

UNIVERSITÀ DEGLI STUDI DI PARMA
FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Informatica

**ELABORAZIONE DI SCANSIONI LASER
PER LA COSTRUZIONE INCREMENTALE
DI MAPPE**

Relatore:
Chiar.mo Prof. STEFANO CASELLI

Correlatori:
Ing. DARIO LODI RIZZINI

Tesi di laurea di:
GIONATA BOCCALINI

ANNO ACCADEMICO 2007-2008

ciao!!

Indice

1	Introduzione	2
2	Scan Matching	4
2.1	Relazione tra scansioni laser e mappe	5
2.2	Iterative Closest Point	6
2.3	Canonical Scan Matching	7
2.4	Acquisizione scansioni	7
2.5	Risultati	9
3	Rappresentazioni dell'ambiente	15
3.1	Segmentazione delle scansioni	15
3.1.1	Split & Merge	16
3.1.2	Trasformata di Hough	18
3.1.3	Utilizzo della Trasformata di Hough	20
3.2	Rappresentazioni tramite mappe	22
3.3	Mappe a griglia e collezioni di mappe	24
3.4	Mappe a features	28
4	Associazione di scansioni e mappe	32
4.1	Metodi di associazione	33
4.2	Stima dell'orientamento principale	36
5	Risultati	38
5.1	Estrazione di features con Split&Merge	38
5.2	Errori sulla stima della traiettoria	39
5.3	Allineamento scansioni mappe	41
6	Conclusioni	45

Capitolo 1

Introduzione

La localizzazione di un sistema robotico autonomo è da sempre uno dei problemi più ardui e di più difficile comprensione nell'ambito della robotica. Per localizzazione si intende la capacità di un robot mobile di sapere, o di riconoscere, la sua posizione all'interno di un ambiente in un determinato istante; questo ambiente è spesso non conosciuto al robot, che quindi deve anche creare una *mappa* dello spazio circostante per poter poi essere in grado di capire la sua posizione attuale. Questo procedimento prende il nome di *SLAM* (Simultaneous Localization and Mapping), e si basa su tecniche di scansione e riconoscimento dell'ambiente, alcune delle quali saranno descritte in questa tesi.

Una scansione, che può essere acquisita tramite sensori quali il laser scanner o il sonar, è una nuvola di punti disposti su un piano, nel caso di un robot mobile che opera nel piano, che rappresentano gli ostacoli percepiti dal robot durante il suo movimento nell'ambiente. In questo modo possono giungere al robot le informazioni basilari sullo spazio circostante, ricostruite tramite algoritmi di allineamento di scansioni che sono largamente utilizzati in questa tesi.

Una scansione laser è uno dei metodi maggiormente utilizzati con il quale un robot può osservare il mondo esterno, che per sua stessa natura è in genere di difficile interpretazione, e crea quindi diversi tipi di imprecisioni, causate dalle limitazioni nell'uso dei sensori e dalle approssimazioni che possono essere fatte. Anche gli algoritmi necessari per il trattamento dei dati includono spesso delle approssimazioni, e quindi il risultato è soggetto a margini di errore più o meno grandi. Per avere una stima di questo margine sono stati eseguiti alcuni test sugli algoritmi utilizzati per capire il grado di precisione a cui possono arrivare.

Questo lavoro di tesi ha come obiettivo lo studio e la realizzazione di un sistema in grado di analizzare ed elaborare le scansioni laser acquisite per ottenere una rappresen-

tazione coerente dell'ambiente in cui si trova il robot, e risolvere alcuni dei problemi connessi, come la localizzazione del robot mobile.

L'attività si è concentrata principalmente su due aspetti. In primo luogo il recupero delle relazioni spaziali locali tra scansioni consecutive, o comunque acquisite in posizioni tra loro prossime. In letteratura questo problema è stato risolto con metodi di allineamento delle scansioni o *scan matching*. Le tecniche di scan matching consentono di ridurre gli errori introdotti dall'odometria, ma non eliminano le inconsistenze dovute all'incertezza nella relazione spaziale tra pose lontane. La ricerca di una rappresentazione dell'ambiente di più alto livello consente da un lato il recupero di relazioni spaziali globali e agevola la manipolazione della mappa da parte di algoritmi.

Il secondo aspetto affrontato consiste nella elaborazione dei dati contenuti nella mappa con lo scopo di ottenere informazioni sulla posizione globale del robot. Il problema del *data-association* è uno dei punti principali necessari alla costruzione di un algoritmo di SLAM, e consiste nell'associazione di dati contenuti nelle misure sensoriali con quelli ottenuti dalla mappa. Un buon risultato nell'associazione di scansioni e mappe è fondamentale per costruire un robusto metodo di localizzazione: questo risultato sarà poi utilizzato per risolvere il problema della chiusura del *loop*, cioè il riconoscimento da parte del robot di una zona di spazio che è già stata esplorata.

La tesi è organizzata nel seguente modo. Nel capitolo 2 vengono introdotti i concetti basilari riguardo le tecniche di tracciamento della posizione di un robot mobile, e la rappresentazione dell'ambiente tramite scansioni e mappe. Successivamente vengono espone alcune nozioni teoriche sugli algoritmi utilizzati, lo *Scan Matching* in particolare, in relazione alle specifiche implementazioni usate, e vengono presentati i risultati sperimentali ottenuti dai test condotti per stimare la precisione di questi algoritmi. Nel capitolo 3 vengono descritte alcune delle possibili rappresentazioni per l'ambiente esplorato dal robot mobile, tra cui l'utilizzo di diversi tipi di mappe, e vengono dettagliati alcuni algoritmi di elaborazione per scansioni necessari alla creazione di mappe. Nel capitolo 4 è presentato il problema derivante dall'associazione di scansioni e mappe, cioè capire quando queste rappresentano la stessa porzione di territorio; sono espone le soluzioni implementate durante questo lavoro di tesi diversificate dalla consistenza dei risultati ottenuti.

Il capitolo 5 espone i risultati dei test eseguiti sugli algoritmi, sia in forma numerica che grafica, in particolare quelli riguardanti una stima della traiettoria percorsa dal robot e quelli inerenti al confronto tra scansioni e mappe.

Infine nel capitolo 6 sono espone le considerazioni finali sul lavoro svolto e sui possibili sviluppi futuri.

Capitolo 2

Scan Matching

In letteratura, il termine scan matching si riferisce al problema della ricerca di una trasformazione rigida che garantisca la migliore sovrapposizione di due scansioni acquisite con laser scanner, un sensore di distanza 2D. L'algoritmo confronta due scansioni consecutive e cerca di trovare un insieme di trasformazioni 2D (una traslazione e una rotazione) che le facciano sovrapporre il più possibile. Quindi se il robot nella posizione di riferimento cattura una scansione S_{ref} , si sposta in una nuova posizione, e cattura una nuova scansione S_{new} , lo scan matching cerca di individuare la nuova posizione del robot confrontando le due scansioni e sovrapponendole il più possibile [1]. Questo algoritmo ha bisogno di una stima iniziale per avere un riferimento nel confronto tra le scansioni: la stima è fornita dal valore dell'odometria, che fornisce un allineamento approssimato. Successivamente l'allineamento viene migliorato riducendo iterativamente la distanza tra le scansioni. In alcuni algoritmi il matching dei punti viene effettuato servendosi delle direzioni tangenti, ed estraendo le rette che congiungono i punti; in altri casi la corrispondenza tra le scansioni è definita punto per punto, prendendo come riferimento un punto della scansione sorgente, e cercando il punto più vicino nella nuova scansione, con algoritmi come l'Iterative Closest Point (ICP), descritto nel paragrafo 2.2.

I risultati di un algoritmo di scan matching dipendono molto dal tipo di scansione in esame, e da alcune discrepanze che si possono rilevare tra due scansioni prese in due pose diverse: queste discrepanze possono essere dovute all'errore commesso dal sensore laser durante la scansione, ed anche all'occlusione cui può essere soggetta una scansione. Infatti alcune zone dell'ambiente possono essere visibili nella scansione di riferimento, ma non nella successiva; queste parti vengono ignorate, mentre l'algoritmo si concentra sul matching delle zone condivise da entrambe le mappe. Per questo motivo in generale gli algoritmi di scan matching danno migliori risultati quan-

do le scansioni sono prese a breve distanza tra di loro, e quando sono presenti elementi architettonici ben precisi (come muri o angoli), che aumentano la definizione delle scansioni.

In questo lavoro di tesi è stata utilizzata una particolare versione dello scan matching, che sarà descritta nel paragrafo 2.3, mentre i risultati sperimentali delle stime effettuate saranno presentati in 2.5.

2.1 Relazione tra scansioni laser e mappe

Durante il suo movimento un robot mobile sul piano può assumere diverse configurazioni rispetto ad un sistema di riferimento solidale con l'ambiente, dette *pose* e comprendono due coordinate per la posizione, e un angolo che dà l'orientazione rispetto ad un sistema di riferimento, relativo o assoluto. La tecnica di misura più semplice per stabilire la relazione tra coppie di pose è l'odometria, e consiste nel calcolare la nuova posizione del robot basandosi su valori dei sensori di movimento, montati direttamente sull'albero motore o sulle ruote del robot. Tale informazione è inaffidabile e imprecisa, soprattutto se viene compiuto un percorso "tortuoso", dove gli errori introdotti dalla diversa percorrenza di una ruota rispetto all'altra in curva sono rilevanti. Per ovviare a questo problema è possibile utilizzare un laser scanner, il quale restituisce un profilo abbastanza preciso dell'ambiente *localmente* visibile. Una scansione laser così ottenuta contiene comunque dati incerti, ma migliora drasticamente le prestazioni quando si tratta di calcolare le diverse pose assunte dal robot. L'incertezza può derivare dalla scarsa qualità del sensore ovviamente, ma anche dalla topologia dell'ambiente: per esempio in una zona *indoor*, con pochi elementi architettonici e pareti rettilinee lo scan matching ha prestazioni tipicamente inferiori rispetto ad una zona con numerosi punti di rottura della linearità dell'ambiente. Alcuni esempi di questo saranno mostrati in seguito. Una parte di questo lavoro di tesi è proprio dedicata ad ottenere una stima sull'errore commesso da algoritmi quali lo scan matching che consente di sovrapporre al meglio due zone di punti, le scansioni, e garantisce risultati migliori rispetto ad altre tecniche come l'odometria. Il confronto di scansioni acquisite dal robot in posizioni diverse permette di recuperare informazioni sull'ambiente nel suo complesso, e di costruire quindi una mappa globale della zona, composta da altre mappe locali. Queste mappe contengono solo una sezione dello spazio circostante, ma vengono composte insieme e il risultato è memorizzato dentro una struttura dati adeguata, per generare una "pianta" completa dell'ambiente. Le mappe utilizzate in questa tesi sono di due tipi: le mappe a griglia, composte da una matrice di celle in cui ogni cella può contenere un valore nu-

merico che rappresenta la porzione di spazio occupato in quella posizione, e la mappatura a *features*, che sono costituite da un certo numero di elementi estratti dalle scansioni, elementi che rappresentano le principali caratteristiche strutturali dell'ambiente.

2.2 Iterative Closest Point

L'algoritmo ICP è un metodo iterativo basato sull'associazione punto per punto tra due scansioni, ed è molto usato nell'implementazione di algoritmi di scan matching. Le fasi principali dell'algoritmo possono essere riassunte come segue: per ogni punto P_i nella scansione S_{new} viene usata una "regola" per determinare il corrispondente punto P'_i , il quale può trovarsi nella scansione S_{ref} , oppure può essere il risultato dell'interpolazione di due punti contenuti nella scansione di riferimento. Successivamente dall'insieme di corrispondenze viene estrapolata una soluzione in forma chiusa per la rotazione e la traslazione relativa tra le scansioni. L'algoritmo viene ripetuto fino ad ottenere la convergenza e la soluzione è applicata per ridurre l'errore di posizione tra le pose assunte dal robot.

