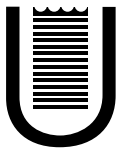


UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA
IN INGEGNERIA DELL'AUTOMAZIONE

TESI DI LAUREA

LOCALIZZAZIONE DI ROBOT MOBILI MEDIANTE FILTRI PARTICELLARI

RELATORE
Ing. F.MARTINELLI

CANDIDATO
CARLO PARRONI

ANNO ACCADEMICO 2006/2007

Indice

Introduzione

Il lavoro qui svolto concentra la sua attenzione sulla robotica mobile, scienza che rende un robot capace di muoversi liberamente e in piena autonomia nella'ambiente che lo circonda. Un robot industriale conosce la propria posizione, a meno di piccoli errori, mentre un robot mobile può non avere a disposizione tale informazione e deve essere in grado di misurarla e stimarla rispetto a punti di riferimento noti e in base al cammino fatto (*dead reckoning*). Quando robot mobili si trovano ad operare in un ambiente nel mondo reale richiedono dei sistemi di localizzazione affidabili per poter svolgere al meglio il compito che viene loro assegnato. Per questo nasce l'esigenza di dover far capire al robot con discreta precisione dove si trova. Riuscire a fornire al robot queste informazioni è pertanto fondamentale per realizzarne la più completa autonomia ed è quindi sempre strettamente necessario un buon algoritmo di localizzazione da includere all'interno di un'architettura di controllo di un robot.

A causa delle difficoltà che si incontrano nel cercare di ottenere una stima affidabile della posizione, negli ultimi due decenni la localizzazione è stata, e continua ad essere tutt'oggi, un campo di ricerca molto attivo. Le principali insidie sono rappresentate dalla necessità di superare le incertezze dovute alla non perfetta conoscenza del sistema robot - ambiente e di integrare al meglio i dati provenienti da differenti tipi di sensori. L'acquisizione crescente di dati, l'abilità di riconoscere oggetti o luoghi familiari, il saper dare risposte *real time*, sono solo alcune delle capacità che devono

essere sviluppate per poter garantire una risoluzione soddisfacente del problema.
Due sono le principali situazioni in cui ci si può imbattere affrontando il problema della localizzazione: nella prima il robot deve riuscire a stimare dove si trova partendo completamente da zero, ovvero senza nessuna conoscenza *a priori* riguardo alla propria posizione di partenza (*global localization*), mentre nell'altra si assume che la configurazione iniziale sia approssimativamente nota e obiettivo principale è quello di monitorare l'evoluzione della posizione del robot (*position tracking*).
Altro aspetto fondamentale è assunto dalla percezione. Per un essere umano identificare la propria posizione in un ambiente chiuso è molto semplice: chiunque è in grado di indicare in modo generico dove si trovi all'interno di un ambiente che sta osservando, o di localizzarsi relativamente agli oggetti che lo circondano. Non si può dire altrettanto per un robot. In questo caso, infatti, bisogna considerare che il ragionamento è rappresentato da algoritmi, da principi computazionali, e che i dispositivi sensoriali che percepiscono l'ambiente circostante e gli algoritmi usati nell'interpretazione dei valori misurati raggiungono un grado di complessità molto elevato. Una buona percezione è, dunque, anche per i più semplici comportamenti robotici, un requisito imprescindibile, in quanto le informazioni continuamente percepite da un robot sono l'unico mezzo con il quale esso riesce a interagire con il mondo esterno. I sensori possono misurare grandezze che ne determinano poi il movimento, quali la sua velocità, le forze ad esso applicate, oppure la struttura dell'ambiente che lo circonda. Tra i numerosi sensori dedicati all'osservazione dell'ambiente si possono distinguere sensori visivi, sensori di distanza e di prossimità, sensori di contatto e di orientamento.
Alla luce di quanto finora detto, per costruire un buon sistema di localizzazione, i fattori che risultano determinanti per stimare la posizione di un robot sono essenzial-

mente due: la qualità delle sorgenti sensoriali e la rappresentazione dello spazio.

Questa tesi tratta proprio del problema della localizzazione, affrontato e risolto mediante l'utilizzo di filtri particellari. Il lavoro è organizzato in sei capitoli. Nel primo capitolo viene presentato il problema della localizzazione di un robot mobile. L'attenzione viene focalizzata sulla *localizzazione probabilistica*, con particolare interesse per l'approccio bayesiano al problema. Nel secondo capitolo si affrontano le tematiche dei metodi Monte Carlo e dei filtri particellari, utilizzati per risolvere problemi di stima. Nel terzo capitolo si descrive il modello di sistema e il modello dei sensori e le varie fasi che hanno portato alla realizzazione dell'algoritmo di localizzazione. Il quarto capitolo descrive il *filtro particellare con filtro di Kalman esteso* (*EKPF*) e il *filtro particellare adattativo* (*APF*), evidenziando eventuali vantaggi e svantaggi delle tecniche presentate. Nel quinto capitolo sono riportati i risultati delle simulazioni, ottenuti con il software realizzato, e il confronto tra le tecniche di localizzazione implementate. Le conclusioni ed eventuali sviluppi futuri riguardanti l'argomento trattato sono esposte nel sesto capitolo.

Capitolo 1

La localizzazione di un robot mobile

1.1 Presentazione del problema

Affrontare il problema della localizzazione consiste nel rendere un robot capace di riconoscere autonomamente la propria posizione nell'ambiente. Infatti, i robot mobili, per essere in grado di portare a termine in maniera efficiente i compiti che vengono loro assegnati, devono essere in grado di riconoscere la loro posizione all'interno dell'ambiente in cui operano. Per questo nasce l'esigenza di sviluppare algoritmi e metodi utili che riescano ad individuare il robot nello spazio. Analiticamente il problema può essere affrontato come una stima dello stato di un sistema dinamico e può essere risolto mediante diverse tecniche, alcune delle quali saranno descritte nel seguito. Sebbene esistano molti metodi specifici per stimare lo stato di un sistema a partire da un insieme di misurazioni, la maggior parte di questi non considera in maniera esplicita la natura *rumorosa* delle informazioni a disposizione, ricavate molto spesso con l'utilizzo di sensori. Proprio per questo motivo, fondamentale importanza hanno assunto nel tempo le tecniche di tipo stocastico, che affrontano il problema basandosi su strumenti matematici che prendono spunto dalla teoria bayesiana [3, 4, 7].

In letteratura si trovano numerose soluzioni al problema, come in [11, 15–17, 23].
Tali tecniche possono essere distinte in tre grandi classi, che si differenziano a seconda del tipo di problema di localizzazione che sono chiamate a risolvere:

- tracciamento della posizione (*position tracking*);
- localizzazione globale (*global localization*);
- rapimento del robot (lett. *kidnapped robot problem*).

Gli approcci che risolvono il problema del *position tracking* hanno come scopo quello di monitorare la posizione del robot mediante i dati sensoriali che ricevono ad ogni passo. Essi assumono che la configurazione iniziale sia nota, perciò ad ogni istante è possibile limitare lo spazio di ricerca delle probabili posizioni. Nonostante da un lato tale assunzione garantisca una più semplice risoluzione del problema, dall'altro non dà la possibilità di determinare la posizione del robot nel caso in cui venga commesso un grande errore nella stima all'istante precedente, in quanto l'errore si propaga nel tempo. Quindi, una volta che il robot perde la posizione è molto improbabile che sia in grado di recuperarla. Questa è una debolezza delle tecniche di localizzazione che risolvono il *position tracking* e che ne limita molto l'utilizzo, a favore delle tecniche che invece affrontano la *global localization*.
Queste ultime sono in grado di stimare la posizione del robot senza avere alcun tipo di informazione sulla locazione iniziale dello stesso, per cui anche in situazioni di globale incertezza. In questo caso è necessario ottenere informazioni che riescano a valutare e a limitare le ipotesi e siano in grado di fornire una buona stima della sua posizione attuale [5]. Il problema della localizzazione globale è più difficile del precedente, dal momento che l'errore di localizzazione del robot può essere arbitrariamente grande.
Ancora più complesso è però il *kidnapped robot problem*, nel quale un robot, di cui

si conosce la posizione, ad un certo istante viene trasportato in un altro punto senza preavviso e senza informazioni su dove viene posizionato. La differenza rispetto al problema di localizzazione globale risiede nel fatto che il robot, al momento del *rapimento*, crede con convinzione di trovarsi in tutt'altra posizione rispetto a quella nella quale viene trasportato. Questo problema viene spesso utilizzato per testare l'abilità del robot nel recuperare autonomamente la posizione in seguito ad un totale fallimento di localizzazione [8].

I tre diversi ambiti della localizzazione appena descritti sono accomunati da uno stesso obiettivo: determinare una funzione densità di probabilità, che nel seguito verrà chiamata semplicemente *pdf*, che rappresenti al meglio la reale configurazione del robot

$$p(x_t|u_{1:t}, z_{1:t}). \tag{1.1}$$

Infatti, se si indica con x_t lo stato di un sistema dinamico che descrive il comportamento di un robot mobile, con u_t l'insieme delle letture fornite dall'odometria e con z_t quelle ricavate dai sensori al tempo t, è possibile riformulare il problema della localizzazione come (1.1), in cui il pedice $1 : t$ indica tutti i vettori dall'istante iniziale fino all'istante corrente t.

Ciò che invece differenzia le varie forme sotto cui si presenta il problema è la diversa considerazione della *pdf*. Ad esempio, in un problema di *position tracking*, la funzione densità di probabilità è rappresentata mediante una distribuzione unimodale, ossia dotata di un solo massimo, e viene riformulata come

$$p(x_t|x_{t-1}, u_{1:t}, z_{1:t}) \tag{1.2}$$

dal momento che si è a conoscenza anche dello stato x_{t-1} all'istante di tempo precedente. Viceversa, in un problema di *global localization*, si utilizzano distribuzioni multimodali e ipotesi multiple proprio per evitare l'unicità della configurazione di partenza e per garantire un recupero dello stato del sistema anche in presenza di errori considerevoli.

1.2 Localizzazione probabilistica

L'idea della stima *probabilistica* dello stato risale agli anni del filtro di Kalman [14], periodo in cui si cominciano ad adottare funzioni gaussiane a più dimensioni per rappresentare la *pdf*. Prima di approfondire da un punto di vista matematico il problema, si illustrano i concetti che sono alla base della localizzazione probabilistica. Se si suppone di dover risolvere un problema di localizzazione globale, nel quale il robot non conosce la propria posizione di partenza, l'approccio probabilistico rappresenta tale incertezza mediante una *distribuzione uniforme*. A questo punto il robot per prima cosa interroga i propri sensori cercando di ottenere informazioni riguardanti l'ambiente che lo circonda. L'idea principale è quella di modificare la credenza (in letteratura *belief*), ossia la funzione densità di probabilità che descrive la posizione del robot, aumentando la probabilità di trovarsi nella configurazione che maggiormente si avvicina alle misure ricavate in precedenza e diminuendo allo stesso tempo le meno probabili. È da notare come la densità di probabilità risultante sia una funzione *multi-modale*, adatta a rappresentare il fatto che le informazioni a disposizione sono insufficienti per determinare in maniera univoca la configurazione attuale del robot. Va sottolineato inoltre che, nonostante ad alcune configurazioni del robot venga diminuita la probabilità di essere quella reale, esse ancora non possiedono una probabilità nulla. Questa è la caratteristica principale della localizzazione probabilistica, che tiene conto del

rumore che affligge le letture dei sensori e pertanto tende a non scartare, almeno inizialmente, alcuna ipotesi sulla posizione e sull'orientamento del robot. Man mano che il robot si muove e riceve nuove informazioni sull'ambiente in cui si trova, la tecnica di localizzazione adottata modifica in maniera opportuna la densità di probabilità, che viene sempre più smussata (*smooth*) passo dopo passo rispetto a quella dell'istante precedente. Procedendo con tale tecnica, la maggior parte delle probabilità saranno centrate nell'intorno di una singola configurazione e il robot, a questo punto, sarà abbastanza sicuro della posizione in cui si trova. La figura (1.1) riportata di seguito rappresenta un semplice esempio che dovrebbe chiarire quanto appena descritto.

1.3 Approccio Bayesiano alla localizzazione

In questo paragrafo viene descritto il processo di stima stocastico dello stato, partendo dalle basi della teoria bayesiana e fornendo una formulazione matematica del problema per spiegare come sia possibile effettuare una stima dello stato incrementale e online.

I *filtri bayesiani* [7,27,28] sono strumenti matematici che hanno come obiettivo quello di stimare lo stato x di un sistema dinamico mediante misure sensoriali. Nel caso della localizzazione di un robot mobile, il sistema dinamico è rappresentato dall'insieme robot-ambiente, lo stato è la posizione del robot (solitamente composto da due coordinate cartesiane che indicano la posizione nello spazio di stato e una coordinata angolare che indica l'orientamento), e le misure possono includere letture odometriche e sensoriali. Inoltre i filtri bayesiani assumono che l'ambiente sia *Markoviano*, cioè i dati passati e futuri sono considerati indipendenti fra loro se è noto lo stato corrente. L'idea chiave della teoria bayesiana è determinare la funzione densità di probabilità (*pdf*) *a posteriori* dello stato basata su tutte le informazioni disponibili, incluso l'insie-

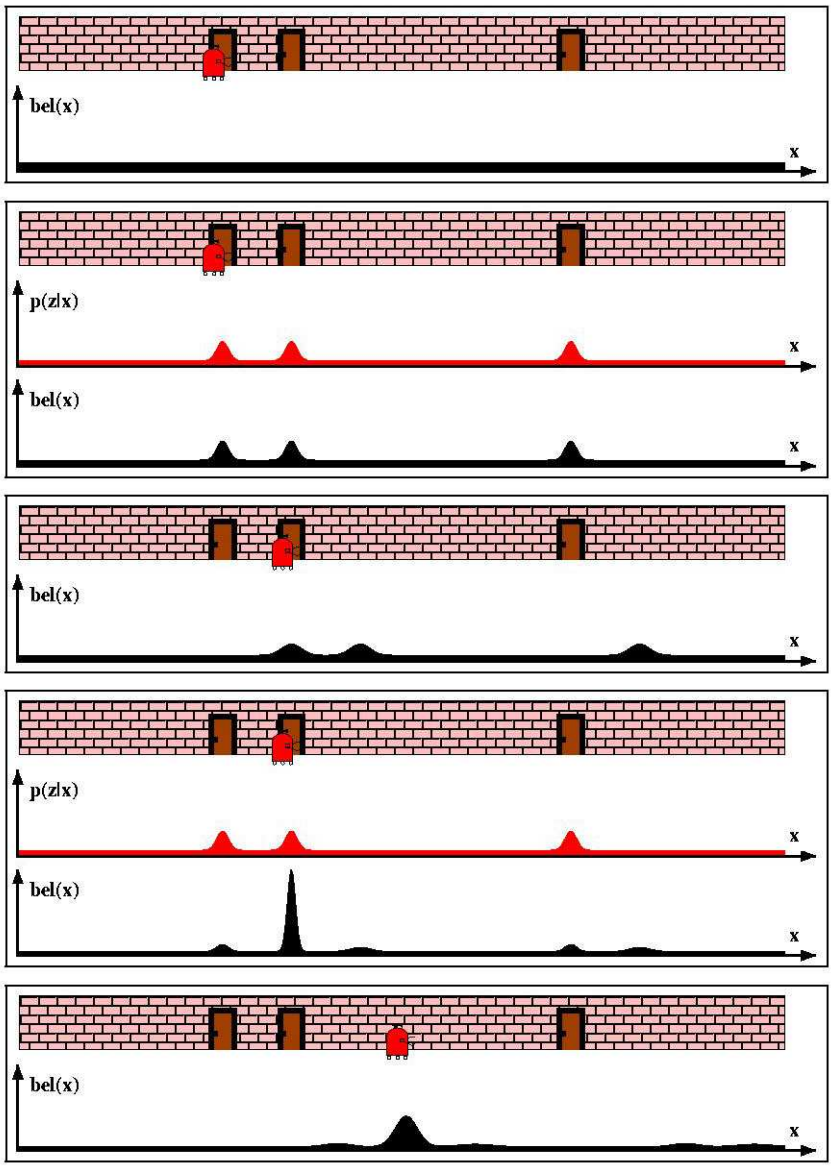


Figura 1.1: Localizzazione probabilistica. Semplice esempio di localizzazione globale.

me delle misurazioni ricevute. La distribuzione viene tipicamente indicata con *belief* ed è descritta come

$$Bel(x_k) = p(x_k|d_{0..k}). \tag{1.3}$$

Qui x_k denota lo stato all'istante $t = k$, e $d_{0..k}$ rappresentano invece i dati dall'istante iniziale $t = 0$ all'istante k. Per i robot mobili è possibile distinguere due differenti

tipi di dati: *dati percettuali*, quali sono le misure dei laser, e *dati odometrici*, come le informazioni che si ricavano direttamente dal movimento del robot. Indicando i primi con z_k, e con u_k i secondi, si ha che la (1.3) può essere riscritta come

$$Bel(x_k) = p(x_k|z_k, u_{k-1}, z_{k-1}, u_{k-2}, ..., z_0). \tag{1.4}$$

I filtri bayesiani vogliono calcolare la densità di probabilità in modo *ricorsivo*, in quanto per molti problemi è richiesta una stima ad ogni istante in cui si ricevono dati dai sensori. Quindi un filtro ricorsivo è la soluzione più conveniente, anche perchè in tal modo i dati ricevuti possono essere processati in maniera sequenziale piuttosto che continua, così da non rendere necessario nè l'immagazzinamento di tutti i dati ricevuti nè il riprocessamento dei dati se una nuova misura diventa disponibile. La *belief* all'istante $t = 0$ caratterizza l'iniziale conoscenza dello stato del sistema. In assenza di essa, quest'ultimo viene tipicamente inizializzato mediante una distribuzione uniforme sull'intero spazio di stato. Nella localizzazione di robot mobili, una distribuzione uniforme corrisponde al problema della *localizzazione globale*, per il quale la posizione iniziale del robot non è nota.

Per ottenere una formulazione matematica che ci permetta di risalire alla $Bel(x_k)$, si osserva che la (1.4) può essere riscritta utilizzando la *regola di Bayes* come

$$Bel(x_k) = \frac{p(z_k|x_k, u_{k-1}, ..., z_0)p(x_k|u_{k-1}, ..., z_0)}{p(z_k|u_{k-1}, ..., z_0)}. \tag{1.5}$$

Dal momento che il denominatore della (1.5) è una costante rispetto alla variabile x_k, la regola di Bayes viene solitamente indicata come

$$Bel(x_k) = \eta p(z_k|x_k, u_{k-1}, ..., z_0)p(x_k|u_{k-1}, ..., z_0), \tag{1.6}$$

dove con η viene indicata la costante di normalizzazione

$$\eta = p(z_k|u_{k-1}, ..., z_0)^{-1}. \tag{1.7}$$

Come già accennato sopra, i filtri bayesiani si basano sull'*ipotesi di Markov*, per la quale i dati futuri sono indipendenti da quelli passati, supponendo però la conoscenza dello stato attuale. Da un punto di vista matematico tale ipotesi implica che

$$p(z_k|x_k, u_{k-1}, ..., z_0) = p(z_k|x_k) \tag{1.8}$$

e quindi la (1.6) può essere semplificata come segue:

$$Bel(x_k) = \eta p(z_k|x_k)p(x_k|u_{k-1}, ..., z_0). \tag{1.9}$$

Espandendo ora il termine più a destra della (1.9) si ottiene:

$$Bel(x_k) = \eta p(z_k|x_k) \int p(x_k|x_{k-1}, u_{k-1}, ..., z_0)p(x_{k-1}|u_{k-1}, ..., z_0)dx_{k-1}. \tag{1.10}$$

Si utilizza nuovamente l'ipotesi di Markov per semplificare il primo termine dell'integrale, $p(x_k|x_{k-1}, u_{k-1}, ..., z_0)$:

$$p(x_k|x_{k-1}, u_{k-1}, ..., z_0) = p(x_k|x_{k-1}, u_{k-1}) \tag{1.11}$$

che consente di riscrivere la (1.10) nella seguente espressione:

$$Bel(x_k) = \eta p(z_k|x_k) \int p(x_k|x_{k-1}, u_{k-1})p(x_{k-1}|u_{k-1}, ..., z_0)dx_{k-1}. \tag{1.12}$$

Sostituendo la definizione iniziale (1.4) di *belief* nella (1.12) si ottiene l'importante *equazione ricorsiva* che caratterizza il funzionamento dei filtri bayesiani:

$$Bel(x_k) = \eta p(z_k|x_k) \int p(x_k|x_{k-1}, u_{k-1})Bel(x_{k-1})dx_{k-1}. \tag{1.13}$$

Questa è l'equazione di aggiornamento ricorsiva usata nei filtri bayesiani e, insieme alla *belief* iniziale, definisce uno stimatore ricorsivo dello stato di un sistema parzialmente osservabile. Molti algoritmi di filtraggio sono costruiti ricorsivamente per

poter assicurare un tempo noto e fisso di computazione. Per poter essere implementata, la (1.13) necessita di conoscere due distribuzioni condizionate: la prima, $p(x_k|x_{k-1}, u_{k-1})$, è ottenibile dal *modello di evoluzione* (o *motion model*) e descrive come il sistema evolve nel tempo, mentre la seconda, $p(z_k|x_k)$, è il *modello di osservazione* (o *sensor model*) e rappresenta la densità della misurazione z dato lo stato x del sistema. Entrambi i modelli appena menzionati sono tipicamente *stazionari*, cioè non dipendono direttamente dal tempo t. La stazionarietà permette di semplificare la notazione, indicando i modelli $p(x'|x, u)$ e $p(z|x)$, rispettivamente.

I filtri fin qui descritti consistono principalmente in due fasi e i modelli appena menzionati operano ognuno in una di esse. La fase di *predizione* (*prediction phase*) infatti utilizza il modello di evoluzione per predire la *pdf a priori*. Dal momento che lo stato è solitamente soggetto a disturbi non noti modellati mediante un rumore casuale, la predizione generalmente trasla e deforma la *pdf*. La fase di *aggiornamento* (*update phase*) utilizza le ultime misurazioni ricevute per modificare la predizione sulla funzione densità di probabilità. Questo è ottenuto mediante il *teorema di Bayes*, che come appena visto è un meccanismo che consente di aggiornare la conoscenza dello stato del sistema adoperando informazioni extra dai nuovi dati.

I filtri bayesiani descritti in questo paragrafo sono *filtri esatti*, che non hanno cioè bisogno di approssimazioni e che possono essere adoperati in tutte le situazioni in cui è mantenuta l'*assunzione di Markov*. Purtroppo essi sono di difficile implementazione, dal momento che la (1.13) presenta un'integrazione che non permette una semplice realizzazione nella maggior parte dei casi reali. Per questo motivo si adottano filtri che utilizzano la stessa idea dei filtri bayesiani, ma che sono approssimati e di più semplice realizzabilità.

1.4 Altri approcci al problema della localizzazione

In questa sezione sono presentati alcuni approcci al problema della localizzazione di un robot mobile presenti in letteratura. Inizialmente viene proposta una soluzione che prevede l'utilizzo del *filtro di Kalman*. Successivamente si analizza la localizzazione con *filtro di Kalman Esteso* (*EKF*) per poi passare alla *localizzazione Markov*. Per finire si riporta l'approccio basato su tecniche *Monte Carlo*, che verrà ampiamente ripreso nei capitoli successivi quando verranno descritti i *filtri particellari*. Questi metodi hanno come caratteristica particolare quella di approssimare una distribuzione di probabilità che associa ad ogni punto dello spazio la possibilità che il robot si trovi nella corrispondente configurazione, aggiornandola ogni qualvolta vengano ricevuti nuovi dati sensoriali.

Le tecniche appena menzionate sono di seguito descritte solamente da un punto di vista concettuale; per ulteriori chiarimenti e soprattutto per la derivazione matematica dei vari algoritmi, si rimanda il lettore ai riferimenti bibliografici indicati.

1.4.1 Filtro di Kalman

Lo strumento forse più noto applicato al problema della localizzazione di un robot mobile e che per primo diede risultati soddisfacenti è il *filtro di Kalman*, che prende il nome da R.E. Kalman che già nel 1960 pubblicò un articolo sul filtraggio ricorsivo di sistemi discreti lineari [20]. Il filtro di Kalman è un algoritmo matematico composto da un insieme di equazioni che implementano la (1.13) attraverso due diverse fasi: predizione (in letteratura *prediction*) e aggiornamento (*update*). Esso assume che la densità a posteriori in ogni istante di tempo sia gaussiana e quindi parametrizzabile da una media ed una covarianza. È inoltre un filtro *ottimo* nel caso gaussiano lineare, in quanto è in grado di minimizzare la covarianza d'errore stimata. Se però la

$p(x_{k-1}|z_{1:k-1})$ è gaussiana, si può dimostrare che anche $p(x_k|z_{1:k})$ continua ad essere gaussiana, introducendo inevitabilmente alcune assunzioni da mantenersi:

- i rumori agenti sul sistema e sulle misure devono essere descritti da una distribuzione gaussiana;
- il modello che descrive il sistema deve essere noto e lineare;
- il modello dei sensori deve essere una funzione lineare nota dello stato.

È stato anche provato che le tecniche di localizzazione basate sul filtro di Kalman soddisfano discreti criteri di robustezza e di accuratezza. Un'ulteriore limite però consiste nella richiesta che lo stato iniziale x_0 del sistema considerato sia riconducibile ad una variabile gaussiana, pertanto $x_0 \sim \mathcal{N}(m_0, P_0)$, con m_0 e P_0 la media e la matrice di covarianza di x_0, rispettivamente. A meno di considerare $P_0 \simeq \infty$, l'assunzione che le variabili del sistema siano gaussiane rende inapplicabile questo tipo di tecniche a problemi di localizzazione globale. I filtri di Kalman si basano su modelli sensoriali che generano misure corrotte, la cui incertezza, come detto, deve essere descritta da una distribuzione gaussiana. Questo però è spesso un dato impossibile da ottenere nella realtà. Un'interessante presentazione del filtro e del suo funzionamento è riportata in [30].

1.4.2 Filtro di Kalman Esteso

Per cercare di ovviare alle problematiche incontrate nel paragrafo precedente, si può fare riferimento al *filtro di Kalman esteso* (*EKF*), basato su approssimazioni lineari locali, che permettono di utilizzare il filtro di Kalman anche per sistemi non lineari [6, 19]. Questa tecnica di filtraggio approssima dunque la $p(x_k|z_{1:k})$ mediante una gaussiana. Il *EKF* utilizza i termini del primo ordine nell'espansione di Taylor delle

funzioni non lineari. Esiste anche un altro tipo di filtro di Kalman esteso che tiene in considerazione termini di grado superiore al primo nell'espansione di Taylor, ma è caratterizzato da una complessità computazionale piuttosto elevata.

