UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Localizzazione e identificazione di persone da robot mobile in ambienti sconosciuti

Relatore: Emanuele Menegatti

Correlatore: Matteo Munaro

Laureando: Gianluca Tartaggia

Data di laurea: 6 luglio 2015

ANNO ACCADEMICO 2014/2015

Alla mia famiglia.

Sommario

La robotica di servizio è un ambito della ricerca che ha riscosso sempre più successo negli ultimi anni. In questa tesi è stato sviluppato un sistema in grado di utilizzare un robot mobile al fine di esplorare un ambiente localizzando e identificando le persone al suo interno. Inoltre è stata esplorata la possibilità di utilizzare tecniche di skeletal recognition per migliorare il tracking di persone da robot mobile. Il framework utilizzato per lo sviluppo del codice è ROS. La realizzazione complessiva ha previsto la collaborazione di algoritmi per la navigazione del robot e per la rilevazione e il riconoscimento delle persone, nonché di sensori laser e RGB-D per le rilevazioni dell'ambiente e delle persone.

Indice

Capitolo 1

Introduzione

La robotica ha raggiunto il suo successo grazie all'industria manifatturiera. I robot utilizzati in questo ambito sono per lo più bracci manipolatori usati prevalentemente nelle linee di montaggio. Questi robot permettono di sostituire l'uomo in quei lavori in cui è necessario eseguire delle operazioni in situazioni pericolose o che richiedono un grande sforzo fisico. Un altro ambito della robotica che si sta affermando sempre di più negli ultimi anni è la robotica di servizio. In questo ambito confluiscono una grande varietà di discipline come la meccanica, l'elettronica, l'informatica e le scienze sociali. A differenza della robotica industriale, lo scopo della robotica di servizio è quello di assistere ed interagire con l'essere umano. Per raggiungere questo scopo sono necessarie due grandi capacità. Un robot di servizio deve essere in grado di muoversi coerentemente con l'ambiente circostante, in modo da non recare danni ad oggetti o persone e deve essere in grado di rilevare e riconoscere le persone nell'ambiente.

Per muovere un robot all'interno di un ambiente tipicamente umano sono richiesti algoritmi in grado di capire dove il robot si trova e capire quale strada è meglio percorrere all'interno della mappa. Inoltre, per poter funzionare in modo autonomo, il robot deve essere in grado di decidere in quali posizioni spostarsi in base alla situazione che ha di fronte. D'altra parte, per far sì che un robot possa interagire con le persone, è necessario disporre di algoritmi in grado di rilevare una persona all'interno di un ambiente o distinguere una persona da altre persone. Queste due capacità sono state rese possibili dagli avanzamenti nell'ambito della robotica e della computer vision negli ultimi decenni.

In questa tesi è stato proposto un sistema in grado di unire questi due aspetti al fine di far interagire un robot mobile con gli utenti presenti in un ambiente. La tesi è stata divisa in due parti. La prima parte ha previsto la creazione di un sistema in grado di localizzare e re-identificare delle persone all'interno di un ambiente inizialmente sconosciuto. Questa parte consiste nel localizzare le persone durante l'esplorazione dell'ambiente, nel re-identificarle a lungo termine e infine seguirle con un algoritmo di people tracking. Nella seconda parte è stato proposto l'utilizzo di una tecnica di re-identificazione a breve termine per migliorare il tracking di persone usato nella prima parte.

1.1 Stato dell'arte

In questa sezione saranno proposti gli stati dell'arte per entrambe le parti sviluppate.

1.1.1 Ricerca di persone all'interno di un ambiente

I lavori presenti nello stato dell'arte riguardano contesti come la ricerca di persone all'interno di ambienti USAR(Urban search and rescue)[1], la sorveglianza [2],[4] e l'assistenza agli anziani [3],[5].

L'obiettivo della ricerca di persone in ambienti USAR è il soccorso tempestivo in quelle aree sconvolte da eventi naturali e che mettono a rischio la vita dei soccorritori. La ricerca di persone in questo contesto è resa complicata dal fatto che le vittime potrebbero essere bloccate in spazi molto ristretti e quindi non facilmente rilevabili. Inoltre gli ambienti da esplorare potrebbero presentare conformazioni non planari. In [1] il sistema proposto consiste nell'utilizzo di un modulo di object detection eseguito parallelamente all'esplorazione. L'esplorazione è in grado di costruire una mappa a partire dai dati di un laser scanner e di un sensore RGB-D per tenere in considerazione possibili cavità poste a qualsiasi altezza.

La ricerca di persone per i sistemi di videosorveglianza consiste nel cercare degli intrusi all'interno di un ambiente o rilevare situazioni anomale. In [2] e [3] il sistema proposto prevede l'utilizzo di un sensore acustico per rilevare delle posizioni all'interno di un ambiente in cui è stato rilevato un suono anomalo. Una volta giunto sul posto, il sistema utilizza delle tecniche di background subtraction seguito da tecniche di machine learning per rilevare delle situazioni anomale come la caduta di una persona. Il sistema può passare anche in modalità esplorazione utilizzano delle tecniche di face detection e face recognition per rilevare e riconoscere le persone nell'ambiente. In [4] il sistema proposto posiziona il robot in corrispondenza di alcuni punti della mappa prefissati ed esegue un confronto con l'immagine della stessa scena acquisita in un momento precedente. Il confronto viene effettuato utilizzando PCA-SIFT per trovare le corrispondenze tra le due immagini seguito da una tecnica di segmentazione del colore per rilevare le parti differenti nelle due immagini. Infine, se sono state rilevate delle differenze nelle due immagini, si cerca di controllare se sono presenti persone da seguire attraverso un leg detector.

Per quanto riguarda l'assistenza agli anziani, i lavori presenti nello stato dell'arte si occupano di minimizzare il tempo necessario al ritrovamento della persona. Dato che una ricerca casuale non fornisce la soluzione migliore, in [5] e [6] sono stati utilizzati dei modelli probabilistici per guidare il posizionamento del robot. L'idea seguita in generale consiste nell'associare una probabilità ad ogni posizione in cui muovere il robot. La probabilità dipende dal numero di volte in cui una persona è stata vista in una certa posizione. In [5] le persone vengono rilevate durante la navigazione attraverso un leg detector insieme ad un face detector. Se uno di questi due moduli segnala una rilevazione, viene utilizzato un hog people detector per controllare più accuratamente la presenza di una persona. Se anche questo modulo fornisce un risultato positivo, allora si procede ad una fase di interazione con l'utente. In [6] la rilevazione di un utente viene effettuata utilizzando una

telecamera PTZ in combinazione con un face detector. La rilevazione viene effettuata solo dopo che il robot ha raggiunto una posizione di osservazione.

1.1.2 Re-identificazione di persone a breve termine

La re-identificazione di persone è un ambito della computer vision che si occupa del riconoscimento di una persona a partire da immagini video. Il problema della re-identificazione è un problema ancora aperto dato che una persona può essere ripresa da molti punti di vista, in condizioni di occlusioni parziali, con forti variazioni di luminosità e con risoluzioni video molto basse. Attualmente la re-identificazione di persone si divide in tecniche di riconoscimento a breve e a lungo termine. Il lavoro si è concentrato soprattutto sulla re-identificazione a breve termine.

Le tecniche maggiormente utilizzate riguardano il colore [7], [8], le texture [9] o la costruzione di un modello [10] dei soggetti analizzati. In [7] è stato proposto un metodo che si basa sul concetto di simmetria e asimmetria di una persona. Inizialmente vengono calcolati due assi orizzontali sulla base dell'asimmetria dei colori. Questi due assi suddividono la persona in testa, torso e gambe. La testa non viene considerata a causa del suo basso contenuto informativo. Sul torso e sulle gambe viene calcolato un asse di simmetria. Per ogni regione calcolata vengono valutati tre aspetti complementari del colore: il contenuto cromatico usando un istogramma di colore HSV, la dislocazione del colore attraverso il concetto di Maximally Stable Color Regions (MSCR), e la presenza di strutture altamente ricorrenti(RHSP). I valori calcolati per ogni caratteristica sono pesati in base alla distanza dal loro asse di simmetria per attenuare il più possibile la presenza di rumore.

In [8] la re-identificazione si basa sul concetto di Pictorial Structure. Per ogni soggetto vengono individuate le diverse parti del corpo che lo compongono per poi formare l'intera struttura della persona. Per ogni parte vengono calcolati il contenuto cromatico del colore usando un istogramma di colore HSV e la dislocazione del colore attraverso il concetto di Maximally Stable Color Regions (MSCR). Questi valori vengono concatenati e usati per il confronto tra immagini differenti.

In [9] è stato proposto un metodo che utilizza sia le informazioni del colore e sia le informazioni estratte dalle texture per re-identificare una persona. A differenza dagli altri metodi queste informazioni vengono estratte solo su alcuni punti specifici delle persone. Inoltre le informazioni del colore sono calcolate considerando la tonalità per ottenere l'invarianza alla luminosità e alla gamma. Le informazioni sulla texture vengono ottenute attraverso il descrittore SURF.

In [10] il metodo proposto prevede la creazione di un modello tridimensionale della persona. A partire da un sensore RGB-D, i punti associati ad ogni persona sono convertiti in coordinate cilindriche rispetto al loro centro di massa. L'angolo zero corrisponde alla direzione del moto. Ogni punto viene suddiviso in intervalli sulla base dell'altezza e dell'angolo. Per ogni intervallo viene calcolato il numero di punti contenuti e il valore medio del colore associato. Infine, nella fase di associazione, viene applicata una normalizzazione sul colore al fine di rendere robusto il modello alle variazioni di luminosità.

1.2 Struttura del documento

Nel capitolo 2 verranno date informazioni sulla struttura hardware e software utilizzata. Nel capitolo 3 sarà spiegato come è stato implementato il sistema in grado di localizzare e re-identificare le persone durante l'esplorazione di un ambiente sconosciuto. Nel capitolo 4 sarà proposto un metodo innovativo per il miglioramento del tracking di persone usato nella prima parte. Infine nel capitolo 5 saranno riportate le conclusioni del lavoro svolto e i possibili sviluppi futuri.

Capitolo 2

Framework e hardware utilizzati

2.1 ROS

ROS(Robot Operating System) [26] è un framework open source creato per lo sviluppo di software per la robotica. La versione di ROS utilizzata per questo progetto è Hydro. ROS mette a disposizione degli sviluppatori funzionalità come astrazione hardware, controllo di dispositivi a basso livello, comunicazione tra nodi e gestione di pacchetti.

2.1.1 Concetti fondamentali di ROS

ROS si basa su una struttura a grafo i cui processi possono scambiarsi messaggi attraverso lo stile architetturale publisher/subscriber. I concetti fondamentali su cui ROS si basa sono:

- **Nodi**: un nodo è un processo che viene eseguito all'interno di ROS. Il nome nodo deriva dalla rappresentazione a grafo di un sistema a tempo di esecuzione. Un nodo può essere collegato con qualsiasi altro nodo specificando un messaggio da inviare e il topic su cui pubblicare o ricevere il messaggio.

- **Messaggi**: un messaggio è una struttura dati che può contenere tipi di dato primitivi(int, float, double, bool), array e tipi predefiniti. Un messaggio può contenere altri messaggi permettendo ad uno sviluppatore di creare messaggi personalizzati.

- **Topic**: ogni nodo può scambiare informazioni con qualsiasi altro nodo. Lo scambio di informazioni avviene specificando il nome di un topic, cioè una stringa che identifica simbolicamente il luogo di scambio.

- **Servizi**: un servizio è un altro metodo per scambiare messaggi in ROS. A differenza dei topic, che implementano uno scambio asincrono, i servizi si basano sull'interazione request/reply. La comunicazione avviene inviando un messaggio ben formattato contenente i campi di richiesta e i campi di risposta.

- **Azioni**: le azioni permettono di avere un controllo maggiore sulla richiesta di un servizio. Le azioni permettono di richiedere ad un nodo lo svolgimento di un servizio e controllare lo stato dell'esecuzione in corso.

Per poter supportare lo sviluppo di un progetto, ROS mette a disposizione degli sviluppatori alcuni strumenti grafici tra cui rqt_graph. rqt_graph permette di avere una visione del grafo contenente i nodi in esecuzione. Un esempio di grafo visualizzato in rqt_graph è dato in figura 2.1.

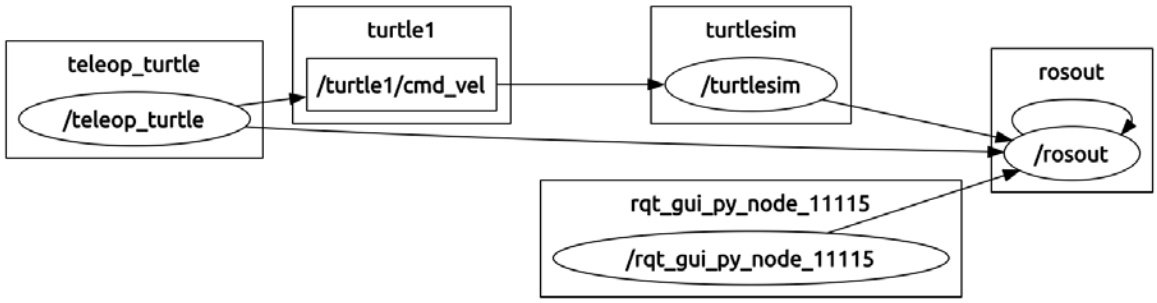


Figura 2.1: Grafo che rappresenta la comunicazione tra il nodo teleop_turtle e il nodo turtlesim attraverso il topic /turtle/cmd_vel.

2.1.2 Pacchetti

Per la collaborazione e lo sviluppo di diversi progetti, ROS organizza la sua struttura in pacchetti. Un pacchetto è una cartella contenente diversi file che specificano l'intero progetto. I file contenuti riguardano il codice sorgente, un file CMakeLists per la compilazione, un file XML per specificare le dipendenze da altri pacchetti, i file per definire la struttura dei messaggi, i file per definire i parametri in ingresso di un nodo e i file utilizzati per eseguire i nodi del pacchetto.

2.1.3 TF

Nell'ambito dello sviluppo software per la robotica un aspetto importante è la trasformazione tra sistemi di riferimento. Questi sistemi di riferimento possono riguardare le coordinate e l'orientazione dei giunti di un robot oppure la localizzazione di una piattaforma mobile rispetto ad un sistema di riferimento fisso. Una TF(Transform Frame) specifica la trasformazione tra due sistemi di riferimento attraverso un vettore di traslazione e un quaternione per indicare la rotazione. In ROS la gestione delle TF avviene attraverso una struttura ad albero che consente allo sviluppatore di accedere ai valori di trasformazione tra frame sia in fase di sviluppo e sia in fase di debugging.

2.1.4 Rviz

Rviz è uno strumento grafico in grado di renderizzare una vasta gamma di oggetti. E' costituito da una finestra in cui vengono selezionati gli oggetti da visualizzare e una finestra per l'effettiva visualizzazione. Di fatto Rviz si comporta come un nodo ROS e può quindi ricevere le informazioni da visualizzare come messaggi attraverso dei topic. In generale è possibile visualizzare forme, immagini, nuvole di punti, sistemi di riferimento, dati di rilevazione ambientali, mappe dell'ambiente, percorsi e altro ancora.

2.2 OpenNI e Nite Skeletal Tracker

OpenNI(Open Natural Interaction) è un middleware utilizzato per ricevere ed elaborare i dati provenienti dai dispositivi per l'interazione naturale come Microsoft Kinect o Asus Xtion Live. OpenNI mette a disposizione i driver necessari ad acquisire i dati video e delle librerie per lo sviluppo di applicazioni per il riconoscimento vocale, il rilevamento delle mani e dello scheletro.

In questa tesi è stato utilizzato Nite Skeletal Tracker(NST) come algoritmo per il rilevamento dello scheletro delle persone. Questo algoritmo è in grado di restituire 15 giunti(figura 2.2) ad una frequenza di 30 frame per secondo. Il suo funzionamento si basa sull'immagine di profondità acquisita dalla Kinect(per ulteriori dettagli vedere 2.3.3) insieme a delle tecniche di machine learning per inferire la posizione delle parti del corpo di una persona.