La soluzione viene calcolata minimizzando la distanza tra le scansioni, espressa dalla seguente formula:

$$E_{dist}(\omega, T) = \sum_{i=1}^n |R_\omega P_i + T - P'_i|^2$$

dove n rappresenta il numero di corrispondenze trovate, mentre ω è l'angolo di rotazione e $T = (T_x, T_y)$ è il vettore di traslazione. Il punto di maggior interesse dell'algoritmo risiede nella scelta della regola di associazione da utilizzare. Nella formula originale dell'algoritmo viene utilizzata la *Closest Point Rule*, che dà il nome all'algoritmo, e consiste nel creare le corrispondenze scegliendo il punto più vicino al punto di riferimento nella scansione S_{new} . Questa soluzione garantisce buoni risultati ma l'algoritmo converge lentamente [1]; per aumentare la velocità di convergenza è stata introdotta una nuova regola, chiamata *Matching-Range Point*, che calcola le corrispondenze in modo che sia più semplice estrarre l'orientazione come risultato dell'algoritmo. La regola si basa sulla relazione tra le orientazioni di due punti corrispondenti, P_i e P'_i : $\theta' \approx \theta + \omega$, che implica, per rotazioni ω piccole ed ignorando la traslazione, che il punto P'_i è una buona approssimazione del punto P_i . Per trovare un solo risultato nella corrispondenza si è scelto di limitare $\omega = \theta' - \theta \leq B_\omega$; quindi il punto P'_i deve soddisfare la $|\theta' - \theta| \leq B_\omega$ e deve essere il più vicino a P_i .

2.3 Canonical Scan Matching

Lo scan matching utilizzato in questo lavoro differisce da quello standard nell'implementazione dell'ICP. L'algoritmo utilizzato prende il nome di PLICP [2] a causa dell'uso di una metrica punto-retta, che cerca di ovviare ai problemi noti dell'ICP classico, come la convergenza ad un valore errato se l'errore iniziale è grande, o la convergenza lenta dovuta anche alla presenza di oclusioni nella scansione. L'uso della metrica punto-retta consente all'algoritmo di convergere più rapidamente rispetto all'ICP classico e, nel caso in cui le scansioni siano costituite da poli linee, l'algoritmo è in grado di convergere in un numero finito di step con una buona probabilità. Questo perché la metrica punto-retta approssima meglio l'andamento di una poli linea rispetto ad una metrica punto-punto. Lo svantaggio principale di questa implementazione consiste nella poca robustezza dell'algoritmo quando tra le due scansioni c'è una grande differenza di posizione e orientamento. Questo problema tuttavia non si pone in questo lavoro, siccome le scansioni sono state prese ad intervalli di tempo e di spazio molto piccoli.

2.4 Acquisizione scansioni

Per ottenere un qualsiasi tipo di risultato da un algoritmo di scan matching è necessario acquisire i dati provenienti dal laser scanner, e salvarli in un formato standard in modo da aumentare la portabilità dell'applicazione. Di seguito viene descritto il procedimento di cattura di una scansione, composto da diverse fasi. I dati provengono da un laser scanner Sick LMS 200 [3] installato su una piattaforma robotica Pioneer I prodotta dalla Active Media [4] (figura 2.1), equipaggiata con sensori odometrici e computer on board.

Il laser scanner funziona come un sensore di prossimità, capace di misurare la distanza tra il robot (o meglio il centro del sistema di riferimento fissato sul laser) e l'ostacolo più vicino. Per effettuare la misura viene emesso un segnale laser con velocità nota, il segnale rimbalza contro un ostacolo e ritorna al sensore, che calcola il tempo impiegato dal segnale e quindi la distanza in metri. Per aumentare le prestazioni vengono emessi più raggi (*beam*), fino a coprire un certo angolo di visuale, che per il laser scanner usato è di 180° . A seconda della risoluzione specificata può cambiare il numero di beam emessi per ogni scansione; le risoluzioni ammesse per il modello LMS 200 sono: un quarto di grado ($0, 25^\circ$), mezzo grado ($0, 5^\circ$) o un grado (1°). In questo lavoro è stata utilizzata la risoluzione di 0.5° , quindi ogni scansione è composta da 360 beam.

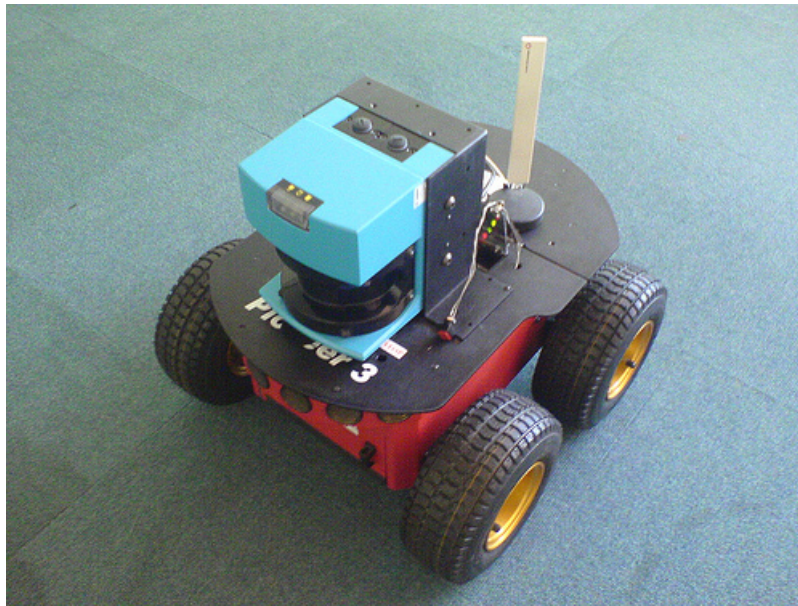
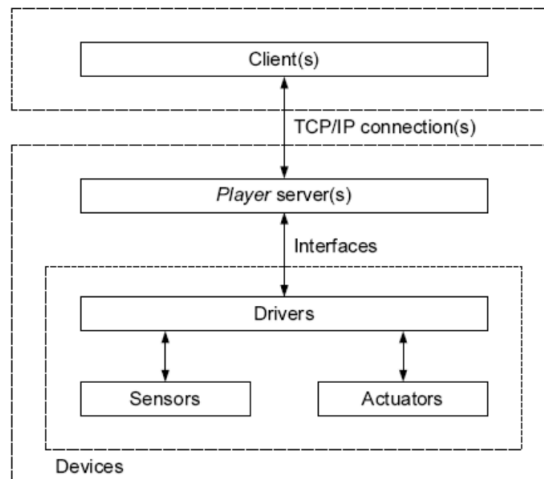


Figura 2.1: Piattaforma Pioneer

Per interfacciarsi con la piattaforma si è fatto ricorso al framework robotico Player-Stage [5], che fornisce uno strato che separa l'hardware dalla specifica applicazione di SLAM, la quale comunica con Player tramite socket TCP, permettendo così l'esecuzione da un nodo remoto. Stage permette di simulare la presenza di una piattaforma robotica e di un mondo circostante nel caso non ve ne sia uno a disposizione.



Robot reale

Figura 2.2: Struttura del framework Player-Stage

In figura 2.2 è mostrata la struttura generale del framework nel caso venga usato un robot reale. L'organizzazione interna di Player è costituita da diversi moduli, og-

nuno dei quali si occupa di fornire l'accesso ad una componente del robot, e mette a disposizione i metodi necessari per recuperare i valori delle misurazioni effettuate (dati provenienti dal laser scanner e dai sensori odometrici, e informazioni sulla posizione attuale).

Una volta instaurata la connessione TCP/IP le scansioni rilevate dal laser scanner vengono catturate da un processo Player in esecuzione sul Pioneer, e successivamente memorizzate in un file di log.

Il file è conforme al formato CARMEN [6], ampiamente utilizzato nella raccolta e salvataggio di dati provenienti da scansioni laser: all'inizio di ogni riga significativa è presente un *tag*, cioè una parola chiave che identifica il contenuto del resto della riga. Un esempio di riga di un file di log è il seguente:

```
FLASER BEAMSNUMBER [  $x_1$  .....  $x_n$  ] 0 0 0 ODOMX ODOMY ODOM $\Theta$ 
```

dove gli x_n sono i valori di distanza corrispondenti ad ogni beam, mentre gli ODOM sono i valori x, y e Θ della posa assunta dal robot, secondo l'odometria. Il formato CARMEN rende più semplici le operazioni di lettura del log e l'applicazione diventa così compatibile con praticamente tutti i file di log che è possibile incontrare.

2.5 Risultati

Le scansioni acquisite in ambienti indoor, in cui non è presente arredamento o altri oggetti, possono essere facilmente decomposte in segmenti rettilinei corrispondenti alle pareti e agli altri elementi architettonici. La presenza di elementi geometrici semplici aiuta l'attività di valutazione delle prestazioni dell'algoritmo di scan matching. Infatti questi elementi permettono di individuare facilmente alcune caratteristiche delle scansioni in esame, che poi saranno utilizzate per confrontare le scansioni e stimare l'errore commesso. Nelle prove compiute viene effettuata l'estrapolazione dell'orientazione principale di una scansione laser, che viene usata come metodo di confronto tra scansioni, anche non consecutive, per avere una stima iniziale delle differenze.

Bisogna anche considerare che l'ambiente di test in cui sono stati ottenuti questi risultati è costituito da un corridoio, dove le pareti rettilinee e parallele non favoriscono la precisione dello scan matcher. In figura 2.3 è mostrata una possibile situazione che si verifica in questo tipo di ambienti, dove il matching tra due scansioni potrebbe risultare parziale.

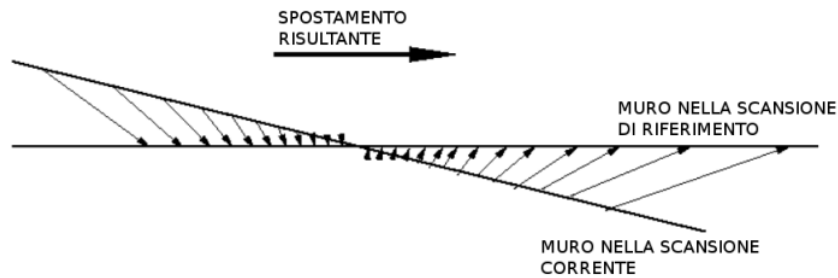


Figura 2.3: Disallineamento tra scansioni in un ambiente con pareti rettilinee

Considerando l'uso della trasformata di Hough per l'estrazione di features in alternativa allo Split&Merge, perché più precisa e meno sensibile alle variazioni dell'*environment*, vengono presentati i risultati ottenuti testando le prestazioni dello scan matcher: il confronto dei segmenti ottenuti, e della media degli errori su θ attraverso l'odometria produce i risultati mostrati in tabella 2.1, in funzione del numero di scansioni saltate.