Negli ultimi anni si è applicata anche la trasformata *unscented* all'interno della struttura dell'*EKF*. Il filtro risultante, conosciuto in letteratura come *Unscented Kalman Filter* (*UKF*), considera un insieme di punti, selezionati in maniera deterministica dall'approssimazione gaussiana della $p(x_k|z_{1:k})$, che vengono propagati attraverso la non linearità del sistema. Successivamente sono ricalcolati i parametri dell'approssimazione gaussiana. Per alcuni problemi questo filtro è stato dimostrato essere migliore del filtro di Kalman esteso, dal momento che riesce ad approssimare meglio le non linearità presenti.

Comunque, nonostante l'*EKF* riesca a superare il problema introdotto dal filtro di Kalman riguardante l'impossibilità di trattare con sistemi non lineari, esso continua ad utilizzare approssimazioni gaussiane della $p(x_t|z_{1:t})$. Se la densità reale è non-gaussiana ed è, ad esempio, una funzione bimodale, allora una gaussiana non potrà mai descriverla in maniera opportuna. Per questo in letteratura si trovano altri approcci per la risoluzione del problema della localizzazione.

1.4.3 Localizzazione Markov

La *localizzazione Markov* è uno dei primi metodi utilizzati per affrontare il problema della localizzazione globale e può essere formulato come un filtro bayesiano esatto. Affronta il problema della stima dello stato mediante dati provenienti dai sensori ed è un algoritmo probabilistico: invece di mantenere una sola ipotesi riguardo alla configurazione del robot, la localizzazione Markov determina una *distribuzione di probabilità* sullo spazio di tutte le possibili ipotesi. La rappresentazione probabilistica permette

di pesare in maniera differente tutte le probabilità e prevede un aggiornamento della *pdf* mediante due modelli probabilistici, il *modello di evoluzione* e il *modello di osservazione*. Quindi il robot per migliorare la stima della propria posizione può operare in due differenti modi:

- muovendosi e cercando di individuare punti caratteristici dell'ambiente che siano in grado di togliere ambiguità alla propria posizione;
- rimanendo fermo e percependo l'ambiente.

Nel primo caso l'aggiornamento della *pdf* è dovuto al solo controllo, senza aver ancora effettuato alcuna misura, tramite la probabilità $p(x_k|x_{k-1}, u_{k-1})$ all'istante $t = k$. Nel secondo caso, invece, viene aggiornata facendo riferimento ai dati sensoriali, mediante la probabilità condizionata $p(z_k|x_k)$.
La localizzazione Markov utilizza inoltre schemi complessi ed accurati per rappresentare l'incertezza sullo stato iniziale e sulle misure dei sensori, spingendosi oltre la più semplice assunzione che debbano avere densità gaussiana, per poter affrontare anche problemi di localizzazione globale. Tale approccio viene utilizzato soprattutto per la navigazione basata su *landmark* e in situazioni in cui lo spazio di stato è organizzato secondo la struttura topologica dell'ambiente. La localizzazione Markov prevede che il robot abbia a disposizione una mappa geometrica dell'ambiente, senza fornirgli ovviamente alcuna informazione sulla propria posizione e sul proprio orientamento all'istante iniziale; assegna perciò una distribuzione uniforme di probabilità a tutte le variabili che compongono lo stato del sistema.

È possibile identificare in letteratura un'ulteriore tipologia di localizzazione Markov, che differisce da quella appena descritta, chiamata *topologica*, per il modo di

rappresentare lo spazio di stato del robot [11].

Localizzazione Markov basata su griglia (*grid-based*) Per trattare con densità multi-modali e non-gaussiane si utilizzano approcci chiamati *grid-based*, che compiono un'integrazione numerica su una griglia di punti uniformemente spaziati [12]. Spesso per l'approssimazione della densità dello spazio di stato si adotta una funzione costante a tratti. Questi metodi sono molto efficaci, ma soffrono di un'eccessivo sovraccarico computazionale e richiedono un tempo apposito *a priori* per ben dimensionare lo spazio di stato. Inoltre anche la risoluzione e la precisione che essi vogliono raggiungere deve essere fissata in anticipo. Negli ultimi anni si è cominciato a cercare di ottenere una rappresentazione dello spazio di stato di risoluzione variabile [5], in modo tale da diminuire la complessità degli algoritmi.

Nonostante la localizzazione Markov sia uno dei metodi di localizzazione probabilistica più robusto e anche piuttosto semplice a livello concettuale, soffre di alcuni inconvenienti che lo rendono difficilmente applicabile. Esso infatti prevede una *richiesta computazionale* molto elevata ed è soggetto ad *instabilità* in ambienti dinamici, dovuta all'intenso utilizzo dell'assunzione di Markov nella derivazione matematica. Non mancano comunque in letteratura valide soluzioni in grado di risolvere queste problematiche [12, 13].

1.4.4 Localizzazione Monte Carlo

La localizzazione Monte Carlo (*MCL*) è un'algoritmo probabilistico, molto simile alla localizzazione Markov, ma leggermente meno robusto e con il vantaggio di essere poco oneroso dal punto di vista computazionale [11]. I primi metodi Monte Carlo risalgono agli anni settanta e di recente sono stati riscoperti e utilizzati in problemi statistici, di

inseguimento della posizione, di *computer vision*, etc. Attualmente la *MCL* è di sicuro una delle tecniche maggiormente usate, in quanto è in grado di risolvere problemi di localizzazione globale, ma non solo; infatti affronta con successo anche il *problema del rapimento del robot* (in letteratura *kidnapped robot problem*). Si basa sul concetto di filtro bayesiano ricorsivo che stima la *pdf a posteriori* mediante l'ausilio di dati sensoriali. La *MCL* utilizza veloci tecniche di campionamento per rappresentare la distribuzione di probabilità del robot. Quando quest'ultimo si muove o percepisce qualcosa dall'ambiente che lo circonda, si utilizza il metodo *importance sampling*, trattato nel prossimo capitolo, per approssimare la *pdf* a posteriori.
La *MCL* presenta numerosi vantaggi rispetto alle altre tecniche di localizzazione:

- è in grado di rappresentare distribuzioni multi-modali e quindi può localizzare *globalmente* un robot;
- risolve situazioni in cui compaiono non linearità nel modello del sistema e nel modello di osservazione;
- riduce di molto la complessità computazionale richiesta;
- presenta una migliore accuratezza;
- è molto semplice da implementare.

L'idea chiave della localizzazione Monte Carlo è rappresentare le ipotesi a posteriori mediante un insieme di N campioni casuali e pesati, detti anche *particelle*, il cui insieme $\mathcal{X} = \{x^i | i = 1..N\}$ costituisce un'approssimazione discreta della densità di probabilità. La particella i-esima è così descritta:

$$(\langle x^i, y^i, \theta^i \rangle, W^i) \tag{1.14}$$

dove $\langle x^i, y^i, \theta^i \rangle$ indica la configurazione del robot, e $W^i \geq 0$ il *fattore di importanza* o *peso* assegnatogli. Si assume $\sum_{n=1}^{N} W^n = 1$.

In analogia con l'approccio generale della localizzazione Markov, la *MCL* prevede due fasi:

Robot motion Quando il robot comincia a muoversi, l'algoritmo di localizzazione Monte Carlo prevede la generazione di N nuove particelle che approssimano la posizione in seguito al movimento effettuato. Ogni particella è generata in maniera casuale, attingendo all'insieme di campioni determinato all'istante precedente. Indicando con *l'* la configurazione $\langle x^i, y^i, \theta^i \rangle$ di una generica particella, la nuova posizione l viene determinata mediante il modello del sistema, $p(l|l', u)$, usando il controllo u come osservazione. Il nuovo valore del peso W^i assegnato alla particella è N^{-1}. In figura 1.2 si può osservare l'effetto della tecnica di campionamento (*sampling*) adottata dalla *MCL*. Partendo da una posizione iniziale nota e facendo eseguire al robot le azioni indicate dalla linea continua, come è mostrato, l'insieme delle particelle approssima la distribuzione di probabilità con crescente incertezza, che rappresenta la graduale perdita di informazione sulla posizione, dovuta allo slittamento e alla imprecisione nel movimento del robot.

Sensor readings Le letture sensoriali sono incluse nella ri-pesatura dell'insieme dei campioni, come avviene con la regola di Bayes nella localizzazione Markov. Sia (l, W^i) (con $i = 1, ..., N$) una generica particella, allora $W^i \longleftarrow \alpha p(z|l)$, dove z contiene le misure dei sensori e α è una costante di normalizzazione che garantisce la condizione $\sum_{n=1}^{N} W^n = 1$. L'inserimento delle letture sensoriali viene solitamente portato a termine in due fasi; nella prima la W^i è moltiplicata

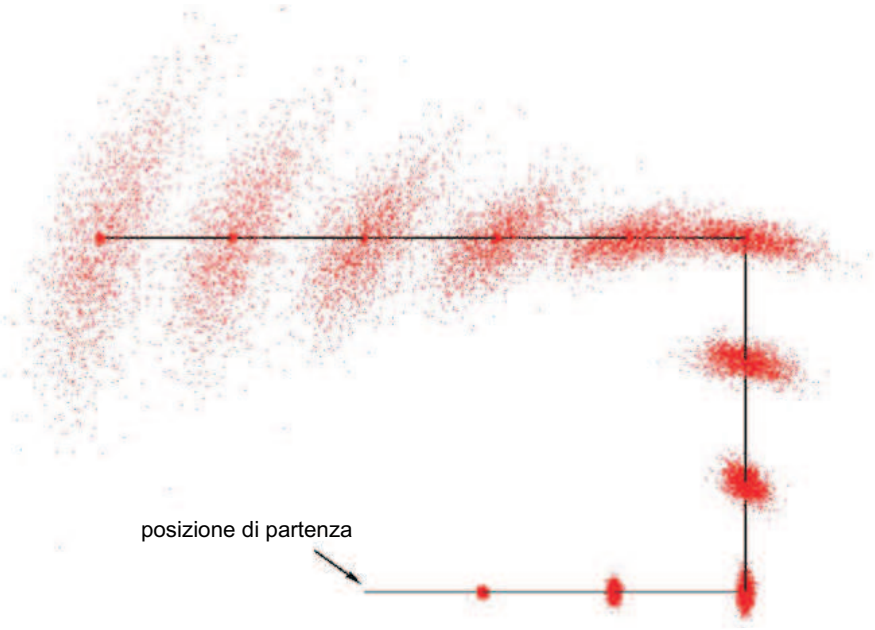


Figura 1.2: Approssimazione della *pdf* basata sul campionamento per un robot privo di misurazioni sensoriali.

per $p(z|l)$, mentre nella seconda i valori ottenuti per le probabilità di ogni singola particella vengono normalizzati.

Proprietà della localizzazione Monte Carlo. Una delle caratteristiche dell'algoritmo di localizzazione Monte Carlo è che è in grado di approssimare distribuzioni di probabilità del tutto arbitrarie. Non viene fatto alcun tipo di assunzioni sulla natura della *pdf* e questo ha favorito il largo impiego che ha avuto la *MCL* rispetto alle altre tecniche di localizzazione. La grandezza dell'insieme delle particelle naturalmente influisce sull'accuratezza e sulla complessità computazionale dell'algoritmo. Il vero vantaggio, comunque, si trova nel modo in cui la *MCL* organizza le risorse computazionali; l'attenzione è sempre focalizzata in regioni dello spazio con alta probabilità, dove realmente c'è la possibilità di trovare il robot.

La *MCL* è un metodo *online*. Esso si presta ad una semplice implementazione e gli algoritmi che lo utilizzano sono in grado di ottenere una risposta al problema della localizzazione in ogni istante; in particolare la qualità della soluzione cresce con il trascorrere del tempo. Il passo di campionamento (*sampling step*) nella *MCL* può terminare in qualsiasi istante. Quindi, non appena giungono al robot dati provenienti dai sensori di cui è dotato, oppure quando viene eseguita un'azione e da questa si ricavano nuove informazioni, la fase di campionamento viene fatta terminare e l'insieme delle particelle risultante viene utilizzato per la prossima operazione.
Il vantaggio delle tecniche di localizzazione Monte Carlo consiste dunque nel poter utilizzare delle distribuzioni arbitrarie. Nel confronto con il filtro di Kalman, c'è convergenza anche se non si è in presenza di distribuzioni gaussiane e di sistemi lineari. Rispetto, invece, ai metodi Markov basati su griglie sono molto efficienti dato che l'attenzione è focalizzata sulle particelle con più alta probabilità. Il problema di tali metodi va ricercato nel numero delle particelle e nelle piccole dimensioni che può avere lo stato del sistema.

Capitolo 2

Metodi Monte Carlo e filtri particellari

2.1 Il metodo Monte Carlo

I metodi Monte Carlo sono algoritmi computazionali che contano sul campionamento casuale ripetuto per ottenere i risultati sperati. Essi sono spesso utilizzati per simulare sistemi fisici o matematici e per la loro flessibilità e affidabilità vengono adoperati quando è molto complesso o addirittura impossibile determinare una soluzione esatta con un algoritmo deterministico.

Non esiste un unico metodo Monte Carlo; infatti questo termine racchiude un'ampia categoria di tecniche. Nonostante ciò, sono approcci che seguono uno stesso cammino, dal momento che tutti definiscono un dominio di possibili campioni, dal quale poi generano gli ingressi in maniera del tutto casuale. Utilizzano tecniche di campionamento statistico e di stima capaci di determinare la soluzione del problema proposto. In questa sezione viene analizzata la categoria del campionamento Monte Carlo (*Monte Carlo sampling*), dedicata allo sviluppo di efficienti tecniche di campionamento per risolvere problemi di stima.

Il concetto matematico alla base dei metodi Monte Carlo è molto semplice. Si

consideri un problema statistico che debba stimare il seguente integrale (*Lebesgue-Stieltjes integral*):

$$\int_X f(x)dP(x), \tag{2.1}$$

dove $f(x)$ è una funzione integrabile nello spazio che si sta considerando. Il campionamento Monte Carlo utilizza un numero di campioni casuali e indipendenti tra loro nello spazio delle probabilità (Ω, F, P) per approssimare il reale valore dell'integrale. Ricavata una collezione di N_p campioni casuali $\{x^{(1)}, ..., x^{(N_p)}\}$, indipendenti e identicamente distribuiti (*i.i.d.*), dalla distribuzione di probabilità $P(x)$, la tecnica Monte Carlo stima la $f(x)$ mediante la relazione

$$\widehat{f}_{N_p} = \frac{1}{N_p}\sum_{i=1}^{N_p} f(x^{(i)}), \tag{2.2}$$

per la quale si ha che $\mathbb{E}[\widehat{f}_{N_p}] = \mathbb{E}[f]$ e $Var[\widehat{f}_{N_p}] = \frac{1}{N_p}Var[f] = \frac{\sigma^2}{N_p}$. Si è dimostrato che $\widehat{f}_{N_p}(x)$ converge a $\mathbb{E}[f(x)]$ e l'ordine di convergenza di questo metodo, per il *teorema del limite centrale*, è pari ad $1/\sqrt{N_p}$:

$$\sqrt{N_p}(\widehat{f}_{N_p}) - \mathbb{E}[f] \sim \mathcal{N}(0, \sigma^2), \tag{2.3}$$

dove σ^2 è la varianza di $f(x)$. Una proprietà cruciale dell'approssimazione Monte Carlo è che l'accuratezza della stima è indipendente dalla dimensione dello spazio di stato e inoltre la varianza è inversamente proporzionale al numero di campioni utilizzati.

A questo punto si presentano due problematiche fondamentali:

- come estrarre i campioni $\{x^{(i)}\}$ dalla distribuzione di probabilità $P(x)$;
- come riuscire a stimare il valore atteso $\mathbb{E}[f(x)] = \int f(x)dP(x)$.

Molte altre questioni sono prese in considerazione nel campionamento Monte Carlo, tra le quali:

- la **consistenza**: uno stimatore è *consistente* se converge al valore reale *quasi certamente*, cioè se $P(\lim_{n\to\infty} X_n = X) = 1$;
- l'**errore**: uno stimatore è *privo di errore* (lett. *unbiased*) se il suo valore atteso è uguale al valore reale;
- l'**efficienza**: uno stimatore è *efficiente* se è tale che il rapporto tra la minima varianza possibile e la sua varianza effettiva è pari ad 1;
- la **robustezza**: uno stimatore si dice *robusto* se non risente di grandi errori nelle misurazioni e dell'incertezza del modello;
- la **varianza minima**: la riduzione della varianza è il compito centrale di molti metodi di approssimazione Monte Carlo.

2.1.1 Importance sampling

L'*Importance Sampling* (*IS*) è una tecnica generale per determinare le proprietà di una particolare distribuzione, avendo a disposizione solamente campioni generati da una densità differente da quella di interesse. È una tecnica di riduzione della varianza usata spesso nel metodo Monte Carlo. L'idea che sfrutta questo approccio è quella di enfatizzare tra le variabili casuali che riceve in ingresso, quelle che hanno un'influenza maggiore sui parametri che si stanno stimando, piuttosto che le altre. Questo è molto importante soprattutto per problemi di grandi dimensioni, nei quali i dati ricevuti sono spesso sparsi e la regione obiettivo da determinare è molto piccola rispetto all'intero ambiente. Il metodo adottato dall'*importance sampling* consiste nello scegliere una distribuzione $q(x)$, chiamata in letteratura *proposal distribution*, e nel ridurre la varianza dello stimatore mettendo in risalto i campioni con peso maggiore.

Riscrivendo il problema precedente come

$$\int f(x)p(x)dx = \int f(x)\frac{p(x)}{q(x)}q(x)dx, \tag{2.4}$$

l'*importance sampling* Monte Carlo si propone di utilizzare N_p campioni indipendenti ricavati dalla $q(x)$, opportunamente scelta, per ottenere una somma pesata per approssimare la (2.4):

$$\widehat{f} = \frac{1}{N_p}\sum_{i=1}^{N_p} W(x^{(i)})f(x^{(i)}), \tag{2.5}$$

dove i termini $W(x^i) = p(x^{(i)})/q(x^{(i)})$ sono detti *fattori d'importanza* o *pesi*. Per assicurare che $\sum_{i=1}^{N_p} W(x^{(i)}) = 1$, i pesi vengono normalizzati in modo da ottenere

$$\widehat{f} = \frac{\frac{1}{N_p}\sum_{i=1}^{N_p} W(x^{(i)})f(x^{(i)})}{\frac{1}{N_p}\sum_{i=1}^{N_p} W(x^{(i)})} \equiv \sum_{i=1}^{N_p} \widetilde{W}(x^{(i)})f(x^{(i)}), \tag{2.6}$$

dove $\widetilde{W}(x^{(i)}) = \frac{W(x^{(i)})}{\sum_{i=1}^{N_p} W(x^{(i)})}$ sono i *fattori d'importanza normalizzati*, o pesi normalizzati. La varianza della stima (2.5) è data da:

$$\begin{aligned}
Var_q[\widehat{f}] &= \frac{1}{N_p}Var_q[f(x)W(x)] \\
&= \frac{1}{N_p}Var_q[f(x)p(x)/q(x)] \\
&= \frac{1}{N_p}\int\left[\frac{f(x)p(x)}{q(x)} - \mathbb{E}_p[f(x)]\right]^2 q(x)dx \\
&= \frac{1}{N_p}\int\left[\frac{(f(x)p(x))^2}{q(x)} - 2p(x)f(x)\mathbb{E}_p[f(x)]\right]dx + \frac{(\mathbb{E}_p[f(x)])^2}{N_p} \\
&= \frac{1}{N_p}\int\left[\frac{(f(x)p(x))^2}{q(x)}\right]dx - \frac{(\mathbb{E}_p[f(x)])^2}{N_p}.
\end{aligned} \tag{2.7}$$

La varianza può essere ridotta quando viene scelta un'appropriata $q(x)$. La stima ottenuta mediante *importance sampling* è affetta da errore, ma è anche *consistente*. Assunta una buona scelta per q, per $N_p \to \infty$ si ha che

$$\widehat{f} \to \frac{\mathbb{E}_q[W(x)f(x)]}{\mathbb{E}_q[W(x)]}. \tag{2.8}$$

È stato anche mostrato che se $\mathbb{E}[\widetilde{W}(x)] < \infty$ e se $\mathbb{E}[\widetilde{W}(x)f^2(x)] < \infty$, la (2.6) converge al valore atteso $\mathbb{E}_p[f]$ in maniera asintotica e per il *teorema del limite centrale*

$$\sqrt{N_p}(\widehat{f} - \mathbb{E}_p[f]) \sim \mathcal{N}(0, \Sigma_f), \tag{2.9}$$

dove

$$\Sigma_f = Var_q\big[\widetilde{W}(x)(f(x) - \mathbb{E}_p[f(x)])\big]. \tag{2.10}$$

Si sottolineano ora alcuni aspetti chiave dell'*importance sampling*.

- Esso è utile in due differenti modi: (i) fornisce uno stile elegante per ridurre la varianza dello stimatore (possibilmente perfino minore della reale varianza) e (ii) può essere utilizzato quando ci si imbatte nella difficoltà di dover campionare direttamente dalla reale distribuzione di probabilità.

- Sebbene teoricamente l'errore della distribuzione *proposal* svanisca con un ordine di convergenza pari a $\mathcal{O}(N_p)$, l'accuratezza della stima non è garantita neanche con un grande numero di campioni N_p. Se $q(\cdot)$ non è vicina a $p(\cdot)$, si può immaginare che i pesi siano altamente non omogenei, cioè frastagliati, e quindi molti campioni sono piuttosto inefficaci a causa del loro contributo trascurabile.

- È possibile trovare in letteratura molte tecniche *importance sampling* (soprattutto *offline*) che presentano delle variazioni rispetto alla tecnica qui descritta, come ad esempio tecniche *importance sampling* adattative, dinamiche, o bayesiane.

2.1.2 Sequential Importance Sampling

Una buona distribuzione $q(\cdot)$ è essenziale per un'efficiente *importance sampling*. Quindi la scelta di un'appropriata distribuzione *proposal* è la chiave per poter applicare con successo tale tecnica. Purtroppo però è solitamente complesso e difficoltoso riuscire

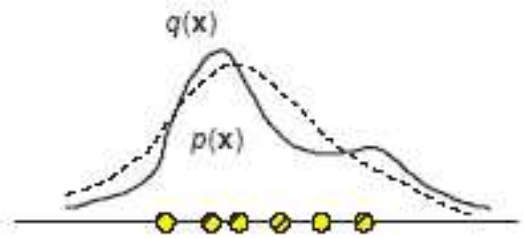


Figura 2.1: *Importance sampling.* La $p(x)$ è la vera *pdf* (linea continua), mentre $q(x)$ è la distribuzione *proposal* (linea tratteggiata).

a trovare una distribuzione che faccia al caso del problema in esame, specialmente in problemi di grandi dimensioni. Un modo naturale di risolvere questa problematica è di costruire la *proposal distribution* sequenzialmente; idea questa che porta al *Sequential Importance Sampling* (*SIS*).

In particolare, se la distribuzione $q(\cdot)$ viene scelta nella forma fattorizzata

$$q(x_{0:k}|z_{1:k}) = q(x_0)\prod_{t=1}^{k} q(x_t|x_{0:k-1}, z_{1:k}), \tag{2.11}$$

allora l'*importance sampling* può essere riformulato in maniera ricorsiva. La derivazione matematica del metodo *SIS* viene proposta più avanti, a proposito dei filtri particellari. Per ora vengono messi in luce alcuni punti chiave.

- Il metodo *SIS* può essere utilizzato anche in calcoli non-bayesiani.
- Lo svantaggio principale consiste nel fatto che i pesi potrebbero avere una grande varianza, il che si ripercuote in una stima inaccurata.
- I pesi di ogni campione tendono a diminuire all'aumentare del tempo, dando vita al *problema di degenerazione dei pesi.* Spesso dopo alcune iterazioni dell'algoritmo, solo pochi $W(x^{(i)})$ saranno diversi da zero. Per cercare di rimediare a

questo problema, si utilizza, in seguito alla normalizzazione dei pesi, la fase di *ricampionamento* (*resampling step*).

2.1.3 Sequential Importance Resampling

La fase di *resampling* si occupa dell'eliminazione dei campioni con fattori di importanza piuttosto bassi e della propagazione invece di quelli con pesi elevati. L'idea principale sulla quale si basa il *Sequential Importance Resampling* (*SIR*) può essere descritta come segue:

- vengono descritti N_p campioni casuali $\{x^{(i)}\}_{i=1}^{N_p}$ dalla distribuzione $q(x)$;
- per ogni campione $x^{(i)}$ si calcolano i pesi $W^{(i)} \propto p(x)/q(x)$;
- si normalizzano i pesi determinati al passo precedente per ottenere $\widetilde{W}^{(i)}$;
- ricampionamento per sostituzione dall'insieme discreto $\{x^{(i)}\}_{i=1}^{N_p}$.

La fase di *resampling* solitamente viene posizionata tra due passi di *importance sampling*, ma non necessariamente è così. In questa fase, i campioni e i relativi pesi associati $\{x^i, \widetilde{W}^i\}$ sono sostituiti da nuovi campioni, tutti aventi lo stesso peso, posto pari a $\widetilde{W}^i = 1/N_p$. Può essere effettuata ad ogni passo, oppure solamente quando è ritenuto necessario. Il *resampling* gioca un ruolo fondamentale nel metodo *importance sampling*, dal momento che se i pesi sono distribuiti in maniera non omogenea, continuare a propagare quelli di poco conto è deleterio, soprattutto da un punto di vista computazionale; pertanto il *resampling* fornisce una possibilità ai campioni "importanti" selezionandoli, e rigenera la distribuzione *proposal*, anche se non necessariamente esso migliora la stima dello stato corrente.

L'utilizzo del *resampling* può essere *deterministico* o *dinamico*. La struttura deterministica prevede che il *resampling* venga effettuato ogni k istanti di tempo, con k

fissato a priori. Nella programmazione dinamica, invece, viene stabilita una sequenza di soglie, che possono essere costanti o tempo varianti, e la varianza dei fattori di importanza è costantemente monitorata; il *resampling* viene effettuato solamente quando la varianza supera una certa soglia.
Ci sono molti metodi di resampling disponibili in letteratura e in [18, 24] è possibile trovare un confronto tra i più importanti, che sono qui brevemente descritti. Tutti hanno in comune di ricevere in ingresso un array contenente i pesi normalizzati delle particelle e di fornirne uno in uscita con gli indici indicanti quali particelle devono continuare ad essere utilizzate e quali no.