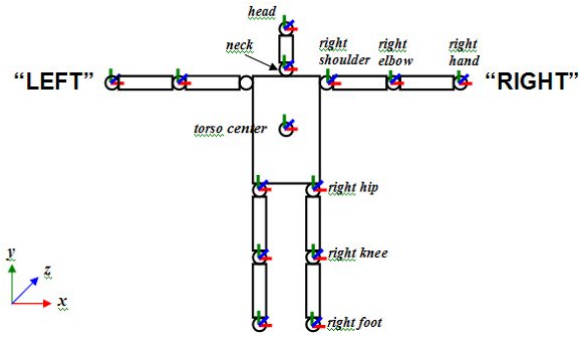


Figura 2.2: Nomi e posizioni dei giunti calcolati da Nite skeletal tracker

2.3 Hardware

2.3.1 Pioneer3AT

Per lo sviluppo di questo lavoro è stato utilizzato il Pioneer3AT come piattaforma mobile. Il Pioneer3AT è un robot sviluppato per la ricerca in operazioni come la mappatura di un ambiente, la navigazione, la visione, la manipolazione, la cooperazione con

altri robot e altri comportamenti per la robotica autonoma. Questo robot è composto da 1 batteria, 1 bottone di stop per situazioni di emergenza, 4 ruote ciascuna comandata da un motore e 8 sonar frontali di profondità. Il robot può sopportare un peso fino a 12Kg e arrivare ad una velocità massima di 0,8m/s. All'inizio del progetto il robot disponeva di una struttura di metallo a cui erano collegati un sensore laser SICK lms100 e una Kinect. In figura 2.3 è rappresentata l'intera struttura del robot.

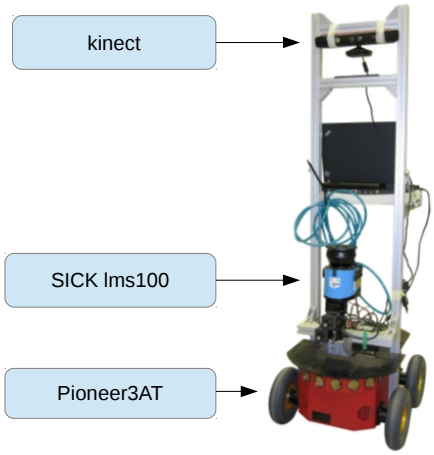


Figura 2.3: La piattaforma mobile usata in questo lavoro dotata di una Kinect e un laser scanner SICK lms100.

Il driver utilizzato per il Pioneer3AT è Rosaria disponibile come pacchetto ROS[1]. Questo driver fornisce un interfaccia per l'utilizzo di robot mobili che implementano la libreria ARIA. Grazie ad essa è possibile leggere informazioni di stato come l'odometria, lo stato della batteria ed inviare comandi per impostare velocità e accelerazione da codice.

2.3.2 Sick laser scanner lms100

Il laser scanner SICK lms100(in figura 2.4 e 2.3) è il dispositivo utilizzato per rilevare l'ambiente circostante. Può rilevare un ostacolo fino a 20 metri, ha una frequenza di aggiornamento di 50hz e un'angolazione massima di 270°. Il driver utilizzato è disponibile nel pacchetto ROS lms1xx[2].

[1]http://wiki.ros.org/ROSARIA
[2]http://wiki.ros.org/LMS1xx

Figura 2.4: Il laser scanner SICK lms100 utilizzato in questo lavoro.

2.3.3 Microsoft Kinect

Kinect è un dispositivo RGB-D sviluppato da Microsoft per la console Xbox 360. Questo sensore è stato pensato per l'interazione tra la console e il giocatore utilizzando esclusivamente i movimenti del corpo umano. Dal punto di vista della ricerca Kinect è un dispositivo a basso costo che ha reso possibile il superamento di problematiche che richiedevano la ricostruzione 3D di una scena in tempo reale.

Come si vede dalla figura 2.5, Kinect è dotato di una telecamera RGB, di un proiettore a raggi infrarossi e di una telecamera sensibile alla stessa banda di frequenze. Le immagini a colori vengono acquisite da una telecamera RGB con risoluzione 640x480 pixel. Le immagini di profondità vengono acquisite utilizzando il proiettore e la telecamera sensibile agli infrarossi. La risoluzione massima per l'immagine di profondità è di 320x240pixel. La profondità massima può arrivare fino a 12 metri. Entrambe le telecamere sono in grado di funzionare a 30 frame al secondo. Il campo visivo del dispositivo è di 57° in orizzontale e di 43° in verticale.

Figura 2.5: Componenti della Kinect

Come descritto in [11] la ricostruzione 3D avviene utilizzando una tecnica chiamata luce strutturata. Questa tecnica consiste nel proiettare un pattern di punti laser sulla scena e rilevarli da una telecamera sensibile alla stessa banda di frequenze. Il pattern risultante viene confrontato con un pattern di riferimento da cui è possibile ricostruire la profondità usando un processo di triangolazione. L'idea alla base del processo consiste nel fatto che un punto proiettato su due oggetti ad una profondità differente viene visualizzato nell'immagine all'infrarosso in una posizione differente. Un'immagine di questo spostamento è visibile in figura 2.6a. Calcolando lo spostamento di ogni punto nell'im-

magine all'infrarosso e utilizzando i parametri intrinseci della Kinect è possibile ricostruire l'immagine di profondità come in figura 2.6b.

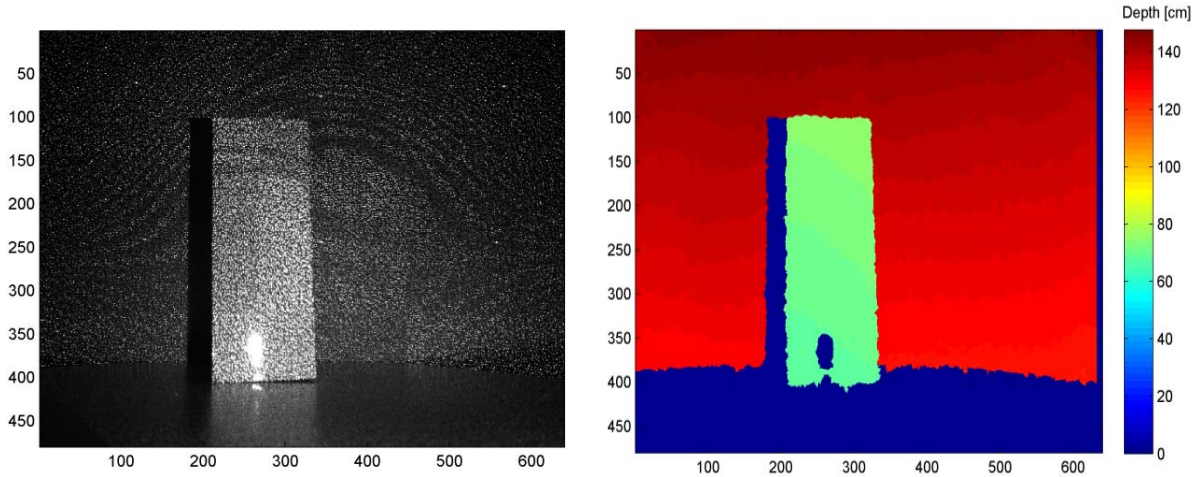


Immagine all'infrarosso

Immagine di profondità

Figura 2.6: Esempio di immagini all'infrarosso e di profondità. Le immagini sono state prese da [11].

Capitolo 3

Localizzazione e identificazione di persone

Lo scopo del sistema spiegato in questo capitolo è quello di esplorare un ambiente sconosciuto al fine di localizzare e re-identificare le persone al suo interno. Questo sistema si colloca principalmente in un contesto di sorveglianza, ma può essere facilmente modificato per cercare delle persone in un ambiente USAR o assistere un anziano.

3.1 Spiegazione del sistema

Le attività principali seguite dal sistema sono le seguenti:

- Rilevazione e filtraggio dell'ambiente: questa attività è necessaria per disporre di dati opportunamente filtrati utilizzati nella costruzione dell'ambiente.
- SLAM(Simultaneus Localization And Mapping): questa attività è necessaria per poter costruire una mappa dell'ambiente e per poter localizzare il robot al suo interno.
- Navigazione: questa attività è necessaria per muovere il robot da un punto iniziale ad un punto finale all'interno della mappa.
- Esplorazione: questa attività è necessaria per muovere il robot nelle zone della mappa non ancora visitate.
- Rilevazione e tracking di persone: questa attività è necessaria a rilevare le persone presenti nel campo visivo della Kinect e ad associare un identificativo univoco per poterle tracciare.
- Identificazione di persone a lungo termine: questa attività è necessaria per poter determinare l'identità delle persone.
- Following: questa attività è necessaria per muovere il robot al fine di seguire una persona.

- Pianificazione ad alto livello: questa attività è necessaria a determinare quali azioni eseguire in base agli eventi che il sistema deve affrontare.

Una rappresentazione dell'intero sistema può essere trovata in figura 3.1. Nel sis-

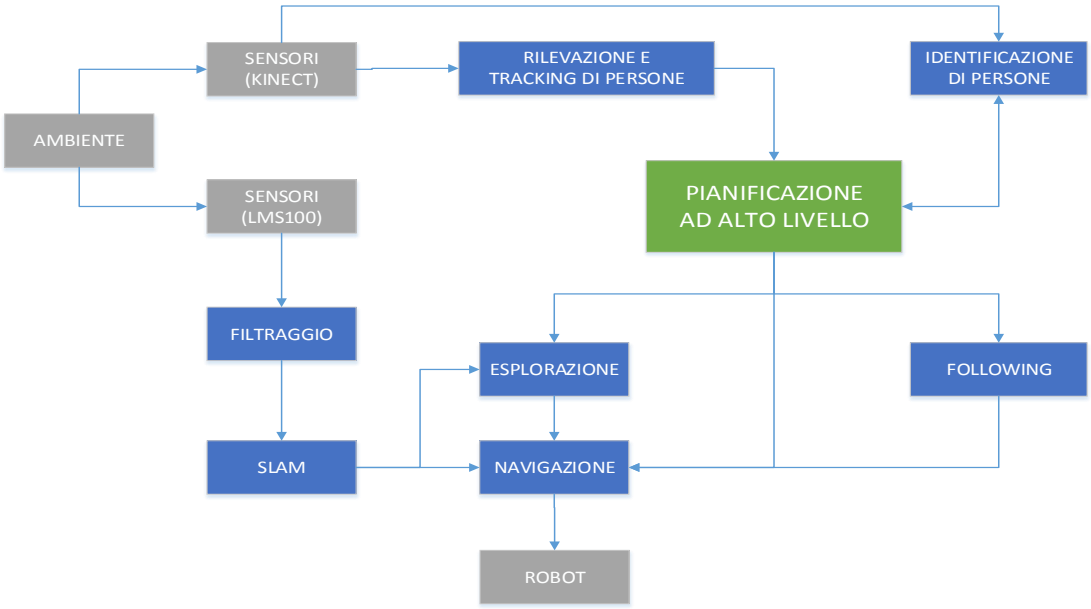


Figura 3.1: Diagramma delle attività eseguite dal sistema

tema sono presenti due flussi di esecuzione paralleli per l'acquisizione dei dati provenienti dall'ambiente. Un flusso di esecuzione riguarda i dati provenienti dal laser scanner che vengono inizialmente filtrati opportunamente per rimuovere delle rilevazioni indesiderate. Successivamente i dati filtrati vengono utilizzati da un algoritmo di SLAM per creare una mappa dell'ambiente e localizzare il robot al suo interno. A partire dalla mappa e dalla localizzazione del robot, il sistema è in grado di esplorare l'ambiente e muovere il robot evitando potenziali ostacoli. L'altro flusso di esecuzione riguarda i dati video provenienti dal sensore Kinect. Questi dati vengono utilizzati sia da un algoritmo di rilevazione e di tracking di persone e sia da un algoritmo di re-identificazione a lungo termine di persone.

Il ruolo principale viene svolto dal pianificatore ad alto livello. Questo algoritmo prende diverse decisioni in base alle situazioni in cui il sistema si trova. Se durante l'esecuzione del sistema non è presente nessuna persona, allora viene utilizzata l'esplorazione per cercare di trovare delle persone presenti nell'ambiente non ancora visitato. Se viene rilevata una persona il sistema esce dall'esplorazione e passa ad una fase di avvicinamento. In questa fase viene utilizzata la navigazione per rendere sicuri i movimenti del robot. Durante questa fase vengono effettuati alcuni controlli per accertarsi dell'effettiva presenza di una persona o della sua raggiungibilità. Al termine di questa fase si passa

all'identificazione della persona. Se la persona viene riconosciuta si passa alla fase di following, altrimenti il robot passa alla prossima operazione.

Per quanto riguarda le attività di SLAM, navigazione ed esplorazione sono stati fatti alcuni confronti con gli algoritmi presenti nello stato dell'arte. In sezione 3.2 saranno spiegate le scelte che hanno portato a selezionare una algoritmo a scapito di un altro. Le attività di rilevazione e identificazione di persone saranno spiegate in maggiore dettaglio in sezione 3.3. Infine in sezione 3.4 sarà spiegato in maggiore dettaglio il funzionamento dell'algoritmo di pianificazione ad alto livello.

3.2 Algoritmi per la navigazione e l'esplorazione

In questa sezione verranno spiegati in maggiore dettaglio le attività di rilevazione e filtraggio dell'ambiente, gli algoritmi di SLAM e gli algoritmi che permettono alla piattaforma mobile di muoversi coerentemente all'interno di un ambiente.

3.2.1 Rilevazione dell'ambiente e relativo filtraggio

Prima di poter muovere il robot all'interno dell'ambiente è necessario poter rilevare la posizione degli ostacoli. A tale scopo è stato usato il sick laser scanner lms100. Le rilevazioni del laser restituite sono costituite da un vettore di distanze. Ogni posizione corrisponde ad un incremento di circa 0.08 radianti rispetto alla rilevazione adiacente. L'incremento procede in senso antiorario. Nelle figure 3.2a e 3.2b sono visualizzati un esempio di rilevazioni laser e l'ambiente associato. Nell'immagine 3.2a si vedono i sistemi

(a) esempio di rilevazioni laser visualizzate in RVIZ (b) ambiente associato

Figura 3.2: Esempio di rilevazioni laser e ambiente associato

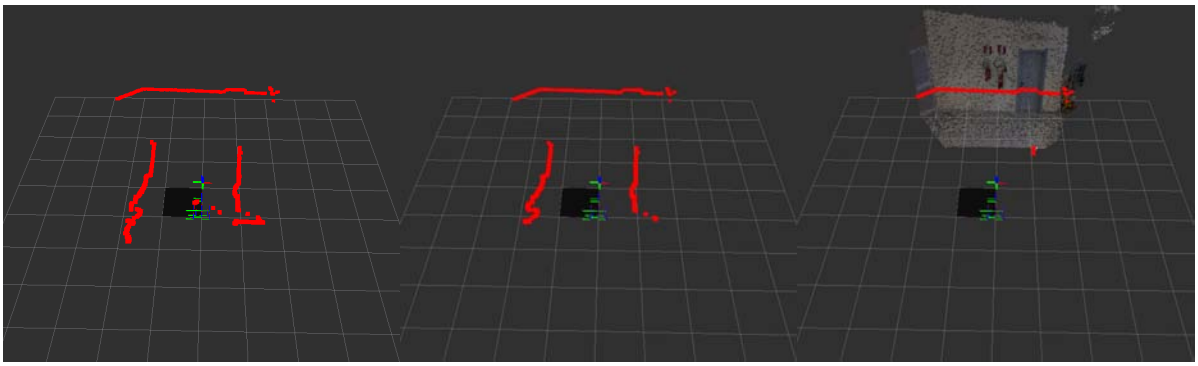
di riferimento della piattaforma mobile e le rilevazioni del laser rappresentati da quadrati.

Per poter utilizzare le rilevazioni del laser è stato necessario operare due filtraggi sull'angolazione massima raggiunta dalle rilevazioni. Il primo filtraggio effettuato deriva

dal fatto che l'ampiezza massima raggiunta dal laser è in grado di rilevare la presenza degli assi del telaio montato a bordo del robot. Questo impedisce ad un algoritmo di navigazione di funzionare correttamente dato che vengono costantemente rilevati ostacoli nelle vicinanze del robot. In figura 3.3a è presente un esempio del problema. Si può notare come nella parte posteriore del robot siano presenti delle rilevazioni. Per risolvere il problema l'angolazione massima è stata ridotta a 240° come si può vedere dalla figura 3.3b.

Il secondo filtraggio deriva dal fatto che l'obiettivo dell'esplorazione è la rilevazione delle persone tramite la Kinect. Questo implica che le rilevazioni del laser che considerano una porzione più ampia della visione della Kinect devono essere eliminate. L'angolazione massima impostata in questo secondo filtraggio è di circa 60° come si può vedere dalla figura 3.3c. In questa figura è stata riportata anche la nuvola di punti ricostruita dalla Kinect. Da notare che il filtraggio per la navigazione potrebbe essere sostituito dal filtraggio per l'esplorazione. Questo però limiterebbe senza motivo la visione del robot durante gli spostamenti e non consentirebbe di evitare gli ostacoli mobili presenti ai lati del robot.