# SCANSIONI SALTATE	ERRORE ANGOLARE MEDIO	DEVIAZIONE STD	DISTANZA MEDIA (M)
0	0.0090	0.0122	0.0095
1	0.0046	0.0064	0.0116
4	0.0102	0.0093	0.0259
8	0.0090	0.0080	0.0449
12	0.0160	0.0166	0.0989
14	0.0124	0.0101	0.1057
18	0.0532	0.0710	0.2753
20	0.0248	0.0264	0.1480
26	0.0291	0.0290	0.2790
35	0.0629	0.1194	0.7185
40	0.0409	0.1026	0.6395
45	0.0291	0.0273	0.7249
55	0.0322	0.0422	0.8596
65	0.0221	0.0214	1.0525

Tabella 2.1: Risultati odometria

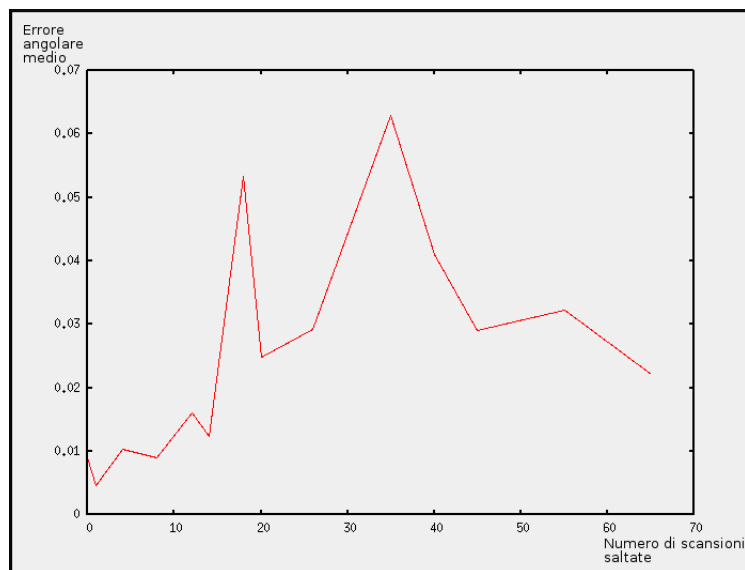


Figura 2.4: Errori medi odometria

All'aumentare dell'intervallo di scansioni ci si attende un andamento strettamente crescente del valore dell'errore. In realtà si osservano alcuni improvvisi picchi di massimo dovuti probabilmente al movimento incerto del robot, e considerando oltretutto un errore nella misura dell'odometria che cresce ad ogni spostamento del robot, sommandosi a quello accumulato fino all'istante precedente. Lo scopo dello scan matching è proprio quello di fornire una stima della posizione del robot in cui sia minimizzato l'errore accumulato dall'odometria.

Nella tabella 2.2 si possono vedere i risultati dell'algoritmo di stima dell'errore descritto nel paragrafo 3.1.3, dopo l'applicazione dello scan matching sulle scansioni dello stesso file di log:

# SCANSIONI SALTATE	ERRORE ANGOLARE MEDIO	DEVIAZIONE STD	DISTANZA MEDIA (M)
0	0.0077	0.0118	0.0981
1	0.0037	0.0059	0.0824
4	0.0080	0.0072	0.0770
8	0.0088	0.0083	0.0710
12	0.0156	0.0163	0.1295
14	0.0125	0.0113	0.1209
18	0.0210	0.0196	0.7203
20	0.0187	0.0176	0.5538
26	0.0193	0.0230	0.4121
35	0.0127	0.0169	0.8532
40	0.0303	0.0727	0.3959
45	0.0228	0.0236	0.4028
55	0.0200	0.0208	0.4333
65	0.0197	0.0213	0.4907

Tabella 2.2: Risultati scan matching

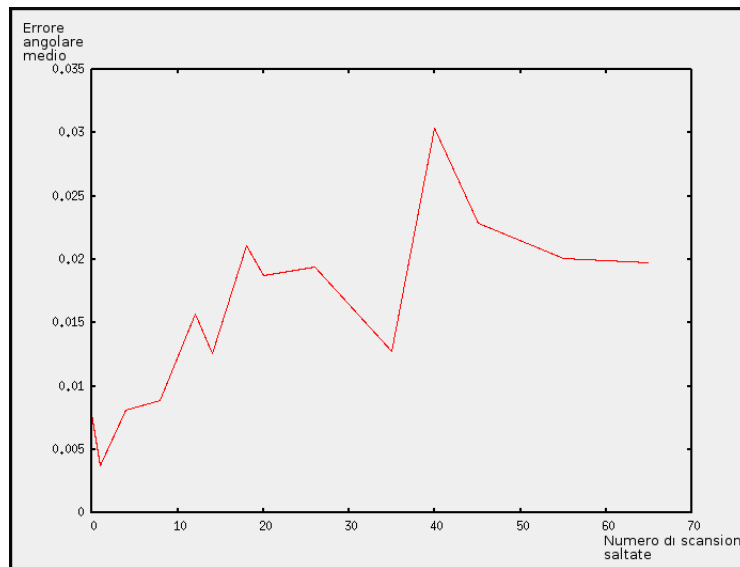


Figura 2.5: Errori medi scan matching

Come si può vedere anche gli errori medi generati dall'uso dello scan matcher hanno un andamento crescente, ma si assestano su valori più bassi di quelli dell'odometria, a dimostrazione del miglioramento nella misura con scansioni laser. In realtà l'algoritmo di scan matching usato è soggetto al problema dello "slittamento" delle scansioni descritto all'inizio di questo paragrafo, e quindi risulta non molto sensibile in un

ambiente omogeneo, con scansioni sempre simili tra di loro e molto profonde (anche 9 metri), quindi l'andamento dipende molto dall'ambiente e dalla soglia usata per il riconoscimento delle features. Anche per questo motivo si può notare una discrepanza nella misura della distanza media percorsa, che risulta minore nel caso dello scan matching in quasi tutti i punti. Infatti con scansioni molto simili lo scan matcher non riesce a stimare con precisione la distanza percorsa, mentre l'odometria, essendo una misura "meccanica", è più precisa in questo caso.

Le tabelle 2.3 e 2.4 mostrano i risultati ottenuti con un altro file di log catturato durante un benchmark, prima i dati raccolti dall'odometria e poi dallo scan matcher:

# SCANSIONI SALTATE	ERRORE ANGOLARE MEDIO	DEVIAZIONE STD	DISTANZA MEDIA (M)
0	0.0082	0.0043	0.0066
5	0.0030	0.0043	0.1457
15	0.0023	0.0041	0.6611
25	0.0042	0.0040	1.1706
35	0.0063	0.0041	1.6765
45	0.0096	0.0054	2.1715
60	0.0075	0.0044	2.4402
70	0.0104	0.0059	2.7873
80	0.0208	0.0255	3.2747
90	0.0181	0.0199	3.7603
100	0.0250	0.0207	4.2566

Tabella 2.3: Risultati odometria partendo da un file di log alternativo

# SCANSIONI SALTATE	ERRORE ANGOLARE MEDIO	DEVIAZIONE STD	DISTANZA MEDIA (M)
0	0.0081	0.0044	0.0077
5	0.0030	0.0045	0.1564
15	0.0022	0.0041	0.6810
25	0.0035	0.0022	1.1945
35	0.0069	0.0039	1.7008
45	0.0093	0.0060	2.2091
60	0.0093	0.0093	2.7951
70	0.0101	0.0064	2.6530
80	0.0154	0.0121	3.7042
90	0.0139	0.0110	3.6759
100	0.0354	0.0611	4.6686

Tabella 2.4: Risultati scan matching partendo da un file di log alternativo

Qui si nota ancora un aumento dell'errore soprattutto in relazione alla distanza percorsa dal robot tra una scansione e l'altra, che è decisamente maggiore rispetto alla serie di scansioni precedente. Con queste particolari scansioni e con la soglia usata per il riconoscimento delle features si può vedere come il miglioramento dato dalla scan matching non sia così marcato come nel caso precedente. In compenso la distanza percorsa ha misure piuttosto simili a parità di scansioni saltate, quindi la conformazione dell'ambiente non ha influenzato la precisione dello scan matcher.

Capitolo 3

Rappresentazioni dell'ambiente

Una scansione acquisita dal robot rappresenta, attraverso i suoi punti, gli ostacoli più vicini alla posizione del laser. E' utile applicare alcuni algoritmi classici per cercare di estrarre informazioni di alto livello dalla nuvola di punti: entrambi i metodi implementati in questo lavoro di tesi si basano sulla segmentazione delle scansioni, e sul concetto di *feature*, cioè un qualsiasi elemento riconoscibile nell'ambiente, riconducibile ad un sottoinsieme di punti di una scansione. Questo aspetto verrà descritto nel prossimo capitolo.

Per quanto una scansione possa essere precisa, il laser scanner utilizzato rileva 360 punti per ogni scansione, che possono risultare insufficienti in alcuni casi. Quando per esempio l'ambiente circostante il robot è "complicato", come un incrocio di corridoi o muri con porte aperte, le occlusioni impediscono di ottenere una rappresentazione completa dell'ambiente. Per questo vengono utilizzati dei modelli costruiti sulla composizione di più scansioni: queste scansioni sono prese ad intervalli regolari, sia di tempo che di spazio, e vengono composte in modo incrementale per costruire delle *mappe locali*. A loro volta queste mappe possono essere composte tra di loro e generare una mappa completa dell'ambiente esplorato dal robot. Queste tecniche saranno descritte nel paragrafo 3.2.

3.1 Segmentazione delle scansioni

Per segmentazione di scansioni si intende la suddivisione dell'insieme di punti di una scansione in alcuni sottoinsiemi, che dipendono dal particolare tipo di features che si vuole estrarre. Un esempio di feature potrebbe essere un muro, che quindi si traduce in una scansione che contiene un insieme di punti allineati, che devono essere riconosciuti e identificati come una retta. I due algoritmi presentati in seguito rappresentano due

tecniche per l'estrazione di features rettilinee. In questo lavoro è stato utilizzato prima il metodo Split&Merge, poi rimpiazzato con la trasformata per aumentare le prestazioni, e nello stesso tempo ridurre la complessità algoritmica. In ogni caso la maggior difficoltà nella segmentazione di un insieme di punti è dovuta al rumore introdotto dal sensore laser nella scansione, che rende imprecisi i contorni degli oggetti.

3.1.1 Split & Merge

Le tecniche basate sul concetto *Split & Merge* (cioè, suddivisione ed aggregazione) eseguono la segmentazione in due fasi:

- fase “Split”: è un processo top-down che suddivide la scansione in regioni elementari.
- fase “Merge”: è un processo bottom-up che permette di raggruppare le regioni elementari in regioni più complesse.

In realtà in questi lavoro è stata utilizzata solo la fase di splitting, necessaria per individuare le features. Questa fase è composta da una procedura ricorsiva che scandisce tutto l'Insieme di punti, la cui condizione di stop è verificata quando l'insieme in esame contiene solo due punti, oppure quando la distanza massima tra il punto che suddivide l'immagine e la retta di regressione lineare è minore di una certa soglia. Da questa soglia dipendono largamente le prestazioni dell'algoritmo.

La fase iniziale è rappresentata in figura 3.1: partendo dai punti di una scansione si traccia la retta di regressione lineare, descritta da una equazione del tipo $y=mx +q$, dove:

- $m = \frac{NS_{xy} - (S_x S_y)}{NS_{xx} - (S_x S_x)}$
- $q = \frac{S_y}{N} - (\frac{S_x}{N}m)$

I coefficienti sono calcolati su tutto l'insieme dei punti della scansione, come segue:

- $S_{xy} = \sum_i x_i y_i$
- $S_{xx} = \sum_i x_i x_i$
- $S_x = \sum_i x_i$
- $S_y = \sum_i y_i$

Una volta ottenuta la retta si trova qual'è il punto con distanza maggiore. Questo sarà il punto che delimita i due successivi insiemi di punti.

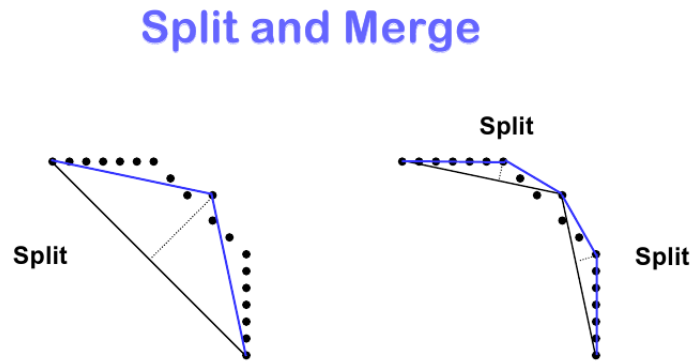


Figura 3.1: Spilt&Merge: fase iniziale

Partendo da questo punto si splitta l'insieme di punti in due sottoinsiemi e poi si itera l'algoritmo su ognuno degli insiemi trovati. In figura 3.2 si può vedere una situazione in cui non è più necessario iterare l'algoritmo sul rimanente insieme di punti, anche in relazione alla soglia usata. I segmenti di colore blu rappresentano i segmenti riconosciuti e possono essere associati a features dell'ambiente.

