UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

Localizzazione di Monte Carlo per Robot Mobili dotati di Visione Omnidirezionale

Relatore: Chiar.mo Prof. *Enrico Pagello*

Correlatore: Dr. *Emanuele Menegatti*

Laureando: *Mauro Zoccarato*

Corso di Laurea in Ingegneria Informatica

Dipartimento di Ingegneria dell'Informazione

Anno Accademico 2001/2002

Ai miei genitori, a mia sorella

Ringraziamenti

Desidero ringraziare sentitamente il mio correlatore Dr. Emanuele Menegatti per il contributo scientifico con il quale mi ha guidato nella scrittura di questa tesi e per la stima che mi ha sempre dimostrato.
Un ringraziamento particolare va al Professor Enrico Pagello per gli utili consigli e suggerimenti che mi ha dato nello sviluppo della tesi.
Vorrei ringraziare entrambi per l'amicizia e la pazienza con le quali mi hanno seguito in questi mesi di lavoro.

Indice

Sommario

Il problema della localizzazione è di fondamentale importanza nella robotica mobile. La tesi affronta il problema della localizzazione di robot mobili dotati di un sistema di visione omnidirezionale. La localizzazione viene affrontata secondo un approccio probabilistico con il Metodo di Monte Carlo. Il nostro obiettivo è quello di fondere due approcci noti: la localizzazione basata sulle immagini e la localizzazione di Monte Carlo. L'utilizzo di un sistema di visione omnidirezionale, un sistema di caratterizzazione delle immagini basato sulla Trasformata di Fourier e del Metodo di Monte Carlo ci permette di superare i limiti che ciascuna delle due tecniche presenta se considerata singolarmente: il fallimento della localizzazione basata sulle immagini nel caso di "perceptual aliasing" dell'ambiente e la necessità della mappa geometrica dell'ambiente di navigazione per la localizzazione di Monte Carlo. Proponiamo inoltre una tecnica per rilocalizzare il robot dopo un fallimento della localizzazione utilizzando le informazioni derivanti dal nostro sistema di visione. Questa tecnica, oltre a risolvere con successo il problema della rilocalizzazione migliora la localizzazione anche durante il normale funzionamento. Presentiamo gli esperimenti eseguiti per testare il nostro sistema con dati acquisiti da un robot reale in due ambienti indoor, un piccolo laboratorio e un lungo corridoio.

Capitolo 1

Introduzione

La tesi affronta un problema fondamentale della robotica mobile che viene tipicamente indicato come *localizzazione basata sulle informazioni sensoriali*. Il problema della localizzazione è stato riconosciuto come "il problema fondamentale della robotica mobile e uno dei requisiti essenziali per fornire il robot di capacità autonome" [Cox e Wilfong, 1990; Cox, 1991]. Si pensi ad un robot mobile, dotato di un sistema di sensori, che deve muoversi autonomamente in un ambiente ed eseguire dei compiti. Per essere in grado di portare a termine i "task" che gli sono stati assegnati deve conoscere la propria posizione nell'ambiente, con un sufficiente livello di confidenza. Deve quindi essere in grado di acquisire, elaborare e interpretare le informazioni che ottiene dai sensori al fine di ottenere la rappresentazione del proprio stato nel modello del mondo che ha a disposizione [Borghi, 1997]. Il modello del mondo può essere una mappa metrica dell'ambiente, ad esempio una planimetria CAD; oppure può essere una rappresentazione topologica dell'ambiente che descrive solo le connessioni tra le zone dell'ambiente, senza informazioni metriche; infine può essere una mappa metrica semplice dell'ambiente, che descrive solo l'area a disposizione del robot e fornisce un sistema di coordinate cartesiane. A seconda di qual'è il modello del mondo a disposizione del robot, la sua rappresentazione dello stato, cioè la sua posizione nell'ambiente viene indicata in modo differente: con un insieme di coordinate in un sistema cartesiano nel caso di mappa metrica; con l'indicazione di un'area in cui si può trovare il robot, nel caso di mappa topologica.

In questa tesi facciamo riferimento al problema della localizzazione per un robot mobile che deve navigare in un ambiente dinamico: secondo la definizione data in [Russell e Norving, 1995], "l'ambiente è dinamico per un robot quando l'ambiente può cambiare mentre tale robot sta deliberando"[1]; questo

[1]La deliberazione è l'atto di "decidere su un'azione" [Russell e Norving, 1995].

succede per la presenza di altri soggetti, robot o persone, che si muovono nell'ambiente ed eventualmente lo modificano. Il robot a cui ci riferiamo è un robot olonomo, cioè in grado di muoversi in qualsiasi direzione senza dover prima ruotare; inoltre il sistema sensoriale principale è la visione, in particolare un sistema di visione *omnidirezionale*: esso fornisce una visione dell'ambiente (che chiamiamo anche scena) a 360° con un'unica immagine. Altro sensore considerato è l'odometro che, misurando le rotazioni delle ruote, fornisce una stima del movimento effettuato dal robot. La stima di posizione fornita dall'odometro è relativa, cioè espressa rispetto alla stima precedente; inoltre è approssimata, per la presenza di errori, data la semplicità del sensore.

I problemi classici della localizzazione sono tre: il "*position tracking*" in cui il sistema di localizzazione conosce la posizione iniziale del robot e deve mantenere la conoscenza della posizione durante la navigazione, correggendo gli errori presenti nei dati di odometria. La *localizzazione globale* in cui il sistema non conosce la posizione iniziale del robot e deve riuscire a determinarla dal nulla, elaborando le sole informazioni acquisite dai sensori durante la navigazione. Il terzo e più difficile dei problemi è quello chiamato generalmente *problema del robot rapito*[2]: qui il robot viene spostato senza che il sistema di localizzazione ne abbia conoscenza, quindi continua a supporre di essere nella posizione precedente e deve essere in grado di accorgersi che la posizione è un'altra.

Data l'incertezza legata all'interpretazione dei dati sensoriali, il robot può conoscere la propria posizione con un certo livello di confidenza, cioè può conoscere solo una densità di probabilità sull'insieme degli stati. Questa densità viene chiamata "*belief*" [Russell e Norving, 1995] e viene aggiornata ogni volta che il robot acquisisce nuovi dati dai sensori[3].
Da quanto detto risulta naturale affrontare il problema della localizzazione da un punto di vista probabilistico e considerarlo come un problema di stima dello stato di un sistema dinamico, a partire da informazioni misurate da sensori, per loro natura affette da errori e disturbi. Per questo può essere utilizzato l'approccio chiamato *Filtro di Bayes*: il belief viene calcolato a partire da una densità iniziale e viene aggiornato all'arrivo di ogni nuova informazione sensoriale, utilizzando dei modelli di aggiornamento dello stato per applicare i dati di movimento e di visione. L'incertezza sui dati misurati viene considerata definendo tali modelli in modo probabilistico. Nell'ambito della localizzazione ci si riferisce con il nome di "*Markov Localisation*"

[2]Il nome "kidnapped robot problem" è stato coniato da Engelson in [Engelson, 1994].
[3]Il termine belief è stato tradotto in "stato di credenza" in [Borghi, 1997] ma di seguito utilizzeremo il termine inglese belief.

[Nourbakhsh *e altri*, 1995; Simmons e Koenig, 1995; Burgard *e altri*, 1996].

Le soluzioni sviluppate negli ultimi anni si differenziano principalmente per il tipo di sensori utilizzati dal robot e per il modo di rappresentare, o meglio approssimare, il belief sulla posizione del robot. Molti ricercatori hanno utilizzato con successo dei sensori di prossimità come laser range finder o sonar [Fox *e altri*, 2000; Thrun *e altri*, 2000; Burgard *e altri*, 2000]: in questo caso il sistema di localizzazione deve cercare la posizione che meglio si adatta alla lettura del sensore, in un mappa metrica nota a priori. Questi sensori richiedono la conoscenza della mappa geometrica dell'ambiente, che spesso non è disponibile o non si vuole utilizzare. Inoltre sono attivi, cioè interagiscono con l'ambiente, tramite l'emissione del raggio laser o delle onde sonore e quindi risultano poco robusti ai disturbi esterni in un ambiente dinamico. Si pensi al caso in cui durante la lettura dell'eco del sonar una persona si muove davanti al sensore: in questo caso la lettura dell'eco viene interpretata in modo sbagliato, poiché rileva un oggetto non modellato dalla mappa. Negli ultimi anni la ricerca si è spostata verso sensori di tipo passivo, in particolare *sensori di visione* in cui la lettura delle informazioni non richiede nessuna interazione diretta con l'ambiente. Questi sensori risultano più robusti da utilizzare in un ambiente dinamico e per questo la nostra scelta si è rivolta verso la visione. I sensori di visione utilizzati maggiormente sono quelli di tipo prospettico [Dellaert *e altri*, 1999; Wolf, 2001; Wolf *e altri*, 2002a] che però forniscono una vista dell'ambiente attorno al robot limitata alla direzione puntata dalla telecamera. La nostra scelta è stata quella di utilizzare un sensore di visione omnidirezionale: il vantaggio è che questo tipo di sensore descrive l'intera scena attorno al robot con una sola immagine, e quindi fornisce una descrizione più completa e accurata dello stato del robot da una certa posizione.

L'altra distinzione che si può fare è, come detto, il tipo di rappresentazione del belief sulla posizione del robot: a seconda della scelta corrisponde una diversa soluzione del Filtro di Bayes. Approssimando il belief con una singola densità gaussiana, tramite media e covarianza la soluzione è quella del ben noto Filtro di Kalman [Gutmann *e altri*, 1998; Gutmann e Fox, 2002]. Questo approccio è molto efficiente ma può essere applicato solo al position tracking, in quanto gli altri due problemi indicati sopra richiedono la gestione di densità di probabilità con più modi corrispondenti alle ipotesi di posizione possibili.

I metodi basati sulle griglie approssimano lo spazio degli stati, cioè delle possibili posizioni con un insieme finito di celle; essi sono stati usati con successo per la localizzazione dei robot Rhino e Minerva utilizzati come guida in un museo [Fox *e altri*, 1999a; Burgard *e altri*, 1999; Thrun *e altri*, 2000]. Il loro svantaggio principale è che richiedono una grande capacità computazionale

perché, per avere una buona approssimazione, le celle devono essere piccole. Il metodo che è stato adottato con successo negli ultimi anni si basa sul *Metodo di Monte Carlo* e approssima il belief con dei campioni [Dellaert *e altri*, 1999; Fox *e altri*, 1999b, 2001; Lenser e Veloso, 2000]. Ciascun campione rappresenta una ipotesi di posizione del robot e l'insieme dei campioni viene aggiornato ad ogni passo in base alle informazioni di movimento. La bontà dell'ipotesi viene valutata in base alle informazioni di visione. Il vantaggio di questa tecnica è la capacità di rappresentare densità arbitrarie, in particolare multimodali e quindi risolve il position tracking e la localizzazione globale [Wolf *e altri*, 2002a]. Inoltre focalizza il calcolo solo sulle ipotesi (campioni) più probabili e risulta quindi molto efficiente. Questo approccio è conosciuto con il nome di *Localizzazione di Monte Carlo* (MCL) e verrà presentato in dettaglio nel capitolo 3.

In questa tesi proponiamo un nuovo modo di fondere due tecniche di localizzazione già usate con successo nella robotica mobile: la *localizzazione basata sulle immagini* e la *localizzazione di Monte Carlo*. Della seconda abbiamo già detto sopra. Nella prima si fornisce una memoria visiva al robot: vengono acquisite delle immagini, dette immagini di riferimento, in posizioni note chiamate posizioni di riferimento. Ciascuna immagine e la relativa posizione vengono memorizzate in un database; mentre il robot naviga nell'ambiente può acquisire un'immagine confrontarla con quelle nella memoria visiva. L'immagine di riferimento più simile fornisce la localizzazione del robot: tale localizzazione è detta *topologica* in quanto non è precisa ma indica solo l'area vicina alla posizione di riferimento dove è probabile che si trovi il robot. I problemi da affrontare sono quelli di come memorizzare e confrontare efficientemente le immagini. Inoltre la localizzazione basata sulle immagini fallisce nel caso di ambienti con periodicità spaziale, indicata con il termine "perceptual aliasing": non è possibile, con le sole informazioni di visione, distinguere tra due immagini simili. La soluzione è quella di fondere i due metodi di localizzazione basata sulle immagini e di Monte Carlo. Tale fusione è già stata proposta da altri ricercatori: in [Wolf *e altri*, 2002a] hanno usato un sistema di estrazione delle "feature" delle immagini prospettiche. In [Kröse *e altri*, 2001] hanno usato un sistema basato su un sensore omnidirezionale e sulla "Principal Component Analysis" (PCA) per ridurre l'occupazione di memoria del database.

In questa tesi vogliamo dimostrare la superiorità della localizzazione di Monte Carlo integrata ad un sistema di visione di tipo omnidirezionale e con un sistema di similarità che usa la trasformata di Fourier dell'immagine, rispetto ai sensori di tipo prospettico [Menegatti *e altri*, 2003a,b]. Il nostro scopo è quello di superare le limitazioni che ciascuna di queste tecniche presenta se utilizzata singolarmente. Abbiamo detto che la localizzazione basata

sulle immagini fallisce in caso di "perceptual aliasing" nell'ambiente; invece la localizzazione di Monte Carlo, così come è stata utilizzata fin'ora con un sensore di visione prospettico [Wolf *e altri*, 2002a], ha bisogno di una mappa dell'ambiente nota a priori per individuare le aree visibili dalla direzione puntata dalla telecamera e necessita di molte immagini di riferimento. Con un sensore omnidirezionale si ha la visione completa della scena attorno al robot e quindi non serve calcolare l'area di visibilità e non serve la mappa. Inoltre l'utilizzo della trasformata di Fourier consente di ottenere l'invarianza rotazionale nel confronto fra immagini, non possibile con la PCA e riduce notevolmente lo spazio necessario per memorizzare un'immagine.
Proponiamo inoltre un nuovo approccio per risolvere il problema del robot rapito: il metodo standard prevede di cercare il robot in tutto l'ambiente [Fox *e altri*, 1999b; Wolf *e altri*, 2002a] ma risulta poco efficiente; il metodo da noi proposto si basa sulle informazioni di visione e permette di cercare il robot solo nelle zone più probabili.

1.1 Struttura della tesi

La tesi è organizzata nel seguente modo:

Capitolo 2 descriveremo la localizzazione basata sulle immagini. Descriveremo il nostro sistema di visione basato su un sensore omnidirezionale e il nostro sistema per dotare il robot di una memoria visiva.

Capitolo 3 presenteremo l'approccio probabilistico alla localizzazione di robot mobili con particolare attenzione alla soluzione che si basa sul Metodo di Monte Carlo.

Capitolo 4 descriveremo la nostra implementazione del sistema di localizzazione di Monte Carlo, integrato al sistema di visione presentato nel capitolo 2; in particolare descriveremo la tecnica proposta per migliorare il comportamento del sistema di localizzazione nel problema del robot rapito. Infine descriveremo il software che abbiamo realizzato per implementare la localizzazione di Monte Carlo.

Capitolo 5 presenteremo i risultati degli esperimenti che abbiamo effettuato sul sistema di localizzazione. Questi test sono stati eseguiti con un programma di simulazione da noi sviluppato, utilizzando dati reali acquisiti da un nostro robot.

Capitolo 2

Localizzazione basata su immagini omnidirezionali

2.1 Introduzione

Un robot mobile che si muove in un ambiente deve pianificare il percorso e i suoi movimenti dalla posizione iniziale fino alla posizione di arrivo; deve inoltre evitare gli ostacoli fissi e mobili che trova durante il moto; deve infine scegliere fra più percorsi possibili. Per effettuare queste scelte e portare a termine il movimento con successo il robot deve conoscere la propria posizione nell'ambiente.
Esistono vari approcci per risolvere questo problema. Un metodo generale consiste nel fornire al robot una descrizione dettagliata dell'ambiente; normalmente si usa una mappa geometrica (ad esempio un modello CAD). Inoltre il robot viene dotato di qualche tipo di sensore: i più comuni sono i sensori di visione con telecamere prospettiche o omnidirezionali oppure sensori di prossimità quali "laser range finder" o anelli di sensori sonar; il robot si localizza cercando la posizione della mappa che coincide maggiormente con i dati letti dal sensore.
Purtroppo i sensori sono rumorosi e per questo vengono ingannati facilmente dalla complessità e dagli aspetti dinamici dell'ambiente. Per superare questo problema vengono usati dei sensori laser molto precisi in combinazione con un sistema in grado di gestire l'incertezza [Thrun *e altri*, 2000; Burgard *e altri*, 2000].
Un altro approccio, comune nelle applicazioni reali, è quello di modificare l'ambiente; vengono inseriti dei punti di riferimento artificiali chiamati "landmark": si usano strisce e punti riflettenti che sono facilmente riconoscibili e che il robot usa per localizzarsi [Hu e Gu, 2000].

Questi due approcci hanno degli svantaggi e non sono sempre realizzabili. Infatti esistono delle situazioni in cui non si ha a disposizione una mappa precisa dell'ambiente: ad esempio in vecchi edifici o in luoghi inesplorati. Inoltre anche se si conosce la mappa precisa essa è inutile in ambienti dinamici dove la configurazione degli oggetti cambia frequentemente (ad esempio per la presenza di persone, altri robot che si muovono oppure l'ambiente che cambia). Per questo è utile che il robot sia in grado di costruire la propria rappresentazione dell'ambiente: spesso la rappresentazione scelta è la mappa topologica in cui vengono descritte le connessioni tra le aree dell'ambiente mentre si tralasciano tutte le informazioni metriche [Choset e Nagatani, 2001; Kuipers, 2000].

La capacità di ragionare sulla topologia e geometria dell'ambiente descritta sopra, è un'abilità posseduta dall'uomo. Un'altra caratteristica che può essere riconosciuta analizzando la navigazione umana, è la capacità di memorizzare, ricordare e riconoscere delle scene, cioè l'insieme di quello che l'uomo vede (nell'uomo questo processo è soggettivo). Questo significa che l'uomo possiede una *memoria visiva* che può utilizzare per capire la propria posizione in grandi ambienti. Esistono evidenze sperimentali che anche animali semplici come le formiche e le api usano la memoria visiva per localizzarsi in grandi ambienti [Collett *e altri*, 1992].

Da queste considerazioni è sorto un nuovo approccio ai problemi di localizzazione e navigazione, la *navigazione/localizzazione basata sulle immagini*; il robot viene dotato di un sistema di visione; gli vengono fornite una serie di viste (scene) acquisite in varie posizioni nell'ambiente. Queste posizioni sono chiamate *posizioni di riferimento* perché il robot si riferisce ad esse per localizzarsi; le corrispondenti immagini sono chiamate *immagini di riferimento* e l'insieme di queste informazioni costituisce la *memoria visiva* del robot.
Quando il robot si muove nell'ambiente può acquisire e confrontare la scena corrente con quelle della sua memoria visiva. Determinando l'immagine di riferimento più simile all'immagine corrente, il robot può desumere la sua posizione. Osserviamo che questa è solo una *localizzazione topologica*: si può affermare che il robot è più vicino alla corrispondente posizione di riferimento (almeno con un certo grado di confidenza o sicurezza, legato al sistema di similarità) più che ad ogni altra mentre non è in generale possibile stabilire una posizione precisa.
In questo modo si è ridotto il problema di localizzazione al problema di determinare quale tra le immagini di riferimento è più simile all'immagine corrente. Il difetto di questo approccio è che la localizzazione è solamente *reattiva*: esiste un corrispondenza diretta e predefinita tra un'immagine di input e la corrispondente localizzazione. Quindi se più di una posizione di riferimento corrisponde ad un'immagine di input allora il robot non è in grado

di decidere tra le possibili ipotesi. Questa situazione si verifica in ambienti con struttura periodica, ad esempio con lo stesso tipo di porte, pareti uniformi, oggetti e illuminazione disposti in modo regolare: è il caso di uffici di uno stesso palazzo che spesso sono arredati in modo simile o di corridoi di ospedali che si ripetono tutti uguali. Un esempio di un tale ambiente è quello che abbiamo chiamato *Test5*, che è costituito da un lungo corridoio e che abbiamo usato per testare il nostro sistema di localizzazione. Si veda la sezione 5.1 per una descrizione dell'ambiente.
I problemi che dobbiamo affrontare per costruire un sistema efficiente di similarità tra immagini sono quelli di come memorizzare e poi comparare le immagini, che possono essere numerose e troppo grandi da immagazzinare in memoria. Abbiamo scelto di utilizzare un sensore omnidirezionale [Ishiguro, 2001; Menegatti *e altri*, 2002a]. La proprietà principale è quella di fornire una vista a 360° in una sola volta. In questo modo servono meno immagini per descrivere l'intero ambiente e inoltre, come dimostreremo, la localizzazione topologica è più accurata in quanto si ha a disposizione tutta la scena in una volta sola.
Per ridurre la quantità di memoria fisica necessaria per costituire la memoria visiva del robot, le immagini di riferimento non vengono memorizzate direttamente ma utilizziamo una rappresentazione compatta, che si basa sull'espansione nella serie di Fourier e che chiamiamo "*Fourier signature*" dell'immagine: per ciascuna immagine memorizziamo solo i coefficienti corrispondenti alle frequenze più basse della serie; infatti, come dimostreremo nella prossima sezione, la parte predominante dello spettro di potenza della trasformata di Fourier dell'immagine è contenuta alle frequenze inferiori alla 15^a. Questo ci permette di trascurare i coefficienti a frequenze maggiori ottenendo una rappresentazione molto compatta dell'immagine omnidirezionale.

Nel seguito del capitolo vedremo come si calcola la "Fourier signature" di un'immagine omnidirezionale, come da questa possiamo ricavare una misura di similarità tra due immagini e come possiamo sfruttare questo sistema per effettuare una *localizzazione gerarchica*: una localizzazione più o meno accurata a seconda delle esigenze di navigazione; ad essa corrisponde un sistema di localizzazione rispettivamente meno o più veloce.

2.2 Caratterizzazione delle immagini

Abbiamo sottolineato nell'introduzione che il primo problema da affrontare nel costruire un sistema localizzazione basata sulle immagini è quello di scegliere come memorizzare e confrontare le immagini di riferimento. Ci poniamo tre obiettivi:

1. minimizzare la quantità di memoria necessaria per costituire la memoria visiva del robot;
2. massimizzare la velocità di confronto tra le immagini;
3. avere un sistema che descrive la scena in modo completo e che quindi permetta di determinare le immagini simili in modo attendibile.

Per ridurre il numero di immagini necessarie abbiamo scelto di utilizzare una telecamera omnidirezionale. Un sensore di questo tipo permette di memorizzare in una singola immagine, l'intera vista a 360° da una certa posizione. Molti altri sistemi sono stati dotati di una normale telecamera prospettica che però fornisce una vista ridotta legata alla direzione puntata dalla telecamera; in questo modo sono necessarie più immagini per avere la visione completa dell'ambiente in una certa posizione e questo richiede quantità di memoria e carichi computazionali troppo elevati e non gestibili in tempo reale. Per risolvere questo problema si può costringere il robot a mantenere un'orientazione fissa in modo che la telecamera punti sempre nella stessa direzione [Cassinis *e altri*, 2002], ma questo limita troppo il moto del robot. Un'altra soluzione consiste nell'estrarre e memorizzare solo alcune "feature" che descrivono l'immagine in modo non ambiguo[1]; un'esempio di questo metodo si trova in [Wolf *e altri*, 2002a] in cui la memoria visiva del robot è costituita da 936 immagini memorizzate in 4MB.
Acquisire così tante immagini può richiedere troppo tempo. Per ridurre ulteriormente la memoria necessaria le immagini vengono pre-elaborate: una tecnica comune è quella di estrarre un insieme di "eigenimages" dalle immagini di riferimento e descrivere queste immagini come combinazione lineare delle "eigenimages". Purtroppo questo metodo non porta ad un'invarianza rotazionale delle immagini che appaiono diverse anche se prese nella stessa posizione ma con orientazioni differenti. Così si deve fissare l'orientazione del robot [Kröse *e altri*, 2001] oppure elaborare ulteriormente le immagini per ottenere l'invarianza voluta [Aihara *e altri*, 1998; Gaspar *e altri*, 2000; Jogan e Leonardis, 2000].
Inoltre per la navigazione non è necessario avere immagini ad alta risoluzione come quelle prospettiche. Le immagini omnidirezionali quindi risultano la scelta migliore: il noto svantaggio della loro bassa risoluzione non è una limitazione e può essere un vantaggio in quanto diminuisce il numero di pixel da elaborare. Quindi per ogni posizione di riferimento abbiamo una sola immagine omnidirezionale.

Il secondo passo è la scelta di come confrontare le immagini fra loro. L'approccio più semplice potrebbe essere il confronto pixel per pixel ma risulta

[1] Le feature sono le caratteristiche distintive di un'immagine.

poco robusto e inefficiente perché richiede di memorizzare l'intera immagine. Abbiamo scelto di utilizzare la *Trasformata di Fourier* dell'immagine omnidirezionale e un sottoinsieme dei coefficienti come caratterizzazione ("signature") dell'immagine stessa [Ishiguro e Tsuji, 1996; Menegatti *e altri*, 2002b]. Osserviamo che non applichiamo la trasformata 2-D all'immagine originale ma calcoliamo la trasformata 1-D di ogni riga del cilindro panoramico corrispondente ad una immagine omnidirezionale. Il cilindro panoramico è l'immagine ottenuta con una operazione di "unwarping" applicata all'immagine omnidirezionale originale. In Figura 2.1 si vede un esempio di immagine omnidirezionale mentre in Figura 2.2 si vede il corrispondente cilindro panoramico. L'intero processo viene descritto schematicamente per altre due immagini in Figura 2.3.

Figura 2.1: Immagine omnidirezionale acquisita in una posizione di riferimento nell'ambiente GridLab3 (vedi capitolo 5).

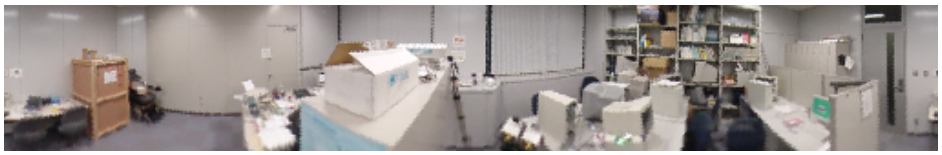


Figura 2.2: Cilindro panoramico ottenuto dall'immagine omnidirezionale di Figura 2.1.