Select with replacement: questo semplice metodo prevede di selezionare una particella assegnandole una probabilità uguale al suo peso. Per poter applicare quest'idea in maniera efficiente, per prima cosa si calcola la somma cumulativa dei pesi delle particelle e subito dopo vengono scelti N_p numeri ordinati e casuali uniformemente distribuiti tra $[0, 1]$. Per somma cumulativa di un vettore $w \in \mathcal{N}^n$ si intende il seguente vettore:

$$\overline{w} = [w_1, w_1 + w_2, \ldots, w_1 + w_2 + \ldots + w_n]. \tag{2.12}$$

Ogni numero casuale tra quelli appena selezionati che appare in un intervallo della somma cumulativa, rappresenta una copia della particella corrispondente che "sopravvive" al passo successivo. Quanto accade può essere così riassunto: se una particella ha un peso associato piccolo, allora l'equivalente intervallo della somma cumulativa risulta piccolo e poche saranno le possibilità di vedere al suo interno una grande quantità dei numeri scelti al passo precedente in maniera casuale. Ben altre probabilità ci saranno invece di vederli apparire in un intervallo corrispondente ad una particella di peso elevato.

Multinomial resampling: questo è probabilmente il più semplice approccio al ricampionamento. Si generano N_p numeri casuali uniformemente distribuiti $u \sim \mathcal{U}(0,1)$ e i passi che compongono l'algoritmo possono essere così riassunti:

- si determina una *funzione di distribuzione cumulativa* (*Cumulative Distribution Function, CDF*) e si calcola $s_i = \sum_{j=1}^{i} \widetilde{W}^{(j)}$;
- si determina la particella s_i tale che $s_{i-1} \leq u < s_i$ e il campione con indice i viene scelto;
- dati $\{x^{(i)}, \widetilde{W}^{(i)}\}$, per $j = 1, ..., N_p$, vengono generati nuovi campioni $x^{(j)}$ dalla duplicazione di $x^{(i)}$, secondo i pesi associati $\widetilde{W}^{(i)}$;
- a tutti i pesi viene assegnato il valore $W^{(i)} = 1/N_p$.

Residual resampling: invece di utilizzare i pesi per decidere quale particella deve essere propagata all'istante di tempo successivo, può essere utilizzato un altro elemento, ovvero $a^j = f(w^j)$. Generalmente una scelta potrebbe essere quella di utilizzare la radice quadrata , quindi $f(w^j) = \sqrt{w^j}$. A questo punto i pesi vengono normalizzati in modo tale che $\sum_{i=1}^{N_p} a^i = N_p$. Ogni particella viene analizzata separatamente e se il suo peso a^j è maggiore o uguale ad uno, k sue copie sopravvivono al passo successivo ($k = \lfloor a^j \rfloor$); altrimenti, le particelle vengono singolarmente propagate in avanti, con un peso pari a a^j. Lo svantaggio principale di questa tecnica di ricampionamento è dovuto al fatto che il numero di particelle non è sempre lo stesso, e può anche aumentare ad ogni iterazione.

Systematic resampling (o ricampionamento a *varianza minima*): questo metodo tratta i pesi come variabili casuali continue nell'intervallo $[0, 1]$. Ogni particella viene replicata e il nuovo insieme di campioni è scelto in modo da minimizzare

$Var[N_i] = \mathbb{E}[(N_i - \mathbb{E}[N_i])^2]$, dove N_i è il numero di repliche della particella i-esima. La complessità di questa procedura è $\mathcal{O}(N_p)$.

Sebbene la fase di *resampling* riesce a limitare il problema della degenerazione

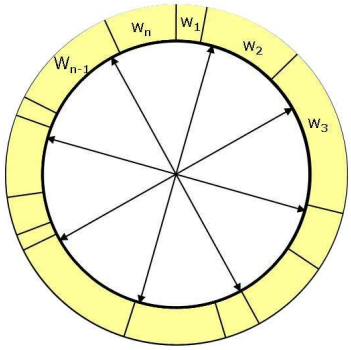


Figura 2.2: Systematic resampling

dei pesi, essa sfortunatamente introduce una nuova problematica. Infatti i campioni che hanno un elevato peso sono statisticamente selezionati molte volte, provocando inevitabilmente che l'algoritmo soffra di perdita di diversità (*loss of diversity*).

Il *multinomial* e il *systematic resampling* sono molto simili da un punto di vista della complessità computazionale. Essi differiscono solamente su come generare la sequenza di numeri casuale. Tutti gli algoritmi sono $\mathcal{O}(N_p)$. Per paragonare i metodi è dunque sufficiente andare ad analizzare le singole operazioni coinvolte negli algoritmi. È per questo che il *multinomial resampling* è più costoso, seguito dal *systematic resampling*. Il *residual resampling* invece è più complicato da classificare, dal momento che è stato provato che circa la metà delle particelle sono determinate in maniera deterministica. Questa riduzione nella complessità computazionale è però cancellata dal ricalcolo dei pesi e da altre piccole operazioni. Per questo solamente mediante le simulazioni è possibile dire se conviene oppure no utilizzare questo algoritmo. Comunque il *systematic resampling* è sicuramente migliore rispetto all'algoritmo *multinomial*, sia

da un punto di vista computazionale che di qualità del ricampionamento.

2.2 I filtri particellari

Un modo alternativo ed efficiente per rappresentare la distribuzione di probabilità è il filtro particellare, alla cui base si trova il concetto di *importance sampling*. In pratica un filtro particellare è un filtro bayesiano in cui la distribuzione di probabilità viene calcolata in maniera ricorsiva tramite simulazioni Monte Carlo [4]. In letteratura approcci di questo tipo sono conosciuti col nome di *bootstrap filtering*, *condensation algorithm*, *interacting particle approximation* e proprio il *particle filtering*. L'idea di fondo è quella di rappresentare la funzione densità di probabilità a posteriori attraverso una serie di campioni (o *particelle*) a cui è associato un peso e calcolare la stima attraverso i campioni e i pesi stessi. Ovviamente questo modello rappresenta un'approssimazione della densità effettiva; tuttavia, incrementando il numero di campioni atti alla rappresentazione, la tecnica Monte Carlo tende alla densità reale e il filtro particellare al filtro bayesiano ottimo.
Ogni particella i è descritta mediante una coppia di valori (x^i, W^i); il primo è un vettore contenente lo stato della particella, mentre il secondo è un indice utilizzato per indicare l'importanza che gli viene assegnata. La funzione densità di probabilità a posteriori, $p(x_{0:k}|z_{1:k})$, viene rappresentata mediante la distribuzione delle particelle e dei pesi corrispondenti. Questi ultimi vengono scelti in base al principio dell'*importance sampling*, affrontato nel precedente paragrafo.

A questo punto è possibile ottenere un primo algoritmo che permette di eseguire una stima approssimata in maniera incrementale della funzione *pdf*: il *Sequential Importance Sampling* (SIS).
L'algoritmo *SIS* è un metodo che pone le basi per la maggior parte dei filtri Monte

Carlo sequenziali sviluppati negli ultimi decenni. L'approccio Monte Carlo sequenziale è una tecnica sviluppata per implementare un filtro bayesiano ricorsivo mediante simulazioni Monte Carlo.

2.2.1 Filtro particellare SIS

Acquisita conoscenza di quelle che sono le basi della statistica bayesiana e delle tecniche Monte Carlo, è ora possibile analizzare i concetti che portano alla teoria dei filtri particellari.

Per evitare di cadere in situazioni di integrali intrattabili, come spesso accade nella teoria bayesiana, la distribuzione di probabilità a posteriori viene rappresentata in maniera empirica mediante una somma pesata di N_p *particelle*, ricavate dalla distribuzione

$$p(x_k|z_k) \approx \frac{1}{N_p}\sum_{k=1}^{N_p}\delta(x_k - x_k^{(i)}) \equiv \widehat{p}(x_k|z_k), \tag{2.13}$$

nella quale $x_k^{(i)}$ sono assunti essere i.i.d., ricavati da $p(x_k|z_k)$, all'istante di tempo $t = k$. Quando N_p è sufficientemente grande, $\widehat{p}(x_k|z_k)$ approssima la reale distribuzione a posteriori $p(x_k|z_k)$. Mediante questa approssimazione, è possibile stimare la media della funzione non lineare

$$\begin{aligned} E[f(x_k)] &\approx \int f(x_k)\widehat{p}(x_k|z_k)dx_k \\ &= \frac{1}{N_p}\sum_{i=1}^{N_p}\int f(x_k)\delta(x_k - x_k^{(i)})dx_k \\ &= \frac{1}{N_p}\sum_{i=1}^{N_p} f(x_k^{(i)}) \equiv \widehat{f}_{N_p}(x). \end{aligned} \tag{2.14}$$

Dal momento che è solitamente impossibile campionare dalla reale *pdf*, è molto più comune campionare da una distribuzione di più semplice implementazione, come la cosiddetta *proposal distribution* $q(x_k|z_k)$, già citata nel paragrafo precedente. Quindi

la (2.14) può essere riscritta, facendo uso della *regola di Bayes*, come

$$\begin{aligned}\mathbb{E}[f(x_k)] &= \int f(x_k)\frac{p(x_k|z_k)}{q(x_k|z_k)}q(x_k|z_k)dx_k \\ &= \int f(x_k)\frac{W_k(x_k)}{p(z_k)}q(x_k|z_k)dx_k \\ &= \frac{1}{p(z_k)}\int f(x_k)W_k(x_k)q(x_k|z_k)dx_k, \end{aligned} \tag{2.15}$$

dove

$$W_k(x_k) = \frac{p(z_k|x_k)p(x_k)}{q(x_k|z_k)}. \tag{2.16}$$

L'equazione (2.15) può anche essere riformulata nel seguente modo

$$\begin{aligned}\mathbb{E}[f(x_k)] &= \frac{\int f(x_k)W_k(x_k)q(x_k|z_k)dx_k}{\int p(z_k|x_k)p(x_k)dx_k} \\ &= \frac{\int f(x_k)W_k(x_k)q(x_k|z_k)dx_k}{\int W_k(x_k)q(x_k|z_k)dx_k} \\ &= \frac{\mathbb{E}_{q(x_k|z_k)}[W_k(x_k)f(x_k)]}{\mathbb{E}_{q(x_k|z_k)}[W_k(x_k)]}. \end{aligned} \tag{2.17}$$

Se si ottengono i campioni $\{x_k^{(i)}\}$ dalla distribuzione $q(x_k|z_k)$, è possibile approssimare la (2.17) mediante

$$\begin{aligned}\mathbb{E}[f(x_k)] &\approx \frac{\frac{1}{N_p}\sum_{i=1}^{N_p} W_k(x_k^{(i)})f(x_k^{(i)})}{\frac{1}{N_p}\sum_{i=1}^{N_p} W_k(x_k^{(i)})} \\ &= \sum_{i=1}^{N_p} \widetilde{W}_k(x_k^{(i)})f(x_k^{(i)}) \equiv \widehat{f}(x), \end{aligned} \tag{2.18}$$

dove

$$\widetilde{W}_k(x_k^{(i)}) = \frac{W_k(x_k^{(i)})}{\sum_{j=1}^{N_p} W_k(x_k^{(j)})}. \tag{2.19}$$

Si supponga ora che la distribuzione *proposal* abbia la stessa forma fattorizzata vista in (2.11) e qui di seguito riportata

$$\begin{aligned}q(x_{0:k}|z_{1:k}) &= q(x_k|x_{0:k-1}, z_{1:k})q(x_{0:k-1}|z_{1:k-1}) \\ &= q(x_0)\prod_{t=1}^{k} q(x_t|x_{0:t-1}, z_{1:t}). \end{aligned} \tag{2.20}$$

La distribuzione $p(x_{0:k}|z_{1:k})$ può essere fattorizzata come

$$\begin{aligned} p(x_{0:k}|z_{1:k}) &= \frac{p(z_k|x_{0:k}, z_{1:k-1})p(x_{0:k}|z_{1:k-1})}{p(z_k|z_{1:k-1})} \\ &= \frac{p(z_k|x_{0:k}, z_{1:k-1})p(x_k|x_{0:k-1}, z_{1:k-1})}{p(z_k|z_{1:k-1})} \times p(x_{0:k-1}|z_{1:k-1}) \\ &= \frac{p(z_k|x_k)p(x_k|x_{k-1})}{p(z_k|z_{1:k-1})} p(x_{0:k-1}|z_{1:k-1}) \\ &\propto p(z_k|x_k)p(x_k|x_{k-1})p(x_{0:k-1}|z_{1:k-1}). \end{aligned} \tag{2.21}$$

È ora quindi possibile ricavare l'equazione *ricorsiva* di aggiornamento dei pesi $W_k^{(i)}$, sostituendo la (2.20) e la (2.21) nella definizione (2.5) di fattori d'importanza:

$$\begin{aligned} W_k^{(i)} &= \frac{p(x_{0:k}^{(i)}|z_{1:k})}{q(x_{0:k}^{(i)}|z_{1:k})} \\ &\propto \frac{p(z_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})p(x_{0:k-1}^{(i)}|z_{1:k-1})}{q(x_k^{(i)}|x_{0:k-1}^{(i)}, z_{1:k})q(x_{0:k-1}^{(i)}|z_{1:k-1})} \\ &= W_{k-1}^{(i)} \frac{p(z_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{0:k-1}^{(i)}, z_{1:k})}. \end{aligned} \tag{2.22}$$

Inoltre, se $q(x_k|x_{0:k-1}, z_{1:k}) = q(x_k|x_{k-1}, z_k)$, allora la distribuzione *proposal* diventa dipendente solamente da x_{k-1} e z_k. Questo fatto è particolarmente utile nel caso in cui sia richiesta solamente la stima di $p(x_n|z_{0:n})$ in ogni istante di tempo. Se si considera questo caso, allora solo x_k^i necessita di essere immagazzinato e si può scartare l'intero percorso $x_{0:k-1}^i$ e tutte le osservazioni passate $z_{1:k-1}$. A questo punto dunque l'equazione di aggiornamento dei pesi (2.22) si semplifica:

$$W_k^{(i)} \propto W_{k-1}^{(i)} \frac{p(z_k|x_k^{(i)})p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, z_k)}. \tag{2.23}$$

A questo punto si ricava l'approssimazione della distribuzione a posteriori $p(x_k|z_{1:k})$:

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_p} W_k^i \delta(x_k - x_k^i) \tag{2.24}$$

dove i pesi sono definiti in (2.23). È stato provato che se $N_p \to \infty$, l'approssimazione (2.24) tende alla reale distribuzione di probabilità $p(x_k|z_{1:k})$.

L'algoritmo SIS quindi consiste nel propagare in maniera ricorsiva i pesi e le particelle, ogni volta che una nuova misurazione è disponibile. Una schematizzazione del suo funzionamento è riportata in Figura (2.3).

Il problema maggiore che si riscontra utilizzando l'algoritmo SIS consiste nel fenomeno della degenerazione, per il quale, dopo alcune iterazioni, la maggior parte delle particelle avrà pesi prossimi a zero. Questo è dovuto al fatto che la varianza dei pesi può solo aumentare col passare del tempo e ciò implica che grandi risorse computazionali sono spese per aggiornare campioni che non influenzeranno la stima finale della distribuzione. In letteratura questo fenomeno è noto col nome di *weight degeneracy* (degenerazione dei pesi) o *sample impoverishment* (impoverimento delle particelle). Una soluzione intuitiva è quella di duplicare le particelle con peso normalizzato elevato, scartando invece quelle con pesi trascurabili. Questa fase viene svolta dalle tecniche di *resampling*. Per avere a disposizione una misura che indichi il grado della degenerazione dei pesi, c'è bisogno di introdurre una grandezza ulteriore [22]. Un valore della degenerazione viene fornito dalla misura chiamata *numero effettivo delle particelle* (*Effective Sample Size*), indicata con N_{eff} e introdotta in [2]:

$$\begin{aligned} N_{eff} &= \frac{N_p}{1 + Var_{q(\cdot|z_{0:n})}[\widetilde{W}(x_{0:n})]} \\ &= \frac{N_p}{\mathbb{E}_{q(\cdot|z_{0:n})}[(\widetilde{W}(x_{0:n}))^2]} \leq N_p. \end{aligned} \tag{2.25}$$

La (2.25) deriva dalla considerazione che $Var[\xi] = \mathbb{E}[\xi^2] - (\mathbb{E}[\xi])^2$ e $\mathbb{E}_q[\widetilde{W}] = 1$. In realtà il vero valore di N_{eff} non è calcolabile, perciò si determina una sua approssimazione, $\widehat{N}_{eff}$, data da

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p} (\widetilde{W}_k^{(i)})^2}. \tag{2.26}$$

Tale valore è una misura approssimata di come l'insieme di particelle approssima bene la *pdf* desiderata, ed è fortemente legato alla varianza della distribuzione dei pesi, in

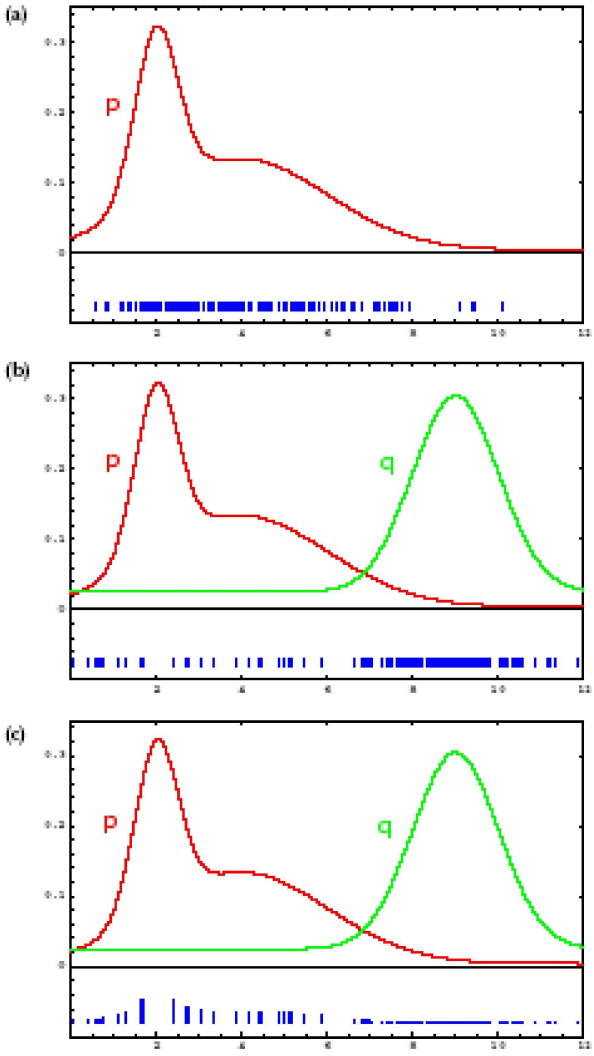


Figura 2.3: Funzionamento di un filtro particellare SIS. (a) Si cerca di approssimare la distribuzione obiettivo p. (b) Invece che campionare direttamente da p, le particelle vengono generate da un'arbitraria densità q. (c) Un'approssimazione di p è ottenuta assegnando i pesi $p(x)/q(x)$ ad ogni particella x.

modo tale che un piccolo valore di $\widehat{N}_{eff}$ indica un'elevata degenerazione. Quando $\widehat{N}_{eff}$ scende al di sotto di una soglia N_T predefinita, solitamente pari a $N_p/2$ oppure $N_p/3$, si avvia la procedura di ricampionamento (*resampling step*).

2.2.2 Filtro particellare SIR

L'algoritmo *Sampling Importance Resampling* (SIR) è l'implementazione di un filtro particellare in cui avviene una fase di ricampionamento per cercare di ridurre il fenomeno della degenerazione. Per quanto già detto a proposito del metodo Monte Carlo, la fase di ricampionamento non riesce a risolvere del tutto tale problematica, in quanto essa riesce solamente a ridurre i tempi di calcolo scartando le particelle associate a pesi di poco conto. Ciò che effettivamente fa è eliminare le particelle con peso piccolo e concentrarsi sulle particelle aventi pesi più grandi.

Il ricampionamento implica la generazione di un nuovo insieme di particelle $\{x_k^i\}_{i=1}^{N_p}$ da una rappresentazione approssimata di $p(x_k|z_{1:k})$ data dalla (2.24):

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_p} W_k^i \delta(x_k - x_k^i). \tag{2.27}$$

I campioni derivanti dalla densità discreta (2.27) sono effettivamente indipendenti e identicamente distribuiti. Pertanto ora i pesi delle particelle vengono posti uguali a $1/N_p$. È possibile implementare questa procedura di ricampionamento in $\mathcal{O}(N_p)$ operazioni ed essa viene ripetuta ad ogni iterazione, utilizzando uno qualsiasi dei metodi analizzati in precedenza. Il ricampionamento *select with replacement* è quello più adottato in letteratura, dal momento che è semplice da implementare, richiede un tempo di calcolo molto basso e minimizza la varianza Monte Carlo. In Figura (2.4) è mostrato il funzionamento di questa tecnica che dovrebbe chiarire quanto detto nel paragrafo precedente. Uno pseudo codice viene descritto in Tabella 3.2, nel prossimo capitolo.

Ora si riportano i passaggi chiave che caratterizzano un algoritmo SIR:

- *sampling*: la nuova generazione di campioni è ricavata dalla precedente, campionandola dalla $q(x_t|x_{t-1})$;

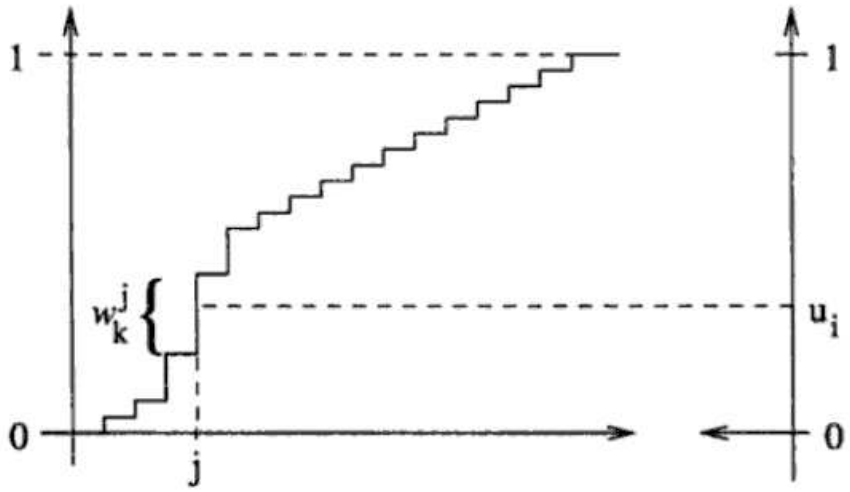


Figura 2.4: Il processo di ricampionamento: $u_i \sim \mathcal{U}[0,1]$; la particella x_k^j ha buone probabilità di essere selezionata e duplicata a causa del suo alto peso w_k^j.

- *importance*: viene assegnato un peso ad ogni particella mediante l'equazione ricorsiva di aggiornamento dei pesi;
- *resampling*: le particelle a cui viene assegnato un peso basso vengono sostituite con particelle aventi pesi più alti. Ciò è necessario per approssimare una distribuzione continua.

Sia i filtri SIS che i filtri SIR basano il loro funzionamento sull'*importance sampling*. La differenza tra i due consiste nella fase di *resampling*, che viene sempre eseguita nei filtri SIR, solitamente alla fine dell'*importance sampling*, mentre nei filtri SIS viene eseguita solo quando necessario, dal momento che i pesi sono calcolati in maniera sequenziale. Pertanto questi ultimi sono meno onerosi da un punto di vista computazionale. Sebbene la fase di resampling è in grado di ridurre gli effetti del problema di degenerazione, introduce altre problematiche non del tutto irrilevanti. A tal proposito la più evidente è quella causata dalle particelle che hanno grandi pesi asso-

ciati e che vengono statisticamente selezionate molte volte. Ciò porta ad una perdita di diversità tra le particelle. Tale problema è noto come *sample impoverishment* (impoverimento delle particelle). Capita che, soprattutto in situazioni di piccoli rumori agenti sui sistemi, tutte le particelle collassino in un singolo punto nel giro di poche iterazioni, riducendo di molto l'incertezza sulla quale poggia l'intera localizzazione probabilistica.

La fase di ricampionamento non è l'unico metodo possibile per ridurre il fenomeno della degenerazione; infatti si può ottenere una sensibile diminuzione mediante una buona scelta della distribuzione $q(\cdot)$, come analizzato nel prossimo paragrafo.

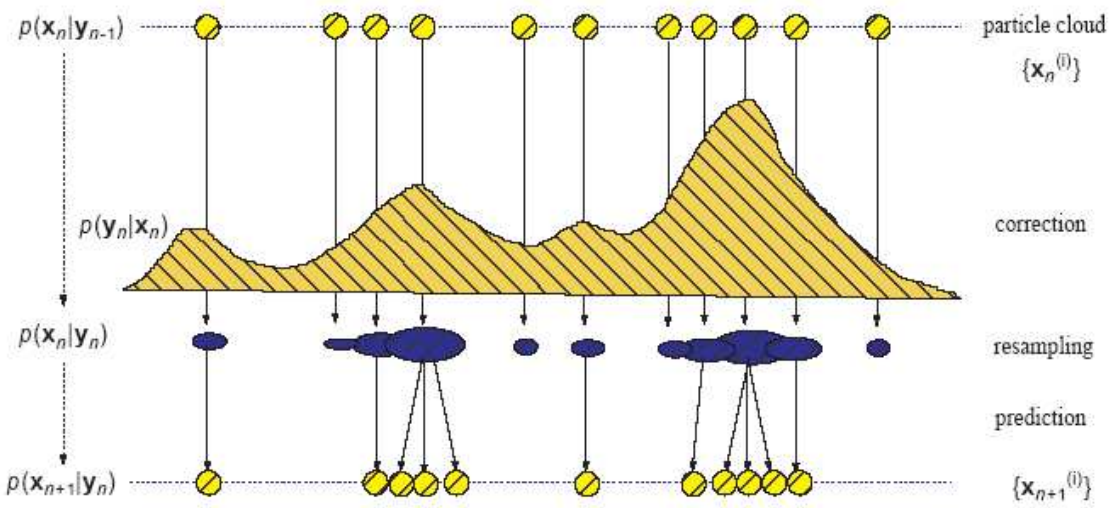


Figura 2.5: Generico filtro particellare con *importance sampling* e *resampling*.

2.2.3 Scelta della *proposal distribution*

Come più volte accennato nel corso della tesi, quando il numero di particelle N_p è sufficientemente grande, $\widehat{p}(x_k|z_k)$ approssima la reale distribuzione a posteriori $p(x_k|z_k)$, annullando quindi il problema della degenerazione. Questa però non è una strada sempre praticabile. Pertanto si è visto come sia possibile utilizzare il ricampionamen-

to, ma questa tecnica non garantisce un'accuratezza molto elevata. Si dimostra ora come una buona scelta della distribuzione *proposal* riesca a garantire ottimi risultati.