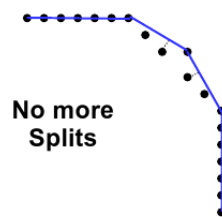


Figura 3.2: Split&Merge: condizione di stop

L'algoritmo completo è descritto nell'algoritmo 1.

I risultati saranno mostrati nel capitolo 5.1.

Algoritmo 1 Algoritmo di Splitting

Require: V_p insieme dei punti della scansione

Require: V_s insieme dei segmenti trovati

```
1: if SizeOf( $V_p$ ) < 2 then
2:   Crea e inizializza un nuovo segmento, da aggiungere a  $V_s$ 
3:   return
4: end if
5: Calcola retta di regressione sull'insieme di punti
6: for  $i := 0, \dots, i < \text{SizeOf}(V_p)$  do
7:   trova il punto  $p$  con distanza massima dalla retta di regressione
8: end for
9:  $th =$  soglia per la distanza tra punti e retta
10: if  $distMax < th$  then
11:   Crea e inizializza un nuovo segmento, da aggiungere a  $V_s$ 
12:   return
13: end if
14: Reitera sull'insieme di punti da 0 al punto  $p$ 
15: Reitera sull'insieme di punti dal punto  $p$  all'ultimo punto
```

3.1.2 Trasformata di Hough

Per individuare le features con maggiore precisione si è deciso di utilizzare principalmente la trasformata di Hough. La trasformata di Hough (HT) viene spesso usata nell'ambito della visione artificiale, come metodo per riconoscere linee e curve in una immagine. Sono stati implementati anche algoritmi di scan matching che lavorano direttamente nel dominio della HT, come presentato in [7].

La trasformata permette di mappare un input in coordinate cartesiane in una funzione di output nel dominio dei parametri ρ e θ : questi sono rispettivamente la distanza di una retta dal centro del piano e la direzione del vettore normale alla retta. Nell'implementazione usata θ viene limitato tra $-\pi/2$ e $\pi/2$, per evitare di memorizzare due volte informazioni periodiche riguardanti la stessa retta. Infatti i punti (θ, ρ) e $(\theta + \pi, \rho)$ rappresentano la stessa retta nello spazio cartesiano.

L'utilizzo della trasformata risulta più immediato se si rappresentano le rette con una parametrizzazione formata da parametri polari, che coincidono con i parametri delle rette utilizzati nel dominio della HT. Quindi è stata scelta la seguente forma:

$$x \cos \theta + y \sin \theta = \rho$$

Rispetto alla parametrizzazione utilizzata nello Split&Merge risulta più adatta al calcolo computazionale, essendo in grado di rappresentare anche rette verticali (con coefficiente angolare infinito).

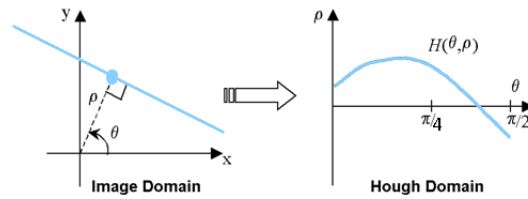


Figura 3.3: Trasformata di Hough

Una volta che sono stati caricati i punti di una scansione, per ogni punto si calcolano i parametri di tutte le rette che potrebbero passare per quel punto e si incrementano le celle di uno spazio n-dimensionale (con n numero dei parametri) che corrispondono alle varie rette. Si ottiene così una funzione di accumulazione definita nello spazio dei parametri.

Alla fine saranno i massimi di questa funzione, ovvero i punti nello spazio dei parametri che hanno il maggior valore di accumulazione, a rappresentare le rette che hanno probabilità elevata di essere presenti nella scansione, come se si trattasse di una ipotesi avvalorata dal maggior numero di conferme sperimentali.

L'idea quindi per estrarre rette da una scansione è formalizzata come segue:

- Rappresentare lo spazio (θ, ρ) attraverso una matrice di accumulazione A ;
- Inizializzare tutti gli elementi di $A(\theta, \rho)$ a zero;
- Per ciascun punto della scansione calcolare $\rho = x \cos \theta + y \sin \theta$;
- Incrementare $A(\theta, \rho)$;
- Ricercare in $A(\theta, \rho)$ i punti di massimo (un punto di massimo in tale matrice corrisponde ad una retta nello spazio immagine originale). In figura 3.4 il punto P è un punto di massimo della funzione di accumulazione e le sue coordinate rappresentano i parametri di una retta nel piano cartesiano.

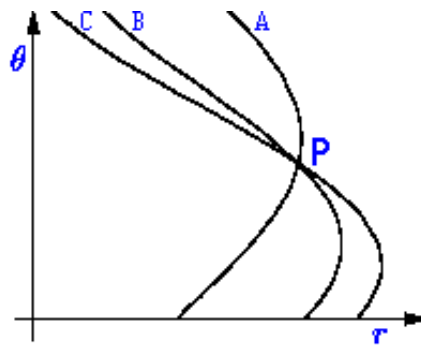


Figura 3.4: HT: rappresentazione nello spazio dei parametri

3.1.3 Utilizzo della Trasformata di Hough

Per realizzare la funzione di accumulazione descritta è stata utilizzata una mappa in due dimensioni, dove sulle ascisse è rappresentata θ , e sulle ordinate ρ , come descritto nel capitolo 3.3. Ogni cella contiene un contatore (un numero intero) che viene incrementato ogni volta che la trasformazione di due coordinate cartesiane produce in output i valori di ρ e θ della cella corrente. Si fissa poi una soglia per i contatori in modo da settare il numero minimo di punti *collineari* nel piano cartesiano.

L'algoritmo è mostrato in 2, e si basa sull'estrazione di features riconosciute con la HT, dopo aver letto le scansioni laser. Le coppie di scansioni vengono associate tra di loro tramite l'odometria, e per ogni coppia vengono calcolati gli errori medi confrontando i θ delle rette estratte. Una seconda mappa viene riempita con i valori della trasformata dopo aver eseguito lo scan matching. Anche qui viene calcolato l'errore quadratico medio per le orientazioni, ma solo per i segmenti che hanno un $\Delta\rho$ minore di una soglia, per limitare i casi che contribuiscono all'errore.

Alla fine dell'algoritmo viene fatta una media aritmetica degli errori per odometria e scan matching, un confronto tra questi valori è presentato nel capitolo 2.5.

Algoritmo 2 Stima dell'errore con Trasformata di Hough

Require: *scanPrev* una scansione letta in un punto del log

Require: *scanCurr* la scansione corrente, letta successivamente

```
1: while lettura scansione do
2:   calcola odometria tra le scansioni
3:   costruisce le mappe 2D con la Traformata di Hough
4:   riempie i vettori segRef e segCurr che contengono i segmenti riconosciuti per
   le due scansioni
5:   for  $i := 0, \dots, i < \text{SizeOf}(\text{segRef})$  do
6:     for  $j := 0, \dots, j < \text{SizeOf}(\text{segCurr})$  do
7:       confronta il segmento  $i$  con il segmento  $j$ 
8:       calcola l'errore quadratico medio tra le orientazioni dei segmenti
9:       restituisce la media aritmetica degli errori per l'odometria
10:    end for
11:  end for
12:  applica lo scan matching tra le scansioni scanRef e scanCurr
13:  costruisce le mappe 2D con la Traformata di Hough
14:  riempie i vettori segRef e segCurr che contengono i segmenti riconosciuti per
   le due scansioni
15:  for  $i := 0, \dots, i < \text{SizeOf}(\text{segRef})$  do
16:    for  $j := 0, \dots, j < \text{SizeOf}(\text{segCurr})$  do
17:      confronta il segmento  $i$  con il segmento  $j$ 
18:      calcola l'errore quadratico medio tra le orientazioni dei segmenti
19:      restituisce la media aritmetica degli errori per lo scan matching
20:    end for
21:  end for
22:  calcola la media degli errori trovati, e la deviazione standard
23:  scrive i risultati su un file di log
24: end while
```

3.2 Rappresentazioni tramite mappe

Una singola scansione non può rappresentare in modo completo lo spazio attorno al robot. Quindi è conveniente costruire una mappa dell'ambiente: dapprima una mappa locale, composta dalle ultime scansioni ricavate dal robot, e poi una mappa globale dove le mappe locali sono composte insieme, come in figura 3.5. Le mappe locali vengono anche chiamate *patch* per indicare il loro ruolo di “copertura” dell'ambiente in modo capillare.

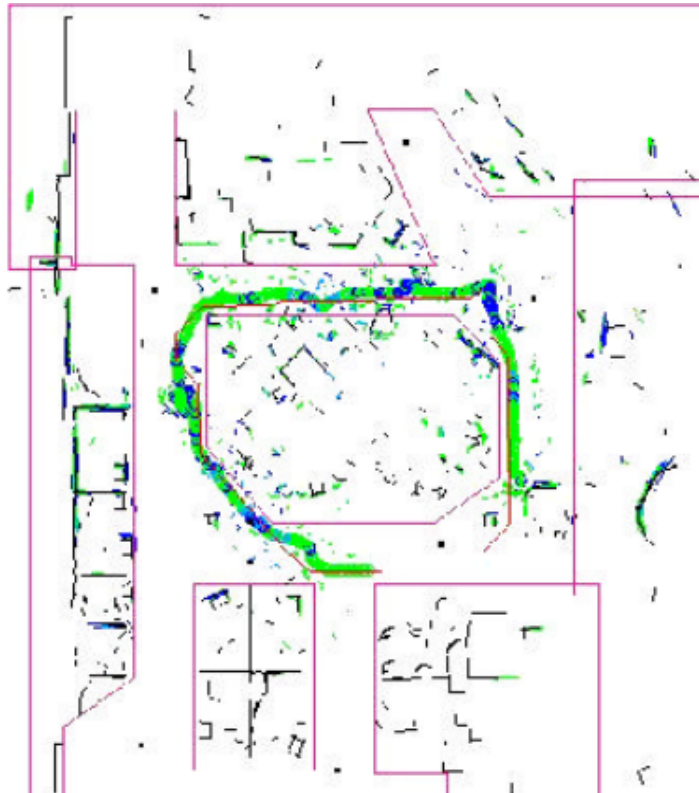


Figura 3.5: Esempio di mappa globale

Gli usi delle mappe sono molteplici: dalla localizzazione del robot alla pianificazione del suo movimento, e al controllo della direzione di movimento. Esistono diversi tipi di mappe, quelle utilizzate in questo lavoro sono: le mappe a griglia, descritte nel capitolo 3.3, e le mappe composte da features, descritte nel capitolo 3.4.

Nella maggioranza dei casi l'ambiente circostante non è conosciuto dal robot, quindi è necessario effettuare una creazione *incrementale*, dove le mappe vengono costruite componendo le scansioni prese dal robot proprio durante il suo movimento. In figura 3.6 è mostrato uno schema della costruzione: ad ogni posa del robot corrisponde una osservazione (e quindi una scansione), e conoscendo due pose successive del robot si trova la relazione tra le posizioni, che va a definire la posizione nella mappa.

