UNIVERSITÁ DEGLI STUDI DI PARMA
FACOLTÁ DI INGEGNERIA
Corso di laurea di Ingegneria Informatica

REALIZZAZIONE DI UN ALGORITMO DI SCAN MATCHING PER SISTEMI DI MISURA DELLA DISTANZA CON TECNOLOGIA LASER

Relatore:
Chiar.mo Prof. MONICA REGGIANI

Correlatori:
Chiar.mo Prof. STEFANO CASELLI
Dott. Ing. DARIO LODI RIZZINI

Tesi di laurea di:
GIOVANNI CAPRA

ANNO ACCADEMICO 2004-2005

Grazie a tutti e stato davvero bello fare questa tesi, spero vada a buon termine...tanto questa e una prova!

Indice

Capitolo 1

Introduzione

La percezione è dell'ambiente esterno è una delle componenti fondamentali di un sistema robotico. La rilevazione di informazioni dall'ambiente, infatti, assume grande importanza negli ambiti della localizzazione, della navigazione e della ricostruzione di mappe. Questo perchè uno degli obiettivi principali della robotica è quello di realizzare dei sistemi che possano svolgere autonomamente alcuni compiti e che quindi devono essere in grado di determinare la propria posizione e di navigare anche in ambienti sconosciuti. Le applicazioni per questo tipo di sistemi sono numerose e comprendono il monitoraggio di aree, l'interazione dell'uomo con il robot, l'intrattenimento, la sicurezza e molto altro.
Le informazioni su ciò che circonda il robot sono acquisite attraverso sensori. Esistono varie tipologie di sensori. Molto diffusi ad esempio sono i sensori di visione. L'utilizzo di questo tipo di dispositivo è in larga espansione a causa dei prezzi relativamente bassi delle videocamere e della grande disponibilità del mercato. Sono sensori che possiedono un'elevata potenza di calcolo ma intrinsecamente sono poco precisi. Notevole diffusione hanno anche i sensori di prossimità che ormai fanno parte dell'equipaggiamento standard di un robot e che garantiscono una buona affidabilità. Un posto di rilievo fra questi dispositivi è occupato dal laser scanner che risalta per la precisione e la risoluzione che è in grado di garantire. Proprio la precisione di questo strumento rende possibile la ricostruzione del profilo determinato degli ostacoli presenti nell'ambiente sul piano di scansione. Ottenere queste rilevazioni dettagliate è alla base di tutte le applicazioni più rilevanti della robotica.

Mediante il confronto fra due scansioni prese dal robot in posizioni diverse risulta possibile infatti calcolare la differenza di posizionamento del robot stesso una volta ricavata lo differenza di rotazione e di traslazione delle due scansioni. Si riesce in tal modo a "controllare" la navigazione del robot.
Il problema dell'allineamento delle scansioni, o scan matching, consiste nella ricerca di un moto rigido, solitamente bidimensionale, che porti alla sovrapposizione di due insiemi di punti ottenuti come informazioni sull'ambiente. Se il rilevatore laser è fisso rispetto al robot, la risoluzione dello scan matching permette di ricostruire il movimento del robot e di localizzarlo. Una scansione rappresenta solo un panorama parziale dello spazio in cui si sta agendo, quindi integrando e confrontando fra loro numerose scansioni, rilevate per esempio ogni qualvolta il robot muta la sua posizione, si riesce ad ottenere una ricostruzione più completa dell'ambiente. L'allineamento e la fusione di ogni scansione con quelle rilevate precedentemente risiede alla base di questa applicazione. In tal modo il risultato finale è la definizione di una vera e propria mappa dell'ambiente che circonda il robot e con cui esso deve interagire, come, per esempio, nell'evitarne gli ostacoli.
Questo lavoro di tesi si prefigge, da un lato di valutare le prestazioni di un laser scanner (laser Sick LMS 200), dall'altro di studiare il problema dello scan matching e di realizzare in libreria, attraverso l'uso del linguaggio di programmazione ad oggetti C++, un algoritmo di scan matching.
Le valutazioni sulle prestazioni del dispositivo laser sono state rappresentate attraverso l'uso della teoria della probabilità. Sono state prese infatti come modelli di riferimento rappresentazioni basate su variabili aleatorie con una certa distribuzione.
Sebbene l'impiego pratico dello scan matching sia relativamente recente esiste un discreto numero di algoritmi. Nella fase di analisi dello scan matching vengono analizzati alcuni dei metodi maggiormente conosciuti od utilizzati. La scelta di quale tecnica di matching sviluppare in questo progetto non è stata presa in termini di efficienza e prestazioni ma si è preferito puntare a tecniche già consolidate e con buona facilità di calcolo. Questo però non preclude la possibilità di implementare altre tecniche di matching in possibili sviluppi futuri. Si è deciso di sviluppare l'architettura della libreria in modo flessibile, ovvero essa fornisce soluzioni adatte ad organizzare varie modalità di matching e a gestire la scelta e l'uso di una di esse in

particolare.
Il lavoro di tesi è organizzato nel modo seguente. Nel capitolo 2, dopo una breve introduzione sui sensori, viene descritto lo strumento utilizzato per effettuare i test con la libreria, il laser SICK LMS 200, viene descritto il driver di interfacciamento tra il laser e il computer e infine vengono effettuate alcune considerazioni sulle prestazioni del dispositivo. Nel capitolo 3 vengono presentati inizialmente alcune applicazioni in relazione alla ricostruzione dei profili degli ambienti. Viene invece discusso successivamente nello specifico il problema dello scan matching e vengono presentati alcuni fra i più celebri algoritmi di scan matching, tra cui anche quelli implementati dalla libreria. Il capitolo 4 riporta una descrizione dettagliata delle classi della libreria, esegue un analisi degli strumenti e delle tecniche utilizzate per la realizzazione e infine riporta i risultati dei test applicativi. Nel capitolo 5 sono tratte le conclusioni finali sulla realizzazione della libreria e alcune considerazioni su possibili sviluppi futuri.

Capitolo 2

Scanner Laser SICK LMS 200

2.1 Uso dei sensori in robotica

Uno degli obiettivi della robotica mobile è la realizzazione di robot in grado di navigare in ambienti non strutturati. L'acquisizione di informazioni sull'ambiente è dunque essenziale per poter ottenere informazioni di posizione, rilevare ostacoli, pianificare il percorso del robot. I sensori, i dispositivi che restituiscono al robot informazioni sul contesto in cui opera sotto forma di misurazioni e dati numerici, risultano dunque fondamentali per il moto ed il posizionamento di un robot. Delle varie classificazioni per i sensori proposte in lettaratura la più significativa è quella che distingue sensori *propriocettivi* e sensori *eterocettivi* [1]. I propriocettivi sono i sensori che restituiscono informazioni sullo stato interno del robot, come, per esempio, lo stato dell'alimentazione, la velocità dei motori o l'angolo di sterzata di una ruota. Si tratta quindi di variabili che sono sotto il diretto controllo del robot. I sensori eterocettivi invece rilevano informazioni sull'ambiente esterno che circonda il robot e sono solitamente i dispositivi che vengono associati al termine "sensoriale". Tra i sensori eterocettivi spiccano i cosiddetti *sensori di prossimità*. Questi sono in grado di misurare la grandezza fisica relativa alla distanza fra il sensore e gli oggetti che gli sono vicini. Il loro funzionamento è basato sull'emissione di un segnale, che si propaga con una velocità nota, sulla riflessione di tale segnale da parte di oggetti esterni e sul calcolo del *tempo di volo*, cioè dell'intervallo di tempo fra l'invio e la ricezione del segnale. I sensori di prossimità si distinguono fra loro per il tipo di

segnale utilizzato.

- *Sensori ultrasonici(sonar)*. Questi sensori inviano onde di pressione con frequenza superiore a quella udibile dall'orecchio umano, generalmente fra i 40e i 180 kHz.

- *Laser*. Il dispositivo utilizza pacchetti di onde infrarosse inviandole verso un oggetto e ricevendo la sua riflessione (che può essere totale o parziale). Solitamente è possibile inviare il raggio secondo la direzione desiderata adoperando uno specchio rotante.

2.2 Descrizione dello scanner laser

La realizzazione di un algoritmo di Scan Matching per sistemi di misura della distanza è stata strettamente correlata, in questo progetto di tesi, all'utilizzo di uno scanner laser. I dati misurati con un dispositivo laser,infatti, sono solitamente impiegati allo scopo di determinare la distanza degli oggetti e definirne la loro posizione. Più concretamente, nello sviluppo del progetto è stato utilizzato lo *Scanner Laser SICK LMS 200*, in dotazione del Dipartimento di Ingegneria Informatica di Parma (si veda la figura 2.1).

Il principio alla base del funzionamento del dispositivo LMS è alquanto semplice. Il sistema opera attraverso la misura del *tempo di volo* degli impulsi laser: un fascio (`beam`) di impulsi laser viene emesso ed esso viene riflesso ogni qualvolta incontri un superficie riflettente. La riflessione viene poi registrata dal receiver dello scanner. Il tempo occorso tra l'emissione e la ricezione dell'impulso laser è direttamente proporzionale alla distanza tra lo scanner e l'oggetto (tempo di volo). Il fascio di impulsi inoltre viene deviato da uno specchio interno che ruota in modo tale da realizzare una scansione a forma di ventaglio (*fan-shaped scan*) dell'area circostante il dispositivo (radar laser). Il contorno dell'oggetto è determinato dall'insieme degli impulsi ricevuti. I dati misurati inoltre sono disponibili in tempo reale, per un'eventuale valutazione, in formato binario attraverso l'uso di una interfaccia seriale.
I dispositivi laser stanno avendo grande sviluppo poichè offrono una soluzione per un gran numero di applicazioni:

Figura 2.1: Scanner Laser SICK LMS 200.

- determinazione del volume degli oggetti,
- determinazione della posizione degli oggetti,
- prevenzione di collisioni per veicoli,
- classificazione degli oggetti,
- automazione dei processi,
- monitoraggio di aree e spazi aperti,
- determinazione del volume o dei contorni d oggetti di grandi dimensioni,
- ... e altro.

Vediamo ora in maniera più approfondita le specifiche e le caratteristiche del Laser SICK LMS 200.

Questo dispositivo realizza un sistema di misura della distanza che opera senza contatto attraverso la scansione dell'area circostante in due dimensioni. La scansione, come già spiegato precedentemente in questo paragrafo, avviene attraverso l'emissione di beams di impulsi. La *risoluzione* dello scanner laser dipende direttamente dalla frequenza con cui questi beams vengono emessi dal dispositivo. LMS 200 prevede tre possibili risoluzioni: l'intervallo con cui gli impulsi vengono rilasciati può essere settato ad un quarto di grado ($0,25\,°$), a mezzo grado ($0,5\,°$) o ad un grado ($1\,°$).
In combinazione con la risoluzione, vi è un'altra specifica dello scanner: la *massima risoluzione angolare* di scanning. Il dispositivo, infatti, non emette impulsi tutto intorno a sè ma limita il suo raggio d'azione ad un angolo predefinito, nel nostro caso ad un angolo equivalente a $180\,°$(la scansione viene effettuata in senso antiorario). Questo vale nel caso in cui si decida di utilizzare le due risoluzioni minori (quindi un grado e mezzo grado), poichè, per motivi di precisione e accuratezza, la massima risoluzione angolare, nel caso in cui si stia utilizzando una risoluzione ad un quarto di grado, è impostata a $100\,°$. In entrambi i casi però bisogna notare che l'angolo di apertura viene risulta sempre essere simmetrico al centro dello scanner laser. Facendo un rapido conto si nota che il massimo numero di valori ritornati da una scansione risulta essere: 181 per la risoluzione ad un grado, 361 per la risoluzione a mezzo grado e 401 per la risoluzione ad un quarto di grado. A seconda della risoluzione utilizzata varia anche un'altro parametro, quello del *tempo di acquisizione* di una scansione. Il laser può effettuare uno scanning in 13, 26 o 53 millisecondi in base alla configurazione che si è scelto di utilizzare. Si riesce ad osservare da questi dati che i tempi di risposta sono estremamente veloci in tutti e tre i casi, e questo rende il dispositivo particolarmente adatto per applicazioni in real-time. I dati appena descritti sono riassunti e schematizzati nella tabella 2.1.

Risoluzione	$0,25\,°$	$0,5\,°$	$1\,°$
Max. Risoluzione angolare	$100\,°$	$180\,°$	$180\,°$
Max. numero di valori	401	361	181
Tempo di risposta	53 ms	26 ms	13 ms

Tabella 2.1: Specifiche dello scanner LMS 200.

Il *range* di azione del dispositivo LMS è abbastanza ampio, si possono effet-

tuare scansioni che riconoscano oggetti fino ad 80 metri di distanza. Naturalmente man mano che la distanza aumenta l'errore di precisione nel rilevamento cresce. In condizioni di buona visibilità e a temperatura ambiente, si misura un errore di precisione di circa ± 15 millimetri, in un range che va dagli 1 agli 8 metri, e un errore di precisione di circa ± 4 centimetri in un range che va dagli 1 ai 20 metri. La varietà di applicazioni per le quali l'LMS risulta essere adatto è dovuto non solo all'ampio raggio di azione ma anche al fatto che esso riesce a lavorare in una molteplicità di situazioni ambientali. Il suo comportamento, ad esempio, non muta considerevolmente in un range di temperature che variano dai $-30\,^{\circ}\mathrm{C}$ ai $+70\,^{\circ}\mathrm{C}$.
Per quanto riguarda le specifiche elettriche del dispositivo SICK LMS 200, esso richiede un'alimentazione di 24 Volt. Nella realizzazione di questo progetto, per alimentare lo scanner nella fase di raccolta dati, si sono sempre adoperate due batterie in piombo da 12 Volt ciascuna collegate in serie tramite un cavo. Il laser riesce a comunicare con un computer attraverso un'interfaccia di tipo seriale, che riporta quindi i dati in formato binario. Tale interfaccia è in grado di supportare sia una porta seriale di tipo RS232, sia una porta seriale di tipo RS422 (quest'ultima più veloce rispetto alla precedente e usata in maggior misura nel campo industriale). Questo tipo di collegamento non si è rivelato del tutto ottimale, come discusso nel paragrafo 2.4. I dati descritti in questo paragrafo sono tutti tratti dalla documentazione fornita dalla casa di fabbricazione dello scanner laser [2].