Il cilindro panoramico è una funzione periodica lungo l'asse x (la lunghezza): questo semplifica il calcolo della trasformata ed è la rappresentazione naturale di una invarianza rotazionale. Come abbiamo sottolineato sopra,

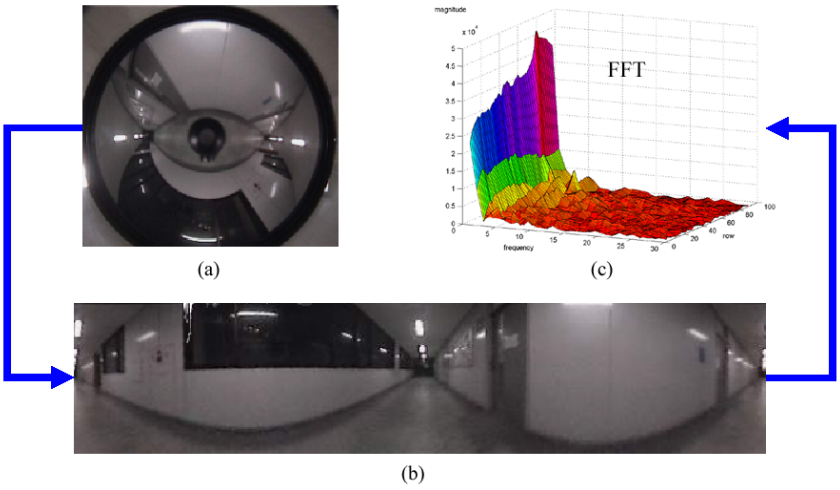


Figura 2.3: Processo di generazione della "Fourier signature" di un'immagine. Dall'immagine omnidirezionale (a) otteniamo il cilindro panoramico (b) tramite un'operazione di "unwarping". Infine calcoliamo i coefficienti della Trasformata di Fourier di ogni riga del cilindro panoramico: il grafico è visualizzato in (c).

vogliamo essere in grado di determinare l'immagine di riferimento più simile alla vista corrente indipendentemente dalla sua orientazione. Per questo è necessario introdurre una invarianza rotazionale nel calcolo della similarità: utilizzando i coefficienti di Fourier come "signature" dell'immagine questo si ottiene automaticamente. Infatti se il robot acquisisce due immagini omnidirezionali nella stessa posizione ma con orientazioni diverse allora le due immagini sono in realtà la stessa immagine ruotata rispetto al centro e tale rotazione coincide con la differenza in gradi tra le due orientazioni del robot. Allora i due cilindri panoramici sono la stessa immagine solo con una traslazione lungo l'asse delle x.
Se indichiamo con $f(x)$ il primo cilindro panoramico e con $f(x-a)$ il cilindro panoramico traslato di una quantità a lungo l'asse x, allora applicando il Teorema di Traslazione della trasformata [Cariolaro, 1996] otteniamo:

$$\mathcal{F}\{f(x-a)\} = e^{-j2\pi as}\mathcal{F}\{f(x)\} \quad , \tag{2.1}$$

dove $\mathcal{F}\{f(x)\}$ è la trasformata di Fourier di $f(x)$. Quindi la trasformata del segnale traslato è uguale alla trasformata del segnale originale moltiplicato per il numero complesso $e^{-j2\pi as}$ che ha modulo unitario. Questo significa che il modulo della trasformata di Fourier dell'immagine traslata non è cambiato e tra le due trasformate c'è solo una variazione di fase proporzionale all'ampiezza della traslazione a.

Immagine	Memoria richiesta (in bit)	Memoria richiesta
omnidirezionale	$640 \times 480 \times 24$	$\simeq$ 1 Mbyte
cilindro panoramico	$512 \times 90 \times 24$	$\simeq$ 140 Kbyte
Fourier signature	$90 \times 15 \times 2 \times 8$	$\simeq$ 2.7 Kbyte

Tabella 2.1: Riduzione della memoria necessaria per la "signature" dell'immagine omnidirezionale, che otteniamo utilizzando la trasformata di Fourier.

Sfruttando la proprietà descritta associamo l'ampiezza della trasformata all'aspetto (vista) dell'ambiente da una certa posizione mentre associamo la fase della trasformata all'orientazione del robot. In questo modo se il robot ruota su se stesso, sul posto la vista dell'ambiente e la relativa ampiezza della trasformata non cambiano. Quello che cambia è la fase della trasformata che varia proporzionalmente alla variazione di orientazione.
Con il metodo descritto abbiamo ottenuto l'invarianza rotazionale cercata. Inoltre abbiamo anche un modo per calcolare la differenza di orientazione tra l'immagine corrente e un'immagine di riferimento[2]: questa si calcola dalla differenza di fase tra le trasformate delle due immagini.

La riduzione di memoria che otteniamo con il sistema utilizzato è notevole e si può vedere in Tabella 2.1. L'immagine omnidirezionale originale è di tipo "true color" di 640×480 pixel: viene memorizzata usando 8 bit per ciascuno dei 3 canali in formato PPM. Questo significa che occupa circa 7.3 Mbit, cioè circa 1 Mbyte di memoria. Tramite l'operazione di "unwarping" otteniamo un cilindro panoramico di 512×90 pixel ancora "true color". Quindi il cilindro panoramico occupa circa 1.1 Mbit, cioè circa 140 Kbyte di memoria. Infine a partire dal cilindro panoramico otteniamo la "Fourier signature" calcolando l'ampiezza e la fase della trasformata di Fourier di ogni riga. Riportiamo il grafico dell'ampiezza dei coefficienti di Fourier di una di queste immagini in Figura 2.4. Osserviamo che la potenza dello spettro è contenuta in modo dominante alle basse frequenze, minori della 15^a. Quindi possiamo rappresentare la "signature" dell'immagine omnidirezionale con i soli valori delle prime 15 componenti della trasformata di Fourier. Quindi per ciascuna delle 90 righe memorizziamo i primi 15 coefficienti di ampiezza e fase della trasformata, ogni coefficiente con 8 bit: la "Fourier signature" occupa circa 21.6 Kbit, cioè circa 2.7 Kbyte.
Volendo confrontare con il metodo presentato in [Wolf, 2001; Wolf *e altri*, 2002a] in cui 936 immagini vengono memorizzate in circa 4 Mbyte, con il nostro metodo un numero così elevato di immagini richiederebbe circa 2.3 Mbyte.

[2]Che è l'orientazione del robot quando è stata acquisita tale immagine.

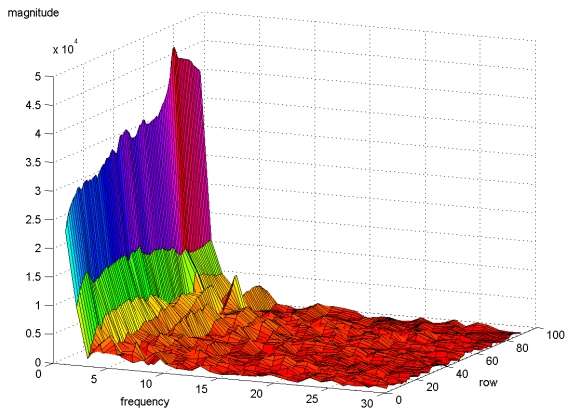


Figura 2.4: Ampiezza dei coefficienti di Fourier di un'immagine omnidirezionale. Sono mostrate solo le prime 30 componenti. Si nota che le componenti dopo la 15^a assumono valori molto bassi: per questo possiamo utilizzare le sole basse frequenze, fino alla 15^a, come "signature" dell'immagine e trascurare le altre nel calcolo della similarità.

2.3 Calcolo della similarità tra immagini

Per il calcolo della similarità di due immagini omnidirezionali definiamo innanzitutto una *funzione di dissimilarità* che prende come parametri le "signature" delle immagini, cioè due vettori di m valori, ciascuno che rappresenta l'ampiezza dei primi m coefficienti di Fourier associati alle due immagini.
La funzione di dissimilarità tra due immagini omnidirezionali O_i, O_j è calcolata come la norma L_1 dei coefficienti di Fourier dei cilindri panoramici corrispondenti. Ciascun cilindro è composto da l righe. Inoltre consideriamo solo i primi m coefficienti della serie di Fourier poiché le componenti alle alte frequenze hanno potenza trascurabile:

$$Dis(O_i, O_j) = \sum_{y=0}^{l-1} \sum_{k=0}^{m-1} |F_{iy}(k) - F_{jy}(k)| \quad , \tag{2.2}$$

dove $F_{iy}(k)$ e $F_{jy}(k)$ sono i coefficienti di Fourier alla frequenza k della y-esima riga di O_i e O_j rispettivamente.

Si vede dalla Figura 2.5 che il valore della funzione $Dis(O_i, O_j)$ cresce linearmente con la distanza tra l'immagine corrente e quella di riferimento; questo vale nel breve raggio mentre per distanze maggiori la funzione satura ad una costante che corrisponde ad un valore di "no-matching" restituito quando il calcolo della sommatoria dell'equazione 2.2 supera una soglia scelta per velocizzare l'algoritmo. Questa scelta è giustificata dal fatto che in tal

caso le immagini sono state prese in posizioni distanti e non c'è più nessuna correlazione tra esse.

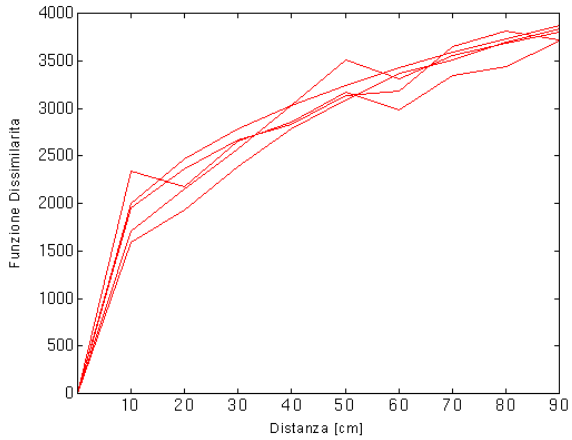


Figura 2.5: Grafico del valore della funzione di dissimilarità in relazione alla distanza tra le immagini. Ciascuna linea rappresenta una coppia diversa di immagini. Si vede che la relazione è lineare per piccole distanze, mentre tende a saturare per distanze elevate.

Per convenienza definiamo anche una *funzione di similarità* ottenuta riscalando i valori di dissimilarità tra 0 e 1000:

$$Sim(O_i, O_j) = 1000 - 1000 \frac{Dis(O_i, O_j) - \min_{i,j} Dis(O_i, O_j)}{\max_{i,j} Dis(O_i, O_j) - \min_{i,j} Dis(O_i, O_j)} \quad . \quad (2.3)$$

Il valore di $Sim(O_i, O_j)$ è decrescente con la distanza ed esprime correttamente il significato di una stima di similarità: maggiore per immagini più vicine e via via decrescente per immagini più lontane.

Osserviamo che il valore assoluto della funzione di similarità non è importante e dipende dalle caratteristiche dell'ambiente. Sono rilevanti solo i valori relativi della similarità dell'immagine corrente rispetto a tutte le immagini di riferimento. L'immagine di riferimento più simile all'immagine corrente è quella a cui corrisponde il valore più basso della funzione di dissimilarità e quindi il valore più alto, pari a 1000, della funzione similarità.

Quindi data l'immagine corrente possiamo ottenere la sua "Fourier signature" con il processo descritto nella sezione precedente. A questo punto possiamo calcolare i valori di similarità tra quest'immagine e quelle di riferimento secondo l'equazione (2.3) che calcola la norma $L1$, scalata tra 0 e 1000, tra le "Fourier signature". Le immagini di riferimento con valori più alti della funzione Sim sono quelle considerate più simili a quella corrente.

In particolare l'immagine di riferimento con valore di similarità pari a 1000 fornisce la *localizzazione topologica* del robot, poiché possiamo affermare che il robot è più vicino alla posizione di questa immagine di riferimento rispetto a tutte le altre. In realtà questo è vero solo se non sono presenti delle notevoli occlusioni o disturbi nelle immagini, che possono impedire la corretta individuazione dell'immagine più simile. Vedremo nel capitolo 5 l'effetto delle occlusioni sul sistema di calcolo delle similarità.

Ricordiamo infine che la localizzazione basata sulle immagini e in particolare il sistema appena descritto fallisce in ambienti con struttura periodica come nel caso dei labirinti, o più comunemente di ambienti con corridoi uguali. In questi casi le viste dell'ambiente sono le stesse in posti diversi e la vista corrente può coincidere con più di una vista di riferimento: il sistema descritto non è in grado di decidere tra le varie posizioni. Questo è un problema di "perceptual aliasing" ed è la situazione in cui si può trovare un uomo che si perde in un ambiente in cui tutti i posti sembrano simili e non esistono dei punti di riferimento distinti. Per orientarsi una persona utilizza altri indizi, come i cartelli stradali oppure cerca di memorizzare le direzioni delle strade percorse. Un robot può utilizzare dei sensori più sofisticati della visione come i sensori GPS ("Global Positioning System"). Ma spesso questi sensori non sono utilizzabili, ad esempio in ambienti indoor oppure all'esterno ma coperti da alberi dove il segnale GPS non può arrivare, oppure risultano troppo costosi da installare sul robot rispetto all'applicazione a cui è dedicato; per questo è necessario utilizzare delle tecniche che siano in grado di gestire l'incertezza delle situazioni in cui il robot ha evidenza di trovarsi in due o più posizioni distinte. La soluzione che abbiamo adottato si basa sul ben noto metodo di Monte Carlo già utilizzato da altri ricercatori in questo e in altri campi di applicazione, quali il "tracking" di oggetti e la stima di segnali. Lo presenteremo nel prossimo capitolo.

2.4 Localizzazione gerarchica

L'idea della localizzazione gerarchica deriva da un'esperienza comune a tutti noi quando camminiamo in una grande città che non conosciamo e usiamo una mappa per orientarci: mentre ci muoviamo lungo la strada principale non ci serve conoscere con precisione la posizione nella mappa ma per entrare in un edificio o fare una deviazione dobbiamo ridurre l'incertezza sulla posizione utilizzando un numero maggiore di indizi dell'ambiente. Le stesse considerazioni valgono per un robot che naviga in uno spazio aperto confrontato con un robot che deve entrare in un ufficio passando per una porta: il robot avrà bisogno della sua posizione con un diverso livello di accuratezza,

che dipende dall'azione che deve compiere e dalla conformazione dell'ambiente dove si trova.
Chiamiamo questo processo con il nome di *localizzazione gerarchica*.
Il concetto di navigazione con diversi livelli di accuratezza non è nuovo. Altri autori hanno ottenuto questo risultato utilizzando due strategie diverse di navigazione basata sulla visione: ad esempio in [Gaspar *e altri*, 2000] la navigazione topologica (meno accurata) viene sostituita all'occorrenza dalla navigazione chiamata "visual path following" (più precisa). Quest'ultima però richiede un accurato sistema di controllo.

Con il sistema di visione che abbiamo presentato sopra otteniamo una diversa accuratezza di localizzazione sfruttando le proprietà del sistema di localizzazione topologica basato sulle immagini.
Il nostro metodo si basa sul significato dei coefficienti della "Fourier signature" calcolati dal cilindro panoramico: infatti, quando calcoliamo la trasformata di Fourier di un segnale di luminosità, come una riga del cilindro panoramico, decomponiamo il segnale nelle sue componenti rispetto ad una base di funzioni; queste funzioni sono legate alla variazione spaziale di luminosità, cioè ai "pattern di luminosità. La prima funzione base che corrisponde alla frequenza zero è il segnale di luminosità costante e il relativo coefficiente esprime il livello di luminosità del segnale. Le stesse considerazioni valgono per frequenze più elevate.
Come si vede dalle equazioni (2.2) e (2.3) quando calcoliamo la similarità tra due immagini sommiamo i contributi a diverse frequenze ma osserviamo che il contributo maggiore viene dato dalle basse frequenze.
Inoltre la luminosità media delle immagini varia molto lentamente all'aumentare della distanza tra le immagini; invece la distribuzione e la presenza dei pattern di luminosità, che rappresentano gli oggetti dell'ambiente, variano più velocemente. Quindi le componenti alle basse frequenze delle trasformate di due immagini sono simili per intervalli più ampi di distanza rispetto alle componenti ad alte frequenze perché sono ques'ultime a contenere le caratteristiche distintive delle immagini. Concludiamo quindi che fermando il calcolo della similarità alle prime frequenze allora l'immagine corrente verrà considerata simile ad un numero maggiore di immagini di riferimento: normalmente queste immagini di riferimento sono vicine alla vera immagine di input ma possono essere anche molto distanti se esiste "perceptual aliasing" [Menegatti *e altri*, 2002b, 2003a]. Otteniamo in questo modo una localizzazione topologica più ampia.

Quindi otteniamo una localizzazione gerarchica semplicemente scegliendo il numero di componenti di Fourier da confrontare nella funzione di dissimilarità (2.2). Con la tecnica che abbiamo illustrato la localizzazione gerarchica si ottiene automaticamente e permette di risparmiare potenza computazio-

nale: quando il robot ha bisogno di una localizzazione non accurata allora può fermare il calcolo della somma interna dell'equazione (2.2) ad un k basso. Per avere una localizzazione più precisa è sufficiente estendere la somma a valori più elevati di k.

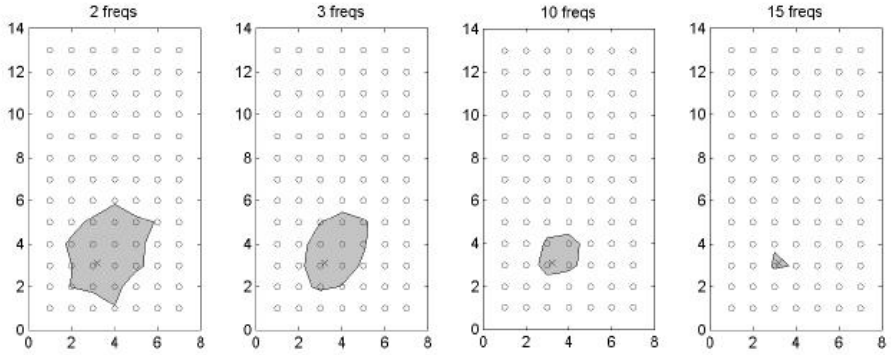


Figura 2.6: Esempio di localizzazione gerarchica. La localizzazione è stata eseguita considerando un numero diverso di frequenze per il calcolo della similarità. I cerchi vuoti rappresentano le immagini di riferimento mentre il cerchio pieno rappresenta la posizione corrente dell'immagine di input. L'area grigia rappresenta le possibili localizzazioni topologiche.

In Figura 2.6 si vede un esempio di localizzazione gerarchica nel più semplice dei due ambienti dove abbiamo eseguito gli esperimenti che presenteremo nel capitolo 5. Questo ambiente è lo stesso delle immagini 2.1 e 2.2. I cerchi vuoti rappresentano le posizioni delle immagini di riferimento mentre il cerchio pieno indica la vera posizione del robot. L'area grigia rappresenta le possibili posizioni del robot. Il numero di componenti di Fourier utilizzate per calcolare la similarità aumenta da sinistra a destra: si vede come parallelamente l'area delle possibili posizioni si restringe.

In Figura 2.7 presentiamo un esempio di "perceptual aliasing" nel secondo ambiente di test chiamato *Test5*, che è quello delle immagini di Figura 2.3. Si faccia riferimento alla sezione 5.1 per la descrizione degli ambienti. I grafici mostrano i livelli di similarità con colori diversi: il rosso indica massima similarità mentre il blu indica minima similarità. I quattro grafici si riferiscono a quattro immagini di input diverse ma presentano tutti la stessa situazione: l'immagine di input risulta simile a più immagini di riferimento, anche appartenenti a zone diverse dell'ambiente. In particolare il grafico di Figura 2.7 (a) si riferisce all'immagine di input 10 che si trova alla posizione di 1374 cm lungo l'asse x. Essa risulta maggiormente simile all'immagine di riferimento alla posizione 1380 cm che è quella corretta, poiché è la più

vicina. Ma risulta molto simile anche alle immagini di riferimento alle posizioni 320, 620, 1060 cm. In questo caso con la sola localizzazione basata sulle immagini non potremmo affermare con sicurezza la posizione del robot in quanto esistono almeno quattro ipotesi di posizione probabili.

In Figura 2.8 presentiamo i grafici di similarità dell'immagine di input 10 calcolati utilizzando un diverso numero di frequenze: 5, 7, 15 e 50 frequenze[3]. Si vede in (a) e (b) che con un numero basso di frequenze 5 e 7 la localizzazione topologica è meno accurata rispetto a 15 frequenze (c): non solo esistono delle false ipotesi in altre posizioni ma che i picchi di similarità in corrispondenza di queste ipotesi sono ampi cioè poco selettivi. Questo succede perché usando poche frequenze la funzione di similarità non riesce a cogliere le differenze distintive tra le immagini. Notiamo infine dalla (d) che utilizzando un numero maggiore di frequenze, pari a 50, non otteniamo una localizzazione topologica più selettiva. Per questo possiamo affermare che la scelta migliore per il calcolo della similarità è quella di utilizzare 15 frequenze.

Riassumendo: con il sistema descritto possiamo associare un valore di similarità tra immagini; esso fornisce un metodo diretto per il calcolo di una localizzazione gerarchica, confrontando gli spettri di frequenza dell'immagine corrente con quelli delle immagini di riferimento. Rinunciando ad avere una accuratezza maggiore si ottiene un minimo carico computazionale. Dimostreremo nel capitolo 5 che il nostro sistema di similarità è robusto alle occlusioni; inoltre mostreremo come sfruttare le proprietà descritte per fondere questa tecnica con la localizzazione di Monte Carlo, in modo da superare i limiti di entrambe.

[3]Ricordiamo che vengono usate le m frequenze più basse.

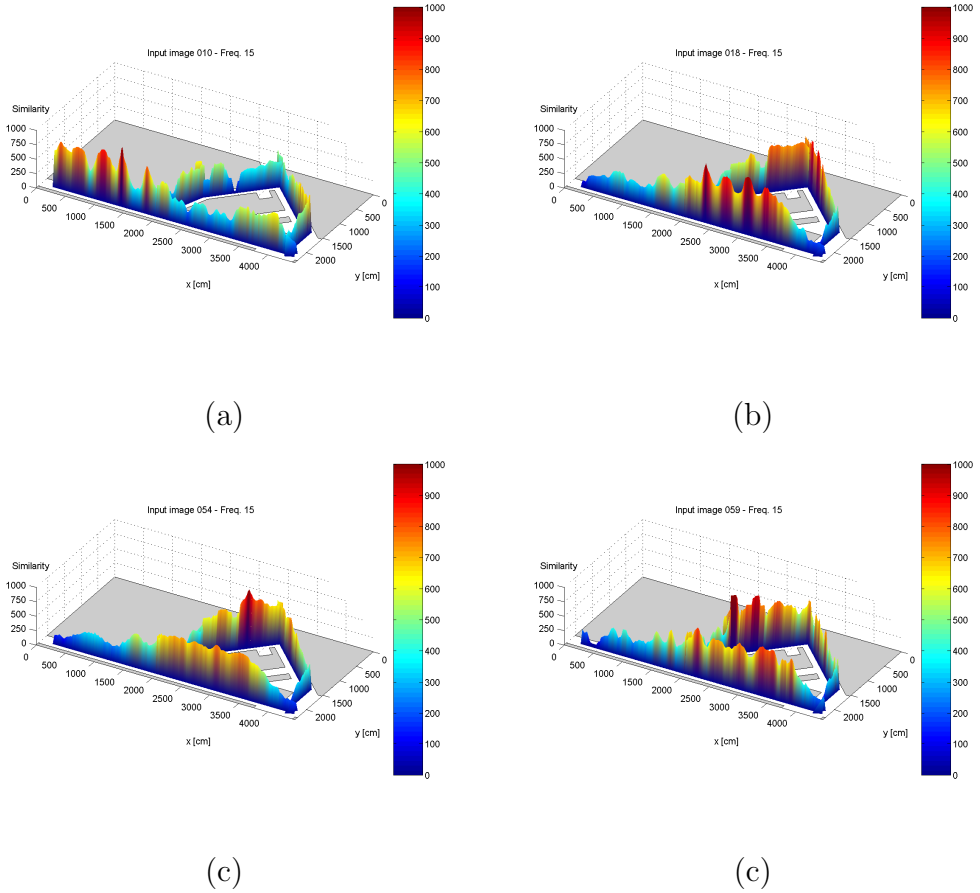
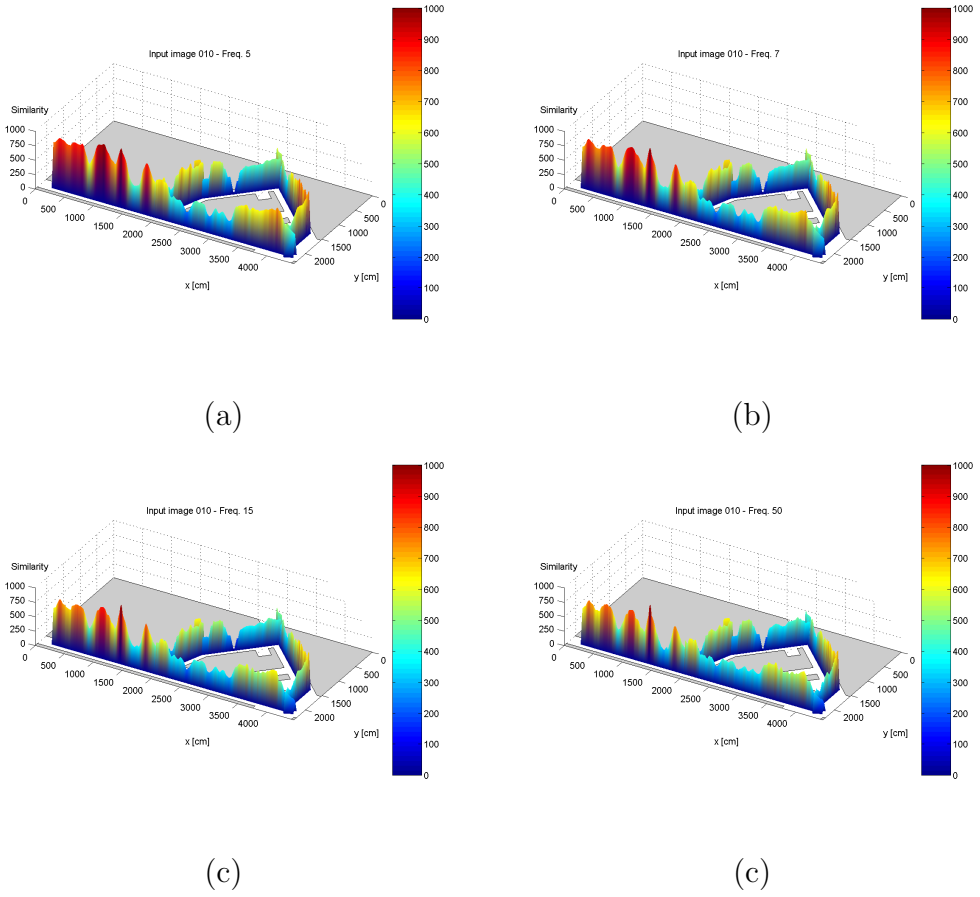


Figura 2.7: Esempi di "perceptual aliasing" in diverse posizioni nell'ambiente *Test5*. La barra di colore indica il grado di similarità tra le immagini: il colore rosso indica la massima similarità mentre il blu indica la minima similarità. In Figura (a) si vede che l'immagine di input 10 risulta simile alla corretta immagine di riferimento alla posizione 1380 cm lungo l'asse x ma anche alle immagini di riferimento a 320, 620 e 1060 cm. Lo stesso comportamento si può osservare nelle figure (b), (c) e (d).



(a) (b)

(c) (c)

Figura 2.8: Grafici della similarità per l'immagine di input 10 a diverse frequenze. Si vede in (a) e (b) che con un 5 e 10 frequenze la localizzazione topologica è più ampia rispetto (c) che corrisponde a 15 frequenze. In (d) si vede che oltre le 15 frequenze la localizzazione topologica non diventa più selettiva.

Capitolo 3

Localizzazione di Monte Carlo per Robot Mobili

In questo capitolo presenteremo l'approccio probabilistico alla localizzazione di robot mobili inteso come problema di stima di un sistema dinamico a partire da informazioni acquisite dai sensori e soggette a disturbi. Questo approccio rientra nella categoria generale dei Filtri di Bayes che verranno presentati nella sezione 3.1. La stima viene effettuata considerando le densità di probabilità delle variabili da stimare condizionate alle informazioni provenienti dai sensori.
Nelle applicazioni con requisiti di real time, come la robotica mobile, è necessario ricorrere a soluzioni approssimate perché il trattamento di densità di probabilità continue è troppo oneroso. Negli ultimi anni quindi sono stati sviluppati molti metodi approssimati per la soluzione del Filtro di Bayes, che nella localizzazione di robot viene chiamato anche Localizzazione di Markov [Nourbakhsh *e altri*, 1995; Simmons e Koenig, 1995; Burgard *e altri*, 1996; Fox *e altri*, 1999b]; alcune di queste soluzioni saranno presentate nella sezione 3.2. L'attenzione di molti ricercatori si è spostata recentemente verso una soluzione che si basa sul campionamento delle densità di probabilità, quindi utilizza delle approssimazioni discrete piuttosto che le corrispondenti grandezze continue. Questo approccio prende il nome di Localizzazione di Monte Carlo (MCL) nelle applicazioni di robotica mobile e verrà presentato in sezione 3.4. Il metodo è già stato applicato con successo alla localizzazione [Dellaert *e altri*, 1999; Fox *e altri*, 1999b, 2001; Lenser e Veloso, 2000; Wolf *e altri*, 2002a].
Nel capitolo 4 presenteremo la nostra implementazione della MCL, integrata al nostro sistema di visione omnidirezionale e proporremo una nuova tecnica per migliorare il comportamento nel caso del problema del robot rapito.