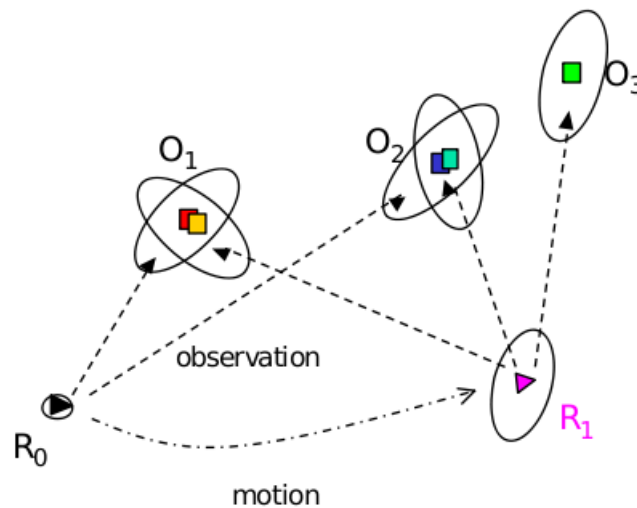


Figura 3.6: Costruzione incrementale di mappe

Le varie pose assunte dal robot devono essere relazionate fra di loro al fine di costruire una *rete di vincoli* che permetta il posizionamento preciso delle mappe e la minimizzazione delle errore accumulato durante il movimento del robot. Il procedimento di stima dei vincoli fra le pose viene presentato in [8], in cui vengono descritte le trasformazioni necessarie per il passaggio da una posa ad un'altra. Assumendo di avere due pose P_a e P_b , ed una differenza di posizioni individuata da un'altra posa D . Se il robot assume la posa iniziale in P_b , si muove e assume la posa D , allora la posa P_a sarà ottenuta dalla composizione delle due pose precedenti calcolata come:

$$x_a = x_b + x_d \cos \theta_b - y_d \sin \theta_b$$

$$y_a = y_b + y_d \sin \theta_b + x_d \cos \theta_b$$

$$\theta_a = \theta_b + \theta_d$$

E' utile anche definire l'operazione inversa, la decomposizione di pose, che restituisce una posa relativa tra due pose iniziali. Le coordinate della posa risultante sono date da:

$$x = (x_a - x_b) \cos \theta_b + (y_a - y_b) \sin \theta_b$$

$$y = -(x_a - x_b) \sin \theta_b + (y_a - y_b) \cos \theta_b$$

$$\theta = \theta_a - \theta_b$$

Queste operazioni vanno a definire i vincoli tra le pose che andranno a comporre la rete: quindi i nodi della rete sono le pose (e le mappe locali ad esse associate) mentre

i vincoli sono rappresentati dalle relazioni geometriche tra le pose. In questo lavoro verrà utilizzato un grafo per modellizzare la rete, come esposto nel capitolo 3.3.

Una volta completata la creazione della mappa sorge il problema di riconoscere gli elementi significativi (dopo averli estratti), e confrontarli con alcune scansioni laser per effettuare una sorta di matching. Un approccio a questo procedimento verrà descritto nel capitolo 4.

3.3 Mappe a griglia e collezioni di mappe

La soluzione più semplice al problema della memorizzazione di un insieme di scansioni consiste nel creare una mappa vista come una matrice di celle in due dimensioni. In questo lavoro è stata utilizzata una implementazione della classe Map2D per creare le patch: questa matrice è in realtà costruita con un vettore in due dimensioni e ogni cella può essere indirizzata con gli indici globali (contatori per il numero di cella) oppure tramite le coordinate di un punto. Ogni mappa ha una posa che definisce il centro, è ha una dimensione variabile. Per il settaggio della precisione della mappa sono stati predisposti degli *step*, che definiscono quante celle sono contenute nella dimensione attuale della mappa. Un esempio di mappa è riportato in figura 3.7.

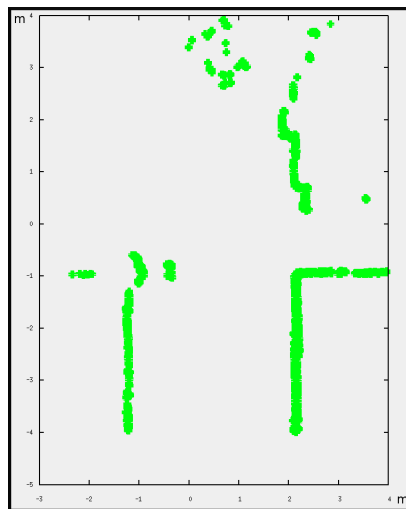


Figura 3.7: Esempio di patch

Ad ogni cella della matrice è associato un numero compreso tra 0 e 1 che rappresenta la probabilità di occupazione della cella. Quindi per ogni punto inserito nella mappa viene calcolato il valore di copertura da aggiungere alle celle adiacenti: la colorazione della mappa è assimilabile ad una PDF Gaussiana con centro nel punto inserito. Le celle adiacenti saranno influenzate dall'inserimento secondo la:

$$Map(x, y) = Map(x, y) + \exp\left(\frac{-d^2}{2}\right)$$

dove d è la distanza tra la cella di coordinate (x,y) e la cella contenente il punto inserito. Le coordinate (x,y) variano nell'ottocinato del punto inserito, e la dimensione di questa variazione è settata da una soglia, per limitare l'influenza dell'inserimento di un punto sulle celle adiacenti. La dimensione di default della mappa è stata fissata a 8x8 metri, misura che garantisce il contenimento della maggior parte delle scansioni nella mappa. Lo svantaggio di questa rappresentazione si trova nella occupazione di memoria, dovuto alla inevitabile memorizzazione nella mappa di aree di scarso interesse. Le mappe basate su features invece risultano più compatte in quanto sono memorizzati solo gli elementi di maggior interesse, e la parametrizzazione usata per rappresentare questi elementi consente di occupare meno memoria.

Una volta che una mappa è stata "colorata" questa rappresenta solo una porzione locale dell'ambiente. Per avere la mappa completa è necessario unire e relazionare le mappe in una struttura più complessa rappresentata da un grafo non orientato. La classe sviluppata si interfaccia con altre classi per la gestione delle mappe e per il riconoscimento delle features rettilinee. Il diagramma UML per l'implementazione del grafo è mostrato in figura 3.8: si possono notare i metodi per l'inserimento di un punto nel grafo (con la creazione delle mappe necessarie), e i metodi per la restituzione del valore contenuto in una cella. Sono stati implementati anche i metodi necessari alla stampa di una mappa o del grafo completo, e quello per l'estrazione dell'orientazione principale dei segmenti riconosciuti nella dalla mappa.

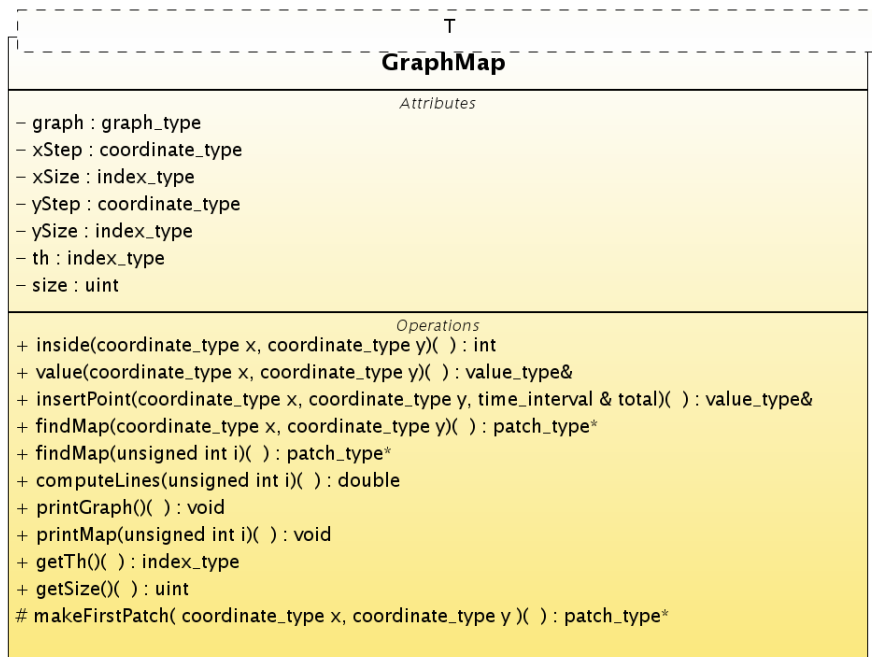


Figura 3.8: UML per la classe GraphMap

La struttura del grafo si basa come detto sulle mappe 2D, e include anche la classe *HoughLine* usata per l'estrazione di segmenti dalla mappa tramite trasformata di Hough. La struttura completa dell'applicazione è mostrata in figura 3.9, e comprende anche un programma di test, che verrà presentato insieme ai risultati ottenuti nel capitolo 5.

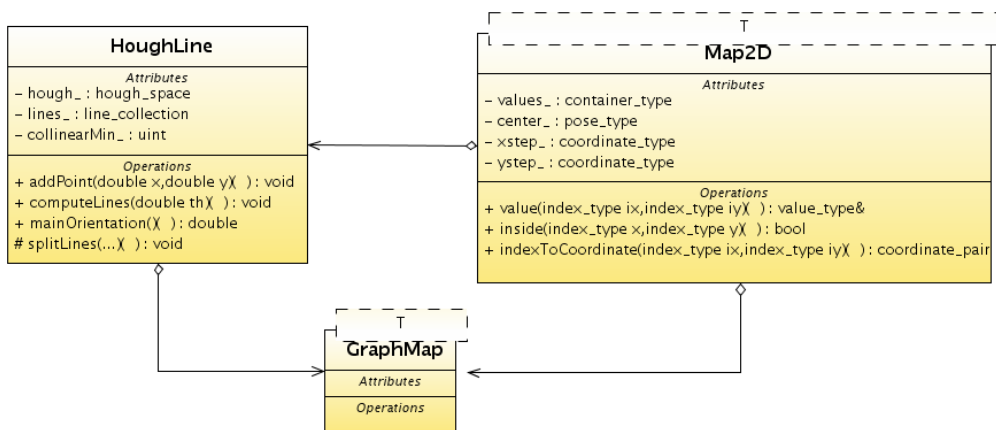


Figura 3.9: Diagramma UML completo

L' funzionalità più interessante della struttura è l'inserimento di un punto nel grafo: le scansioni sono lette dal file di log e i punti che le compongono sono aggiunti "in

tempo reale” nel grafo. All’atto del primo inserimento il grafo è vuoto e deve essere creata una nuova patch; per i punti successivi si procede come segue:

- si scansiona il grafo alla ricerca della patch più vicina al punto P, confrontando le distanze tra il punto e i centri di ogni mappa presente nel grafo;
- si verifica se P è contenuto nella patch più vicina appena trovata; nel caso sia all’interno della mappa locale la cella corrispondente viene “colorata”;
- se P è fuori dal grafo si deve creare una nuova patch, calcolando la posizione del nuovo centro:

$$X_c = \text{round}\left(\frac{P_x - \text{center}_x}{xSize \ xStep}\right) \ xSize \ xStep$$

$$Y_c = \text{round}\left(\frac{P_y - \text{center}_y}{xSize \ xStep}\right) \ xSize \ xStep$$

dove $(\text{center}_x, \text{center}_y)$ sono le coordinate del centro della mappa più vicina, e $xSize \ xStep$ rappresentano la dimensione orizzontale della patch. La posa $(X_c, Y_c, 0.0)$ viene composta con il centro della mappa più vicina e il risultato viene settato come centro della nuova mappa. In questo modo si crea un insieme di patch che “ricopre” perfettamente la zona interessata dalle scansioni, come si può vedere dalla figura 3.10. Questa tecnica consente anche di risolvere il problema dovuto ad alcune scansioni che potrebbero non essere completamente contenute in un mappa, e quindi occuparne parzialmente un'altra. Infatti i punti al di fuori della mappa corrente andranno semplicemente a creare un'altra mappa che sarà vincolata dalla relazione tra i centri delle due mappe [9].