2.3 Driver utilizzato

Avendo il laser numerose possibilità di configurazione occorre necessariamente un programma che permetta di impostare le varie modalità di utilizzo. Il costruttore fornisce assieme al laser scanner il software di prova *LMS/LMI user software v 5.11* per piattaforma Windows. Questo programma, una volta collegato lo scanner al computer, permette all'utente di definire varie configurazioni: si può scegliere quale risoluzione delle tre supportate adoperare, su quale angolo di azione effettuare la scansione ($\leq 180\circ$) e a quale distanza limitare il campo di azione. Inoltre il software è predisposto per fornire un'interfaccia grafica che visualizzi la mappa a ventaglio dell'area presa in esame durante il funzionamento del dispositivo laser. È possibile osservare un esempio della rappresentazione grafica di una scansione

fornita dal software in questione in figura 2.2.

Un equivalente per il sistema operativo Linux non viene purtroppo rilasciato dalla

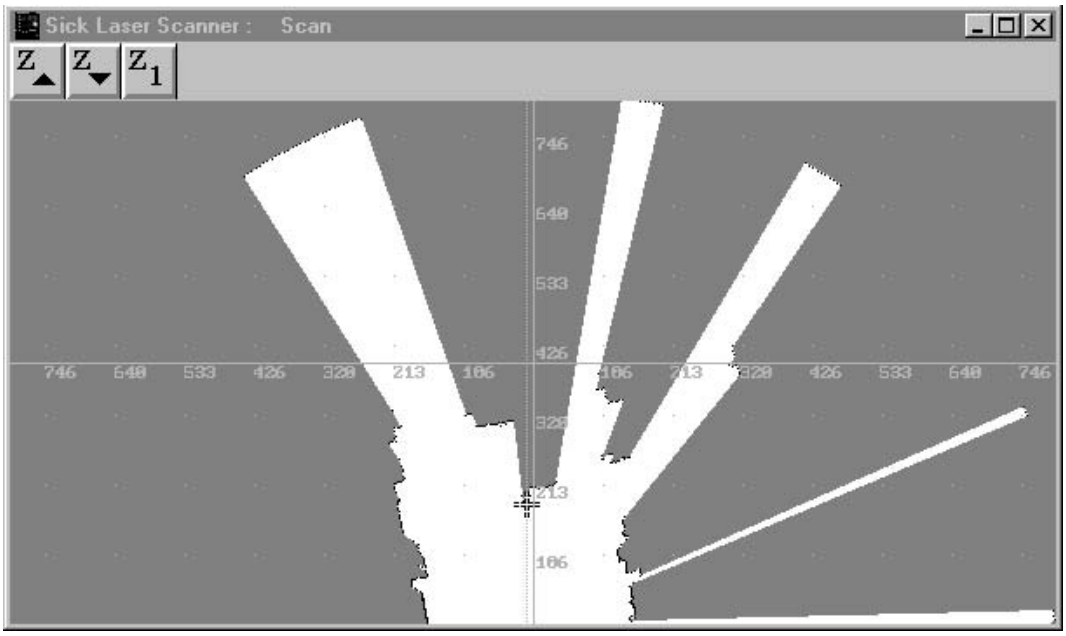

Figura 2.2: Scansione visualizzata con LMS user software.

casa di fabbricazione e bisogna quindi realizzare un proprio driver se si vuole gestire lo strumento anche in tale ambito. Lo scopo di questo lavoro di tesi non era quello di realizzare un driver per l'LMS perciò tale tipo di problema non è stato in questa sede affrontato, ma si è scelto di usufruire di driver già realizzati e trovati sulla rete. In particolare sono stati utilizzati i driver sviluppati nel 2002 da Stefano Zanero e Claudio Merloni dell'Istituto Politecnico dell'Università di Milano. Essi sono interamente realizzati in linguaggio di programmazione ad oggetti C++ e ne viene rilasciato il codice sorgente. Il codice è completamente opensource, in modo che si possa utilizzare o modificare a seconda delle esigenze. Le librerie implementate forniscono i metodi per la comunicazione con il dispositivo e l'acquisizione dei dati. Assieme al driver vero e proprio vengono fornite diverse classi per il filtraggio che consentono di ottenere un profilo più dolce e di eliminare il rumore presente alle alte frequenze. Il concetto alla base di tutte le classi del driver è quello di *PolyLine*. I dati vengono restituiti dallo scanner, tramite la porta seriale, sotto forma di

un semplice array di misure di distanza, che vengono rilevate eseguendo una scansione in senso antiorario. Da questo array viene estratta una rappresentazione più maneggevole, la PolyLine appunto, ovvero una struttura costituita da un elenco di punti logicamente considerati uniti sequenzialmente da dei segmenti.
Il driver vero e proprio è costituito dalla classe *LaserDriver* che contiene al suo interno i metodi fondamentali per il corretto funzionamento del dispositivo. La funzione `scan()`, definita in LaserDriver, permette di eseguire una scansione e ritorna un puntatore ad un oggetto LaserData, contenitore in cui vengono immagazzinati un insieme di misure fatte dal laser. Tale metodo è quindi alla base del driver visto che quest'ultimo ha la funzione principale di comunicare con lo scanner mediante lo scambio di caratteri attraverso un'interfaccia seriale. Oltre a questo metodo fondamentale, all'interno della classe LaserDriver sono presenti, come membri privati, le funzioni per settare la configurazione dello strumento, tra le quali riveste un ruolo principale il metodo `SetLaserScan(...)` che permette di impostare la risoluzione e l'angolo di azione di una scansione. Questo metodo essendo privato, in teoria, non dovrebbe essere usato dagli utenti che però possono utilizzare per la configurazione il wrapper pubblico `setResolution()`. I dati immagazzinati in un oggetto di tipo LaserData possono essere restituiti sotto forma di PolyLine attraverso il metodo `getData()`, membro pubblico della classe *LaserData*. Inoltre è possibile scegliere se visualizzare i dati in coordinate polari o in coordinate cartesiane, usando rispettivamente i metodi `getR()`, `getTheta()` e `getX()`, `getY()`, implementati nella classe *LaserPoint*. La struttura del driver è riassunta nel diagramma delle classi in figura 2.3.
Insieme alle classi che implementano il driver vengono forniti dei file sorgenti Java che comprendono la libreria *DrawingPanel.java* e due programmi: *PolyViewer.java*, che visualizza in un'interfaccia grafica i dati presi da un file aggiornandoli alla pressione di un tasto, e *PolyViewerLoop.java*, che viceversa aggiorna continuamente i dati.
Per i fini di questo progetto di tesi, l'utilizzo del dispositivo laser è stato limitato all'acquisizione di dati sui quali verificare la correttezza dell'algoritmo sviluppato. Il driver [3] utilizzato non si è rivelato particolarmente adatto a questa modalità di utilizzo, poichè maggiormente finalizzato alla restituzione di dati di elevata qualità attraverso l'uso di numerosi filtri. Purtroppo le operazioni di filtraggio causano un

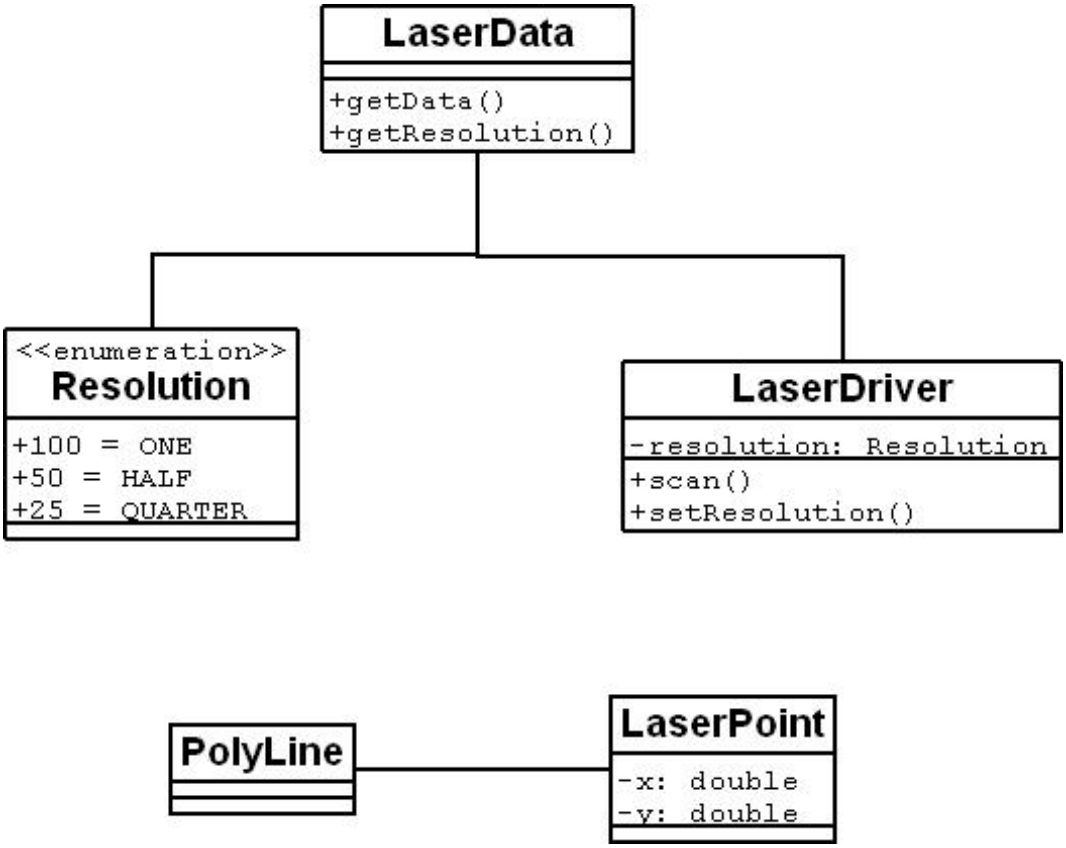

Figura 2.3: Diagramma UML delle classi del driver.

tempo maggiore di elaborazione e quindi di restituzione dei dati acquisiti, rallentando l'intero processo. Inoltre il filtraggio elimina informazioni e questo potrebbe creare problemi ai fini della precisione dell'algoritmo. Il driver perciò non risulta funzionale se si desiderano raccogliere molte scansioni in un tempo limitato.
Con lo scopo di aumentare le prestazioni del dispositivo laser, riuscendo a sfruttare a pieno le potenzialità dello strumento, nel Laboratorio di Robotica dell'Università di Parma è in corso la ricerca e la parziale riscrittura di driver, attraverso il linguaggio di programmazione in C++, modificando e migliorando un driver preesistente in PlayerStage. Con tale progetto si cercherà di eliminare tutti quei processi che possano in qualche modo rallentare l'elaborazione, in modo da poter effettivamente sfruttare lo scanner per applicazioni in tempo reale. Inoltre sarà possibile attraverso l'uso di questo driver interfacciare lo scanner LMS con una porta seriale RS422,

già acquistata, che risulta nettamente più veloce delle porte RS232 presenti sui computer del laboratorio.

2.4 Analisi delle prestazioni del laser

L'assenza di dati sperimentali diretti ottenuti con lo strumento SICK LMS 200 ha reso necessario l'esecuzione di test di precisione e più in generale di prestazioni. I test prevedevano soprattutto una larga raccolta di dati su cui compiere le analisi. Tutti gli esperimenti sono stati effettuati collegando lo scanner ad un computer portatile con processore AMD Turion 64 ML-30 e 512 MegaByte di RAM. Essendo tale computer sprovvisto di una porta seriale si è rivelato necessario utilizzare un convertitore serial-to-USB per poter interfacciare il laser con il terminale. Bisogna sottolineare il fatto che l'impiego di un adattatore non ha in alcun modo alterato i risultati dei test essendo la velocità di trasmissione della porta seriale perfettamente compatibile con la velocità della porta USB. L'alimentazione necessaria per lo scanner è stata fornita da due batterie al piombo da 12 Volt l'una collegate in serie.
Le applicazioni di interesse non richiedono l'impiego del laser per la ricostruzione della mappatura dello spazio, o comunque non si richiedono particolari caratteristiche ambientali. Per questo motivo, nonostante la presenza di numerosi ostacoli quali armadi, tavoli, computer e altro, le scansioni sono state tutte rilevate all'interno del laboratorio di Robotica presso la palazzina 1 della sede scientifica di Ingegneria. Un esempio di una rilevazione acquisita dallo scanner è visibile in figura 2.4.
L'obiettivo principale di queste prime analisi era la verifica della precisione dei rilevamenti eseguiti dallo scanner LMS, o meglio l'analisi dell'*errore di precisione* a cui è soggetta una scansione. Come già accennato precedentemente, il dispositivo opera attraverso l'emissione di beams di impulsi laser e la ricezione della riflessione degli stessi impulsi ogni qualvolta incontrino un ostacolo, calcolando il tempo di volo dei raggi e conseguentemente la distanza fra oggetto e scanner. Purtroppo questo calcolo risulta sempre soggetto ad approssimazioni o ad errori di precisione. Lo scopo delle analisi fatte è proprio quello di verificare l'entità e la distribuzione dell'errore di approssimazione, supponendo di poterlo descrivere con una variabile aleatoria. Si è proceduto quindi alla raccolta di dati in maniera un po' particolare: per verificare la precisione delle rilevazioni dello scanner non occorre registrare tut-