L'algoritmo utilizzato per entrambi i filtraggi è presente nel pacchetto ROS `laser_filters` [1]. Con questo pacchetto è possibile eliminare tutte le rilevazioni inferiori ad una soglia minima e superiori ad una soglia massima.



(a) dati del laser non filtrati (b) filtraggio per la navigazione (c) filtraggio per l'esplorazione

Figura 3.3: Esempio dei filtraggi applicati sulle rilevazioni laser

3.2.2 SLAM

Un altro passaggio necessario per muovere il robot nell'ambiente è la costruzione di una mappa e la localizzazione del robot al suo interno. Questo procedimento viene eseguito dagli algoritmi di SLAM. In generale un algoritmo di SLAM si basa sull'odometria del robot e sulle rilevazioni provenienti dall'ambiente esterno. Le rilevazioni provenienti dall'ambiente sono elaborate ad ogni variazione dell'odometria al fine di estrarre dei punti

[1] http://wiki.ros.org/laser_filters

di riferimento o landmarks. I landmarks vengono confrontati in momenti consecutivi per poter calcolare la posizione del robot durante i suoi spostamenti.

Confronto tra algoritmi di SLAM

In ROS sono presenti numerosi algoritmi di SLAM. Per selezionarne uno è stato preso come riferimento il lavoro sviluppato in [13]. In questo lavoro sono stati messi a confronto 5 algoritmi di SLAM disponibili in ROS: HectorSLAM[14], Gmapping[15], KartoSLAM[2], CoreSLAM[16] e LagoSLAM[17]. I parametri utilizzati per il confronto sono l'errore commesso durante la creazione di una mappa e il carico computazionale richiesto. L'errore commesso è stato calcolato come la somma normalizzata delle distanze in metri tra gli ostacoli nella mappa di riferimento e gli ostacoli della mappa ricostruita. I risultati sono stati riportati nelle tabelle 3.1. In entrambi i casi si vede come HectorSLAM, Gmapping e KartoSLAM sono in grado di ricostruire una mappa con una precisione elevata e con un carico computazionale comparabile. Tra i tre algoritmi KartoSLAM presenta le performance migliori seguito da HectorSLAM e infine da Gmapping. Dato che KartoSLAM non è disponibile per la versione di ROS utilizzata, l'algoritmo selezionato è HectorSLAM.

	HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM
test1	1.1972	2.1716	1.0318	14.75333	3.0264
test2	0.5094	0.6945	0.3742	7.9463	0.8181
test3	1.0656	1.6354	0.9080	7.5824	2.5236
cpu	6.1107%	7.0873%	5.4077%	5.5213%	21.0839%

Tabella 3.1: Errore commesso nella ricostruzione di un ambiente seguendo tre traiettorie differenti e carico computazionale richiesto. I risultati sono stati presi da [13].

Risultati forniti da HectorSLAM

I risultati restituiti da un algoritmo di SLAM sono la mappa dell'ambiente e la localizzazione del robot all'interno della mappa. La localizzazione all'interno della mappa si traduce nei frame `/map`, `/odom` e `/base_link`. Il frame `/map` è il sistema di riferimento fisso nel mondo. Il frame `/odom` è il sistema di riferimento dell'odometria del robot. Se un algoritmo di SLAM non è attivo questo frame prende il posto del frame `/map`. Se invece è presente un algoritmo di SLAM, la sua posizione viene aggiornata nel tempo per correggere gli errori dell'odometria. Infine il frame `/base_link` rappresenta il sistema di riferimento del robot. L'algoritmo di SLAM fornisce la trasformazione tra i tre sistemi di riferimento nel seguente ordine: $map \rightarrow odom \rightarrow base\_link$.

[2]http://wiki.ros.org/slam_karto

La mappa restituita da un algoritmo di SLAM viene rappresentata come una griglia. Ogni cella della griglia può contenere il valore occupato, libero o non conosciuto. Dall'immagine 3.4a è possibile vedere l'immagine della griglia che rappresenta un ambiente. Il colore nero, bianco o grigio indicano se lo spazio è occupato, libero o non conosciuto rispettivamente. Nell'immagine a lato è raffigurata anche la collocazione dei frame `/map`, `/odom` e `/base_link`.

(a) (b)

Figura 3.4: Esempio di mappa restituita da HectorSLAM e dei frame /map, /odom e /base_link

3.2.3 Navigazione

Per poter muovere il robot tra due punti all'interno della mappa è necessario disporre di un algoritmo di navigazione. Per lo sviluppo di questo progetto sono state prese in considerazione due navigazioni: la navigation stack di ROS[19] e la navigation Dodich[20].

Navigation stack di ROS e navigation Dodich

Entrambe le navigazioni si basano su un pianificatore globale e un pianificatore locale. Un pianificatore globale ha lo scopo di calcolare un percorso per collegare un punto iniziale ad un punto finale all'interno della mappa. Un pianificatore locale ha lo scopo di calcolare le velocità da inviare al robot per seguire il percorso calcolato dal pianificatore globale e allo stesso tempo evitare gli ostacoli mobili. Il pianificatore globale implementato in ROS si basa su Dijkstra, mentre il pianificatore globale implementato nella navigazione Dodich si basa su Focused D*[21]. Per quanto riguarda il pianificatore locale in ROS sono disponibili gli algoritmi Trajectory Rollout[23] e il Dinamic Windows Approch(DWA)[22]. Quest'ultimo è il l'algoritmo utilizzato dalla navigazione Dodich. Entrambi questi algoritmi si basano sulla ricerca di una traiettoria da seguire nello spazio delle velocità lineari e angolari.

Effettuando dei test sulle due navigazioni sono state riscontrate alcune differenze più o meno rilevanti. La caratteristica più importante tenuta in considerazione in questo

progetto è la reattività con la quale le due navigazioni sono in grado di ricalcolare un percorso dopo che è stata cambiata la posizione finale da raggiungere. Da questo punto di vista la navigazione Dodich non è utilizzabile dato che ad ogni nuovo obiettivo il robot viene fermato. Al contrario la navigazione di ROS permette di modificare in tempo utile la traiettoria del robot senza la necessità di doverlo fermare. La velocità con la quale i due algoritmi sono in grado di funzionare è un aspetto importante dato che una possibile applicazione di questo lavoro è il following di una persona in movimento.

Altre differenze riguardano la possibile integrazione della navigazione con altre parti del sistema e la qualità dei movimenti effettuati. Per quanto riguarda il primo aspetto, a differenza della navigazione Dodich, la navigazione ROS permette di conoscere lo stato dell'operazione in corso e consente di sapere se una posizione è raggiungibile o meno. La conoscenza di queste due informazioni permette di semplificare l'utilizzo della navigazione durante l'esplorazione dell'ambiente o durante l'avvicinamento ad un utente. Per quanto riguarda il secondo aspetto la navigazione Dodich permette di ottenere un comportamento del robot migliore. Questo riguarda situazioni come posizionarsi vicino ad un ostacolo, calcolare traiettorie sicure e distanti da possibili collisioni o posizionarsi correttamente verso l'orientazione richiesta.

Spazio di configurazione della navigation stack di ROS

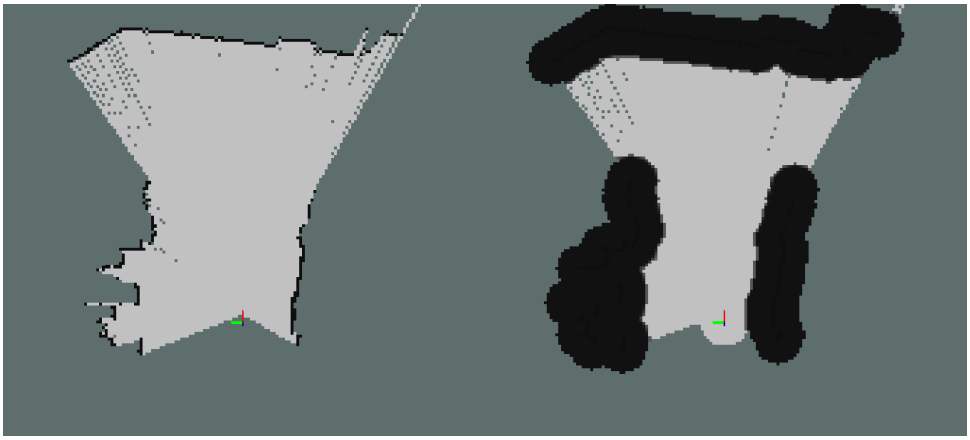
Un algoritmo di navigazione necessita di uno spazio di configurazione per poter funzionare. Lo scopo di questo spazio è ridurre le dimensioni del robot reale ad un punto in modo da rendere più semplice il calcolo di un percorso. Dato che la navigazione di ROS consiste di due pianificatori, sono presenti due spazi di configurazione. Uno riguarda il pianificatore globale e uno il pianificare locale. Entrambi consistono in una mappa di costo in cui gli ostacoli vengono allargati in base alla dimensione del robot. Ogni posizione nella mappa contiene un valore che può indicare se la posizione è libera, vicina ad un ostacolo, occupata o non conosciuta. Dalle figure 3.5a e 3.5b si può vedere la differenza tra la mappa restituita da HectorSLAM e la mappa di costo utilizzata dalla navigazione.

Configurazione della navigation stack di ROS

A differenza della navigazione Dodich, la navigation stack di ROS ha richiesto l'impostazione di alcuni parametri di configurazione per essere utilizzata. Questi parametri permettono di definire il comportamento dei pianificatori e delle mappe di costo associate. I parametri impostati per le mappe di costo riguardano i sistemi di riferimento per la localizzazione provenienti da HectorSLAM, le dimensioni della mappa, la forma del robot e il tipo di filtraggio da cui ottenere le rilevazioni dell'ambiente.

I tipi di parametri delle mappe di costo sono gli stessi. I valori assunti da questi parametri possono coincidere o differire. I parametri coincidenti impostati sono:

- global_frame: impostato a /map. Specifica il nome del sistema di riferimento della mappa.



(a) mappa restituita da HectorSLAM (b) mappa di costo utilizzata dalla navigazione

Figura 3.5: Esempio di mappa restituita da HectorSLAM e di mappa di costo utilizzata dalla navigazione

- robot_base_frame: impostato a /base_link. Specifica il nome del sistema di riferimento del robot.
- static_map: impostato a false. Specifica se la mappa di costo è conosciuta o non conosciuta.
- obstacle_range: impostato a 8.0 metri. Specifica il valore entro il quale considerare i valori del laser scanner.
- raytrace_range: impostato a 8.0 metri. Specifica il valore entro il quale aggiornare lo spazio libero.
- trasform_tolerance: impostato a 0.5 secondi. Specifica la tolleranza in secondi per scartare le trasformazioni tra i sistemi di riferimento della mappa e del robot.
- observation_sources: impostato a laser_scan_sensor. Specifica se acquisire i dati da un sensore laser o da un sensore RGB-D.
- marking: impostato a true. Specifica se inserire gli ostacoli nella mappa di costo.
- clearing: impostato a true. Specifica se aggiornare lo spazio libero.

I parametri che invece differiscono sono:

- width: impostato a 100.0m per la mappa di costo globale e a 8.0m per la mappa di costo locale. Specifica la dimensione della larghezza della mappa di costo.
- height: impostato a 100.0m per la mappa di costo globale e a 8.0m per la mappa di costo locale. Specifica la dimensione dell'altezza della mappa di costo.

- resolution: impostato a 0.05m per la mappa di costo globale e a 0.025m per la mappa di costo locale. Specifica la dimensione del lato delle celle della mappa di costo.

- update_frequency: impostato a 0.5hz per la mappa di costo globale e a 4hz per la mappa di costo locale. Specifica la frequenza di aggiornamento della mappa di costo.

- publish_frequency: impostato 0.5hz per la mappa di costo globale e a 2hz per la mappa di costo locale. Specifica la frequenza di pubblicazione della mappa di costo.

- robot_radius: impostato a 0.40m per la mappa di costo globale e a 0.30m per la mappa di costo locale. Specifica il raggio del robot all'interno della mappa di costo.

- inflation_radius: impostato a 0.45m per la mappa di costo globale e a 0.35m per la mappa di costo locale. Specifica il raggio di inflazione all'interno della mappa di costo.

I parametri impostati per i pianificatori riguardano principalmente il pianificatore locale. Questi parametri permettono di impostare i limiti sulla velocità e sull'accelerazione del robot, la tolleranza rispetto alla posizione finale da raggiungere e alcuni parametri che riguardano la valutazione delle traiettorie simulate. I parametri impostati sono i seguenti:

- max_vel_x: impostato a 0.3m/s. Specifica la velocità lineare massima.

- min_vel_x: impostato a 0.1m/s. Specifica la velocità lineare minima.

- max_rotational_vel: impostato a 0.3rad/s. Specifica la velocità di rotazione massima sul posto.

- min_in_place_vel_theta: impostato a 0.3rad/s. Specifica la velocità di rotazione minima sul posto.

- max_vel_theta: impostato a 0.4rad/s. Specifica la velocità di rotazione massima in movimento.

- min_vel_theta: impostato a -0.4rad/s. Specifica la velocità di rotazione minima in movimento.

- latch_xy_goal_tolerance: impostato a true. Permette di aumentare la tolleranza sulla posizione finale da raggiungere.

- occdist_scale: impostato a 0.20. Permette di rendere meno vantaggiose le traiettorie che passano vicino ad un ostacolo.

- allow_unknown: impostato a false. Questo parametro riguarda il pianificatore globale e permette di non calcolare percorsi che escono dalla mappa conosciuta.

I parametri che rivestono maggiore importanza riguardano il pianificatore locale. In particolare è stato necessario impostare due parametri che hanno permesso di rendere utilizzabile la navigazione: `latch_xy_goal_tolerance` e `occdist_scale`. Il parametro `latch_xy_goal_tolerance` è necessario per evitare che il robot continui a ruotare su se stesso una volta raggiunta la posizione finale. Questo comportamento è dovuto al fatto che il robot non è sempre in grado di localizzarsi in modo corretto. Il parametro `occdist_scale` è uno dei pesi utilizzati per valutare le traiettorie calcolate dal pianificatore locale. In particolare questo parametro permette di influire sul costo di una traiettoria in funzione della sua distanza da un ostacolo. Dato che inizialmente il robot passava troppo vicino ai bordi di un ostacolo, il suo valore è stato aumentato.

3.2.4 Esplorazione

L'esplorazione è il processo attraverso il quale il robot è in grado di costruire la mappa di un ambiente sconosciuto. Per la versione di ROS utilizzata sono disponibili due algoritmi: `frontier_exploration` e `hector_exploration`.

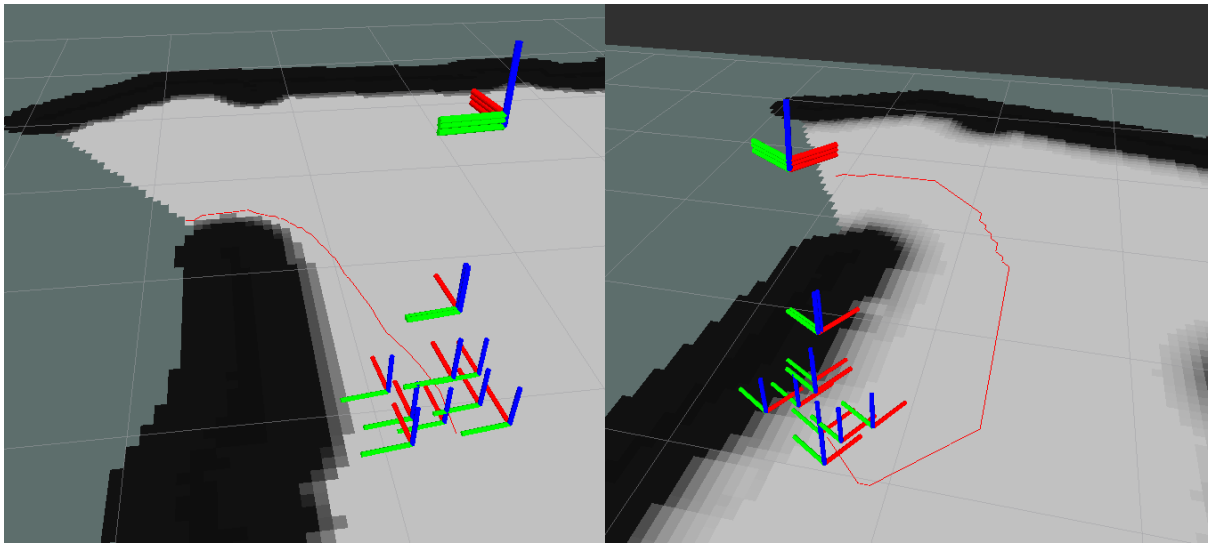
Frontier exploration e Hector exploration

Entrambi gli algoritmi si basano sul lavoro di [24]. L'idea di base prevede di calcolare le posizioni di frontiera tra la parte della mappa conosciuta e non conosciuta.