Figura 3.10: Collezione di patch: i nodi del grafo costituiscono una mappa globale

In figura 3.10 è mostrato un esempio di mappa globale generata a partire da un file di log catturato durante un benchmark: sono state lette solo le prime 1000 scansioni che vanno a comporre la prima zona dell'ambiente. La colorazione delle patch non è visibile (tutti i punti appaiono verdi), ma garantirà una maggiore precisione quando verrà affrontato il problema del data association, e quindi sarà calcolata la correlazione tra una patch e una scansione catturata in un qualsiasi punto dell'ambiente.

Uno dei maggiori problemi nel *mapping* di vasti ambienti è dato dal lungo movimento necessario al robot per percorrere tutto lo spazio: l'accumulo dell'errore odometrico impedisce di usare i valori dell'odometria per fissare i vincoli tra le pose. E' quindi necessario utilizzare un algoritmo di loop closure per recuperare i corretti valori geometrici, e individuare quando il robot completa un *loop*, cioè visita una regione della mappa già esplorata in precedenza. Qui infatti la topologia dell'ambiente deve essere riconosciuta dal robot, per evitare di creare successivamente delle mappe disallineate [9].

3.4 Mappe a features

Le mappe a features sono costituite da collezioni di elementi significativi estratti dall'ambiente. Offrono alcuni vantaggi rispetto alle mappe a griglia come il minore spazio

occupato in memoria (generalmente) e la maggiore predisposizione a tutti quegli algoritmi di confronto tra scansioni e mappe o tra mappe e mappe che associano le features di una con quelle dell'altra. Ovviamente l'estrazione di queste features necessita di un algoritmo particolare, ed a questo scopo è stata utilizzata la classe *houghLine*: tramite la trasformata di Hough si estraggono i segmenti che vengono memorizzati in una lista.

L'algoritmo 3 mostra i principali punti della fase di estrazione: in ogni cella della mappa 2D è memorizzata una lista di punti, se la lista contiene più di *CollinearMin* punti allora si può ritenere valida per applicare la trasformata. La retta viene riconosciuta e identificata dai parametri (θ, ρ) , e successivamente si passa alla fase di *splitting* della retta.

Questa fase è necessaria per ottenere una maggiore segmentazione della mappa e facilitare così il compito del riconoscimento dell'orientazione principale. Per *splitting* si intende la suddivisione (ricorsiva) di ogni retta in modo da essere più fedele ai punti memorizzati nella mappa: se, per esempio, non ci sono punti della mappa in prossimità della metà della retta, allora è opportuno che la retta venga suddivisa in due segmenti che rispecchiano la distribuzione dei punti nella mappa. Per lo *splitting* dei segmenti è importante conoscerne gli estremi, che vengono calcolati tenendo conto della distanza esistente tra i punti della mappa: se due punti sono ad una distanza maggiore di una certa soglia allora è più probabile che questi punti siano i due estremi di due segmenti diversi. Il calcolo degli estremi viene fatto nel seguente modo:

$$D = \cos(\theta + \pi/2) x + \sin(\theta + \pi/2) y$$

dove θ è il parametro angolare del segmento estratto secondo la trasformata di Hough, mentre x e y sono le coordinate dei punti del segmento. Tutti i punti vengono scansionati per trovare il punto minimo e massimo che rappresentano i due estremi. L'algoritmo 3 evidenzia anche un procedimento di ulteriore linearizzazione aggiunto in seguito alla trasformata, costituito da una regressione lineare sui punti trovati, prima di calcolare i parametri del segmento estratto. Questa funzione dovrebbe garantire migliori prestazioni dell'algoritmo.

Tuttavia uno dei maggiori problemi dell'intera procedura sta nel diverso numero di punti tra le scansioni (360 o anche 180 punti) e le mappe, che con la colorazione utilizzata sono molto più ricche di punti. Da questo deriva un numero significativamente diverso di rette riconosciute, che saranno poi di difficile associazione, come descritto nel capitolo 4. Si può cercare di ovviare a questo impostando il miglior valore possibile per alcune costanti, come il minimo numero di punti collineari, o la minima distanza tra 2 punti consecutivi, e soprattutto cercando valori diversi per le scansioni rispetto a

Algoritmo 3 Estrazione di rette da una patch 2D

Require: *hough* una matrice di punti ottenuta con la trasformata di Hough

Require: *points* collezione di punti allineati

Require: *lines* collezione dei segmenti estratti

Require: *th* soglia per la distanza tra due punti consecutivi

```
1: for ith := 0, ... , ith < hough.xSize() do
2:   for irho := 0, ... , irho < hough.ySize() do
3:     if numero di punti contenuti nella cella > collinearMin then
4:       calcola regressione lineare sui punti allineati
5:       calcola paramtri del segmento estratto
6:       SPLITTING DEL SEGMENTO
7:       if SizeOf(points) < 2 then
8:         ci sono solo due punti allineati
9:         return
10:      end if
11:     for i := 0, ... , i < SizeOf(points) do
12:       calcola estremi per il segmento corrente
13:     end for
14:     for i := 0, ... , i < SizeOf(points) do
15:       dist := distanza tra due punti consecutivi
16:       if dist > th then
17:         il segmento viene suddiviso in due segmeti i e j
18:         SPLITTING DEL SEGMENTO i
19:         SPLITTING DEL SEGMENTO j
20:       end if
21:     end for
22:     if segmento i non è stato suddiviso ulteriormente then
23:       inserimento del segmento i in lines
24:     end if
25:     if segmento j non è stato suddiviso ulteriormente then
26:       inserimento del segmento j in lines
27:     end if
28:   end if
29: end for
30: end for
31: return
```

quelli utilizzati per le mappe.

Capitolo 4

Associazione di scansioni e mappe

In questo capitolo viene descritto il problema derivante dall'associazione di caratteristiche di una scansione con le corrispondenti caratteristiche nella mappa [10] cioè, dati un set di misure sensoriali ed una mappa locale dell'ambiente riuscire ad estrarre gli elementi principali di entrambe e creare delle associazioni tra quelli che presentano relazioni topologiche. L'estrazione di features è quindi il primo passo da compiere per l'associazione, la quale può essere rappresentata da una apposita struttura, come un albero. Dati:

- n features estratte dalla mappa: $F = \{f_1 \dots f_n\}$;
- m features estratte dalla scansione: $E = \{e_1 \dots e_n\}$;

l'obiettivo è ottenere un ipotesi che associ ogni features della scansione e_i con almeno una feature della mappa f_i . Una rappresentazione grafica è mostrata in figura 4.1. Le ipotesi ottenibili sono $(n + 1)^m$, ma in realtà possono essere ridotte siccome vi sono degli interi rami dell'albero che possono non essere esplorati perché il risultato della loro esplorazione non può migliorare l'ipotesi attuale. Le tecniche utilizzate sono diverse, ma le due principali comprendono la ricerca della features più vicina, e l'utilizzo della *joint compatibility*, concretizzata in un algoritmo di tipo *joint compatibility branch and bound (JCBB)* [11], che lavorando su alberi trova la miglior ipotesi che mantiene la compatibilità.

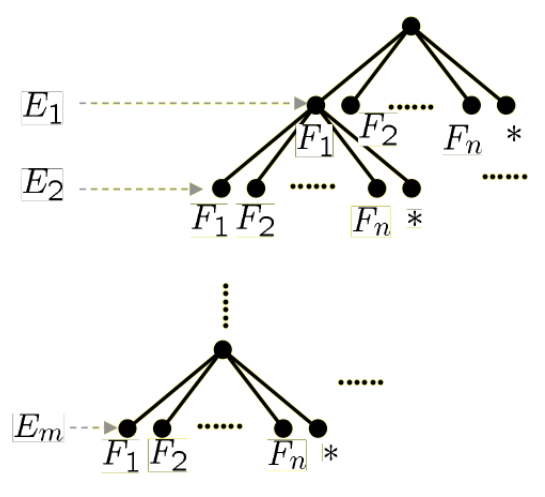


Figura 4.1: Albero di associazioni

In generale ci si attiene ad uno schema per il confronto e l'associazione: è necessaria una prima fase di stima, in cui la scansione e la mappa devono essere orientate nello stesso modo (con un certo margine di errore), e poi un affinamento dell'orientazione e della posizione reciproca. La prima fase può essere implementata come descritto nel paragrafo 4.2, mentre la seconda fase richiede un calcolo della posizione relativa tra mappa e scansione. E' possibile far combaciare i centri, oppure si può tentare di calcolare i baricentri e farli sovrapporre. Questa ultima tecnica non ha dato grandi risultati, considerando la diversa conformazione visibile tra una mappa e una scansione presa in un qualsiasi punto dell'ambiente.

4.1 Metodi di associazione

Una volta che la mappa e la scansione sono posizionate in modo "compatibile" si procede con alcuni metodi di stima locale della sovrapposizione: il metodo più semplice, che è stato implementato per primo, consiste nel calcolo della correlazione tra gli insiemi di punti. Con la mappa e la scansione sovrapposte si calcola la correlazione per diverse posizioni della scansione, in modo da ricercare la posizione di massima correlazione. Partendo dalla posizione iniziale la scansione viene tralata nell'ottocinato, con alcune costanti numeriche che possono intervenire sulla precisione del movimento e sulla dimensione dello spostamento. Per ogni nuova posizione viene calcolata la correlazione, prendendo come valore per la patch il "peso" della cella in corrispondenza dei punti della scansione (per la scansione viene preso come valore 1). Lo spostamento influisce molto sulla velocità e sulla precisione dell'algoritmo, per questo la stima

della posizione iniziale è molto importante e pregiudica i risultati dell'algoritmo, che saranno mostrati nel paragrafo 5.3. Tra i valori di correlazione ottenuti si considera il massimo, che viene confrontato con una soglia; il valore di questa soglia è di difficile interpretazione, data la scarsa precisione dell'algoritmo, e la difficoltà nel trovare la posizione di massima correlazione.

Il secondo metodo consiste in una vera associazione di features che utilizza una appropriata struttura dati per la memorizzazione delle associazioni. In questo lavoro l'associazione tra scansioni e mappe è stata effettuata riuscendo ad "accoppiare" i segmenti estratti da una scansione con quelli estratti dalla mappa. Il fine ultimo di questa associazione consiste nel riuscire a riconoscere una zona dell'ambiente in cui il robot è già stato, cioè risolvere il problema del *loop closing*; questo implica capire quando il robot ha compiuto un "giro" completo in un vasto ambiente, e riconoscere la situazione per evitare disallineamenti nella costruzione incrementale della mappa. La chiusura del loop avviene mediante il confronto tra una precisa mappa e una scansione contenuta nella mappa ma acquisita in un istante successivo, supponendo che l'ambiente abbia una conformazione circolare e che il robot mobile sia in grado di percorrere tutto lo spazio immagazzinando tutte le informazioni percepite. Un'altra importante ipotesi in questo caso riguarda la *staticità* dell'ambiente, che tipicamente risulta soddisfatta, siccome le caratteristiche principali di un luogo rimangono immutate nel tempo.

L'algoritmo utilizzato è descritto nell'algoritmo 4 ed è una implementazione del *JCBB*. Si ricerca nell'albero delle associazioni quella migliore che mantenga la compatibilità tra le features. Prima di tutto è necessario implementare una funzione di confronto tra le rette, che può essere basata per esempio sulla distanza tra il punto che in ciascuna retta è più vicino al centro dell'ambiente. Questo è anche il punto di intersezione con la normale passante per l'origine. Se due rette sono espresse rispetto allo stesso sistema di riferimento, la funzione restituisce una piccola distanza se le rette hanno un orientamento simile e sono tra loro vicine. Per effettuare il bound nell'algoritmo è importante verificare, ad ogni nuova associazione creata, se questa mantiene la compatibilità congiunta, e se la distanza complessiva dei segmenti associati è inferiore ad una certa soglia. È indispensabile anche che non ci siano segmenti della scansione associati con più di un segmento della mappa, e viceversa. L'algoritmo prende in ingresso due collezioni di features, estratte dalla mappa e dalla scansione, e necessita di due associazioni: una che viene aggiornata durante l'elaborazione e una che mantiene in memoria la miglior associazione trovata fino ad un certo istante.