3.1 Filtraggio Bayesiano

In molti problemi reali è richiesta la stima dello stato di un sistema dinamico a partire da informazioni misurate sul sistema tramite sensori. Queste misure, chiamate osservazioni, sono dotate di rumore e quindi risultano inaffidabili e non permettono la stima diretta dello stato. Quasi sempre si ha a disposizione la conoscenza a priori del fenomeno studiato, cioè si ha un modello del sistema e si può conoscere suo stato iniziale. I problemi che presentano questa struttura vengono spesso indicati con il nome di problemi di stima in condizioni di incertezza; essi possono essere rappresentati con un *Modello Bayesiano* cioè tramite una densità di probabilità a priori per lo stato non conosciuto e funzioni di probabilità che legano lo stato alle osservazioni. Utilizzando il Teorema di Bayes si calcola la densità a posteriori dello stato e da questa si effettuano le analisi di inferenza richieste. Spesso le osservazioni arrivano in modo sequenziale nel tempo: aggiornando la densità a posteriori appena i dati diventano disponibili è possibile eseguire un'inferenza in linea. In questo caso si ha il vantaggio di non dover memorizzare tutti i dati osservati prima di poterli elaborare.
I metodi di soluzione a questi problemi sono chiamati *Filtri di Bayes*.

Se il sistema è lineare e gaussiano allora il Filtro di Kalman fornisce la soluzione ottima cioè permette di calcolare completamente la densità a posteriori. Osserviamo però che spesso i sistemi reali non soddisfano queste ipotesi e inoltre presentano elevate dimensioni di stato che precludono la soluzione ottima. Si ricorre quindi a metodi approssimati di stima quali il Filtro di Kalman Esteso e i metodi basati sulle griglie ("Grid-based") [Arulampalam *e altri*, 2002]. Questi metodi presentano alcuni svantaggi notevoli: il Filtro di Kalman non funziona con densità multimodali mentre l'approccio basato sulle griglie è dispendioso dal punto di vista computazionale.
L'approccio che permette di superare questi svantaggi fa parte della categoria dei *Metodi di Monte Carlo Sequenziali* (SMC). Esso si basa sulla simulazione tramite campioni della densità da approssimare.
Negli ultimi anni sono stati sviluppati numerosi algoritmi SMC in diverse aree: robotica, navigazione di terra, visione artificiale, inseguimento di bersagli, guida di missili, reti neurali, finanza e biologia delle popolazioni. Un'introduzione a queste applicazioni si può trovare in [Doucet *e altri*, 2001b]. Gli algoritmi sono chiamati con nomi specifici nei vari campi di applicazione: ad esempio Filtri a Particelle, Filtri "Bootstrap", "Survival of the fittest".

Nel seguito presenteremo in generale l'approccio Bayesiano alla stima in tempo reale di sistemi dinamici, non lineari e non gaussiani. Verrà poi introdotto in dettaglio il Metodo di Monte Carlo come possibile soluzione approssimata del problema.

3.1.1 Formalizzazione del problema

Per l'analisi di un sistema dinamico servono due modelli: il *modello del sistema* e il *modello delle misure o di osservazione* [Arulampalam *e altri*, 2002]. Il primo descrive l'evoluzione dello stato del sistema nel tempo mentre il secondo descrive la relazione tra lo stato e le misure dotate di rumore.
Formalmente lo stato e le misure sono rappresentate come processi aleatori e quindi i due modelli sono espressi in forma probabilistica, tramite le densità di probabilità (PDF).

La sequenza degli stati viene indicata $\{s_t : t \in \mathbb{N}\}, s_t \in \mathcal{S}$ ed è un processo di Markov di densità iniziale $p(s_0)$. La sequenza delle osservazioni viene indicata con $\{y_t : t \in \mathbb{N}^*\}, y_t \in \mathcal{Y}$ e inoltre si assume che le osservazioni siano indipendenti tra loro dato il processo $\{s_t : t \in \mathbb{N}\}$. Il modello del sistema è rappresentato dall'equazione di transizione $p(s_t|s_{t-1})$ mentre il modello di osservazione è espresso dalla densità marginale $p(y_t|s_t)$. La conoscenza al tempo t viene indicata con $s_{0:\,t} \doteq \{s_0, \ldots, s_t\}$ e $y_{1:\,t} \doteq \{y_1, \ldots, y_t\}$ che sono rispettivamente le sequenze degli stati e delle osservazioni fino al tempo t.

Lo scopo è quello di stimare ricorsivamente nel tempo la densità a posteriori $p(s_{0:\,t}|y_{1:\,t})$ o più frequentemente la densità marginale $p(s_t|y_{1:\,t})$ e una funzione di aspettazione, indicata in modo del tutto generale dall'espressione

$$I(f_t) \doteq \int f_t(s_{0:\,t}) p(s_{0:\,t}|y_{1:\,t}) ds_{0:\,t}$$

dove esplicitando f_t si ottiene una particolare aspettazione: ad esempio scegliendo $f_t(s_{0:\,t}) = s_{0:\,t}$ si ottiene la media statistica.

Al tempo t, possiamo stimare la densità a posteriori utilizzando la Regola di Bayes (A.3):

$$p(s_{0:\,t}|y_{1:\,t}) = \frac{p(y_{1:\,t}|s_{0:\,t})p(s_{0:\,t})}{p(y_{1:\,t})} \quad ,$$

dove $p(y_{1:\,t}) = \int p(y_{1:\,t}|s_{0:\,t})p(s_{0:\,t})ds_{0:\,t}$ dal Teorema della probabilità totale (A.2). Si ottiene poi un'equazione ricorsiva:

$$p(s_{0:\,t}|y_{1:\,t}) = p(s_{0:\,t-1}|y_{1:\,t-1}) \frac{p(y_t|s_t)p(s_t|s_{t-1})}{p(y_t|y_{1:\,t-1})} \quad .$$

Per la densità marginale $p(s_t|y_{1:\,t})$ possiamo scrivere dividendo in due passi:

$$\text{Predizione :} \qquad p(s_t|y_{1:\,t-1}) = \int p(s_t|s_{t-1}) p(s_{t-1}|y_{1:\,t-1}) ds_{t-1} \ , \tag{3.1}$$

$$\text{Aggiornamento :} \qquad p(s_t|y_{1:\,t}) = \frac{p(y_t|s_t)p(s_t|y_{1:\,t-1})}{p(y_t|y_{1:\,t-1})} \quad , \tag{3.2}$$

dove nell'equazione (3.1) è stata usata la proprietà di Markovianità dello stato $p(s_t|s_{t-1}, y_{1:\,t-1}) = p(s_t|s_{t-1})$ mentre nella (3.2) la costante di normalizzazione $p(y_t|y_{1:\,t-1}) = \int p(y_t|s_t)p(s_t|y_{1:\,t-1})ds_t$ dipende dal modello di osservazione $p(y_t|s_t)$.
Osserviamo che l'equazione (3.1) applica il modello del sistema $p(s_t|s_{t-1})$ alla densità $p(s_{t-1}|y_{1:\,t-1})$ nota al tempo precedente $t-1$. Otteniamo così la densità $p(s_t|y_{1:\,t-1})$ che non tiene ancora conto dell'osservazione y_t. Quando quest'ultima diventa disponibile, con l'equazione (3.2) aggiorniamo la densità ottenuta dalla predizione e calcoliamo la densità a posteriori $p(s_t|y_{1:\,t})$ al tempo t.

Le equazioni (3.1) e (3.2) sono la base della soluzione ottima al problema di stima Bayesiana. Si vuole sottolineare che queste equazioni derivano semplicemente dalla Regola di Bayes e dal Teorema della probabilità totale (vedi appendice A); ma esse nascondono una complessità notevole: in genere è difficile ottenere una espressione analitica delle densità coinvolte perché gli integrali sono complessi e di elevata dimensione[1], in genere legata alla complessità e alla elevata dimensione dello spazio delle osservazioni $\mathcal{Y}$.

3.2 Metodi di soluzione del Filtro di Bayes

Le soluzioni ottime al problema di stima Bayesiano sono il Filtro di Kalman e i Metodi Grid-based [Arulampalam *e altri*, 2002]. Ciascuno di questi metodi è applicabile solo se sono verificate particolari ipotesi semplificative sul sistema studiato, che spesso però risultano troppo restrittive per i sistemi reali. È possibile estendere questi due metodi rilassando le ipotesi: si ottengono così soluzioni approssimate. Altrimenti è possibile utilizzare il Metodo di Monte Carlo. I diversi metodi di soluzione del Filtro di Bayes si differenziano soprattutto sul modo di rappresentare la densità di probabilità e sulle ipotesi necessarie per la loro applicazione.

3.2.1 Filtri di Kalman

Il Filtro di Kalman assume che il processi coinvolti nel problema siano gaussiani quindi gaussiana è anche la densità a posteriori e il modello di osservazione; inoltre assume che la relazione tra lo stato s_t del sistema e le osservazioni $y_{1:\,t}$ sia lineare. La soluzione si ottiene calcolando ricorsivamente media e covarianza delle densità secondo le equazioni (3.1) e (3.2).

[1] Ad esempio gli integrali necessari per il calcolo delle costanti di normalizzazione o dell'aspettazione [Doucet *e altri*, 2001a].

Notiamo che le ipotesi necessarie all'applicazione del Filtro di Kalman risultano troppo restrittive per molti sistemi reali, che non sono quasi mai lineari e inoltre possono essere descritti solo da densità multimodali, non rappresentabili con una singola gaussiana.
Attenuando l'ipotesi di linearità si ottiene il Filtro di Kalman Esteso (EKF) in cui le relazioni non lineari vengono linearizzate, espandendole in serie di Taylor e considerando solo il primo termine; la soluzione poi è quella del Filtro di Kalman classico con l'ipotesi della gaussianità. Questa soluzione mantiene il vantaggio della semplicità ma fornisce risultati inadeguati se le densità coinvolte sono multimodali o comunque non gaussiane.

3.2.2 Metodi basati sulle griglie

Questo metodo fornisce una soluzione ottima nell'ipotesi che lo spazio degli stati del sistema $\mathcal{S}$ sia discreto e finito e sia nota e calcolabile la densità $p(y_t|s_t)$. Infatti in questo caso le densità sono rappresentate con funzioni costanti a tratti e gli integrali presenti nelle equazioni (3.1) e (3.2) diventano delle sommatorie finite e quindi sono calcolabili facilmente [Arulampalam *e altri*, 2002]. L'ipotesi però è troppo restrittiva poiché lo spazio degli stati è spesso di dimensione infinita.

Se lo spazio degli stati è continuo ma può essere decomposto in un numero finito di celle allora si può applicare una approssimazione detta "Grid-based" che considera come spazio degli stati finito l'insieme delle celle, su cui calcolare le densità[2]. Affinché questo metodo dia una buona approssimazione è necessario che la griglia di celle sia sufficientemente densa ma questo porta a carichi computazionali elevati che escludono un funzionamento in tempo reale (per l'applicazione alla localizzazione di robot mobili vedi [Fox *e altri*, 1999b] e [Gutmann e Fox, 2002]). Il problema è che il calcolo della densità a posteriori coinvolge sempre tutte le celle, anche quelle con probabilità irrilevante che non sono interessanti e quindi questo metodo spreca risorse inutilmente. Inoltre lo spazio degli stati e la precisione sono predefiniti e quindi il metodo basato sulle griglie risulta poco flessibile.

3.3 Filtri a particelle

Una soluzione approssimata alla stima delle distribuzioni (3.1) e (3.2) può essere ottenuta ricorrendo all'algoritmo "Sequential Importance Sampling" (SIS) che appartiene alla categoria dei Metodi di Monte Carlo Sequenziali.
L'idea chiave è quella di rappresentare la densità a posteriori con un insieme

[2]Per semplicità spesso come stati si considerano i centri delle celle.

di campioni casuali ("samples") con associato un peso ed effettuare la stima in base a questi campioni e al loro peso. Al crescere del numero di campioni considerati si ottiene una rappresentazione equivalente della funzione di densità.

L'algoritmo SIS è alla base di molti metodi di Monte Carlo che sono stati sviluppati negli ultimi anni in ambiti specifici dove sono chiamati con nomi diversi: Filtri a particelle [Carpenter *e altri*, 1999], Filtri Bootstrap [Gordon *e altri*, 1993] e "Condensation algorithm" [Isard e Blake, 1998; Dellaert *e altri*, 1999]. La convergenza di questo tipo di algoritmi è stata dimostrata e si può trovare in dettaglio in [Crisan, 2001; Crisan e Doucet, 2000].

3.3.1 "Sequential Importance Sampling"

Di seguito si userà la seguente notazione: $\{s_{0:\,t}^{(i)} : i = 1, \ldots, N\}$ è l'insieme di N campioni casuali, indipendenti e identicamente distribuiti (i.i.d.); essi vengono chiamati anche particelle[3] e sono ottenuti in accordo alla densità $p(s_{0:\,t}|y_{1:\,t})$; con $\{s_{0:\,t}^{(i)}, \omega_t^{(i)} : i = 1, \ldots, N\}$ si indica l'insieme delle particelle con il relativo peso. Per ipotesi i pesi sono sempre normalizzati in modo che $\sum_i \omega_t^{(i)} = 1$.
La densità a posteriori può essere quindi approssimata in modo discreto con i campioni pesati:

$$p(s_{0:\,t}|y_{1:\,t}) \approx \sum_{i=1}^{N} \omega_t^{(i)} \delta(s_{0:\,t} - s_{0:\,t}^{(i)}) \quad . \tag{3.3}$$

I pesi $\omega_t^{(i)}$ sono calcolati secondo il principio dell'"Importance Sampling" [Doucet, 1998; Doucet *e altri*, 2000; Arulampalam *e altri*, 2002] che presentiamo di seguito e che permette di superare la difficoltà legata al estrazione di campioni dalle densità a posteriori coinvolte nel Filtro di Bayes.
Consideriamo la densità di probabilità $\pi(s)$ tale che $p(s) \propto \pi(s)$ e che sia possibile valutare $\pi(s)$. Se inoltre $q(s)$ (detta *Funzione di importanza*) è una densità da cui è possibile estrarre campioni $s^{(i)} \sim q(s)$ allora la densità $p(s)$ può essere approssimata tramite campioni con:

$$p(s) \approx \sum_{i=1}^{N} \omega^{(i)} \delta(s - s^{(i)}) \quad , \tag{3.4}$$

dove il peso normalizzato della particella i-esima è:

$$\omega^{(i)} \propto \frac{\pi(s^{(i)})}{q(s^{(i)})} \quad , \tag{3.5}$$

[3] Di seguito utilizzeremo indifferentemente i termini campioni e particelle.

e quindi riferendoci all'equazione (3.3), se $s^{(i)}_{0:\,t} \sim q(s_{0:\,t}|y_{1:\,t})$ scriviamo:

$$\omega^{(i)}_t \propto \frac{p(s^{(i)}_{0:\,t}|y_{1:\,t})}{q(s^{(i)}_{0:\,t}|y_{1:\,t})} \quad . \tag{3.6}$$

Nel caso sequenziale si vuole approssimare ad ogni iterazione $p(s_{0:\,t}|y_{1:\,t})$ con un nuovo insieme di campioni avendo a disposizione una approssimazione di $p(s_{0:\,t-1}|y_{1:\,t-1})$ dal passo precedente. La funzione di importanza viene scelta in modo che fattorizzi come:

$$q(s_{0:\,t}|y_{1:\,t}) = q(s_t|s_{0:\,t-1}, y_{1:\,t})q(s_{0:\,t-1}|y_{1:\,t-1}) \quad ; \tag{3.7}$$

da questa si ottiene un'equazione ricorsiva per il calcolo dei pesi:

$$\omega^{(i)}_t \propto \omega^{(i)}_{t-1} \frac{p(y_t|s^{(i)}_t)p(s^{(i)}_t|s^{(i)}_{t-1})}{q(s^{(i)}_t|s^{(i)}_{0:\,t-1}, y_{1:\,t})} \quad . \tag{3.8}$$

Nel caso più comune siamo interessati solo alla stima filtrata di $p(s_t|y_{1:\,t})$: questo permette di tralasciare i campioni e le osservazioni dei passi precedenti e semplifica $q(s_t|s_{0:\,t-1}, y_{1:\,t}) = q(s_t|s_{t-1}, y_t)$; otteniamo i pesi:

$$\omega^{(i)}_t \propto \omega^{(i)}_{t-1} \frac{p(y_t|s^{(i)}_t)p(s^{(i)}_t|s^{(i)}_{t-1})}{q(s^{(i)}_t|s^{(i)}_{t-1}, y_t)} \quad , \tag{3.9}$$

e da questi la densità a posteriori:

$$p(s_t|y_{1:\,t}) \approx \sum_{i=1}^{N} \omega^{(i)}_t \delta(s_t - s^{(i)}_t) \quad . \tag{3.10}$$

L'algoritmo "Sequential Importance Sampling" (SIS) consiste in una propagazione ricorsiva di pesi e campioni secondo l'equazione (3.9) all'arrivo di una misura. Al crescere di $N \to \infty$ l'approssimazione (3.10) si avvicina alla densità di probabilità a posteriori quindi risolve il Filtro di Bayes.

3.3.2 Ricampionamento

Il Filtro a particelle SIS ha un problema detto *Fenomeno di degenerazione*: dopo poche iterazioni dell'algoritmo solo un campione ha peso non trascurabile. È stato dimostrato in [Doucet, 1998] che la degenerazione avviene sempre poiché la varianza dei pesi cresce nel tempo[4]. A causa della degenerazione dopo poche iterazioni gran parte del carico computazionale viene

[4]Pesi calcolati con il principio di Campionamento per Importanza ("Importance Sampling").

sprecato per particelle che sono irrilevanti per la stima della densità $p(s_t|y_{1:t})$. Inoltre questa densità viene rappresentata in modo non adeguato poiché si basa su un unica particella (equivale a considerare un unico modo della densità multimodale e tralasciarne altri che però potrebbero essere rilevanti) [Arulampalam *e altri*, 2002; Doucet *e altri*, 2001a].
Per risolvere del problema della degenerazione si può utilizzare un numero di campioni N elevato: questo comporta un aumento non accettabile del tempo di elaborazione, soprattutto per sistemi con vincoli di tempo reale, come nel caso di localizzazione di robot. Un'altro modo è quello di scegliere opportunamente la funzione di importanza; sottolineiamo come la scelta della funzione di importanza sia uno dei passi cruciali nella progettazione di un filtro a particelle. Per approfondire questi argomenti si veda [Doucet, 1998] e [Arulampalam *e altri*, 2002].

Un metodo ulteriore per ridurre la degenerazione è quello del Ricampionamento ("Resampling"); esso sarà utilizzato nell'algoritmo di localizzazione che introdurremo nella sezione 3.4. Lo scopo del ricampionamento è quello di eliminare i campioni con peso basso per concentrare la computazione su quelli più rilevanti che hanno peso alto [Gordon *e altri*, 1993].
In pratica ad ogni iterazione dell'algoritmo SIS si ricampiona N volte con sostituzione dalla densità approssimata $p(s_t|y_{1:t}) \approx \sum_{i=1}^{N} \omega_t^{(i)} \delta(s_t - s_t^{(i)})$.
Il campione $s_t^{(i)}$ viene selezionato $N_t^{(i)}$ volte in modo che $Pr(\widetilde{s}_t^{(j)} = s_t^{(i)}) \approx \omega_t^{(i)}$.
Questo procedimento produce un insieme di campioni indipendenti ed identicamente distribuiti (i.i.d.) $\{\widetilde{s}_t^{(j)} : j = 1, \ldots, N\}$ e quindi il peso associato ad ogni campione viene scelto pari a $\widetilde{\omega}_t^{(j)} = 1/N$.

L'algoritmo base di ricampionamento utilizza la ricerca binaria [Carpenter *e altri*, 1999; Kitagawa, 1996]. Esso è stato utilizzato nell'algoritmo di localizzazione di Monte Carlo sviluppato in [Wolf, 2001]: questa scelta non è efficiente poiché il ricampionamento di un insieme di N particelle richiede $O(N \log N)$ operazioni mentre è possibile eseguire questa procedura in $O(N)$ operazioni. Tra i metodi proposti abbiamo fatto riferimento al *Ricampionamento sistematico* nella versione generale [Carpenter *e altri*, 1999] e nelle versioni *stratificato* e *deterministico* proposto da Kitagawa e Arulampalam *e altri*. Abbiamo scelto di usare quest'ultimo nell'implementazione dell'algoritmo di localizzazione perché risulta semplice da implementare ed efficiente [Kitagawa, 1996; Arulampalam *e altri*, 2002; Fearnhead, 1998].

Con l'introduzione del passo di ricampionamento otteniamo l'algoritmo "Sampling Importance Resampling" (SIR) detto anche Filtro a particelle o Algoritmo Bootstrap [Isard e Blake, 1998]:

1. la funzione di importanza q è la densità a priori $p(s_t|s_{t-1})$;

2. deve essere possibile valutare la funzione $p(y_t|s_t)$;

3. il campionamento viene eseguito ad ogni passo.

Per i pesi vale che $\omega_t^{(i)} \propto \omega_{t-1}^{(i)} p(y_t|s_t^{(i)})$ ma poiché il ricampionamento viene eseguito ad ogni passo e rende uniforme il peso dei campioni cioè $\omega_{t-1}^{(i)} = 1/N$ allora

$$\omega_t^{(i)} \propto p(y_t|s_t^{(i)}) \quad . \tag{3.11}$$

Il Ricampionamento è schematizzato con pseudo-codice nell'Algoritmo 3.1 mentre il generico Filtro a particelle è descritto nell'Algoritmo 3.2.

Abbiamo evidenziato che il ricampionamento riduce gli effetti della degenerazione eliminando i campioni irrilevanti. Esso però introduce altri problemi pratici. Per primo riduce la possibilità di parallelizzare l'algoritmo perché tutti i campioni devono essere combinati. Inoltre, il ricampionamento provoca un impoverimento dei campioni perché quelli con peso elevato vengono selezionati e inseriti più volte nel nuovo insieme, che quindi contiene campioni ripetuti: questo può portare al collassamento in un singolo campione se il disturbo del sistema è molto basso. Ricordiamo infatti che i filtri a particelle sono adatti a sistemi dinamici in cui sia presente un disturbo non nullo. Nella localizzazione di robot mobili l'errore sui dati non è quasi mai trascurabile perché i sensori sono rumorosi e quindi il Filtro a particelle può essere usato con successo.

3.4 Localizzazione con il Metodo di Monte Carlo

In questa sezione presentiamo il problema della localizzazione di robot mobili affrontata con il metodo probabilistico presentato prima.
Come abbiamo già spiegato nella sezione 3.1, nello studio dei sistemi dinamici spesso è necessaria la stima dello stato non noto del sistema a partire da informazioni parziali e/o non accurate per la presenza di errori o disturbi. Se si affronta il problema in senso probabilistico e se si ha a disposizione una conoscenza a priori e un legame che lega lo stato alle informazioni si può ricorrere all'*Approccio Bayesiano*: nel caso in cui le informazioni arrivano in modo sequenziale nel tempo, esso permette una stima ricorsiva cioè ripetuta all'arrivo di ciascuna informazione; in questo caso si parla di *Filtro Bayesiano*. L'ipotesi necessaria per applicare questo metodo di stima è che l'ambiente sia Markoviano, cioè che i dati del passato e del futuro siano stocasticamente indipendenti se è noto lo stato attuale del sistema.

Algoritmo 3.1 RICAMPIONAMENTO ($\{s^{(i)}, \omega^{(i)}\}_{i=1}^{N}$)

Parametri: l'insieme di campioni con peso $\{s^{(i)}, \omega^{(i)}\}_{i=1}^{N}$
Risultati: l'insieme di campioni aggiornato $\{\widetilde{s}^{(j)}, \widetilde{\omega}^{(j)}\}_{j=1}^{N}$

- calcolo la CDF -
$cdf_1 = \omega_1$
for $i = 2 : N$ **do**
  $cdf_i = cdf_{i-1} + \omega^{(i)}$
end for
- inizio dal primo campione $s^{(1)}$ -
$i = 1$
- punto di partenza casuale -
$u_1 \sim \mathbb{U}[0, 1/N]$
for $j = 1 : N$ **do**
  - mi muovo lungo la CDF -
  $u_j = u_1 + \frac{1}{N}(j-1)$
  while $u_j > cdf_i$ **do**
    $i = i + 1$
  end while
  $\widetilde{s}^{(j)} = s^{(i)}$
  $\widetilde{\omega}^{(j)} = 1/N$
end for

Algoritmo 3.2 FILTRO A PARTICELLE generico ($\{s_{t-1}^{(i)}, \omega_{t-1}^{(i)}\}_{i=1}^{N}, y_t$)

Parametri: l'insieme di campioni $\{s_{t-1}^{(i)}, \omega_{t-1}^{(i)}\}_{i=1}^{N}$ che approssima la densità $p(s_{t-1}|y_{1:\,t-1})$ al tempo precedente $t-1$ e l'osservazione attuale y_t
Risultati: il nuovo insieme di campioni $\{\widetilde{s}_t^{(i)}, \widetilde{\omega}_t^{(i)}\}_{i=1}^{N}$ che approssima la densità $p(s_t|y_{1:\,t})$ al tempo t

- passo di Campionamento per Importanza -
for $i = 1 : N$ **do**
  campiona $s_t^{(i)} \sim p(s_t|s_{t-1}^{(i)})$
  assegna il peso $\omega_t^{(i)} = p(y_t|s_t^{(i)})$ secondo l'equazione (3.11)
end for
normalizza i pesi $\omega_t^{(i)}$
- passo di Ricampionamento con l'Algoritmo 3.1 -
$\{\widetilde{s}_t^{(j)}, 1/N\}_{j=1}^{N} = \text{RICAMPIONAMENTO}(\{s_t^{(i)}, \omega_t^{(i)}\}_{i=1}^{N})$

Nel caso della localizzazione di robot mobili il sistema dinamico è costituito dall'ambiente e dal robot che si muove in esso. Notiamo che il robot è un elemento dinamico dell'ambiente in quanto si muove e in generale effettua delle operazioni su di esso (ad esempio può spostare oggetti); ma l'ambiente stesso può essere considerato dinamico per la presenza di elementi, indipendenti dal robot, che hanno la possibilità di muoversi: l'esempio più importante è quello di persone che si muovono nell'ambiente [Dellaert *e altri*, 1999] oppure di altri robot (ad esempio nella competizione RoboCup [Lenser e Veloso, 2000]).

Lo stato del sistema è la posizione del robot nell'ambiente che spesso è specificata da due coordinate in uno spazio Cartesiano due-dimensionale e da una coordinata polare che indica l'orientazione del robot.
Le informazioni a disposizione vengono acquisite dai sensori tipici montati a bordo di robot mobili e sono:

- i dati forniti dall'odometro: vengono misurati dai sensori collegati alle ruote dei robot [Borghi, 1997]; l'odometria è poco affidabile in quanto soggetta a errori che, anche se piccoli, si accumulano e provocano un errore di stima della traiettoria che può diventare notevole. Gli errori sono legati allo slittamento delle ruote ma anche a salti provocati dal passaggio tra due superfici diverse [Fox *e altri*, 1999b] o provocati dallo scontro del robot con un ostacolo o con altri robot [Lenser e Veloso, 2000]. Essendo l'odometria un sistema di localizzazione relativo allora l'errore commesso cresce nel tempo e non permette una localizzazione accurata quindi è necessaria una tecnica per correggere questi errori;

- i dati del sistema di visione. Vengono utilizzate tipicamente telecamere prospettiche o omnidirezionali. I disturbi sono legati alla presenza di oggetti in movimento, occlusioni, condizioni di luminosità variabili e anche al fatto che per obiettivi di efficienza, l'image matching viene effettuato sulla base di caratteristiche distintive estratte dalle immagini: questo comporta una minor complessità ma può far perdere delle caratteristiche importanti;

- i dati acquisiti da sensori di prossimità quali laser o sonar: con questi sensori vengono misurate le distanze dagli oggetti vicini che poi vengono confrontate con quelle di una mappa nota a priori. I disturbi sono legati alla presenza di oggetti nell'ambiente non descritti nella mappa, che rendono difficile determinare il giusto matching.

