UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA
DELL'AUTOMAZIONE

A.A. 2007/2008

Tesi di Laurea

LOCALIZZAZIONE DI ROBOT MOBILI IN AMBIENTE DINAMICO

RELATORE

Prof. Francesco Martinelli

CANDIDATO

Antonio De Paola

Indice

Capitolo 1

Introduzione

La robotica si occupa dello studio di macchine che possano sostituire l'uomo nell'esecuzione di un compito, sia esso fisico o decisionale. Tali macchine sono dotate di opportuni dispositivi atti a percepire l'ambiente circostante (sensori) e a interagire con esso (attuatori), al fine di eseguire un compito specifico. La capacità di connettere in maniera intelligente azione e percezione è affidata a un sistema di governo che sia in grado di comandare l'esecuzione dell'azione nel rispetto degli obiettivi imposti. Il presente lavoro si colloca nell'ambito della robotica mobile, quella branca della robotica che si occupa di tutte le problematiche relative a robot mobili, macchine dotate di un sistema di locomozione che consente loro di spostarsi liberamente nell'ambiente in cui devono operare. Tale tesi affronta, con un approccio probabilistico, un problema di localizzazione, ossia la stima nel tempo della posizione di un robot in movimento, elemento fondamentale per consentire al robot di svolgere il lavoro assegnatogli. Questo problema viene trattato nel caso in cui sia disponibile la mappa dell'ambiente in cui si muove il robot e tale ambiente sia dinamico, ossia presenti al suo interno oggetti (fissi e in movimento) non descritti dalla mappa. Tale localizzazione può essere sia locale, qualora si conosca con esattezza la posizione iniziale del robot, sia globale, nel caso la posizione iniziale del robot risulti sconosciuta. L'approccio utilizzato per la

realizzazione di tale tesi è stato prettamente teorico-simulativo: la validità ed efficienza dell'algoritmo realizzato per la localizzazione sono state verificate mediante una simulazione che, se da un lato fornisce risultati che si discostano inevitabilmente da quelli reali, d'altro canto consente di testare facilmente quanto realizzato in ambienti e condizioni molto varie. Il lavoro svolto si articola in un'iniziale descrizione degli strumenti matematici utilizzati, seguita dalla presentazione del modello usato per descrivere il robot, i suoi sensori e la mappa; l'integrazione di questi due elementi porta all'implementazione dell'algoritmo di localizzazione e a una sua successiva ottimizzazione dei parametri, seguita da una presentazione dei risultati ottenuti. Viene inoltre allegato, in appendice, il listato del programma che realizza in ambiente MATLAB quanto descritto ai punti precedenti.

Capitolo 2

Filtri e stima dello stato

2.1 Filtro di Bayes

Un filtro, nell'ambito di questo lavoro, è uno stimatore ricorsivo: esso riceve in input $bel(x_{t-1})$,z_t e u_t (rispettivamente stima dello stato al tempo t-1, misura al tempo t e ultimo controllo effettuato), restituendo $bel(x_t)$, ossia la stima dello stato al tempo t. Il filtro di Bayes rappresenta la sua forma più generale e quella che garantisce il risultato migliore anche se, come si vedrà in seguito, esso risulta difficilmente applicabile nella pratica. Il filtro opera sostanzialmente in due fasi: esso inizialmente calcola $\overline{bel(x_t)}$, la nuova densità di probabilità in seguito al controllo, nella quale la probabilità relativa al singolo stato x_t è l'integrale del prodotto tra la densità di probabilità $bel(x_{t-1})$ e la probabilità che il controllo u_t causi la transizione da x_{t-1} a x_t:

$$\overline{p}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx \tag{2.1.1}$$

La densità così ottenuta viene ulteriormente modificata: la probabilità del singolo stato viene moltiplicata per la probabilità che, trovandosi in tale stato, venga rilevata la misura z_t. Tale prodotto viene quindi normalizzato mediante la costante η, opportunamente calcolata per far sì che la stima dello stato $bel(x_t)$ abbia integrale unitario

e sia quindi una densità di probabilità:

$$p(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t) \tag{2.1.2}$$

Il filtro procede quindi iterativamente nella stima dello stato al tempo $t+1$ data la stima al tempo t; è evidente come, per il funzionamento corretto del filtro, sia necessario fornire $bel(x_0)$, ossia la stima dello stato al tempo $t = 0$.