Il primo approccio prevede di scegliere la $q(x_k|x_{k-1}^i, z_k)$ in modo tale da minimizzare la varianza $Var(\widetilde{W}_k^i)$, così da massimizzare $\widehat{N}_{eff}$. Il valore ottimo della funzione densità di probabilità $q(\cdot)$, condizionata dal valore dello stato della particella *i-esima* x_{k-1}^i all'istante precedente e dalle misurazioni all'istante attuale z_k, che minimizza la varianza dei pesi $\widetilde{W}_k^i$ è dato da:

$$\begin{aligned} q(x_k|x_{k-1}^i, z_k)_{opt} &= p(x_k|x_{k-1}^i, z_k) \\ &= \frac{p(z_k|x_k, x_{k-1}^i)p(x_k|x_{k-1}^i)}{p(z_k|x_{k-1}^i)}. \end{aligned} \tag{2.28}$$

Sostituendo la (2.28) nell'espressione (2.23) si ottiene:

$$\begin{aligned} W_k^i &\propto W_{k-1}^i p(z_k|x_{k-1}^i) \\ &= W_{k-1}^i \int p(z_k|x_k^{'})p(x_k^{'}|x_{k-1}^i)dx_k^{'}. \end{aligned} \tag{2.29}$$

Questa scelta è ottima dal momento che per una data coppia di valori (x_{k-1}^i, W_k^i), a qualunque particella sia estratta dalla densità $q(x_k|x_{k-1}^i, z_k)_{opt}$ viene assegnato lo stesso peso. Quindi $Var(W_k^i) = 0$. Essa soffre però di due svantaggi:

- richiede l'abilità di campionare dalla distribuzione $p(x_k|x_{k-1}^i, z_k)$;

- si deve essere in grado di valutare l'integrale sul nuovo stato $x_k^{'}$.

Ci sono due casi in cui è possibile utilizzare la scelta ottima della $q(\cdot)$. Il primo si ha quando x_k è parte di un insieme finito di elementi. In tal caso, l'integrale (2.29) può essere riscritto come una somma, e il campionamento da $p(x_k|x_{k-1}^i, z_k)$ è possibile. Una valutazione analitica viene fatta per il secondo caso, considerando $p(x_k|x_{k-1}^i, z_k)$ una distribuzione gaussiana. È lecito fare questa ipotesi, perchè si può verificare nel

caso in cui le dinamiche sono non lineari e le misurazioni lineari. Un sistema di questo tipo è dato da:

$$x_k = f_k(x_{k-1}) + v_{k-1} \tag{2.30}$$

$$z_k = H_k x_k + n_k \tag{2.31}$$

dove

$$v_{k-1} \sim \mathcal{N}(h_{k-1}; Q_{k-1}) \tag{2.32}$$

$$n_k \sim \mathcal{N}(g_k; R_k) \tag{2.33}$$

e f_k è una funzione non lineare, H_k è la matrice di osservazione, v_{k-1} e n_k sono sequenze gaussiane mutuamente indipendenti con $Q_{k-1} > 0$ e $R_k > 0$. In questo caso si può ottenere

$$p(x_k|x_{k-1}, z_k) = \mathcal{N}(x_k; m_k, \Sigma_k) \tag{2.34}$$

e

$$p(z_k|x_{k-1}) = \mathcal{N}(z_k; H_k f_k(x_{k-1}), Q_{k-1} + H_k R_k H_k^T). \tag{2.35}$$

dove $\Sigma_k^{-1} = Q_{k-1}^{-1} + H_k^{-T} R_k^{-1} H_k$ e $m_k = \Sigma_k(Q_{k-1}^{-1} f_k(x_{k-1}) + H_k^T R_k^{-1} z_k$.
Per molti altri modelli, questo calcolo analitico non è fattibile. È comunque possibile costruire approssimazioni sub-ottime della $q(\cdot)$ mediante tecniche di linearizzazione locale, che usano distribuzioni *proposal* che sono approssimazioni gaussiane di $p(x_k|x_{k-1}, z_k)$.

Un'ulteriore scelta, spesso molto conveniente, consiste nello scegliere la distribuzione $q(\cdot)$ pari alla distribuzione di probabilità al passo precedente (detta *prior*):

$$q(x_k|x_{k-1}^i, z_k) = p(x_k|x_{k-1}^i). \tag{2.36}$$

Così facendo si ottiene la seguente equazione ricorsiva di aggiornamento dei pesi:

$$W_k^i \propto W_{k-1}^i p(z_k|x_k^i). \tag{2.37}$$

Questa è la decisione più comune nello scegliere la *proposal distribution*, dal momento che è intuitiva e di semplice implementazione. Nonostante ciò, la scelta di tale funzione rimane sempre un aspetto cruciale nella realizzazione di filtri particellari, che ne influenza molto le prestazioni.

2.2.4 Proprietà dei filtri particellari

Una volta analizzata la derivazione matematica dei filtri particellari, si può ben capire come essi siano soggetti ad errori di approssimazione. In particolare possono essere distinte diverse sorgenti d'errore ed alcuni svantaggi che l'algoritmo di localizzazione presenta.

La prima considerazione si riferisce al numero di particelle che viene utilizzato, dal momento che, per quanto possa essere grande, rimane sempre un numero finito. Questo introduce inevitabilmente un errore sistematico nella stima della distribuzione di probabilità a posteriori. Per esempio, si consideri la situazione estrema in cui $N_p = 1$. In questo caso ogni esecuzione dell'algoritmo SIS produrrà ovviamente una sola particella e pertanto la fase di ricampionamento dovrà accettare inevitabilmente questo campione, a prescindere dal suo peso W_k^i. Inoltre la probabilità $p(z_k|x_k^i)$ non gioca alcun ruolo nella fase di aggiornamento. Quindi, se si ha a disposizione un solo campione, il filtro particellare genera particelle dalla distribuzione $p(x_k|u_{1:k})$, anzichè utilizzare la *pdf* obiettivo $p(x_k|u_{1:k}, z_{1:k})$. Si noti come vengano ignorate del tutto le misurazioni. Questo fatto, se da una parte sembra molto strano, dall'altra ha un colpevole, che viene identificato nella normalizzazione, implicita nella fase di ricampionamento. In generale, questo problema consiste nel fatto che i valori W_k^i vengono estratti da uno spazio N_p-dimensionale, ma subito dopo la normalizzazione essi si ritrovano in uno spazio di dimensione $N_p - 1$. Questo accade perchè l'ultimo peso

viene determinato mediante gli $N_p - 1$ pesi precedenti, andando a sottrarre all'unità la somma di tutti i pesi. Fortunatamente, per valori più grandi del numero di particelle, l'effetto della perdita di dimensionalità diventa molto meno pronunciato.

Nei filtri particellari può accadere che la mancanza di casualità introdotta dalla fase di ricampionamento e già accennata nel paragrafo precedente provochi dei problemi nella stima dello stato del sistema. Per capire questa situazione, si fa riferimento di nuovo ad un caso estremo, cioè il caso di un robot il cui stato non cambia mai, ovvero $x_t = x_{t-1}$. Un ottimo esempio sarebbe quello di un problema di localizzazione mobile di un robot che rimane fermo nell'ambiente in cui si trova. Inoltre si assuma che il robot non possieda sensori, quindi non sia in grado di stimare la propria posizione e sia anche ignaro del suo stato. Inizialmente l'insieme delle particelle $\mathcal{X}_0$ sarà generato dalla *pdf a priori*, quindi le particelle saranno sparse per tutto lo spazio di stato. La situazione è abbastanza scoraggiante: il *resampling* porterà ad un fallimento dell'algoritmo di localizzazione e l'*importance sampling* non produrrà nuove particelle. Con probabilità uno, quindi, N_p identiche copie di un singolo stato rimarranno in vita. La diversità e la casualità che contraddistinguono il filtro particellare andranno a scomparire a causa della ripetizione della fase di ricampionamento. All'occhio di un osservatore esterno, sembrerà che il robot abbia univocamente determinato la propria posizione, mentre in realtà questo è alquanto improbabile soprattutto perchè esso non è dotato di alcun sensore che gli possa fornire informazioni utili.

L'esempio riportato mette in evidenza un limite dei filtri particellari; in particolare il processo di ricampionamento porta alla perdita di diversità nell'insieme delle particelle, fatto che già da solo manifesta un errore di approssimazione. Questo errore è chiamato *varianza* dello stimatore. Controllare la varianza del filtro particellare è molto importante per ogni implementazione pratica. Per diminuirne il valore si può

procedere con la riduzione della frequenza con la quale viene effettuato il ricampionamento (in particolare quando lo stato è statico, come nel caso estremo visto in precedenza, non si dovrebbe mai attuare il ricampionamento). La scelta di quando effettuare quest'ultimo è molto complessa. Ricampionare troppo spesso aumenta il rischio di perdere in diversità, se invece le particelle vengono ricampionate con troppa poca frequenza, può accadere che molte di esse vadano a finire in regioni di bassa probabilità. Per questo si misura la varianza dei pesi e si opera come visto a proposito dell'algoritmo SIS.

Nel paragrafo precedente si è discusso dell'importanza assunta dalla scelta della distribuzione *proposal*, che necessariamente si deve avvicinare molto alla distribuzione reale (*target*) da stimare. L'efficienza del filtro particellare dipende essenzialmente da questo fattore, ma anche una perfetta corrispondenza tra le due distribuzioni potrebbe non servire per ottenere una buona stima. Infatti, il filtro particellare soffre terribilmente problemi di localizzazione globale all'interno di ambienti simmetrici. La distribuzione uniforme con la quale viene inizializzato l'algoritmo è soggetta ad innumerevoli casi di ambiguità. Se due particelle si trovano in due zone completamente opposte dell'ambiente, ma ricevono le stesse misurazioni dai sensori, quello che accade è che l'equazione di aggiornamento dei pesi assegna all'incirca lo stesso peso ad entrambe. Non è in grado l'algoritmo di stabilire univocamente quale sia la reale posizione del robot, perchè l'ambiente nel quale esso si trova è caratterizzato da zone simmetriche che non facilitano la localizzazione. Questa problematica viene ripresa nel prossimo capitolo (vedi Figura 3.3).

Capitolo 3

Algoritmo di localizzazione basato su filtro particellare

In questo capitolo si descrive il lavoro svolto per risolvere il problema della localizzazione mediante l'utilizzo dei filtri particellari e verranno analizzate tutte le fasi che compongono l'algoritmo. Per poter fare ciò, bisogna prima di tutto introdurre il modello (*motion model*) che descrive l'evoluzione nel tempo dello stato del robot che si vuole stimare. Viene inoltre presentato il modello di osservazione (*sensor model*), utilizzato per determinare la posizione del robot mediante informazioni sensoriali che descrivono l'ambiente che lo circonda.

3.1 Modello di sistema

Si considera un robot che si muove in un ambiente bidimensionale, nel quale è possibile identificarne la posizione in modo univoco mediante due coordinate cartesiane (x_k, y_k) ed una angolare θ_k che ne indica l'orientamento. Un qualsiasi movimento arbitrario $[\Delta x, \Delta y]^T$ può essere rappresentato mediante una rotazione seguita da una traslazione (vedi Figura 3.1). Pertanto, se si indica con $[x_k, y_k, \theta_k]$ la posizione di partenza del robot in $t = k$, ad ogni istante in cui c'è un movimento da compiere, si può pensare che il robot effettui una rotazione $\delta\theta_k = \theta_{k+1} - \theta_k$ per rivolgersi verso la posizione da

raggiungere e che poi esso trasli in avanti di $\delta\rho_k = \sqrt{\Delta x^2 + \Delta y^2}$. La configurazione finale assunta è descritta dal seguente sistema di equazioni:

$$\begin{cases} x_{k+1} = x_k + \delta\rho_k cos(\theta_k) \\ y_{k+1} = y_k + \delta\rho_k sin(\theta_k) \\ \theta_{k+1} = \theta_k + \delta\theta_k \end{cases} \tag{3.1}$$

Lo stato del robot è dunque univocamente identificabile mediante la tripla $\{x, y, \theta\}$. Nel modello $\delta\rho_k$ e $\delta\theta_k$ rappresentano la traslazione e la rotazione effettuate durante lo spostamento. Il rumore viene applicato in modo separato ai due tipi di movimento dal momento che sono assunti indipendenti. Pertanto si ha che

$$\begin{aligned} \delta\rho_k &= \delta\rho_k^e + n_{\rho,k} \\ \delta\theta_k &= \delta\theta_k^e + n_{\theta,k} \end{aligned} \tag{3.2}$$

dove $\delta\rho_k^e$ e $\delta\theta_k^e$ sono misurazioni ottenute dalle letture degli encoder, mentre i rumori, la cui varianza è assunta crescere linearmente con la distanza percorsa, sono dati da

$$\begin{aligned} n_{\rho,k} &\sim \mathcal{N}(0, K_\rho \delta\rho_k^e) \\ n_{\theta,k} &\sim \mathcal{N}(0, K_\theta \delta\rho_k^e) \end{aligned} \tag{3.3}$$

dove K_ρ e K_θ sono due costanti positive.

3.2 Modello dei sensori

Il modello dei sensori, o *modello percettivo*, lega la parte osservabile del sistema al suo stato. Bisogna perciò trovare un'equazione che sia in grado di connettere le osservazioni alla configurazione del robot. Se si indica con $z_{i,k}$ il vettore contenente la coppia di valori $(\varepsilon_{i,k}, \psi_{i,k})$, rispettivamente la distanza e il verso in cui si trova l'oggetto più vicino nella direzione di osservazione del sensore i-esimo all'istante $t = k$, per trovare l'espressione analitica del modello percettivo sarebbe necessario conoscere

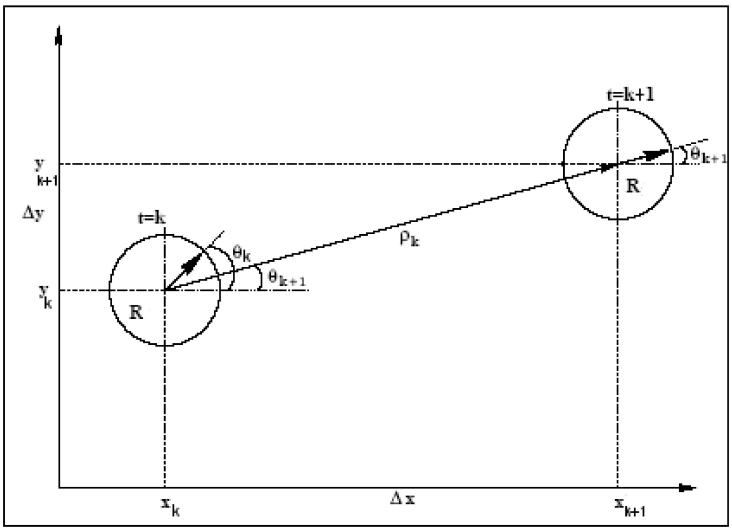


Figura 3.1: Un generico movimento $[\Delta x, \Delta y]^T$ del robot R può essere sempre scomposto come somma di una rotazione $\delta\theta_k$ e di una traslazione $\delta\rho_k$.

una funzione $h : \mathbb{R}^3 \rightarrow \mathbb{R}^2$. Tale funzione permette di stabilire dove si trova l'ostacolo più vicino e a quale distanza dal robot. Il modello dei sensori adottato è pertanto così descritto:

$$z_{i,k} = h_i[x_{r,k}, \Theta] + \nu_{i,k} \tag{3.4}$$

per $i = 1, ..., m$, con m numero di sensori predisposti sul robot, $z_{i,k}$, come detto, è il vettore contenente la i-esima misura all'istante k, Θ rappresenta l'ambiente, $\{\nu_{i,k}\}$ sono sequenze indipendenti di variabili gaussiane a media nulla e $x_{r,k} = [x_k, y_k, \theta_k]^T$. La funzione $h_i[x_{r,k}, \Theta]$ che completa il modello è stata scelta in questo lavoro uguale a

$$h_i[x_{r,k}, \Theta] = \sqrt{(x_k - x_{p_i,k})^2 + (y_k - y_{p_i,k})^2} \tag{3.5}$$

dove $(x_{p_i,k}, y_{p_i,k})$ rappresenta il punto di intersezione dell'i-esimo sensore con l'ostacolo più vicino lungo la sua direzione all'istante $t = k$ (vedi Figura 3.2).

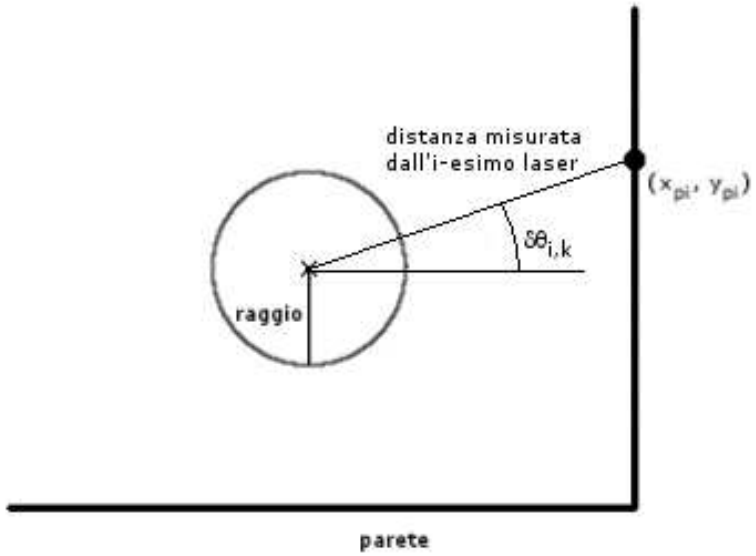


Figura 3.2: Schematizzazione del sistema robot-sensori. Un generico sensore i rileva la distanza tra se stesso e l'ostacolo più vicino; (x_{p_i}, y_{p_i}) sono le coordinate del punto di intersezione con l'ostacolo suddetto; $\theta_{i,k}$ è l'angolo che il sensore forma con l'asse x del sistema di riferimento cartesiano.

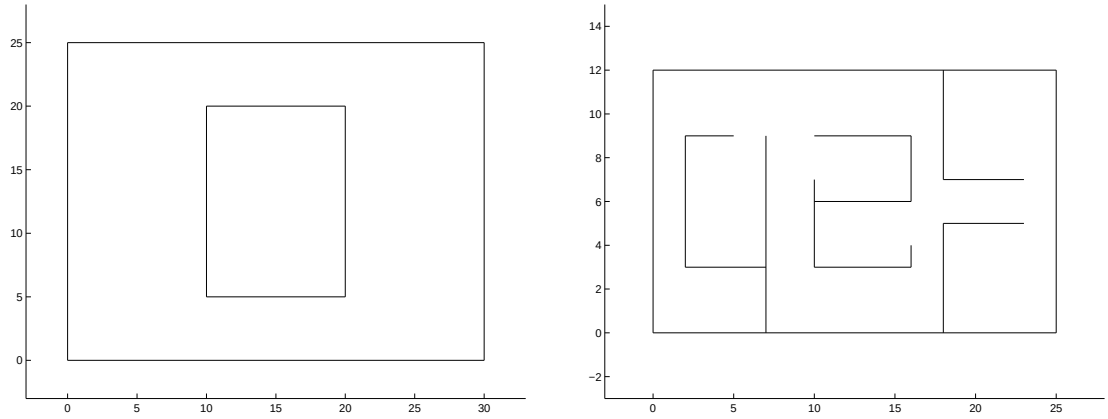
3.3 La mappa

In questo lavoro non si è prestata una particolare attenzione a come poter descrivere in maniera accurata l'ambiente all'interno del quale far muovere il robot. È per questo che si è semplicemente cercato di implementare nel modo più semplice possibile, una mappa abbastanza dettagliata che abbia un discreto numero di ostacoli e un'asimmetria piuttosto elevata, dal momento che in ambienti simmetrici i filtri particellari non riescono ad identificare univocamente la posizione del robot. In Figura (3.3) vengono riportati due differenti tipi di ambienti impiegati in fase di simulazione. Per descrivere la mappa si è utilizzata una matrice, le cui righe descrivono il segmento i-esimo, mentre le colonne le coordinate del punto iniziale e finale del segmento che raffigura

l'ostacolo. Una schematizzazione di quanto fatto viene riportata in (3.6), dove N_{obs} indica il numero complessivo di ostacoli, mentre $(x_{i,j}, y_{i,j})$ rappresentano le coordinate del punto iniziale ($j = 1$) o finale ($j = 2$) dell'ostacolo i-esimo.

$$\begin{bmatrix} x_{1,1} & y_{1,1} & x_{1,2} & y_{1,2} \\ x_{2,1} & y_{2,1} & x_{2,2} & y_{2,2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N_{obs},1} & y_{N_{obs},1} & x_{N_{obs},2} & y_{N_{obs},2} \end{bmatrix} \tag{3.6}$$

Ovviamente per descrivere un ambiente piuttosto complesso e ricco di elementi, è necessaria una matrice con un elevato numero di righe. Questo per l'algoritmo realizzato è un problema lasciato irrisolto e che potrà essere affrontato in futuro. All'aumentare del numero di segmenti sulla mappa, aumenta inevitabilmente anche l'elaborazione dei dati e il tempo di simulazione.



(a) (b)

Figura 3.3: Ambiente, realizzato in *Matlab*, all'interno del quale il robot deve essere in grado di localizzarsi. (a) Ambiente simmetrico (b) Ambiente asimmetrico

3.4 Implementazione dell'algoritmo

Il principale obiettivo dei filtri particellari è riuscire a stimare una variabile di interesse che evolve nel tempo, solitamente mediante una *pdf* multi-modale e non gaussiana. La variabile di interesse qui considerata è ovviamente la configurazione (posizione e orientamento) del robot mobile $x_{r,k} = [x_k, y_k, \theta_k]^T$. Come più volte accennato, l'idea che sta alla base di tali filtri è quella di costruire una rappresentazione dell'intera *pdf* obiettivo, mediante un insieme di particelle. Lo stato da stimare cambia continuamente, mediante una serie di comandi, secondo il modello che ne descrive il comportamento. Inoltre, ad un certo istante, giungono dati che descrivono le osservazioni fatte dal robot e limitano il possibile valore della stima ad un campo più ristretto dell'intero spazio di stato. Sono utilizzate molte copie dello stato, ognuna associata ad un peso che indica la qualità di quella specifica particella. Una stima della posizione del robot è ottenuta mediante la somma pesata di tutte le particelle. L'algoritmo basato su filtro particellare è ricorsivo e opera solitamente in due fasi principali: *predizione* e *aggiornamento*. Dopo un'azione, ogni particella viene modificata secondo il modello di sistema (*fase di predizione*), che include l'aggiunta di rumore casuale in modo tale da simularne l'effetto sulla posizione del robot. Successivamente, il peso delle particelle viene ricalcolato in base alle ultime letture sensoriali disponibili (*fase di aggiornamento*). A volte i campioni con pesi infinitamente piccoli sono eliminate attraverso la fase di ricampionamento.

Se all'istante $t = k$ è nota la *pdf* del sistema all'istante di tempo precedente $t = k-1$, allora viene modellato l'effetto dell'azione, in modo tale da ottenere *a priori* la *pdf* (predizione). In altre parole, la fase di predizione utilizza il modello di sistema per simulare l'effetto che il movimento imposto al robot ha sull'insieme delle particelle.

La fase di aggiornamento utilizza, invece, le informazioni ottenute dalle letture sensoriali per aggiornare i pesi delle particelle, così da descrivere accuratamente la *pdf* desiderata.

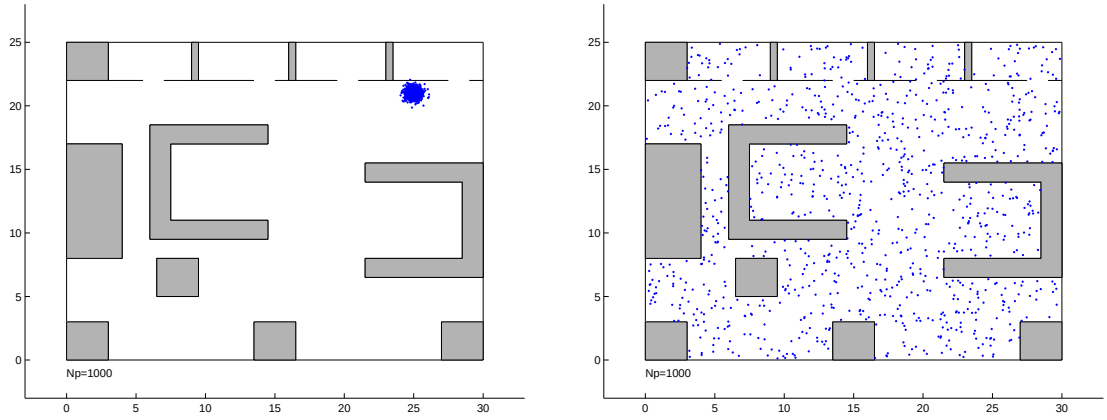
3.4.1 Inizializzazione

La prima fase dell'algoritmo di localizzazione si basa sull'ipotesi a cui fanno riferimento i filtri bayesiani, ossia la conoscenza della *pdf* all'istante di tempo $t = 0$ ($p(x_{r,0})$). È pertanto necessario conoscere la forma della distribuzione di probabilità iniziale dello stato.

Nel caso in cui si ha a disposizione la configurazione iniziale del robot, è conveniente utilizzare come $p(x_{r,0})$ una funzione gaussiana centrata nel punto in cui esso è posizionato e avente una varianza dipendente dall'errore da cui è affetta la misurazione. Anche se si suppone che non ci siano errori di misura (cosa questa assai improbabile), non conviene generare tutte le particelle iniziali nello stesso punto dello spazio di stato. Una delle condizioni che, infatti, permette un buon funzionamento dell'algoritmo, è il mantenimento della diversità.

Se la localizzazione è invece di tipo globale, la posizione del robot è totalmente sconosciuta, e in questo caso la $p(x_{r,0})$ sarà una densità di probabilità uniforme. Campionandola, le particelle ottenute si distribuiscono uniformemente sull'intero spazio di stato.

L'algoritmo realizzato è in grado di poter risolvere entrambi i problemi di localizzazione, dal momento che c'è la possibilità di scegliere se avere o meno a disposizione la conoscenza della posizione di partenza $x_{r,0} = [x_0, y_0, \theta_0]^T$. In Figura (3.4) viene evidenziata la differenza nella distribuzione delle particelle all'istante iniziale.



(a) (b)

Figura 3.4: Inizializzazione dell'algoritmo di localizzazione supponendo (a) *nota* e (b) *non nota* la posizione iniziale del robot. Nel primo caso le particelle sono distribuite secondo una funzione gaussiana centrata nell'intorno del punto in cui si trova il robot e avente una varianza dipendente dall'errore da cui è affetta la misurazione, mediante una distribuzione uniforme nel secondo.

