificazione risultando robusta sia a variazioni di illuminazione che di abbigliamento dei soggetti.
Le tecniche proposte hanno ovviamente anche alcune limitazioni. Due delle più significative, il campo di vista limitato e le scarse prestazioni in condizioni di forte illuminazione, derivano dal sensore Kinect, mentre l'accuratezza delle informazioni sulla posa, fornite dallo skeletal tracker, influisce direttamente sulle prestazioni di re-identificazione dell'algoritmo.
Infine, come sviluppo futuro, si prevede di combinare i risultati della tecnica di *Person Point Cloud Matching* con la classificazione basata sulle distanze dello scheletro, in modo da sfruttare la complementarietà dei due descrittori evidenziata dagli esiti sperimentali.

Appendice A

Implementazione

A.1 Pacchetti `PeopleModel` e `PeopleReidentification`

Il pacchetto `PeopleModel` si occupa della creazione dei modelli 3D da impiegare poi per la re-identificazione con l'algoritmo di *Person Point Cloud Matching*, a partire dai dati RGB-D e dalle informazioni sullo scheletro. La pipeline di elaborazione si suddivide nelle seguenti operazioni:

1. registrazione depth map e user map nelle coordinate del frame dell'immagine RGB
2. creazione della point cloud a partire dalla depth map registrata e dall'immagine RGB
3. mappatura coordinate 2D e 3D dei joint dello scheletro sul frame RGB
4. segmentazione dell'utente dal background e labelling delle parti del corpo
5. trasformazione in posa standard della point cloud
6. allineamento ICP e aggiunta della cloud corrente in posa standard al modello 3D incrementale
7. filtering (voxel grid, Statistical Outlier Removal e Moving Least Squares smoothing)

I dati di input sono composti nel seguente modo:

- immagine RGB
- depth map
- user segmentation mask
- informazioni skeletal tracker in testo semplice

Nel caso del pacchetto `PeopleReidentification` gran parte delle operazioni rimane invariata, la differenza principale si ha al punto 6, dove la point cloud in posa standard del frame corrente viene registrata tramite ICP a tutti i modelli generati in fase di training con il pacchetto `PeopleModel` e viene calcolata la fitness score di tale allineamento.

A.2 Pacchetti `ActionModel` e `bagToFilesConverter`

Il pacchetto `ActionModel` si occupa di interfacciarsi con il middleware OpenNI per effettuare il tracking dello scheletro. Originariamente questo pacchetto pubblicava sotto forma di transform (`tf`) di ROS le coordinate 3D dei joint e la loro orientazione. Durante il lavoro di tesi le funzionalità di questo pacchetto sono state ampliate per pubblicare anche la *user map* dell'utente in un apposito topic (`/usermap`).
Il pacchetto `bagToFilesConverter` si occupa di creare i file di input elencati in precedenza a partire dai dati provenienti dal sensore RGB e depth e dai dati pubblicati dal pacchetto `ActionModel`. Vengono monitorati i messaggi pubblicati nei seguenti topic:

- `/openni/rgb/image_color`: topic nel quale vengono pubblicate le immagini RGB provenienti dal sensore
- `/openni/depth/image`: topic nel quale vengono pubblicate le depth map provenienti dal sensore
- `/usermap`: topic nel quale viene pubblicata la maschera di segmentazione degli utenti. Questa maschera è un immagine con la stessa risoluzione della depth map (640x480 pixel) nella quale ogni pixel contiene il valore corrispondente all'ID dell'utente al quale appartiene oppure 0 se il pixel corrisponde ad un punto del background.
- `/tf`: per ognuno dei 15 joint dello scheletro fornito dal tracker OpenNI viene pubblicato un messaggio contenente le sue coordinate 3D e l'orientazione del corrispondente link rispetto al frame `openni_rgb_optical_frame`.