Nel caso di spazio di stato continuo, quale sarà quello trattato nel seguito di questo lavoro, il filtro di Bayes richiede di effettuare al generico tempo t, per ciascuno degli infiniti stati x_t, un integrale (2.1.1) e un prodotto (2.1.2). Data l'impossibilità per qualsiasi elaboratore di effettuare un numero infinito di operazioni in tempo finito, appare evidente la necessità di adottare, nel seguito del lavoro, uno strumento matematico diverso.

2.2 Particle Filter

Una possibile soluzione per il problema delle infinite operazioni, evidenziato al paragrafo precedente, consiste nei filtri non parametrici, i quali descrivono la stima dello stato mediante un numero finito di parametri. Il filtro non parametrico utilizzato nel seguito di questo lavoro di tesi è il particle filter, che descrive $bel(x_t)$ con un insieme di campioni casuali da essa generati; in sostanza esso descrive una certa densità di probabilità mediante un insieme di numeri casuali generati secondo la densità stessa. Ovviamente tale rappresentazione risulta un'approssimazione della stima, d'altro canto essa consente di abbattere drasticamente i tempi di calcolo. Indicando con $x_t^{[i]}$ un campione della stima dello stato, ossia un'ipotesi di come possa essere realmente lo stato al tempo t, si ha

$$X_t = x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \tag{2.2.1}$$

con M numero dei campioni generati ad ogni passo e X_t l'insieme degli stessi.

Il particle filter, quindi, approssima $bel(x_t)$ con l'insieme di campioni X_t. Idealmente, la possibilità che l'ipotesi di stato x_t sia inclusa in X_t è proporzionale a quella calcolata con il filtro di Bayes:

$$x_t^{[m]} \sim p(x_t|z_{1:t}, u_{1:t}) \tag{2.2.2}$$

Risulta evidente, quindi, che maggiore è la densità di campioni in una regione dello spazio di stato, maggiore è la probabilità che il vero stato si trovi in questa regione. In realtà ciò è verificato esclusivamente per M che tende a infinito, tuttavia l'errore è trascurabile per un numero sufficiente elevato di campioni ($M \geq 100$).
Il particle filter opera ricorsivamente, calcolando la stima $bel(x_t)$ a partire da quella calcolata al passo precedente $bel(x_{t-1})$. Nel dettaglio, ciò avviene nei seguenti passaggi:

- Innanzitutto esso genera un ipotetico stato $x_t^{[m]}$ a partire dal campione $x_{t-1}^{[m]}$ e dal controllo u_t. Per fare questo è importante sottolineare come sia necessario definire una tecnica di sampling che consenta di generare campioni a partire da $p(x_t|u_t, x_{t-1})$. L'insieme dei campioni generati in questo modo, a partire da ciascuno dei campioni presenti, viene detto $\overline{bel}(x_t)$

- Il passo successivo consiste nel calcolo del cosiddetto *importance factor* $\omega_t^{[m]}$ per ciascuno dei campioni $x_t^{[m]}$. L'importance factor è così definito:

$$\omega_t^{[m]} = p(z_t|x_t^{[m]}) \tag{2.2.3}$$

- L'ultimo passo consiste nel resampling, ossia nel reistanziare un numero M di campioni a partire da $\overline{bel}(x_t)$, dove ciascun campione $x_t^{[m]}$ ha probabilitá $\frac{\omega_t^{[m]}}{\sum_i \omega_t^{[i]}}$

di essere istanziato. Tale passo risulta di fondamentale importanza: esso ricrea un insieme di M campioni, distribuiti (approssimativamente) secondo la densità di probabilità

$$bel(x_t) = \eta p(z_t|x_t^m)\overline{bel}(x_t) \tag{2.2.4}$$

Di fatto tale processo fa sì che in X_t alcuni dei campioni coincidano ma soprattutto esso elimina tutti quei campioni in $\overline{bel}(x_t)$ che avevano un basso *importance factor*, permettendo quindi di concentrare le risorse computazionali nelle regioni dello spazio di stato più rilevanti.

Capitolo 3

Modellazione del sistema

3.1 Odometria

Coerentemente con l'approccio probabilistico che si è deciso di adottare, si rende necessario un modello dello stesso tipo per descrivere la cinematica del robot e permettere quindi di implementarla adeguatamente nel filtro. La posizione del robot verrà descritta, assumendo di operare in uno spazio bidimensionale, come una terna di valori $[x, y, \theta]$, con x e y coordinate cartesiane nel sistema di riferimento scelto e θ angolo di orientamento del robot rispetto all'asse delle ascisse.