3.4.1 Calcolo del belief

Il filtro di Bayes risulta quindi lo strumento adatto per affrontare il problema della localizzazione di robot mobili dotati di visione, sonar o laser[5]. Secondo la notazione tipica della letteratura robotica e dell'Intelligenza Artificiale la densità di probabilità a posteriori sulla posizione del robot viene chiamata con il nome inglese "belief"[6] [Russell e Norving, 1995].

Di seguito si utilizza la seguente notazione:

- $\mathcal{S}$ è lo spazio degli stati; in genere è dato da $\mathcal{S} = \mathbb{R} \times \mathbb{R} \times [-180, 180)$;
- $s_t = (x_t, y_t, \theta_t)$ è lo stato al tempo $t \geq 0$: x_t e y_t sono le coordinate cartesiane nello spazio, in cm o mm mentre θ_t è l'orientazione del robot in gradi;
- a_t è l'odometria rilevata nel movimento dallo stato s_t allo stato s_{t+1}; essa è strettamente legata al tipo di robot usato e in particolare al sistema di movimento e alla sua schematizzazione ad alto livello (vedi sezione 4.1);
- O_t è l'informazione al tempo t rilevata dal sistema di visione; esistono applicazioni che fanno uso di anelli di sensori sonar o "laser range finders";
- $d_{0:t} \doteq \{O_t, a_{t-1}, O_{t-1}, \ldots, a_0, O_0\}$ è l'insieme dei dati sensoriali di odometria e visione fino al tempo t. Notiamo che al tempo t è nota l'odometria fino al tempo precedente $t-1$;
- $Bel(s_t) = p(s_t|d_{0:t}) = p(s_t|O_t, a_{t-1}, \ldots)$ è la densità di probabilità a posteriori sulla posizione del robot, di cui si vuole una stima.

Assumiamo che le osservazioni e le azioni (movimenti) vengano effettuati in sequenza alternata e ripetuta nel tempo. Questo verrà giustificato dal modello del movimento (così chiameremo di seguito il modello del sistema) del robot presentato nel prossimo capitolo. Il Filtro di Bayes stima ricorsivamente il belief a partire dalla conoscenza iniziale del sistema. Nel caso del position tracking la conoscenza iniziale consiste nella densità di probabilità che descrive la posizione nota del robot: nel nostro sistema usiamo

[5] Nel seguito faremo riferimento al caso della visione, ma si ricorda che il metodo è stato applicato inizialmente con sensori laser e sonar.

[6] Abbiamo già detto nell'Introduzione che il termine può essere tradotto in *stato di credenza* [Borghi, 1997] ma di seguito useremo sempre il termine inglese.

una densità gaussiana. Nel caso della localizzazione globale non si ha nessuna informazione sulla posizione del robot e quindi si considera una densità uniforme sullo spazio degli stati.

Abbiamo detto sopra che per applicare il Filtro di Bayes si assume l'ipotesi Markovianità; questa influenza la struttura delle equazioni che descrivono i modelli del movimento e di osservazione. In particolare l'osservazione O_t risulta statisticamente indipendente dalle osservazioni passate e dalle informazioni odometriche se è noto lo stato attuale:

$$p(O_t|s_t, a_{t-1}, \ldots, O_0) = p(O_t|s_t, a_{t-1}, d_{0:t-1}) = p(O_t|s_t) \quad ; \tag{3.12}$$

inoltre noti s_{t-1} e a_{t-1} lo stato s_t è statisticamente indipendente dalle osservazioni e dai dati dell'odometria passati $d_{0:t-1}$:

$$p(s_t|s_{t-1}, a_{t-1}, d_{0:t-1}) = p(s_t|s_{t-1}, a_{t-1}) \quad . \tag{3.13}$$

L'ipotesi (3.12) è giustificata da fatto che le informazioni della scena ottenute dal sensore descrivono lo stato attuale indipendentemente dal percorso passato. L'ipotesi (3.13) è legata al modello del moto secondo il quale la posizione di arrivo di un movimento dipende solo dalla posizione di partenza e dal movimento effettuato.

Ricaviamo ora l'equazione ricorsiva per l'aggiornamento del belief in modo analogo a quanto fatto in generale nella sezione 3.1; essa sarà applicabile ad ogni passo del robot:

$$\begin{aligned} Bel(s_t) = p(s_t|d_{0:t}) &= \frac{p(O_t|s_t, a_{t-1}, \ldots, O_0)p(s_t|a_{t-1}, \ldots, O_0)}{p(O_t|a_{t-1}, \ldots, O_0)} \\ &= \frac{p(O_t|s_t, a_{t-1}, d_{0:t-1})p(s_t|a_{t-1}, d_{0:t-1})}{p(O_t|a_{t-1}, d_{0:t-1})} \quad , \end{aligned} \tag{3.14}$$

dove abbiamo usato semplicemente la Regola di Bayes del tipo (A.5).
Sfruttando l'ipotesi di Markovianità (3.12) otteniamo:

$$Bel(s_t) = \frac{p(O_t|s_t)p(s_t|a_{t-1}, d_{0:t-1})}{p(O_t|a_{t-1}, d_{0:t-1})} \quad .$$

Ora utilizziamo il Teorema della probabilità totale integrando rispetto s_{t-1}:

$$\begin{aligned} p(s_t|a_{t-1}, d_{0:t-1}) &= \int p(s_t|s_{t-1}, a_{t-1}, d_{0:t-1})p(s_{t-1}|a_{t-1}, d_{0:t-1})ds_{t-1} \\ &= \int p(s_t|s_{t-1}, a_{t-1})Bel(s_{t-1})ds_{t-1} \quad , \end{aligned}$$

dove nel secondo passaggio abbiamo usato l'ipotesi di markovianità (3.13) ed inoltre abbiamo riconosciuto la definizione del belief ottenibile dal passo precedente.
L'equazione finale si ottiene definendo $\eta = p(O_t|a_{t-1}, d_{0:t-1})$ costante di normalizzazione:

$$Bel(s_t) = \eta\, p(O_t|s_t) \int p(s_t|s_{t-1}, a_{t-1}) Bel(s_{t-1}) ds_{t-1} \quad , \tag{3.15}$$

che si può spezzare in due parti:

$$\begin{aligned} \textit{Movimento: } Bel^-(s_t) &= \int p(s_t|s_{t-1}, a_{t-1}) Bel(s_{t-1}) ds_{t-1} \quad , \\ \textit{Osservazione: } Bel(s_t) &= \eta\, p(O_t|s_t) Bel^-(s_t) \quad . \end{aligned}$$

L'equazione finale permette di calcolare il belief in modo ricorsivo ad ogni passo a partire da $Bel(s_0) = p(s_0)$; quest'ultimo deve essere opportunamente definito per rappresentare la conoscenza iniziale sulla posizione del robot nei vari tipi di localizzazione; è necessario conoscere l'equazione del *Modello del movimento* $p(s_t|s_{t-1}, a_{t-1})$ che lega due posizioni successive tramite un movimento a_{t-1} e la cui forma deve essere scelta in base allo schema di odometria del robot (vedi sezione 4.1); infine serve l'equazione del *Modello di osservazione* $p(O_t|s_t)$ che esprime la probabilità di fare l'osservazione O_t nella posizione s_t e nel caso d'interesse dipende dal sistema di visione (vedi sezione 4.2).
Osserviamo inoltre che entrambi i modelli sono tempo-invarianti e quindi è possibile omettere l'indicazione del tempo.

3.4.2 Approssimazione con i campioni

L'equazione (3.15) è stata ottenuta in modo abbastanza semplice utilizzando la regola di Bayes e la proprietà di Markovianità. Potrebbe quindi sembrare di facile soluzione ma in realtà se lo spazio degli stati è continuo, come nel caso della localizzazione, la sua soluzione analitica è un compito arduo che risulta difficile da conciliare con le esigenze di efficenza e velocità necessarie in un ambito in tempo reale quale è la navigazione di un robot. Da qui la necessità di risolvere l'equazione in modo approssimato; nell'ambito della robotica mobile sono state utilizzate varie soluzioni già presentate in generale nella sezione 3.2.
I Filtri di Kalman sono stati usati nella localizzazione di robot mobili dotati di "laser range finder": essi forniscono una soluzione molto precisa ed efficiente nel position tracking ma falliscono nella localizzazione globale perché

descrivono adeguatamente solo densità unimodali [Gutmann *e altri*, 1998].
I metodi basati sulle griglie sono stati usati sia per il position tracking che per la localizzazione globale ma richiedono capacità computazionali troppo elevate. Esempi importanti di questi metodi sono quelli dei robot guida Rhino e Minerva [Burgard *e altri*, 1999; Fox *e altri*, 1999a; Thrun *e altri*, 2000]. Inoltre queste tecniche hanno bisogno di una mappa dell'ambiente nota a priori poiché sono state usate con sensori di prossimità quali sonar o laser. Per approfondire questi metodi vedi anche [Gutmann *e altri*, 1998; Gutmann e Fox, 2002; Fox, 2001, 2002].
Negli ultimi anni l'interesse si è rivolto verso l'utilizzo del Metodo di Monte Carlo (MCM) che ha portato allo sviluppo di una forma particolare dell'algoritmo a particelle [Fox *e altri*, 1999b; Dellaert *e altri*, 1999; Lenser e Veloso, 2000; Fox *e altri*, 2001; Wolf *e altri*, 2002a] adatto alla localizzazione di robot mobili; in questo ambito ci si riferisce alla Localizzazione di Monte Carlo (MCL). Rispetto ai metodi citati sopra la MCL ha il vantaggio fondamentale di essere in grado di rappresentare densità di probabilità arbitrarie cioè multimodali e non-lineari risolvendo così la localizzazione globale. Inoltre esso concentra i calcoli sulle zone dello spazio delle configurazioni con alta probabilità e quindi è più efficiente dei metodi che usano griglie predefinite. La tecnica di Monte-Carlo è stata usata inizialmente con sensori di prossimità [Gutmann *e altri*, 1998; Fox *e altri*, 2001] con i quali si cerca di trovare il "match" migliore tra la scansione e la mappa geometrica. È stata utilizzata con sistemi di visione prospettica [Gutmann e Fox, 2002; Lenser e Veloso, 2000; Wolf *e altri*, 2002a] che però continuano ad avere bisogno di una mappa geometrica dell'ambiente per determinare le zone visibili dalla direzione puntata dalla telecamera.
In questa tesi proponiamo di fondere la MCL con il sistema di visione omnidirezionale, presentato nel capitolo 2, sfruttandone le caratteristiche per ridurre il numero di immagini di riferimento ed evitare di usare la mappa geometrica dell'ambiente. Tale metodo verrà descritto dettagliatamente nel capitolo 4. Di seguito presentiamo il metodo generale.

L'idea della MCL è quella di rappresentare il belief $Bel(s)$ tramite un insieme di N campioni pesati e distribuiti secondo $Bel(s)$:

$$Bel(s) \approx \left\{ s^{(i)}, \omega^{(i)} \right\}_{i=1}^{N} \quad , \tag{3.16}$$

dove ogni campione $s^{(i)}$ rappresenta un possibile stato, quindi una possibile posizione del robot. Ad esso viene associato un numero non negativo $\omega^{(i)}$ detto *fattore di importanza* che determina il peso (=importanza) del campione stesso, cioè ne misura la "bontà" come ipotesi di posizione. I fattori di importanza sono sempre normalizzati a uno.

In questo modo si riesce a rappresentare una densità continua di probabilità come $Bel(s)$ tramite un numero finito di campioni che vengono aggiornati con i modelli del moto e di osservazione secondo lo schema tipico dei filtri a particelle.

L'equazione 3.15 viene calcolata in tre passi da sinistra a destra ripetuti per tutti gli N campioni [Fox *e altri*, 2001]:

1. campiona uno stato s_{t-1} da $Bel(s_{t-1})$ estraendo un campione casuale $s_{t-1}^{(i)}$ dall'insieme dei campioni $\{s_{t-1}^{(i)}, \omega_{t-1}^{(i)}\}_{i=1}^{N}$ che rappresenta $Bel(s_{t-1})$; quindi l'estrazione avviene in base alla densità discreta definita dai fattori di importanza, cioè in base a $\sum_i \omega_{t-1}^{(i)} \delta(s - s_{t-1}^{(i)})$;

2. applica il movimento a_{t-1} al campione $s_{t-1}^{(i)}$ ed estrai il nuovo campione $s_t^{(j)}$ dalla densità $p(s_t|s_{t-1}, a_{t-1})$. Si ottiene così la densità predittiva $p(s_t|s_{t-1}, a_{t-1})Bel(s_{t-1})$;

3. assegna il peso $\omega_t^{(j)} \propto p(O_t|s_t^{(j)})$ al campione $s_t^{(j)}$, dove la proporzionalità indica che è necessaria una normalizzazione. Qui si assegna ad un campione un peso proporzionale alla probabilità di fare l'osservazione O_t nello stato $s_t^{(j)}$.

Facciamo notare che questa è una possibile implementazione dell'algoritmo a particelle, presentata in [Fox *e altri*, 2001; Wolf *e altri*, 2002a] ma ne esistono altre versioni sviluppate in ambiti diversi dalla localizzazione di robot, vedi [Doucet *e altri*, 2001b]. Inoltre ricordiamo che l'algoritmo converge come è stato dimostrato in generale in [Crisan e Doucet, 2000].

In questa versione della MCL abbiamo usato la funzione di importanza $q \doteq p(s_t|s_{t-1}, a_{t-1})Bel(s_{t-1})$, per approssimare la densità a posteriori

$$\frac{p(O_t|s_t)p(s_t|s_{t-1}, a_{t-1})Bel(s_{t-1})}{p(O_t|a_{t-1}, d_{0:t-1})} \quad , \tag{3.17}$$

e quindi i fattori di importanza sono dati da:

$$[p(s_t|s_{t-1}, a_{t-1})Bel(s_{t-1})]^{-1} \frac{p(O_t|s_t)p(s_t|s_{t-1}, a_{t-1})Bel(s_{t-1})}{p(O_t|a_{t-1}, d_{0:t-1})} \quad , \tag{3.18}$$

che risultano proporzionali a $p(O_t|s_t)$ cioè sono calcolabili dall'equazione del modello di osservazione in quanto il termine al denominatore $p(O_t|a_{t-1}, d_{0:t-1})$ è costante rispetto allo stato s_t (e quindi è lo stesso per tutti i campioni) e può essere compreso nella normalizzazione.

Questa strategia MCL è quella di base che viene utilizzata nella localizzazione e che abbiamo implementato in questo lavoro. Come riportato dalla letteratura essa da buoni risultati sia nella localizzazione globale che nel position tracking.
Sono state sviluppate delle versioni ulteriori che fanno uso di funzioni di importanza particolari [Fox *e altri*, 2001]; in esse si introduce una fase ulteriore per generare dei campioni in base alla densità del modello di osservazione [Lenser e Veloso, 2000]; questo permette di superare le difficoltà soprattutto nel problema del kidnapped robot quando non vengono generati campioni nelle vicinanze della posizione vera e quindi la localizzazione fallisce.
Nel prossimo capitolo presenteremo la soluzione che abbiamo proposto noi per la risolvere il problema del robot rapito. Essa si basa sulla localizzazione topologica del sistema di visione omnidirezionale che risulta molto più accurata rispetto al caso in cui venga utilizzato un sensore prospettico [Wolf *e altri*, 2002a; Lenser e Veloso, 2000]. Possiamo quindi usare le informazioni di visione per generare dei campioni che consentono di rilocalizzare il robot dopo un rapimento. Inoltre sfruttando la localizzazione gerarchica possiamo decidere di prendere in considerazione un numero maggiore di ipotesi semplicemente usando meno frequenze nel calcolo delle similarità, ottenendo anche un minore carico computazionale.
I risultati sperimentali presentati nel capitolo 5 confermano la validità del nostro metodo nell'affrontare il robot rapito; vedremo inoltre che la nostra tecnica permette di migliorare il comportamento del nostro sistema di localizzazione anche durante la fase di localizzazione globale del robot.

Capitolo 4

Implementazione

In questo capitolo presenteremo la nostra implementazione del sistema di localizzazione di Monte Carlo. In particolare verranno presentati i modelli di descrizione del movimento del robot, il modello di osservazione per l'assegnazione dei pesi ai campioni e la strategia che abbiamo proposto per risolvere il problema del robot rapito. Infine verrà descritto il software realizzato per implementare la MCL.

4.1 Modello del movimento

4.1.1 Robot di riferimento

Per implementare il sistema di localizzazione di Monte Carlo abbiamo fatto riferimento ad un robot mobile reale chiamato *Barney*; questo robot è disponibile presso lo IAS-Lab, "Intelligent Autonomous Systems Laboratory" dell'Università di Padova. Esso viene utilizzato per la competizione *RoboCup* dalla squadra locale *Artisti Veneti* ed è stato costruito per questo scopo.
Abbiamo inserito alcune informazioni tecniche in appendice B. Qui ricordiamo che Barney è un robot olonomo, cioè è in grado di muoversi in ogni direzione senza dover prima ruotare.
Il sistema di visione di Barney è un di tipo omnidirezionale formato da una telecamera montata in verticale e da un particolare specchio che si può vedere in Figura B.2. Questo sistema permette di avere una visione completa, a 360°, dell'ambiente circostante con l'acquisizione di una sola immagine.

Altri autori hanno fatto riferimento ad un robot non olonomo e dotato di una telecamera prospettica [Wolf, 2001; Wolf *e altri*, 2002a]. Come abbiamo già osservato in precedenza questo sistema sensoriale ha un raggio di visione limitato, quindi servono più immagini a diverse orientazioni per avere una

visione completa dell'ambiente da una certa posizione. Per questo motivo il robot è costretto ad effettuare un numero maggiore di rotazioni, anche sul posto, per avere una conoscenza completa della scena. In genere questo porta ad un maggiore dell'errore di odometria.
Diversamente il nostro robot olonomo e con telecamera omnidirezionale riduce al minimo le rotazioni, eliminando così una fonte di errore odometrico. La nostra scelta ci ha portato a sviluppare un schema diverso di movimento. Nel seguito presenteremo i modelli che descrivono il moto del robot e le informazioni di visione.

4.1.2 Movimento del robot

Durante il normale funzionamento il robot riceve i comandi di movimento dal sistema di guida che lo conduce nella navigazione da una posizione ad un'altra. Il moto tra due posizioni dell'ambiente non avviene quasi mai in linea retta a causa dello slittamento e delle differenti velocità delle ruote. In generale il moto del robot può essere considerato continuo mentre l'odometria discretizza il moto rappresentandolo come più movimenti in successione che passano attraverso delle posizioni intermedie. In ciascuna posizione intermedia il robot acquisisce le informazioni dai sensori dell'odometria e dal sensore di visione (per quest'ultimo vedi sezione 4.2).
Ricordiamo che gli odometri più recenti utilizzano due encoder ottici accoppiati con assi rotanti come l'asse delle ruote o quello dei motori [Borghi, 1997]. Le informazioni odometriche sono ottenute campionando a intervalli regolari il rotolamento delle ruote e questo produce un descrizione discretizzata del movimento del robot, come abbiamo sottolineato sopra.

Facendo riferimento alla Figura 4.1 il movimento reale e continuo del robot tra due posizioni $s_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$ e $s_t = (x_t, y_t, \theta_t)$ viene descritto da una terna (α_u, T, θ_f):

- un *angolo di uscita* α_u dalla posizione s_{t-1} verso s_t, rispetto all'orientazione iniziale θ_{t-1} (che il robot prende come suo riferimento); questo significa che il robot si muove nella direzione simulata $\theta_T = \theta_{t-1} + \alpha_u$ nel sistema di riferimento assoluto. Poiché il robot a cui ci riferiamo è olonomo non è necessaria alcuna rotazione iniziale per allinearsi all'angolo di uscita;

- una *traslazione* T in linea retta nella direzione θ_T. La traslazione T è misurata in cm o mm;

- una *rotazione* θ_f rispetto all'orientazione iniziale θ_{t-1}: secondo lo schema il robot arriva sul punto di destinazione con la stessa orientazione

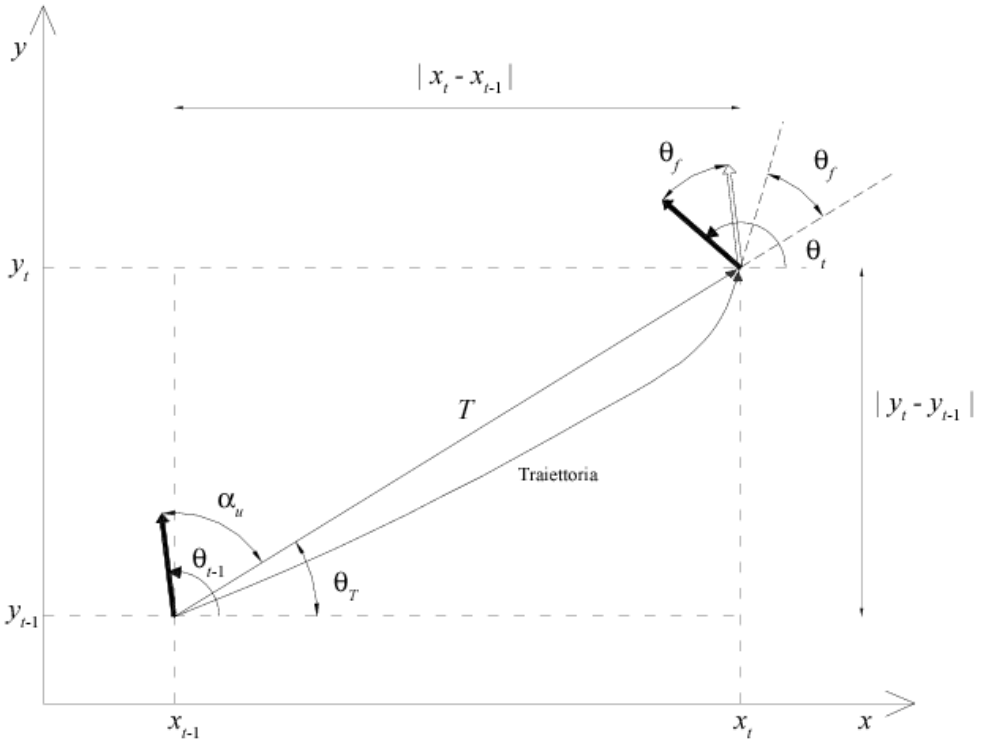


Figura 4.1: Schema del movimento del robot tra due posizioni $s_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$ e $s_t = (x_t, y_t, \theta_t)$: la traiettoria reale è continua ma viene descritta da una terna (α_u, T, θ_f) cioè da un angolo di uscita α_u, una traslazione T e una rotazione θ_f.

di partenza θ_{t-1}; nella realtà l'orientazione sarà cambiata a causa della traiettoria non rettilinea; il robot quindi simula una rotazione di θ_f rispetto all'orientazione iniziale, arrivando così alla posizione finale s_t.

Questo schema di movimento può essere confrontato con quello sviluppato in [Wolf, 2001]: qui il movimento viene descritto con due rotazioni e una traslazione: la prima rotazione sulla posizione di partenza serve per allineare il robot non olonomo alla direzione verso la posizione di destinazione; dopo la traslazione è prevista una rotazione sulla posizione di arrivo per allineare il robot all'orientazione voluta; osserviamo che questa orientazione finale è legata alla scena che si vuole acquisire con la telecamera prospettica.

Ritornando al nostro modello l'angolo θ_T della direzione simulata tra i due punti $s_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})$ e $s_t = (x_t, y_t, \theta_t)$ viene calcolato come l'angolo

tra $s_{t-1} = (x_{t-1}, y_{t-1})$ e $s_t = (x_t, y_t)$ nel modo seguente:

$$\theta_T = \begin{cases} 0 & \text{se} \quad T = 0 \quad ; \\ \arccos((x_t - x_{t-1})/T) & \text{se} \quad y_t > y_{t-1} \quad ; \\ -\arccos((x_t - x_{t-1})/T) & \text{se} \quad y_t \leq y_{t-1} \quad . \end{cases}$$

Qui abbiamo utilizzato la convezione che la funzione arccos restituisca il valore dell'angolo nell'intervallo $[0, 180]$ espresso in gradi.
La traslazione T viene calcolata come distanza euclidea tra i due punti di coordinate (x_{t-1}, y_{t-1}) e (x_t, y_t):

$$T = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2} \quad .$$

4.1.3 Modello di predizione

Abbiamo già osservato che il movimento rilevato del robot è soggetto a errori: l'odometria riportata dai sensori delle ruote non fornisce dei dati sufficientemente affidabili per poter essere utilizzati direttamente per la localizzazione. Possiamo classificare le sorgenti di errore come sistematiche o non sistematiche [Borghi, 1997]. Tra le prime ricordiamo:

- scorretta calibrazione dei parametri dell'algoritmo odometrico: ad esempio il diametro delle ruote o la loro distanza;
- diametro diverso delle ruote;
- limitata risoluzione dell'encoder che misura il rotolamento delle ruote;
- elevato periodo di campionamento.

Tra le sorgenti di tipo non sistematico ricordiamo:

- superficie del suolo ruvida;
- incertezza sulla reale distanza fra i punti di contatto delle ruote con il suolo;
- slittamento dovuto a eccessiva accelerazione;
- slittamento nel passaggio tra superfici di materiale diverso;
- scontro del robot con ostacoli o con altri robot.

Nello schema di movimento del robot dobbiamo considerare l'incertezza legata agli errori. Conviene schematizzare solo gli errori di tipo sistematico in quanto quelli non sistematici sono difficili da riconoscere e difficili da modellare [Lenser e Veloso, 2000]. Gli errori sistematici, anche se piccoli non possono essere trascurati poiché si accumulano e portano a errori notevoli. È necessario quindi considerare la presenza di questi errori nella definizione del *Modello di predizione* $p(s_t|s_{t-1}, a_{t-1})$ (è il modello che nel capitolo 3 è stato chiamato Modello del movimento).

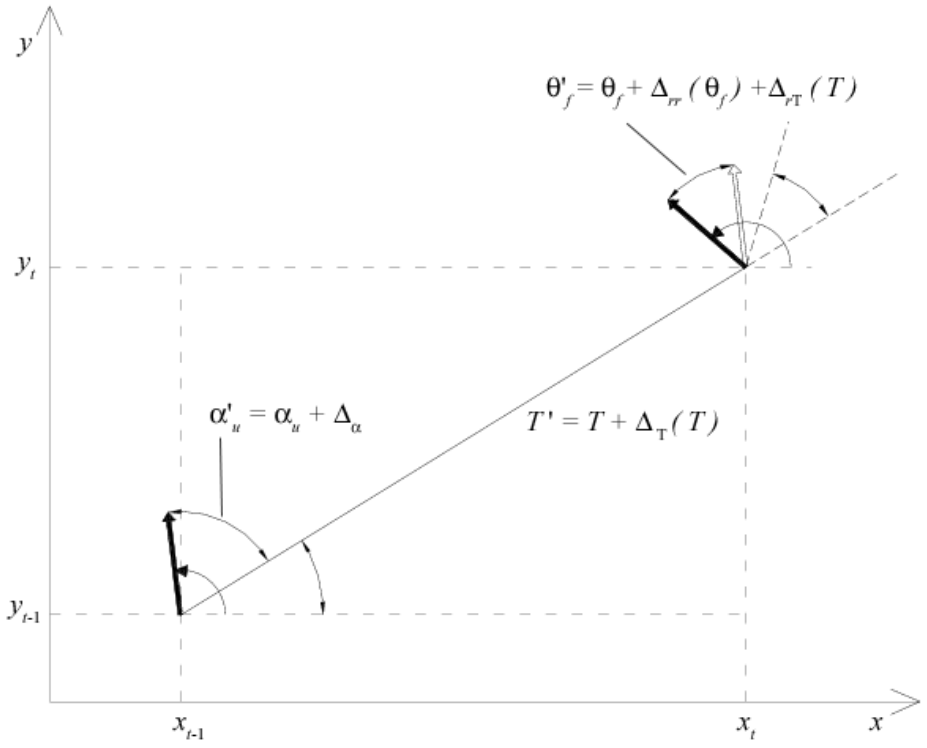


Figura 4.2: Modello degli errori di odometria da cui deriviamo il modello di predizione.