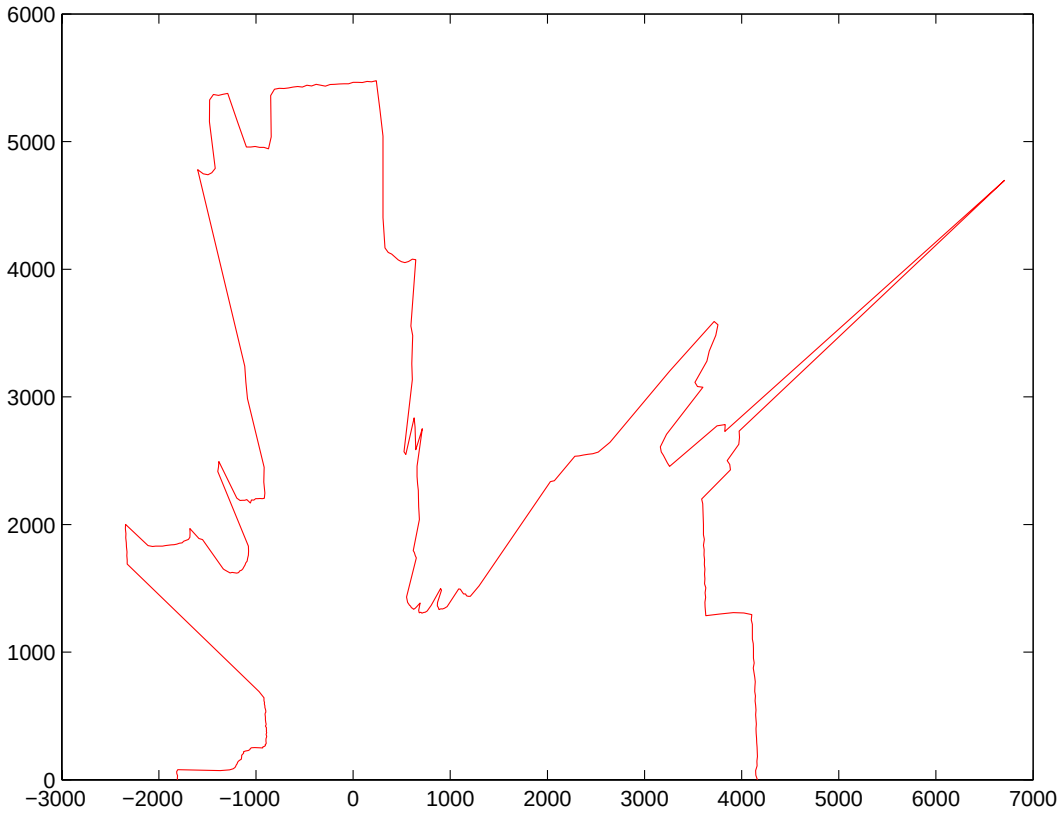

Figura 2.4: Scansione del laboratorio di robotica.

ti i punti di una scansione, ma è sufficiente focalizzare l'attenzione su un singolo beam. Il programma di test consiste in un semplice listato nel linguaggio di programmazione C++, che prevede l'esecuzione di 4000 scansioni acquisendo i dati in coordinate polari (ovvero restituendo la distanza dallo scanner e l'angolazione con la quale i raggi sono stati emessi). Di questi viene memorizzato in un file di testo specificato da riga di comando solo il dato relativo allo stesso singolo beam, ovvero sempre la distanza relativa allo stesso angolo. Il beam preso in considerazione nel nostro caso è stato quello centrale avente un'angolazione in radianti uguale a $\frac{\pi}{2}$. Ai fini del risultato delle analisi la scelta di un beam piuttosto di un altro non è restrittiva e non incide sulle conclusioni. L'acquisizione delle 4000 scansioni è stata ripetuta numerose volte in modo da ottenere una buona base statistica per le analisi. La scelta di reiterare l'esecuzione del programma di test quattromila volte è stata

forzata da alcuni limiti nella velocità di acquisizione dei dati da parte del laser (per raccogliere 4000 dati occorrevano circa 20 minuti). Sarebbe stato più opportuno prendere un numero maggiore di campioni (sull'ordine dei diecimila), in modo da avere maggiore precisione nei risultati finali dell'analisi, anche se alla fine il numero di dati raccolti si è rivelato comunque sufficiente per gli scopi prefissati.
Conclusa la fase di acquisizione dei campioni si è proceduto alla successiva fase di analisi. A tale scopo è stato utilizzato il software *Matlab version 6.5 R 13* per il sistema operativo Windows. Nell'analisi dei dati risulta interessante il confronto dell'istogramma sperimentale con la funzione di densità di probabilità (PDF) proposta da Sebastian Thrun, Wolfram Burgard e Dieter Fox[4] come modello per un beam. Il modello proposto integra fra loro quattro tipi di rilevamento di errori: piccole misure di errori durante il corretto rilevamento, errori dovuti ad oggetti inaspettati (come il passaggio di persone ad esempio), errori dovuti al fallimento del riconoscimento di ostacoli e rumori random inaspettati. Più in dettaglio:

- **Rilevamento corretto con errori locali**
  Anche se il sensore misura correttamente la distanza degli oggetti, il valore che ritorna è sempre soggetto a rumore dovuto alla risoluzione limitata dei sensori, ad eventi atmosferici sulla misura del segnale e così via. Tale errore è solitamente modellato da una curva *gaussiana* con una determinata *media* z_t^{k*} e una determinata *deviazione standard* σ.

$$p(z_t^k|x_t, m) = \begin{cases} N(z_t^k, z_t^{k*}, \sigma^2) & \text{se } 0 \leq z_t^k \leq z_{max} \\ 0 & \text{altrimenti} \end{cases} \tag{2.1}$$

  .

  dove

$$N(z_t^k, z_t^{k*}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2} \cdot \frac{z_t^k - z^{k*}{}_t}{\sigma^2}}$$

- **Oggetti inaspettati**
  La presenza temporanea di oggetti tra il sensore e l'ipotetico oggetto sulla traiettoria di un beam altera la misura e la pdf. Il modello sensoriale usato da un localizzatore ne deve tener conto. Matematicamente, la probabilità del-

la misura della distanza in queste situazioni è descritta da una *distribuzione esponenziale*.

$$p(z_t^k|x_t, m) = \begin{cases} \lambda e^{-\lambda z_t^k} & \text{se } 0 \leq z_t^k \leq z_{max} \\ 0 & \text{altrimenti} \end{cases} \tag{2.2}$$

- **Fallimenti nel rilevamento**
  Inoltre va considerata la possibilità di rilevare falsi negativi. Un risultato tipico di un fallimento del sensore è una misura di tipo max-range: il sensore ritorna il suo massimo valore raggiungibile z_{max}. Si modella questo caso rappresentandolo con una *delta di Dirac*, centrandola nel massimo valore raggiungibile.
  $$p(z_t^k|x_t, m) = \begin{cases} 1 & \text{se z = } z_{max} \\ 0 & \text{altrimenti} \end{cases} \tag{2.3}$$

- **Misurazioni random**
  I sensori occasionalmente producono falsi positivi. Per fare un esempio letture fantasma possono essere prodotte da cross talking, interferenze o altro. Ricorrendo ad un semplice modello a PDF con *distribuzione uniforme* sull'intero range di misure del sensore $[0, z_{max}]$ si riesce a valutare anche questi fenomeni.
  $$p(z_t^k|x_t, m) = \begin{cases} \frac{1}{z_{max}} & \text{se z = } z_{max} \\ 0 & \text{altrimenti} \end{cases} \tag{2.4}$$

Attraverso un banalissimo script di Matlab, i dati memorizzati in un file di testo sono stati inseriti in una matrice di due colonne: la prima colonna contenente il valore della distanza e la seconda contenente l'angolo del beam. Sono stati poi visualizzati i valori degli elementi della prima colonna e inoltre di essi si sono calcolate la media e la deviazione standard. Per rappresentare l'andamento dell'errore, invece,siè fatto ricorso all'uso di un istogramma. Sull'asse delle ascisse è stato riposto il valore della distanza (l'intero range dei possibili valori è stato suddiviso in 50 intervalli) e sull'asse delle ordinate il numero di rilevazioni aventi un dato valore. I risultati ottenuti si possono osservare in figura 2.5.

Nel riquadro a sinistra in rosso sono visualizzati i quattromila valori della distanza presi sempre sullo stesso beam. La linea retta verde orizzontale continua rap-

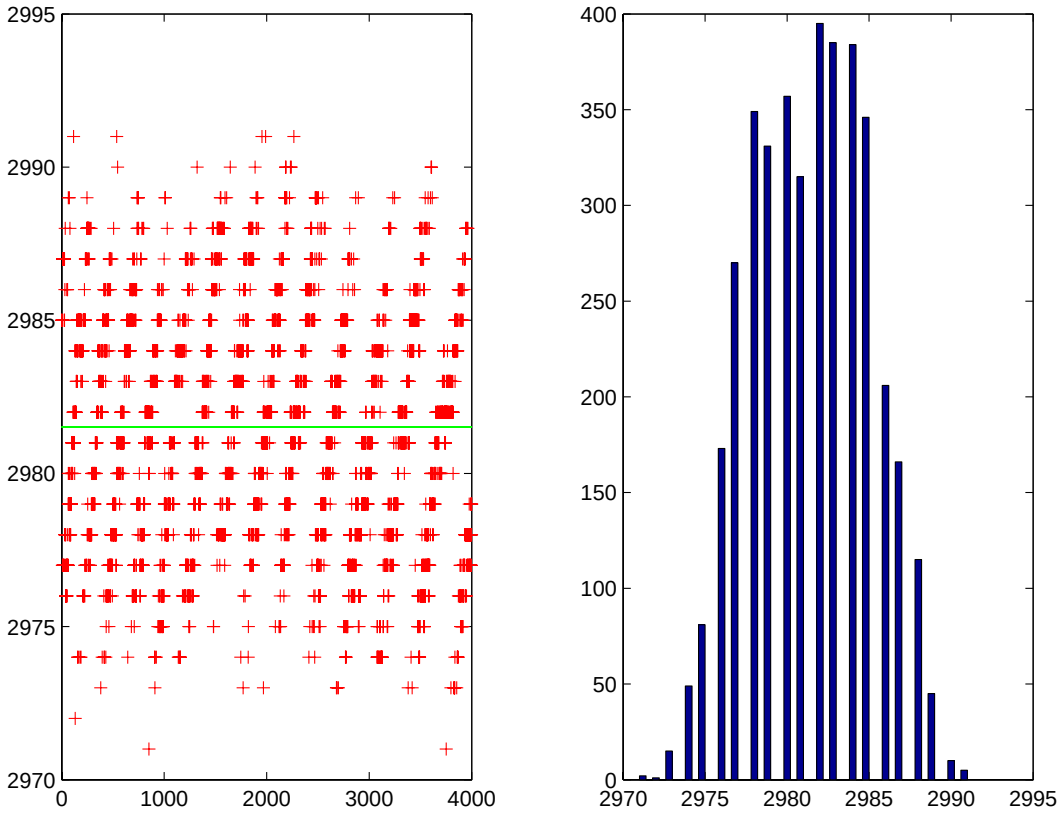

Figura 2.5: Andamento dell'errore.

presenta la media di tali misure. In tutti gli esperimenti eseguita, quindi ogni volta con quattromila dati diversi, si è potuta rilevare una deviazione quadratica media σ^2 di circa 12,7 centimetri, valore propriamente non trascurabile. Nel riquadro a destra invece è visualizzato l'andamento dell'errore e, come balza subito all'occhio, esso rispecchia chiaramente una distribuzione di tipo gaussiana. Il picco della curva si ottiene naturalmente molto vicino al valore della media calcolata. Rispetto al modello riportato[4] è qui visibile solo l'effetto dell'errore locale dovuto al rumore gaussiano. Nelle osservazioni sperimentali effettuate in presenza di ostacoli temporanei non è stata riscontrata una significativa differenza rispetto al modello gaussiano ed in ogni caso la distribuzione ottenuta si discosta di molto dal modello esponenziale. Probabilmente sarebbe stato necessario modulare opportunamente la distanza dell'oggetto occludente dal sensore.

In casi isolati l'istogramma si è discostato da un andamento gaussiano presentando un minimo in prossimità dell'asse di simmetria, quindi intorno al valor medio. Tale PDF pseudo-bimodale potrebbe essere attribuita allo scarso numero di misure effettuate. Il fenomeno può comunque essere ritenuto trascurabile, data la rarità della sua comparsa.

Capitolo 3

Scan Matching

Nel capitolo precedente sono state presentate le caratteristiche principali dello sensore laser. Ciò che lo contraddistingue da altri sensori di prossimità è l'elevata risoluzione angolare e la precisione (caratteristiche che sono state verificate sperimentalmente). Grazie a tale strumento è possibile ricostruire il profilo degli ostacoli che si presentano sul piano di scansione. Ciò risulta estremamente importante per numerose applicazioni in robotica. Le principali sono discusse brevemente qui di seguito:

- **Odometria**
  L'odometria è il metodo di navigazione più utilizzato per il posizionamento dei robot mobili e generalmente sfrutta sensori di tipo propriocettivi. L'idea fondamentale alla base dell'odometria è l'integrazione di informazioni incrementali sul moto durante un periodo determinato, che purtroppo porta inevitabilmente all'accumulo di errori. In particolare l'accumulazione di errori di orientamento può causare grandi errori di posizionamento che crescono proporzionalmente alla distanza percorsa dal robot. Un esempio di questa imprecisione è visibile in figura 3.1. Nonostante queste limitazioni, l'odometria è una parte importante del sistema di navigazione. Per esempio gli algoritmi più diffusi per la localizzazione integrano le informazioni odometriche [5]. I metodi di scan matching propongono un diverso approccio nella ricostruzione dello spostamento del robot.