I messaggi pubblicati su questi topic vengono sincronizzati e salvati su disco nel formato opportuno: file PGM per immagine RGB, depth e user map e file di testo per le informazioni dello scheletro. In particolare questi file di testo sono organizzati secondo la seguente sintassi:

ID	X_{3D}	Y_{3D}	Z_{3D}	X_{2D}	Y_{2D}	T	Q	A	B	X	Y	Z	W

- `ID` è l'identificativo dell'utente;
- X_{3D}, Y_{3D}, Z_{3D} e X_{2D},Y_{2D} sono rispettivamente le coordinate 3D e 2D del joint;
- `T` è la flag dello stato di tracking (2 `Tracked`, 1 `Inferred`, 0 `Not Tracked`);
- `Q` è una flag non usata inserita per compatibilità con il formato impiegato per il tracker Microsoft nel quale indica se l'utente è parzialmente al di fuori del campo visivo del sensore;
- `A` e `B` joint inziale e finale del link corrente;
- `X`, `Y`, `Z`, `W` quaternione che esprime la rotazione del link identificato da (`A`, `B`).

Appendice B

Configurazione software

B.1 CUDA

1. Installazione driver Nvidia

   ```
   sudo apt-get install nvidia-current nvidia-current-dev
   ```

2. Download CUDA Toolkit e NVIDIA GPU Computing SDK
   Link: http://developer.nvidia.com/cuda/cuda-download

   ```
   cudatoolkit_4.2.9_linux_64_ubuntu11.04.run
   gpucomputingsdk_4.2.9_linux.run
   ```

3. Rimozione precedenti versioni SDK (se presenti) e reboot

   ```
   sudo rm -rf ~/NVIDIA_GPU_Computing_SDK
   sudo rm -rf /usr/local/cuda
   ```

4. Installazione CUDA Toolkit

   ```
   sudo ./cudatoolkit_4.2.9_linux_64_ubuntu11.04.run
   ```

   Il toolkit viene installato per impostazione di default nella directory `/usr/local/cuda`

5. Impostazione variabili d'ambiente Aggiungere a /.bashrc le seguenti righe per sistemi a 32 bit:

   ```
   export PATH=/usr/local/cuda/bin:$PATH
   export LD_LIBRARY_PATH=/usr/local/cuda/lib:$LD_LIBRARY_PATH
   ```

oppure nel caso di sistemi a 64 bit:

```
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
```

6. Installazione del Nvidia GPU Computing SDK

   ```
   ./gpucomputingsdk_4.2.9_linux.run
   ```

   Per evitare problemi è consigliabile non installare come utente root. L'SDK viene installato di default nella directory `/NVIDIA_GPU_Computing_SDK`.

7. Installazione librerie di sistema addizionali

   ```
   sudo apt-get install freeglut3-dev build-essential
   libx11-dev libxmu-dev libxi-dev libgl1-mesa-glx
   libglu1-mesa libglu1-mesa-dev
   ```

B.2 ROS

1. Configurazione del repository ROS

   ```
   sudo sh -c
   'echo "deb http://packages.ros.org/ros/ubuntu precise main" >
   /etc/apt/sources.list.d/ros-latest.list'

   wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
   ```

2. Installazione ROS

   ```
   sudo apt-get update
   sudo apt-get install ros-fuerte-desktop-full
   ```

3. Installazione stack aggiuntivi ROS

   ```
   sudo apt-get install ros-fuerte-openni-camera
    ros-fuerte-openni-launch ros-fuerte-openni-tracker
   ```

4. Impostazione variabili d'ambiente

   ```
   echo "source /opt/ros/fuerte/setup.bash" >> ~/.bashrc
   . ~/.bashrc
   ```

B.3 PCL trunk

1. Download codice sorgente da SVN

   ```
   svn co http://svn.pointclouds.org/ros/branches/
   fuerte/perception_pcl_unstable/perception_pcl_unstable
   ```

2. Copia files nella directory degli stack di ROS

   ```
   cp -r perception_pcl_unstable /opt/ros/fuerte/stacks/
   ```

3. Compilazione

   ```
   rosmake pcl17
   ```

Bibliografia

[1] Baltieri, D.; Vezzani, R.; Cucchiara, R.,: *Sarc3d: a new 3d body model for people tracking and re-identitcation*, in Proceedings of the 16th international conference on Image analysis and processing, ser. ICIAP'11, 2011, pp. 197-206.

[2] Barbosa, B.I.; Cristani, M.; Del Bue, A., Bazzani; L., Murino; V.: *Re-identification with rgb-d sensors*, in First International Workshop on Re-Identification, 2012

[3] Bedagkar-Gala, A.; Shah, S.: *Multiple person re-identification using part based spatiotemporal color appearance model*, in Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pp. 1721 –1728 (2011). DOI 10.1109/ICCVW.2011.6130457

[4] Farenzena, M.; Bazzani, L.; Perina, A.; Murino, V.; Cristani, M.: *Person re-identification by symmetry-driven accumulation of local features*, in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 2360 –2367 (2010). DOI 10.1109/CVPR.2010.5539926

[5] Freedman, B.; Shpunt, A.; Machline, M.; Arieli, Y.: *Depth Mapping Using Projected Patterns*, 2010.