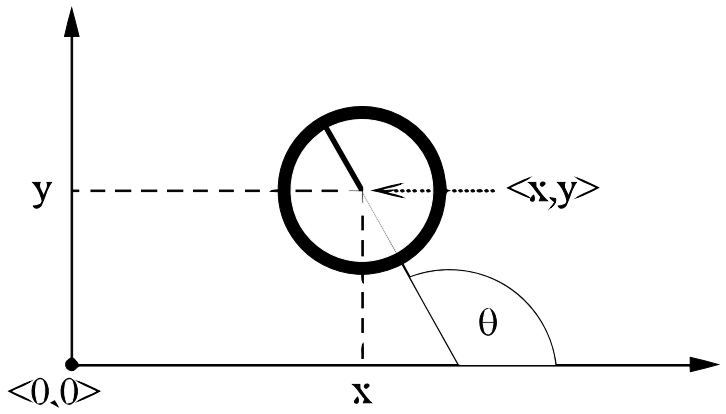


Figura 3.1: Sistema di coordinate adottato per il robot

Il modello probabilistico adottato dev'essere in grado di calcolare $p(x_t|u_t, x_{t-1})$,

con x_k e u_k rispettivamente posizione del robot e controllo effettuato al tempo k. Nel caso in esame il controllo corrisponderà allo spostamento relativo effettuato dal robot secondo le misure odometriche, ottenute integrando le informazioni degli encoder posti sulle ruote. In un generico spostamento effettuato al tempo t, alle posizioni x_{t-1} e x_t nelle coordinate del sistema assoluto di riferimento corrispondono le coordinate $\overline{x}_{t-1}$ e $\overline{x}_t$ nel sistema di riferimento interno del robot, la cui relazione con le coordinate globali è sconosciuta. Ciò è causato da errori nella misurazione degli encoder e dallo slittamento delle ruote sulla superficie su cui si muove il robot, fenomeni che non sono mai completamente eliminabili. Le coordinate nel riferimento interno del robot, che sono in generale le uniche note, possono tuttavia essere utilizzate per calcolare quelle globali, assumendo che la differenza tra $\overline{x}_t$ e $\overline{x}_{t-1}$ sia una buona stima della differenza tra x_t e x_{t-1}. Per estrarre gli spostamenti realmente effettuati dal robot e quindi l'odometria, si assume che il robot in esame si sposti grazie a delle ruote e sia in grado di effettuare rotazioni intorno alla sua posizione attuale e traslazioni nella direzione in cui è orientato. Data quindi una generica coppia di coordinate di partenza e arrivo del robot, si scompone il relativo spostamento in una rotazione, seguita da una traslazione e da un'altra rotazione. La prima rotazione serve a orientare il robot in modo che esso si direzioni verso il punto d'arrivo, la traslazione permette il raggiungimento di tale punto mentre la seconda rotazione consente al robot di assumere l'orientamento finale.

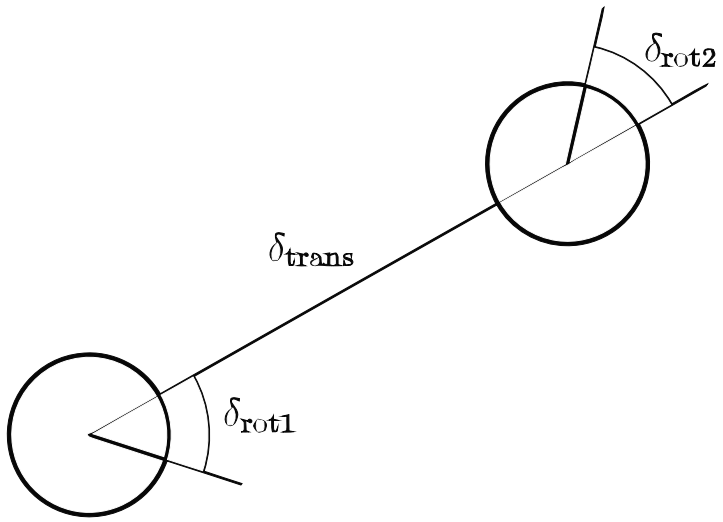


Figura 3.2: Scomposizione in tre spostamenti elementari

Ciascuno di tali spostamenti elementari nel piano viene assunto, nel modello in esame, affetto da rumore di tipo gaussiano. È evidente che, data una posizione iniziale, a ogni posizione d'arrivo del robot corrisponda una ed una sola terna di valori $(\delta^*_{rot1}, \delta^*_{trasl}, \delta^*_{rot2})$ che indichi lo spostamento del robot tra i due punti. Una volta ottenute le informazioni odometriche del robot $(\delta_{rot1}, \delta_{trasl}, \delta_{rot2})$, risulta $p(x_t|u_t, x_{t-1}) = p(err) = p(err_1) \cdot p(err_2) \cdot p(err_3)$ con err_i l'errore tale che