L'algoritmo `frontier_exploration`[3] fornisce la possibilità di esplorare una mappa indicando il poligono che racchiude l'area da esplorare. L'implementazione si appoggia sulla navigazione ROS per muovere il robot. L'algoritmo `hector_exploration`[25] non è utilizzabile sotto forma di nodo ROS, ma fornisce una classe che espone tutte le funzionalità necessarie ad esplorare una mappa. Oltre ad individuare una frontiera da esplorare, l'algoritmo cerca di valutare i diversi percorsi per raggiungerla. Questa valutazione viene fatta per evitare di seguire dei percorsi che passano troppo vicino ad un ostacolo o che si addentrino in aree prive di punti di riferimento necessari per una corretta localizzazione.

I due algoritmi non sono molto differenti e con le giuste modifiche entrambi permettono di esplorare un ambiente sconosciuto. Per lo sviluppo del progetto è stato preferito `hector_exploration` dato che fornisce la possibilità di calcolare una traiettoria migliore della semplice navigazione. Un esempio delle traiettorie calcolate dai due algoritmi sono presenti nelle figure 3.6a e 3.6b. Nelle immagini si vedono i sistemi di riferimento del robot, la mappa di costo e il percorso da seguire in rosso. A sinistra è stato riportato il percorso calcolato dall'algoritmo `frontier_exploration`. Come si vede dall'immagine il percorso termina a ridosso di un ostacolo portando e delle possibili difficoltà all'algoritmo di navigazione nel pianificare una traiettoria da seguire. A destra è stato riportato il percorso calcolato dal nodo `hector_exploration`. In questo caso il percorso da seguire termina in una zona libera da ostacoli e la traiettoria intermedia tende ad essere maggiormente distanziata dagli ostacoli presenti nell'ambiente.

[3] http://wiki.ros.org/frontier_exploration



(a) percorso calcolato da frontier_exploration (b) percorso calcolato da hector_exploration

Utilizzo di Hector exploration

Il codice originale fornito da `hector_exploration` è composto dalle seguenti componenti:

- La classe `hector_exploration_planner` contenente i metodi necessari all'individuazione delle frontiere all'interno della mappa e di un percorso in grado di raggiungerle.
- La classe `hector_path_follower` contenente i metodi necessari per seguire un percorso prefissato all'interno della mappa.
- Alcuni nodi in grado di esplorare l'ambiente utilizzando le due classi appena citate.

Da un punto di vista ad alto livello, le classi `hector_exploration_planner` e `hector_path_follower` forniscono di fatto un' algoritmo di navigazione. La classe `hector_exploration_planner` implementa un pianificatore globale, mentre la classe `hector_path_follower` implementa un pianificatore locale.

Per riuscire ad utilizzare l'algoritmo di esplorazione fornito dalla classe `hector_exploration_planner` è stato necessario sostituire la classe `hector_path_follower`. Questo pianificatore locale è in grado di seguire correttamente un percorso prefissato, ma non è in grado di evitare un ostacolo mobile o fermare il movimento del robot nel caso in cui la posizione finale da raggiungere sia occupata. Questa limitazione rende il sistema poco sicuro per le persone all'interno dell'ambiente e può causare potenziali collisioni tra gli oggetti dell'ambiente e il robot. Un esempio di queste due situazioni si vede nelle figure 3.6 e 3.7. In figura 3.6 è mostrato il caso in cui il robot calcola un percorso che termina a ridosso di un ostacolo. In figura 3.7 è mostrato il caso in cui la traiettoria del robot viene ostruita da un ostacolo mobile.

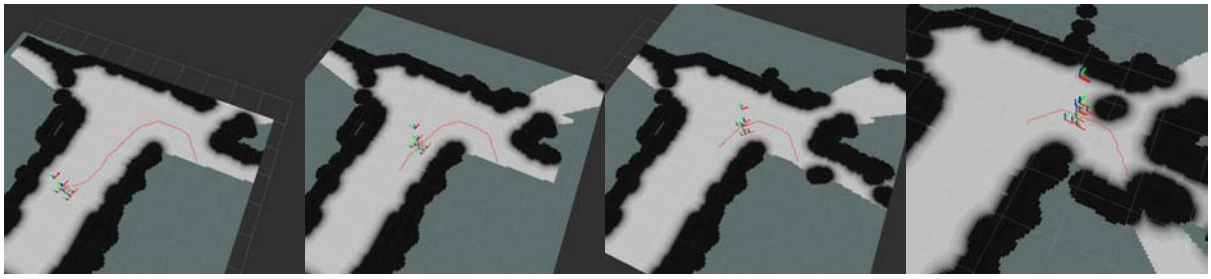


Figura 3.6: Obiettivo finale vicino ad un ostacolo

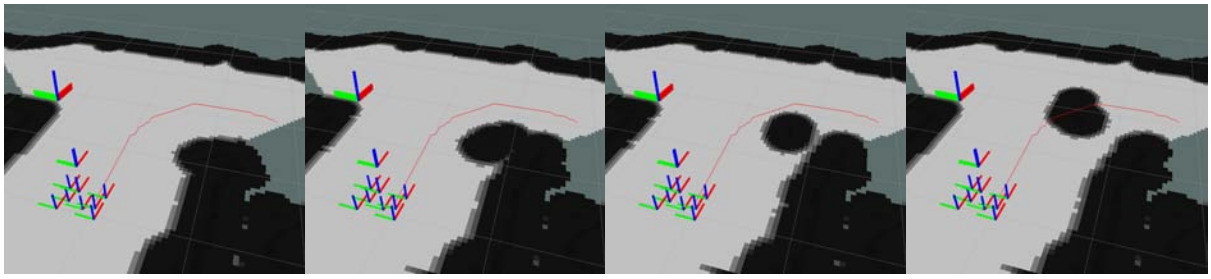


Figura 3.7: Percorso ostruito da un ostacolo mobile

Entrambe le situazioni sono state risolte sostituendo il pianificatore locale `hector_path_follower` con il pianificatore locale `base_local_planner` fornito da ROS e utilizzato normalmente durante la navigazione. Questo pianificatore locale è in grado di modificare la propria traiettoria per evitare un ostacolo mobile e di restituire un risultato per indicare se la posizione finale è raggiungibile oppure no.

I parametri di configurazione usati per il pianificatore locale sono gli stessi usati per la navigazione e spiegati in sezione 3.2.3. Per quanto riguarda i parametri usati per la mappa di costo l'unica differenza riguarda il filtraggio delle rilevazioni laser come spiegato in sezione 3.2.1.

3.3 Algoritmi per la rilevazione e la re-identificazione degli utenti

In questa sezione saranno spiegati in maggiore dettaglio gli algoritmi che permettono di rilevare e re-identificare le persone all'interno dell'ambiente.

3.3.1 Re-identificazione a breve termine

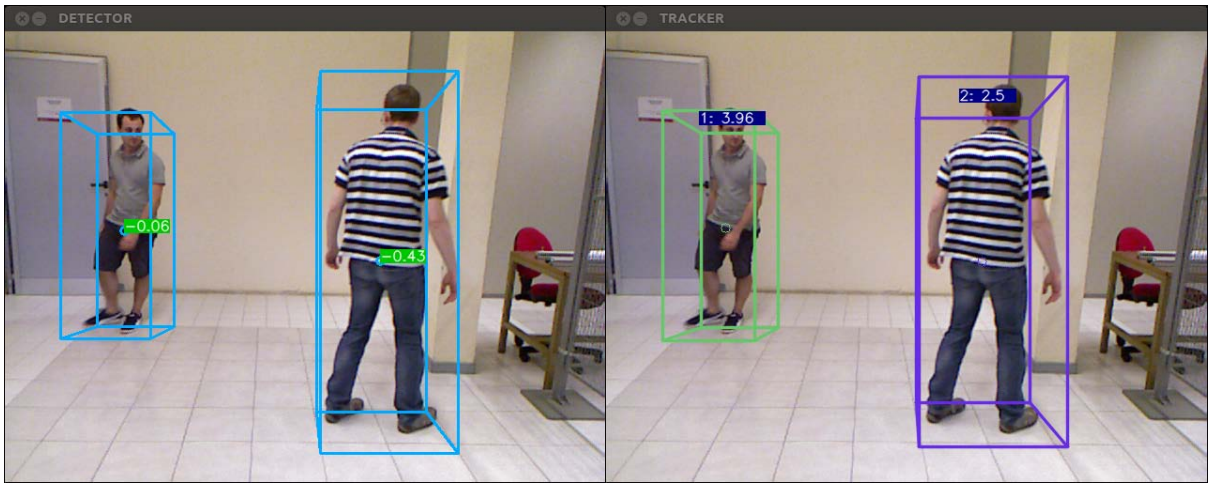
L'algoritmo utilizzato è quello proposto in [18]. L'algoritmo si compone dei seguenti nodi ROS:

- `ground_detector`: è il primo nodo ad essere eseguito e ha la funzione di stimare i coefficienti del pavimento nella scena. Il funzionamento prevede l'input di tre punti da parte dell'utente nell'immagine a colori da cui viene calcolato il piano che approssima il pavimento nella pointcloud associata.

- `people_detector`: questo nodo ha la funzione di rilevare le persone all'interno della scena. Le persone individuate nella scena sono chiamate detection. Il funzionamento prevede la rimozione del pavimento dalla scena, la creazione di cluster dalla pointcloud rimanente e l'applicazione di un HOG people detector all'immagine a colori nelle regioni in cui i cluster sono localizzati.

- `people_tracker`: questo nodo ha la funzione di associare nel tempo le detection provenienti dal nodo `people_detector`. Ogni detection viene associata frame dopo frame a un'istanza, chiamata traccia, rappresentativa di una singola persona. Le caratteristiche utilizzate in fase di associazione sono un filtro di Kalman per stimare la posizione di una persona, un classificatore AdaBoost applicato all'istogramma di colore di una persona e la confidenza dell'HOG people detector come misura per distinguere le persone dai falsi positivi rilevati dal nodo `people_detector`.

Nelle figure 3.8a e 3.8b sono presenti alcune immagini rappresentative dell'algoritmo. A sinistra sono visualizzate le bounding box contenenti le persone rilevate e i rispettivi valori di confidenza, a destra sono stati stampati gli identificativi che rappresentano le tracce associate ad ogni persona insieme alla distanza dalla Kinect.



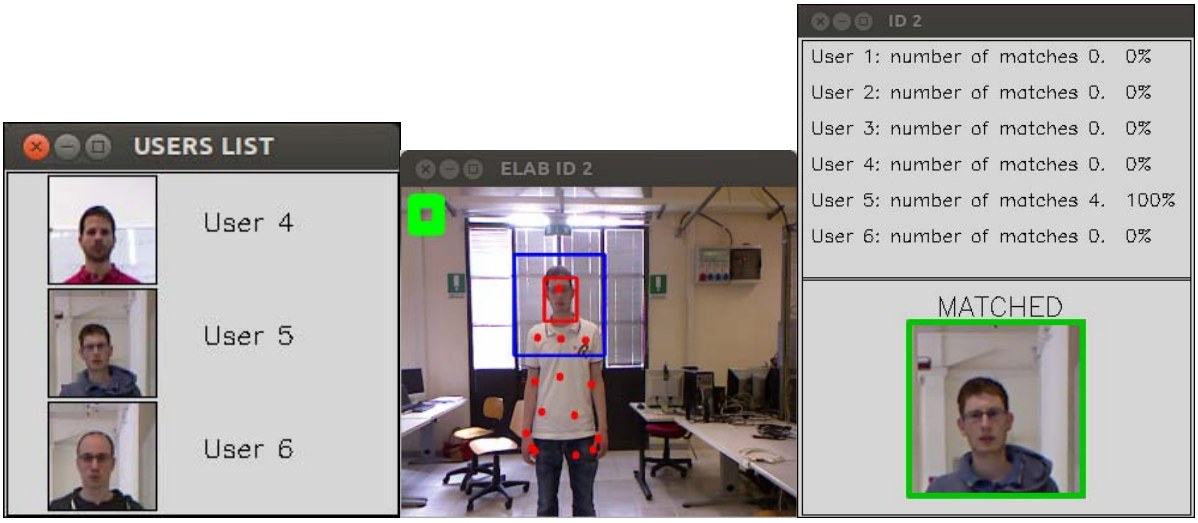
(a) rilevazione di due persone (b) tracce associate alle due persone rilevate

Figura 3.8: Immagini rappresentative dell'algoritmo di re-identificazione a breve termine.

3.3.2 Re-identificazione a lungo termine

L'algoritmo per la re-identificazione di persone a lungo termine era già disponibile a inizio progetto. I descrittori su cui si basa sono le distanze tra i giunti dello scheletro fornito da OpenNI e le distanze tra i punti del viso forniti dall'algoritmo di face recognition presente nella libreria OpenCV. L'algoritmo consiste in una fase di training e una fase di testing. Nella fase di training vengono acquisiti 30 descrittori per lo scheletro e 30 descrittori per il viso. Questa procedura viene ripetuta per ogni soggetto che si vuole poter re-identificare nella fase di testing. Nella fase di testing si cerca di confrontare i descrittori acquisiti da un soggetto con i descrittori trovati nella fase di training. Il confronto può utilizzare singolarmente o congiuntamente i descrittori del viso e dello scheletro.

L'interfaccia grafica dell'algoritmo è costituita da un totale di tre finestre(figura 3.9). In una finestra sono mostrate le immagini degli utenti conosciuti dal sistema. In una seconda finestra è visualizzata l'immagine della persona che si sta re-identificando. Nell'immagine sono visualizzati anche i giunti dello scheletro associato e un rettangolo che ne inquadra il volto. Al termine della fase di re-identificazione viene mostrato l'esito della re-identificazione specificando se è stato trovata una corrispondenza con gli utenti presenti nel sistema.



(a) utenti conosciuti dal sistema (b) utente sottoposto a re-identificazione (c) risultato della re-identificazione

Figura 3.9: Finestre visualizzate dall'algoritmo di re-identificazione a lungo termine

3.3.3 Modifiche apportate

Per quanto riguarda l'algoritmo di re-identificazione a breve termine, è stato modificato il messaggio in uscita dal nodo `people_tracker`. La modifica ha permesso di ag-

giungere la confidenza restituita dal nodo `people_detector` per poter avere un controllo sulla qualità delle rilevazioni delle persone.

Per quanto riguarda l'algoritmo di re-identificazione a lungo termine, è stato modificato il processo di re-identificazione per poter considerare solo una persona alla volta. Per poter indicare la persona da re-identificare, viene indicata la posizione del suo centroide. In questo modo è possibile effettuare la re-identificazione su un solo soggetto selezionando lo scheletro il cui centroide si trova a una distanza minore di 30cm. Se non è disponibile uno scheletro nelle vicinanze, allora la persona che si sta cercando di re-identificare viene considerata non riconosciuta.

3.4 Algoritmo di pianificazione ad alto livello

In questa sezione verrà spiegato il funzionamento dell'algoritmo di pianificazione ad alto livello. In particolare le attività svolte prevedono l'esplorazione dell'ambiente, la rilevazione delle persone, l'avvicinamento alle persone rilevate, la loro identificazione e infine il following delle persone riconosciute.

3.4.1 Funzionamento

Il funzionamento dell'algoritmo di pianificazione ad alto livello attraversa diversi stati. Ogni stato permette di gestire una situazione differente in cui il robot si può trovare durante l'esecuzione. Il funzionamento è illustrato in figura 3.10.

Lo stato iniziale è lo stato **rotation** che prevede l'esecuzione di una rotazione del robot su se stesso. Questo stato è necessario per mappare l'area attorno al robot e per individuare potenziali utenti nelle vicinanze. Il funzionamento prevede l'invio di un segnale al nodo `exploration` che si occuperà dell'effettiva esecuzione della rotazione. Al termine della rotazione l'algoritmo di esplorazione segnala il completamento dell'operazione.

Lo stato successivo è lo stato **waiting**. Questo stato è necessario a controllare la presenza di utenti da re-identificare. Se non è presente nessun utente viene inviato un messaggio al nodo `exploration` per iniziare o continuare l'esecuzione dell'esplorazione. Se è presente un utente da re-identificare, viene inviato un messaggio al nodo `exploration` per sospendere l'esplorazione e si passa allo stato **accost**. Il criterio per la selezione di un utente si basa su alcune variabili di stato associate ad ogni utente e sulla confidenza massima fino a quel momento raggiunta. Una spiegazione più precisa verrà data nella prossima sezione.