[6] Gandhi, T.; Trivedi, M.M.: *Panoramic Appearance Map (PAM) for Multi-camera Based Person Re-identification*, Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on , vol., no., pp.78,78, Nov. 2006 doi: 10.1109/AVSS.2006.90

[7] Gray, D.; Tao, H.: *Viewpoint invariant pedestrian recognition with an ensemble of localized features*, in Computer Vision ECCV 2008, Lecture Notes in Computer Science, vol. 5302, pp. 262–275. Springer Berlin Heidelberg (2008)

[8] Leyvand, T.; Meekhof, C.; Wei, Y.C.; Sun, J.; Guo, B.: *Kinect identity: Technology and experience*, in Computer 44(4), 94 –96 (2011). DOI 10.1109/MC.2011.114

[9] Liefeng Bo; Lai, K.; Xiaofeng Ren; Fox, D.: *Object recognition with hierarchical kernel descriptors*, in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on , vol., no., pp.1729,1736, 20-25 June 2011 DOI: 10.1109/CVPR.2011.5995719

[10] Munaro, M.; Fossati, A.; Basso, A.; Menegatti, E.; Van Gool, L.: *One-Shot Person Identification with a Consumer Depth Camera*, submitted to Person Re-Identification, Springer, 2013.

[11] Oliver, J.; Albiol, A.; Albiol, A.: *3D descriptor for people re-identification*, in Pattern Recognition (ICPR), 2012 21st International Conference on , vol., no., pp.1395,1398, 11-15 Nov. 2012

[12] Preis, J.; Kessel, M.; Werner, M.; Linnhoff-Popien, C.: *Gait recognition with Kinect*, in Proceedings of the First Workshop on Kinect in Pervasive Computing (2012)

[13] Shotton, J.; Fitzgibbon, A.; Cook, M.; Sharp, T.; Finocchio, M.; Moore, R.; Kipman, A.; Blake, A.: *Real-time human pose recognition in parts from single depth images*, in CVPR, pp. 1297–1304, 2011

[14] Shotton, J.; Girshick, R.; Fitzgibbon, A.; Sharp, T.; Cook, M.; Finocchio, M.; Moore, R.; Kohli, P.; Criminisi, A.; Kipman, A.; Blake, A.: *Efficient Human Pose Estimation from Single Depth Images*, in Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.PP, no.99, pp.1,1, 0

[15] Viola, P.A.; Jones, M.J.: *Robust real-time face detection*, in ICCV, p. 747, 2001

[16] Wang, C., Zhang, J., Pu, J., Yuan, X.,Wang, L.: *Chrono-gait image: A novel temporal template for gait recognition*, in Computer Vision, ECCV 2010 - 11th European Conference on Computer Vision, Proceedings, Lecture Notes in Computer Science, pp. 257–270. Springer-Verlag (2010). DOI 10.1007/978-3-642-15549-9 19. URL http://opus.bath.ac.uk/21847/. 11th European Conference on Computer Vision, ECCV 2010. 5-11 September 2010. Heraklion, Crete, Greece.

[17] Wang, S.; Lewandowski, M.; Annesley, J.; Orwell, J.: *Re-identification of pedestrians with variable occlusion and scale*, in Computer VisionWorkshops (ICCV-Workshops), 2011 IEEE International Conference on, pp. 1876 –1882 (2011). DOI 10.1109/ICCVW.2011.6130477

[18] Zhao, W.; Chellappa, R.; Phillips, P.J.; Rosenfeld, A.: *Face recognition: A literature survey*, ACM Comput. Surv. 35(4), 399–458 (2003). DOI 10.1145/954339.954342. URL http://doi.acm.org/10.1145/954339.954342

Ringraziamenti

Al mio relatore prof. Emanuele Menegatti.

Al dott. Matteo Munaro e a tutto lo IAS-Lab
per avermi assistito in questi mesi.

A tutti coloro che leggeranno questa tesi.