- **Map-based positioning**

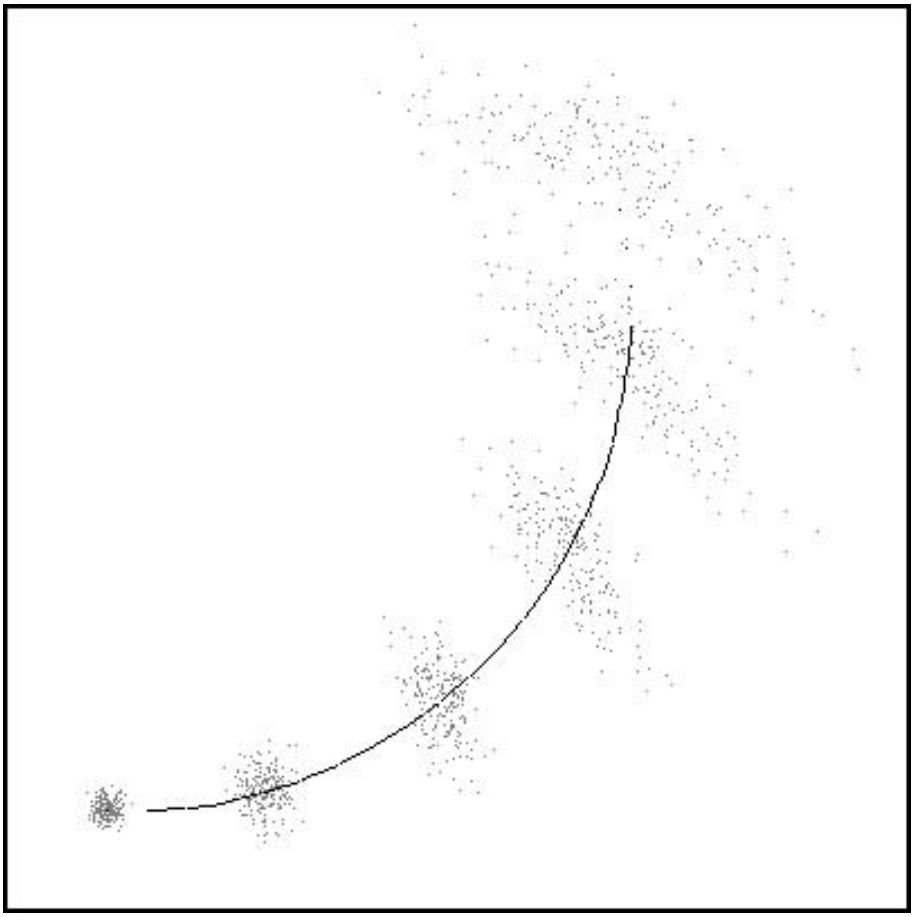

Figura 3.1: Incertezza dell'odometria.

Il Map-based positioning è una tecnica nella quale il robot usa i propri sensori per creare una mappa dell'ambiente circostante. Vengono utilizzati perciò principalmente i sensori eterocettivi. Questa mappa locale viene poi comparata con una mappa globale precedentemente immagazzinata in memoria. Se viene riscontrata una corrispondenza fra le due mappe allora il robot può computare la sua posizione attuale e il suo orientamento nell'ambiente circostante. Uno dei più importanti aspetti della navigazione map-based è il *Map-matching*, processo che stabilisce la corrispondenza fra la mappa locale e la mappa globale. [1]

- **Simultaneous Localization and Mapping (SLAM)**
  Il problema che il Simultaneous Localization and Mapping tenta di risolvere

consiste nella possibilità di lasciare libero un robot di muoversi in un ambiente sconosciuto usando i sensori di prossimità e simultaneamente di costruire una mappa dell'ambiente attraverso i dati raccolti dai sensori [6]. Siccome una scansione rappresenta una vista parziale dell'ambiente, confrontando e integrando fra loro molte scansioni prese in posizioni differenti si riesce ad ottenere una descrizione dell'ambiente più completa. Il punto fondamentale è allineare in maniera appropriata le scansioni in modo che esse possano essere fuse fra loro. La difficoltà risiede nel fatto che l'informazione odometrica da sola non è sufficiente per determinare il posizionamento e inoltre non si riescono ad utilizzare modelli preesistenti per correggere gli errori di posizionamento perchè l'ambiente in cui si sta lavorando è sconosciuto. Un approccio generale per la costruzione del modello dell'ambiente è l'integrazione incrementale dei nuovi dati al modello. ogni scansione rilevata dal sensore viene allineata con la scansione precedente rilevata o con il modello cumulativo globale.

3.1 Formulazione del problema di Scan Matching

Un problema di Scan Matching consiste nel determinare, dati due insiemi di punti, un moto rigido bidimensionale che porti ad una sovrapposizione dei due insiemi di punti. Tipicamente un insieme di punti è il risultato di un'acquisizione sensoriale, in genere con un laser. Una situazione comune in cui lo scan matching si applica è la seguente: un robot situato inizialmente in una posizione P_{ref}, dove per P_{ref} si intende il vettore delle coordinate e dell'orientamento del robot sul piano di movimento, effettua una scansione S_{ref} con un sensore di prossimità fissato rigidamente su di esso. In seguito ad uno spostamento esso si porta in posizione P_{new} e nella nuova posizione effettua una nuova scansione, S_{new}. Il compito dello scan matching è quello di determinare con esattezza la differenza tra la posizione P_{new} e la posizione P_{ref} mediante l'allineamento delle due scansioni. Riassumendo, il problema di matching si può formulare come segue: assumendo che la posizione di S_{new} sia P'_{new}, occorre trovare una rotazione ω e una traslazione T per S_{new} tale che, dopo la trasformazione, S_{new} sia allineata con S_{ref}.
L'approccio più comune al problema dello scan matching risiede nel definire una

misura della distanza fra le due scansioni e ricercare una appropriata trasformazione rigida che minimizza tale distanza. Lo spazio in cui effettuare questa distanza è essenzialmente tri-dimensionale (rotazione e traslazione bi-dimensionale).
Una prima generale classificazione degli algoritmi di scan matching prevede due categorie [1]:

- *Matching icon-based*: accoppia i punti di una rilevazione con i punti caratteristici della mappa precedentemente memorizzata basandosi sulla distanza minima. La posizione del robot è risolta attraverso la minimizzazione dell'errore della distanza tra i punti e i loro corrispondenti sulla mappa. Basandosi sulla nuova posizione, le corrispondenze sono ricalcolate e il processo viene ripetuto fino a quando l'errore sulla distanza non cade sotto un valore di soglia prestabilito.

- *Matching feature-based*: fa corrispondere i dati acquisiti ad un piccolo insieme di punti caratteristici che saranno poi confrontati con la mappa memorizzata.

Nel paragrafo 2.4 sono stati messi in evidenza i limiti dello strumento laser: la rumorosità, l'errore di approssimazione, etc. L'allineamento perfetto risulta pertanto impossibile. Inoltre a seguito dello spostamento alcuni degli ostacoli individuati in una data scansione non sono visiili in quella successiva, a causa del limitato campo di visibilità del laser e della presenza di oggetti occludenti. Per questo motivo nel corso degli anni la ricerca ha portato alla formulazione di numerosi algoritmi di scan matching, ognuno di essi volto a migliorare la precisione dell'allineamento e la velocità delle operazioni. Nel paragrafo successivo vengono presentati alcuni degli algoritmi di matching descritti in letteratura.

3.2 Algoritmi di Scan Matching

3.2.1 Scan Matching con trasformata di Hough

Uno degli algoritmi usati per fare scan matching è l'*Hough Scan Matching (HSM)* [7] che deve il proprio nome all'impiego della [**?**]. La trasformata di Hough è usata

da tempo nell'ambito della visione artificiale per ricercare all'interno di un'immagine digitale pattern corrispondenti a curve geometriche.
Anche nel caso dello scan matching l'obiettivo è l'individuazione delle rette che congiungono i punti di una scansione. La ricerca si svolge a partire da una equazione parametrica della retta:

$$x \cdot cos(\theta) + y \cdot sin(\theta) = \rho. \tag{3.1}$$

La ricerca delle rette viene poi effettuata nel piano dei parametri tracciando le curve duali rispetto ai punti (x, y). Nel caso delle rette parametriche come (3.1) si ottiene:

$$\begin{aligned} \rho &= \sqrt{x^2+y^2} \cdot \left[\frac{x}{\sqrt{x^2+y^2}} cos(\theta) + \frac{y}{\sqrt{x^2+y^2}} sin(\theta) \right] \\ &= \sqrt{x^2+y^2} \cdot cos[\theta - atan2(y, x)] \end{aligned}$$

La trasformata di Hough dell'insieme di punti dati è un funzionale che associa a ciascun (ρ, θ) e alla equazione della retta, il numero di curve duali che passano per il dato punto. I passaggi descritti appena sopra sono visibili in figura 3.2, nella quale il cerchietto rosso evidenzia l'intersezione delle curve nello spazio parametrico. Nell'elaborazione su di un calcolatore ciò che si calcola è in realtà la trasformata di Hough discreta (DHT): il piano dei parametri è suddiviso in celle di ampiezza data cui è associato il numero di curve parametriche passanti all'interno della cella.
L'algoritmo proposto [7] sfrutta le proprietà della HT della retta parametrizzata come in (3.1). In particolare la rotazione dei punti nel piano cartesiano produce un movimento rigido nel piano dei parametri (ρ, θ). Scelto un opportuno funzionale $g[\,\cdot\,]$ che applicato alla HT sia invariante alla rotazione, risulta definito lo *spettro di Hough*:

$$HS_g(\theta) = HS_g(\theta + \phi) \forall \phi$$

Lo spettro di Hough HS è allora invariante alla rotazione. Una scelta possibile per $g[\,\cdot\,]$ è la funzione energia della sequenza dell'insieme nel dominio discreto:

$$g[f] = \sum_i f_i^2$$

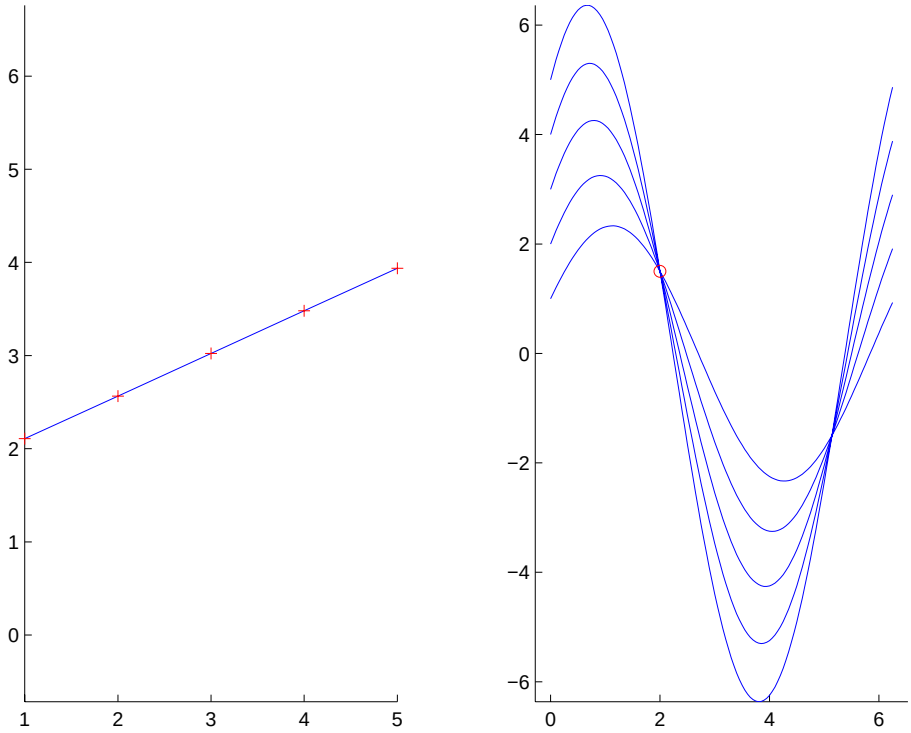

Figura 3.2: Rappresentazione delle rette nello spazio dei parametri e trasformata di Hough.

Il ricorso allo spettro di Hough semplifica la ricerca dell'angolo di rotazione, riconducendola alla semplice correlazione dei due HS associati alle scansioni da allineare. La stima della traslazione viene invece effettuata confrontando le HT una volta che l'angolo di rotazione ϕ è stato stimato.
In sintesi i passi dell'algoritmo sono i seguenti:

1. La trasformata di Hough e lo spettro di Hough (che ha la proprietà di essere invariante rispetto alla traslazione)sono calcolati sia per i dati rilevati dal sensore sia per i dati di riferimento.

2. I massimi locali della cross-correlazione degli spettri sono utilizzati per avanzare ipotesi sull'angolo di rotazione θ.

3. Per ogni ipotesi su θ delle limitazioni sul vettore di traslazione T sono prodotte dalla correlazione delle colonne della trasformata di Hough.

4. Due alternative:

- le limitazioni lineari sono combinate per produrre ipotesi su T e le soluzioni sono ordinate con una funzione di probabilità.
- i risultati delle correlazioni sono accumulati in un buffer che produce un output di densità.

3.2.2 Scan Matching sviluppato da Cox

Un precursore dei metodi di scan matching è quello sviluppato da Ingemar J. Cox [8]. Il matching, secondo tale tecnica, è composto principalmente dall'estrazione di tratti caratteristici (*feature*), solitamente dei segmenti, dal modello (una mappa dell'ambiente in cui si sta lavorando precedentemente memorizzata) e dalla corretta corrispondenza tra l'immagine e i feature del modello attraverso l'uso di una qualche forma di ricerca limitata. Si tratta pertanto di un algoritmo di tipo feature-based. Ciò rende questa tecnica diversa dalle altre presentate in questo capitolo. Come sarà possibile vedere tutti gli altri algoritmi si fondano sul confronto fra tutti i punti di un immagine con tutti i punti dell'altra immagine (algoritmi icon-based). Si può notare che se lo scarto fra l'immagine e il modello è relativamente piccolo, allora per ogni punto dell'immagine il segmento corrispondente nel modello si avvicina molto ad essere il segmento più vicino al punto nel modello. La determinazione della corrispondenza (parziale) tra i punti dell'immagine e le linee del modello si riduce ad una semplice ricerca del segmento più vicino ad ogni punto. L'operazione che sta alla base dell'algoritmo consiste nella determinazione di una corrispondenza approssimata fra i punti dell'immagine e le linee del modello tale da ridurre lo scostamento fra le due immagini. L'iterazione di questo metodo porta al seguente algoritmo:

1. Per ogni punto dell'immagine trovare il segmento del modello più vicino al punto. Chiamare questo *target*.
2. Trovare la corrispondenza che minimizza la distanza totale quadratica tra i punti dell'immagine e i loro target.
3. Muovere i punti a seconda della corrispondenza trovata in (2).