Lo stato **accost** è necessario per controllare l'esistenza di una posizione raggiungibile a distanza di due metri attorno ad un utente rilevato. Questa distanza permette di operare in sicurezza e di avere una visuale sufficiente della persona per la fase di identificazione. Se non esiste una posizione da raggiungere, l'utente viene temporaneamente escluso e si torna allo stato **waiting**. Se esiste una posizione da raggiungere viene utilizzata la navigazione di ROS per raggiungere l'utente. Se l'utente viene raggiunto con successo si passa allo stato **pre_identification**. Se l'utente non viene raggiunto significa che non esiste un percorso per raggiungere l'utente. In questo caso l'utente viene escluso

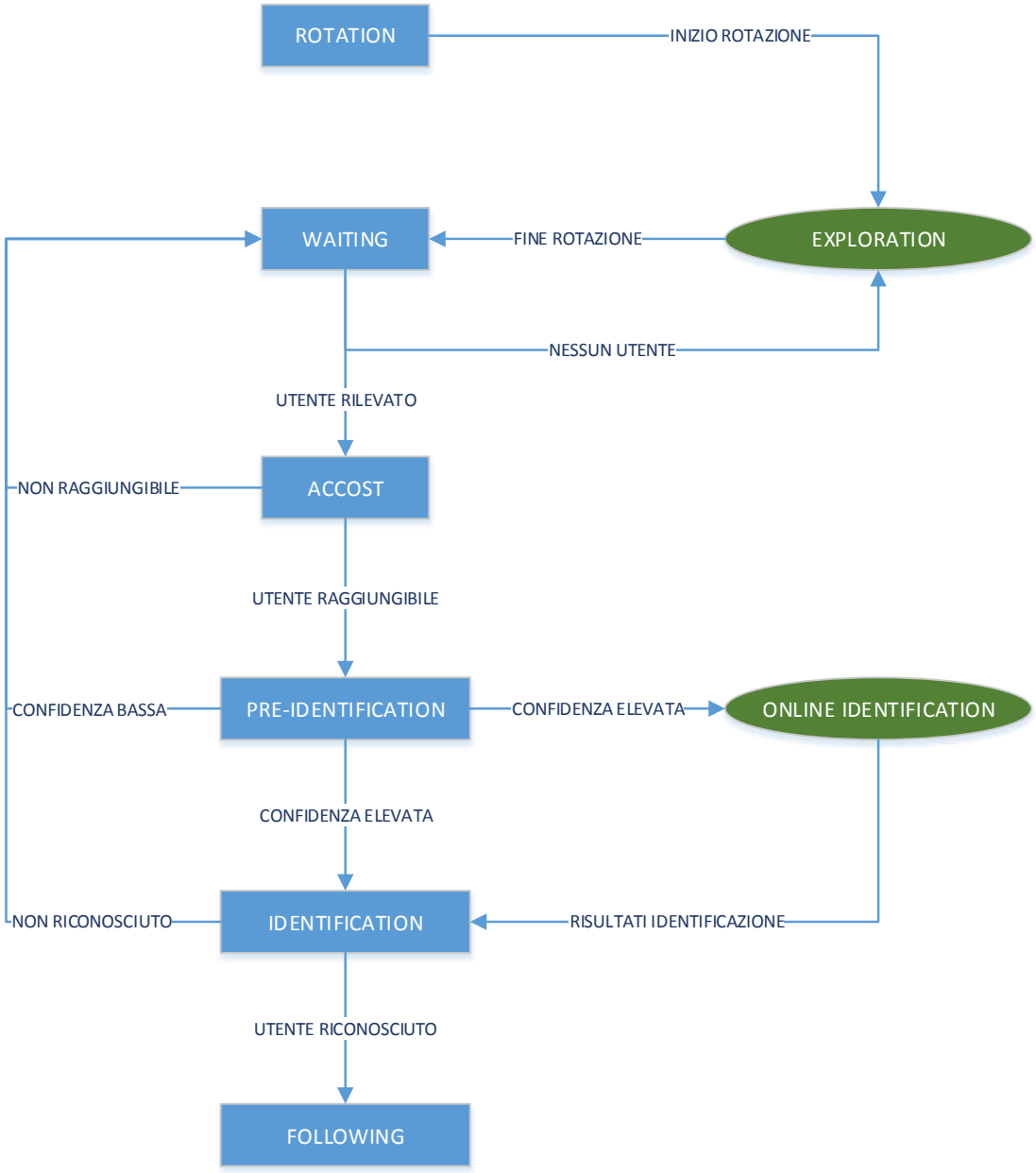


Figura 3.10: Diagramma degli stati seguiti dall'algoritmo di pianificazione ad alto livello

definitivamente per evitare che il robot cerchi di avvicinarsi più volte ad un utente non raggiungibile.

Lo stato **pre_identification** ha due funzioni: ruotare il robot verso l'utente ed effettuare un secondo controllo sulla confidenza massima raggiunta. La rotazione del robot verso l'utente è necessaria a causa del fatto che la navigazione non è sempre in grado di posizionare il robot orientato verso la persona. Il secondo controllo sulla confidenza è necessario per garantire con maggiore sicurezza che la rilevazione dell'algoritmo di people detection sia effettivamente una persona. In questo secondo controllo è stata usata una confidenza più elevata rispetto al primo controllo dato che in questa fase il robot è orientato verso la persona ad una distanza di soli due metri. Se anche questo secondo controllo viene superato, si passa allo stato **identification** e viene inviata la posizione dell'utente da re-identificare al nodo `onlineIdentification`. Se il controllo non viene superato l'utente viene escluso definitivamente e si passa allo stato **waiting**.

Quando viene attivato lo stato **identification** il nodo `coordinator` si mette in attesa del risultato dell'identificazione. Se il risultato è positivo significa che il nodo `onlineIdentification` è riuscito a trovare una corrispondenza tra gli utenti conosciuti e si passa allo stato **following**. Se il risultato è negativo significa che l'algoritmo di re-identificazione non è stato in grado di trovare una corrispondenza corretta tra gli utenti conosciuti e si torna allo stato **waiting**.

Lo stato **following** ha il compito di muovere il robot per seguire la persona che è stata identificata con successo. Per seguire la persona corretta viene utilizzato l'id restituito dall'algoritmo di people tracking. Se la persona viene persa il robot viene fermato e rimane in attesa per un certo intervallo di tempo. Se entro questo intervallo la persona non riappare allora si torna allo stato **waiting**.

Durante l'esecuzione dei vari stati il nodo `exploration` può segnalare il completamento del processo di esplorazione. In questo caso, se non ci sono più utenti da re-identificare, il procedimento riparte dallo stato **rotation**. Gli utenti localizzati fino a quel momento e la mappa esplorata vengono cancellati.

3.4.2 Stato degli utenti

La rappresentazione degli utenti all'interno del sistema prevede l'utilizzo delle informazioni provenienti dall'algoritmo di people tracking e ad alcune informazioni di stato:

- `id`: questo campo contiene l'identificativo associato ad un utente risultante dalla fase di tracking.

- `centroid`: questo campo contiene la posizione dell'utente rispetto al frame /odom. Il frame /odom è un sistema di riferimento restituito dagli algoritmi di SLAM.

- `maxConfidence`: questo campo contiene la confidenza massima con cui è stato visto un utente.

- `IsOopComputable`: questo campo indica se è disponibile una posizione di osservazione per raggiungere l'utente.
- `IsUnreachable`: questo campo indica se l'utente non può essere raggiunto.
- `IsAlreadyIdentified`: questo campo indica se l'utente è già stato identificato.

I campi `id` e `centroid` sono necessari all'aggiornamento dei dati di un utente. L'aggiornamento viene effettuato basandosi sull'assunzione che ogni utente deve rimanere fermo nell'ambiente. In questo modo è possibile utilizzare il campo `centroid` per poter capire se un utente è già stato visto. Per rendere il vincolo meno stringente viene utilizzato anche il campo `id` per permettere piccoli spostamenti dell'utente quando questo non viene visto per brevi periodi di tempo.

Il campo `maxConfidence` è stato utilizzato a causa della tendenza del modulo di people detection a confondere gli oggetti dell'ambiente con delle persone. Il problema è stato risolto considerando il valore della confidenza in due momenti distinti. Nella stato **waiting** e nello stato **pre_identification**. Nel primo stato vengono considerati gli utenti che non hanno una confidenza molto elevata. Questo è necessario a causa del fatto che una persona effettiva può non avere una buona confidenza se parzialmente occluso o visto di sfuggita. Nello stato **pre_identification** vengono considerati validi solo gli utenti con una confidenza sufficientemente elevata. Questo perché lo stato **pre_identification** viene invocata solo dopo che la posizione di osservazione è stata raggiunta ed è quindi disponibile una buona visuale della persona.

Gli ultimi tre campi sono necessari per evitare di considerare quegli utenti che sono già stati sottoposti al processo di identificazione senza successo. Il campo `IsOopComputable` serve a scartare gli utenti da cui non è stato possibile trovare una posizione di osservazione valida. Questa posizione viene calcolata su una circonferenza di raggio due metri in un punto libero da ostacoli. L'esecuzione effettiva viene demandata al nodo `exploration` dato che dispone della mappa da cui vedere l'effettiva presenza di potenziali ostacoli. Dato che la posizione di osservazione dipende dalla conoscenza dell'ambiente circostante, ogni qual volta viene attivata l'esplorazione il campo `IsOopComputable` viene resettato. Il campo `IsReachable` serve a scartare gli utenti che non sono stati raggiunti a causa di problemi legati alla navigazione. Il campo `IsAlreadyIdentified` serve a scartare gli utenti che sono già stati identificati ma senza successo.

3.5 Applicazione dell'algoritmo di navigazione per il following di persone

L'algoritmo di following era già presente ad inizio progetto. Questo algoritmo ha lo scopo di muovere il robot per seguire un utente. Il suo funzionamento consiste nel calcolare la distanza dall'utente che si vuole seguire e posizionare il robot a circa due metri con una orientazione frontale. Questo algoritmo è molto semplice ed efficace, ma

può funzionare correttamente solo in un ambiente privo di ostacoli. Questo è dovuto al fatto che non viene tenuto conto l'ambiente circostante.

Per risolvere il problema è stata aggiunta la possibilità di inviare la posizione da raggiungere alla navigazione di ROS. La posizione viene calcolata come prima in una circonferenza di raggio due metri. Dato che nella fase di following l'utente può muoversi, la posizione in cui inviare la navigazione di ROS deve essere aggiornata continuamente. Per abbassare il carico computazionale l'aggiornamento viene fatto solo se l'utente si è spostato per più di 10cm dalla posizione precedente. Inoltre, dato che la navigazione non può muovere il robot in retromarcia, quando l'utente da seguire si trova a una distanza minore di due metri il robot viene fermato. Da questa posizione il robot può seguire l'utente solo orientandosi verso di esso.

3.6 Test effettuati

Sono state fatte diverse prove per testare l'intero sistema. Queste prove possono essere riassunte in tre tipi di test. Ogni test ha previsto l'esecuzione di una parte dell'intero sistema per poter comprendere più facilmente le problematiche relative ad ogni singola componente. I test fatti inizialmente hanno previsto l'uso della sola esplorazione. Questi test sono stati fatti per capire se il robot è in grado di esplorare un ambiente senza l'intervento di un operatore. I test fatti successivamente hanno previsto l'aggiunta delle rilevazioni delle persone. Questi test sono stati fatti per capire se il robot è in grado di avvicinarsi e orientarsi correttamente verso un utente, scartare gli utenti già re-identificati e procedere con l'esplorazione nel caso non siano presenti persone da re-identificare. Infine è stato testato l'intero sistema con l'intento di capire se il robot è in grado di riconoscere una persona e infine di seguirla.

Nelle sezioni successiva saranno dati degli esempi di esecuzione per ogni test. In sezione 3.6.4 saranno discusse le difficoltà incontrate. L'hardware utilizzato consiste di un processore i7-4500U, di 8GB di ram Ddr3 e di una scheda video Nvidia GeForce GT 740M.

3.6.1 Test sull'esplorazione

Come già descritto in sezione 3.4.1, inizialmente il robot compie una rotazione su se stesso. Dalle immagini riportate in figura 3.11 si vede come l'area attorno al robot sia stata progressivamente esplorata. Dato che non possono essere stati rilevati utenti(l'algoritmo di rilevazione di persone non era attivato), il prossimo passo riguarda il calcolo di una posizione di frontiera da raggiungere. Questa posizione viene calcolata da `hector_exploration` insieme al percorso da seguire per raggiungerla(figura 3.12).

Ogni qual volta il robot raggiunge la fine del percorso, viene calcolato un nuovo percorso da seguire fino all'esplorazione completa dell'intero ambiente. Nelle immagini di figura 3.13 sono illustrati alcuni percorsi calcolati ogni volta che il robot raggiunge la posizione finale calcolata al passo precedente.