$$(\delta^*_{rot1}, \delta^*_{trasl}, \delta^*_{rot2}) = (err_1, err_2, err_3) + (\delta_{rot1}, \delta_{trasl}, \delta_{rot2}) \tag{3.1.1}$$

3.2 Mappa e misure

L'implementazione di un algoritmo di localizzazione svolta in tale lavoro richiede una modellazione della mappa che soddisfi due requisiti: una versatilità tale da permettere un'adeguata rappresentazione degli ambienti reali e una struttura che ne renda possibile una facile integrazione con il modello odometrico e delle misure. A questo proposito si è scelto di rappresentare la mappa come una spezzata chiusa, indicando le

coordinate dei punti d'intersezione tra i segmenti consecutivi nel sistema di riferimento assoluto; un semplice algoritmo realizza una rappresentazione equivalente, descrivendo i vari segmenti mediante il loro coefficiente angolare e delimitazioni sull'asse delle ascisse o delle ordinate. Per quanto riguarda i sensori, si è deciso di adottare un modello a raggio: il robot viene schematizzato planarmente come una circonferenza da cui si dipartono, equispaziati angolarmente, un certo numero di semirette. Appare immediato che il calcolo delle misure ideali, in tale modello, venga effettuato semplicemente individuando, per ciascun sensore, le intersezioni del relativo raggio con i segmenti che compongono la mappa. Calcolata la distanza di ciascuna di queste intersezioni dalla posizione del robot, il minimo rappresenterà la misura ideale cercata. Poichè è prevista anche una simulazione che verifichi le prestazioni dell'algoritmo di localizzazione realizzato, è necessario creare un algoritmo che simuli la misurazione reale effettuata dal robot e per fare ciò bisogna innanzitutto modellare oggetti estranei non descritti dalla mappa. In questo lavoro si sono supposte presenti nell'ambiente esclusivamente persone, ma il modello è facilmente estensibile a qualsiasi altro tipo di oggetto. Una generica persona, o per meglio dire la sua sezione sul piano passante per i sensori del robot, viene schematizzata come un rettangolo di dimensione 7.5x15 la cui posizione è descritta da una terna di valori $[x, y, \theta]$ di tipo analogo a quella utilizzata precedentemente per il robot. Un'apposita funzione scompone il rettangolo in quattro segmenti, ciascuno dei quali definito da un coefficiente angolare e dalle sue delimitazioni sull'asse delle ascisse. Le misure reali verranno ottenute in modo simile a quelle ideali, ma in questo caso l'intersezione dei raggi dei sensori e quindi le relative distanze dal robot andranno calcolate non solo rispetto ai segmenti della mappa, ma anche rispetto a quelli che schematizzano tutte le persone presenti nell'ambiente. Bisogna, inoltre, introdurre altre due modifiche rispetto al calcolo delle misure ideali:

le misure maggiori del raggio massimo dei sensori andranno troncate a tale valore e inoltre verrà aggiunto un rumore gaussiano per simulare quello dei sensori.