4. Ripetere i passi (1-3) fino alla convergenza della procedura. La composizione delle corrispondenze di tutti i passi è la corrispondenza totale desiderata.

Il passo (2) è computazionalmente il punto maggiormente complesso. Per questo motivo sono state introdotte due aprossimazioni. Per prima cosa ogni target viene cambiato dal segmento alla linea infinita contenente il segmento. Inoltre, siccome la dipendenza dei punti mossi dall'angolo di rotazione θ non è lineare, questa dipendenza è approssimata ai termini di primo ordine in 0.
Ogni corrispondenza può essere descritta come una rotazione di un certo angolo θ seguita da una traslazione di T, e questo è denotato da (t,θ). La corrispondenza (t,θ) correla ogni punto x $\rightarrow$ R(θ) $\cdot$ (x-c)+(c+T), dove c è il centro di rotazione e R è la matrice di rotazione

$$R = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$$

Si tende a scegliere come centro di rotazione il baricentro dell'immagine che si vuole confrontare con il modello. Questo perchè avere un centro che si muove insieme all'immagine ha il vantaggio che una trasformazione (t_1, θ_1) seguita da una trasformazione (t_2, θ_2) ha lo stesso effetto sulla figura di una trasformazione $(t_1 + t_2, \theta_1 + \theta_2)$.

Si desidera trovare il valore di (t,θ) che minimizza

$$S = \sum_i \Big([R(\theta) \cdot (v_i - c) + (c + t)]' \cdot u_i - r_i \Big)$$

dove v_i sono i punti dell'immagine, u_i i loro corrispondenti target e ' denota la trasposizione. La fase di minimizzazione dell'errore quadratico viene ripresa da numerose tecniche di matching ed è alla base degli algoritmi illustrati in seguito.

3.2.3 Iterative Closest Point (ICP)

L'*Iterative Closest Point (ICP)* è uno degli algoritmi di scan matching maggiormente utilizzati e si basa sull'associazione di corrispondenze punto a punto. Si tratta di un algoritmo iterativo dove per prima cosa si calcolano le corrispondenze tra le scansioni che si vogliono confrontare e successivamente si minimizza l'errore di distanza per computare lo scarto fra le scansioni. Questo metodo si fonda sull'idea

seguente. Per ogni punto P_i di una scansione, che chiameremo S_{new}, si usa una semplice regola (indipendente dalla rotazione e dalla traslazione) per determinare un punto corrispondente P'_i sulla scansione o mappa di riferimento, che chiameremo S_{ref}. Poi per ogni coppia di punti corrispondenti si computa una soluzione ai minimi quadrati per le relative rotazione e traslazione. Questa soluzione viene applicata per ridurre l'errore di posizionamento fra le due scansioni. Si ripete questo processo fino a che esso non converge.
La soluzione ai minimi quadrati deriva dalla minimizzazione della seguente funzione per la distanza, la quale è definita su *n* coppie di punti corrispondenti P(x_i,y_i) e P′(x'$_i$,y'$_i$):

$$E_{dist}(\omega, T) = \sum_{i=1}^{n} \mid R_\omega \cdot P_i + T - P'_i \mid^2 \tag{3.2}$$

$$= \sum_{i=1}^{n} \Big((x_i cos(\omega) - y_i sin(\omega) + T_x - x'_i)^2 + (x_i sin(\omega) + y_i cos(\omega) + T_y - y'_i)^2 \Big) \tag{3.3}$$

dove ω è l'angolo di rotazione e T=(T_x,T_y) è il vettore di traslazione. Attraverso la minimizzazione di E_{dist} si può ottenere una soluzione in forma chiusa per T_x, T_y e ω, come mostrato qui di seguito:

$$\omega = arctan \frac{S_{xy'} - S_{yx'}}{S_{xx'} + S_{yy'}} \tag{3.4}$$

$$T_x = \bar{x}' - (\bar{x} \cdot cos(\omega) - \bar{y} \cdot sin(\omega)) \tag{3.5}$$

$$T_y = \bar{y}' - (\bar{x} \cdot sin(\omega) + \bar{y} \cdot cos(\omega)) \tag{3.6}$$

dove

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \bar{y} = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} y_i$$

$$\bar{x}' = \frac{1}{n}\sum_{i=1}^{n} x'_i, \qquad \bar{y}' = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} y'_i$$

$$S_{xx'} = \sum_{i=1}^{n}(x_i - \bar{x})(x'_i - \bar{x}'), \quad S_{yy'} = \textstyle\sum_{i=1}^{n}(y_i - \bar{y})(y'_i - \bar{y}')$$

$$S_{xy'} = \sum_{i=1}^{n}(x_i - \bar{x})(y'_i - \bar{y}'), \quad S_{yx'} = \textstyle\sum_{i=1}^{n}(y_i - \bar{y})(x'_i - \bar{x}').$$

La corrispondenza tra i punti della scansione e i punti dell'immagine di riferimento viene effettuata attraverso la scelta del punto più vicino. Ogni punto di S_{new} viene accoppiato con il rispetto punto più vicino di S_{ref}. Da qui il nome di Iterative Closest Point. La distanza tra i punti delle due scansioni viene solitamente calcolata con la formula della distanza lineare euclidea:

$$dist = \sqrt{(x' - x)^2 + (y' - y)^2} \tag{3.7}$$

Si è dimostrato che l'algoritmo di ICP converge sempre monotonicamente ad un minimo locale che rispetta la funzione ai minimi quadrati della distanza. Inoltre se l'angolo di rotazione è piccolo, l'algoritmo di ICP riesce a risolvere con una buona approssimazione il vettore di traslazione. Un inconveniente di questo metodo di scan matching risiede nel fatto che l'algoritmo converge molto lentamente, specialmente se si è in presenza di modelli curvi. Per tale motivo in vari sviluppi di ricerca l'ICP è stato modificato per ottenere migliori prestazioni. Due varianti dell'algoritmo di Iterative Closest Point sono presentate di seguito.

Algoritmo di Iterative Dual Corrispondence (IDC)

Nell'algoritmo di *Iterative Dual Corrispondence (IDC)*, proposto da Feng Lu ed Evangelos Milios[9], la tecnica della corrispondenza fra i punti delle scansioni basata sul closest point viene combinata con un altro metodo di associazione, che prende il nome di *Matching-Range-Point*.

Considerando un punto P e il proprio punto corrispondente $P' = R_\omega P + T$, se si ignora la traslazione si ha che $\mid P' \mid \approx \mid P \mid$. L'angolo polare θ di P e l'angolo polare $\hat{\theta}$ di P' sono correlati dalla relazione $\hat{\theta} \approx \theta + \omega$. Questo implica che il corrispondente di P, sotto effetto della rotazione, è un punto che ha range polare uguale a quello di P, e gli angoli polari dei punti corrispondenti differiscono di un angolo di rotazione ω. Ora, in presenza di piccole traslazioni, ci si può aspettare che il punto P′ con lo stesso range di P sia possibilmente una buona approssimazione del vero punto corrispondente di P. Questa approssimazione restituisce utili informazioni sull'angolo di rotazione ω.
Per essere sicuri che questo metodo trovi un unica reale corrispondenza, si ricerca il punto con lo stesso range all'interno di una regione locale vicina a P dell'immagine di riferimento. Supponendo che si possa stimare una banda B_ω per la rotazione ω (cioè $\mid \omega \mid \leq B_\omega$), si ha che $\hat{\theta} \in [\theta - B_\omega, \theta + B\omega]$. Questo significa che P′ deve ricadere all'interno del settore delimitato da $\theta \pm B_\omega$. Detto ciò la regola di Matching-Range-Point è la seguente: *per un punto P, il suo corrispondente punto sul riferimento è P′ dove P′ soddisfa* $\mid \hat{\theta} - \theta \mid \leq B_\omega$ *e* $\mid P' \mid$ *è il più vicino a* $\mid P \mid$. Basandosi su questa regola è stato sviluppato un algoritmo iterativo, iterative matching-range-point (IRMP), nel quale B_ω controlla la grandezza dell'area in cui ricercare la corrispondenza e anche la massima rotazione possibile risolta in una iterazione. Si genera empiricamente B_ω utilizzando una funzione esponenziale decrescente: $B_\omega(t) = B_\omega(0) \cdot e^{-\alpha t}$.
Questo algoritmo converge notevolmente più velocemente dell'algoritmo di ICP. Conviene perciò combinare le due regole in modo da poter sfruttare sia la velocità di convergenza del matching-range-point sia la stabiltà del closest point. Da questo concetto nasce l'algoritmo di Iterative Dual Corrispondence, che utilizza entrambe le regole nel modo seguente.
Per ogni iterazione occorre fare i seguenti passi:

1. Per ogni punto P_i,

    a. Applicare la regola del closest point per determinare il punto corrispondente P'_i di P_i.

    b. Applicare la regola di matching-range-point per determinare il punto corrispondente P''_i di P_i.

2. Computare la soluzione ai minimi quadrati (ω_1, T_1) dal set di coppie (P_i,P'_i), i = 1,...,n (che sono state ottenute con la regola del closest point).

3. Computare la soluzione ai minimi quadrati (ω_2, T_2) dal set di coppie (P_i,P''_i), i = 1,...,n (che sono state ottenute con la regola del matching-range-point).

4. Prendere (ω_2, T_1) come soluzione per la traslazione nell'iterazione corrente.

La combinazione dei due metodi sembra portare a risultati significativamente migliori di quelli ottenuti con le singole regole. Questo algoritmo non solo migliora la stabilità delle iterazioni, ma incrementa anche notevolmente la velocità di convergenza. Inoltre l'algoritmo IDC risulta insensibile alla scelta del parametro B_ω.

Algoritmo di Scan Matching Metric-Based (MbICP)

La distanza euclidea, calcolata negli algoritmi di ICP e di IDP, non tiene conto del fatto che i punti lontani dal sensore possono essere lontani dai propri corrispondenti a causa delle rotazioni del sensore. Il problema centrale degli algoritmi basati sull'ICP è quello di trovare una misura (per trovare il corrispondente più vicino e applicare la minimizzazione) che in qualche modo prenda in considerazione la traslazione e la rotazione del sensore allo steso tempo. Javier Minguez, Florent Lamiraux e Luis Montesano[10], dell'Università di Saragoza in Spagna, hanno definito una misura della distanza per lo spazio immagine del sensore che tenga conto contemporaneamente della traslazione e della rotazione. Successivamente questa nuova definizione della distanza è stata utilizzata in tutti e due i passi dell'ICP, formalizzando un nuovo algoritmo che prende il nome di *Metric-Based ICP (MbICP)*. La distanza viene definita come *la norma del più piccolo moto rigido di trasformazione che collega un punto ad un altro*. Una trasformazione rigida nel piano è definita dal vettore $q = (x, y, \theta)$ rappresentante la posizione e l'orientamento ($-\pi < \theta < \pi$) del sensore dello scanner nel piano. La norma di q è:

$$\| q \| = \sqrt{x^2 + y^2 + L^2\theta^2} \tag{3.8}$$

dove L è un numero reale positivo omogeneo ad una lunghezza. Dati due punti $p_1 = (p_{1x}, p_{1y})$ e $p_2 = (p_{2x}, p_{2y})$ in $\mathbb{R}^2$, si definisce la distanza fra p_1 e p_2 come:

$$d_p(p_1, p_2) = min\{\parallel q \parallel \text{ tale che } q(p_1) = p_2\} \tag{3.9}$$

dove

$$q(p_1) = \begin{pmatrix} x + cos(\theta)p_{1x} - sin(\theta)p_{1y} \\ y + sin(\theta)p_{1x} + cos(\theta)p_{1y} \end{pmatrix} \tag{3.10}$$

A questo punto si introduce un'approssimazione, valida quando la minima norma della trasformazione è piccola, ipotizzando $\theta = 0$. L'insieme di trasformazioni rigide che soddisfa la condizione $q(p_1) = p_2$ può essere approssimato dall'insieme di soluzioni (x, y, θ) del sistema seguente:

$$\begin{aligned} x + p_{1x} - \theta p_{1y} &= p_{2x} \\ y + \theta p_{1y} + p_{1y} &= p_{2y} \end{aligned}$$

Quello che serve è trovare la soluzione che minimizza la norma di q. Per un dato θ, tale norma è data dall'equazione seguente, dopo aver sostituito ad x ed y le espressioni ricavate dal sistema precedente nella (3.8):

$$\parallel q \parallel^2 = (\delta_x + \theta p_{1y})^2 + (\delta_y - \theta p_{1x})^2 + L^2\theta^2$$

dove $\delta_x = p_{2x} - p_{1x}$ e $\delta_y = p_{2y} - p_{1y}$. Espandendo l'espressione summenzionata si arriva alla conclusione che la distanza tra p_1 e p_2 è approssimata da:

$$d_p(p_1, p_2) = \sqrt{\delta_x^2 + \delta_y^2 - \frac{(\delta_x p_{1y} - \delta_y p_{1x})^2}{p_{1x}^2 + p_{1y}^2 + L^2}} \tag{3.11}$$