Nelle righe 18-20 viene controllato il numero di associazioni potenziali che è possibile fare con le rimanenti features, e se questo è maggiore del miglior numero di associazioni viene rilanciato l'algoritmo sui dati rimanenti.

Algoritmo 4 Joint Compatibility Branch and Bound

Require: $sLines$ collezione di segmenti estratti dalla scansione

Require: $mLines$ collezione di segmenti estratti dalla mappa

Require: $curr$ insieme delle associazioni corrente

Require: $best$ insieme delle migliori associazioni

```
1:  $s_i$  = segmento corrente in  $sLines$ 
2: if  $s_i == sLines.end()$  then
3:   sono stati elaborati tutti i segmenti della scansione
4:   if  $curr.size() > best.size()$  then
5:      $best := curr$ 
6:   else if  $curr.size() = best.size()$  and  $assocDistance(curr) < assocDistance(best)$ 
7:     then
8:        $best := curr$ 
9:     return
10:  end if
11: else
12:    $m_j$  = segmento corrente in  $mLines$ 
13:   for  $j := 0, \dots, j < mLines.size()$  do
14:     if  $lineDistance(s_i, m_j) < th$  and  $compatibility(curr)$  then
15:       inserimento nuova associazione  $(s_i, m_j)$  in  $curr$ 
16:       reitera JCBB sui segmenti da  $s_i$  a  $sLines.end()$ 
17:     end if
18:   end for
19:   if  $curr.size() + distance(s_i, sLines.end()) > best.size()$  then
20:     inserimento nuova associazione nulla  $(s_i, 0)$  in  $curr$ 
21:     reitera JCBB sui segmenti da  $s_i$  a  $sLines.end()$ 
22:   end if
23: end if
24: return
```

Questo metodo presenta notevoli difficoltà di controllo e gestione, considerando anche l'algoritmo ricorsivo di visita dell'albero, e non ha permesso di ottenere risultati significativi.

.....
correlazione tra mappe..... utilizzata una mappa per il loop closing e non una scansione

4.2 Stima dell'orientamento principale

La stima dell'orientamento preferenziale di una mappa (o di una scansione) è ottenibile confrontando tutti gli orientamenti dei segmenti che la compongo, e facendo una media pesata sul numero di punti contenuti in ogni segmento. L'algoritmo generale è mostrato nell'algoritmo 5. Alla fine dell'algoritmo si rendono necessarie delle correzioni sull'orientamento trovato, in modo che tutte le scansioni con la stessa orientazione vengano riconosciute come tali: infatti una scansione "verticale" può avere come orientamento principale sia $\pi/2$ che $-\pi/2$, è quindi necessario fare in modo che tutti i valori rispettino la stessa convenzione.

Una volta trovato l'orientamento principale di scansione e mappa si calcola la nuova orientazione per la scansione, in modo che i due orientamenti coincidano. Siano:

- θ_M : orientamento della mappa rispetto al frame globale;
- θ_{LM} : orientamento principale dei segmenti rispetto al centro mappa;
- θ_{LS} : orientamento principale dei segmenti rispetto al centro scansione;
- θ_S : nuovo orientamento della scansione rispetto al frame globale;

il nuovo orientamento della scansione si trova imponendo che l'orientamento dei segmenti rispetto al frame globale sia lo stesso:

$$\theta_S = \theta_M + (\theta_{LM} - \theta_{LS})$$

L'orientazione θ_S e le coordinate del nuovo centro scansione formano una posa, che sarà composta con il vecchio centro per ottenere la nuova posizione.

L'algoritmo indicato si adatta bene alle patch, in cui sono presenti molti punti e quindi vengono riconosciuti molti segmenti (nell'ordine delle migliaia). Nel caso delle scansioni i punti contenuti sono sensibilmente minori, e di conseguenza anche i segmenti riconosciuti vanno dai 2 ai 25-30. Quindi, per le scansioni, l'algoritmo calcola la media semplice delle orientazioni, e da risultati migliori in un maggior numero di casi.

Algoritmo 5 Calcolo orientazione principale di un insieme di segmenti

Require: L un insieme di segmenti

Require: $s.\theta$ orientazione del segmento s

Require: $s.num$ numero di punti del segmento s

```
1: for  $i := 0, \dots, i < L.Size()$  do
2:   for  $j := 0, \dots, j < L.Size()$  do
3:      $\Delta\theta := s_i.\theta - s_j.\theta$ 
4:     if  $\text{abs}(\cos(\Delta\theta)) > 0.99$  then
5:       i due segmenti hanno un orientamento paragonabile
6:       calcola orientazione  $\theta_M = \frac{(s_i.\theta s_i.num + s_j.\theta s_j.num)}{(s_i.num s_j.num)}$ 
7:     end if
8:   end for
9: end for
10: calcolo nuovo orientamento  $\theta_S$ 
11:  $\theta_S = \theta_M + (\theta_{LM} - \theta_{LS})$ 
12: return
```

Capitolo 5

Risultati

5.1 Estrazione di features con Split&Merge

Di seguito è mostrato il risultato dell'estrazione di segmenti partendo da un log catturato in un ambiente con pareti quasi rettilinee. La scansione è sovrapposta con le features individuate:

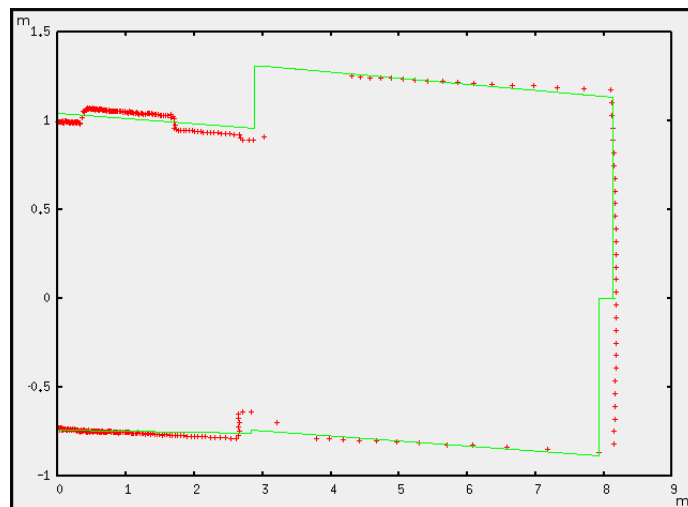


Figura 5.1: '+' punti della scansione, '-' segmenti riconosciuti

Ciò che si può notare è che agli insiemi di punti allineati con cardinalità troppo bassa non viene associato un segmento. Questo comportamento dipende molto dalla soglia utilizzata come distanza massima tra un punto e la retta di regressione lineare, come descritto nel paragrafo 3.1.1. Un aumento di questa soglia comporta un peggior riconoscimento dei gruppi contenenti pochi punti, e un'approssimazione grossolana dei

tratti lontani dalla sorgente della scansione. Inoltre alcuni insiemi di punti appartenenti a segmenti diversi ma molto vicini sono rappresentati da uno stesso segmento.

La convergenza di questo algoritmo non è sempre garantita, quindi per migliorare le prestazioni dell'applicazione nel suo complesso è stato deciso di utilizzare la trasformata di Hough, che da risultati migliori, soprattutto per l'estrazione di segmenti rettilinei.

5.2 Errori sulla stima della traiettoria

Per stimare l'errore sulla traiettoria il robot è stato munito di un dispositivo di tracciamento che segnasse la traiettoria reale del robot mentre avanzava. Successivamente è stato scritto un behavior che istruiva il robot a mantenersi il più possibile al centro dell'ambiente: viene controllata periodicamente la distanza dalle pareti e vengono fatte le opportune correzioni di traiettoria. Muovendo il robot con questo behavior è stata ottenuta una traiettoria visibile. Una volta ottenuta una linea abbastanza precisa è stata campionata ogni 20 cm e le misure sono state memorizzate in un file di log che sarà in input all'algoritmo di calcolo. L'algoritmo legge le scansioni dal file di log ottenuto con il laser scanner, e i punti della traiettoria reale. Componendo le pose assunte dal robot (x , y , orientazione θ) in ogni scansione con quelle della scansione precedente (di riferimento) si costruisce la traiettoria approssimata prima con l'odometria e poi con lo scan matching. Il grafico di questi punti è mostrato in figura 5.2. È stata calcolata la distanza media tra i punti prodotti dallo scan matching e quelli reali, in modo da dimostrare che lo scan matching migliora l'identificazione della posizione del robot nell'ambiente, rispetto alle prestazioni offerte dall'odometria. Per ogni punto della traiettoria reale è stato trovato il punto con distanza minima, tra quelli ottenuti con l'odometria e quelli ottenuti con lo scan matching; queste distanze rappresentano lo scostamento delle traiettorie da quella reale, e vanno a stimare la precisione delle misure sensoriali.

Oltre a questo è stato indispensabile correggere l'orientazione iniziale delle traiettorie per evitare una discrepanza, e quindi normalizzare l'orientazione delle tre misure.

Di seguito sono mostrati i risultati numerici nella tabella 5.1, e il grafico delle traiettorie in figura 5.2.

#	DISTANZA PERCORSA (M)	ERRORE ODOMETRIA (M)	ERRORE SCAN MATCHING (M)
0	0.0000	0.0010	0.0000
1	0.6007	0.0052	0.0055
2	1.2003	0.0149	0.0132
3	1.8000	0.0365	0.0255
4	2.2000	0.0505	0.0389
5	2.6000	0.0620	0.0359
6	3.0000	0.0714	0.0162
7	3.4001	0.0834	0.0263
8	4.0010	0.1142	0.0130
9	4.4018	0.1253	0.0128
10	4.8032	0.1501	0.0248
11	5.2038	0.1495	0.0340
12	5.4040	0.1473	0.1041
13	5.6045	0.1517	0.3016
14	5.8049	0.1521	0.5011
15	6.0050	0.1460	0.7003
16	6.2052	0.1560	0.9002

Tabella 5.1: Risultati della prova sperimentale sulla traiettoria

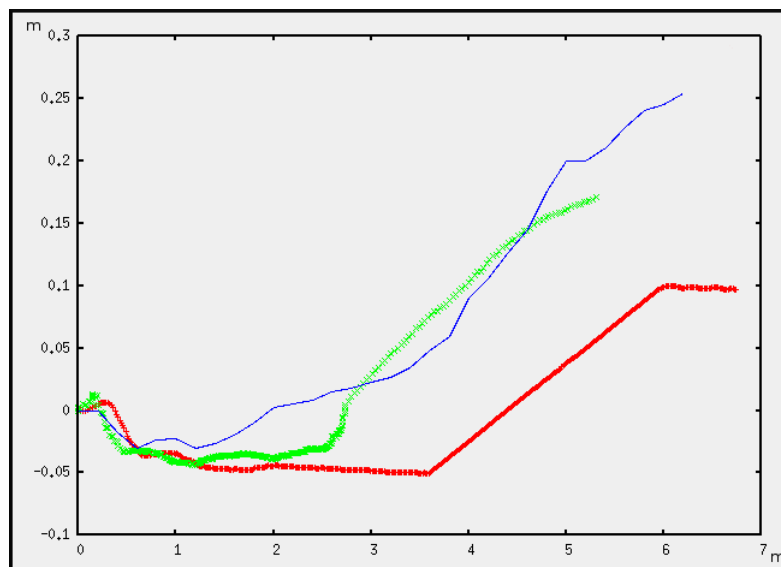


Figura 5.2: '—' traiettoria reale, '+' odometria, 'x' scanmatcher

In totale il robot ha percorso 6 metri e 20 cm, l'errore commesso dallo scan matching è sempre minore di quello dell'odometria tranne negli ultimi punti, l'ultimo metro

circa, dove si nota uno "schiacciamento" della traiettoria dello scan matching che potrebbe essere dovuto ancora alla conformazione dell'ambiente e alla profondità delle scansioni, e soprattutto alla correzione dell'orientamento necessaria per evitare un disallineamento iniziale. Questa correzione infatti è stata fatta considerando l'orientamento dei primi punti della traiettoria reale, e quindi modificando di conseguenza l'orientamento di tutti i punti delle scansioni. Ma questa trasformazione porta ad una "deformazione" della traiettoria ottenuta tramite scan matching verso la fine della misurazione.