Consideriamo quindi solo gli errori di odometria sistematici. Essi sono legati direttamente al moto del robot cioè si verificano solo a seguito di spostamenti e/o rotazioni. Gli errori che possiamo riconoscere nel nostro modello di movimento sono:

- *Errore assoluto sull'angolo di uscita*: è l'errore assoluto Δ_α nella rilevazione dell'angolo di uscita da parte dell'odometria. Esso è assoluto nel senso che non dipende dall'entità dell'angolo.

- *Errore sulla traslazione*: è l'errore $\Delta_{\mathrm{T}}(T)$ sulla distanza percorsa nella traslazione tra i due punti. Esso dipende dalla lunghezza della traslazione.

- *Errore sulla rotazione (dipendente dalla rotazione)*: è l'errore $\Delta_{rr}(\theta)$ sulla rilevazione della rotazione effettuata. Esso dipende dall'entità della rotazione.

- *Errore sulla rotazione (dipendente dalla traslazione)*: è l'errore $\Delta_{r\mathrm{T}}(T)$ sulla rotazione legato alla traslazione T effettuata.

Lo schema di errore è mostrato in Figura 4.2; per il calcolo dei valori effettivi di odometria otteniamo le seguenti equazioni:

$$\begin{aligned} \alpha'_u &= \alpha_u + \Delta_\alpha \quad ; \\ T' &= T + \Delta_{\mathrm{T}}(T) \quad ; \\ \theta' &= \theta + \Delta_{rr}(\theta) + \Delta_{r\mathrm{T}}(T) \quad . \end{aligned}$$

In letteratura molti autori utilizzano un modello di errore di tipo gaussiano [Fox *e altri*, 1999b; Lenser e Veloso, 2000; Wolf, 2001; Wolf *e altri*, 2002a]; si veda [Gutmann *e altri*, 1998; Gutmann e Fox, 2002] per un'introduzione più generale. Ci sembra quindi ragionevole assumere l'ipotesi che ciascuno degli errori introdotti sopra sia modellabile con un errore di tipo gaussiano di media nulla e con le seguenti deviazioni standard:

- σ_α è la deviazione standard dell'errore sull'angolo, in gradi;

- σ_{T} è la deviazione standard dell'errore sulla distanza per unità di traslazione cioè in cm/m o mm/m;

- σ_{rr} è la deviazione standard del'errore di rotazione per unità di rotazione, in gradi/360°;

- $\sigma_{r\mathrm{T}}$ è la deviazione standard del'errore di rotazione per unità di traslazione, in gradi/m.

In base alle considerazioni fatte fin qui di possiamo scrivere l'algoritmo 4.1 che esegue il passo di predizione, cioè applica il movimento ai campioni della MCL, considerando l'incertezza nell'informazione odometrica: a ciascun campione s di $Bel(s_{t-1})$ viene applicato il modello di predizione $p(s_t|s_{t-1}, a_{t-1})$; l'azione a_{t-1} è rappresentata dai valori di odometria (α_u, T, θ_f) e da questi si stimano i valori che tengono conto dell'incertezza $(\alpha'_u, T', \theta'_f)$ tramite il modello per l'errore di odometria presentato sopra.
Precisiamo che la funzione GaussianRandom(σ) restituisce un numero casuale distribuito secondo una distribuzione di probabilità gaussiana di media nulla e varianza σ^2.

Algoritmo 4.1 PREDIZIONE (s,a)

Parametri: il campione $s = (x, y, \theta)$ che rappresenta una posizione con orientazione e l'odometria $a = (\alpha_u, T, \theta_f)$ che descrive il movimento effettuato dal robot

Risultati: un nuovo campione $\widetilde{s} = (\widetilde{x}, \widetilde{y}, \widetilde{\theta})$ che rappresenta la nuova posizione tenendo conto dell'incertezza nell'informazione odometrica

$\alpha'_u = \alpha_u + \text{GaussianRandom}(\sigma_\alpha)$
$T' = T \cdot (1 + \text{GaussianRandom}(\sigma_{\text{T}}))$
$\theta'_f = \theta_f + \theta_f \cdot \text{GaussianRandom}(\sigma_{rr}) + T \cdot \text{GaussianRandom}(\sigma_{r\text{T}})$
$\widetilde{x} = x + T' \cdot cos(\theta + \alpha'_u)$
$\widetilde{y} = y + T' \cdot sin(\theta + \alpha'_u)$
$\widetilde{\theta} = \theta + \theta'_f$

4.2 Modello di osservazione

Il modello di osservazione è quella parte dell'algoritmo di localizzazione che simula la percezione visiva umana e la sua memoria visiva. Abbiamo utilizzato il sistema di calcolo della similarità descritto in dettaglio nel capitolo 2 e che riassumiamo brevemente di seguito.
Ci riferiamo ad un robot mobile con un sistema di visione omnidirezionale. Ciascuna immagine acquisita descrive l'intera scena a 360° dell'ambiente attorno al robot. La memoria visiva del robot è costituita da un database di *immagini di riferimento* acquisite in *posizioni di riferimento* conosciute, durante una fase preparatoria, precedente alla navigazione vera e propria del robot. Per motivi di efficienza non memorizziamo direttamente le immagini ma manteniamo la loro "Fourier signature".
Durante la vera navigazione il robot acquisisce un'immagine che abbiamo chiamato *immagine di input*. Il sistema la confronta con quelle di riferimento. Si ottengono così dei valori di dissimilarità dall'equazione (2.2) e scalando questi valori si ottiene un valore di similarità con l'equazione (2.3). In particolare l'immagine di riferimento con valore più alto di similarità fornisce una *localizzazione topologica*: il robot è più vicino alla posizione di riferimento corrispondente a tale immagine rispetto ad ogni altra posizione di riferimento. Cambiando il numero di frequenze usate per il calcolo della similarità otteniamo una localizzazione gerarchica, più o meno accurata. La localizzazione geometrica si ottiene con la MCL: il punto fondamentale è la definizione del *Modello di osservazione* $p(O|s)$.

4.2.1 Calcolo dei pesi dei campioni

Riportiamo per comodità le equazioni per il calcolo del valore di similarità fra due immagini omnidirezionali O_i e O_j. Per i dettagli si faccia riferimento alla sezione 2.3.
La funzione di dissimilarità calcolata come la norma L_1 dei primi m coefficienti di Fourier dei cilindri panoramici corrispondenti:

$$Dis(O_i, O_j) = \sum_{y=0}^{l-1} \sum_{k=0}^{m-1} |F_{iy}(k) - F_{jy}(k)| \quad . \tag{4.1}$$

La funzione di similarità, ottenuta riscalando i valori di dissimilarità tra 0 e 1000:

$$Sim(O_i, O_j) = 1000 - 1000 \frac{Dis(O_i, O_j) - \min_{i,j} Dis(O_i, O_j)}{\max_{i,j} Dis(O_i, O_j) - \min_{i,j} Dis(O_i, O_j)} \quad , \tag{4.2}$$

che, come abbiamo già detto, risulta decrescente con la distanza.

Il sistema quindi acquisisce l'immagine O, calcola i valori della funzione $Sim(O, O_r)$ rispetto alle M immagini di riferimento O_r. Il Modello di osservazione $p(O|s)$ viene definito in modo empirico come la media pesata dei valori di similarità delle immagini di riferimento vicine alla posizione s. Riferendosi all'algoritmo MCL, per assegnare il peso $\omega^{(i)}$ al campione $s^{(i)}$ usiamo l'equazione seguente:

$$\omega^{(i)} = \frac{1}{|C|} \sum_{g_j \in C} S_j \cdot (D - dist(s^{(i)}, g_j)) \quad , \tag{4.3}$$

dove C è l'insieme delle posizioni di riferimento che distano al più D dal campione, e S_j è il valore di similarità relativo all'immagine di riferimento in posizione g_j. Più il campione è vicino ad un'immagine di riferimento, più il valore di similarità dell'immagine contribuisce al peso. Questo fatto è giustificato dalla dipendenza lineare della similarità illustrata prima.

Osserviamo che nell'equazione (4.3) la funzione $dist(s_1, s_2)$ è una funzione di distanza tra due posizioni. Nel caso più semplice essa può essere definita come la distanza euclidea tra i due punti: in questo modo rinunciamo alla stima dell'orientazione del robot, che non contribuisce all'assegnazione del peso. Più in generale si può pensare di definire la funzione $dist(s_1, s_2)$ come in [Wolf, 2001] considerando anche le orientazioni:

$$dist(s_1, s_2) \doteq \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \beta \min(|\theta_1 - \theta_2|, |\theta_2 - \theta_1|) \quad ,$$

dove β è una opportuna costante di scalatura necessaria per poter confrontare angoli e distanze.

L'approccio che abbiamo usato è simile a quello già proposto da Burgard in [Wolf, 2001; Wolf *e altri*, 2002a,b] con alcune differenze importanti, legate principalmente ai diversi sistemi di visione usati.
Qui facciamo riferimento ad un robot mobile con un visione omnidirezionale; per dotare il robot di una memoria visiva viene costruito un database di immagini acquisite in vari punti dell'ambiente, ad esempio secondo una griglia regolare; queste immagini vengono chiamate *immagini di riferimento* e le posizioni note, in cui sono acquisite, *posizioni di riferimento*. Queste sono le uniche informazioni metriche sull'ambiente che utilizziamo. Infatti il nostro sistema di localizzazione di Monte Carlo non fa uso di nessuna mappa geometrica dell'ambiente. Inoltre fornisce una localizzazione attendibile anche senza considerare l'orientazione dei campioni: ricordiamo infatti che in generale un campione rappresenta una possibile ipotesi di posizione del robot e quindi consiste di una posizione cartesiana (x, y) e di un'orientazione θ.
Diversamente in [Wolf, 2001; Wolf *e altri*, 2002a,b] viene usata una mappa geometrica dell'ambiente, fornita a priori al robot, per determinare l'area di visibilità per ciascuna immagine di riferimento nel database; questo è reso necessario dal fatto che Burgard utilizza una telecamera prospettica, quindi nell'assegnare il peso ad una campione deve usare solo le immagini di riferimento che sarebbero visibili dal robot se si trovasse nell'ipotesi di posizione espressa dal campione: queste sono le immagini di riferimento nella cui area di visibilità cade il campione. Per lo stesso motivo Burgard deve considerare anche l'orientazione per assegnare un peso corretto al campione; infatti se l'orientazione del campione è troppo lontana dalla direzione osservata dall'immagine memorizzata nel database, allora la scena descritta non è visibile dal robot e tale immagine non deve essere considerata nel calcolo del peso del campione.
Nel nostro caso invece possiamo tralasciare l'orientazione poiché il nostro robot, essendo olonomo, non deve ruotare prima di muoversi. Inoltre utilizzando il sensore omnidirezionale abbiamo ottenuto un'invarianza rotazionale e quindi una scena viene riconosciuta indipendentemente dall'orientazione del robot: quindi non siamo obbligati a controllare l'orientazione del campione per assegnargli il peso. Pensiamo comunque che considerando l'orientazione anche nel nostro sistema si possa ulteriormente migliorarne il comportamento.

4.3 Strategia per il Robot Rapito

Il problema del robot rapito, che viene indicato in letteratura come "kidnapped robot problem" [Engelson, 1994], è il più complesso dei compiti affidati

al sistema di localizzazione di un robot. In questa situazione il robot ha la convinzione di trovarsi in una certa posizione mentre si trova in una posizione completamente diversa. È importante che il sistema di localizzazione sia in grado di capire questa situazione e di recuperare velocemente la posizione corretta altrimenti il robot eseguirà i propri compiti in modo sbagliato o addirittura catastrofico. La capacità del sistema di recuperare la posizione dopo un rapimento viene considerata come indice della capacità di recupero autonomo della posizione dopo un fallimento della localizzazione.

Alcune delle possibili cause di fallimento della localizzazione nelle applicazioni reali sono le seguenti:

- scontro con un'altro robot o con un ostacolo che faccia slittare le ruote e quindi perdere il riferimento odometrico;
- occlusione notevole del sistema di visione che impedisca al robot di localizzarsi per lunghi periodi di tempo;
- il robot viene spostato senza che ne abbia la conoscenza (ad esempio nella competizione RoboCup [Lenser e Veloso, 2000]).

Nella pratica per testare il sistema in condizioni di rapimento si fa acquisire al sistema di localizzazione la convinzione sulla propria posizione e poi:

- in esperimenti su robot si sposta il robot senza far muovere le ruote in modo tale che il sistema non rilevi nessuna odometria;
- nel caso di esperimenti svolti in simulazione, si forniscono al robot le informazioni di odometria e visione di una posizione diversa.

La strategia standard per affrontare il problema del rapimento è quella di rimpiazzare, dopo ogni passo di predizione, un certo numero fisso di campioni con campioni casuali generati in modo uniforme nell'intero ambiente [Fox *e altri*, 1999b; Wolf *e altri*, 2002a]. Questa tecnica si basa sulla speranza di generare almeno un campione vicino alla eventuale nuova posizione del robot. Come è stato dimostrato in [Wolf *e altri*, 2002a] questa tecnica è robusta perché riesce a recuperare sempre la posizione del robot ma a nostro avviso non è efficiente. Infatti essa richiede un elevato numero di passi per rilocalizzare il robot, in particolare in ambienti grandi dove la probabilità di generare un campione vicino alla vera posizione è bassa; inoltre servono più passi perché il singolo campione riesca a creare un "cluster" di campioni attorno alla nuova posizione, poiché gli altri campioni sono accumulati ancora attorno alla vecchia posizione.

Queste considerazioni ci hanno portato a sviluppare una nuova soluzione per il robot rapito che sfrutti le proprietà del nostro sistema di localizzazione basato sulle immagini, descritte precedentemente [Menegatti *e altri*, 2003a,b]. La soluzione che proponiamo è la seguente[1]: invece di generare nuovi campioni in modo casuale nell'ambiente, generiamo questi campioni attorno alle posizioni di riferimento corrispondenti alle immagini che sono più simili all'immagine di input corrente, cioè attorno alle localizzazioni topologiche. In questo modo i nuovi campioni vengono generati solo attorno alle posizioni che risultano più probabili secondo il nostro sistema di similarità. Possiamo usare questa tecnica perché, utilizzando un sensore omnidirezionale otteniamo in una singola osservazione una localizzazione topologica più accurata rispetto ad un sensore prospettico. Inoltre possiamo sfruttare la proprietà di localizzazione gerarchica descritta nella sezione 2.4: fermando il calcolo della somma nell'equazione (2.2), che abbiamo riportato anche sopra, otteniamo una localizzazione topologica più ampia o detto in altro modo meno selettiva. Infatti abbiamo visto in Figura 2.8 che utilizzando un numero di frequenze basso nel calcolo della similarità i picchi di similarità sono più allargati. In pratica se utilizziamo solo le basse frequenze l'immagine di input corrente viene considerata simile non solo all'immagine di riferimento corrispondente, che in genere è quella più vicina, ma anche ad altre immagini di riferimento: quest'ultime in caso di "perceptual aliasing" possono essere anche distanti dall'immagine di input. Quindi con questa tecnica riusciamo a concentrare i nuovi campioni attorno a tutte le possibili ipotesi di posizione. Ciascuna di queste ipotesi sarà poi "valutata" tramite il normale funzionamento della MCL. Ricordiamo anche che quando calcoliamo le similarità usando m frequenze possiamo conservare i risultati parziali che corrispondono a frequenze minori: quindi otteniamo in automatico anche le similarità a frequenze più basse $m' <= m$ senza ulteriore carico computazionale.

Come vedremo nel capitolo 5 con la nostra tecnica riusciamo a recuperare velocemente la posizione dopo un rapimento e inoltre velocizziamo anche la fase iniziale della localizzazione globale in quanto concentriamo più rapidamente i campioni attorno alle vere posizioni, senza comunque scartare le altre ipotesi di posizione prima di averne verificata la validità.

4.4 Software per l'algoritmo di MCL

Abbiamo implementato l'algoritmo di localizzazione di Monte Carlo, descritto nelle sezioni precedenti, utilizzando il linguaggio di programmazione C++.

[1]Per brevità ci riferiamo alla nostra tecnica come la "Topological kidnap strategy", mentre alla tecnica standard come la "Uniform kidnap strategy".

Abbiamo realizzato tutte le classi e le funzioni di supporto, a partire dalla definizione di un punto cartesiano, fino alle classi più complesse come, ad esempio, quella che definisce l'ambiente dove naviga il robot. Nella progettazione abbiamo usato i concetti tipici della programmazione ad oggetti in C++: in particolare, la *derivazione* e l'*ereditarietà* sono state usate per costruire le classi definendo livelli successivi di complessità e completezza delle funzionalità richieste; il *mascheramento dell'informazione* è stato usato per garantire la correttezza nell'accesso e modifica delle informazioni memorizzate nelle classi. Quest'ultimo è stato molto utile durante le fasi di modifica di alcune parti di software nel passaggio da una versione alla successiva, in quanto ha permesso di cambiare solo le parti interessate dalla modifica e non il resto del programma.
Complessivamente l'ultima versione del software per la MCL contiene 17 classi e circa 20 funzioni di supporto.
Per la programmazione C++ abbiamo fatto riferimento a [Lippman, 1993; Eckel, 2000]. Per la realizzazione di alcune classi e funzioni di tipo geometrico abbiamo fatto riferimento a [Preparata e Shamos, 1985; de Berg *e altri*, 2000].
Di seguito descriviamo brevemente le principali classi che abbiamo definito:

- classe *Point*: definisce una posizione del robot con orientazione. La posizione è espressa con due coordinate cartesiane (x, y) mentre l'orientazione è intesa come angolo in gradi nell'intervallo $[-180, 180)$;

- classe *Sample*: deriva da *Point* e rappresenta un campione (una posizione con orientazione) con relativo peso;

- classe *Polygone*: definisce un poligono nel piano. Serve per la definizione delle aree non raggiungibili di un ambiente;

- classe *Rectangule*: deriva da *Polygone* e definisce un rettangolo nel piano. Viene usato per definire l'area dell'ambiente di interesse per il robot;

- classe *Environment*: definisce un ambiente di navigazione del robot; in particolare memorizza le aree non accessibili al robot;

- classe *EnvironmentWithReference*: deriva da *Environment* e memorizza le posizioni delle immagini di riferimento;

- classe *EnvironmentWithReferenceInput*: deriva direttamente da *EnvironmentWithReference* e memorizza le posizioni delle immagini di input[2];

[2]L'astrazione realizzata con i tre livelli per le informazioni di un ambiente permette di

- classe *ErrorModel*: memorizza i parametri dell'errore di odometria;

- classe *PredictionModel*: memorizza i parametri del modello di predizione della MCL;

- classe *Odometry*: memorizza i valori di odometria rilevati a seguito di un movimento del robot;

- classe *LocalizationModel*: viene usato come classe "contenitore" per memorizzare tutti i parametri dell'algoritmo di MCL[3];

- classe *Robot*: classe che simula il comportamento di un robot olonomo;

- classe *SimilarityGrid*: classe di supporto usata per contenere i valori di similarità;

- classe *Samples*: classe fondamentale che memorizza l'insieme dei campioni della MCL; contiene inoltre la definizione di tutte le funzioni necessarie per realizzare la MCL.

Ricordiamo che per la generazione di numeri casuali secondo le distribuzioni uniforme e gaussiana abbiamo utilizzato una libreria scritta in C++ da Maurizio Loreti [Loreti, 2003].
Inoltre, in appendice C si può trovare una descrizione dell'ambiente grafico all'interno del quale è stato integrato il software di MCL.

far accedere ciascuna parte del programma alle sole informazioni necessarie, nascondendo, ad esempio, le posizioni di input all'algoritmo di localizzazione.

[3]Qui abbiamo usato il concetto di composizione [Lippman, 1993] capitolo 8.

Capitolo 5

Risultati sperimentali

Abbiamo descritto nei capitoli precedenti il sistema di calcolo delle similarità tra immagini (capitolo 2), la localizzazione di Monte Carlo (capitolo 3), le scelte di implementazione e le soluzioni proposte (capitolo 4). In questo capitolo descriviamo gli esperimenti che abbiamo svolto per testare il nostro sistema di localizzazione basato sulle immagini e il nostro sistema di localizzazione di Monte Carlo, ad esso integrato. La descrizione di alcuni dettagli implementativi dell'algoritmo di MCL e del sistema di sperimentazione possono essere trovati in appendice C.
Gli esperimenti sono stati eseguiti in modalità offline con un programma di simulazione da noi sviluppato ma i dati della visione elaborati sono stati acquisiti da un robot reale in due ambienti diversi.

5.1 Descrizione degli ambienti di prova

Gli esperimenti sono stati svolti in due ambienti indoor diversi. Entrambi si trovano al Piano 7 della Wakayama University in Giappone.

5.1.1 Il laboratorio: GridLab3

Il primo ambiente è un piccolo laboratorio rettangolare di circa 6m $\times$ 7m complessivi che abbiamo chiamato *GridLab3*. La mappa del laboratorio è quella di Figura 5.1. Questo laboratorio è affollato da molti oggetti e da molti mobili di altezze diverse, scrivanie e armadi. Nella mappa gli oggetti bassi sono quelli chiari, colorati in verde; gli oggetti alti sono quelli scuri colorati in rosso. Le immagini di riferimento sono state acquisite secondo una griglia regolare di forma rettangolare 7 $\times$ 13, a 25 cm di distanza l'una dall'altra: le relative posizioni di riferimento sono quelle indicate da puntini cerchiati.

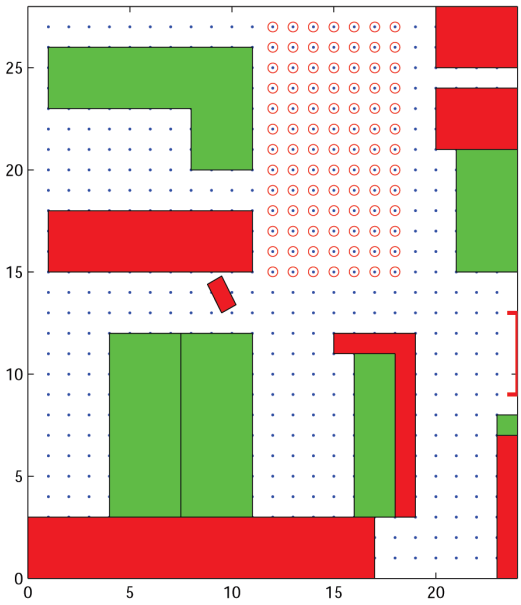


Figura 5.1: Mappa dell'ambiente *GridLab3* usato per i test preliminari sull'algoritmo di MCL. Questo ambiente è un laboratorio di circa 6m × 7m. Gli oggetti chiari (colore verde) sono bassi mentre quelli più scuri (colore rosso) sono alti. Le posizioni di riferimento sono indicate con i punti cerchiati. In queste posizioni sono state acquisite le immagini di riferimento. L'area di movimento per il robot è il rettangolo che circonda queste posizioni.

Il database di immagini è costituito da 91 immagini omnidirezionali. Una di queste immagini è quella di Figura 2.1 con il relativo cilindro panoramico in Figura 2.2. Le immagini di input sono state acquisite anch'esse secondo una griglia regolare 7 × 13, ciascuna al centro del quadrato formato da quattro posizioni di riferimento adiacenti. Le immagini di input non sono mostrate in Figura 5.1. Queste immagini sono 72 e sono utilizzabili come immagini di input che il robot vede lungo il proprio percorso. Abbiamo a disposizione solo queste immagini di input e ciò limita la mobilità del robot che può effettuare un percorso qualsiasi ma deve sempre muoversi da una posizione di input ad un'altra, non necessariamente adiacente. Per questo motivo e visto che l'ambiente non è molto grande abbiamo effettuato su di esso solo dei test preliminari, per verificare la correttezza di funzionamento del nostro sistema di localizzazione. In questo ambiente abbiamo anche testato la Topological kidnap strategy per la soluzione al problema del robot rapito effettuando un primo confronto con la tecnica standard Uniform strategy [1].

[1] Abbiamo presentato queste due tecniche nella sezione 4.3.

5.1.2 Il lungo corridoio: Test5

Il secondo ambiente in cui abbiamo testato il sistema di MCL è molto più interessante. Le sue caratteristiche infatti rendono difficile la localizzazione. Lo abbiamo chiamato *Test5*. L'area dell'ambiente è molto grande: è una parte del Piano 7 di circa 50m × 25m. L'area di movimento è costituita dal lungo corridoio che percorre l'area. La planimetria è quella di Figura 5.2. Il corridoio è lungo complessivamente circa 100 m e si può dividere in cinque parti: un primo corridoio lungo 50 m e altri quattro corridoi più corti che formano un cappio ricongiungendosi a metà del corridoio lungo. Le immagini di riferimento sono state acquisite lungo una linea al centro di ciascuno dei cinque corridoi ad una distanza di 20 cm l'una dall'altra. Le immagini di riferimento sono 452. Le immagini di input sono state acquisite sempre lungo la linea centrale dei corridoi ma a distanze non regolari. Le immagini di input sono 69 distribuite fra i cinque corridoi. Una delle immagini di riferimento di questo ambiente è quella di Figura 2.3. Questo ambiente è particolarmente interessante poiché i corridoi sono molto simili, con porte, pareti e incroci uguali. Inoltre le condizioni di luminosità pur non essendo uniformi lungo il corridoio, poiché la disposizione delle luci sul soffitto non è continua, si ripetono periodicamente. Tutte queste caratteristiche dell'ambiente provocano un forte "perceptual aliasing" come abbiamo già osservato nella sezione 2.4 e in particolare nella serie di grafici di Figura 2.7. In questo ambiente la localizzazione basata sulle sole immagini fallisce. È quindi l'ambiente adatto per effettuare dei test veramente significativi sul sistema di localizzazione e in particolare sulle due strategie per il robot rapito.

5.2 Verifica della localizzazione basata sulle immagini

In questa sezione presentiamo i test che abbiamo effettuato per verificare il sistema di localizzazione topologica basato sulle immagini che abbiamo presentato nel capitolo 2.