Bisogna sottolineare il fatto che questa distanza è minore della distanza euclidea.
La ridefinizione della distanza fa sì che muti considerevolmente anche la tecnica di minimizzazione dell'errore rispetto all'algoritmo di ICP. Dati due punti $P_i = (p_{ix}, p_{iy})$ e $P_i' = (p_{ix}', p_{iy}')$, utilizzando l'approssimazione della distanza 3.11, per la

minimizzazione si ottiene la seguente espressione:

$$E_{dist}(q) = \sum_{i=1}^{n} \left(\delta_{ix}^2 + \delta_{iy}^2 - \frac{(\delta_{ix} p_{iy} - \delta_{iy} p_{ix})^2}{p_{ix}^2 + p_{iy}^2 + L^2} \right) \tag{3.12}$$

dove

$$\begin{aligned} \delta_{ix} &= p'_{ix} - p'_{iy}\theta + x - p_{ix} \\ \delta_{iy} &= p'_{ix}\theta + p'_{iy} + y - p_{ix} \end{aligned}$$

L'espressione 3.12 è una funzione quadratica di tipo wrt:

$$E_{dist}(q) = q^T A q - 2b^T q + c$$

dove c è un numero costante, A la matrice simmetrica

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$$

$$\begin{aligned} a_{11} &= \sum_{i=1}^{n} (p_{ix}^2 + L^2) \\ a_{12} &= \sum_{i=1}^{n} p_{ix} p_{iy} \\ a_{13} &= \sum_{i=1}^{n} (p'_{ix} p_{ix} p_{iy} - p'_{iy}(p_{ix}^2 + L^2)) \\ a_{22} &= \sum_{i=1}^{n} (p_{iy}^2 + L^2) \\ a_{23} &= \sum_{i=1}^{n} (-p'_{iy} p_{ix} p_{iy} + p'_{ix}(p_{iy}^2 + L^2)) \\ a_{33} &= \sum_{i=1}^{n} (p'_{ix}(p_{iy}^2 + L^2) + p'_{iy}(p_{ix}^2 + L^2) - 2 p_{ix} p_{iy} p'_{ix} p'_{iy}) \end{aligned}$$

e

$$b = \begin{pmatrix} \sum_{i=1}^{n}((p'_{ix} - p_{ix})(p_{ix}^2 + L^2) + (p'_{iy} - p_{iy})p_{ix}p_{iy}) \\ \sum_{i=1}^{n}((p'_{ix} - p_{ix})p_{ix}p_{iy} + (p'_{iy} - p_{iy})(p_{iy}^2 + L^2)) \\ \sum_{i=1}^{n}((p'_{ix} - p_{ix})[p'_{ix}p_{ix}p_{iy} - p'_{iy}(p_{ix}^2 + L^2)] + \\ + (p'_{iy} - p_{iy})[-p'_{iy}p_{ix}p_{iy} + p'_{ix}(p_{iy}^2 + L^2)]) \end{pmatrix}$$

Il valore di q che minimizza $E_{dist}(q)$ è il seguente

$$q_{min} = A^{-1}b \tag{3.13}$$

L'algoritmo MbICP, che utilizza le formulazioni appena descritte risulta migliore degli algoritmi descritti precedentemente in termini di robustezza, convergenza e precisione. Gli esperimenti eseguiti con questo tipo di algoritmo rivelano che, a differenza dell'ICP, il MbICP restituisce nella maggior parte dei casi risultati senza falsi negativi. I dati sperimentali hanno anche evidenziato che i risultati migliori si ottengono con L all'incirca uguale a 3.

3.3 Considerazioni finali sugli algoritmi

La trattazione degli algoritmi di scan matching fatta nel paragrafo precedente mette in luce alcune caratteristiche che unificano o differenziano le varie tipologie di matching.

Sicuramente un fattore discriminante, per quello che riguarda i metodi analizzati, è da ricercare nell'esistenza di due tipi di paradigmi: l'uso della correlazione di due insiemi di punti e la minimizzazione di una distanza calcolata dopo aver effettuato un'associazione fra i due insiemi di punti. L'algoritmo che sfrutta le proprietà della trasformata di Hough è un chiaro esempio della prima situazione. In esso, infatti, si ricorre alla correlazione fra gli spettri di Hough dei due insiemi di punti da allineare per ottenere l'angolo di rotazione. Il secondo paradigma, invece, prevede come primo passo un'associazione fra i punti delle due scansioni, secondo un determinato criterio che può variare da algoritmo ad algoritmo, e successivamente la minimizzazione della distanza calcolata fra i punti corrispondenti. Questo principio risiede alla base dell'algoritmo di Cox e dell'algoritmo di Iterative Closest Point con tutte le sue varianti.

Un'altra differenza che è possibile riscontrare è l'uso in alcuni metodi di matching di modelli geometrici ricavati dai punti dati, mentre in altri si fa ricorso direttamente ad un'associazione di tipo punto-punto. Un abbinamento diretto fra i punti delle scansioni viene fatto nell'ICP, nell'IDC e nell'MbICP. Il criterio secondo cui fare questa associazione è solitamente riconducibile al problema del closest point. Come descritto precedentemente, però, la distanza viene calcolata secondo metriche diverse a seconda di quale algoritmo si stia adoperando. Modelli geometrici sono invece utilizzati sia nel metodo ideato da Cox sia nell'Hough scan matching. Nel primo i punti della scansione di riferimento sono uniti a formare segmenti che rappresentino i tratti caratteristici del modello, nel secondo si ricorre ad un uso pesante di rette parametriche e di spazio dei parametri.
Un'ulteriore fattore discriminante riguarda il metodo di ricerca della trasformazione rigida che possa sovrapporre le scansioni. In alcuni casi si procede avanzando prima ipotesi sull'angolo di rotazione e successivamente alla luce di queste supposizioni si ricava il valore della traslazione. È questo sicuramente il caso dell'ICP in cui i valori del vettore di traslazione sono calcolati utilizzando l'angolo di rotazione precedentemente trovato. Identica è la situazione per l'IDC che sfrutta direttamente l'algoritmo di iterative closest point, anche se di questo poi viene presa in considerazione solo la traslazione. Differentemente da quanto è stato appena detto alcune tecniche, come il Metric-based ICP, effettuano una ricerca della rotazione simultanea a quella della traslazione. Per ottenere ciò nell'MbICP è stato necessario addirittura introdurre una nuova metrica in modo che nel calcolo della distanza si tenesse conto non solo della traslazione fra le due scansioni ma anche dell'angolo di rotazione.
Non esiste un criterio generale per stabilire se un algoritmo di scan matching sia migliore di un altro, è possibile che a seconda delle situazioni in cui si lavora siano adatte tecniche differenti. Di quelli presentati sicuramente poco utilizzato negli ultimi anni è il metodo sviluppato da Cox, ormai superato, ma esso ha una propria importanza essendo, come già detto, una sorta di precursore delle tecniche moderne. L'ICP è senza dubbio una delle tecniche maggiormente adoperate per il matching fra due scansioni e quindi si sono raggiunti buoni risultati nella sua applicazione. Bisogna però evidenziare che una stima relativamente precisa del vettore di traslazione si ottiene solo in presenza di angoli di rotazione piccoli, perciò è consigliabile usare l'ICP in questo tipo di situazioni. Un altro difetto di questo al-

goritmo potrebbe risiedere nella velocità non troppo elevata di convergenza. Questo problema viene risolto nell'algoritmo MbICP attraverso la ricerca simultanea della rotazione e della traslazione. Ciò che però si guadagna in velocità con questo metodo si perde in facilità di calcolo. L'introduzione di una nuova metrica complica notevolmente la computazione della distanza fra i punti delle due scansioni. Inoltre non si capisce in che modo calcolare la stima ideale per il parametro L della nuova distanza (si veda (3.8)) a seconda delle situazioni in cui ci si trova. Analizzando la ricerca effettuata nel corso degli anni si può dire che la tecnica più accreditata sia l'utilizzo della minimizzazione dell'errore della distanza e il paradigma ICP. Ciò è dovuto proprio alla diffusione del suo utilizzo e quindi ai numerosi passi in avanti fati sotto il profilo dell'efficienza e della robustezza. Per questo motivo nella realizzazione della libreria, come verrà descritto nel capitolo 4, si è scelto di implementare principalmente un algoritmo di tipo Iterative Closest Point.

Capitolo 4

Progettazione dell'applicazione

4.1 Obiettivi

Questo lavoro di tesi si prefigge come obiettivo finale lo sviluppo di un software che gestisca un algoritmo di *Scan Matching* mediante l'uso del linguaggio di programmazione ad oggetti *C++*. Ciò che si vuole ottenere dalla realizzazione di questo progetto è uno strumento che sia in grado di stabilire, date due scansioni, per quale angolo di rotazione e per quale vettore di traslazione si debba ruotare e traslare una delle scansioni per poter avere una parziale o totale sovrapposizione con l'altra con una buona approssimazione. Come trattato nel capitolo 3 questo problema riveste un ruolo importantissimo nell'uso dei sensori in robotica, per quanto riguarda il tracciamento dell'odometria, la localizzazione e la possibilità di ricostruire profili e mappe.
Prima di passare all'analisi dettagliata dello sviluppo del progetto, occorre innanzitutto fare alcune considerazioni su quale algoritmo di Scan Matching sia stato opportuno implementare. Le opzioni tra cui scegliere (si veda per questo l'analisi svolta nel paragrafo 3.2) erano numerose. Dato la natura esplorativa di questo lavoro di tesi e la necessità di procedere in modo incrementale, si è preferito puntare su di un algoritmo consolidato e di ampia diffusione come l'ICP con alcune sue varianti.
La prima variante è quella sviluppata nel 1995 da Feng Lu ed Evangelos Milios dell'Università di North York, in Canada, e che prende il nome di *Iterative Dual Correspondence - IDC*. La seconda variante presa in esame è quella proposta nel

2005 da Javier Minguez, Florent Lamiraux e Luis Montesano dell'Università di Saragoza in Spagna nell'articolo *Metric-Based Scan Matching Algorithms for Mobile Robobt Displacement Estimation*, che cambia nettamente prospettiva rispetto alla precedente grazie all'utilizzo, non più della metrica euclidea ma di una nuova metrica creata appositamente per valutare efficacemente l'entità della rotazione. La giustificazione matematica dei due algoritmi è stata presentata nel paragrafo 3.2.3. Il punto di forza di questo software risiede proprio nella possibilità di scegliere quale delle due varianti utilizzare per il calcolo dello Scan Matching. Infatti, attraverso l'uso di tecniche e strumenti, che saranno approfonditi nei paragrafi successivi, si è reso il programma flessibile, cioè in grado di supportare il calcolo dell'algoritmo sia attraverso la metrica euclidea, sia attraverso la metrica adottata da Minguez.
Mostrate fino ad ora le finalità prime che si prefigge questo tipo di progetto, è venuto il momento di elencare alcune specifiche che il software dovrebbe essere in grado di supportare.
Come già accennato, un requisito fondamentale è quello della flessibilità. Il programma dovrebbe essere in grado non solo di poter consentire la scelta del tipo di algoritmo da utilizzare per il calcolo del closest point, ma anche di consentire la gestione dei *tipi* (in particolare dei punti ricavati dalle scansioni) e delle strutture dati.
Un'altra specifica da soddisfare è senza dubbio quella riguardante l'efficienza e la portabilità in vista delle esecuzioni in tempo reale. Il software deve poter consentire un calcolo veloce ed efficace, evitando strutture pesanti che rallentino l'esecuzione a causa delle numerose iterazioni. Un robot in movimento,per esempio, non può permettersi di perdere tempo ogni volta che deve ricalcolare la sua localizzazione o ricreare una mappa di ciò che lo circonda. Per ottenere questi obiettivi si è dovuto fare ricorso ad alcuni compromessi tra l'efficacia e l'efficienza degli strumenti utilizzati. Riportando un esempio concreto, una delle scelte più difficili sotto questo punto di vista è stata quella di decidere quale tipo di struttura utilizzare per la memorizzazione e la gestione dei dati rilevati. In un primo istante si è presa in considerazione la possibilità di immagazzinare i dati in una struttura di tipo *KD-Tree*[11]. Tale container risulta molto efficace, consente uno scorrimento e un ordinamento dei dati notevolmente veloce. Purtroppo abbinata a queste caratteristiche vi è anche una certa difficoltà di implementazione. Ciò ha fatto sì che si

scegliessero contenitori più semplici, come quelli della standard library[12], sia per mantenere una certa facilità di utilizzo della libreria, sia per ottenere risultati concreti in un minor tempo. Altre semplificazioni, a discapito dell'efficienza, sono state effettuate all'interno dell'algoritmo vero e proprio. Ad esempio, le distanze adoperate nella fase di associazione tra le scansioni sono sempre state calcolate tra due punti. Spesso, invece, nello sviluppo teorico degli algoritmi queste distanze sono calcolate tra un punto ed una linea spezzata. Questa differenza potrebbe portare ad errori nell'approssimazione.