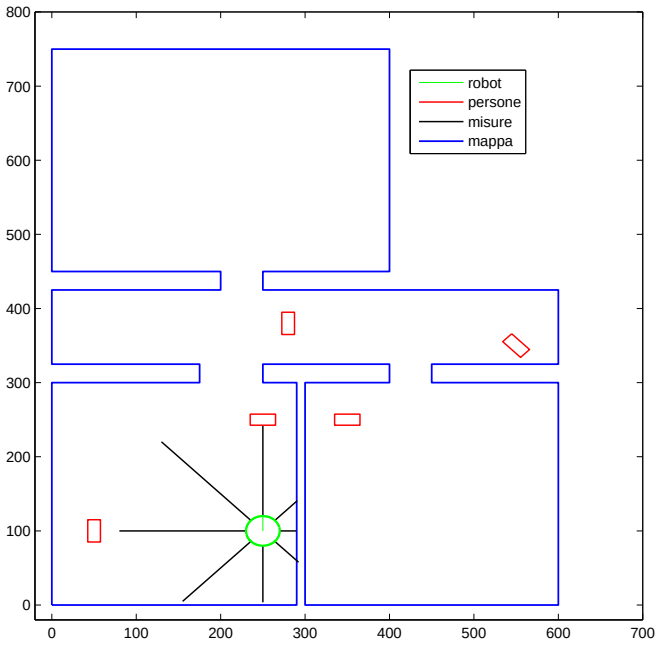


Figura 3.3: Modellazione del robot e dell'ambiente

Capitolo 4

Localizzazione

4.1 Applicazione del particle filter

Una volta modellati adeguatamente odometria, sensore e mappa, è possibile approcciarsi al problema della localizzazione robotica cercando di implementare una versione adeguata del particle filter. Il primo passo consiste nel definire $bel(x_t)$, ossia la stima dello stato al generico tempo t. Poichè nel caso in esame lo stato del robot è la sua posizione, definita mediante la terna (x, y, θ) in un sistema di riferimento assoluto, $bel(x_t)$ consisterà in un insieme di M campioni, ciascuno istanza di una possibile posizione. La probabilità che lo stato reale si trovi in un certo intervallo dello spazio di stato R^3 sarà proporzionale al numero di campioni presenti in esso. Una volta definita la stima dello stato, il passo successivo consiste nel modellare lo spostamento del robot mediante un sampling dei campioni: il generico spostamento viene descritto mediante le posizioni iniziali e finali, dalle quali si calcolano i tre spostamenti elementari descritti al Cap. 3.1 ($\delta_{rot1}, \delta_{trans}, \delta_{rot2}$).

Il nuovo insieme di campioni, che costituirà $\overline{bel}(x_t)$ è ottenuto imponendo i tre spostamenti elementari a ciascuno degli elementi di $bel(x_{t-1})$, ipotizzando che tali spostamenti siano affetti da errore, parametrizzato dai parametri α.

Il risultato di questo primo sampling, nel caso la posizione iniziale sia univocamente

nota, è rappresentato in figura:

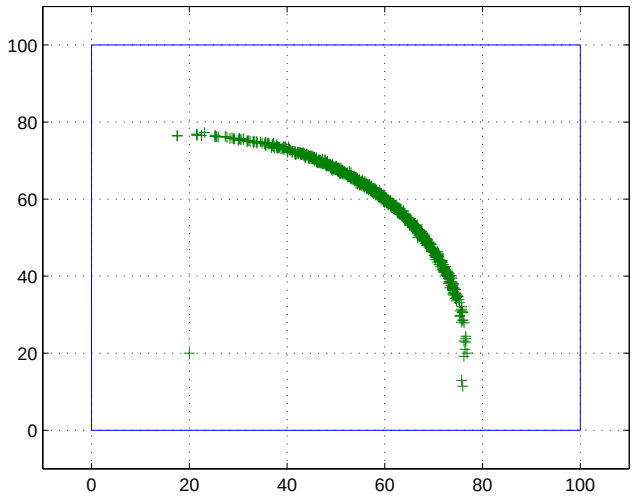


Figura 4.1: Sampling dei campioni con odometria

Appare evidente che, se ci si basasse esclusivamente su tale modello per stimare la posizione del robot, si incorrerebbe in un errore sempre crescente, in quanto l'incertezza dovuta agli errori delle letture odometriche e allo slittamento delle ruote andrebbe inevitabilmente a sommarsi a quella dei precedenti. Risulta quindi indispensabile effettuare una nuova stima dello stato, ovvero un nuovo sampling dei campioni, basato sulle misure dei sensori presenti sul robot, il cui errore non è cumulativo nel corso dello spostamento. Si procede quindi ad assegnare a ciascuno dei campioni di $\overline{bel}(x_t)$ un "peso" di resampling che si sceglie uguale a $p(z_t|m, x_t)$. Per ottenere un resampling piuttosto accurato si pone il problema di realizzare una densità di probabilità che tenga conto di tutti gli elementi che causano un discostamento delle misure reali da quelle ideali. L'approccio utilizzato nella simulazione prevede quindi l'utilizzo di una combinazione lineare di quattro densità di probabilità, ciascuna delle quali modella una fonte d'errore:

1. **Rumore dei sensori**: nell'ipotesi in cui il sensore misuri effettivamente la distanza del robot dal più vicino elemento della mappa, la misura restituita è comunque soggetta ad errore causato da numerosi fattori, tra cui risoluzione limitata dei sensori e effetti atmosferici sul segnale di misura. Il rumore risultante viene quindi schematizzato come una gaussiana con media z_t^{k*} e deviazione standard σ_{hit}, dove z_t^{k*} rappresenta la misura ideale e σ_{hit} un parametro del modello di misura. Indicata tale densità di probabilità con p_{hit}, si ha:

$$p_{hit}(z_t^k|m, x_t) = \begin{cases} \eta\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) & \text{se } 0 \leq z_t^k \leq z_{max} \\ 0 & \text{altrimenti} \end{cases}$$

in tale densità di probabilità, z_t^k è calcolata attraverso il ray casting mentre $\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2)$ rappresenta una densità di probabilitànormale con media z_t^{k*} e varianza σ_{hit}^2:

$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) = \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{(z_t^k - z_t^{k*})^2}{2\sigma_{hit}^2}} \tag{4.1.1}$$

Poichè $\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2)$ viene troncata per $z > z_{max}$, affinchè $p_{hit}(z_t^k|m, x_t)$ sia una densità di probabilità e quindi $\int_{-\infty}^{+\infty} p_{hit}(z_t^k|m, x_t)dz_t^k = 1$ risulta necessario inserire il fattore di normalizzazione η:

$$\eta = \left(\int_0^{z_{max}} \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2)dz_t^k \right)^{-1} \tag{4.1.2}$$

2. **Oggetti imprevisti**: se all'interno dell'ambiente in cui si muove il robot sono presenti altri elementi non descritti dalla mappa (quali ad esempio persone in movimento), i sensori possono rilevare misure molto minori del valore previsto e causare quindi un funzionamento errato del filtro; per ovviare a questo inconveniente si schematizzano tali eventi come rumore dei sensori. Bisogna

innanzitutto osservare che la probabilità da parte dei sensori di intercettare elementi estranei e quindi restituire misurazioni errate diminuisce con la distanza. Dette z_1 e z_2 la distanza dal sensore di due persone che sono presenti con la stessa probabilità p nello spazio di misurazione del sensore e supposto $z_1 < z_2$, la probabilità di misurare z_1 è pari alla probabilità che sia presente la prima persona, mentre la probabilità di misurare z_2 è pari alla probabilità che sia presente la seconda persona e contemporaneamente non sia presente la prima. Matematicamente, tale situazione è descrivibile mediante una densità di probabilità esponenziale troncata in z_t^{k*}, in quanto eventuali oggetti presenti oltre elementi della mappa non verranno in alcun caso rilevati:

$$p_{short}(z_t^k|m, x_t) = \begin{cases} \eta\lambda_{short}e^{-\lambda_{short}z_t^k} & \text{se } 0 \leq z_t^k \leq z_t^{k*} \\ 0 & \text{altrimenti} \end{cases}$$

Come nel caso precedente, poichè l'esponenziale è limitato all'intervallo $[0; z_t^{k*}]$ e quindi:

$$\int_{-\infty}^{\infty} \lambda_{short}e^{-\lambda_{short}z_t^k} dz_t^k = 1 - e^{-\lambda_{short}z_t^{k*}} \tag{4.1.3}$$

è necessario inserire un fattore di normalizzazione η:

$$\eta = \frac{1}{1 - e^{-\lambda_{short}z_t^{k*}}} \tag{4.1.4}$$

3. **Misure mancate**: in alcuni casi può capitare che il sensore manchi completamente gli ostacoli, o perchè non riesce a percepirli per limiti insiti nel funzionamento fisico dello stesso o perchè essi si trovano oltre il raggio massimo r_{max} di misurazione. Poichè tale fenomemo nella pratica avviene piuttosto spesso, è necessario tenerne conto nel nostro modello, mediante una densità di probabilità discreta:

$$p_{max}(z_t^k|m, x_t) = \begin{cases} 1 & \text{se } z = z_{max} \\ 0 & \text{altrimenti} \end{cases}$$

4. **Misure casuali**: può capitare che nella pratica i sensori restituiscano valori completamente inspiegabili a causa di una serie di fattori che risultano difficilmente modellabili; per tenere conto comunque di tali misure e evitare che esse riducano la precisione del filtro, esse vengono modellate con una densità uniforme su tutto il raggio di misura del sensore:

$$p_{rand}(z_t^k|m, x_t) = \begin{cases} \frac{1}{z_{max}} & \text{se } 0 \leq z_t^k < z_{max} \\ 0 & \text{altrimenti} \end{cases}$$