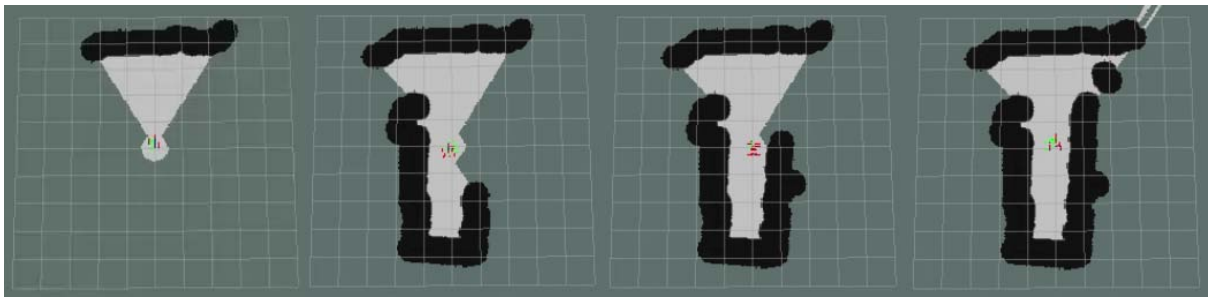


Figura 3.11: Esplorazione dell'ambiente durante la rotazione iniziale

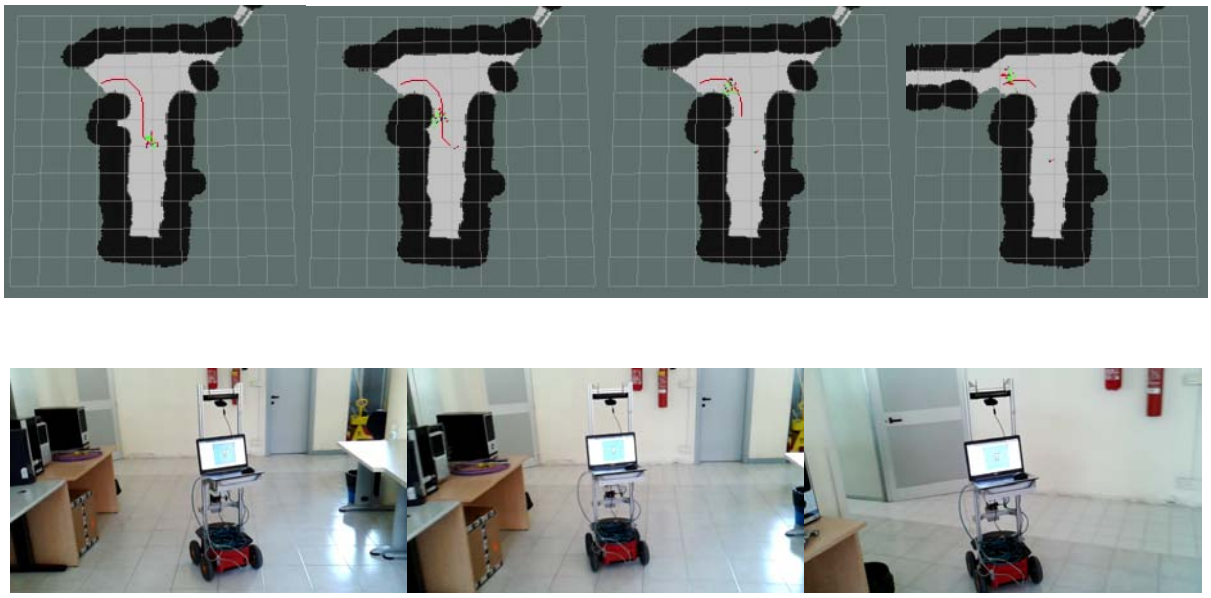


Figura 3.12: Percorso seguito dal robot per raggiungere i punti di frontiera

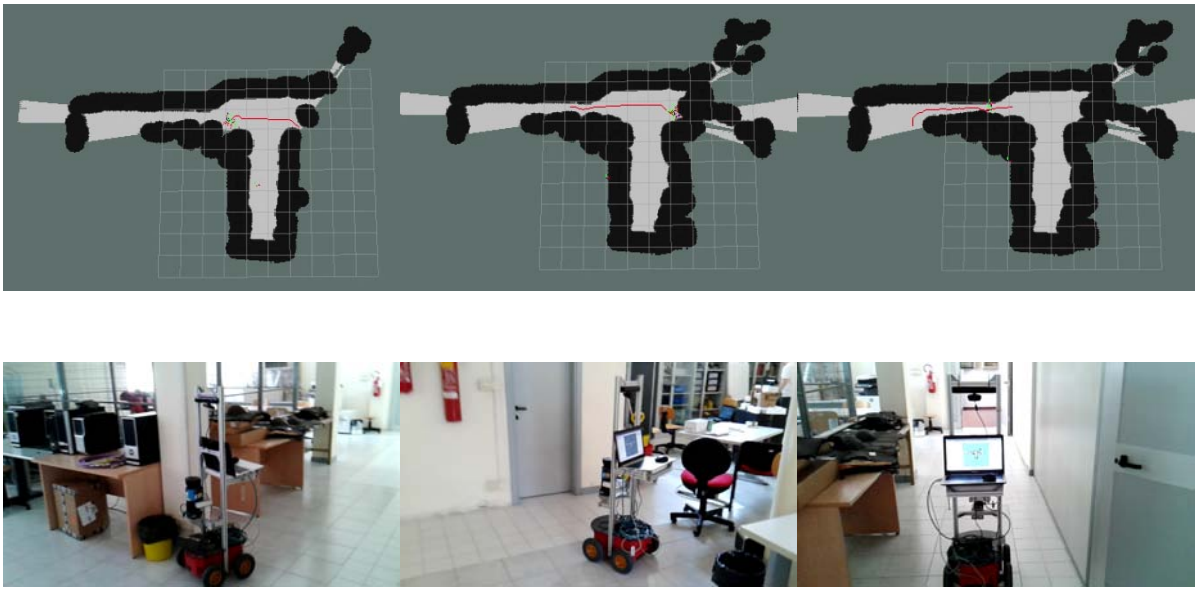
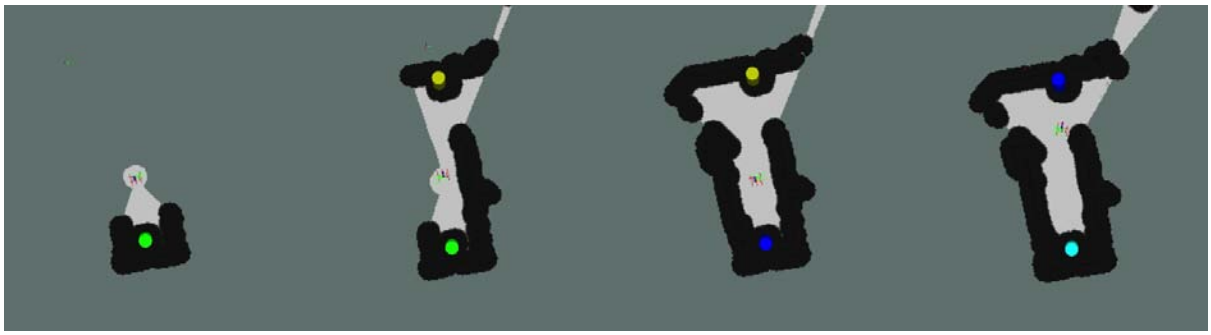


Figura 3.13: Nuovi percorsi calcolati al raggiungimento di una frontiera

Una volte terminata l'esplorazione la mappa viene azzerata e il processo ricomincia dalla rotazione iniziale.

3.6.2 Test sulla rilevazione degli utenti

In questo test sono state posizionate delle persone nell'ambiente. Due persone erano già visibili durante la rotazione iniziale del robot. In figura 3.14a e 3.14b sono riportati dei cilindri per indicare la loro posizione. Il colore verde indica un buon livello di confidenza, il colore giallo indica un livello basso. Dato che sono presenti due persone da rilevare il sistema seleziona la persona più vicina per la fase di avvicinamento. Nell'immagine 3.14c la persona scelta è stata indicata in blu. Una volta raggiunta una posizione di osservazione il robot si orienta verso la persona e passa alla fase di re-identificazione a lungo termine. Questa fase viene solo simulata dato che in questo test non si vuole testare il riconoscimento delle persone. Al termine della simulazione il sistema considera l'utente come già re-identificato e procede a re-identificare il successivo. Come prima il robot si avvicina alla persona selezionata e procede a simulare il suo riconoscimento(3.14d). Gli utenti che sono stati sottoposti alla fase di re-identificazione sono segnalati in azzurro. Se non sono più presenti utenti da re-identificare il sistema passa alla fase di esplorazione dell'ambiente come visto nel primo test. Se durante l'esplorazione viene rilevata una persona raggiungibile il sistema ripete gli stessi passi appena visti. Se la persona non è raggiungibile l'esplorazione continua la sua esecuzione.



(a) (b) (c) (d)

(e) (f)

Figura 3.14: Rilevazione di due persone durante la rotazione iniziale e conseguente avvicinamento

3.6.3 Test sul riconoscimento degli utenti

Nel test spiegato è stata utilizzata una sola persona per semplicità. Come spiegato nei test precedenti il robot esegue una rotazione iniziale e si avvicina agli utenti rilevati per la fase di re-identificazione a lungo termine. Durante questa fase vengono confrontate le caratteristiche dello scheletro e del viso della persona di fronte alla Kinect con le caratteristiche delle persone acquisite durante la fase di training. Nel test eseguito sono stati utilizzati solo i descrittori del viso dato che permettono di ottenere risultati migliori. Una volta che una persona viene riconosciuta il robot procede a seguirla(immagini in figura 3.15).

Figura 3.15: Fase di inseguimento di una persona

3.6.4 Limitazioni del sistema

Durante i test della sola esplorazione non sono mai avvenute collisioni con ostacoli rilevabili dal laser scanner. Al contrario nel caso di sedie, mobili o fili elettrici il robot ha richiesto più volte l'intervento di un operatore.

Nel caso dei test per la rilevazione delle persone sono stati riscontrati principalmente due problemi. Il primo problema riguarda la possibile non rilevazione di un utente da parte del modulo di people detection. Questo problema deriva dal fatto che le persone non vengono sempre rilevate con una buona confidenza. Il secondo problema riguarda la visuale limitata della Kinect. Questo comporta l'impossibilità di rilevare le persone posizionate subito dopo una svolta per due motivi. Durante l'aggiramento di un ostacolo la Kinect non ha una visione laterale sufficientemente ampia e una volta inquadrata la persona la sua immagine sarà troppo grande per essere contenuta nell'intero campo visivo. Solitamente se il robot si trova a una distanza inferiore a 1.50m, allora la persona non verrà rilevata.

Nel caso dei test per il riconoscimento delle persone sono stati riscontrati due problemi. Il primo problema riguarda la scarsa capacità dell'algoritmo di re-identificazione di riconoscere effettivamente le persone. Nel test riportato l'utente era stato riconosciuto dopo quattro tentativi. Il secondo problema non riguarda un cattivo funzionamento dell'algoritmo, ma una disponibilità insufficiente di risorse per funzionare. Infatti, durante l'esecuzione dell'intero sistema, vengono eseguite in parallelo le attività di SLAM, navigazione, esplorazione, rilevazione delle persone e skeletal tracking. La conseguenza di

questo sovraccarico comporta difficoltà soprattutto nella localizzazione e nei movimenti del robot durante l'esplorazione.

Capitolo 4

Utilizzo delle informazioni dello scheletro per il people tracking

In questa parte della tesi verrà proposto l'uso di una tecnica di skeletal recognition per il miglioramento dell'algoritmo di people tracking utilizzato nella prima parte durante la fase di following. Questa parte è stata sviluppata per via del fatto che il tracking di persone non avviene sempre correttamente. Come sarà spiegato, se una persona viene occlusa da altre persone o se esce e rientra nella visuale della Kinect, è possibile che il suo identificativo associato dall'algoritmo di tracking possa erroneamente cambiare. Lo scopo di questa parte consiste nel diminuire il numero di cambi id associati ad ogni persona. Di conseguenza, durante la fase di following di una persona, il robot sarà in grado di seguire in modo più robusto una persona.

4.1 Spiegazione del sistema

In questa sezione verranno presentati i componenti utilizzati per lo sviluppo del sistema.

4.1.1 Wrapper per lo skeletal tracker NST

I dati relativi ad ogni scheletro vengono pubblicati utilizzando il pacchetto `nite2tf`. Questo pacchetto pubblica le informazioni relative ad ogni scheletro attraverso due topic. Il primo è il topic `/skeletons` da cui è possibile risalire alle confidenze dei singoli giunti che indicano l'accuratezza della loro stima. Il secondo è il topic `/tf` da cui è possibile risalire alla posizione e alla rotazione dei giunti rispetto ad un punto di riferimento.

4.1.2 Algoritmo di people tracking

Questo algoritmo è già stato spiegato in sezione 3.3. Qui è stata riportata una spiegazione più breve.

- nodo **ground_detector**: è il primo nodo ad essere eseguito e ha la funzione di stimare i coefficienti del pavimento nella scena. Il nodo riceve le immagini a colori e le pointcloud da `OpenNI`.

- nodo **people_detector**: questo nodo ha la funzione di rilevare le persone all'interno della scena. Le persone individuate nella scena sono chiamate detection. Il nodo riceve le immagini a colori e le pointcloud da `OpenNI`.

- nodo **remapping**: questo nodo è stato aggiunto in questo lavoro e ha la funzione di calcolare dei valori, chiamati descrittori, a partire dalla posizione dei giunti di ogni scheletro. Al termine dell'operazione questi valori vengono associati alle detection provenienti dal nodo `people_detector`.

- nodo **tracker**: questo nodo ha la funzione di associare nel tempo le detection provenienti dal nodo `people_detector`.

I nodi `ground_detector`, `people_detector` e `tracker` erano già presenti a inizio progetto e una loro spiegazione più approfondita può essere trovata in [18].

Nella figura 4.1 è riportata un'immagine rappresentativa dei componenti appena esposti.

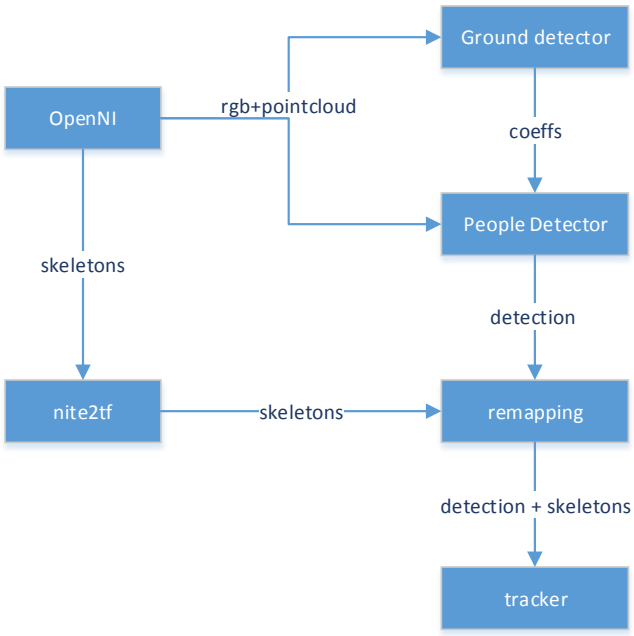


Figura 4.1: Immagine del sistema per il tracking di persone

4.2 Realizzazione del nodo remapping

In questo sezione verrà spiegato in maggior dettaglio il funzionamento nel nodo `remapping`.

4.2.1 Calcolo dei descrittori dello scheletro

I descrittori dello scheletro sono un insieme di valori calcolati in corrispondenza dei giunti riproiettati nell'immagine RGB o direttamente nella pointcloud. In figura 4.2 è stata riportata l'immagine di una persona e i relativi giunti dello scheletro. In corrispondenza di questi giunti vengono estratti dei valori con un algoritmo di feature description.

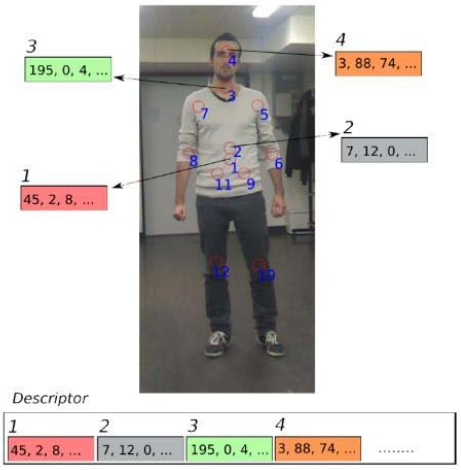


Figura 4.2: Esempio di giunti proiettati su una persona e la relativa estrazione dei descrittori

Questi valori sono stati calcolati prendendo spunto dal lavoro svolto in [27]. In questo lavoro sono stati provati diversi algoritmi di feature description 2D(SIFT[29], SURF[30], etc...) e 3D(PFH[31], FPFH[32], etc...). Tutti i descrittori testati sono presenti nelle librerie OpenCV[1] e PCL[2]. Dai risultati riportati in figura 4.3 si può vedere come le feature 3D siano meno discriminative di quelle 2D a causa di un'immagine di profondità troppo rumorosa e che tra le feature 2D, SIFT restituisce i risultati migliori.

Per poter calcolare effettivamente un descrittore SIFT nei punti riproiettati nell'immagine è stato necessario tenere in considerazione anche la scala e la rotazione. La scala indica il diametro del punto considerato e viene calcolata con la seguente formula:

$$r = f \times \frac{R}{z} \tag{4.1}$$

[1] www.opencv.org
[2] www.pointclouds.org

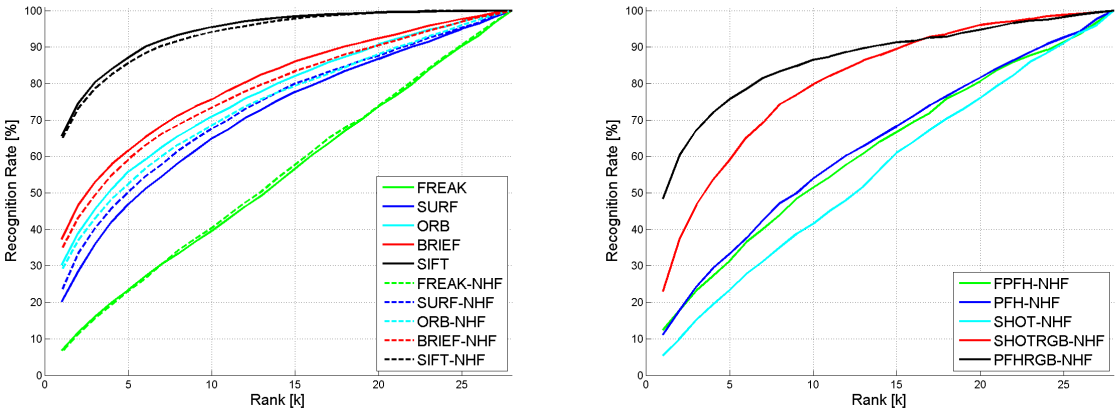


Figura 4.3: Recognition rate degli algoritmi di feature description provati in [27].

dove r è il raggio espresso in pixel, z è la profondità del punto considerato, f è la lunghezza focale del sensore e R il raggio calcolate in coordinate del mondo. Per quanto riguarda la rotazione, questo parametro è stato posto a zero dato che le persone in posizione verticale solitamente non oscillano attorno al proprio asse sagittale.

4.2.2 Associazione tra detection e descrittori dello scheletro

L'associazione tra detection e descrittori degli scheletri è stata effettuata utilizzando la distanza euclidea tra il centroide di una detection e il centroide di uno scheletro. Per avere una rappresentazione grafica del procedimento sono state riportate due immagini in figura 4.4. A sinistra è possibile vedere il risultato dell'algoritmo di people detection. A destra è possibile vedere il risultato dell'algoritmo di skeletal tracking insieme a dei puntini verdi che indicano i centroidi delle detection. Come si può intuire, in base alla distanza tra i centroidi degli scheletri e i centroidi delle detection è possibile effettuare l'associazione corretta. La distanza massima consentita per l'associazione tra uno scheletro e una detection è stata impostata a 30cm. Questa soglia serve ad evitare associazioni errate nel caso in cui sia assente la detection corretta.