4.2 Struttura del progetto

4.2.1 Classi sviluppate

Nel progettare un algoritmo la complessità non risiede nel design delle classi ma nelle classi interne. Il compito del design in questo caso è gestire la caratteristica della flessibilità. Si è voluto puntare in questo modo alla realizzazione di un'interfaccia che risulti il più leggero possibile e che sia facilmente maneggiabile da ogni utente che intenda servirsene. Bisogna ricordare, come già trattato nel paragrafo 4.1, che l'utilità di questo progetto non è fine a se stessa, ma pensata in funzione di applicazioni nel campo della robotica e di sviluppi real-time.
La struttura dell'intero software può essere sintetizzata attraverso i diagrammi delle classi rappresentati in figura 4.1 e 4.2:

Le classi astratte sono sostanzialmente tre :

- *ScanFactory*
- *ScanSet*
- *ScanMatcher*

La classe `ScanSet` definisce al suo interno la struttura dati in cui i punti di un'eventuale scansione possono essere immagazzinati. Il ricorso ad un'interfaccia demanda alle singole istanze il compito di definire il contenitore. Nello specifico, si possono scegliere che tipo di punto il contenitore debba immagazzinare e che tipo di struttura (ad esempio un vettore, una lista, etc.) si debba utilizzare per

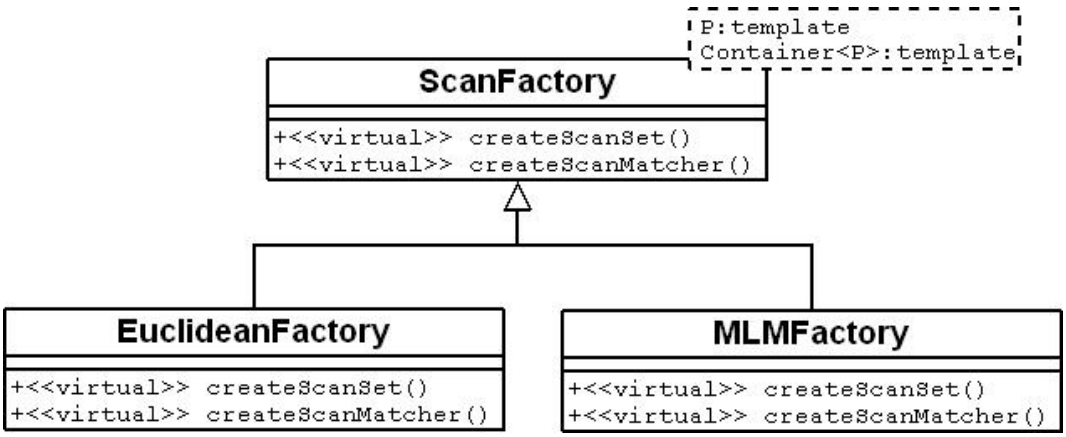

Figura 4.1: Diagramma delle classi delle factory.

l'immagazzinamento. Le modalità attraverso le quali ottenere questo polimorfismo saranno trattate più in profondità nel paragrafo 4.2.2.
Sono state previste le implementazioni `EuclideanSet` e `MLMSet` ereditano dalla classe ScanSet e fondamentalmente non fanno altro che ridefinire un metodo della classe padre. In EuclideanSet viene implementato il metodo che restituisce la distanza fra due punti calcolata con la formula euclidea, mentre in MLMSet viene implementato il metodo che restituisce la distanza fra due punti con la formula ideata da Minguez, Lamiraux e Montesano (si veda per questo il paragrafo 3.2.3).

La classe `ScanMatcher` è la classe che si assume il ruolo più delicato dell'intero progetto. Ha come compito principale, infatti, quello di sviluppare ed eseguire l'algoritmo di scan matching. Risiedono all'interno di detta classe i metodi che si propongono di calcolare l'associazione tra i punti di due scansioni secondo il criterio del closest point, di minimizzare l'errore di approssimazione nel calcolo dell'angolo di rotazione e del vettore di traslazione, e di reiterare i passi previsti dall'algoritmo di ScanMatching. Il cuore della classe ScanMatcher è il metodo *match()*, la funzione che implementa e coordina l'esecuzione dell'algoritmo e che restituisce quelli che dovrebbero essere l'angolo di rotazione e il vettore di traslazione finali e definitivi.
Anche in questo caso, come per ScanSet, la classe è stata sviluppata in modo tale

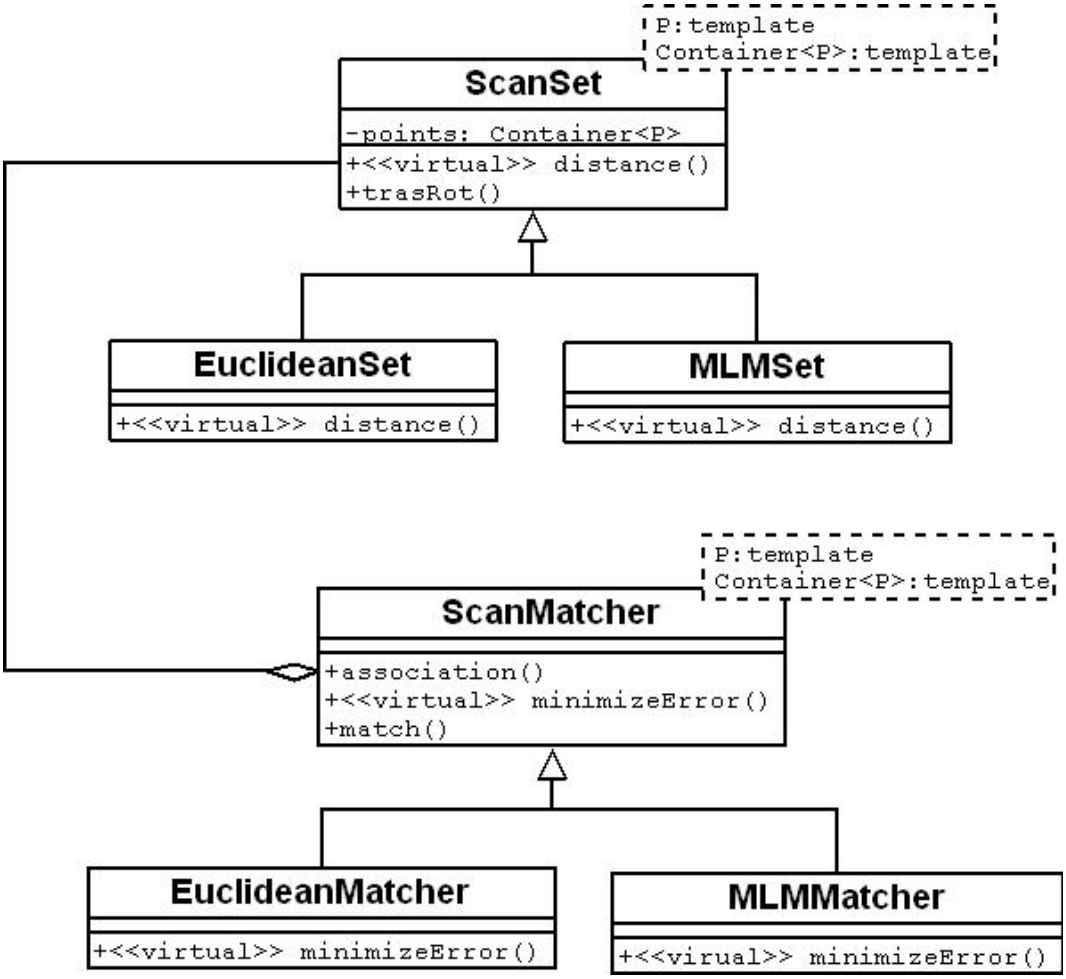

Figura 4.2: Diagramma delle classi ScanSet e ScanMatcher.

da garantire una certa flessibilità, lascia, cioè, la possibilità di applicare l'algoritmo su vari tipi di punti e non limita l'uso del programma ad un modello predefinito. Analogamente allo ScanSet, dallo ScanMatcher ereditano due classi: `EuclideanMatcher` e `MLMMatcher`. Queste due classi ridefiniscono il metodo *minimizeError()* per la classe padre. EuclideanMatcher implementa la minimizzazione dell'errore di approssimazione secondo il modello definito dall'algoritmo di IDC, mentre MLMMatcher implementa tale approssimazione con la tecnica descritta da Minguez, Lamiraux e Montesano nel loro articolo (paragrafo 3.2.3).

La classe `ScanFactory` ha solo un semplicissimo ruolo, quello di creare oggetti di tipo ScanSet e oggetti di tipo ScanMatcher e fa riferimento al pattern *Abstract-Factory* (si veda per questo il paragrafo 4.2.2). Gli unici metodi implementati all'interno di questa classe sono i metodi *createScanSet()* e *createScanMatcher()*. Entrambi ritornano un puntatore ad un nuovo oggetto, di tipo ScanSet o ScanMatcher a seconda del caso, che l'utente può poi utilizzare per l'immagazzinamento dei dati e per l'esecuzione dell'algoritmo.
Come ormai si può immaginare, esistono due classi figlio di ScanFactory: la classe `EuclideanFactory`, che definisce i metodi per la creazione di oggetti di tipo EuclideanSet e oggetti di tipo EuclideanMatcher, e la classe `MLMFactory`, che definisce i metodi per la creazione di oggetti di tipo MLMSet e oggetti di tipo MLM-Matcher.

Ai fini dello sviluppo del software per la gestione dell'algoritmo di Scan Matching le classi precedentemente descritte sono sufficienti. Per motivi di comodità e con uno sguardo all'immediata applicazione pratica con lo scanner laser SICK LMS 200, si è preferito implementare altre due semplici classi:

- `Point`, che definisce un punto nello spazio bidimensionale e fornisce i metodi per settare le coordinate X e Y del punto;
- `Insertion`, il cui compito è quello di immagazzinare i dati contenuti in un semplice file di testo all'interno di un contenitore di tipo ScanSet.

Viene comunque lasciata aperta la possibilità all'utente finale di utilizzare altre classi per rappresentare Point. Vi sono però dei vincoli che assolutamente devono essere rispettati: poichè l'algoritmo funziona solo nello spazio bidimensionale, il tipo di punto che si vuole definire deve avere solo due dimensioni; inoltre all'interno della classe Punto devono essere ridefiniti l'operatore di assegnamento, "operator=()", e l'operatore di sottoscrizione, "operator[]()", essendo largamente utilizzati all'interno del programma.

4.2.2 Strumenti utilizzati

Durante la fase di analisi generale riguardante le tecniche di Scan Matching si è potuto constatare la presenza di una grande varietà di esse. Nella realizzazione di questo progetto si è voluto prendere in esame solo due di queste tecniche, ma non è detto che quelle scelte siano le migliori e le più accurate fra tutte. In eventuali applicazioni future potrebbe nascere la necessità, poichè magari rivelatesi maggiormente precise o con minore complessità computazionale, di adottare altre metodologie di calcolo dello Scan Matching oltre o in sostituzione alle due già implementate. Inoltre, come evidenziato nel paragrafo precedente, il grado di libertà non, si limita alla scelta della tecnica di associazione fra le scansioni, ma si estende alla possibilità di decidere che genere di contenitore utilizzare e su quale tipo di dato svolgere l'algoritmo. La struttura e gli strumenti utilizzati per lo sviluppo del software sono stati appositamente selezionati per dare un carattere di flessibilità al progetto e per permettere l'uso di numerose tecniche di matching.
Lo strumento per gestire la molteplicità delle soluzioni e delle scelte possibili risiede nel costrutto *template*, supportato dal linguaggio di programmazione ad oggetti C++. Tale costrutto è il fondamento della cosiddetta *programmazione generica*. La programmazione generica è quel paradigma di programmazione che si fonda sulla costruzione dei tipi definendone i requisiti fondamentali e sulla definizione di algoritmi i quali operano su tipi che soddisfino tali requisiti[13].
Tutte le classi realizzate nel progetto, ad eccezione delle classi Point e Insertion che, come descritto nel paragrafo 4.2.1, rivestono un ruolo secondario, sono state implementate utilizzando la tecnica dei template. Per facilitare la comprensione dell'uso di tale costrutto è bene osservarne un esempio.

Nel listato 4.1 è mostrata la definizione di una tipica classe template e il codice da editare per creare un oggetto appartenente a tale classe. Si può notare che nell'istanza che genera l'oggetto Template `newTemplate` è necessario specificare i tipi `P` e `Container` presenti nella definizione della classe template. Ogni qualvolta si desideri creare un nuovo oggetto è possibile specificare tipi differenti, creando così una varietà di oggetti appartenenti alla stessa classe ma con caratteristiche diverse fra loro. Vi sono chiaramente poi delle limitazioni nell'uso dei template. I tipi che si specificano devono essere compatibili con la classe, devono pertanto essere costruiti

```
template<class P, template<class> class Container>
class Template
{
    public:
    //...

    private:
    //...
};


Template<P, Container<P> newTemplate;
```

Listato 4.1: Esempio di classi template.

in modo tale da poter essere utilizzati dai metodi della classe template (per esempio potrebbero dover avere definiti al loro interno particolari operatori). Il controllo di questi vincoli avviene chiaramente nel processo di compilazione. La possibilità di scelta fra varie categorie di punti su cui operare lo scan matching e fra varie categorie di contenitori in cui memorizzare i punti è stata ottenuta definendo la classe `ScanSet` come una classe template del tutto similare a quella riportata nell'esempio.

Il polimorfismo della struttura oltre che all'uso del costrutto template è dovuto al concetto di *Ereditarietà* proprio della programmazione orientata agli oggetti. L'Ereditarietà si fonda sulla presenza di un'interfaccia base dichiarata in modo esplicito e sull'implementazione di classi da essa derivate. Attraverso l'uso di uno speciale specificatore, `virtual`, si rende possibile allo sviluppatore ridefinire alcuni metodi della classe base all'interno delle classi derivate. La sintassi del costrutto appena presentato viene mostrata nel listato 4.2.

Grazie all'uso dello specificatore `virtual` la classe Son al suo interno potrà ridefinire il metodo Method della classe base Father, facendo eseguire in questo modo due cose diverse a seconda se ci si trovi al cospetto del metodo appartenente al figlio o del metodo appartenente al padre. Questo tipo di tecnica è stata adottata per creare le dipendenze fra la classe `ScanSet` e le classi `EuclideanSet`

```
class Father
{
    public:
    //...

    virtual void Method(){
        //...
        make Something;
    }

    private:
    //...
};


class Son : public Father
{
    //...
    virtual void Method() {
        //...
        make SomethingDifferent;
    }
};
```

Listato 4.2: Esempio dell'uso dello specificatore virtual.

e `MLMSet` con la ridefinizione del metodo per calcolare la distanza fra due punti. Analogamente si è ottenuto lo stesso effetto tra la classe `ScanMatcher` e le classi `EuclideanMatcher` e `MLMMatcher` con la ridefinizione del metodo per il calcolo dell'approssimazione dell'errore.