3.4.2 Predizione

I filtri particellari, analogamente a tutti i filtri ricorsivi, hanno la proprietà di aggiornarsi con frequenza costante. Questo significa che due stati stimati consecutivamente saranno separati da un intervallo di tempo fisso. Per predire la distribuzione di probabilità della posizione del robot dopo un movimento, è necessario avere a disposizione il modello che descrive l'effetto del rumore sulla posizione risultante, ovvero il modello di sistema ricavato in precedenza. Ogni particella all'istante di tempo $t = k$ evolve secondo il sistema descritto in (3.1), che può essere particolareggiato, per ogni $i = 1, ..., N_p$, come segue

$$\begin{cases} x_{k+1}^i = x_k^i + \delta\rho_k^{e,i} cos(\theta_k^i) \\ y_{k+1}^i = y_k^i + \delta\rho_k^{e,i} sin(\theta_k^i) \\ \theta_{k+1}^i = \theta_k^i + \delta\theta_k^{e,i} \end{cases} \tag{3.7}$$

Si noti come le misurazioni $\delta\rho_k^{e,i}$ e $\delta\theta_k^{e,i}$ degli encoder siano diverse da particella a particella, dal momento che sono state implementate come $\delta\rho_k^{e,i} = \mathcal{N}(\delta\rho_k, randnK_\rho\delta\rho_k)$ e $\delta\theta_k^{e,i} = \mathcal{N}(\delta\theta_k, randnK_\theta\delta\rho_k)$. Durante la fase di predizione, l'algoritmo tenta di stimare la posizione corrente partendo dallo stato e dai controlli effettuati nell'istante di tempo discreto precedente:

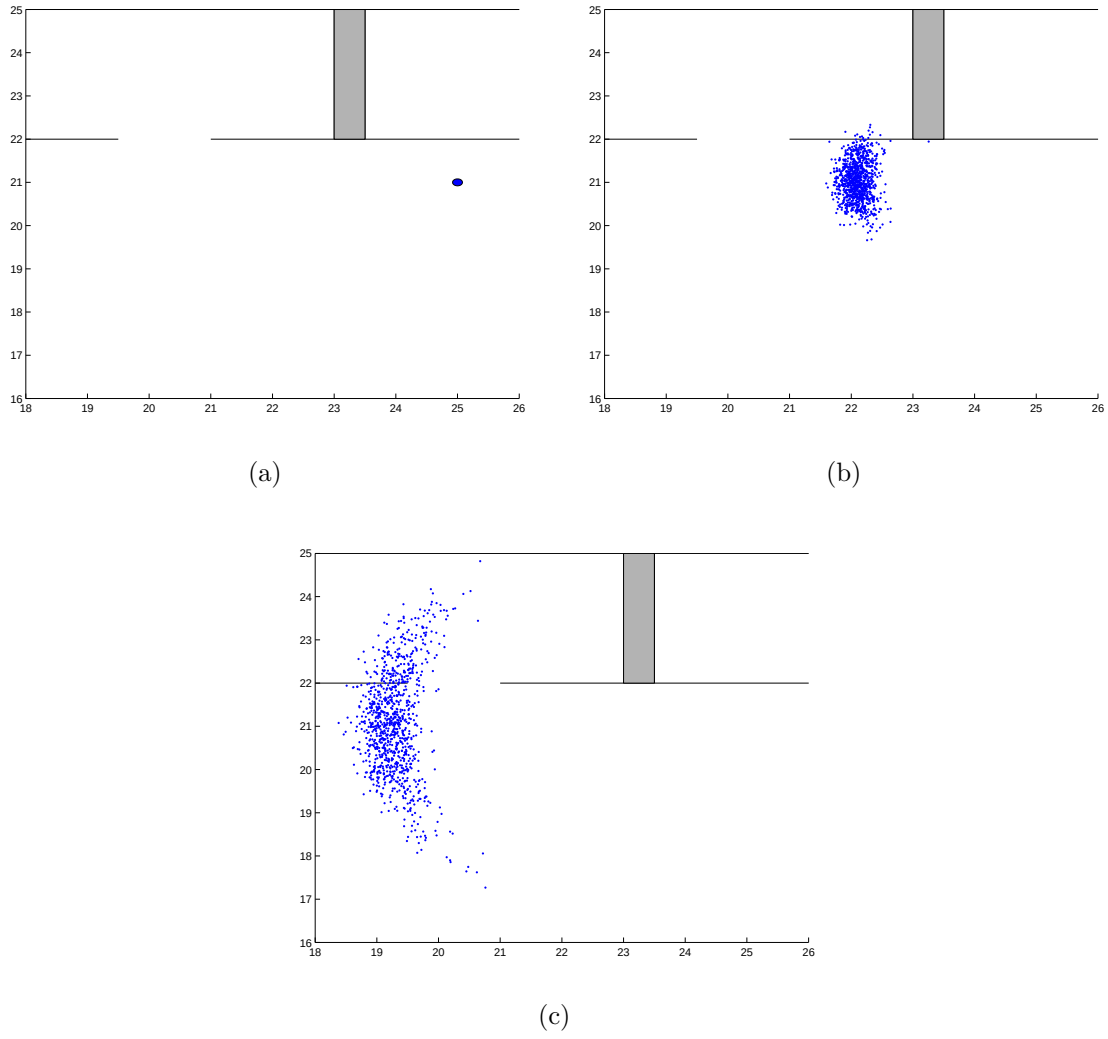
$$x_{r,k}^i \sim p(x_{r,k}|x_{r,k-1}^i) \tag{3.8}$$

dove con $\sim$ si vuole intendere *campionato da*.

Questa fase è totalmente *dead reckoning*, ovvero basata unicamente sull'odometria. È fondamentale individuare eventuali errori sistematici (ad esempio se il robot ha una velocità superiore a quella che i suoi sensori odometrici rilevano, oppure se navigando tende a orientarsi verso un lato, etc.) e non sistematici, modellizzabili attraverso il rumore gaussiano, presente nelle equazioni di moto del robot. A tal proposito si pone l'attenzione sul fatto che, per quanto riguarda la traslazione in avanti del robot, ci sono due differenti sorgenti di errore: la prima è dipendente dall'attuale distanza percorsa, mentre la seconda è relativa ai piccoli cambiamenti dell'orientamento durante la traslazione stessa che allontanano, seppur di poco, il robot dalla direzione desiderata. Quest'ultimo effetto è chiamato *drift* e può essere modellato aggiungendo una piccola quantità di rumore gaussiano sull'orientamento del robot [26].

Una caratteristica fondamentale del filtro particellare consiste nel fatto che particelle nella stessa posizione e con lo stesso orientamento all'istante $t = k$ assumono due posizioni diverse nell'istante di tempo successivo $t = k + 1$. Tale fenomeno, chiamato *multihypothesis tracking*, è dovuto alle particelle che evolvono nel tempo secondo il sistema (3.7), influenzato dal rumore gaussiano che assume un ruolo di *generatore di diversità*. Un esempio dell'effetto della fase di predizione è evidenziato in Figura (3.5).

Inoltre, si riportano in Tabella (3.1) i passaggi implementati in *Matlab* per simulare la predizione.



(a) (b)

(c)

Figura 3.5: Effetto della fase di predizione sulle particelle, che tendono ad evolversi secondo il modello di sistema, ognuna in modo indipendente.

3.4.3 Aggiornamento

La fase di aggiornamento è quella in cui il robot ricava il vettore $z_k = [z_{1,k}, z_{2,k}, ..., z_{m,k}]$, contenente le misure provenienti dai suoi m sensori. Tali dati sono delle vere e proprie

Fase di predizione
for $k = 1$ **to** N_{passi} **do** {Ad ogni passo}
for $i = 1$ **to** N_p **do** {Per ogni particella}
$\delta\rho_k^{e,i} = \delta\rho_k + \sqrt{K_\rho \delta\rho_k} randn;$
$\delta\theta_k^{e,i} = \delta\theta_k + \sqrt{K_\theta \delta\rho_k} randn;$
$\theta_k^i = \theta_{k-1}^i + \delta\theta_k^{e,i};$
$x_k^i = x_{k-1}^i + \delta\rho_k^{e,i} cos(\theta_k^i);$
$y_k^i = y_{k-1}^i + \delta\rho_k^{e,i} sin(\theta_k^i);$
$\delta\theta_k^{e,i} = \delta\theta_k + \sqrt{K_\theta \delta\rho_k} randn;$
$\theta_k^i = \theta_{k-1}^i + \delta\theta_k^{e,i};$
$S_k^i = [x_k^i, y_k^i, \theta_k^i]^T;$
end for
end for
Return(S)

Tabella 3.1: Algoritmo di predizione dello stato

osservazioni, che indicano sia la distanza esistente tra i sensori e l'ostacolo più vicino, sia l'angolo compreso tra l'asse x del sistema di riferimento cartesiano e la congiungente il robot con l'ostacolo identificato (Figura 3.2). Il peso di una particella indica quanto questa rappresenti una stima attendibile dello stato e il suo valore dipende da quello che il robot percepisce: se le osservazioni sono coerenti con la posizione del campione, il peso sarà alto, altrimenti diminuirà, seguendo il profilo di una gaussiana generata sul sottospazio osservabile. Viene pertanto valutata la probabilità che ogni campione ha di essere la reale posizione del robot e si assegna un peso ad ogni particella mediante la relazione ricavata in (2.23) e qui riportata:

$$W_k^{(i)} \propto W_{k-1}^{(i)} \frac{p(z_k|x_k^{(i)}) p(x_k^{(i)}|x_{k-1}^{(i)})}{q(x_k^{(i)}|x_{k-1}^{(i)}, z_k)} \tag{3.9}$$

I pesi vengono poi successivamente normalizzati

$$\widetilde{W}_k^i = \frac{W_k^i}{\sum_{i=1}^{N_p} W_k^i}. \tag{3.10}$$

Particolare difficoltà è stata incontrata nella determinazione dell'equazione (3.9), nella quale compaiono tre differenti distribuzioni: la $p(z_k|x_k^{(i)})$, nota come *likelihood di-*

stribution, la *prior distribution* $p(x_k^{(i)}|x_{k-1}^{(i)})$ e la *proposal distribution* $q(x_k^{(i)}|x_{k-1}^{(i)}, z_k)$. Come già ampiamente trattato in precedenza, grande importanza assume in questa fase la scelta della $q(x_k^{(i)}|x_{k-1}^{(i)}, z_k)$. Infatti, se da una parte la distribuzione *prior* può essere ricavata dal modello di sistema e la *likelihood* dal modello dei sensori, dall'altra la *proposal* viene scelta in maniera arbitraria. A tal proposito, si ricorda che si sta risolvendo il problema mediante il metodo *importance sampling*, che prevede, a causa dell'elevata complessità della distribuzione desiderata, la possibilità di determinarne una nuova, dalla quale campionare le particelle. L'importante è che tale densità sia piuttosto semplice da trattare e approssimi il più possibile la reale *pdf*. Per realizzare l'algoritmo di localizzazione si è deciso di scegliere la $q(\cdot)$ uguale alla *prior*

$$q(x_k^{(i)}|x_{k-1}^{(i)}, z_k) = p(x_k^{(i)}|x_{k-1}^{(i)}), \tag{3.11}$$

in modo tale da ridurre l'equazione di aggiornamento dei pesi a

$$W_k^i \propto W_{k-1}^i p(z_k|x_k^i). \tag{3.12}$$

A questo punto, per poter correttamente localizzare il robot, viene adottata un'adeguata strategia di aggiornamento dei pesi, come proposto in [1]. Essa prevede l'utilizzo dell'errore quadratico tra le misurazioni provenienti dai sensori predisposti sul robot e quelle stimate dalla particella i-esima:

$$d_k^i = (z_k - z_k^i)^2, \tag{3.13}$$

per $i = 1, ..., N_p$, dove z_k^i rappresenta le misure ricavate dalla particella i posizionata in (x_k^i, y_k^i).

L'equazione di aggiornamento dei pesi deriva da un ragionamento semplice ed intuitivo, che non ricalca la teoria fin qui analizzata. Se l'errore d_k^i relativo ad una generica particella è molto grande, significa che essa si trova in una posizione dell'ambiente

abbastanza distante dalla reale posizione da stimare, pertanto si può attribuire al corrispondente peso W_k^i un valore basso. Se invece le misurazioni effettive dei sensori e quelle della particella i sono simili, è lecito assegnare a quest'ultima un elevato fattore d'importanza, indice di buona qualità delle misure. Di conseguenza i pesi vengono aggiornati secondo la relazione

$$W_k^i = W_{k-1}^i \frac{\beta}{d_k^i}, \tag{3.14}$$

dove β è un fattore di scala, di valore trascurabile per la successiva normalizzazione (3.15).

La fase di aggiornamento si conclude con la normalizzazione dei pesi, eseguita per soddisfare il vincolo:

$$\sum_{i=1}^{N_p} W_k^i = 1. \tag{3.15}$$

A questo punto può essere ottenuta l'approssimazione (2.27) della funzione densità di probabilità desiderata:

$$p(x_k|z_k) \approx \sum_{i=1}^{N_p} W_k^i \delta(x_k - x_k^i). \tag{3.16}$$

In Figura 3.6 viene mostrata la fase di aggiornamento delle misure mediante le osservazioni effettuate dai sensori. Nella figura successiva sono evidenziate le informazioni che ad ogni istante il robot ha a disposizione per potersi localizzare al meglio.

3.4.4 Ricampionamento

Come già sottolineato in precedenza, l'idea chiave del *resampling* è quella di eliminare le particelle più leggere rimpiazzandole con altre. Queste ultime dovranno essere generate dai campioni più pesanti e, quindi, con probabilità di successo maggiore. Si moltiplicano/eliminano le particelle $x_{r,k}^i$ con alto/basso fattore di importanza w_k^i, rispettivamente. Per ottenere N_p particelle casuali, nuovi campioni $x_{r,k}^i$ vengono di-

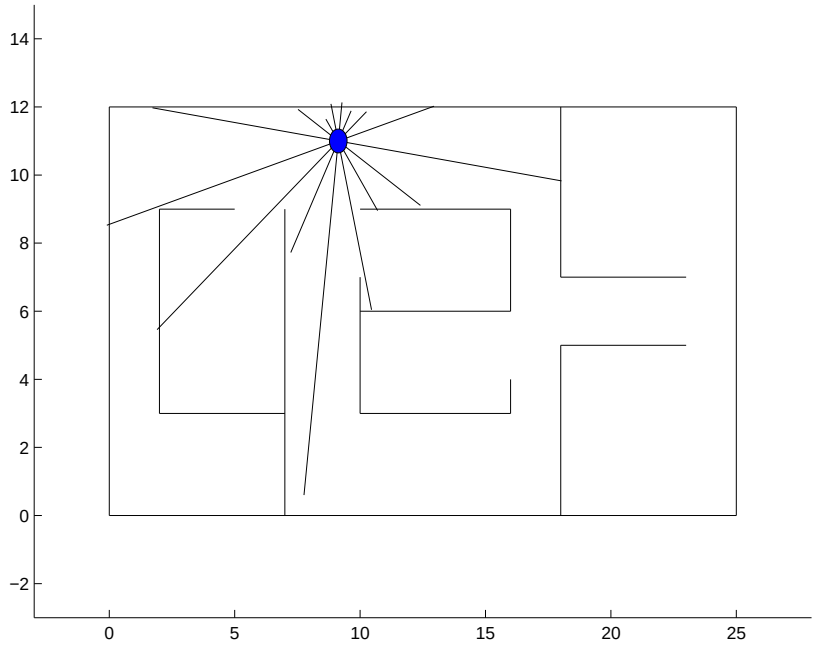
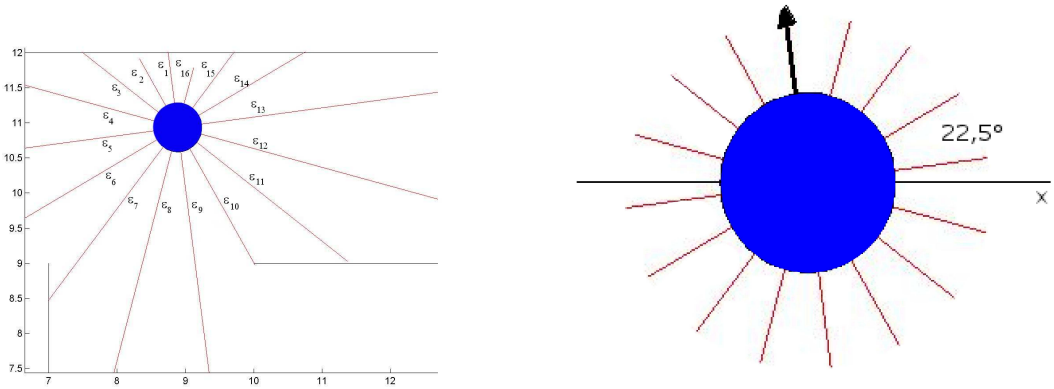


Figura 3.6: Esempio di funzionamento dei sensori di cui è dotato il robot. Ogni sensore fornisce la propria distanza dall'ostacolo più vicino e il relativo orientamento.



(a) (b)

Figura 3.7: (a) Distanza ε_i dall'ostacolo più vicino nella direzione del laser e (b) orientamento ψ_i di ogni sensore $(i = 1, ..., 16)$. In (b), la freccia indica la direzione del robot e ogni sensore è spaziato dal precedente di $22,5°$.

Algoritmo di ricampionamento *Select with Replacement*

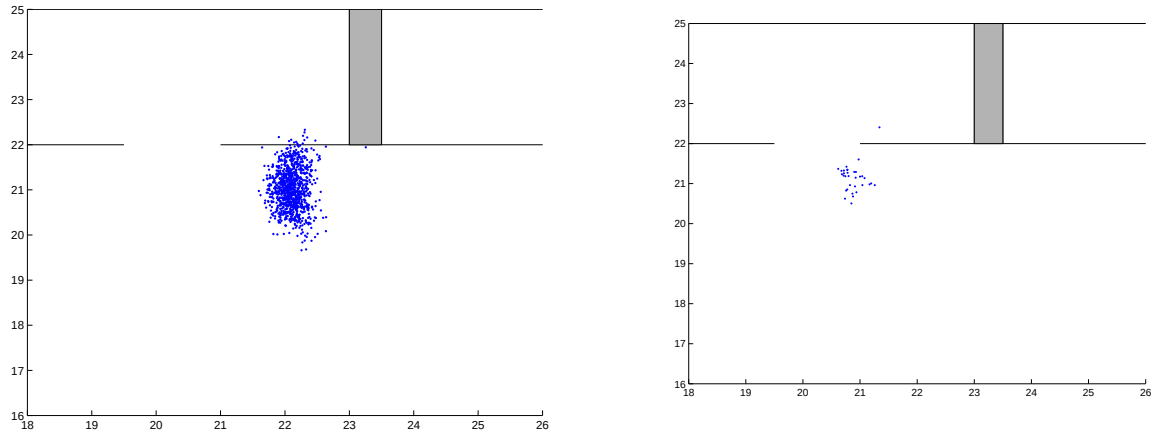
Require: $\sum_{i=1}^{N_p} W_k^i = 1$
  $Q = cumsum(W_k)$; {Viene calcolata la somma cumulativa Q}
  $t = rand(N_p + 1)$; {t è un array di $N_p + 1$ numeri random}
  $T = sort(t)$; {Ordina il vettore t}
  $T[N_p + 1] = 1; i = 1; j = 1;$
  while$(i \leq N_p)$ **do**
    if$T[i] < Q[j]$ **then**
      $Index[i] = j;$
      $i = i + 1;$
    else
      $j = j + 1;$
    end if
  end while
Return($Index$)

Tabella 3.2: Algoritmo di ricampionamento

stribuiti secondo la probabilità $p(x_{r,k}^i|z_k)$, ponendo $w_k^i = 1/N_p$. Il ricampionamento ha il difetto principale di non mantenere una certa diversità e di essere troppo deterministico. Inoltre, ha un'elevata complessità computazionale, poichè la lista delle particelle deve essere sempre esplorata per intero.

Nella realizzazione dell'algoritmo viene utilizzato il metodo di resampling *select with replacement*, descritto nel capitolo precedente e di cui si riportano in Tabella 3.2 i passaggi necessari alla sua implementazione. In ingresso viene fornito all'algoritmo il vettore dei pesi W_k^i all'istante $t = k$ e l'unica richiesta che deve essere soddisfatta è quella che il vettore dei pesi sia normalizzato prima della fase di resampling.

In Figura 3.8 si evidenzia l'effetto di una fase di ricampionamento sulle particelle per lo stesso esempio riportato in Figura 3.5. Come si può facilmente notare, molte particelle che non erano in grado di rappresentare al meglio la *pdf* vengono eliminate, a vantaggio di quelle più probabili aventi peso maggiore.



(a) (b)

Figura 3.8: Dimostrazione dell'effetto della fase di ricampionamento sull'andamento delle particelle

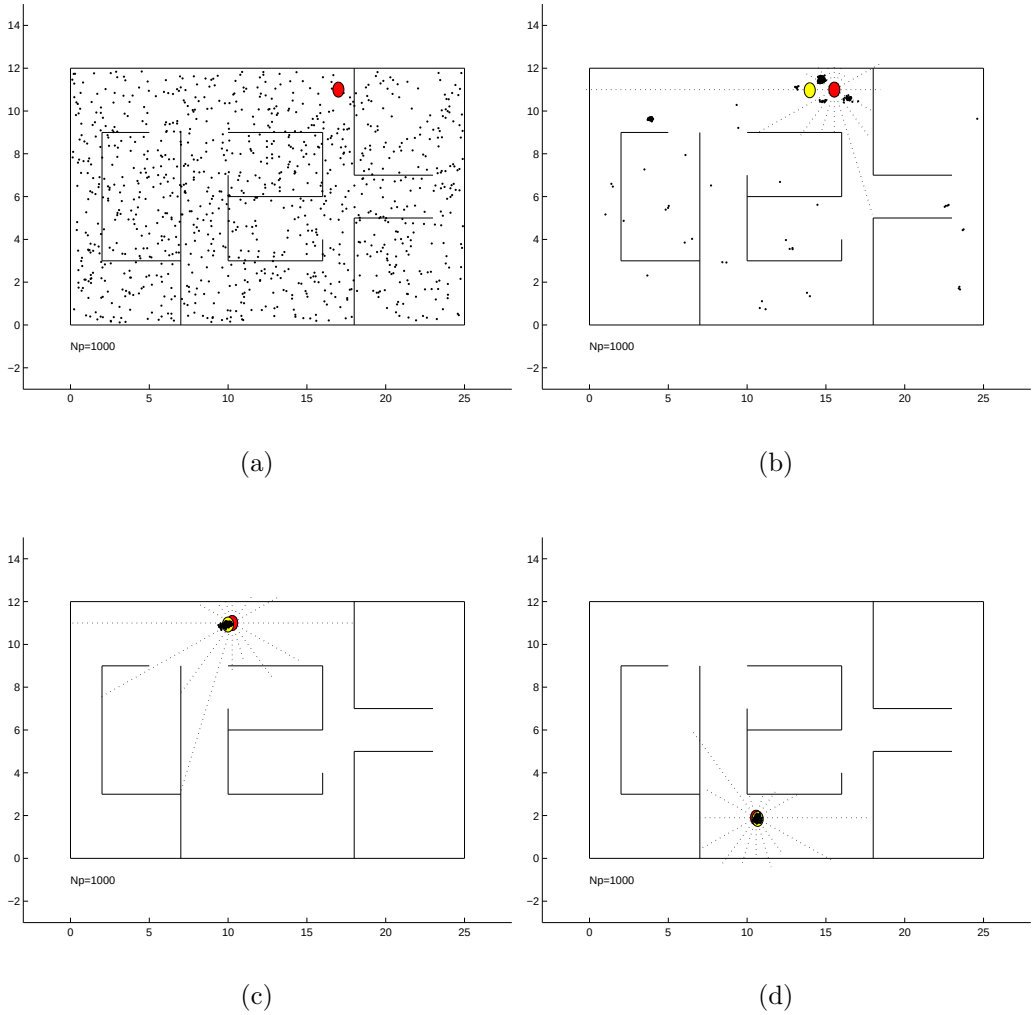
3.4.5 Stima dello stato

Data la distribuzione delle particelle all'istante $t = k$, in seguito alla fase di ricampionamento, l'algoritmo realizzato termina con il calcolo della posizione stimata del robot. Ci sono tre differenti metodi di calcolo che possono essere utilizzati per ottenere una stima dello stato del sistema:

- media pesata: $x_{r,k}^{est} = \sum_{j=1}^{N_p} W_k^j x_{r,k}^j$;
- particella con peso maggiore: $x_{r,k}^{est} = x_{r,k}^j$ tale che $W_k^j = max(W_k^m) : m = 1...N_p$;
- media pesata in un intorno della particella con peso maggiore (*media robusta*).

Il primo metodo incontra qualche problema quando tratta con distribuzioni multimodali, mentre il secondo introduce un errore costante di discretizzazione. Il metodo più efficace sarebbe la media robusta, ma esso è anche il più pesante da un punto di vista computazionale. Nell'algoritmo realizzato, per ricavare la stima, si è imple-

mentato il primo metodo. In Figura (3.9) è riportato un esempio di funzionamento dell'algoritmo fin qui descritto. Sono quattro *frame* tratti in differenti istanti di tempo, rappresentanti l'evoluzione della distribuzione delle particelle che da una incertezza globale (a) si concentrano man mano tutte nell'intorno della posizione del robot stimata (d).



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

Figura 3.9: Esempio di funzionamento dell'algoritmo di localizzazione realizzato. L'insieme di particelle rappresenta la *pdf* del robot durante la localizzazione globale. Le figure riportano la distribuzione (a) all'istante iniziale $k = 0$, (b) $k = 5$, (c) $k = 20$ e (d) $k = 60$. Il cerchio rosso indica la posizione reale del robot e quello giallo la sua stima.

Capitolo 4

Altre tecniche di localizzazione basate su filtro particellare

Vengono qui trattate due varianti dell'algoritmo di localizzazione basato su filtro particellare, che nascono dall'esigenza di risolvere alcune problematiche. Il primo è il filtro particellare con filtro di Kalman esteso come distribuzione *proposal* (*Extended Kalman Particle Filter*), mentre il secondo è il filtro particellare adattativo (*Adaptive Particle Filter*).