Si nota comunque che almeno all'inizio la misura proposta dallo scan matching è più fedele, soprattutto quando la traiettoria prende una direzione precisa (circa a 3 metri), dove lo scan matching è più rapido e preciso nel seguire il cambiamento di direzione.

5.3 Allineamento scansioni mappe

Per testare l'applicazione è stato realizzato un algoritmo, visibile in 6, che simula la costruzione della mappa partendo da un file di log, e successivamente cerca il miglior allineamento possibile tra una patch (o una serie di patch) e una scansione presa dal log. Le patch e la scansione vengono scelte tramite degli argomenti passati per riga di comando all'eseguibile. E' anche prevista la possibilità di eseguire lo scan matching delle scansioni in ingresso, nel caso in cui il file di log utilizzato non contenga già i valori dello scan matching.

I primi risultati dell'algoritmo si possono identificare nella costruzione del grafo, in cui i nodi sono rappresentati dalle patch. In figura 3.10 è mostrato un esempio di mappa. Si notano alcuni punti "dispersi" nelle mappe dalla 25 alla 29, siccome non sono state usate tutte le scansioni del file di log: infatti quei punti appartengono a zone che sarebbero coperte se la mappa fosse completa. In generale i vincoli costruiti tra le pose, e lo scan matching, rendono le dimensioni e le proporzioni abbastanza precise, in modo da poter elaborare la mappa successivamente.

Una volta completata la creazione della mappa vengono eseguite le fasi critiche dell'algoritmo, cioè l'estrazione dei segmenti e il calcolo dell'orientazione principale.

L'estrazione dei segmenti avviene senza particolari problemi quando si prendono in esame delle patch rettilinee, come la numero 4 o la numero 6 in figura 3.10, mentre può presentare alcuni problemi nel caso di patch come la numero 5, in figura 5.3. L'incrocio di più vie rappresenta una situazione più difficile da analizzare, in cui la definizione di orientamento principale va in crisi. Come descritto nel capitolo 3.4 la suddivisione dei

Algoritmo 6 Test dell'applicazione

Require: $m1$ e $m2$ identificativi numerici della prima e ultima patch da analizzare

Require: s scansione laser

Require: $sMax$ numero di scansioni da includere nella mappa

- 1: ciclo di costruzione della mappa
 - 2: **for** $i := 0, \dots, i < sMax$ **do**
 - 3: lettura scansione s_i
 - 4: **for** $j := 0, \dots, j < s_i.Size()$ **do**
 - 5: inserimento del punto p_j nel grafo
 - 6: colorazione della patch attorno al punto p_j
 - 7: **end for**
 - 8: **end for**
 - 9: ciclo di calcolo della correlazione tra patch e scansione s
 - 10: lettura della scansione s
 - 11: estrazione orientazione principale della scansione
 - 12: **for** $i := m1, \dots, i \leq m2$ **do**
 - 13: estrazione orientazione principale della patch
 - 14: correzione orientamento della scansione
 - 15: stima della posizione di massima correlazione
 - 16: calcolo del massimo valore di correlazione
 - 17: **end for**
 - 18: **return**
-

segmenti in tratti più corti permette di approssimare meglio la patch; infatti in figura si notano i segmenti riconosciuti (in verde) che restituiscono una orientazione ben precisa (in questo caso 1.5708 rad, circa $\pi/2$ rad).

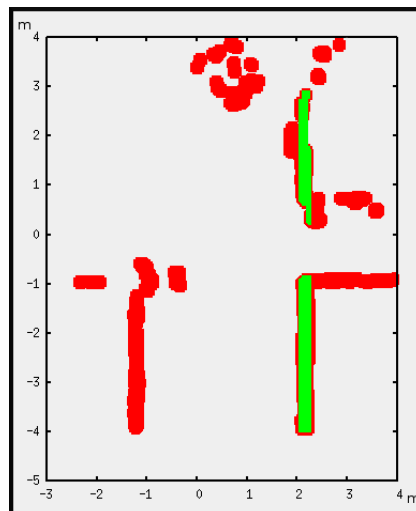


Figura 5.3: Estrazione di segmenti (in verde) da una patch

La figura 5.4 mostra invece i segmenti estratti da una scansione, presa tra quelle

che compongono la patch 5. Come si può vedere i segmenti sono molti meno, ma sono sufficienti per estrarre una corretta orientazione (1.55309 rad).

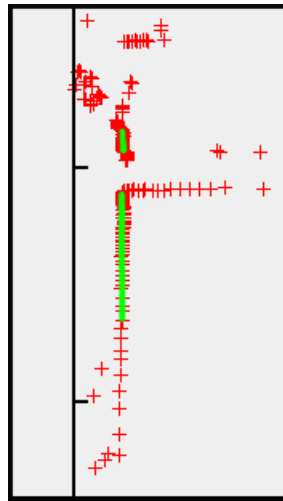


Figura 5.4: Estrazione segmenti da una scansione

Questa orientazione confrontata con quella della patch da una correzione minima, che infatti non influisce sulla posizione reciproca tra scansione e patch. Il successivo calcolo della correlazione avviene tramite sovrapposizione, come in figura 5.5, e da come massimo valore 224.10 che denota quindi una buona corrispondenza tra scansione e mappa, come ci si aspettava.

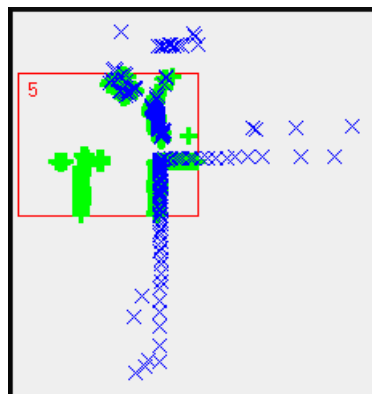


Figura 5.5: Sovrapposizione tra scansione e patch, per il calcolo della correlazione

I problemi principali emergono quando si prende in esame una scansione “lontana” dalla patch considerata, cioè catturata dal robot in un punto distante dalla porzione di territorio coperta dalla patch. In figura è mostrata la posizione iniziale per il calcolo della correlazione: i due orientamenti sono molto simili, essendo l’orientamento calcolato con coordinate locali non risente del lungo movimento del robot, mentre le posizioni

risentono dell'errore accumulato dal robot durante il suo movimento e di conseguenza la scansione e la patch risultano disallineate.

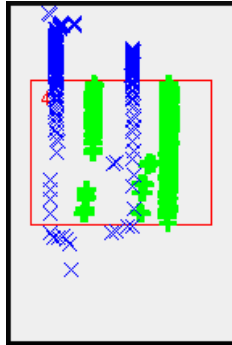


Figura 5.6: Posizione iniziale tra una patch e una scansione presa in un punto a distanza maggiore

In questo caso la correlazione risente dell'errata stima iniziale della posizione e non va oltre il valore di 0.66, nonostante la scansione e la patch siano “compatibili” e quindi sovrapponibili. Ovviamente questo ultimo caso è quello di maggior interesse quando si cerca di chiudere il *loop*, dove si deve confrontare una zona dell'ambiente rappresentata da una patch con una scansione presa dal robot dopo che ha percorso tutto il tragitto per tornare nel punto di partenza.

Capitolo 6

Conclusioni

In questo lavoro di tesi è stato realizzato un metodo in grado di costruire in modo incrementale la mappa di un ambiente partendo dalle scansioni catturate da un robot mobile durante il suo movimento. Durante la realizzazione sono stati affrontati i problemi derivanti dalla successiva elaborazione della mappa ottenuta: l'associazione di features della scansione con quelle della mappa per il riconoscimento della posizione attuale, e la capacità del robot di individuare zone di spazio già esplorate, riconoscendo l'ambiente tramite il confronto tra scansioni e mappe, e tra mappe e mappe.

La fase iniziale del lavoro è stata focalizzata sulla valutazione delle prestazioni dell'algoritmo di scan matching utilizzato nel resto dell'applicazione. Sono stati utilizzati diversi metodi per l'estrazione di features dalle scansioni e la stima dell'errore commesso dallo scan matching è stata ottenuta mediante confronto delle orientazione principali dei segmenti estratti da più scansioni.

Successivamente è stato realizzato l'algoritmo di creazione della mappa, basandosi su una struttura dati a grafo in cui i nodi sono associati alle mappe locali, mentre i vincoli rappresentano i vincoli geometrici tra le pose di riferimento per le mappe. I dati ottenuti dalla creazione della mappa sono stati elaborati con algoritmi per il calcolo della correlazione tra insiemi di punti, e metodi di associazione di segmenti come il *JCBB*. I risultati mostrati sono buoni per quanto riguarda il riconoscimento dell'orientazione principale di mappe e scansioni, mentre il posizionamento iniziale delle mappe per il calcolo della correlazione ha presentato alcune difficoltà in più. I maggiori problemi si sono manifestati nella difficoltà di trovare un buon metodo di calcolo della correlazione che tenesse conto del maggior numero di casi possibili, e nella scarsa precisione delle misure sensoriali che rende difficile il riconoscimento della posizione globale del robot mobile.

I possibili sviluppi futuri potranno mirare ad ottenere una migliore rappresentazione

dell'ambiente, utilizzando per esempio mappe a features, ed a migliorare il metodo di associazione basandosi proprio sulle features che costituiscono l'ambiente, sviluppando l'algoritmo JCBB per la creazione delle associazioni. Infatti procedendo in questa direzione si otterrebbe un sistema più robusto e veloce rispetto all'uso della correlazione, che risulta lento nel calcolo effettivo e comporta uno spreco di spazio in memoria dovuto alla memorizzazione degli ostacoli *e* dello spazio libero nelle mappe a griglia. Migliorando le prestazioni dell'algoritmo si potrebbe pensare ad una applicazione *real-time* della costruzione della mappa, per realizzare un sistema di *SLAM*.

Per quanto riguarda il problema del *loop closing* è necessario trovare un algoritmo che consenta una migliore stima della sovrapposizione iniziale tra due patch molto distanti, considerato l'errore accumulato nella creazione della mappa durante il movimento del robot.

Bibliografia

- [1] Lu, Milios, *Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans*, 1994
- [2] Andrea Censi. *An ICP variant using a point-to-line metric*, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, May 2008.
- [3] *Sick laser scanner*, <http://www.sick.com/home/en.html>
- [4] *Active Media* home page, <http://www.mobilerobots.com/>
- [5] *Player-Stage Framework*, <http://playerstage.sourceforge.net/>
- [6] *CARMEN Toolkit*, <http://carmen.sourceforge.net/>
- [7] Andrea Censi, Luca Iocchi, and Giorgio Grisetti. *Scan matching in the Hough domain*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pages 2739-2744, Barcelona, Spain, 2005.
- [8] Lu, Milios, *Globally Consistent Range Scan Alignment for Environment Mapping*, 1997.
- [9] Gutman, Konolige, *Incremental Mapping of Large Cyclic Environments*, 1999
- [10] Juan D. Tardós, *Data Association in SLAM*, Summer School on SLAM, 2002
- [11] José Neira, *Consensus in Data Association*, Summer School on SLAM, 2006