5.2.1 Prove di robustezza

Il nostro scopo è stato quello di sviluppare un sistema di localizzazione che permetta al nostro robot di navigare autonomamente e con accuratezza anche in ambienti dinamici e popolati; in questo caso bisogna tenere in considerazione che ci possono essere degli oggetti in movimento, come persone o altri robot, che possono provocare una temporanea occlusione nell'immagi-

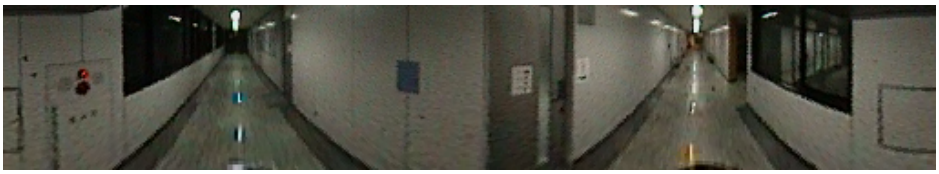
Figura 5.2: Mappa dell'ambiente *Test5* usato per i test sull'algoritmo di MCL. L'area dell'ambiente è di circa 50m × 25m. Le zone in grigio sono quelle non raggiungibili dal robot. In colore bianco si vedono i cinque corridoi; si noti il cappio formato dai quattro più corti che si ricongiungono a metà del corridoio lungo. che formano un cappio. Le posizioni di riferimento sono indicate con dei puntini blu.

ne acquisita. Quindi una parte della scena descritta dell'immagine di input è nascosta dall'ostacolo momentaneo mentre questo ostacolo non è presente nelle immagini di riferimento. Questo rende più difficile la corretta individuazione dell'immagine più simile introducendo delle false localizzazioni cioè dei falsi "perceptual aliasing". Di seguito presentiamo la serie test che abbiamo effettuato sul nostro sistema di similarità basato sulla "Fourier signature" e che ne hanno dimostrato la robustezza alle occlusioni.
I test sono stati eseguiti nel seguente modo, utilizzando le immagini dell'ambiente *Test5*: abbiamo considerato una serie di immagini di input con occlusione crescente e calcolando la similarità tra ciascuna di queste immagini e tutte immagini di riferimento. In questo modo i valori di similarità ottenuti sono scalati nel modo descritto nella sezione 2.3. Abbiamo poi considerato la similarità rispetto alle undici immagini di riferimento più vicine a quella di input, controllando se l'immagine più vicina corrisponde alla localizzazione topologica, cioè al valore 1000 di similarità.
Poiché non avevamo a disposizione delle immagini con persone che camminano attorno al robot, abbiamo creato delle immagini ad hoc. L'immagine di input originale, senza occlusione, è l'input numero 12 che si trova alla posizione di 1622 cm lungo l'asse x nel corridoio più lungo. A questa imma-

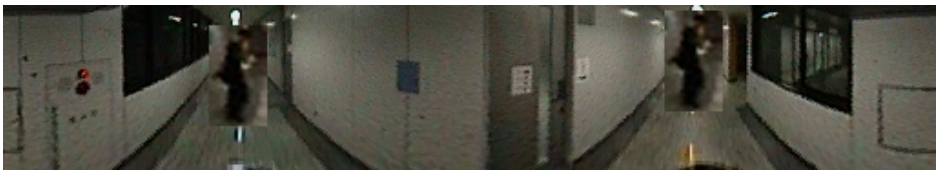
gine abbiamo incollato l'immagine di una persona che cammina, più volte, fino ad un massimo di dieci occlusioni. L'immagine incollata è di 32×55 pixel mentre l'immagine originale è di 512×90 pixel quindi ogni immagine incollata provoca un'occlusione di circa il 4% del cilindro panoramico.
Abbiamo confrontato queste immagini con tutte quelle di riferimento ma abbiamo considerato i valori di similarità solo delle undici immagini vicine, dalla numero 67 che si trova a 1520 cm fino alla 77 che si trova a 1720 cm. L'immagine di riferimento più vicina è la numero 72 che si trova a 1620 cm lungo il corridoio. In Figura 5.3 mostriamo la serie di immagini panoramiche ottenute dalla input 12 e utilizzate per il test. In Figura 5.4 si vedono i corrispondenti grafici dei valori di similarità per le undici immagini di riferimento più vicine a quella di input. Osserviamo che tutti i test sono stati eseguiti considerando 15 frequenze nel calcolo della similarità.

La Figura 5.4 (a) mostra il grafico della similarità dell'immagine di input originale senza occlusioni. Si vede che la localizzazione topologica è data dall'immagine di riferimento numero 72 a cui corrisponde il valore 1000 di similarità. Questa è la localizzazione corretta poiché la 72 è l'immagine più vicina all'input. In (b) e (c) si vedono i grafici rispettivamente nel caso di 2 e 3 occlusioni e la localizzazione rimane corretta sull'immagine 72. Il grafico (d) corrisponde al caso di 4 occlusioni (circa il 16% del cilindro panoramico originale). Qui si vede che l'immagine 72 non viene più riconosciuta come la più simile; l'immagine più simile ora è la numero 332 che la cui similarità non si vede nel grafico poiché si trova distante, addirittura su un altro corridoio. Osservando l'imagine 332 in Figura 5.5 (a) si nota che alcuni pattern di luminosità che distinguevano quest'immagine da quella di input originale, vengono cancellati dalla presenza delle occlusioni, mentre ad esempio rimane il grosso pattern creato dalla porta. I grafici (e) e (f) mostrano i casi di 8 e 10 occlusioni in cui l'immagine di input non viene più considerata simile alle immagini di riferimento vicine, in particolare anche la 72; qui l'immagine più simile è la numero 334 (Figura 5.5 (b)) che si trova in un altro corridoio. Questo è un caso estremo, un "worst case", in cui il sistema effettua una localizzazione topologica completamente errata; osserviamo però, che l'occlusione artificiale che abbiamo creato incollando un'immagine rettangolare crea delle variazioni notevoli nei pattern di luminosità, molto superiori rispetto a quelli creati da una persona reale che cammina vicino al robot.

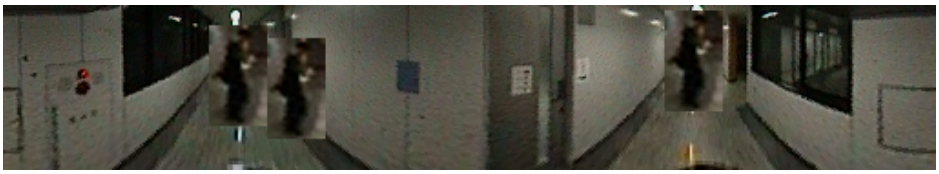
Possiamo quindi affermare che il nostro sistema di similarità tra le immagini basato sulla "Fourier signature" è robusto alle occlusioni se queste non sono eccessive da modificare completamente le zone di luminosità dell'immagine.



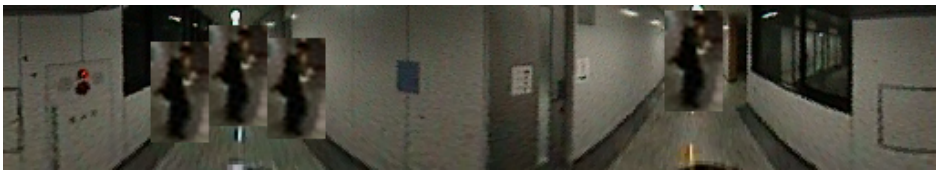
(a)



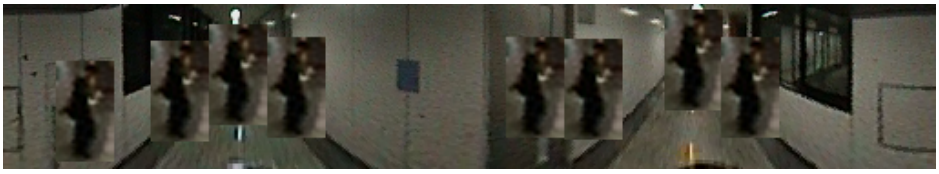
(b)



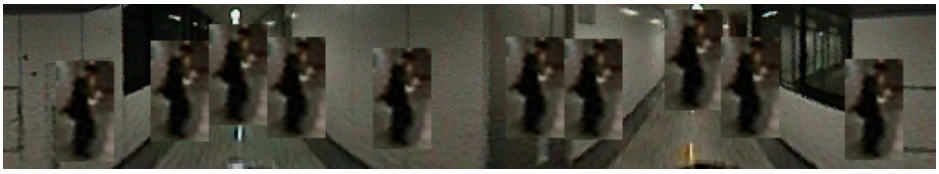
(c)



(d)



(e)



(f)

Figura 5.3: Immagini di input per il test di robustezza all'occlusione: l'immagine (a) è l'input numero 12 originale. Alle immagini (b), (c), (d), (e) e (f) sono state aggiunte rispettivamente 2, 3, 4, 8 e 10 immagini di una persona che cammina. Si noti che l'immagine incollata ha una forma rettangolare e crea una variazione notevole del pattern di luminosità, superiore a quella creata da una persona reale.

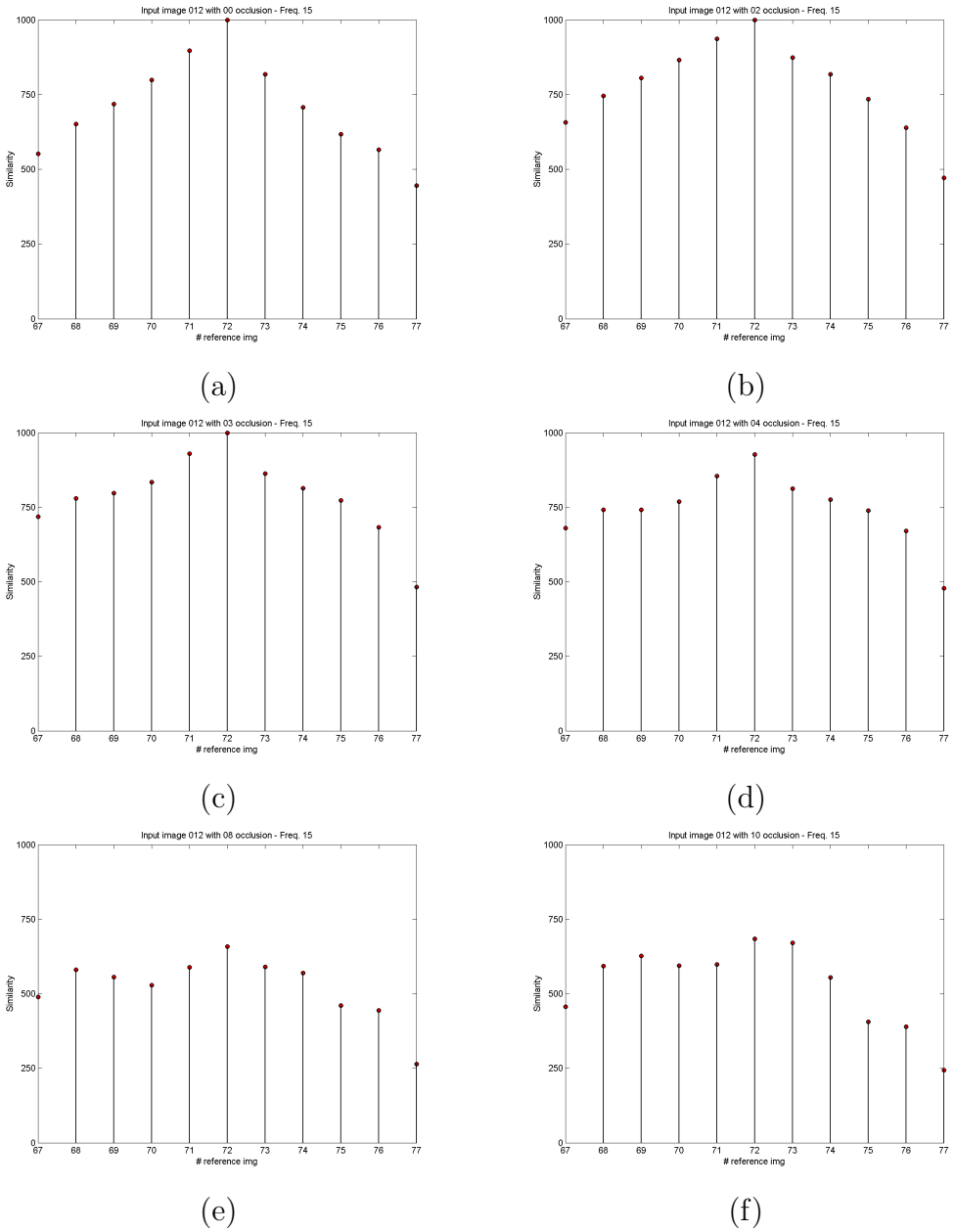
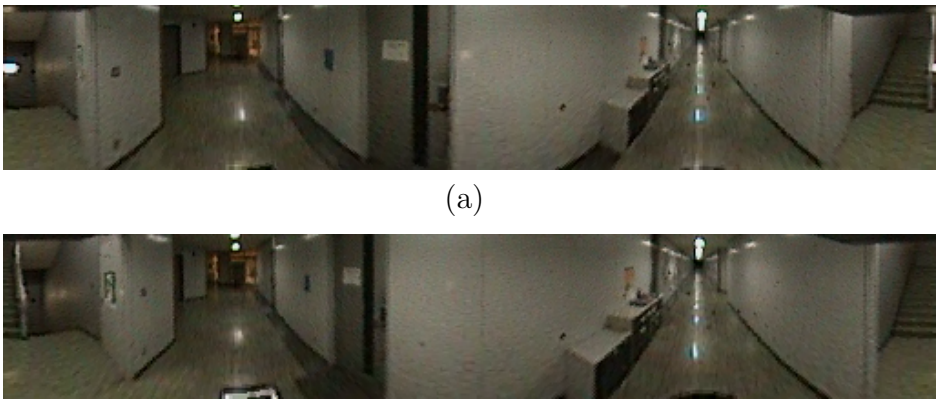


Figura 5.4: Grafici delle similarità nel test di robustezza all'occlusione. I grafici si riferiscono alle immagini (a), (b), (c), (d), (e) e (f) di Figura 5.3 e mostrano la similarità rispetto alle undici immagini di riferimento più vicine. In (a) si vede che l'immagine più simile a quella di input originale è la numero 72 che in effetti è la più vicina. In (b) e (c) si vede che con due e tre occlusioni la localizzazione topologica rimane corretta. In (d) l'immagine più simile non è più la 72 ma il valore di similarità resta abbastanza elevato. In (e) e (f) invece l'immagine non è più riconosciuta simile alla numero 72 ma alla numero 334, che non appare nel grafico poiché è distante. Questo è un "worst case" poiché l'occlusione artificiale è molto elevata e crea dei pattern artificiali di luminosità che non si verificano nel caso di occlusioni con persone reali.



(a)

(b)

Figura 5.5: Immagini panoramiche dell'ambiente *Test5*. L'immagine (a) è la reference numero 331 che viene considerata più simile a quella con quattro occlusioni di Figura 5.3 (d). L'immagine (b) è la numero 334 che viene "confusa" con l'input con dieci occlusioni di Figura 5.3 (f)

5.3 Verifica della MCL

In questa sezione presentiamo una serie di esperimenti che abbiamo effettuato per verificare il funzionamento del sistema di localizzazione di Monte Carlo, che abbiamo sviluppato per integrare il sistema di localizzazione basata sulle immagini. Presentiamo la prima serie di esperimenti preliminari effettuati sull'ambiente *GridLab3*. Poi presentiamo una serie di esperimenti svolti sull'ambiente *Test5*.

5.3.1 Prove nell'ambiente *GridLab3*

L'ambiente è quello descritto nella sezione 5.1.1. Come abbiamo già osservato è un piccolo laboratorio con molti mobili e oggetti. L'area di movimento del robot è piuttosto piccola, di circa 2m $\times$ 4m, ed è il rettangolo che circonda le immagini di riferimento di Figura 5.1. Vista la semplicità dell'ambiente di movimento abbiamo svolto solo dei test preliminari al fine di confermare il funzionamento del sistema [Menegatti *e altri*, 2003a].
Abbiamo effettuato numerose prove di localizzazione globale, position tracking e kidnapped robot. Il sistema è stato in grado di localizzare sempre il robot e di mantenere la posizione nei passi successivi.

In Figura 5.6 si vedono alcuni snapshot del sistema mentre effettua una localizzazione globale e position tracking sul percorso (a). Il percorso del robot è quello disegnato con la linea blu, i campioni sono disegnati con dei

punti rossi, mentre la stima fatta dal sistema è disegnata in nero. I punti di colore verde scuro, posizionati secondo una griglia regolare, sono le posizioni delle immagini di riferimento.
Ricordiamo che la posizione del robot viene stimata come la media delle posizioni rappresentate dai campioni. Per questi test abbiamo usato l'algoritmo di MCL standard, senza nessuna tecnica per il robot rapito.
Abbiamo eseguito questo esperimento con 500 campioni. All'inizio il sistema non ha nessuna informazione sulla posizione del robot infatti in (b) si vede che i campioni vengono inizializzati in modo uniforme sull'intera area di movimento. Poi il robot inizia a muoversi e ad acquisire ed elaborare le informazioni dell'odometria e della visione. In (c), dopo due passi, si vede che i campioni sono ancora molto sparpagliati anche se sono scomparsi molti dei campioni che erano stati generati in basso, più lontani dalla vera posizione del robot. In (d) i campioni si sono quasi del tutto addensati attorno alla posizione del robot mentre in (e), dopo 6 passi si vede che il sistema ha individuato la posizione con una buona approssimazione: ha effettuato una localizzazione globale. Nei passi successivi (f), (g) e (h) il sistema riesce a mantenere la posizione del robot durante il moto, mantenendo concentrati i campioni attorno alla vera posizione: il sistema effettua un position tracking correggendo l'errore presente nei dati di odometria che rileva dalle ruote.

In un successivo esperimento abbiamo cercato di capire il numero minimo di campioni necessario per effettuare una localizzazione corretta e affidabile. Abbiamo osservato sperimentalmente che è necessario un numero di campioni dello stesso ordine di grandezza del numero di immagini di riferimento. Questo fatto conferma l'intuizione generale secondo cui, per effettuare la localizzazione globale è necessario generare dei campioni vicini alla vera posizione del robot. Poiché inizialmente di campioni vengono generati in modo uniforme nell'ambiente e poiché le M immagini di riferimento del *GridLab3* sono disposte su una griglia regolare allora utilizzando un numero di campioni inferiore a M alcune immagini di riferimento non avranno nessun campione vicino. Esiste quindi la possibilità di non generare campioni vicini alla vera posizione; visto che qui non usiamo nessuna tecnica per il robot rapito il sistema non ha modo di generare successivamente dei campioni vicini alla vera posizione, se non per pura casualità. Quindi è possibile che il sistema non riesca ad eseguire la localizzazione globale, come confermato dalla serie di snapshot di Figura 5.7.
Nel *GridLab3* abbiamo 91 immagini di riferimento quindi abbiamo provato ad effettuare dei test con 20 e 100 campioni. In Figura 5.7 si vede che con 20 campioni il sistema non riesce ad effettuare la localizzazione globale. Infatti in (a) si vede che il sistema non ha generato nessun campione vicino alla posizione iniziale del robot. Successivamente i nuovi campioni vengono

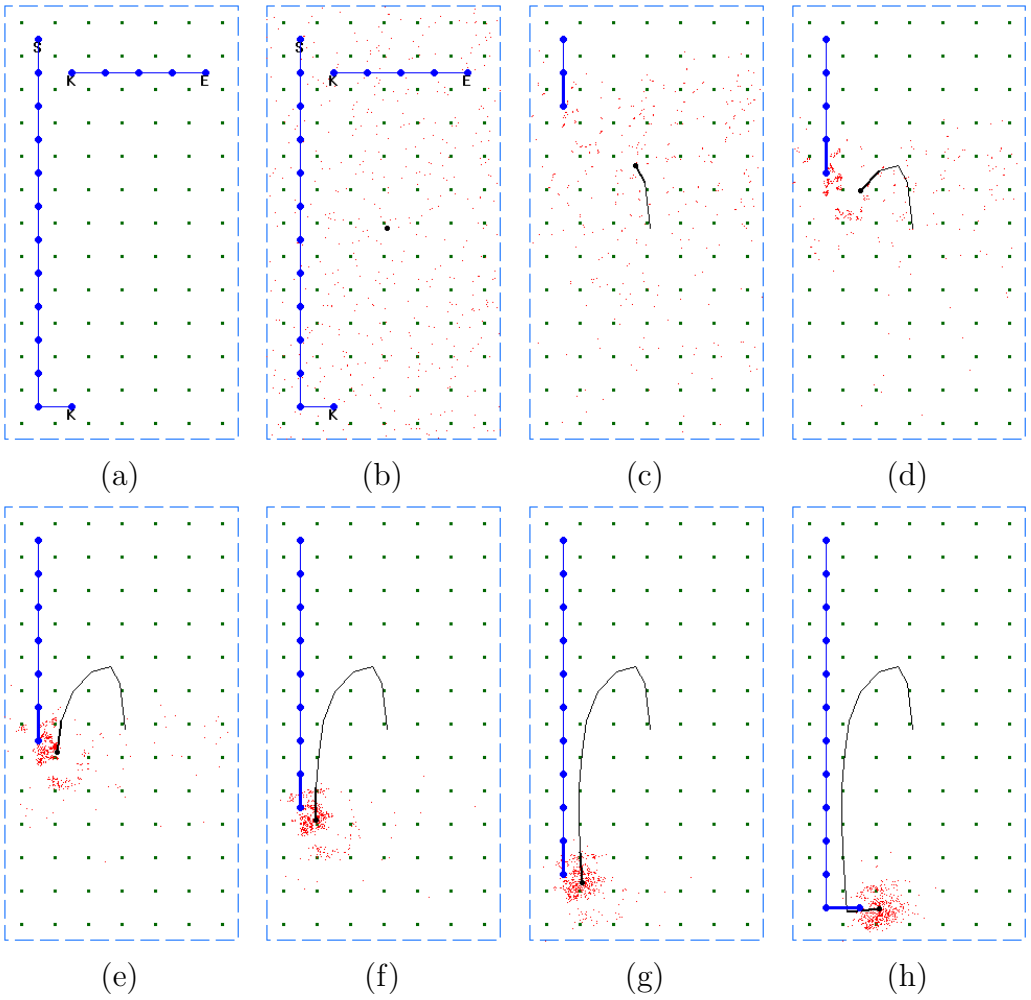
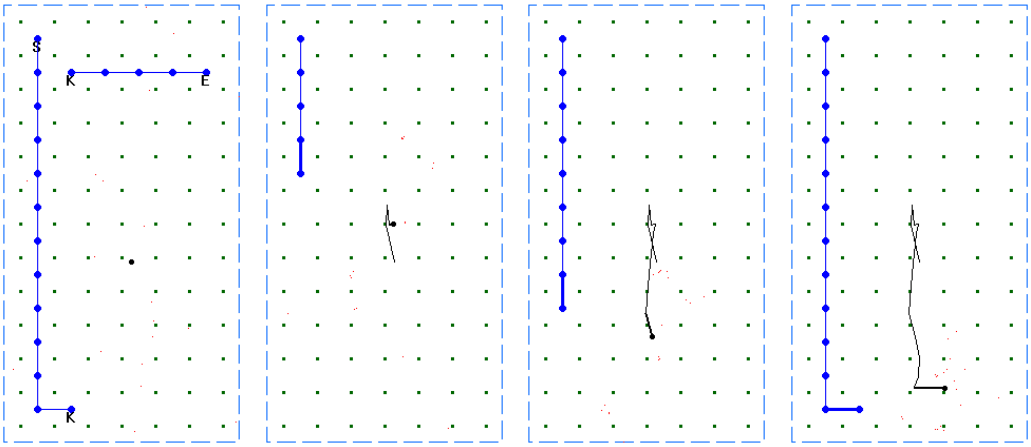


Figura 5.6: Localizzazione di Monte Carlo nel laboratorio *GridLab3*. Il test è stato eseguito con 500 campioni. Il percorso è mostrato in (a), mentre in (b) si vedono i campioni dopo l'inizializzazione: essi sono distribuiti uniformemente sull'area poiché la posizione iniziale non è nota. In (c) e (d) si vede il sistema durante la localizzazione globale. In (e), dopo 6 passi il sistema ha determinato la posizione con buona accuratezza. In (f), (g) e (h) si vede il position tracking, dopo che il sistema ha determinato la posizione.

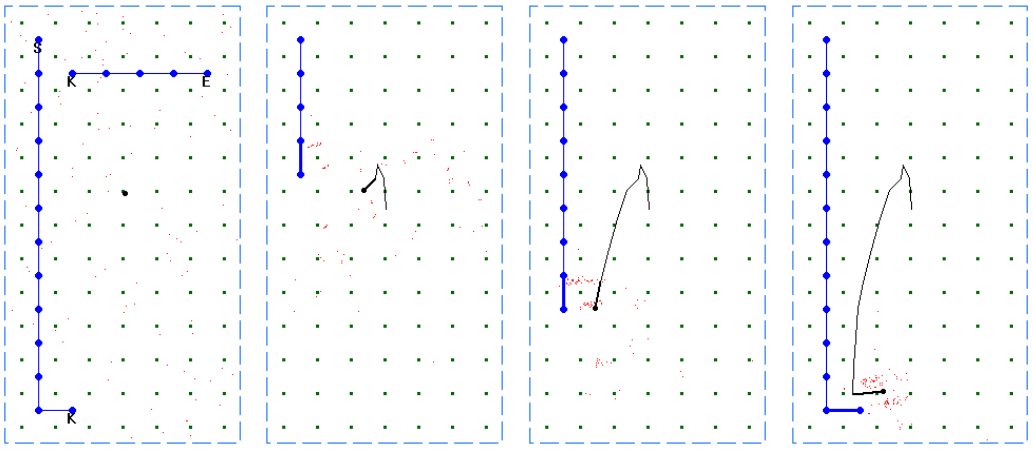
generati solo dal passo di predizione, a partire dai campioni del passo precedente perché non viene usata nessuna strategia per il robot rapito. Quindi il sistema non riesce a localizzare il robot perché i pochi campioni restano concentrati in zone distanti dalla posizione reale. In Figura 5.8 invece si vede lo stesso esperimento eseguito con un numero di campioni pari a 100, cioè dello stesso ordine di grandezza del numero di immagini di riferimento. In questo caso il sistema genera all'inizio dei campioni vicini alla posizione del robot e quindi riesce a localizzarlo. I campioni però restano poco concentrati e quindi la stima è meno precisa rispetto al caso con 500 campioni mostrato in Figura 5.6.



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

Figura 5.7: Localizzazione fallita nel laboratorio *GridLab3* con 20 campioni. Il percorso è lo stesso di Figura 5.6 ma in questo caso abbiamo usato un numero di campioni molto basso, pari a 20: si vede in (a) che nessun campione viene generato vicino alla posizione iniziale del robot quindi il sistema fallisce nella localizzazione (b), (c) e (d).

Sull'ambiente GridLab3 abbiamo effettuato un'ultimo test per verificare il funzionamento della Topological kidnap strategy che abbiamo proposto per risolvere il problema del robot rapito. Abbiamo effettuato un primo confronto con la strategia standard. Ricordiamo che in questo problema viene eseguita una localizzazione globale in modo che il sistema acquisisca la corretta posizione. A questo punto il robot viene rapito cioè, viene fisicamente portato in un'altra pozione senza che il sistema di localizzazione se ne accorga. In questo modo il sistema mantiene la consapevolezza di muoversi a partire dalla vecchia posizione e non si accorge che invece è stato spostato. La localizzazione di Monte Carlo semplice non può recuperare la nuova posizione a meno di un caso fortuito in cui almeno un campione viene generato vicino

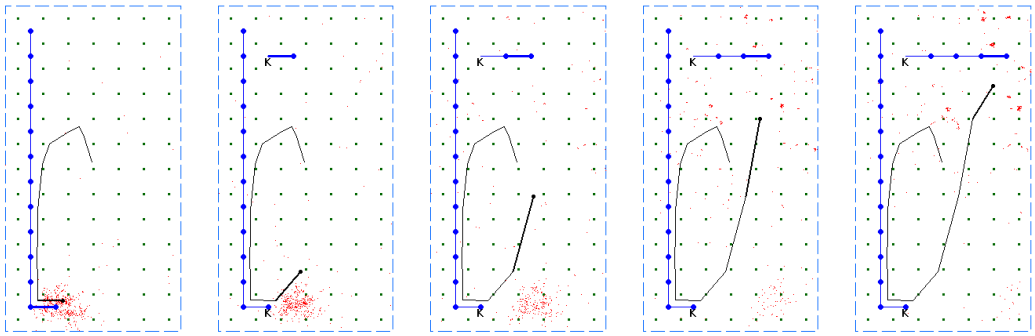


(a) (b) (c) (d)

Figura 5.8: Localizzazione di Monte Carlo nel laboratorio *GridLab3* con 100 campioni. Il percorso è lo stesso di Figura 5.6 (a). In questo caso abbiamo usato un numero di campioni basso, pari a 100, dello stesso ordine di grandezza del numero di immagini di riferimento: si vede che solo dopo 8 passi la posizione stimata si avvicina a quella reale; però i campioni non si accumulano attorno alla vera posizione, come invece accadeva in Figura 5.6 (e), quindi la stima è meno precisa anche dopo 12 passi.