4.1 Filtro particellare con filtro di Kalman esteso

Un modo più accurato di generare la distribuzione *proposal* per l'approssimazione della *pdf* è la linearizzazione locale. Questo metodo prende in considerazione l'ultima osservazione disponibile, si basa sull'espansione di Taylor del primo ordine delle distribuzioni *likelihood* e *prior* e utilizza l'assunzione gaussiana su tutte le variabili casuali. Il *filtro di Kalman esteso* (*EKF*) approssima lo stimatore ottimo che minimizza l'errore quadratico medio attraverso il calcolo della media dello stato, avendo a disposizione tutte le osservazioni. Ciò viene fatto in maniera ricorsiva nel tempo, propagando l'approssimazione gaussiana della distribuzione a posteriori e combinandola con le nuove misurazioni disponibili ad ogni istante. Il filtro di Kalman esteso

calcola, dunque, l'approssimazione ricorsiva della densità reale a posteriori data da

$$p(x_k|z_{1:k}) \approx p_{\mathcal{N}}(x_k|z_{1:k}) = \mathcal{N}(\overline{x}_t, \widehat{P}_t). \tag{4.1}$$

Utilizzando tale approccio nella tecnica di filtraggio particellare, un singolo *EKF* viene impiegato per ogni particella, in modo tale da generare e propagare la distribuzione *proposal* gaussiana che segue

$$q(x_k^i|x_{0:k-1}^i, z_{1:k}) \doteq \mathcal{N}(x_k|z_{1:k}) \quad i = 1, ..., N_p. \tag{4.2}$$

Quanto detto può essere così semplificato: all'istante $t = k - 1$, vengono determinate la media e la covarianza della distribuzione per ogni singola particella, mediante l'uso delle equazioni del filtro di Kalman esteso e delle nuove misurazioni derivanti dai sensori. Quindi, è necessario propagare la matrice di covarianza $\widehat{P}^i$ e specificare le covarianze che caratterizzano il rumore del processo e del sistema di misurazione. In secondo luogo, l'i-esima particella all'istante $t = k$ viene campionata dalla distribuzione ottenuta. In Tabella 4.1 è riportato uno pseudo codice dell'algoritmo.

Il filtro di Kalman esteso è uno stimatore ottimo nel caso lineare e gaussiano e il suo utilizzo porta ad un migliore algoritmo di campionamento, nel quale la varianza di ogni distribuzione *proposal* cambia nel tempo. In ogni caso, nonostante si applichi tale tecnica al filtro particellare, si ha nuovamente a che fare sia con l'assunzione gaussiana sulla forma della distribuzione di probabilità a posteriori, sia con approssimazioni dovute alla linearizzazione. Inconvenienti questi già analizzati nei precedenti capitoli, che rendono difficoltosa la diffusione di tali algoritmi.

La realizzazione della tecnica di localizzazione derivante è del tutto simile a quella del filtro particellare affrontata nel precedente capitolo, con l'unica differenza riguardante la fase di *predizione* nella quale anziché adoperare il modello di sistema, si sono

implementate le equazioni del filtro di Kalman esteso. La fase di *resampling*, di *aggiornamento* e *normalizzazione* dei pesi rimangono inalterate.

Extended Kalman Particle Filter

Inizializzazione

Per $i = 1, ..., N_p$, campionare x_0^i da $p(x_0)$

For $t = 1, 2, ..., k$

Importance Sampling

For $i = 1, ..., N_p$

- calcolare i Jacobiani F_t^i e G_t^i del modello di sistema, H_t^i e U_t^i del modello dei sensori
- aggiornare le particelle con il filtro di Kalman esteso (EKF)

$\overline{x}_{t|t-1}^i = f(x_{t-1}^i)$

$P_{t|t-1}^i = F_t^i P_{t-1}^i F_t^{iT} + G_t^i Q_t G_t^{iT}$

$K_t = P_{t|t-1}^i H_t^{iT} [U_t^i R_t U_t^{iT} + H_t^i P_{t|t-1}^i U_t^{iT}]^{-1}$

$\overline{x}_t^i = \overline{x}_{t|t-1}^i + K_t(z_t - h(\overline{x}_{t|t-1}^i))$

$\widehat{P}_t^i = P_{t|t-1}^i - K_t H_t^i P_{t|t-1}^i$

- campionare $\widehat{x}_t^i \sim q(x_t^i | x_{0:t-1}^i, z_{1:t}) = \mathcal{N}(\overline{x}_t^i, \widehat{P}_t^i)$

end for

For $i = 1, ..., N_p$ calcolare i fattori di importanza

$$W_t^i \propto \frac{p(z_t|\widehat{x}_t^i) p(\widehat{x}_t^i|\widehat{x}_{t-1}^i)}{q(\widehat{x}_t^i|x_{0:t-1}^i, z_{1:t})}$$

end for

For $i = 1, ..., N_p$ normalizzare i fattori di importanza

$$\widetilde{W}_t^i = W_t^i \left[\sum_{j=1}^{N_p} W_t^j \right]^{-1}$$

end for

Resampling

Moltiplicare/Eliminare le particelle $(\widehat{x}_{0:t}^i, \widehat{P}_{0:t}^i)$ con alti/bassi pesi $\widetilde{W}_t^i$, rispettivamente, per ottenere N_p campioni casuali

Output: un insieme di campioni che approssimano la distribuzione a posteriori come

$$p(x_{0:t}|y_{1:t}) \approx \widehat{p}(x_{0:t}|y_{1:t}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \delta(x_{0:t} - x_{0:t}^i)$$

end for

Tabella 4.1: Pseudocodice dell'algoritmo di localizzazione basato sul filtro particellare con filtro di Kalman esteso.

4.2 Filtro particellare adattativo

Con questa variante dei filtri particellari, si vuole limitare l'errore introdotto da questi ultimi agente sulla rappresentazione della *pdf*. L'obiettivo è determinare il numero di particelle ad ogni iterazione dell'algoritmo, in modo tale che, con probabilità $1 - \delta$, l'errore tra la reale distribuzione di probabilità e l'approssimazione calcolata sia minore di un certo ε. Per ricavare ciò, si assume che la *pdf* reale sia descritta da una distribuzione discreta e costante a tratti. Per tale rappresentazione è così possibile determinare il numero di campioni tali che la distanza tra la stima della massima probabilità ottenuta mediante il campionamento e la distribuzione reale non superi una predeterminata soglia (ε). La distanza appena menzionata viene misurata mediante il concetto espresso da *Kullback-Leibler* (noto come *KL-distance*) e il corrispondente algoritmo è definito come *KLD-sampling*.
A questo punto, per prima cosa, si determina l'equazione per ricavare il numero di particelle necessario per approssimare la distribuzione di probabilità discreta. In seguito si mostra come modificare l'algoritmo basato sul filtro particellare realizzato, in modo tale da renderlo adattativo. Per ulteriori approfondimenti si consigliano [9, 10, 21].

Per ottenere un limite superiore sull'errore di approssimazione, si deve prima di tutto avere ben chiaro il concetto di *distanza di Kullback-Leibler*. Essa è una misura della differenza tra due distribuzioni di probabilità, p e q:

$$K(p, q) = \sum_x p(x) log \frac{p(x)}{q(x)}. \tag{4.3}$$

Questa distanza non è mai negativa ed è zero se e solo se le due distribuzioni sono identiche. Non è una grandezza simmetrica e non soddisfa la proprietà triangolare. Nonostante ciò, è ritenuta una grandezza standard per misurare la differenza tra due

densità di probabilità.

Supponiamo che N_p particelle siano ricavate da una distribuzione discreta, caratterizzata da k differenti *bins* (letteralmente *contenitori*) e che il vettore $\overline{X} = (X_1, ..., X_k)$ indichi il numero di particelle contenute in ogni *bin*. $\overline{X}$ è distribuito secondo una distribuzione multinomiale, cioè $\overline{X} \sim M_k(N_p, p)$, dove $p = [p_1, ..., p_k]$ indicano le probabilità di ogni *bin*. La stima della massima probabilità di p, usando N_p campioni, è data da $\widehat{p} = N_p^{-1}\overline{X}$. A questo punto si può determinare il rapporto di probabilità, dato da

$$\log \lambda_{N_p} = \sum_{j=1}^{k} X_j \log \left(\frac{\widehat{p}_j}{p_j} \right) \tag{4.4}$$

Dal momento che X_j è uguale a $N_p\widehat{p}_j$, si ottiene che:

$$\log \lambda_{N_p} = N_p \sum_{j=1}^{k} \widehat{p}_j \log \left(\frac{\widehat{p}_j}{p_j} \right). \tag{4.5}$$

Dalla (4.3) e dalla (4.5) si può concludere che il rapporto di probabilità è N_p volte la distanza-KL tra la stima della probabilità maggiore e la reale distribuzione:

$$\log \lambda_{N_p} = N_p K(\widehat{p}, p). \tag{4.6}$$

Si può dimostrare che $\log \lambda_{N_p}$ converge ad una distribuzione *chi-square*, con $k - 1$ gradi di libertà:

$$2 \log \lambda_{N_p} \rightarrow \chi^2_{k-1}. \tag{4.7}$$

con $N_p \rightarrow \infty$. Una distribuzione di probabilità *chi-square* è una delle più diffuse utilizzate in statistica. Questo tipo di densità viene impiegato nello studio della varianza campionaria, quando la distribuzione sottostante è normale, e nel test per la bontà di adattamento.

Adesso si indichi con $p_p(K(\widehat{p}, p) \leq \varepsilon)$ la probabilità che la distanza-KL determinata sia

minore o uguale di ε (sotto l'assunzione che p sia la reale distribuzione). La relazione tra tale probabilità e il numero di particelle necessario può essere ricavata come segue:

$$\begin{aligned} p_p(K(\widehat{p},p) \leq \varepsilon) &= p_p(2N_pK(\widehat{p},p) \leq 2N_p\varepsilon) \\ &= p_p(2\log\lambda_{N_p} \leq 2N_p\varepsilon) \\ &\doteq p(\chi^2_{k-1} \leq 2N_p\varepsilon) \end{aligned} \tag{4.8}$$

La relazione appena ottenuta deriva dalla (4.6). Dato un $\alpha \in (0,1)$, si indica come *quantile* α-esimo di una generica distribuzione *chi-square* $y \sim \chi^2$, il valore χ^2_α, tale che $p(y < \chi^2_\alpha) = \alpha$. Usando questa definizione, è possibile scrivere la seguente proprietà:

$$p(\chi^2_{k-1} \leq \chi^2_{k-1,1-\delta}) = 1-\delta. \tag{4.9}$$

A questo punto se si sceglie N_p tale che $2N_p\varepsilon$ sia uguale a $\chi^2_{k-1,1-\delta}$, si ricava dalle relazioni precedenti che:

$$p_p(K(\widehat{p},p) \leq \varepsilon) \doteq 1-\delta. \tag{4.10}$$

Si è finalmente ottenuta una relazione che lega il numero di particelle N_p e la qualità dell'approssimazione della *pdf*. Quindi, se si sceglie

$$N_p = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta}, \tag{4.11}$$

è garantito, dalla (4.10), che, con probabilità $1-\delta$, la distanza-KL tra la stima della probabilità e la reale distribuzione è minore di ε. Per determinare N_p, secondo la (4.11), è necessario dunque calcolare i quantili della distribuzione *chi-square*. Una buona approssimazione della (4.11) proposta in letteratura è data da

$$N_p = \frac{k-1}{2\varepsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3, \tag{4.12}$$

dove $z_{1-\delta}$ è il più grande $1-\delta$ quantile della distribuzione normale standard $\mathcal{N}(0,1)$. Tale valore, per tipici valori di δ, è disponibile nelle tabelle statistiche standard (vedi

l'Appendice A).

L'approccio che utilizza la relazione appena determinata fa uso di una griglia fissata a priori per approssimare la distribuzione multinomiale. Durante la fase di predizione, l'algoritmo determina se i campioni generati si trovano o meno all'interno di una cella vuota della griglia (la griglia viene ad ogni fase di aggiornamento resettata). Se la generica cella è vuota, il numero di *bin* k viene aumentato e la cella b corrispondente viene marcata come non vuota. Dopo ogni campione, il numero di particelle viene aggiornato, utilizzando l'equazione (4.12) con l'attuale valore di k. L'aumento delle particelle termina non appena tutti i contenitori b sono stati marcati come non vuoti, dal momento che in tal modo k non cresce più e di conseguenza N_p si stabilizza al valore cercato.

Questo conclude la derivazione del numero di particelle necessario per approssimare una distribuzione discreta con limite superiore ε. Si nota come il numero richiesto sia proporzionale all'inverso dell'errore limite, cioè $N_p \propto \frac{1}{\varepsilon}$; quindi tanto più piccolo è l'errore richiesto, tanto maggiore sarà il numero di particelle necessario per soddisfare la relazione.

Si affronta adesso il problema di includere il risultato appena ottenuto nell'algoritmo dei filtri particellari. Il problema che si presenta è la non conoscenza della distribuzione reale di probabilità a posteriori. La soluzione adottata è quella di lavorare con la distribuzione ricavata dalla fase di predizione, utilizzandola come una stima della densità a posteriori. Inoltre la (4.12) mostra come non sia necessario determinare l'intera distribuzione discreta, ma è sufficiente calcolare il numero k di *bin* finchè non vengono tutti riempiti. Sebbene questa quantità non sia inizialmente nota, si generano tutti i campioni dalla distribuzione a priori e si stima il valore di k durante il campionamento.

Uno pseudo-codice della fase di aggiornamento del filtro particellare adattativo realizzato con il *KLD-sampling* è riportato in Tabella 4.2.

Algoritmo *KDL-sampling*

Inputs: $S_{t-1} = \{(x_{t-1}^i, W_{t-1}^i) | i = 1, ..., N_p\}$ $\{$*Rappresenta la* $Bel(x_{t-1})\}$,
le misurazioni z_t, i limiti ε e δ, numero minimo di campioni $n_{\chi min}$
//Inizializzazione:
$S_t = 0$, $N_p = 0$, $n_\chi = 0$, $k = 0$, $\alpha = 0$;
do *//**Generare particelle...***
//Sampling, predizione dello stato successivo:
Campionare $x_t^{N_p}$ da $p(x_t|x_{t-1})$ utilizzando $x_{t-1}^{N_p}$;
//Calcolo del peso della particella considerata:
$W_t^{N_p} = p(z_t|x_t^{N_p})$;
//Aggiornamento del fattore di normalizzazione:
$\alpha = \alpha + W_t^{N_p}$;
//Aggiornamento dell'insieme delle particelle:
$S_t = S_t \bigcup \{(x_t^{N_p}, W_t^{N_p})\}$;
if ($x_t^{N_p}$ *si trova in un bin b vuoto*) **then**
//Aggiornamento del numero di contenitori non vuoti:
$k = k + 1$;
//Viene marcato il contenitore riempito:
$b =$*non vuoto*;
if $k \geq 2$ **then**
//Aggiornamento del numero desiderato di particelle:
$$n_\chi = \frac{k-1}{2\varepsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta}\right\}^3;$$
end if
end if
//Aggiornamento numero di campioni generati:
$N_p = N_p + 1$;
while ($N_p < n_\chi$ or $N_p < n_{\chi min}$) *//**... finchè il limite-KL viene raggiunto***
for $i = 1, ..., N_p$ **do**
//Normalizzazione dei pesi:
$W_t^i = \frac{W_t^i}{\alpha}$;
end for
Return S_t

Tabella 4.2: Filtro particellare adattativo: fase di campionamento.

Come si può notare, l'aggiornamento del numero di *bin* k non vuoti viene effettuato successivamente alla generazione dei campioni ed è inoltre eseguito in modo incrementale, analizzando ogni singola particella se si trova oppure no in un *bin* vuoto. Ogni volta che viene aggiornato k, si utilizza la (4.12) per calcolare il numero n_χ di particelle richieste. Nelle fasi iniziale del campionamento, k cresce all'incirca ad ogni particella generata, dal momento che tutti i contenitori sono vuoti. L'aumento di k comporta un aumento di campioni desiderati n_χ, ma col passare del tempo, accade che molti contenitori vengono riempiti e quindi n_χ cresce solo occasionalmente. Al contrario, N_p aumenta con la generazione di ogni nuova particella, pertanto l'algoritmo termina non appena N_p raggiunge il valore n_χ.

L'implementazione del filtro particellare adattativo è abbastanza semplice. La differenza con l'algoritmo standard consiste nel fatto che adesso si deve tenere memoria del numero k di contenitori non vuoti e delle particelle desiderate n_χ. Tutte le fasi sono state realizzate seguendo il ragionamento svolto nel precedente capitolo. Per ulteriori chiarimenti riguardanti questo approccio, si consiglia [9, 10, 21].

Capitolo 5

Simulazioni

In questo capitolo si riportano i risultati di alcune simulazioni effettuate utilizzando il software realizzato. Tutto è stato svolto in ambiente *Matlab 6.5*, programma per il calcolo numerico e linguaggio di programmazione interpretato, creato dalla *MathWorks*. Le simulazioni riguardano diversi casi, più o meno complessi, mirate ad evidenziare pregi e difetti dei filtri particellari. Verranno utilizzate le due varianti dei filtri particellari analizzate nel capitolo precedente. Per prima cosa si descrive la fase iniziale del software realizzato, che consente di modificare arbitrariamente i dati della simulazione.

5.1 Dati della simulazione

Il software è dotato di un'interfaccia grafica iniziale, visualizzabile eseguendo il file *main.m*, che consente di inserire i dati desiderati ad ogni simulazione. In Tabella 5.1 sono riportate le informazioni che possono essere modificate dall'utente, seguite da una breve descrizione. I valori di default di tali parametri sono elencati in Tabella 5.2 e la Figura (5.1) visualizza invece l'interfaccia creata di cui si sta parlando. Di seguito una spiegazione di ogni parametro.

Algoritmo di localizzazione L'utente può scegliere quale algoritmo di localizza-

Algoritmo di localizzazione	Permette di scegliere tra tre differenti tipi di algoritmo (*PF*,*EKPF*,*APF*)
Inizializzazione	Problema da affrontare: *global localization* oppure *position tracking*
Numero di passi	Numero di *step* (k) complessivi della simulazione
Numero di sensori	Numero di sensori (N_{sensor}) del robot
Numero di particelle	Numero di particelle (N_p) da utilizzare per la localizzazione
Varianza dei sensori	Varianza da considerare sulle letture effettuate dai sensori
Err.odometrico trasl.	Costante positiva (K_ρ) che caratterizza la varianza del rumore che affligge le letture odometriche
Err.odometrico rotaz.	Costante positiva (K_θ) che caratterizza la varianza del rumore che affligge le letture odometriche
Raggio robot	Indica il raggio (R) della circonferenza del robot
Stima dello stato	Permette di scegliere come effettuare la stima dello stato del robot ($x_{r,k} = [x_k, y_k, \theta_k]^T$)
Ordine sensori	Si può decidere se posizionare i sensori solamente su una semicirconferenza o sull'intera circonferenza del robot
Scelta mappa	Si possono svolgere simulazioni in diversi ambienti

Tabella 5.1: Dati della simulazione che è possibile cambiare mediante l'interfaccia grafica creata.

Filtro particellare
Problema di localizzazione globale
$k_{tot} = 100$
$N_{sensor} = 16$
$N_p = 1000$
$\xi_{laser} = 0.1m^2$
$K_\rho = 0.01m$
$K_\theta = 0.02rad^2/m$
$R = 0.1m$
Stima dello stato mediante media pesata
Sensori disposti sull'intera circonferenza
Mappa asimmetrica numero 6

Tabella 5.2: Dati simulativi di default.

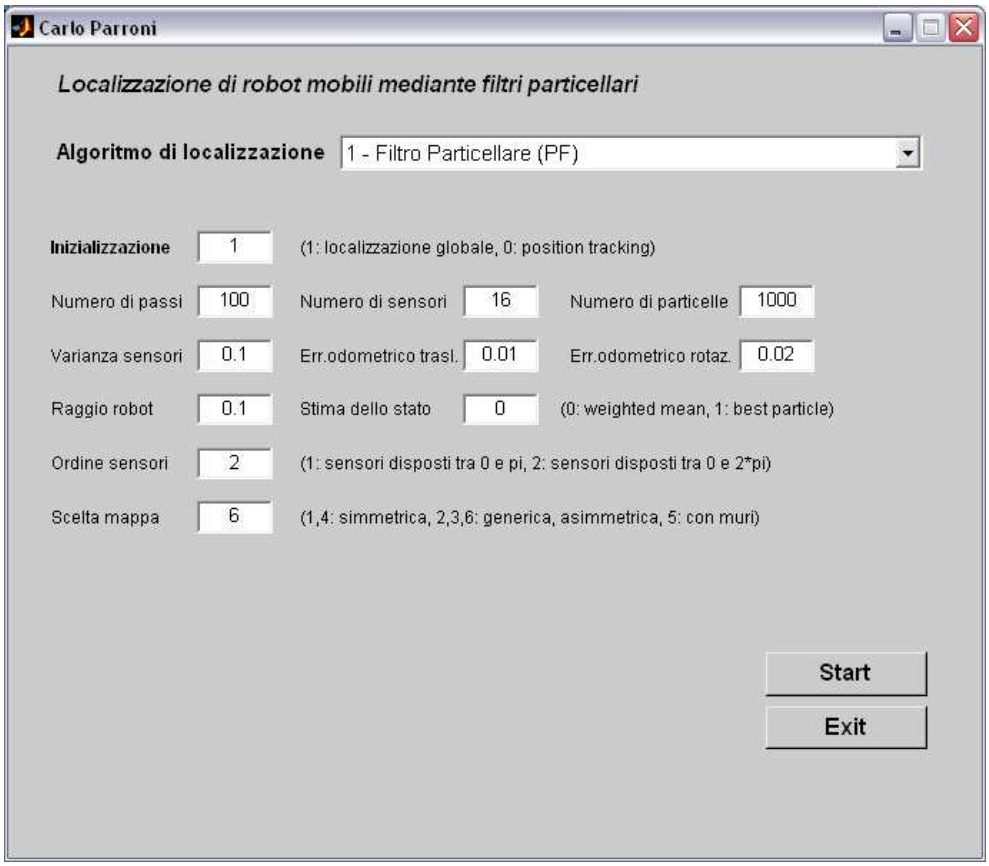


Figura 5.1: Interfaccia grafica del software realizzato che permette di inserire i dati desiderati della simulazione, ottenuta mediante il comando *uicontrol* $('PropertyName1', value1, 'PropertyName2', value2, ...)$ di *Matlab*.

zione utilizzare per la simulazione corrente. Si può scegliere tra i tre algoritmi presentati in questa tesi, quindi localizzazione mediante *filtro particellare*, *filtro particellare con filtro di Kalman esteso* e *filtro particellare adattativo*. I dati riportati nella Tabella 5.1 sono comuni a tutte e tre le tecniche appena menzionate; oltre ad essi, se si desidera svolgere la simulazione con il *EKPF* ci sarà la possibilità di modificare il valore della *deviazione standard* all'istante iniziale, mentre se si sceglie il *APF* si può inserire il parametro *delta* δ caratterizzante la probabilità (4.10), il *limite superiore d'errore* ε, il *minimo numero desiderato*

di particelle $n_{\chi min}$, la dimensione delle celle che formano la griglia in cui viene suddivisa la mappa e il quantile $z_{1-\delta}$ della distribuzione normale $\mathcal{N}(0,1)$.

Inizializzazione Si può decidere se si vuole affrontare un problema di localizzazione globale (1) oppure il più semplice *position tracking* (0). In quest'ultimo caso si da la possibilità all'utente di cambiare anche la varianza delle particelle generate all'istante iniziale nell'intorno della posizione nota del robot.

Numero di passi Semplicemente si può decidere quanto far durare la simulazione. Si ricorda che si sta lavorando con un sistema a tempo discreto, pertanto la scelta deve essere data in termini di numero di passi in cui si vuole suddividere il percorso prefissato del robot.

Numero di sensori Anche in questo caso si deve solamente inserire il numero di sensori con cui equipaggiare il robot per la simulazione. Tale parametro è legato all'*ordine sensori*, dal momento che se si decide di piazzare N_{sensor} sull'intera circonferenza del robot, ognuno di essi sarà distante dal precedente e dal successivo $2\pi/N_{sensor}$ gradi. Quindi maggiore è N_{sensor} e più accurate saranno le osservazioni che il robot riceve ad ogni istante.

Numero di particelle Questo è forse il parametro più importante da inserire. Infatti, se si ha a disposizione un numero troppo basso di particelle in un problema di localizzazione globale, facilmente può accadere di non riuscire a trovare una soluzione, mentre un numero troppo elevato risolve sicuramente il problema ma rende la simulazione molto lenta. Negli esempi riportati nei paragrafi successivi, si metterà in evidenza il valore che viene assegnato a questo parametro. Nel caso di scelta dell'algoritmo di localizzazione *APF*, con N_p si intende ovviamente il numero di particelle iniziale.

Varianza sensori Viene data la possibilità di inserire un valore che indichi la varianza del rumore che agisce sulle letture sensoriali. È evidente che maggiore è tale valore, peggiori sono le misurazioni che effettua il robot che avrà maggiori difficoltà a localizzarsi.

Err.odometrico trasl. Questo è il parametro costante positivo K_ρ, già incontrato nella descrizione del modello di sistema, caratterizzante la varianza del rumore che influenza le letture degli encoder (vedi (3.3)). In particolare si riferisce al rumore dovuto al movimento di traslazione del robot.

Err.odometrico rotaz. Analogo al precedente, con la differenza che deriva dal rumore dovuto al movimento di rotazione del robot (vedi (3.3)).

Raggio robot Parametro che serve per indicare a quale distanza si trovano posizionati i sensori dal centro del robot. Descrive il raggio della sua circonferenza ed è molto importante da tenere in considerazione per ottenere delle misurazioni sensoriali corrette (vedi Figura (3.2)).

Stima dello stato Consente di scegliere come effettuare ad ogni istante di tempo la stima dello stato. È possibile optare per la *media pesata* (0) oppure semplicemente assegnare alla stima la posizione e l'orientamento della particella con peso maggiore (1).

Ordine sensori Come già discusso a proposito del *numero di sensori*, si può decidere se posizionarli sull'intera circonferenza del robot (2) oppure solo su una semicirconferenza (1).

Scelta mappa Si può scegliere tra sei diversi ambienti di simulazione $(1-6)$. Alcuni di questi verranno presentati più avanti durante le simulazioni.

5.2 Problema di position tracking

Per tutti gli esempi che seguono, al termine della simulazione viene determinato un errore medio di stima $\mathcal{E}_{medio}$; esso considera nient'altro che la distanza media tra la reale posizione del robot e quella stimata ad ogni istante. Si ha pertanto che $\mathcal{E}_{medio} = \sqrt{\frac{1}{N_{passi}} \sum_k [(x_k - \widehat{x}_k)^2 + (y_k - \widehat{y}_k)^2]}$, dove $\widehat{x}_k$ e $\widehat{y}_k$ rappresentano la stima dello stato all'istante k.