Le quattro diverse densità descritte devono quindi essere combinate linearmente mediante dei parametri z_{hit},z_{short},z_{max} e z_{rand} tali che $z_{hit} + z_{short} + z_{max} + z_{rand} = 1$, in modo che la nuova funzione sia ancora una densità di probabilità. Otteniamo quindi la forma definitiva della $p(z_k^t|x_t, m)$:

$$p(z_t^k|x_t, m) = \begin{pmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{pmatrix}^T \cdot \begin{pmatrix} p_{hit}(z_t^k|x_t, m) \\ p_{short}(z_t^k|x_t, m) \\ p_{max}(z_t^k|x_t, m) \\ p_{rand}(z_t^k|x_t, m) \end{pmatrix}^T$$

A questo punto è possibile calcolare $\omega_t^{[m]} = p(z_k^t|x_t, m)$ per ciascuno dei campioni di $\overline{bel(x_t)}$, ossia il loro "peso" per il successivo sampling che genererà $bel(x_t)$. A tale riguardo, si pone il problema di effettuare tale sampling su una pseudodensità discreta che non risulta in alcun modo parametrizzabile. Nella simulazione in esame si è scelto un approccio che, a fronte dell'introduzione di un certo errore dovuto al determinismo dell'operazione, garantisce una maggiore velocità di calcolo: ogni campione c_i appartentente a $\overline{bel(x_t)}$ sarà presente n_i volte, con n_i all'incirca proporzionale a $\omega_t^{[i]}$:

$$n_i = \left\lfloor \frac{\sum_{k=1}^{i} \omega_t^k}{P} \right\rfloor - \left\lfloor \frac{\sum_{k=1}^{i-1} \omega_t^k}{P} \right\rfloor$$

con $P = \frac{\sum_{k=1}^{M} \omega_t^k}{M}$.Tale modo di procedere, se da un lato fornisce un algoritmo meno robusto (effetto che può essere in ogni caso bilanciato da un aumento del numero M

di campioni), dall'altro assicura una sua più rapida convergenza alla posizione reale, in quanto tende a ignorare e a non ricampionare quei campioni che hanno un peso minore di una certa soglia. Ottenuto a questo punto $bel(x_t)$, termina il generico passo dell'algoritmo di localizzazione, di cui un esempio è riportato in figura:

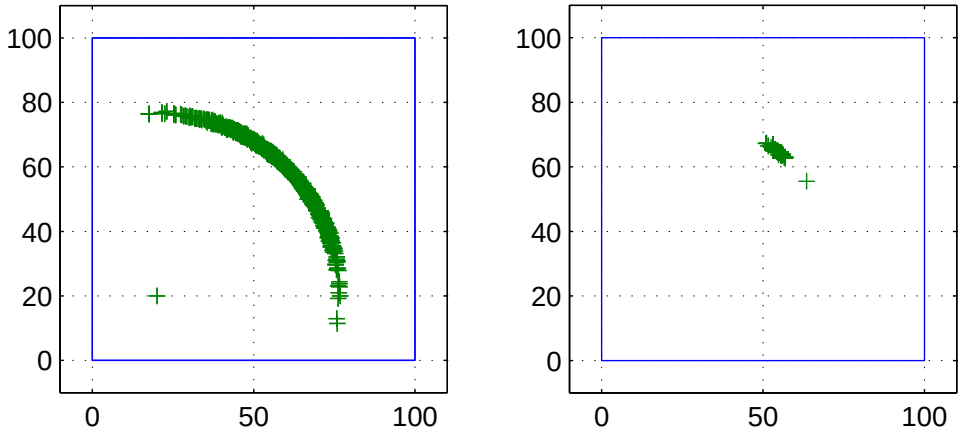


Figura 4.2: Stima dello stato prima e dopo il secondo sampling

4.2 Stima dei parametri

Nella descrizione dell'algoritmo di localizzazione, specificatamente nel calcolo della $p(z_k^t|x_t, m)$, si sono utilizzati una serie di parametri (z_{hit},z_{short},z_{max},z_{rand},σ_{hit},λ_{short}), che sono stati assunti implicitamente noti. Tali valori nella pratica sono di fatto sconosciuti, in quanto influenzati da innumerevoli fattori, quali ad esempio la struttura della mappa in esame, il raggio di misura dei sensori del robot o il numero di oggetti ignoti presenti nell'ambiente. Tuttavia, utilizzando i dati di una simulazione abbastanza accurata e un apposito algoritmo, è possibile effettuare una stima suffi-