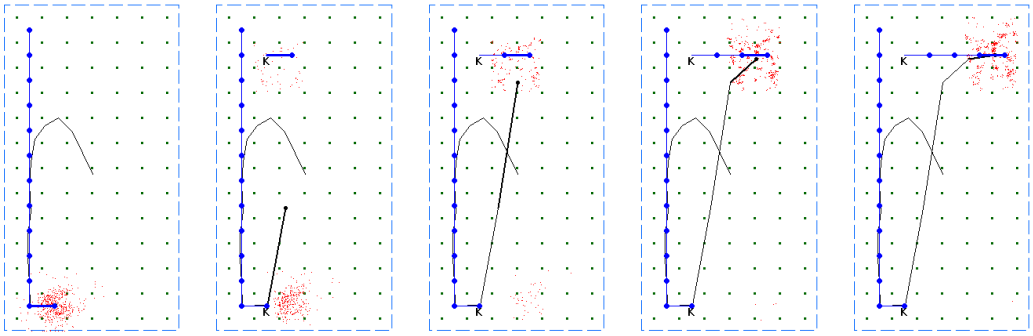
alla nuova posizione; ma questo è praticamente impossibile se il robot viene spostato in una posizione molto distante dalla zona dove sono concentrati i campioni. È necessaria quindi una strategia apposita che permetta di generare dei nuovi campioni vicini alla nuova posizione. L'approccio standard è quello di generare i nuovi campioni in modo uniforme nell'area di movimento del robot. Si basa sulla considerazione che è statisticamente probabile che generando un numero sufficiente di campioni, almeno uno venga generato vicino alla nuova posizione, ma questo può richiedere molti passi; inoltre, come vedremo, generando un singolo campione vicino sono necessari molti passi per concentrare tanti campioni attorno a questa posizione. Durante questo tempo il sistema riporterà una localizzazione errata. La strategia che abbiamo proposto utilizza le informazioni che provengono dalla visione: ad ogni passo il sistema genera un certo numero di campioni attorno alle immagini di riferimento che vengono considerate più simili a quella di input corrente vista dal robot. L'approccio si basa quindi sulla localizzazione topologica del nostro sistema di similarità, che come abbiamo dimostrato nei capitoli precedenti è affidabile e robusta alle occlusioni. Inoltre considerando i valori di similarità calcolati con un numero basso di frequenze, il sistema può prendere in considerazione molte più ipotesi di posizione. Quindi questa strategia si rivela robusta nel caso di "perceptual aliasing" perché prende in considera-

zione tutte le ipotesi di posizione descritte dal sistema di visione. Di seguito descriviamo i risultati di questi test. Gli snapshot sono quelli riportati in Figura 5.9 e in Figura 5.10. Il percorso è lo stesso delle prove precedenti solo che dopo il passo 12 il robot viene spostato in una nuova posizione lontana, senza che il sistema di localizzazione ne abbia conoscenza. In pratica questo viene realizzato, passando in input al programma di simulazione i dati di odometria e di visione relativi alla nuova posizione dopo il rapimento. Il numero di campioni reinizializzati è lo stesso per entrambe le strategie. Si vede in Figura 5.9 (b) che con la strategia uniforme pochi campioni vengono generati attorno alla nuova posizione e quindi il sistema impiega molti passi per accorgersi che il robot è stato spostato. Quindi i campioni rimangono concentrati nei pressi della vecchia posizione per molti passi dopo il rapimento. La pozione viene comunque recuperata, anche se con poca accuratezza in (e) dove però è evidente che i campioni sono ancora poco concentrati. Negli snapshot di Figura 5.10 si vede invece che con la strategia che abbiamo proposto il sistema si accorge già dopo due passi, in (c), che il robot è stato spostato. Questo è possibile perché i nuovi campioni sono generati in base alle informazioni di visione attorno alle posizioni più probabili. Si vede che comunque il sistema non elimina subito la vecchia ipotesi, rappresentata dai campioni che continuano a sopravvivere in basso in (c) e (d). In (e) invece il sistema ha definitivamente rilocalizzato il robot e concentrato i campioni attorno alla nuova posizione.



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

Figura 5.9: Problema del robot rapito con la strategia standard. Ad ogni passo vengono generati alcuni campioni (in genere 50) in modo uniforme nell'ambiente. Si può notare in (d) e (e) che il sistema rilocalizza il robot ma con molti passi e inoltre non riesce a concentrare i campioni attorno alla nuova posizione del robot.



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

Figura 5.10: Problema del robot rapito con la strategia proposta "Topological kidnap strategy". Ad ogni passo vengono generati dei campioni attorno alle localizzazioni topologiche, cioè alle immagini di riferimento più simili a quella corrente. Si vede in (c) che già dopo 2 passi dopo il rapimento il sistema si è accorto che il robot non si trova più nella posizione precedente. In (d) e (e) il robot è stato correttamente rilocalizzato e i campioni si sono concentrati attorno alla nuova posizione.

5.3.2 Prove nell'ambiente *Test5*

Come abbiamo osservato precedentemente il laboratorio *GridLab3* è un ambiente piuttosto semplice; l'area di movimento del robot è piuttosto limitata quindi i percorsi di prova piuttosto semplici. Per questo abbiamo effettuato delle prove ulteriori nell'ambiente che abbiamo chiamato *Test5*. Come abbiamo descritto nella sezione 5.1.2 l'ambiente è molto grande e l'area di movimento è costituita da una serie di cinque corridoi, di cui uno lungo con gli altri quattro più corti che formano un cappio ricongiungendosi a metà del corridoio lungo. Abbiamo visto nella sezione 2.4, nei grafici di Figura 2.7, che questo ambiente presenta un forte "perceptual aliasing", in particolare nel corridoio più lungo. Questo succede perché l'ambiente è costruito in modo ripetitivo, con lo stesso tipo di porte e pareti e con illuminazione che si ripete periodicamente. In un'ambiente di questo tipo la sola localizzazione basata sulle immagini fallisce poiché non è in grado di decidere fra le varie ipotesi di posizione. Ci è sembrato quindi l'ambiente adatto per testare il sistema di localizzazione in condizioni così critiche.

Abbiamo eseguito una serie di test per confrontare le due strategie per il robot rapito: la strategia da noi proposta e la strategia standard. Abbiamo ripetuto questo confronto anche in questo ambiente perché sono presenti molte false ipotesi, e quindi dopo un rapimento la nostra strategia deve riuscire a gestire tutte le ipotesi di posizione selezionando la migliore. Questa prova permette di stabilire la robustezza della tecnica usata in presenza di

"perceptual aliasing".
Il percorso su cui abbiamo eseguito questo test è quello di Figura 5.11, disegnato in blu; i pallini blu indicano le posizioni di input lungo il percorso, dove il sistema ha letto le informazioni odometriche e di visione. Il cammino è costituito da 68 passi e percorre tutti e cinque i corridoi dell'ambiente. Inoltre il robot viene rapito al passo 15 mentre si sta muovendo lungo il corridoio più lungo e viene spostato circa 16 metri più avanti lungo lo stesso corridoio. Poi il robot continua a muoversi percorrendo gli altri quattro corridoi in circolo. Osserviamo che questa situazione è particolarmente critica. Infatti il rapimento viene fatto lungo il corridoio lungo che, come abbiamo visto nella sezione 2.4, presenta un forte "perceptual aliasing". Inoltre la posizione in cui viene spostato è una posizione in cui il sistema di similarità riporta molte false ipotesi, come si vede in Figura 2.7. Quindi il sistema di localizzazione deve essere in grado di recuperare la posizione dopo il rapimento e gestire le ipotesi scegliendo la migliore. Sempre in Figura 5.11, ma di colore verde, si vede il percorso stimato in base ai dati odometrici. A questo percorso è stato aggiunto un errore di parametri $(\sigma_\alpha, \sigma_{\mathrm{T}}, \sigma_{rr}, \sigma_{r\mathrm{T}}) = (1, 1, 3, 3)$: l'errore introdotto non è molto elevato, e infatti il percorso secondo l'odometria ha una forma che rispecchia quella realmente eseguita dal robot, almeno all'inizio[2]. L'errore però si accumula ad ogni passo e quindi il percorso stimato dall'odometria diventa sempre più impreciso. Notiamo inoltre che l'odometria non rileva che il robot è stato rapito e quindi da un certo punto in poi riporta una posizione che in realtà è irraggiungibile dal robot (area grigia).

All'inizio dell'esperimento il sistema effettua una localizzazione globale in quanto non conosce la posizione iniziale del robot. Quindi inizialmente i campioni vengono generati in modo random su tutto l'ambiente; questa situazione è mostrata in Figura 5.12: si vede che i campioni sono distribuiti uniformemente sull'intera area dell'ambiente, anche in zone che non sono raggiungibili dal robot, disegnate in grigio. Questo significa che non controlliamo la generazione dei campioni, visto che questo richiederebbe una mappa dell'ambiente. Il nostro sistema, come abbiamo detto, non conosce la planimetria dell'ambiente e genera i campioni in tutta l'area. I campioni che vengono generati in zone non consentite risulteranno lontani dalle posizioni di riferimento, visto che queste immagini possono essere acquisite solo in zone raggiungibili. Ma allora questi campioni verranno pesati con peso nullo dal sistema e verranno eliminati dal passo di ricampionamento della MCL. Osserviamo che questa è una situazione critica per la localizzazione: poiché l'ambiente è grande e le zone non raggiungibili sono ampie, come si vede in Figura 5.2, molti dei campioni vengono "sprecati" in zone dove il robot

[2]Ricordiamo che l'errore introdotto è di tipo gaussiano e si accumula ad ogni passo.

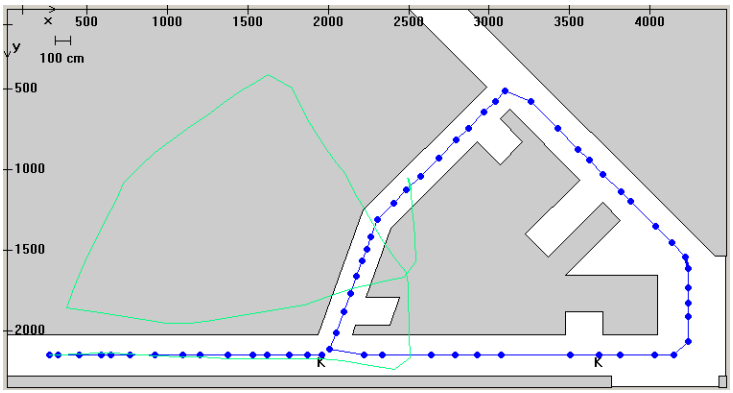


Figura 5.11: Percorso utilizzato per le prove nell'ambiente *Test5*. Il percorso è disegnato in blu. I pallini blu indicano le posizioni dove il robot ha misurato l'odometria e ha acquisito l'immagine. Il robot percorre il corridoio lungo da sinistra verso destra. Al passo 15 del percorso il robot viene rapito e spostato più avanti nello stesso corridoio. Poi continua percorrendo i restanti quattro corridoi compiendo un circolo. In verde è stato disegnato il percorso rilevato dall'odometria, a cui è stato aggiunto un errore di parametri $(\sigma_\alpha, \sigma_{\mathrm{T}}, \sigma_{rr}, \sigma_{r\mathrm{T}}) = (1, 1, 3, 3)$; si vede che l'odometria è imprecisa, per la presenza dell'errore aggiuntivo e inoltre non riesce a rilevare che il robot è stato rapito.

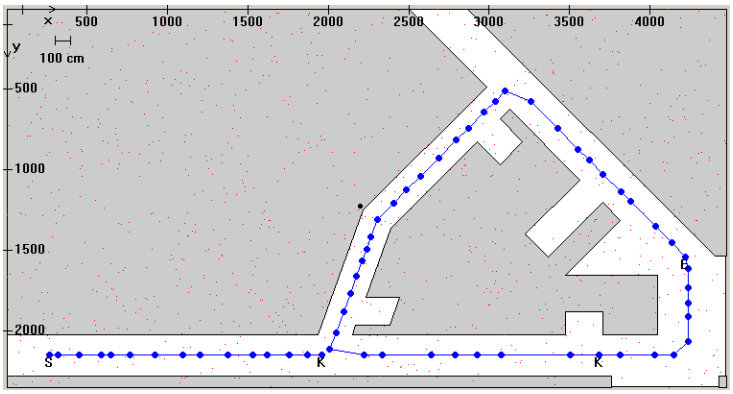


Figura 5.12: Inizializzazione dei campioni della MCL per la localizzazione globale nel percorso di Figura 5.11. I campioni vengono inizializzati in modo uniforme su tutta l'area dell'ambiente, anche in zone non raggiungibili dal robot.

non può essere. Quindi il sistema non ha a disposizione tutti i campioni per effettuare la localizzazione globale ma solo una piccola parte. Si vede in Figura 5.13 che la MCL è robusta anche a questa situazione.

In Figura 5.13 mostriamo alcuni snapshot dell'esperimento, e confrontiamo le due strategie di soluzione al problema del robot rapito: la serie di sinistra si riferisce alla strategia standard, con campioni generati in modo uni-

forme ad ogni passo; la serie di destra invece mostra la situazione negli stessi momenti ma con la strategia proposta da noi, che genera dei nuovi campioni ad ogni passo in base alle informazioni della visione. Si vede che la nostra strategia migliora la localizzazione globale; infatti il nostro sistema localizza il robot, dal nulla, in circa cinque passi, mentre con la strategia standard ne servono circa il doppio. Inoltre si può notare che il nostro sistema concentra molto i campioni e ottiene una stima più accurata. Il fatto che i campioni sono molto concentrati non pregiudica il funzionamento quando il robot viene rapito, perché il nostro sistema prende in considerazione tutte le nuove ipotesi descritte dalla localizzazione topologica (confronta con capitolo 2), generando dei campioni vicini a queste posizioni. Ritornando al confronto fra le due strategie, si nota che la nostra rilocalizza molto velocemente il robot dopo il rapimento, in circa quattro passi; la strategia uniforme invece impiega più di dieci passi per recuperare la corretta posizione. Questo si spiega con il fatto che noi non generiamo i nuovi campioni a caso nell'ambiente, ma solo dove è più probabile che si trovi il robot. La nostra tecnica risulta robusta poiché, sfruttando la localizzazione gerarchica andiamo a considerare tutte le nuove ipotesi probabili. Visto che abbiamo dimostrato la robustezza del sistema di visione omnidirezionale integrato alla "Fourier signature" possiamo fidarci delle localizzazioni topologiche che risultano, sapendo che la corretta localizzazione topologica verrà sempre individuata. Osserviamo infine che nell'ultima serie di Figura 5.13 si vede che la nostra tecnica permette una maggiore accuratezza nella stima, proprio perché non generiamo campioni a caso che se sopravvivono ingannano la stima della posizione.

Infine in Figura 5.14 mostriamo un caso in cui il sistema di visione riporta più ipotesi di posizione probabili; la posizione è quella della input 18: si vede dal grafico di similarità di Figura 2.7 (b) (che riportiamo per comodità in Figura 5.15), che le false ipotesi sono almeno quattro ma i campioni sopravvivono solo attorno a tre[3]; nel passo successivo le ipotesi sbagliate vengono eliminate.

In Figura 5.16 mostriamo i grafici dell'andamento dell'errore medio di localizzazione lungo il percorso, per ciascuna delle due strategie. L'errore di localizzazione è stato calcolato come la distanza tra la corretta posizione del robot e la posizione stimata dal sistema. Ciascuna prova è stata ripetuta trenta volte e si è fatta la media dei risultati. In (a) si vede l'intero grafico di errore: in blu è mostrato l'andamento per la nostra strategia mentre in rosso è quello della strategia standard. Notiamo che il nostro sistema è molto più veloce ad effettuare la localizzazione globale, e dopo la localizzazione la stima rimane accurata. Inoltre riesce a recuperare velocemente la posizione

[3]Gli snapshot mostrano la situazione del sistema dopo il ricampionamento.

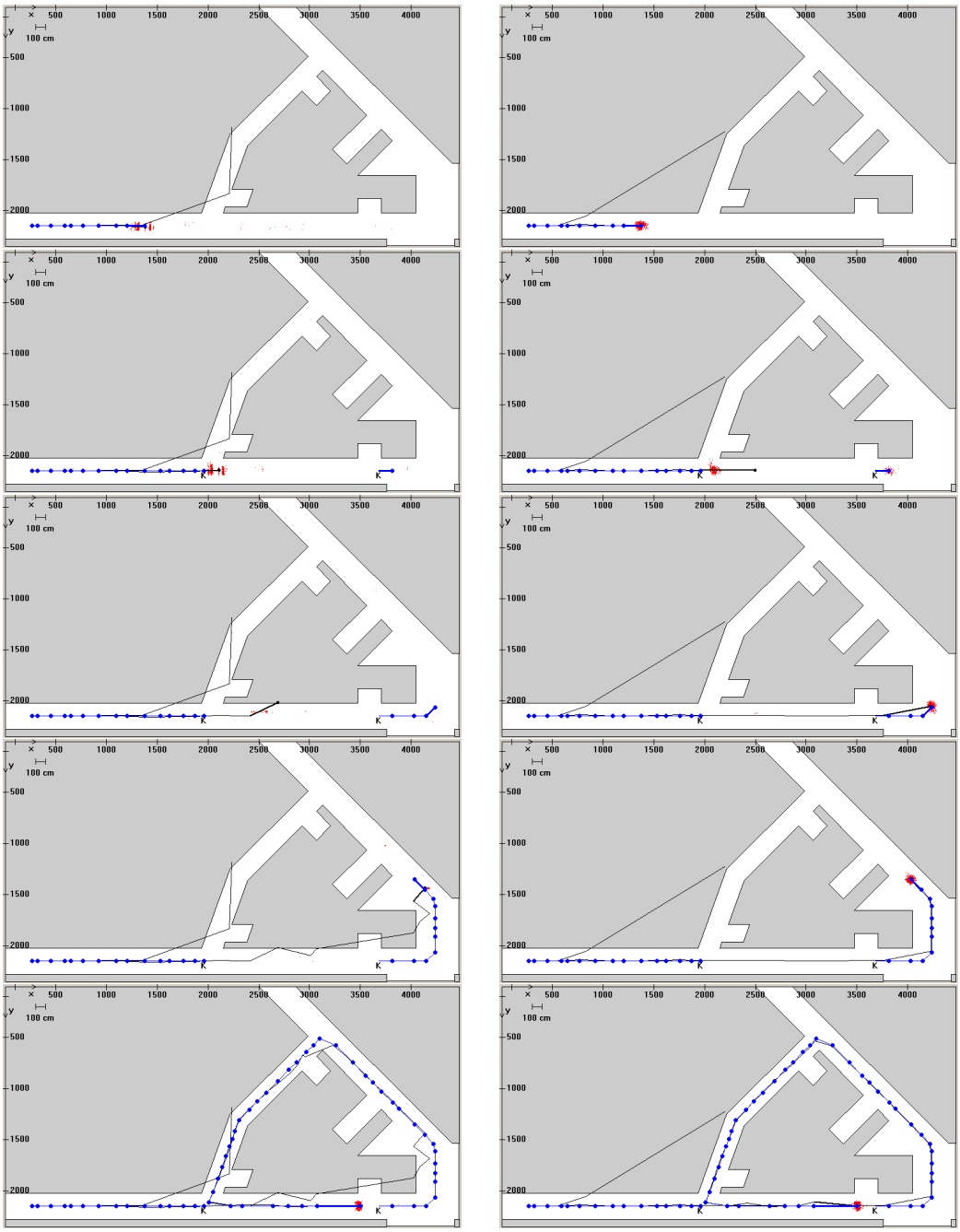
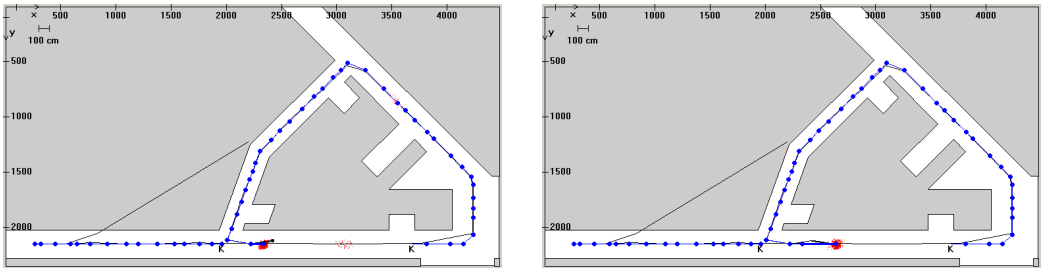


Figura 5.13: Snapshot dell'esperimento sul percorso con rapimento nell'ambiente *Test5*. La serie di immagini a sinistra si riferisce alla MCL con la strategia uniforme per i kidnapping, mentre la serie di destra mostra la MCL con la strategia basata sulla localizzazione topologica, da noi proposta.



(a) (b)

Figura 5.14: Campioni generati dalla "Topological strategy" attorno alle ipotesi probabili: le ipotesi scompaiono nel passo successivo.

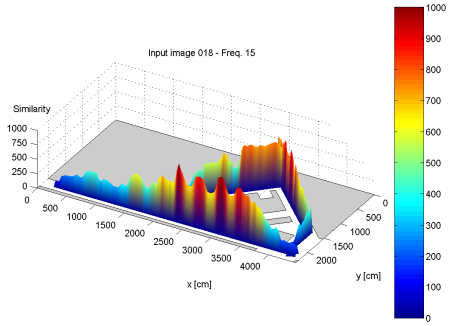
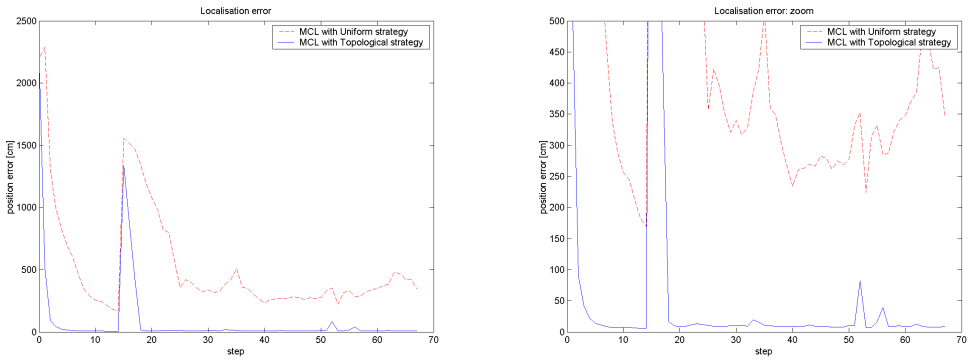


Figura 5.15: "Perceptual aliasing" per l'immagine di input numero 18 nell'ambiente *Test5*. Le false ipotesi sono quelle corrispondenti ai picchi rosso scuro.

dopo il rapimento. Il grafico (b) è uno zoom del precedente e permette di apprezzare la migliore precisione della stima ottenuta con il nostro sistema e la stabilità dell'errore una volta che è stato localizzato il robot. Infatti la linea rossa corrispondente alla strategia standard è molto meno stabile cioè l'errore tende ad aumentare quando alcuni dei campioni generati a caso sopravvivono al ricampionamento anche se sono lontani dalla vera posizione.



(a) (b)

Figura 5.16: Errore di localizzazione lungo il percorso di Figura 5.11. La "Topological strategy" che abbiamo proposto è più veloce e più robusta nel recuperare la posizione dopo un rapimento, rispetto alla strategia standard. In (b) si vede uno zoom del grafico che permette di apprezzare l'accuratezza della nostra strategia nel rilocalizzare il robot anche dopo il rapimento.

Capitolo 6

Conclusione

In questa tesi abbiamo proposto un nuovo approccio per fondere due tecniche di localizzazione note: la localizzazione basata sulle immagini e la localizzazione di Monte Carlo. Il nostro sistema di visione si basa su un sensore di tipo omnidirezionale e le immagini vengono caratterizzate con la "Fourier signature" cioè calcolandone la Trasformata di Fourier: in questo modo otteniamo una rappresentazione compatta ed efficiente dell'immagine. Il confronto tra le immagini viene fatto calcolando una funzione di similarità come la norma $L1$ tra le "Fourier signature" delle immagini. Abbiamo dimostrato che questo sistema di calcolo della similarità è robusto alle occlusioni e permette di ottenere in automatico l'invarianza rotazionale. Inoltre utilizzando un numero di frequenze diverso per il calcolo delle similarità otteniamo una localizzazione gerarchica, cioè più o meno accurata. Il sistema di visione è stato integrato al sistema di localizzazione di Monte Carlo che utilizza dei campioni per rappresentare le ipotesi di posizione del robot. Abbiamo mostrato che la MCL integrata al nostro sistema di visione omnidirezionale è superiore alle tecniche già proposte da altri ricercatori che si basano su sensori di tipo prospettico. L'approccio proposto per affrontare il problema del robot rapito, che si basa sulla generazione di nuovi campioni secondo le informazioni ottenute dalla visione, velocizza la rilocalizzazione dopo il rapimento e anche la localizzazione globale del robot, rispetto alla tecnica standard che genera i campioni in modo uniforme. Sfruttando la localizzazione gerarchica possiamo decidere di considerare più ipotesi di posizione, valutate poi tramite l'algoritmo MCL nel modo normale. Abbiamo presentato i test effettuati con i dati acquisiti da un robot olonomo in due ambienti: un piccolo laboratorio e un lungo corridoio. I test hanno confermato il funzionamento del nostro sistema anche nel caso di forte "perceptual aliasing", in cui la localizzazione basata sulle immagini fallisce: l'utilizzo della tecnica di Monte Carlo per gestire l'incertezza permette di affrontare con successo la localizzazione anche in ambienti

altamente periodici. Inoltre abbiamo dimostrato che utilizzando un sistema di visione omnidirezionale, che fornisce una localizzazione più affidabile rispetto ad uno prospettico, possiamo realizzare una localizzazione di Monte Carlo che non ha bisogno della mappa geometrica dell'ambiente.

6.1 Sviluppi futuri

Possiamo individuare una limitazione del nostro metodo nel fatto che nella nostra implementazione della localizzazione di Monte Carlo non abbiamo considerato l'orientazione del robot, e quindi dei campioni. Questo è stato possibile in quanto abbiamo usato un sensore di visione omnidirezionale: l'invarianza rotazionale garantisce che una scena venga riconosciuta indipendentemente dall'orientazione del robot. Riteniamo comunque che integrando anche le informazioni di orientazione delle immagini il sistema possa ottenere un ulteriore miglioramento nella localizzazione globale, consentendo di eliminare più velocemente le ipotesi poco probabili.
Inoltre i risultati sono stati ottenuti utilizzando un programma di simulazione: nonostante i dati di input usati siano dati acquisiti da un robot reale, riteniamo che il passaggio dalla simulazione ai robot reali sia un passo fondamentale per dimostrare definitivamente la validità del metodo di fusione proposto. Quindi l'attività di ricerca futura si rivolgerà alla verifica "sul campo" dell'approccio.
Altro campo che riteniamo interessante è quello dello studio del comportamento del sistema in ambienti outdoor, in particolare lo studio dell'influenza dei pattern di luminosità sul nostro sistema di localizzazione.

Appendice A

Richiami di Teoria della Probabilità

Si richiamano di seguito alcune nozioni di teoria della probabilità utilizzate nei precedenti capitoli. Per approfondire si faccia riferimento ad un qualsiasi testo di probabilità. In particolare si può consultare [Dall'Aglio, 1987].

A.1 Probabilità condizionata

Definizione di probabilità condizionata
Sia dato un evento C possibile, cioè di probabilità strettamente positiva. Per ogni evento A si definisce la *probabilità condizionata* di A rispetto all'*evento condizione* C come:

$$P[A|C] \doteq \frac{P[A,C]}{P[C]} \quad .$$

Proprietà notevole
La probabilità congiunta di due eventi A e B condizionati da C si scrive:

$$P[A,B|C] = P[A|B,C]\, P[B|C] \quad . \tag{A.1}$$

A.2 Probabilità Totale

Teorema della Probabilità Totale
Se gli eventi A_i, $i = 1, 2, \ldots$ formano una partizione dell'evento C (quindi sono incompatibili tra loro cioè disgiunti) e se l'evento $B \subseteq C$ allora:

$$P[B] = \sum_i P[B, A_i] = \sum_i P[B|A_i]\, P[A_i] \quad . \tag{A.2}$$

Per variabili aleatorie x, y continue di densità di probabilità f_x, f_y e densità di probabilità condizionata $f_{y|x}$, si scrive:

$$f_y(b) = \int_{-\infty}^{\infty} f_{y|x}(b|a) f_x(a) da \quad .$$

A.3 Regola di Bayes

Regola di Bayes
Dati due eventi di probabilità positiva B e C vale:

$$P[C|B] = \frac{P[B|C]\, P[C]}{P[B]} \quad . \tag{A.3}$$

Se gli eventi A_i, $i = 1, 2, \ldots$ formano una partizione dell'evento certo:

$$P[A_i|B] = \frac{P[B|A_i]\, P[A_i]}{\sum_j P[B|A_j] P[A_j]} \quad . \tag{A.4}$$

Nel caso di più eventi condizione:

$$P[C|B_1, B_2] = \frac{P[B_2|C, B_1]\, P[C|B_1]}{P[B_2|B_1]} \quad . \tag{A.5}$$

Analoghe equazioni valgono per le densità di probabilità di variabili aleatorie condizionate.