Questa prima simulazione mostra qual'è il comportamento dell'algoritmo nell'affrontare un problema di *position tracking*. Si utilizzano i dati di default (Tabella 5.2) ad eccezione, ovviamente, del parametro *Inizializzazione* posto uguale a 0. La varianza iniziale delle particelle è pari a $0.3m$. Per tale esempio si riportano i risultati ottenuti solamente con il filtro particellare, dal momento che le varianti proposte non sono molto utili in questo tipo di problemi. L'ambiente in cui si svolge la simulazione si può ottenere ponendo uguale a 5 il parametro *scelta mappa*.
Si riporta, in Figura (5.2), la distribuzione delle particelle all'istante iniziale. Si vuole sottolineare che i risultati ottenuti dalla simulazione sono pienamente soddisfacenti, dal momento che la posizione del robot è stata tracciata nell'arco dell'intero percorso, così come mostrato in Figura (5.3). La Figura (5.4), invece, descrive l'errore derivante dalla differenza tra la configurazione del robot e quella stimata, ad ogni istante di tempo k. In blu la distanza tra le due posizioni, mentre in verde si evidenzia la differenza nell'orientamento. Si pone l'attenzione sui due picchi presenti in cui si ha errore massimo; essi sono dovuti alle rotazioni di 90° imposte al robot, momento in cui, oltre al rumore generato dal movimento di traslazione, si aggiunge al modello anche il rumore causato dalla rotazione. A proposito di questa simulazione si determina l'errore medio, che risulta essere pari a $\mathcal{E}_{medio} = 0.287m$.

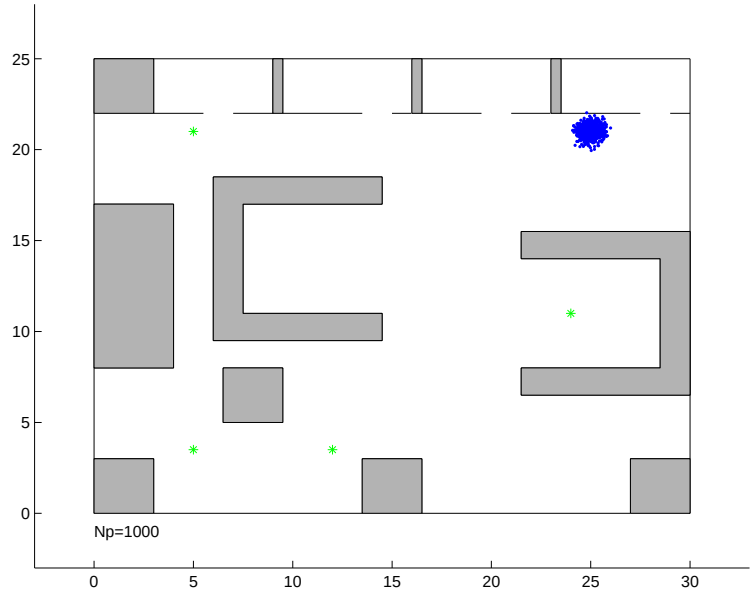


Figura 5.2: Position tracking. Inizializzazione delle particelle nell'intorno della posizione iniziale nota del robot.

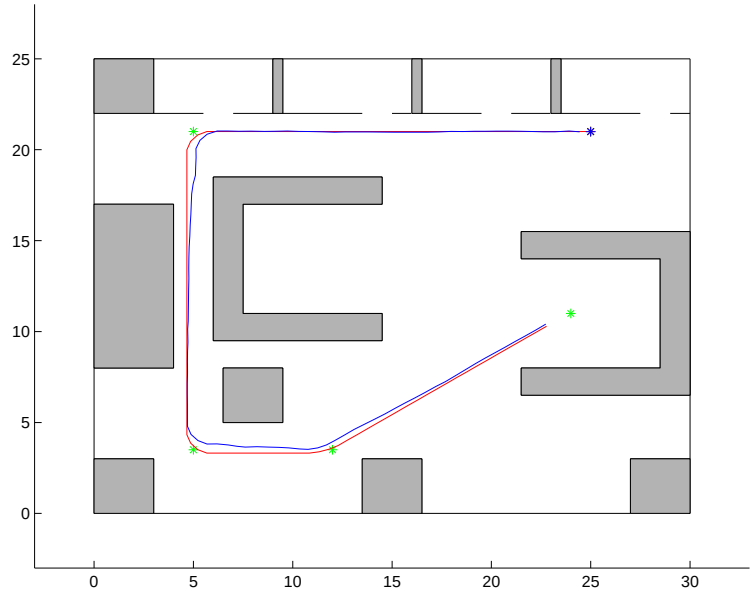


Figura 5.3: Position tracking. Percorso reale del robot (in rosso) e cammino stimato (in blu). In verde sono indicati i punti di passaggio del percorso assegnato al robot.

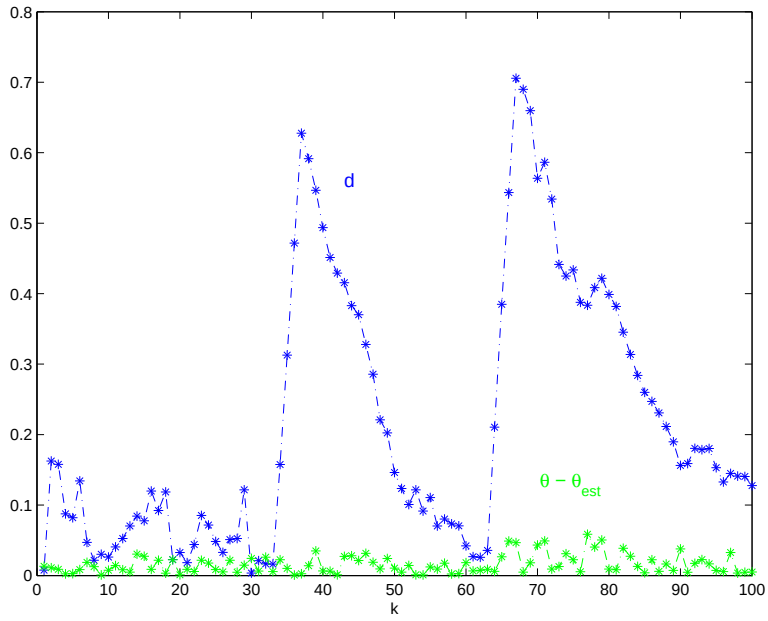


Figura 5.4: Position tracking. In blu la distanza d tra la posizione del robot e quella stimata, ad ogni istante di tempo. In verde la differenza tra orientamento reale θ e stimato θ_{est}.

5.3 Localizzazione globale in ambienti asimmetrici

Il secondo esempio tratta di un problema di localizzazione globale in un ambiente piuttosto asimmetrico; la posizione del robot non è quindi nota a priori e la distribuzione iniziale delle particelle è uniforme sull'intero spazio di stato. Ovviamente, maggiore è il numero di particelle utilizzate, migliore è la stima della *pdf* desiderata, ma nello stesso tempo aumenta la complessità computazionale. Si esegue l'algoritmo lasciando invariati tutti i parametri di default. La situazione all'istante iniziale $t = 0$ è mostrata in Figura (5.5). Questa simulazione è stata svolta mediante tutti e tre i filtri a disposizione. Per primo si è impiegato l'algoritmo di localizzazione con *filtro particellare*. In questo caso la distribuzione delle particelle all'istante $t = 5$ e $t = 15$ è riportata in Figura (5.6). Come si può notare, già dopo solo 15 iterazioni, l'algoritmo

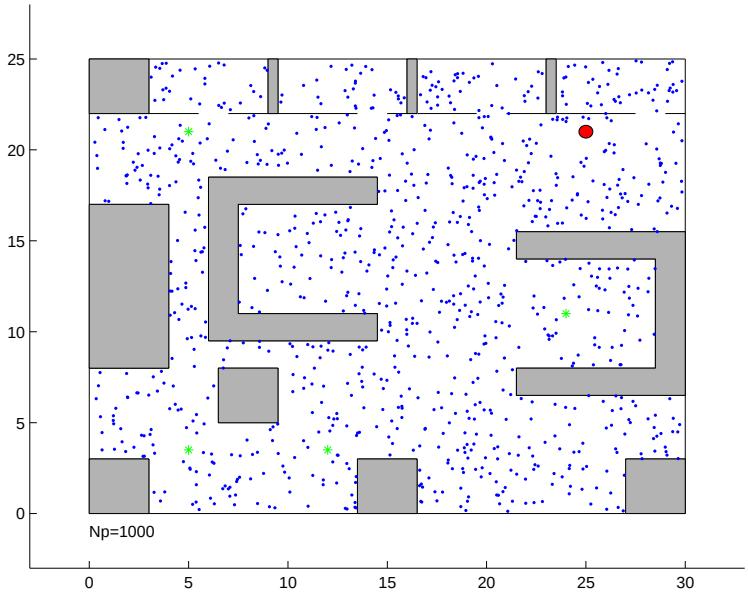
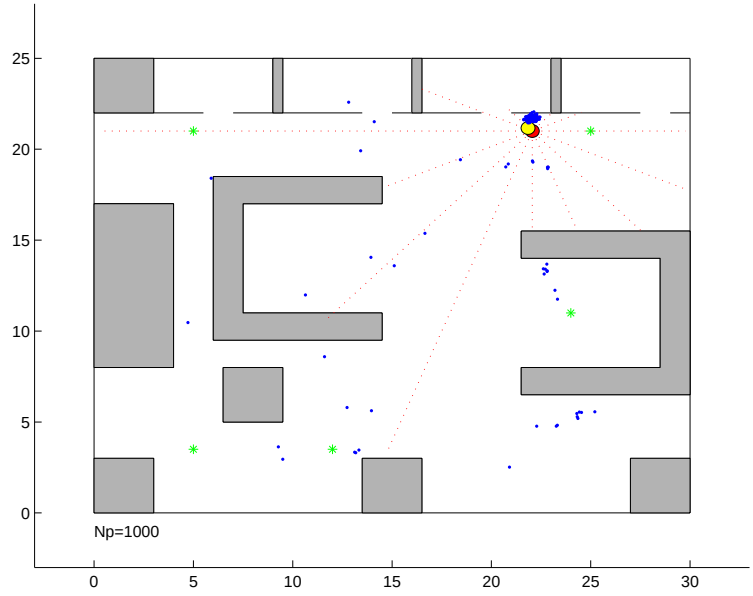


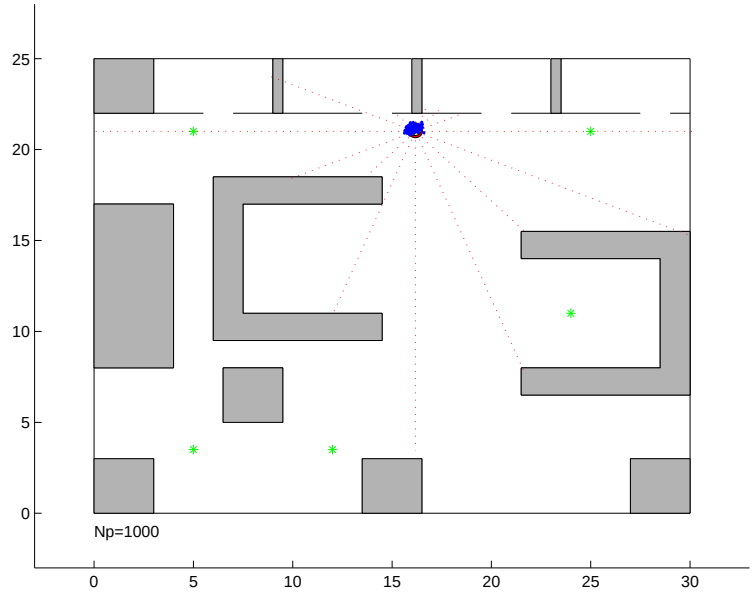
Figura 5.5: Localizzazione globale. Distribuzione iniziale delle particelle sull'intero spazio di stato. In rosso la posizione del robot e in verde i punti di passaggio.

è riuscito ad identificare la posizione del robot con buona precisione ed accuratezza. La Figura (5.7) mostra l'andamento della stima dello stato durante l'intero cammino. L'errore medio ottenuto in questo caso è $\mathcal{E}^{PF}_{medio} = 2.085m$ e non è di molto aiuto andarlo a confrontare con il risultato del problema di *position tracking*, dal momento che in questo caso, per le prime iterazioni, la posizione non è nota, pertanto l'errore di stima è molto più grande ed influenza l'indice d'errore medio finale. Se comunque non si prendono in considerazione i primi 15 passi della simulazione, si ottiene $\mathcal{E}^{PF}_{medio} = 0.265m$, molto simile al risultato dell'esempio precedente.

La stessa simulazione viene eseguita utilizzando il *filtro particellare con filtro di Kalman esteso*. Come si può evidenziare dalla Figura (5.8), la stima sembra essere peggiorata se paragonata al caso precedente. Questa idea viene confermata anche dal calcolo dell'errore medio, che è risultato essere uguale a $\mathcal{E}^{EKPF}_{medio} = 2.997m$, superiore



(a)

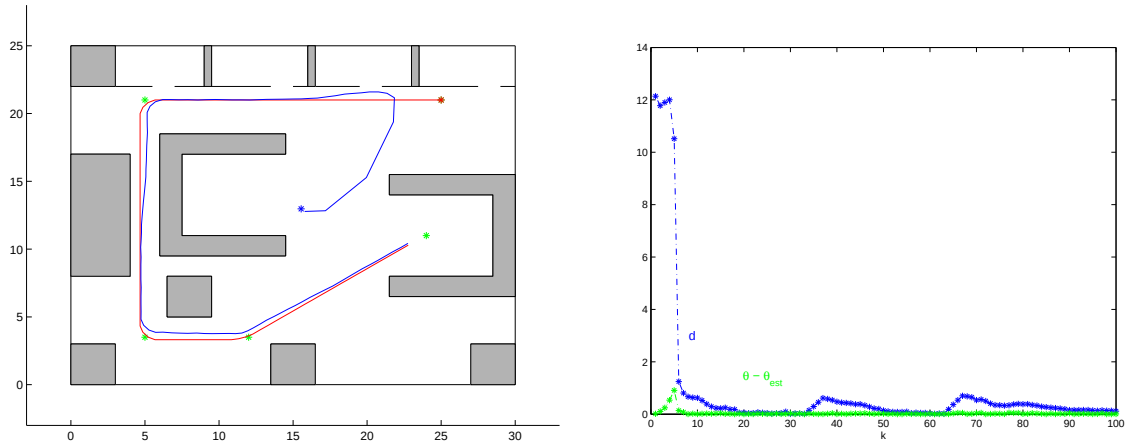


(b)

Figura 5.6: Localizzazione globale con *filtro particellare*. In rosso la posizione del robot, in giallo la stima dello stato in (a) $k = 5$ e in (b) $k = 15$.

a $\mathcal{E}_{medio}^{PF}$.

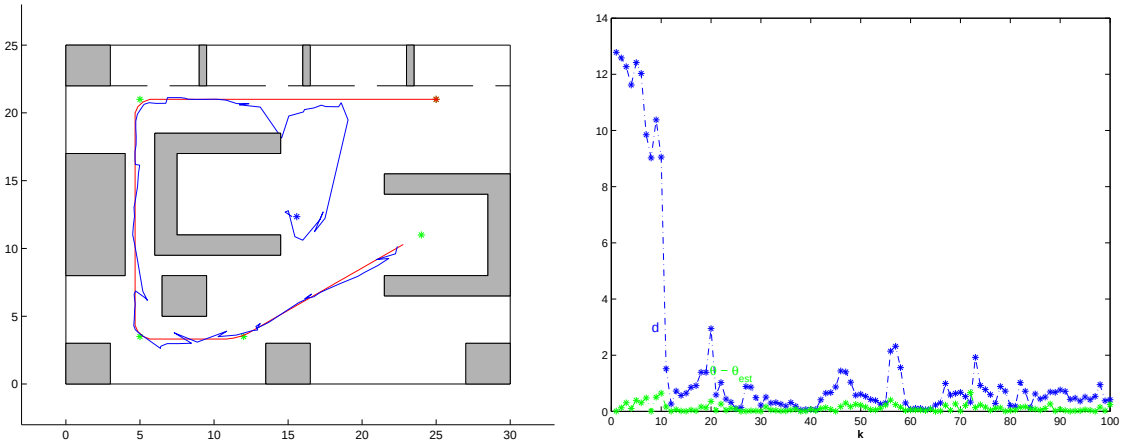
Per finire si riporta anche quanto ottenuto con l'altra variante del filtro particellare, il *filtro particellare adattativo*, il cui funzionamento è mostrato in Figura (5.9). Per questa simulazione si è adottato inizialmente un numero di particelle pari a 3000, ponendo $\delta = 0.01$ e $\varepsilon = 0.1$. Con tale tecnica si ottiene un indice di errore di stima pari a $\mathcal{E}_{medio}^{APF} = 2.287m$. È interessante sottolineare come N_p sia uguale a 50 all'istante finale, ossia N_p raggiunge il minimo numero di particelle desiderato. Il filtro particellare adattativo, infatti, richiede un elevato numero di particelle solamente nelle prime fasi della localizzazione. Con l'aumentare delle informazioni sensoriali ed odometriche a disposizione, N_p diminuisce, fino ad assestarsi al valore minimo $n_{\chi min}$ non appena si è identificata in maniera univoca la configurazione del robot.



(a) (b)

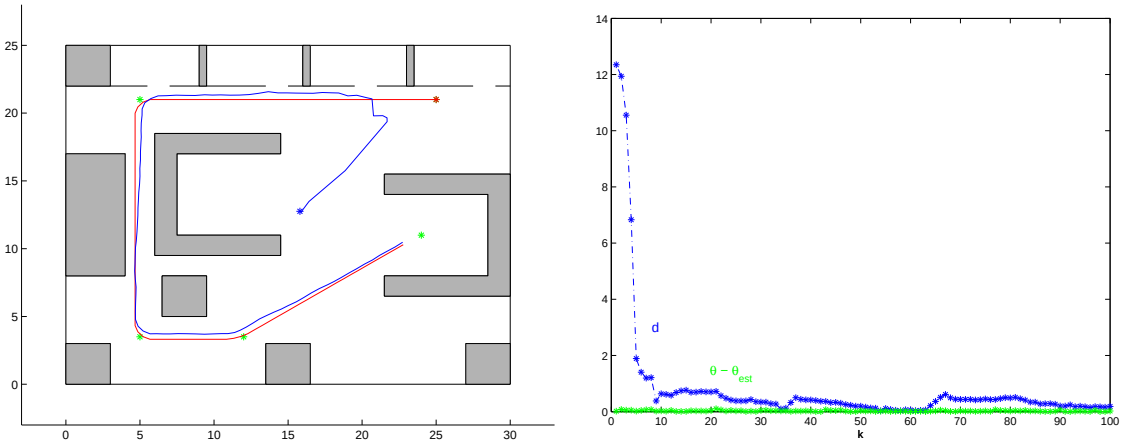
Figura 5.7: Localizzazione globale con *filtro particellare*. (a) Percorso del robot (in rosso) e traiettoria stimata (in blu), (b) distanza tra la posizione d (in blu) e l'orientamento (in verde) del robot e della stima calcolata.

Vengono ora riportati i risultati di un'altra simulazione, svolta in un ambiente differente rispetto al precedente. La mappa qui utilizzata è identificata con il numero 0



(a) (b)

Figura 5.8: Localizzazione globale con *EKPF*. (a) Percorso del robot (in rosso) e traiettoria stimata (in blu), (b) differenza tra la posizione (in blu) e l'orientamento (in verde) del robot e della stima calcolata.



(a) (b)

Figura 5.9: Localizzazione globale con *filtro particellare adattativo*. (a) Percorso del robot (in rosso) e traiettoria stimata (in blu), (b) differenza tra la posizione (in blu) e l'orientamento (in verde) del robot e della stima calcolata.

dall'algoritmo. Nuovamente tutti i dati della simulazione sono posti uguali a quelli di default. Si utilizza per primo il *filtro particellare adattativo*, per il cui utilizzo si sceglie il numero di particelle $N_p = 3000$. Si consiglia di scegliere sempre un numero di particelle abbastanza elevato quando si ha a che fare con il *APF*, dal momento che questo algoritmo genera particelle finchè viene raggiunto il limite-KL, come visto nel capitolo precedente. Pertanto potrebbe capitare di scegliere N_p troppo piccolo, tale che $N_p \geq n_\chi$, fatto questo che porta alla conclusione anticipata dell'algoritmo, in quanto non si è raggiunto il numero di campioni tali che la distanza, misurata mediante la *KL-distance*, tra la stima della massima probabilità ottenuta mediante il campionamento e la distribuzione reale non superi la predeterminata soglia ε (vedi Tabella 4.2). I parametri ε e δ sono posti uguali a 0.1 e 0.01, rispettivamente. L'inizializzazione è mostrata in Figura (5.10). L'esecuzione del programma porta ad un errore medio $\mathcal{E}^{APF}_{medio} = 1.962m$ e già dopo 5 iterazioni la localizzazione si può ritenere soddisfacente, come evidenziato in Figura (5.11). Se si considerasse l'indice di errore a partire dal quindicesimo passo, superata dunque la fase di incertezza globale sulla posizione del robot, si che che $\mathcal{E}^{APF}_{medio} = 0.38m$.

L'utilizzo dell'algoritmo basato su *filtro particellare* porta, invece, ad un errore medio leggermente superiore e pari a $\mathcal{E}^{PF}_{medio} = 2.189m$ (vedi Figura (5.12)). L'errore di solo inseguimento della posizione a partire dall'istante $t = 15$ è invece $\mathcal{E}^{PF}_{medio} = 0.302m$.

5.4 Localizzazione globale in ambienti simmetrici

Obiettivo principale di questo paragrafo è quello di mostrare un limite proprio dei filtri particellari, già accennato nei capitoli precedenti. Si utilizzi a tal proposito, la mappa riportata in Figura (5.13), nella quale è raffigurata anche la distribuzione iniziale delle particelle, scelte, per questo esempio, in numero pari a $N_p = 2000$.

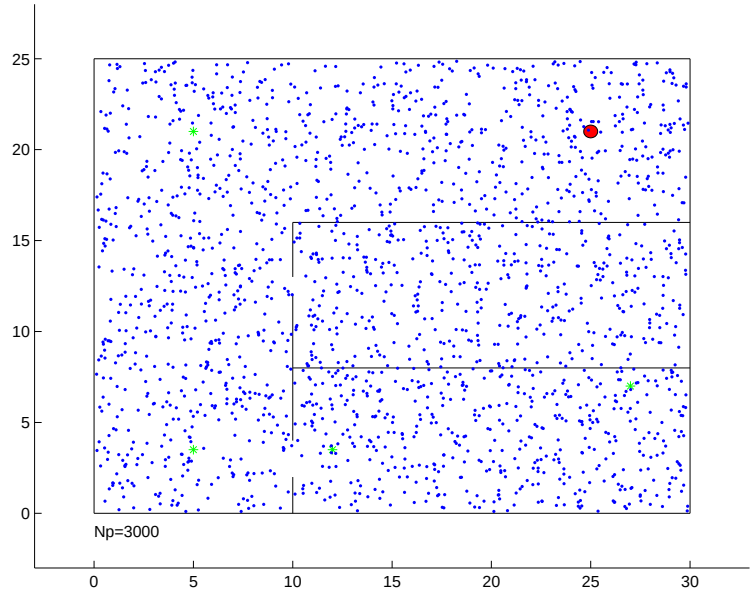
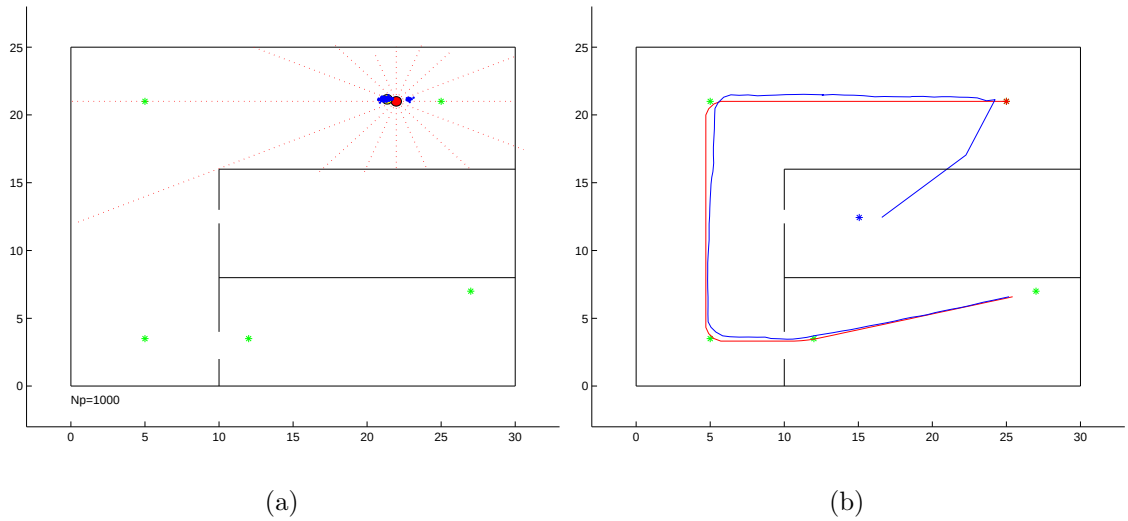
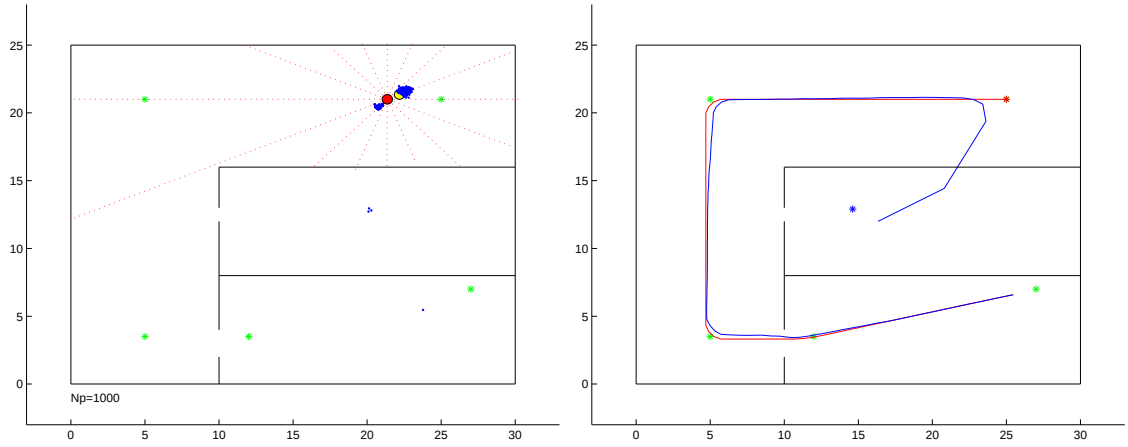


Figura 5.10: Localizzazione globale. Distribuzione iniziale delle particelle sull'intero spazio di stato. In rosso la posizione del robot e in verde i punti di passaggio.



(a) (b)

Figura 5.11: Localizzazione globale con *filtro particellare adattativo*. (a) In rosso la posizione del robot, in giallo la stima dello stato in $k = 5$, (b) percorso del robot (in rosso) e traiettoria stimata (in blu).



(a) (b)

Figura 5.12: Localizzazione globale con *filtro particellare*. (a) In rosso la posizione del robot, in giallo la stima dello stato in $k = 5$, (b) percorso del robot (in rosso) e traiettoria stimata (in blu).