Un altro aspetto importante, per quanto riguarda gli strumenti adoperati, risiede nell'utilizzo delle `factories`. Una factory è una semplice interfaccia che implementa alcuni metodi finalizzati alla creazione di oggetti. Per riportare un esempio, la classe `ScanFactory` possiede al suo interno i metodi *createScanSet* e *createScanMatcher* che ritornano rispettivamente un puntatore ad un oggetto ScanSet ed un puntatore ad un oggetto ScanMatcher. Per rispecchiare la simmetria con le

altre due classi principali della struttura, anche la classe `AbstractFactory` possiede due classi figlio che prendono il nome di `EuclideanFactory` e di `MLMFactory`. Come si può facilmente intuire la prima gestisce la creazione di oggetti EuclideanSet ed EuclideanMatcher, mentre la seconda provvede alla creazione di oggetti MLMSet e MLMMatcher.
Sono stati più volte presentati, in questo capitolo, il polimorfismo e la simmetria caratterizzanti la struttura delle classi del software. Si è volutamente dato questo tipo di caratteristiche al progetto anche in vista di un possibile sviluppo futuro che impieghi l'uso delle cosiddette *Policies*. Il concetto che risiede alla base di questo strumento è quello di decomporre una classe, che può svolgere il proprio compito in varie modalità diverse, in diverse "politiche". Una politica, allora, in poche parole definisce un'interfaccia astratta che può eseguire i propri metodi in vari modi. Queste modalità saranno poi implementate nelle istanze dell'interfaccia, dette *policy classes* [13].

4.3 Applicazioni pratiche e testing

In questo paragrafo vengono presentati i risultati dei test effettuati con la libreria implementata. L'obiettivo di tali esperimenti è stato quello di verificare il corretto funzionamento dell'algoritmo e di valutare la precisione dei risultati ottenuti da esso.
Innanzitutto è stato necessario acquisire tramite il dispositivo laser (capitolo 2) delle scansioni sulle quali applicare l'algoritmo. Tale operazione è stata eseguita attraverso l'uso di un semplice programma di test in C++ che permette di effettuare una scansione e di memorizzare i dati raccolti in un file di testo. In tutti i test la risoluzione del dispositivo laser è stata impostata a mezzo grado e si è utilizzato un range angolare di $180\circ$. In ogni file di testo sono stati memorizzati perciò 361 dati sotto forma di coordinate cartesiane.
Si è scelto di acquisire le informazioni in un ambiente che avesse un profilo semplice e di facile riconoscimento in modo da non introdurre un certo livello di incertezza o di complessità. Pertanto i rilevamenti sono stati effettuati nel corridoio della palazzina 1 della sede scientifica di Ingegneria collegando lo scanner laser ad un computer portatile e posizionando entrambi su di un sostegno mobile. Lo spazio in questione,

infatti, presenta tratti caratteristici facilmente rilevabili, quali le due pareti parallele fra loro e i colonnotti. Anche le dimensioni si adattano particolarmente all'uso dello scanner laser, presentando una lunghezza di circa 8 metri e una larghezza di circa 2 metri. Come è gia stato descritto, in questo range di misure il laser presenta una precisione dell'ordine del millimetro. Tutto ciò risulta visibile in figura 4.3, la quale riporta un esempio delle scansioni. Non essendo stato realizzato in questo lavoro di tesi un software che visualizzasse le rappresentazioni grafiche delle scansioni, esse sono state ottenute attraverso l'implementazione di un semplice script eseguito con il software Matlab version 6.5 R 13 per piattaforma Windows. Nelle circostanze de-

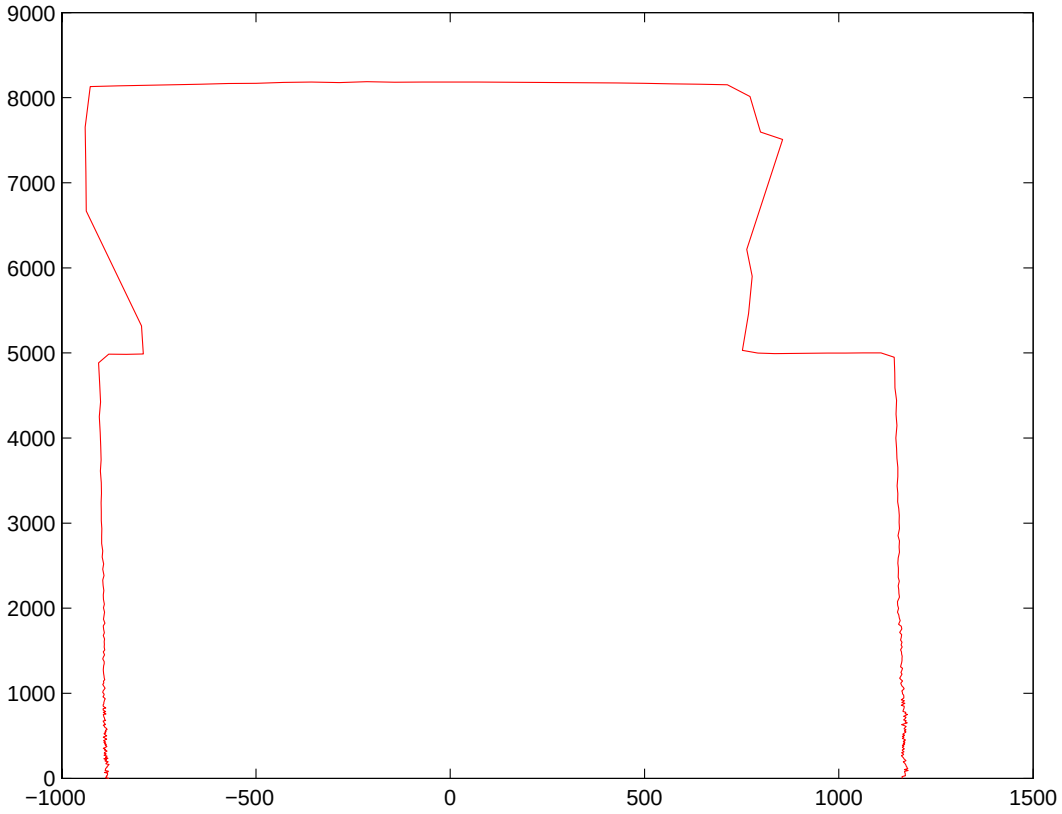

Figura 4.3: Scansione del corridoio della palazzina 1.

scritte precedentemente sono state prese numerose rilevazioni, ognuna delle quali differente dalle altre per angolo di rotazione, per traslazione o per entrambe.
Eseguita questa fase di acquisizione si è proceduto con la fase di analisi dell'algorit-

mo. La verifica del corretto funzionamento della libreria è stata effettuata off-line, poichè i tempi di esecuzione non permettono ancora il suo utilizzo in applicazioni di tipo real-time. Nei test è stata principalmente verificata la correttezza dei dati risultanti dall'utilizzo dell'algoritmo di Iterative Closest Point. Questo perchè, data la diffusione di tale tecnica, si è rivelato maggiormente semplice ricavare dati di confronto su i quali basare le valutazioni.
I risultati sui quali fondare la verifica della correttezza del programma sono stati ricavati sulla base di una stima manuale e attraverso l'esecuzione di uno script Matlab che simula il comportamento dell'ICP. Il software progettato in questo lavoro di tesi si è dimostrato abbastanza preciso ed attendibile per quanto riguarda l'angolo di rotazione. Come si può osservare dalla tabella 4.1 i valori dell'angolo θ non si discostano in maniera rilevante dai dati sulla rotazione ottenuti attraverso la stima manuale e l'esecuzione dello script Matlab. Tutt'altro genere di risultati si otten-

	Manuale	**Matlab**	**Libreria**
Test	θ	θ	θ
Test 1-2	$1°$	$0.56°$	$1.19°$
Test 2-3	$26°$	$21°$	$31°$
Test 3-4	$25.74°$	$19°$	$25.59°$
Test 4-5	$45.55°$	$52°$	$42.21°$
Test 5-6	$44°$	$41.13°$	$41.34°$
Test 6-7	$35°$	$37.62°$	$37.57°$

Tabella 4.1: Risultati ottenuti sull'angolo di rotazione.

gono per quello che riguarda la traslazione. Essi, infatti, sia nel caso dello script Matlab sia nel caso della libreria implementata, differiscono notevolmente da quelli riscontrati con la stima manuale, come è visibile nella tabella 4.2. Sotto questo punto di vista, quindi, il software risulta essere alquanto inadeguato. Il problema più che a livello di implementazione del codice potrebbe risiedere nel meccanismo stesso dell'algoritmo ICP. Questa ipotesi sembrerebbe essere confermata dal fatto che anche un'ulteriore implementazione in C++ dell'algoritmo di ICP, trovata in rete [14], porta a risultati errati del vettore di traslazione.
Per ottenere una buona precisione in entrambe le componenti della trasformazione rigida è probabilmente più opportuno scegliere di implementare algoritmi basati

	Manuale		**Matlab**		**Libreria**	
Test	x	y	x	y	x	y
Test 1-2	270	1310	329	18	716	31
Test 2-3	3.16	5	1170	221	592	683
Test 3-4	0.55	0.43	1089	64	90	451
Test 4-5	-380	800	-1613	1489	-780	820
Test 5-6	-400	-500	-1880	-181	-1106	-152
Test 6-7	50	50	-1476	693	1580	897

Tabella 4.2: Risultati ottenuti sul vettore di traslazione.

sull'ICP ma maggiormente evoluti, come ad esempio l'algoritmo di Iterative Dual Corrispondence.

Capitolo 5

Conclusioni

Questo lavoro di tesi ha previsto la valutazione delle prestazioni di un dispositivo laser e la realizzazione di una libreria per l'esecuzione di un algoritmo di scan matching.
La fase iniziale del progetto è consistita nell'analisi del dispositivo sensoriale scanner laser *Sick LMS 200*. Questo tipo di sensore si contraddistingue dalle altre tipologie di dispositivi sensoriali per l'elevata precisione e risoluzione. Ciò ha portato ad una ricostruzione dell'errore di precisione del sensore nella quale si è potuto constatare un andamento di tipo gaussiano e una concordanza con i principali modelli proposti in letteratura.
La fase più importante e maggiormente impegnativa del progetto di tesi è stata la realizzazione di un software in grado di eseguire un algoritmo di *scan matching* fra due insiemi di punti. Per scan matching si intende la ricerca di una trasformazione rigida attraverso la quale due insiemi di punti possano essere sovrapposti. Si tratta di un problema alla base di molte applicazioni della robotica. Infatti tramite il confronto fra le scansioni rilevate da un sensore applicato su di un robot e concorde con esso si può riuscire a ricostruire la posizione ed il moto del robot stesso o addirittura a ricostruire una mappa dell'ambiente circostante. La caratteristica fondamentale della libreria realizzata risiede nella flessibilità e nella capacità di gestire possibili differenti tecniche di matching grazie all'utilizzo dello strumento *template* e dell'ereditarietà propri della programmazione ad oggetti in C++. Il software così realizzato si rivela estendibile lasciando la possibilità di definire altri algoritmi di

scan matching oltre a quelli implementati. In lettura è proposta una larga varietà di algoritmi di scan matching, dei quali i più conosciuti ed adoperati sono stati analizzati in parte in una fase di ricerca del progetto di tesi. Di questi si è scelto di implementare l'algoritmo di Iterative Closest Point e l'algoritmo Metric-based ICP, variante del precedente. La scelta è stata motivata dalla larga diffusione dell'uso dell'ICP nella sovrapposizione delle immagini e quindi dalla affidabilità raggiunta nel suo utilizzo. I risultati ottenuti hanno evidenziato una buona precisione dell'algoritmo per quanto riguarda la fase di rotazione della trasformazione rigida. Riguardo invece alla traslazione i risultati non sono altrettanto affidabili poichè si discostano largamente dai valori necessari per sovrapporre le due scansioni. Per questo motivo un eventuale sviluppo futuro dovrebbe prevedere l'implementazione di un algoritmo efficiente sia sulla rotazione che sulla traslazione.
Inoltre rendendo sufficientemente leggera e snella la struttura, una possibile applicazione futura di questa libreria riguarda l'utilizzo di essa in sistemi di tipo realtime per, ad esempio, il controllo della navigazione di un robot, la localizzazione o la ricostruzione di una mappa dell'ambiente.

Bibliografia

[1] H.R. Everett e L. Feng J. Borestein. *Where am I?, Sensors and Methods for Mobile Robot Positioning*. 1996.

[2] Documentazione laser sick lms 200. http://www.sick.com.

[3] Stefano Zanero e Claudio Merloni. Driver per laser sick 200 lms. http://www.elet.polimi.it/upload/zanero/soft.htm.

[4] Wolfram Burgard e Dieter Fox Sebastian Thrun. *Probabilistic in robotics*. MIT Press, 2005.

[5] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[6] Feng Lu ed Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 1997.

[7] Luca Iocchi e Giorgio Grisetti Andrea Censi. Scan matching in the hough domain. *IEEE Procedings of International Conference on Robotics and Automation*, 2005.

[8] Ingemar j. Cox. Blanche: Position estimation for an autonomous robot vehicle. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 1989.

[9] Feng Lu ed Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 1997.

[10] Florent Lamiraux e Luis Montesano Javier Minguez. Metric-based scan matching algorithms for mobile robot displacement estimation. citeseer.ist.psu.edu/minguez05metricbased.html, 2005.

[11] Jon Louis Bentley. K-d trees for semidynamic point sets. *Proceedings of the sixth annual symposium on Computational geometry*, 1990.

[12] Nicolai M. Josuttis. *The C++ Standard Library - A Tutorial and Reference*. Addison-Wesley.

[13] Andrei Alexandrescu. *Modern C++ Design*. Addison-Wesley.

[14] Tim Bailey. Codice sorgente di un'implementazione dell'algoritmo di icp in c++ e matlab. http://www.acfr.usyd.edu.au/homepages/academic/tbailey/software/index.%html.

[15] Jens-Steffen Gutmann e Christian Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. 1996.