Anche se inizialmente si potrebbe pensare che sia sufficiente associare una detection allo scheletro a distanza minore, questo sistema non può dare la garanzia di una corretta associazione tra scheletri e detection. Infatti, a causa di una stima non sempre precisa della posizione del centroide di una detection, potrebbe accadere che lo scheletro di una persona sia erroneamente associato con la detection di un'altra persona. Questo può succedere principalmente quando due persone sono molto vicine e il calcolo della distanza tra i centroidi favorisce un'associazione sbagliata.

In figura 4.5 si vede un esempio del problema. La situazione illustra due scheletri nelle vicinanze di una detection. La detection riguarda la persona centrale più vicina alla Kinect. Stampando le distanze tra il centroide di questa detection e il centroide degli

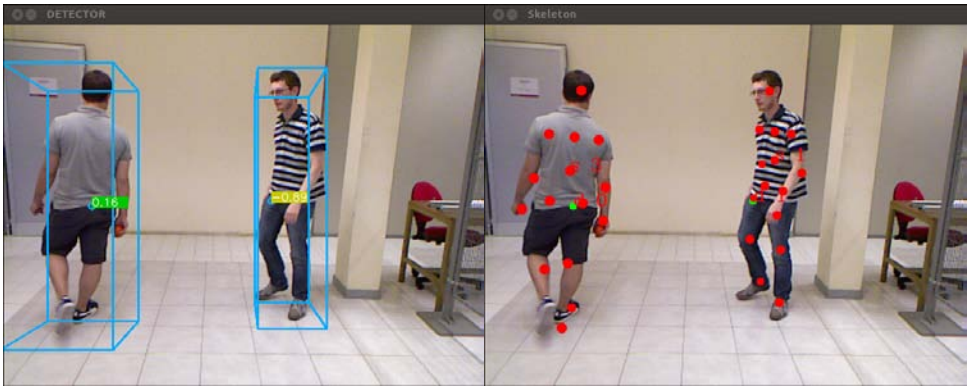


Figura 4.4: Esempio di rilevazioni di persone e dei relativi scheletri associati

Figura 4.5: Esempio di scheletri troppo vicini ad una detection

scheletri è risultata una distanza di 0.21cm per lo scheletro nascosto e di 0.22cm per lo scheletro più vicino alla Kinect. Questo significa che la detection verrebbe associata allo scheletro della persona sbagliata.

In generale i casi che si possono verificare sono i seguenti: sono presenti due scheletri e una detection oppure sono presenti uno scheletro e due detection.

Per riuscire a rilevare entrambi i casi, la soluzione adottata prevede di controllare che una detection sia associata a un solo scheletro e viceversa. Se una detection è stata associata a due scheletri significa che si è verificata una situazione del primo tipo. Se uno scheletro è stato associato a due detection significa che si è verificata una situazione del secondo tipo. Nel caso in cui un'associazione ricada in uno dei due casi si procede ad eliminarla.

4.2.3 Filtraggio dei dati dello scheletro

Sono stati applicati diversi filtraggi per rendere la re-identificazione del tracker più robusta possibile. Il primo filtraggio elimina gli scheletri associati ad una persona che viene occlusa o esce dalla scena. Dato che lo scheletro continua ad essere pubblicato, anche se non più necessario, il problema che ne deriva è la presenza di uno scheletro immobile all'interno della scena e che non è associabile a nessuna persona. Il criterio adottato per eliminare questi scheletri prevede di controllare se il giunto del torso cambia posizione nel tempo. Se il torso rimane sempre nella stessa posizione allora lo scheletro viene eliminato.

Figura 4.6: Esempio di scheletro immobile nella scena

Un esempio del problema è raffigurato nelle immagini in figura 4.6. Nelle tre immagini si vedono due persone in movimento e uno scheletro immobile e non associabile a nessuna persona.

Il filtraggio successivo elimina gli scheletri sulla base della confidenza dei giunti. Se uno scheletro non presenta tutti i giunti stimati bene allora questo viene filtrato. Il motivo di una soglia così stringente deriva da una valutazione visiva della stima degli scheletri quando questi presentano alcuni giunti con una confidenza non massima. Anche se solo

pochi giunti sono male stimati non è detto che visivamente lo scheletro sia ben aderente alla persona. Nella figura 4.7 è presente un esempio di scheletro con una bassa confidenza.

(a) esempio di scheletro stimato in modo errato

(b) esempio di filtraggio sulla confidenza

Figura 4.7: Esempio di scheletro con una bassa confidenza

Infine l'ultimo parametro considerato è la presenza di un'occlusione. Il problema in questo caso è la sovrapposizione di alcuni giunti su un oggetto occludente. Per risolvere il problema è stato inizialmente pensato di eliminare qualsiasi scheletro associato con una detection occlusa.

(a) scheletro occluso non filtrato

(b) scheletro occluso filtrato

Figura 4.8: Esempio di filtraggio su uno scheletro occluso

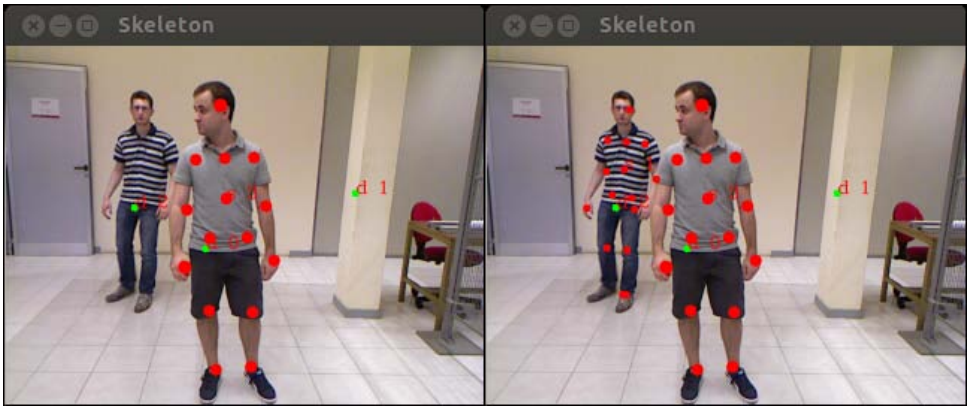
Dato che questo metodo tende a filtrare molti scheletri, anche nei casi in cui questo non sia necessario, è stata successivamente implementata una seconda variante. Prima di eliminare uno scheletro associato ad una detection occlusa viene fatto un controllo tra lo scheletro associato alla detection occlusa e lo scheletro associato alla detection occludente. Se i giunti delle spalle sono sufficientemente distanti, allora lo scheletro associato alla

detection occlusa si può considerare comunque valido. La soglia impostata è di 5cm. Nel caso in cui la detection occludente non abbia uno scheletro, allora lo scheletro associato alla detection occlusa viene eliminato.

La seconda variante permette di eliminare molti meno scheletri rispetto alla prima variante, ma non garantisce che tutti i giunti degli scheletri occlusi siano proiettati nella detection corretta. Questo però accade solo per pochi giunti e non dovrebbe comportare problemi nel confronto tra descrittori.

In figura 4.8 è stato mostrato un esempio del primo filtraggio implementato. Nelle due immagini si vede come la persona più distante sia occlusa dalla persona più vicina. A sinistra è raffigurata la stessa situazione senza filtraggio.

Nella figura 4.9 è stato mostrato l'effetto differente tra il primo e il secondo filtraggio. La distanza tra i giunti dei due scheletri è sufficiente per ritenere valido lo scheletro della persona parzialmente occlusa.



(a) filtraggio occlusioni più agressivo (b) filtraggio occlusioni più rilassato

Figura 4.9: Differenza tra i due tipi di filtraggio sugli scheletri occlusi

4.3 Nodo per il people tracking

Lo scopo di questo nodo è quello di riuscire ad effettuare la re-identificazione a breve termine delle detection presenti nella scena. Originariamente venivano utilizzati tre parametri estratti dalle detection. Il primo parametro è la confidenza dell'HOG detector che permette di distinguere le persone dagli elementi nella scena che vengono erroneamente identificate come persone. Più il valore della confidenza è elevato e maggiore sarà la probabilità di avere rilevato una persona. Il secondo parametro è la stima del moto di una detection. Tramite un filtro di Kalman si prova a stimare dove la persona sarà nel prossimo frame. Questo parametro risulta utile sopratutto se le persone non compiono variazioni improvvise nel loro moto. Il terzo parametro è dato dalla colorazione dell'intera persona calcolata attraverso un classificatore AdaBoost. Se l'obiettivo è la

re-identificazione a breve termine si suppone che la persona non cambi colore quando non è visualizzata.

Tutti e tre questi parametri vengono usati per effettuare un confronto tra tracce e detection. La formula utilizzata per effettuare questo confronto è la seguente:

$$c(i,j) = w_d D(j) + w_m M(i,j) + w_a A(i,j) \tag{4.2}$$

dove $c(i,j)$ è il costo di associazione tra la traccia i e la traccia j, $D(j)$ è la confidenza dell'HOG detector per la detection j, $M(i,j)$ è la distanza di Mahalanobis tra la traccia i e la detection j e $A(i,j)$ è la confidenza del classificatore della traccia i valutato sulla detection j. I valori w_d, w_m, w_a indicano quanto contribuiscono $D(j), M(i,j), A(i,j)$ nel costo totale. Il risultato viene salvato alla posizione (i,j) di una matrice di costo. Una volta volta calcolati tutti i possibili costi, la matrice viene risolta con il metodo Global Nearest Neighbor che cerca di minimizzare il costo di associazione complessivo. Il risultato è una matrice le cui celle indicano a quali tracce le detection sono state associate.

Per confrontare i descrittori dello scheletro è stato usato l'approccio Nearest Neighbor. Dati i descrittori di una traccia il confronto con i descrittori delle detection viene effettuato con la seguente formula:

$$S(i,j) = \sqrt{\sum_{k=1}^{N} \left(D_i[k] - D_j[k]\right)^2} \tag{4.3}$$

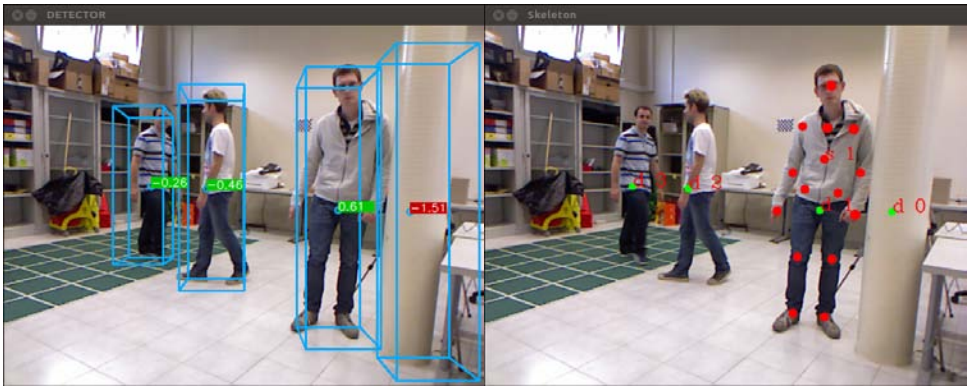
dove $S(i,j)$ è la distanza tra i descrittori dello scheletro della traccia i e della traccia j, D_i è il descrittore della traccia i, D_j è il descrittore della detection j, k è l'indice dei descrittori e N è il numero di elementi all'interno del descrittore.

Per poter aggiungere il contributo dei descrittori dello scheletro nella matrice di costo è stato necessario affrontare un problema. A differenza dagli altri parametri usati per costruire la matrice di costo, non sempre una detection dispone di uno scheletro associato. Questo succede sia perchè lo skeletal tracker non è sempre in grado di fornire uno scheletro associato ad una persona e sia perchè alcuni scheletri vengono eliminati durante il filtraggio del nodo `remapping`. Il problema che ne risulta è l'impossibilità di valutare allo stesso modo le detection associate ad uno scheletro e quelle senza. Una situazione in cui questo si verifica si può vedere nella figura 4.10. Dall'immagine si può notare la presenza di quattro detection anche se nella scena è presente un solo scheletro.

La soluzione adottata prevede di creare due matrici di costo. La prima matrice contiene i costi di associazione tra le detection che sono state associate ad uno scheletro e le tracce presenti nel sistema. I costi di associazione di questa matrice sono stati calcolati aggiungendo all'equazione (4.2) il contributo degli scheletri:

$$\bar{c}(i,j) = c(i,j) + w_s S(i,j) \tag{4.4}$$

dove $c(i,j)$ è il costo di associazione della formula (4.2), $S(i,j)$ viene calcolato come in (4.3) e w_s indica quanto il contributo dello scheletro debba influire. La seconda matrice contiene i costi di associazione tra le detection che non sono state associate ad uno



(a) detection (b) scheletri

Figura 4.10: Situazione in cui alcune detection non sono associabili ad uno scheletro

scheletro e le tracce che non sono già state assegnate risolvendo la seconda matrice. I costi della seconda matrice sono stati calcolati con l'equazione (4.2).

Un esempio per chiarire il procedimento seguito è il seguente. Supponiamo che in un certo istante siano presenti quattro tracce all'interno del sistema e quattro detection provenienti dall'algoritmo di people detection. Se solo due detection sono associate ad uno scheletro, le matrici di costo risultanti saranno simili a quelle riportate in figura 4.11. Le tracce sono riportate nelle righe, le detection sono riportate nelle colonne. La matrice

	D1	D2	D3	D4
T1	20	-15	-	-
T2	-10	25	-	-
T3	15	20	-	-
T4	25	10	-	-

(a) prima matrice di costo

	D1	D2	D3	D4
T1	-	-	-	-
T2	-	-	-	-
T3	-	-	-15	25
T4	-	-	20	-10

(b) seconda matrice di costo

Figura 4.11: Esempio di matrici di costo risolte nella fase di associazione tracce detection

di figura 4.11a è la prima matrice ad essere risolta e contiene i costi di associazione tra le tracce presenti nel sistema e le detection associate ad uno scheletro. In questa matrice le detection senza scheletro sono ignorate. La risoluzione consiste nel minimizzare il costo di associazione complessivo. Le celle disegnate in verde rappresentano l'associazione scelta tra una traccia e una detection. Una volta risolta la prima matrice di costo, si procede a risolvere la seconda matrice di costo(figura 4.11b). In questa matrice sono presenti i costi tra le tracce non associate nella prima matrice di costo e le detection senza scheletro. Nel caso in cui una detection con scheletro non viene associata ad una traccia nella prima matrice di costo, allora la sua associazione sarà risolta nella seconda matrice di costo.

4.4 Integrazione di un face detector con il people detector

Una delle limitazioni del people detector riguarda la distanza minima entro la quale le persone possono essere individuate. Se è visibile la parte superiore di una persona, ma questa si trova molto vicino alla Kinect, è possibile che la sua rilevazione fallisca. Questo deriva dal fatto che se l'immagine della persona esce dal bordo superiore dell'inquadratura, allora il people detector non sarà più in grado di rilevarla correttamente.

Per cercare di risolvere il problema è stato introdotto un face detector basato su feature di HAAR. Le immagini dei cluster su cui non è stata individuata una persona vengono convertite in scala di grigi e poi processate con il face detector. Se viene rilevata una faccia allora il cluster associato viene elaborato per creare la relativa detection. Se non viene rilevata nessuna faccia allora il cluster viene scartato definitivamente.

Dato che il face detector tende a fornire un certo numero di falsi positivi sono state inserite due soglie. La prima soglia permette di considerare solo quei cluster che sono a una distanza sufficientemente ravvicinata. Questo è dovuto al fatto che lo scopo del face detector è valutare solo i cluster frontali e vicini alla Kinect. La soglia è stata impostata a 2 metri. La seconda soglia permette di considerare solo quei cluster con un numero sufficientemente basso di punti. Questa soglia è stata inserita a causa del fatto che la maggior parte dei falsi positivi del face detector sono in realtà oggetti aderenti ad una parete. Di conseguenza il cluster di appartenenza conterrà un numero elevato di punti. La soglia massima è stata fissata a 200 punti.

4.5 Risultati

In questa sezione verranno riportati i test eseguiti sul tracker e le relative considerazioni.

4.5.1 Test

La registrazione su cui è stato testato il tracker dura 75s. Le persone nella scena si muovono con traiettorie non lineari, possono occludersi, uscire e rientrare nel campo visivo della Kinect. In totale ci sono 5 persone. Nella figura 4.12 sono riportate alcune immagini della registrazione.

I test sono stati fatti variando il peso utilizzato per lo scheletro e mantenendo costanti i pesi utilizzati per il colore(-65), il moto(0.05) e la confidenza del detector(-1).

Per valutare l'efficacia della re-identificazione è stato considerato il numero di switch tra le tracce presenti nella scena. Gli switch vengono conteggiati ogni qual volta un utente all'interno della scena cambia il proprio identificativo. Questo può avvenire quando un utente riappare nella scena ma non viene re-identificato in modo corretto oppure quando un utente acquisisce l'identificativo di un altro utente.