La mappa adottata corrisponde alla numero 1 per il programma realizzato. Come si percepisce fin da subito, l'ambiente dove il robot deve localizzarsi non è stato scelto in maniera casuale, ma presenta una caratteristica facilmente individuabile: è perfettamente simmetrico. È proprio questo che si vuole sottolineare, ossia la difficoltà da parte del filtro particellare di riuscire a determinare univocamente la posizione del robot in ambienti che non presentano asimmetrie.

Come mostrato in Figura (5.14a), il filtro particellare non riesce a determinare con esattezza la configurazione del robot, dal momento che non ha le informazioni necessarie per eliminare le particelle superflue. L'ambiguità riguardante la posizione da stimare, non si riesce a superare in un ambiente di questo tipo, neanche col passare del tempo (vedi Figura (5.14b)). Ovviamente questo è un caso estremo, in cui la mappa è perfettamente simmetrica. Tale situazione si può però presentare in tutti quei casi dove in una mappa è possibile individuare due ambienti uguali.

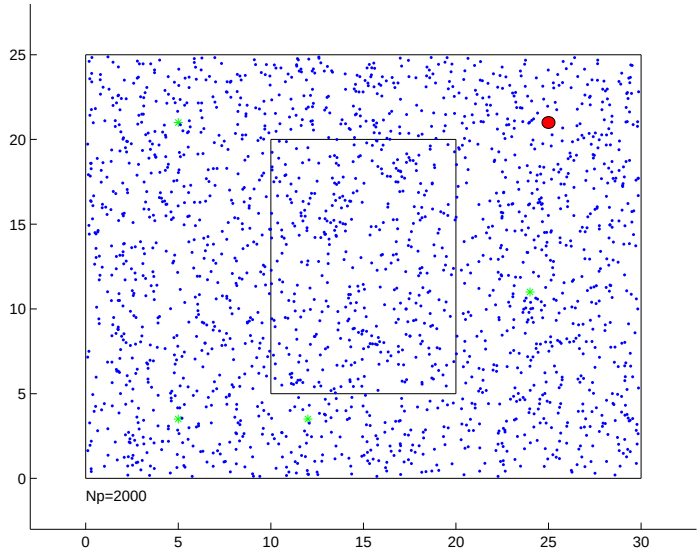
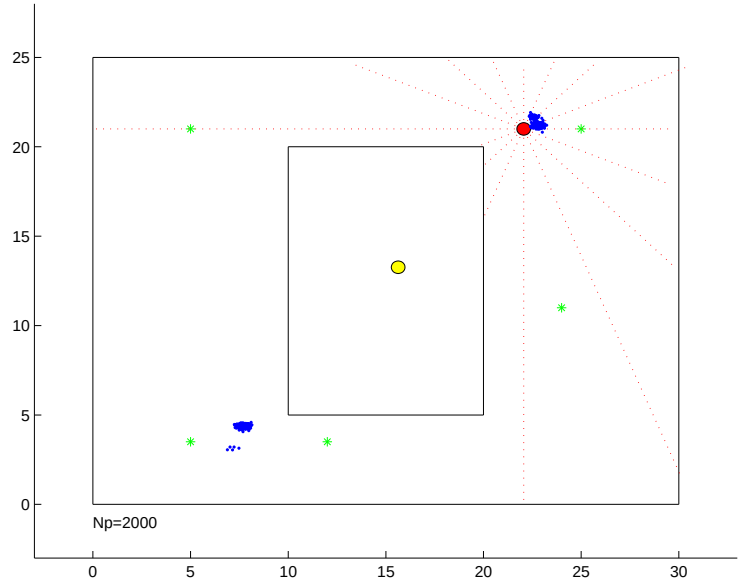


Figura 5.13: Localizzazione globale in un ambiente simmetrico. Distribuzione iniziale delle particelle sull'intero spazio di stato. In rosso la posizione del robot e in verde i punti di passaggio.

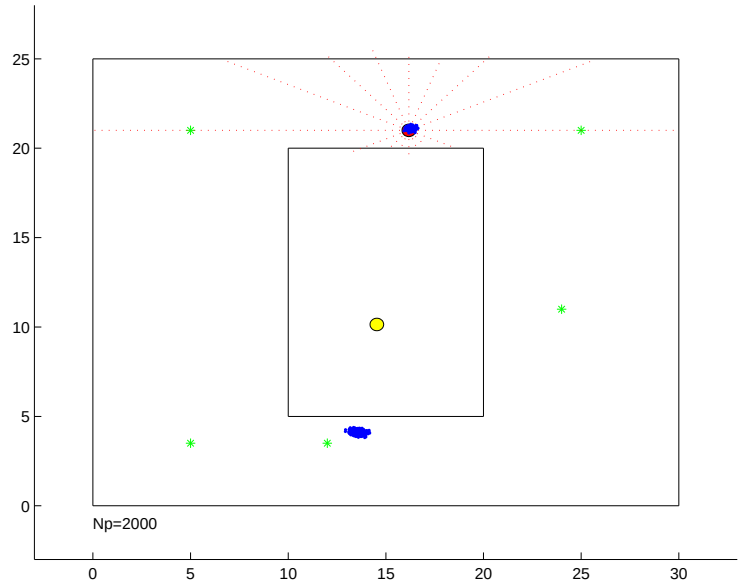
I pesi dei due gruppi di particelle che si vengono a creare, hanno all'incirca lo stesso valore, e lo mantengono per tutta la simulazione. Può accadere comunque che a volte l'algoritmo riesca, nonostante ciò, a localizzare il robot, ma ci sono anche situazioni in cui ciò non accade. Questo dipende essenzialmente dal carattere probabilistico della tecnica utilizzata e l'eventuale risoluzione del problema si deve esclusivamente addebitare ad una fortunata distribuzione iniziale delle particelle nello spazio di stato.

5.5 Confronto tra i metodi proposti

In questo paragrafo vengono confrontate le prestazioni degli algoritmi di localizzazione realizzati, cercando di risolvere un problema di localizzazione globale. A tal proposito, si utilizza la mappa riportata in Figura (5.15). Per poter dire qual'è il filtro che garantisce le migliori prestazioni, non è stata svolta una sola simulazione, bensì ne sono state eseguite 100 per ogni algoritmo, in modo tale da avere a di-



(a)



(b)

Figura 5.14: Localizzazione globale con *filtro particellare* in ambiente simmetrico. In rosso la posizione del robot, in giallo la stima dello stato all'istante (a) $k = 5$ e (b) $k = 15$.

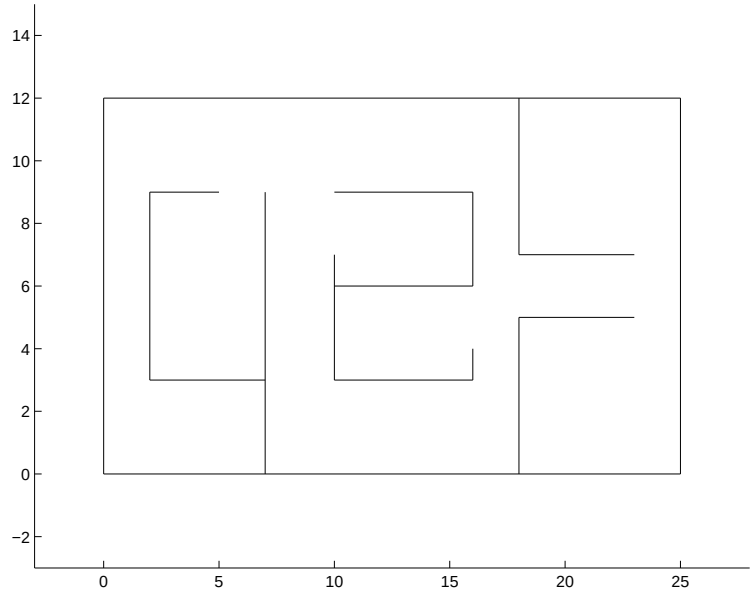


Figura 5.15: Mappa utilizzata per confrontare le prestazioni degli algoritmi realizzati.

sposizione un discreto numero di risultati sulla base dei quali confrontare i metodi utilizzati. Al termine di ogni singola simulazione j, si determina l'indice di errore $\mathcal{E}_j = \sqrt{\frac{1}{N_{passi}} \sum_k [(x_k - \widehat{x}_k)^2 + (y_k - \widehat{y}_k)^2]}$, che viene memorizzato per ottenere, dopo 100 esecuzioni dell'algoritmo, il valor medio $\mathcal{E}_{medio} = \frac{1}{100} \sum_{j=1}^{100} \mathcal{E}_j$. Per ogni simulazione sono mantenuti i dati di default; l'unica differenza consiste nel porre per il filtro particellare e per la sua versione con filtro di Kalman esteso $N_p = 1000$, mentre per il filtro particellare adattativo $N_p = 3000$.

I dati ottenuti vengono analizzati sotto diversi punti di vista. Per prima cosa si fa riferimento alla Tabella 5.3, nella quale sono riportati gli indici di errore medi $\mathcal{E}_{medio}$, calcolati così come descritti in precedenza, e il tempo di esecuzione medio dell'algoritmo, sulla base di 100 simulazioni. Il computer utilizzato per le simulazioni è un *Intel Celeron M*, processore $1.30GHz$, $736MB$ di RAM. È interessante notare come, secondo i dati ottenuti, è il filtro particellare standard a garantire i migliori risultati.

	$\mathcal{E}_{medio}$	$t_{calcolo}$
PF	2.172m	4':33"
EKPF	2.452m	4':48"
APF	2.221m	3':57"

Tabella 5.3: Indice di errore medio degli algoritmi di localizzazione realizzati.

Questo è vero, dal momento che l'indice $\mathcal{E}_{medio}$ è il più piccolo calcolato. Si vuole però porre l'attenzione su quanto riportato nella Tabella 5.4, nella quale sono espressi in maniera più dettagliata altri dati provenienti dalle simulazioni effettuate.

	$k \leq 10$	$10 < k \leq 20$	$20 < k \leq 30$	$k > 30$	% fallimento
PF	48	26	3	16	7%
EKPF	80	0	0	0	20%
APF	76	8	0	4	12%

Tabella 5.4: Numero di simulazioni in corrispondenza delle quali si ottiene una differenza tra la reale posizione del robot e la sua stima minore di $0.5m$ in k passi.

La tabella indica quante volte dopo k passi dall'inizio della simulazione l'algoritmo di localizzazione corrispondente riesce ad ottenere una stima della posizione del robot che dista dalla reale meno di $0.5m$. Quindi quello che si deduce abbastanza agevolmente è che il filtro particellare adattativo ben 76 volte su 100 riesce a risolvere il problema della localizzazione nei primi dieci passi, mentre l'*EKPF* addirittura 80 volte. Poco meno della metà delle simulazioni svolte (48) con il filtro particellare, invece, hanno ottenuto lo stesso risultato. Però, questa capacità di risolvere così velocemente il problema, ha lo svantaggio di presentare una percentuale di fallimento maggiore per l'*APF* e per l'*EKPF* (rispettivamente pari al 12% e al 20%) rispetto al *PF* (8%). Ed è proprio questa maggiore probabilità di insuccesso che, sulle 100 simulazioni eseguite, ha influenzato l'indice di errore medio, in modo tale che risultassero $\mathcal{E}^{APF}_{medio}$ e $\mathcal{E}^{EKPF}_{medio}$ maggiori di $\mathcal{E}^{PF}_{medio}$. Se non si considerano le simulazioni non andate

a buon fine per l'algoritmo basato su filtro particellare, si ha che il valore dell'indice di errore risulta uguale a $\mathcal{E}_{medio}^{APF} = 1.327m$. Scartando in maniera analoga i risultati delle 7 simulazioni che hanno impiegato il filtro particellare senza però riuscire a localizzare in modo corretto la posizione del robot, si ha $\mathcal{E}_{medio}^{PF} = 1.736m$. Analogamente $\mathcal{E}_{medio}^{EKPF} = 1.424m$.

Con questo si è voluto evidenziare che il filtro particellare adattativo effettivamente fornisce una stima migliore rispetto agli altri algoritmi realizzati. In particolare, rispetto al *PF*, una buona stima della posizione del robot viene ricavata in un numero di iterazioni in media minore, a discapito però di una maggiore, seppur di poco, percentuale di fallimento. Ciò che invece accade rispetto all'*EKPF*, è che l'*APF* ottiene la stima desiderata generalmente in un numero di passi superiore, mentre l'*EKPF* ha problemi nel mantenere la posizione ricavata, soprattutto in ambienti abbastanza simmetrici, portando al fallimento della simulazione. In Figura (5.16) si riporta l'andamento dell'errore per le simulazioni svolte.

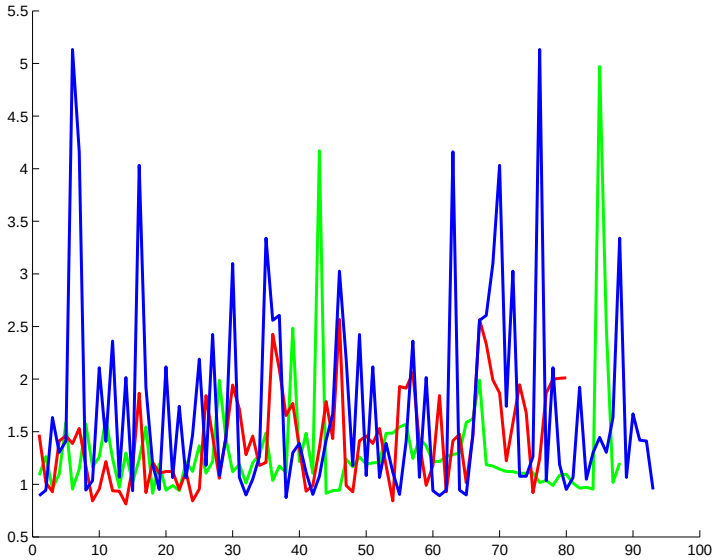


Figura 5.16: Andamento dell'errore per le simulazioni svolte; in blu l'errore del *PF*, in rosso dell'*EKPF* e in verde dell'*APF*.

5.6 Esempio al variare del numero di particelle

In questa ultima simulazione, si vuole evidenziare la migliore stima dello stato che si ottiene se, per uno stesso esempio, si adopera un numero di particelle sempre più grande, in modo tale da avere una distribuzione iniziale il più possibile uniforme. Sono state eseguite 5 simulazioni per ogni valore di N_p e l'indice d'errore $\mathcal{E}_{j,i}$ si riferisce alla j-esima simulazione dell'i-esimo valore di N_p. Si è cominciato con un numero di particelle $N_{p1} = 50$, valore che si è rivelato essere insufficiente per risolvere la localizzazione globale, con addirittura il 100% dei fallimenti. Piano piano si è aumentato il numero impiegato fino a giungere a $N_{p5} = 1000$, che come ci si aspettava ha fornito i risultati migliori. La Tabella 5.5 contiene tutte le informazioni ricavate dalle simulazioni svolte; si noti come l'errore medio $\mathcal{E}_{m,i}$ diminuisca all'aumentare del numero di particelle N_p. Con il tratto "/" si sono indicate le simulazioni fallite.

	$N_{p1} = 50$	$N_{p2} = 100$	$N_{p3} = 300$	$N_{p4} = 500$	$N_{p5} = 1000$
sim.1	/	$\mathcal{E}_{1,2} = 4.03$	/	$\mathcal{E}_{1,4} = 2.07$	$\mathcal{E}_{1,5} = 1.99$
sim.2	/	/	$\mathcal{E}_{2,3} = 2.86$	/	$\mathcal{E}_{2,5} = 2.31$
sim.3	/	/	/	/	$\mathcal{E}_{3,5} = 2.44$
sim.4	/	$\mathcal{E}_{4,2} = 4.38$	$\mathcal{E}_{4,3} = 5.09$	$\mathcal{E}_{4,4} = 2.83$	$\mathcal{E}_{4,5} = 1.94$
sim.5	/	/	$\mathcal{E}_{5,3} = 2.28$	$\mathcal{E}_{5,4} = 2.5$	$\mathcal{E}_{5,5} = 2.12$
media	/	$\mathcal{E}_{m,2} = 4.21$	$\mathcal{E}_{m,3} = 3.41$	$\mathcal{E}_{m,4} = 2.47$	$\mathcal{E}_{m,5} = 2.16$

Tabella 5.5: Indici d'errore ricavati da 5 simulazioni al variare del numero di particelle N_p.

Capitolo 6

Conclusioni e sviluppi futuri

Durante lo svolgimento di questa tesi, si è cercato di risolvere il problema della localizzazione di un robot mobile in un ambiente noto mediante l'utilizzo dei filtri particellari. Tale tecnica di filtraggio può essere impiegata in problemi di stima non lineari e non gaussiani, per i quali le approssimazioni, quali quelle adottate dal filtro di Kalman, non sono ipotizzabili.

Si è riusciti a realizzare un algoritmo che è in grado di stimare la posizione e l'orientamento del robot mediante i soli dati provenienti dai sensori di cui è dotato. In seguito si sono implementate due varianti di tale algoritmo: la prima risolve il problema della localizzazione mediante un filtro particellare che utilizza il filtro di Kalman esteso per la generazione della distribuzione *proposal*, mentre la seconda mediante il filtro particellare adattativo. L'intero programma è stato realizzato con il software di calcolo *Matlab 6.5*. I risultati ottenuti si possono ritenere soddisfacenti, dal momento che le tecniche adottate risolvono il problema, come ampiamente dimostrato nel Capitolo 5, seppur non tutte in maniera affidabile ed accurata. Dalle simulazioni svolte, si sono potuti evidenziare pregi e difetti degli algoritmi di localizzazione impiegati. Ad esempio, il filtro particellare generalmente fornisce i migliori risultati, in particolar modo grazie ad una buona accuratezza e ad un grado di affidabilità piuttosto elevato.

Il filtro particellare adattativo, oltre ad offrire un tempo di calcolo minore rispetto alle altre tecniche utilizzate, si avvicina molto alle prestazioni del filtro particellare, presentando l'unico inconveniente di una percentuale di fallimento leggermente superiore. Il filtro particellare con filtro di Kalman esteso è quello che ha riportato in generale i risultati peggiori. Possiede la maggiore probabilità di insuccesso in problemi di localizzazione globale e pertanto il suo impiego si è rivelato il meno interessante e soddisfacente. Nonostante ciò, l'*EKPF* è il filtro che riesce a risolvere il problema della localizzazione globale prima degli altri due, anche se si riscontra un'elevata difficoltà nel mantenere la corretta stima del robot durante l'intero percorso. Tali svantaggi potrebbero essere dovuti al mancato utilizzo di una soglia sul termine di innovazione e sarebbe interessante aggiungere questo dato per verificare il comportamento dell'algoritmo. Inoltre, come detto nel Capitolo 4, anche l'assunzione gaussiana sulla forma della distribuzione di probabilità a posteriori e le approssimazioni inevitabilmente introdotte con la linearizzazione potrebbero essere la causa di tali inconvenienti.

Nonostante il lavoro svolto abbia portato ai risultati sperati, gli algoritmi realizzati potrebbero subire alcune modifiche per migliorare le prestazioni. Per prima cosa, uno degli aspetti che potrebbe essere maggiormente approfondito e trattato in maniera diversa, è la scelta della *proposal distribution*. Dal momento che i filtri particellari si basano sull'*importance sampling*, tale distribuzione ha un ruolo fondamentale sulle prestazioni dell'algoritmo, poiché viene impiegata nella fase di aggiornamento dei pesi, influenzando dunque inevitabilmente anche il ricampionamento, e da essa vengono estratte le particelle ad ogni iterazione. A tal proposito, in letteratura si trovano numerosi studi e trattazioni, che rendono l'idea di quanto la questione non sia di semplice risoluzione. Non esiste una metodologia che possa essere definita ottima, ma diverse possibilità di scelta da impiegare a seconda delle situazioni che si presentano,

come descritto nel Capitolo 2.

Un ulteriore sviluppo degli algoritmi ricavati potrebbe consistere nell'utilizzo del filtro di Kalman *unscented* [29] nella generazione della distribuzione *proposal*, così come viene proposto in [25]. Tale filtro fornisce stime più accurate del filtro di Kalman esteso e con esso si potrebbe ricavare una migliore stima dello stato.

Un ulteriore sviluppo che dovrebbe essere preso in considerazione riguarda la fase di ricampionamento e in particolare la condizione che ne regola l'esecuzione. Generalmente questa fase non viene effettuata ad ogni istante di tempo $t = k$, ma solo quando il numero effettivo di particelle $\widehat{N}_{eff}$ (vedi (2.25)) scende al di sotto di una predeterminata soglia N_T, funzione di N_p. Ciò che potrebbe migliorare il filtro particellare consiste nel rendere variabile tale soglia. In tal modo si possono eliminare non solo situazioni in cui la fase di resampling viene saltuariamente eseguita a causa di un valore N_T troppo piccolo, ma anche i casi opposti in cui si ha un suo eccessivo impiego poiché la soglia è molto elevata. Riuscire a fare il ricampionamento solo quando strettamente necessario consentirebbe pertanto di evitare di propagare particelle aventi un peso basso, nel caso di soglia troppo piccola, e di ridurre la complessità dell'algoritmo, quando il valore di N_T è talmente grande da garantire sempre l'esecuzione di tale fase.

Appendice A

Nella prima tabella vengono mostrati i valori dei quantili della distribuzione normale standard (media uguale a 0 e scarto quadratico medio uguale a 1).

Ordine del Quantile	Quantile	Ordine del Quantile	Quantile
0.5	0.	0.75	0.67449
0.51	0.0250689	0.76	0.706303
0.52	0.0501536	0.77	0.738847
0.53	0.0752699	0.78	0.772193
0.54	0.100434	0.79	0.806421
0.55	0.125661	0.8	0.841621
0.56	0.150969	0.81	0.877896
0.57	0.176374	0.82	0.915365
0.58	0.201893	0.83	0.954165
0.59	0.227545	0.84	0.994458
0.6	0.253347	0.85	1.03643
0.61	0.279319	0.86	1.08032
0.62	0.305481	0.87	1.12639
0.63	0.331853	0.88	1.17499
0.64	0.358459	0.89	1.22653
0.65	0.38532	0.9	1.28155
0.66	0.412463	0.91	1.34076
0.67	0.439913	0.92	1.40507
0.68	0.467699	0.93	1.47579
0.69	0.49585	0.94	1.55477
0.7	0.524401	0.95	1.64485
0.71	0.553385	0.96	1.75069
0.72	0.582842	0.97	1.88079
0.73	0.612813	0.98	2.05375
0.74	0.643345	0.99	2.32635

La seconda tabella mostra con maggiore dettaglio i quantili estremi, di ordine compreso tra 0.95 e 0.999.

Ordine del Quantile	Quantile	Ordine del Quantile	Quantile
0.95	1.64485	0.975	1.95996
0.951	1.65463	0.976	1.97737
0.952	1.66456	0.977	1.99539
0.953	1.67466	0.978	2.01409
0.954	1.68494	0.979	2.03352
0.955	1.6954	0.98	2.05375
0.956	1.70604	0.981	2.07485
0.957	1.71689	0.982	2.09693
0.958	1.72793	0.983	2.12007
0.959	1.7392	0.984	2.14441
0.96	1.75069	0.985	2.17009
0.961	1.76241	0.986	2.19729
0.962	1.77438	0.987	2.22621
0.963	1.78661	0.988	2.25713
0.964	1.79912	0.989	2.29037
0.965	1.81191	0.99	2.32635
0.966	1.82501	0.991	2.36562
0.967	1.83842	0.992	2.40892
0.968	1.85218	0.993	2.45726
0.969	1.8663	0.994	2.51214
0.97	1.88079	0.995	2.57583
0.971	1.8957	0.996	2.65207
0.972	1.91104	0.997	2.74778
0.973	1.92684	0.998	2.87816
0.974	1.94313	0.999	3.09023

Elenco delle figure

Bibliografia

[1] F. Pascucci G. Ulivi A. Gasparri, S. Panzieri. Pose recovery for a mobile manipulator using a particle filter. In *Proc. of 14th IEEE Mediterranean Symposium on New Directions in Control and Automation*, pages 687–692, Ancona, Italy, 2006.

[2] W.H. Wong A. Kong, J.S. Liu. Sequential imputations and bayesian missing data problems. *J. Amer. Statist. Assoc.*, 89:278–288, 1994.

[3] M. N. Andersen and R. Ø. Andersen. Filtering in hybrid dynamic bayesian networks. Master's thesis, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2003. Supervisor: Lars Kai Hansen.

[4] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[5] W. Burgard, A. Derr, D. Fox, and A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: the Dynamic Markov Localization approach. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[6] R. Chen and J. Liu. Mixture kalman filters. In *J. R. Statist. Soc. B62 pages 493-508.*, 2000.

[7] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. Technical report, McMaster University, 2003.

[8] W.Burgard D.Fox, S.Thrun and F.Dellaert. Particle filters for mobile robot localization. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, New York, 2001. Springer.

[9] D. Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[10] D. Fox. Adapting the sample size in particle filters through kld-sampling. In *International Journal of Robotics Research (IJRR), 22, 2003.*, 2003.

[11] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.

[12] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

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

[14] A. Gelb. *Applied Optimal Estimation.* MIT Press, 1974.

[15] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140:107–113, 1993.

[16] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[17] J.-on Lee M.T. Lim J. Woo, Y.J. Kim. Localization of mobile robot using particle filter. *SICE-ICASE International Joint Conference 2006*, October 2006.

[18] F. Gustafsson J.D. Hol, T.B. Schön. On resampling algorithms for particle filters. In *Nonlinear Statistical Signal Processing Workshop*, Cambridge, United Kingdom, sept 2006.

[19] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

[20] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering 82 (Series D): 35-45.*, 1960.

[21] C. Kwok, D. Fox, and M. Meila. Adaptive real-time particle filters for robot localization. In *Proc. of the IEEE International Conference on Robotics & Automation*, 2003.

[22] J. Liu, R. Chen, and T. Logvinenko. A theoretical framework for sequential importance sampling and resampling. Technical report, Technical report, Stanford University, Department of Statistics, 2000.

[23] L. Marchetti, G. Grisetti, and L. Iocchi. A comparative analysis of particle filter based localization methods. In *Proc. of RoboCup Symposium*, 2006.

[24] E. Moulines R. Douc, O. Cappé. Comparison of resampling schemes for particle filtering. *CoRR*, 2008. informal publication.

[25] N. de Freitas R. van der Merwe, A. Doucet and E. A. Wan. The unscented particle filter. In *NIPS*, pages 584–590, 2000.

[26] Ioannis M. Rekleitis. A particle filter tutorial for mobile robot localization. *Technical Report TR-CIM-04-02*, February 2004.

[27] S.I. Roumeliotis and G.A. Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *ICRA*, pages 2985–2992, 2000.

[28] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

[29] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. pages 153–158, 2000.

[30] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.