A.4 Processi di Markov

Definizione
Sia $s(t), t \in \mathcal{T}$ processo aleatorio. Sia t_0 il "presente", C un evento che descrive l'evoluzione per tempi "futuri" (dopo t_0) e B un evento che descrive l'evoluzione per tempi "passati" (prima di t_0) allora vale:

$$P[C|s(t_0) = a, B] = P[C|s(t_0) = a] \quad , \tag{A.6}$$

dove $C = \{s(u_1) \in C_1, \ldots\}$ con $u_1 > t_0$ e $B = \{s(v_1) \in B_1, \ldots\}$ con $v_1 < t_0$.

In linguaggio non rigoroso, se è noto il presente la conoscenza statistica del processo nel futuro è indipendente dalla storia del processo nel passato.

Una definizione equivalente evidenzia che se è noto il presente allora eventi del futuro e del passato sono indipendenti:

$$\begin{aligned} P[C, B|s(t_0) = a] &= P[C|s(t_0) = a, B]\ P[B|s(t_0) = a] \\ &= P[C|s(t_0) = a]\ P[B|s(t_0) = a] \quad . \end{aligned} \tag{A.7}$$

Si osserva infine che la validità delle (A.6) e (A.7) è assicurata solo se il presente è completamente noto, cioè specificato puntualmente $\{s(t_0) = a\}$ mentre non è garantita con l'evento $\{s(t_0) \in A\}$.

A.5 Regola marginale

Sia $\mathbf{x} = [x_1, \ldots, x_n]$ un vettore aleatorio di densità di probabilità $f_{\mathbf{x}}(\mathbf{a})$. Se si scrive $\mathbf{x} = [\widetilde{\mathbf{x}}, \widehat{\mathbf{x}}]$, dividendo le componenti in due sottovettori, allora la densità del vettore $\widetilde{\mathbf{x}}$ si ottiene integrando la densità $f_{\mathbf{x}}(\mathbf{a})$ rispetto a tutte le componenti che non compaiono in $\widetilde{\mathbf{x}}$ quindi:

$$f_{\widetilde{\mathbf{x}}}(\widetilde{\mathbf{a}}) = \int f_{\mathbf{x}}(\mathbf{a}) d\,\widehat{a} \quad ; \tag{A.8}$$

ad esempio se $\mathbf{x} = [x_1, x_2, x_3]$ allora per la v.a. x_1 si ha:

$$f_{x_1}(a_1) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{\mathbf{x}}(a_1, a_2, a_3) da_2 da_3 \quad . \tag{A.9}$$

Appendice B

Robot di riferimento

Il robot di riferimento è *Barney*: questo robot mobile è disponibile presso lo IAS-Lab, "Intelligent Autonomous Systems Laboratory" dell'Università di Padova e viene utilizzato dalla squadra *Artisti Veneti* che partecipa alla competizione *RoboCup* nella categoria "middle size".
Barney è un robot mobile di tipo "Golem", olonomo cioè in grado di muoversi in ogni direzione senza dover prima ruotare. Il sistema di visione è di tipo omnidirezionale.
In Figura B.1 si vede un'immagine di Barney: si può notare in primo piano una delle tre ruote di tipo svedese che può muoversi in ogni direzione. In Figura B.2 si vede lo specchio omnidirezionale Si può notare il particolare profilo non uniforme; questo garantisce delle risoluzioni diverse per le diverse zone dell'immagine omnidirezionale: risoluzione maggiore per i particolari vicini mentre minore per quelli lontani. Si faccia riferimento a [Menegatti *e altri*, 2002a; Menegatti e Pagello, 2001] per approfondimenti sulle caratteristiche dello specchio.
Di seguito descriviamo alcune delle principali informazioni tecniche di Barney:

- il PC interno è costituito da una scheda PC104 con una scheda adattatrice per PCI;
- la CPU è AMD K6 400 MHz;
- memoria flash da 256 MB al posto dell'Hard disk;
- sistema WaveLan standard IEEE 802.11;
- tre ruote omnidirezionali di tipo svedese da 8 cm;
- tre motori Maxon con gli encoder;

- telecamera analogica Hitachi con frame grabber PCI (la telecamera è orientata verso l'alto, cioè verso lo specchio);
- specchio omnidirezionale in alluminio;
- distribuzione personalizzata linux con kernel 2.4.20.

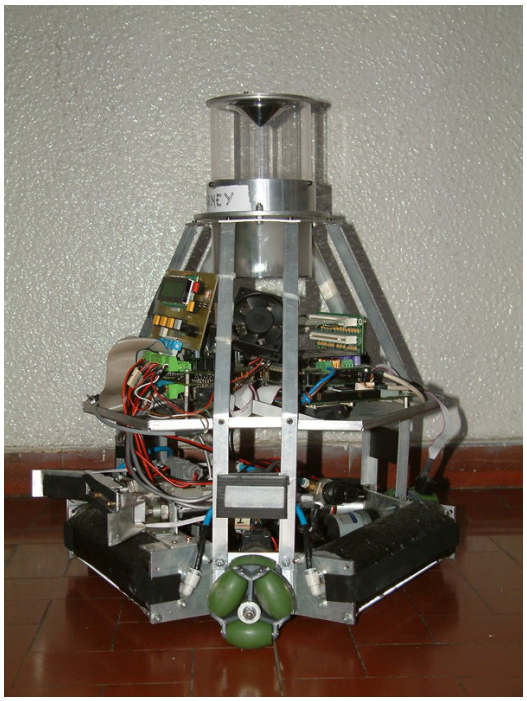


Figura B.1: Immagine del robot mobile Barney. Da notare in primo piano una delle tre ruote svedesi che permette il movimento in qualsiasi direzione. Inoltre si vede lo specchio alla sommità del robot. A fianco delle ruote si notano le bombole per l'aria compressa necessarie per azionare il kicker che si vede sulla sinistra.

Figura B.2: Specchio omnidirezionale. Si vede la particolare forma dello specchio con una sagomatura non uniforme; questa sagoma irregolare permette di avere risoluzioni diverse nel campo visivo del robot: risoluzione maggiore da vicino mentre minore per gli oggetti lontani.

Appendice C

Programma di simulazione della MCL

C.1 Algoritmo di MCL

L'algoritmo di localizzazione di Monte Carlo è stato implementato utilizzando il linguaggio di programmazione C++ Standard ISO. Per la compilazione durante lo sviluppo abbiamo utilizzato DJGPP v.2.01 contenente il compilatore gcc 2.95.2.
Nella progettazione abbiamo seguito uno schema di progettazione orientata agli oggetti. Abbiamo scritto interamente anche tutte le classi di supporto e le molte funzioni di utilità. Per il linguaggio C++ abbiamo fatto riferimento a [Lippman, 1993] e [Eckel, 2000].
Per la generazione di numeri casuali secondo le distribuzioni uniforme e gaussiana abbiamo utilizzato una libreria scritta in C++ da Maurizio Loreti [Loreti, 2003], utilizzabile secondo la licenza GNU GPL.

C.2 Simulatore grafico di MCL

Per verificare il funzionamento della localizzazione di Monte Carlo abbiamo realizzato un'applicazione grafica basata su finestre di dialogo, in ambiente Microsoft Visual C++ 6.0® per Windows®. Le librerie grafiche utilizzate sono le Microsoft Foundation Class (MFC) per lo sviluppo di applicazioni Windows®. Per queste abbiamo fatto riferimento a [Bates e Tompkins, 1998]. Le classi che realizzano la MCL sono state integrate all'interno del programma grafico.

L'applicazione grafica realizzata, che di seguito chiameremo *simulatore*, permette la simulazione della MCL con dati reali. Il simulatore prende in

input tutti i dati necessari per descrivere dell'ambiente di movimento da un file di configurazione. Inoltre prende in input i dati della visione relativi al robot reale. È possibile assegnare al robot un percorso qualsiasi, che passi per le posizioni di input disponibili. È possibile eseguire in automatico (batch) un singolo test oppure una serie di test ripetuti su un percorso fissato, memorizzando i principali parametri che riguardano la stima della posizione del robot, ad esempio l'errore rispetto alla vera posizione[1]. In Figura C.1 si può vedere uno snapshot dell'interfaccia del simulatore. Tutti gli snapshot che abbiamo mostrato nei capitoli precedenti sono stati acquisiti durante il funzionamento di questo programma. Ricordiamo infine che i grafici di similarità e degli errori di localizzazione sono stati realizzati con MATLAB 6®.

Di seguito descriviamo brevemente le principali funzionalità della versione v3 del simulatore, aggiornata al 18 Aprile 2003.

Opzioni di configurazione:

- pulsante di opzione *Localizzazione*: permette di scegliere il tipo di localizzazione da effettuare tra localizzazione globale (GL) e position tracking (PT). La differenza consiste solo nella inizializzazione dei campioni, uniforme in tutto l'ambiente nella GL mentre gaussiana centrata attorno alla posizione nota nel PT. L'opzione di default è GL;

- pulsante di opzione *Metodo di stima*: permette di scegliere se effettuare la stima della posizione calcolando la media semplice dei campioni oppure calcolando la moda cioè il campione più frequente[2]. L'opzione di default è la stima con la media;

- casella di controllo *Visualizza Resampling*: se selezionata permette di visualizzare l'effetto del ricampionamento sull'insieme dei campioni. In pratica i campioni vengono aggiornati con il passo di predizione, vengono disegnati per 1 secondo (il programma effettua uno "sleep"), poi viene effettuato i ricampionamento e i campioni vengono ridisegnati: si vede che alcuni campioni scompaiono (sono quelli con peso basso che vengono eliminati; vedi il capitolo 3) mentre non si apprezza il fatto che i campioni con peso alto vengono scelti più volte, in quanto le occorrenze di uno stesso campione vengono disegnate l'una sopra l'altra. Di default l'opzione è deselezionata;

[1] Come errore di localizzazione abbiamo considerato la distanza tra la posizione stimata e quella reale del robot.

[2] La stima viene fatta sempre dopo il passo di ricampionamento, quindi con i campioni i.i.d che hanno peso pari a $1/N$.

- casella di controllo *Ctrl NotAllow Area*: se selezionata permette di controllare che i campioni non vengano generati in zone non permesse dell'ambiente, ad esempio nelle zone non accessibili. Queste zone sono indicate nel file di configurazione dell'ambiente, letto dal simulatore all'esecuzione. Di default l'opzione è deselezionata;

- caselle di controllo *Uniform Kidnap Strategy* e *Topological Kidnap Strategy*: permettono di effettuare una MCL con una delle due strategie per il robot rapito. Si può selezionare una sola strategia alla volta mentre possono essere entrambe deselezionate per eseguire una MCL standard (valore di default);

- caselle di controllo *Visualizza* presenti nella legenda: permettono di visualizzare o meno alcuni degli elementi grafici disegnati; questi sono: il percorso reale, il percorso da odometria, il percorso stimato e i campioni. Di default sono tutti selezionati.

Pulsanti di configurazione:

- pulsante *Inserisci il percorso*: permette di accedere alla finestra di dialogo per la scelta del percorso da assegnare al robot. Nella versione attuale del simulatore, usata per il *Test5*, questa è l'unica parte legata all'ambiente di test[3];

- pulsante *Configura i parametri*: permette di accedere alla finestra di configurazione dei principali parametri dell'algoritmo di localizzazione MCL (numero campioni, parametri di predizione, ...), delle strategie per il rapimento e dell'errore introdotto nell'odometria. Uno snapshot della finestra si può vedere in Figura C.2.

Altri pulsanti:

- pulsante *Inizia la simulazione*: serve per inizializzare tutte le informazioni che devono essere fissate prima dell'inizio di esperimento di localizzazione. Con la pressione del pulsante viene disabilitata la selezione di tutte le opzioni che riguardano la MCL ma non le opzioni di visualizzazione, mentre vengono abilitati i vari pulsanti per eseguire la simulazione;

- pulsante *Muovi*: permette di eseguire l'esperimento un passo alla volta;

[3]Per cambiare l'ambiente di test è necessario ridefinire questa parte del programma ed effettuare un rebuild. Sono già state approntate le funzionalità per rendere anche questa parte, indipendente dall'ambiente.

- pulsante *Start BATCH*: permette di eseguire l'intero esperimento in automatico, senza dover premere altri pulsanti;

- pulsante *Start TEST* e casella *N. Test*: nella casella è possibile inserire il numero di esperimenti ripetuti da effettuare in automatico; l'inizio della serie di esperimenti avviene con la pressione del pulsante. In questa modalità viene salvato un file con i parametri dell'esperimento e l'errore di localizzazione ad ogni passo per ciascun esperimento effettuato. Il file viene scritto in un formato adatto per essere importato in MATLAB 6®.

Nell'interfaccia principale del programma, in alto a sinistra, vengono visualizzati i parametri principali della MCL e le informazioni riguardanti la posizione stimata, quella reale e l'errore di localizzazione ad ogni passo. La parte principale dell'interfaccia è invece occupata dall'area di disegno in cui viene rappresentata graficamente la situazione di navigazione con localizzazione del robot.

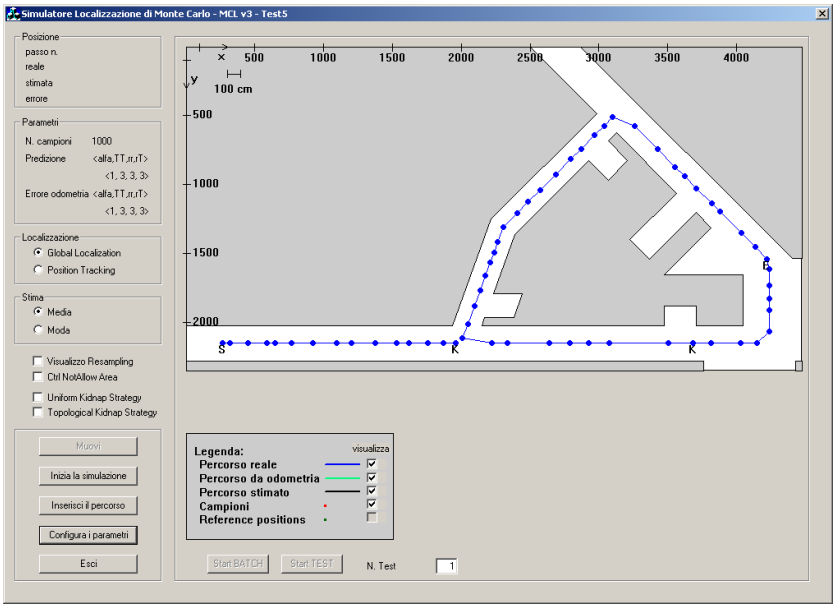


Figura C.1: Snapshot dell'applicazione grafica che abbiamo sviluppato per le simulazioni della localizzazione di Monte Carlo con i dati di visione reali.

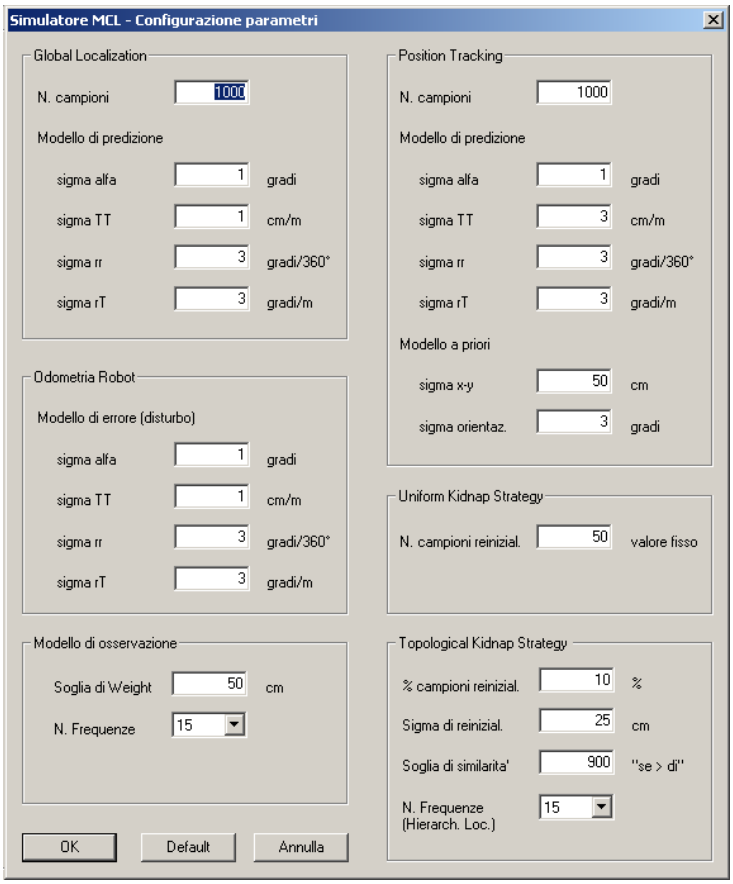


Figura C.2: Snapshot della finestra di configurazione del simulatore che consente la scelta dei parametri della MCL.

Elenco delle figure

Bibliografia

Aihara H.; Iwasa N.; Yokoya N.; Takemura H. (1998). Memory-based self-localisation using omnidirectional images In *Proceedings of the 14th International Conference on Pattern Recognition*. A cura di Jain A. K., Venkatesh S., Lovell B. C., volume I, pp. 1799–1803.

Arulampalam S.; Maskell S.; Gordon N.; Clapp T. (2002). A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, **50**(2), 174–188.

Bates J.; Tompkins T. (1998). *Visual C++ 6*. McGraw-Hill, Milano, 1^a edizione.

Borghi G. (1997). *Un Modello dell'Interazione con l'Ambiente per un Robot Autonomo Dotato di un Sistema Senso-Motorio Non Interpretato*. Tesi di Dottorato di Ricerca, Dipartimento di Elettronica e Informazione, Università degli Studi di Padova, Italia. In Italiano.

Burgard W.; Fox D.; Henning D.; Schmidt T. (1996). Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 896–901.

Burgard W.; Cremers A. B.; Fox D.; Hähnel D.; Lakemeyer G.; Schulz D.; Steiner W.; Thrun S. (1999). Experiences with an Interactive Museum Tour-Guide Robot. *Artificial Intelligence (AI)*, **114**(1–2), 3–55.

Burgard W.; Fox D.; Moors M.; Simmons R.; Thrun S. (2000). Collaborative multi-robot exploration. In *IEEE Proceedings of the International Conference on Robotics & Automation (ICRA)*.

Cariolaro G. (1996). *La Teoria Unificata dei Segnali - Nuova edizione*. UTET Libreria, Torino, 2^a edizione.

Carpenter J.; Clifford P.; Fearnhead P. (1999). Improved Particle Fiter for Nonlinear Problems. In *IEE Proceedings on Radar and Sonar Navigation*, volume 146, pp. 2–7.

Cassinis R.; Duina D.; Inelli S.; Rizzi A. (2002). Unsupervised Matching of Visual Landmarks for Robotic Homing unsing Fourier-Mellin Transform. *Robotics and Autonomous Systems*, **40**(2–3), 131–138.

Choset H.; Nagatani K. (2001). Topological Simultaneous Localisation and Mapping (SLAM): Toward Exact Localization Without Explicit Localization. *IEEE Transactions on Robotics and Automation*, **17**(2), 125–137.

Collett T.; Dillmann E.; Giger A.; Wehner R. (1992). Visual Landmarks and Route Following in Desert Ants. *Journal of Comparative Physiology A*, **170**, 435–442.

Cox I. J. (1991). Blanche - An Experiment in Guidance and Navigation of an Autonomus Robot Vehicle. *IEEE Transactions on Robotics and Automation*, **7**(2), 193–204.

Cox I. J.; Wilfong G. T., (A cura di) (1990). *Autonomous Robot Vehicles*. Springer Verlag.

Crisan D. (2001). Particle Filters - A Theoretical Perspective In *Sequential Monte Carlo Methods in Practice*. A cura di Doucet A., de Freitas N., Gordon N., capitolo 2, pp. 18–41. Springer, New York.

Crisan D.; Doucet A. (2000). Convergence of Sequential Monte Carlo Methods. Technical report, Cambridge University Engineering Department, CUED/F-INFENG/TR381.

Dall'Aglio G. (1987). *Calcolo delle Probabilità*. Zanichelli, Bologna.

de Berg M.; van Kreveld M.; Overmars M.; Schwarzkopf O. (2000). *Computational Geometry - Algorithms and Applications*. Springer, Berlino, 2^a edizione.

Dellaert F.; Burgard W.; Fox D.; Thrun S. (1999). Using the Condensation Algorithm for Robust, Vision-based Mobile Robot Localization. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR)*.

Doucet A. (1998). On Sequential Simulation-Based Methods for Bayesian Filtering. Technical report, Cambridge University Department of Engineering, CUED/F-INFENG/TR 310.

Doucet A.; Godsill S.; Andrieu. C. (2000). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, **10**(3), 197–208.

Doucet A.; de Freitas N.; Gordon N. (2001a). An Introduction to Sequential Monte Carlo Methods In *Sequential Monte Carlo Methods in Practice*. A cura di Doucet A., de Freitas N., Gordon N., capitolo 1, pp. 3–14. Springer, New York.

Doucet A.; de Freitas N.; Gordon N., (A cura di) (2001b). *Sequential Monte Carlo Methods in Practice*. Springer, New York.

Eckel B. (2000). *Thinking in C++*, volume 1,2. Prentice Hall, New Jersey, 2^a edizione.

Engelson S. (1994). *Passive map learning and visual place recognition*. Ph.D. Thesis, Department of Computer Science, Yale University. In Inglese.

Fearnhead P. (1998). *Sequential Monte Carlo Methods in filter theory*. Ph.D. Thesis, Merton College, University of Oxford. In Inglese.

Fox D. (2001). KLD-Sampling: Adaptive Particle Filters. In *Advances in Neural Information Processing Systems 14 (NIPS)*. MIT Press.

Fox D. (2002). KLD-Sampling: Adaptive Particle Filters and Mobile Robot Localization. Technical report, University of Washington Department of Computer Science & Engineering.

Fox D.; Burgard W.; Thrun S. (1999a). Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research (JAIR)*, **11**, 391–427.

Fox D.; Burgard W.; Dellaert F.; Thrun S. (1999b). Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Fox D.; Burgard W.; Kruppa H.; Thrun S. (2000). A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots*, **8**(3), 325–344.

Fox D.; Thrun S.; Burgard W.; Dellaert F. (2001). Particle Filters for Mobile Robot Localization In *Sequential Monte Carlo Methods in Practice*. A cura di Doucet A., de Freitas N., Gordon N., capitolo 19, pp. 401–428. Springer, New York.

Gaspar J.; Winters N.; Santos-Victor J. (2000). Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, **16**(6).

Gordon N.; Salmond D.; Smith A. F. M. (1993). Novel Approach to Nonlinear and Non-Gaussian Bayesian State Estimation. In *IEE Proceedings-F*, volume 140, pp. 107–113.

Gutmann J.-S.; Fox D. (2002). An Experimental Comparison of Localization Methods Continued. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne.

Gutmann J.-S.; Burgard W.; Fox D.; Konolige K. (1998). An Experimental Comparison of Localization Methods. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Hu H.; Gu D. (2000). Landmark-based localisation of industrial mobile robots. *International Journal of Industrial Robot*, **27**(6), 458–467.

Isard M.; Blake A. (1998). Condensation - Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, **29**(1), 5–28.

Ishiguro H. (2001). Development of Low-cost Compact Omnidirectional Vision Sensors In *Panoramic Vision*. A cura di Benosman R., Kang S., capitolo 3, pp. 23–38. Springer.

Ishiguro H.; Tsuji S. (1996). Image-based memory of environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-96)*, pp. 634–639.

Jogan M.; Leonardis A. (2000). Robust localization using panoramic view-based recognition. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR00)*, volume 4, pp. 136–139. IEEE Computer Society.

Kitagawa G. (1996). Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, **5**(1), 1–25.

Kröse B. J. A.; Vlassis N.; Bunschoten R.; Motomura Y. (2001). A probabilistic model for appareance-based robot localization. *Image and Vision Computing*, **19**(6), 381–391.

Kuipers B. (2000). The Spatial Semantic Hierarchy. *Artificial Intelligence (AI)*, **119**, 191–233.

Lenser S.; Veloso M. (2000). Sensor Resetting Localization for Poorly Modelled Mobile Robots. In *IEEE Proceedings of the International Conference on Robotics & Automation (ICRA)*.

Lippman S. B. (1993). *C++ Corso di programmazione*. Addison-Wesley, Milano, 2^a edizione.

Loreti M. (2003). Maurizio Loreti Home Page, http://wwwcdf.pd.infn.it/ loreti/mlo.html, Aprile 2003.

Menegatti E.; Pagello E. (2001). Designing omnidirectional vision systems and inserting them in a distributed vision system for mobile robots In *Enabling Technologies for the PRASSI Autonomous Robot*. A cura di Taraglio S., Nanni V., pp. 98–108. ENEA.

Menegatti E.; Nori F.; Pagello E.; Pellizzari C.; Spagnoli D. (2002a). Designing an omnidirectional vision system for a goalkeeper robot In *RoboCup-2001: Robot Soccer World Cup V*. A cura di Birk A., Coradeschi S., Tadokoro S., pp. 78–87. Springer.

Menegatti E.; Maeda T.; Ishiguro H. (2002b). Image-based Memory for Robot Navigation Using Properties of the Omnidirectional Images. Sottomesso ad una Rivista Internazionale.

Menegatti E.; Zoccarato M.; Pagello E.; Ishiguro H. (2003a). Hierarchical Image-based Localisation for Mobile Robots with Monte-Carlo Localisation. Sottomesso ad una Conferenza Internazionale.

Menegatti E.; Zoccarato M.; Pagello E.; Ishiguro H. (2003b). Image-Based Monte-Carlo Localisation without a Map. Sottomesso ad una Conferenza Internazionale.

Nourbakhsh I.; Powers R.; Birchfield S. (1995). DERVISH an office-navigating robot. *AI Magazine*, **16**(2).

Preparata F. P.; Shamos M. I. (1985). *Computational Geometry - An Introduction*. Springer-Verlag, New York.

Russell S. J.; Norving P. (1995). *Artificial Intelligence - A Modern Approach*. Prentice Hall, New Jersey.

Simmons R.; Koenig S. (1995). Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Thrun S.; Beetz M.; Bennewitz M.; Burgard W.; Cremers A. B.; Dellaert F.; Fox F. D.; Hähnel D.; Rosenberg C.; Roy N.; Schulte J.; Schulz D. (2000). Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. *International Journal of Robotics Research*, **19**(11), 972–999.

Wolf J. (2001). *Bildbasierte Lokalisierung für mobile Roboter*. Master's thesis, Department of Computer Science, University of Freiburg, Germany. In Tedesco.

Wolf J.; Burgard W.; Burkhardt H. (2002a). Robust Vision-based Localization for Mobile Robots Using an Image Retrieval System Based on Invariant Features. In *IEEE Proceedings of the International Conference on Robotics & Automation (ICRA)*, pp. 359–365.

Wolf J.; Burgard W.; Burkhardt H. (2002b). Using an Image Retrieval System for Vision-based Mobile Robot Localization. In *Proceedings of the International Conference on Image and Video Retrieval (CIVR)*.