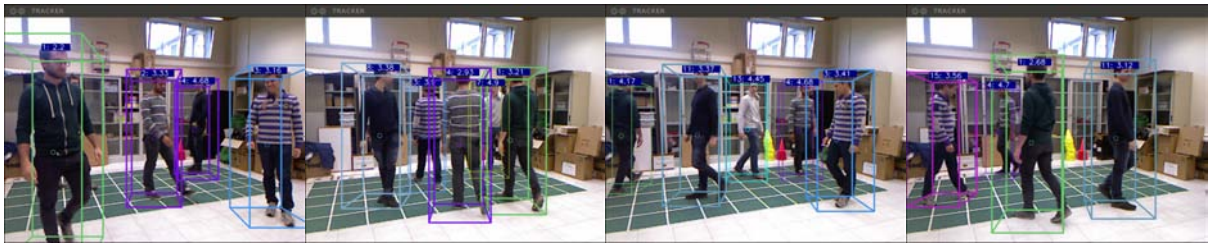


Figura 4.12: Alcuni frame presi dalla registrazione testata

Dato che anche il numero di persone totali rilevate influenza il numero di switch, è stata riportato anche il MOTA [28] che indica il numero di errori commessi in termini di falsi positivi e falsi negativi. In questo modo è più semplice effettuare un confronto con test di altri lavori.

Peso scheletro	Senza scheletro	0	0.01	0.02
Switch	25	31	40	58
MOTA	73%	73%	73%	73%

Tabella 4.1: Risultati ottenuti variando il peso associato allo scheletro nella formula (4.4)

I test sono riportati in tabella 4.1. Il primo valore corrisponde al numero di switch ottenuti senza considerare il contributo dello scheletro. Tutti i valori successivi corrispondono al numero di switch ottenuti assegnando nella formula (4.4) il peso associato.

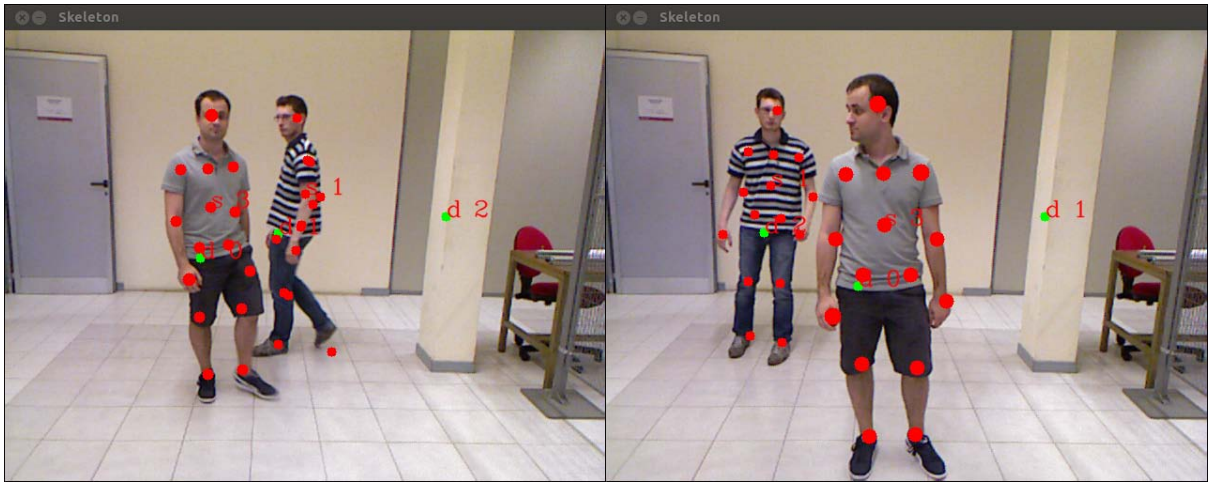
Oltre a delle valutazioni quantitative sono state effettuate della valutazioni qualitative in altre due registrazioni. Le immagini di queste registrazioni sono riportate nelle figure di sezione 4.2. Anche in queste registrazioni l'utilizzo dei descrittori dello scheletro porta ad un peggioramento del numero di switch.

4.5.2 Considerazioni

Dai risultati in tabella 4.1 si può capire che l'utilizzo dello skeletal tracker non porta ad alcun beneficio. Il motivo principale che impedisce una corretta re-identificazione è la scarsa robustezza ai cambi di posa. Questo deriva dal fatto che se i descrittori di una persona vengono confrontati a distanza di tempo non è detto che i punti su cui i giunti vengono proiettati siano sempre gli stessi.

Un esempio di queste situazioni è riportato nelle immagini di figura 4.13. In questi immagini si può vedere come la posa della persona più distante cambi prima e dopo un'occlusione. Il problema che ne consegue è l'impossibilità di garantire una corretta re-identificazione utilizzando i descrittori calcolati con SIFT.

Un altro problema fondamentale riguarda la presenza di due matrici di costo durante la fase di associazione tra le detection e le tracce. Questo comporta un aumento degli



(a) immagine prima di un'occlusione (b) immagine dopo un'occlusione

Figura 4.13: Posa differente di un utente prima e dopo un'occlusione

switch anche se nelle due matrici di costo non è presente il contributo degli scheletri. Questo aumento si può vedere dalla tabella 4.1. Nella prima colonna sono riportati gli switch ottenuti usando i parametri presenti in origine(confidenza, moto, colore) con una solo matrice di costo. Nella seconda colonna sono riportati gli switch ottenuti usando due matrici di costo ma ponendo a zero il peso associato agli scheletri. Il peggioramento è dovuto al fatto che l'utilizzo di due matrici di costo impedisce una minimizzazione del costo globale.

Durante lo svolgimento del progetto era stata presa in considerazione una tecnica multiframe per superare il problema derivante dalla variazione di posa. Questa tecnica consiste nell'accumulare N descrittori all'interno di una traccia e confrontarli con i descrittori di una detection. Purtroppo anche applicando questa tecnica non sarebbe possibile riuscire a migliorare la re-identificazione. Infatti anche se fosse possibile accumulare i descrittori associati ad una persona da ogni punto di vista, non sarebbe poi possibile utilizzarli una volta che questa ritorni visibile dopo un occlusione o dopo l'uscita dall'inquadratura. Questo problema deriva dall'incapacità dello skeletal tracker utilizzato nel calcolare in modo immediato lo scheletro associato ad una persona nei primi frame in cui questa appare nella scena. Di conseguenza non è possibile confrontare la detection di questa persona con le tracce presenti nel sistema.

Capitolo 5

Conclusioni e sviluppi futuri

In questo lavoro è stato sviluppato un sistema in grado di localizzare e re-identificare delle persone all'interno di un ambiente sconosciuto ed è stata esplorata la possibilità di utilizzare delle tecniche di skeleton recognition per migliorare il tracking di persone.

Il sistema per la localizzazione e la re-identificazione di persone all'interno di un ambiente sconosciuto ha previsto la collaborazione di diverse attività tra cui lo SLAM, la navigazione, l'esplorazione, il following di una persona, la rilevazione e il tracking di persone e la loro re-identificazione a lungo termine. Per quanto riguarda lo SLAM, la navigazione e l'esplorazione è stata fatta un'opportuna selezione degli algoritmi presenti nello stato dell'arte. Per quanto riguarda la rilevazione delle persone e la loro identificazione sono stati utilizzati algoritmi forniti ad inizio progetto. Gli algoritmi di navigazione ed esplorazione sono stati opportunamente configurati per poter funzionare in modo più sicuro ed efficacie durante gli spostamenti del robot. L'algoritmo di rilevazione e tracking di persone è stato modificato per poter distinguere tra le false rilevazioni e le persone effettivamente presenti nell'ambiente. L'algoritmo di re-identificazione a lungo termine è stato invece modificato per poter effettuare il riconoscimento solo su una specifica persona. Infine la fase di following è stata migliorata utilizzando l'algoritmo di navigazione al fine di evitare gli ostacoli che il robot può incontrare nel seguire un utente.

I test effettuati hanno dimostrato che il sistema è in grado di far cooperare correttamente tutte le attività di cui è costituito. In particolare il sistema è in grado di esplorare un ambiente, rilevare le persone all'interno di questo ambiente, re-identificarle a lungo termine dopo una fase di avvicinamento ed utilizzare il tracking di persone al fine di seguirle durante la fase di following. Durante lo svolgimento dei test sono state trovate delle possibili limitazioni dovute all'hardware e agli algoritmi utilizzati. I problemi riscontrati riguardano principalmente la mancata rilevazione di una persona da parte dell'algoritmo di people detection oppure il mancato riconoscimento di una persona da parte dell'algoritmo di re-identificazione a lungo termine.

Per quanto riguarda il miglioramento del tracking di persone è stato utilizzato lo skeletal tracker di OpenNI da cui si è cercato di calcolare dei descrittori quanto più affidabili possibile. Il procedimento seguito ha previsto il calcolo dei descrittori a partire dai giunti degli scheletri restituiti da OpenNI, l'associazione dei descrittori con le persone

presenti nella scena e infine l'utilizzo dei descrittori dello scheletro durante la fase di re-identificazione a breve termine. I test quantitativi hanno riguardato una registrazione in cui le persone potevano muoversi con traiettorie non lineari, occludersi, uscire e rientrare nella scena. L'esito dei test è stato negativo in quanto la tecnica utilizzata non è robusta ai cambi di posa. Inoltre è stato constato che l'utilizzo di una tecnica multiframe non porterebbe a nessun miglioramento, dato che lo skeletal tracker di OpenNI non è in grado di estrarre i giunti di una persona appena questa riappare nella scena.

La prima parte di questa tesi apre alla possibilità ad innumerevoli sviluppi futuri. Il sistema sviluppato potrebbe essere utilizzato per poter sorvegliare una zona oppure per poter assistere un anziano. In quest'ultimo caso un possibile miglioramento del sistema potrebbe essere la creazione di una tabella per indicare in quale posizione si trova l'anziano in funzione di una certa ora. Questa tabella può essere inserita manualmente oppure può essere creata dal sistema stesso memorizzando le abitudini di una persona. Un'ulteriore estensione potrebbe riguardare l'utilizzo della Kinect per avere una ricostruzione tridimensionale della scena. Questo permetterebbe di evitare gli ostacoli non rilevabili dal laser scanner rendendo il sistema più sicuro.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] Stefan Kohlbrecher, Johannes Meyer, Thorsten Graber, Karen Petersen, Uwe Klingauf, Oskar von Stryk. *Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots*. RoboCup, 2013.

[2] Z. Cheng, X. Zhang, S. Yu, Y. Ou, X. Wu, Y. Xu. *A Surveillance Robot with Human Recognition Based on Video and Audio.* In IEEE International Conference on Robotics and Biomimetics (ROBIO2010), pp. 1256 - 1261, 2010.

[3] X. Wu, H. Gong, P. Chen and Y. Xu. *Intelligent Household Surveillance Robot.* In IEEE International Conference on Robotics and Biomimetics (ROBIO2008), pp. 1734-1739, 2009.

[4] Di Paola D., Milella A., Cicirelli G. and Distante A. *An autonomous mobile robotic system for surveillance of indoor environments*. international Journal of Advanced Robotic Systems, Vol. 7 No. 1, pp. 19-26, 2010.

[5] Volkhardt M., Gross H.M. *Finding people in home environments with a mobile robot.* in Proc. 6th European Conference on Mobile Robots (ECMR 2013). (Sept 2013), pp. 282-287.

[6] Mehdi S. A, Berns K. *Probabilistic Search of Human by Autonomous Mobile Robot*. 4th International Conference on Pervasive Technologies Related to Assistive Environments (PETRA). Crete, Greece 2011.

[7] M. Farenzena, L. Bazzani, A. Perina, V. Murino, and M. Cristani. *Person re- identification by symmetry-driven accumulation of local features*. in Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010), pages 2360–2367.

[8] M. Andriluka, S. Roth, and B. Schiele. *Pictorial structures revisited: People detection and articulated pose estimation*. in CVPR, pages 1014–1021, 2009.

[9] I.O. de Oliveira and J.L. de Souza Pio. *People reidentification in a camera network*. in Proceedings of the 8th IEEE Conference on Dependable, Autonomic and Secure Computing (DASC 2009), pages 461–466.

[10] J. Oliver, A. Albiol, and A. Albiol. *3d descriptor for people re-identification*. In Proceedings of the 21st IEEE International Conference on Pattern Recognition (ICPR 2012), pages 1395–1398.

[11] K. Khoshelham and S. Elberink. *Accuracy and resolution of kinect depth data for indoor mappingapplications*. Sensors, vol. 12, no. 2, pp.1437 -1454 2012

[12] M. Quigley et al. *ROS: an open-source Robot Operating System*. In IEEE International Conference on Robotics and Automation(ICRA), Workshop on Open Source Software, 2009.

[13] J. Machado Santos, D. Portugal, and R. P. Rocha. *An Evaluation of 2D SLAM Techniques Available in Robot Operating System*. In Proc. of IEEE 13th Int. Symposium on Safety, Security and Rescue Robotics, Sweden, Oct. 2013.

[14] S. Kohlbrecher and J. Meyer and O. von Stryk and U. Klingauf. *A Flexible and Scalable SLAM System with Full 3D Motion Estimation*. Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), 2011.

[15] G. Grisetti, C. Stachniss, and W. Burgard. *Improved techniques for grid mapping with rao-blackwellized particle filters*. Robotics, IEEE Transactions on, vol. 23, no. 1, pp. 34-46, feb. 2007

[16] B. Steux, O. El Hamzaoui. *tinySLAM: A SLAM algorithm in less than 200 lines C-language program*. In Proc. of the Int. Conf. on Control Automation Robotics & Vision (ICARCV), Dec. 2010.

[17] L. Carlone, R. Aragues, J.A. Castellanos, and B. Bona. *A linear approximation for graph-based simultaneous localization and mapping*. In Proc. of the Int. Conf. Robotics: Science and Systems, 2011.

[18] M. Munaro, F. Basso, E. Menegatti. *Tracking people within groups with RGB-D data*. In Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2012, Vilamoura (Portugal), 2012.

[19] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige. *The office marathon: Robust navigation in an indoor office environment*. In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 300-307, May 2010.

[20] M. Seder and I. Petrovic. *Dynamic window based approach to mobile robot motion control in the presence of moving obstacles*. Proc. IEEE Int. Conf. Robot. Autom., pp.1986 -1992 2007

[21] A. Stentz. *The Focussed D* Algorithm for Real-Time Replanning*. International Joint Conference on Artificial Intelligence, 1995.

[22] D. Fox and W. Burgard and S. Thrun. *The Dynamic Window Approach to Collision Avoidance*. IEEE Robotics & Automation Magazine, 4(1), pp. 23-33, 1997.

[23] Brian P. Gerkey and Kurt Konolige. *Planning and Control in Unstructured Terrain.* Discussion of the Trajectory Rollout algorithm in use on the LAGR robot.

[24] Yamauchi, B. 1997. *Frontier-based approach for autonomous exploration.* In Proceedings of the IEEE International Symposium on Computational Intelligence, Robotics and Automation, pp. 146-151.

[25] S. Wirth and J. Pellenz. *Exploration Transform: A Stable Exploring Algorithm for Robots in Rescue Environments.* In IEEE International Workshop on Safety, Security & Rescue Robotics, 2007, pp. 1-5.

[26] M. Quigley et al. *ROS: an open-source Robot Operating System.* In IEEE International Conference on Robotics and Automation(ICRA), Workshop on Open Source Software, 2009.

[27] M. Munaro, S. Ghidoni, D. Tartaro Dizmen and E. Menegatti. *A Feature-based Approach to People Re-Identification using Skeleton Keypoints.* In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Hong Kong (China), 2014.

[28] K. Bernardin and R. Stiefelhagen. *Evaluating multiple object track- ing performance: the clear mot metrics.* J. Image Video Process., 2008:1:1–1:10, January 2008.

[29] D. Lowe. *Object recognition from local scale-invariant features.* In Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV 1999), vol. 2, pp. 1150–1157.

[30] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. *Speeded-up robust features (surf).* Comput. Vis. Image Underst., vol. 110, no. 3, pp. 346–359, June.

[31] R. Rusu, N. Blodow, Z. Marton, and M. Beetz. *Aligning point cloud views using persistent feature histograms.* In Proc. of the 2008 International Conference on Intelligent Robots and Systems (IROS 2008), pp. 3384–3391.

[32] R. Rusu, N. Blodow, and M. Beetz. *Fast point feature histograms (fpfh) for 3d registration.* In Proc. of the 2009 International Confer- ence on Robotics and Automation (ICRA 2009), pp. 3212–3217.