cientemente precisa di tali parametri che garantisca conseguentemente una maggiore accuratezza dell'algoritmo di localizzazione. Si sono scelte a tale proposito due mappe di un generico edificio abitato, con all'interno un numero diverso di persone, si è posto il robot in una stanza e gli si è ordinato di attraversare il corridoio per giungere nella stanza adiacente, in presenza di persone in movimento. Una volta eseguita la simulazione inizializzando intuitivamente i parametri ignoti e memorizzate le misure ricevute dal robot, è possibile operare su di esse per la stima dei parametri ignoti. Introduciamo innanzitutto la variabile c_i che può assumere quattro valori (hit, short, max e rand) a seconda che la misura z_i rientri in uno dei quattro casi illustrati al caso precedente e le variabili $e_{i,hit}$,$e_{i,short}$,$e_{i,max}$,$e_{i,rand}$, ciascuna delle quali indica la probabilità che c_i assuma uno dei quattro valori previsti. Detto Z l'insieme delle misure e Θ l'insieme dei parametri non noti, l'algoritmo calcola le quattro $e_{i,x}$ massimizzando il valore atteso $E[logp(Z|X,m,\Theta)]$ e, assumendo che σ_{hit} e λ_{short} siano corretti:

$$\begin{pmatrix} e_{i,hit} \\ e_{i,short} \\ e_{i,max} \\ e_{i,rand} \end{pmatrix} = \eta \begin{pmatrix} p_{hit}(z_i|x_i,m) \\ p_{short}(z_i|x_i,m) \\ p_{max}(z_i|x_i,m) \\ p_{rand}(z_i|x_i,m) \end{pmatrix}$$

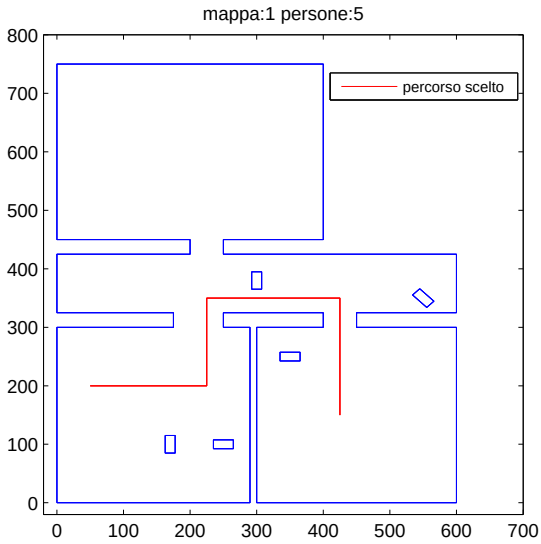
Una volta calcolati i quattro parametri $e_{i,x}$, è immediato calcolare i rispettivi z_x:

$$\begin{pmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{pmatrix} = \mid Z \mid^{-1} \sum_i \begin{pmatrix} e_{i,hit} \\ e_{i,short} \\ e_{i,max} \\ e_{i,rand} \end{pmatrix}$$

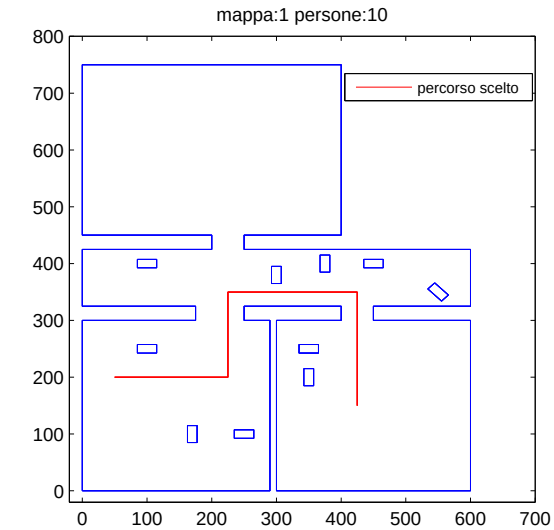
Per il calcolo di $sigma_{hit}$ e λ_{short} si segue un procedimento pressocchè identico: si massimizzano rispettivamente $E[logp(Z_{hit}|X,m,\Theta)]$ e $E[logp(Z_{short}|X,m,\Theta)]$ considerando corretti i valori di z_x appena calcolati:

$$\sigma_{hit} = \sqrt{\frac{1}{\sum_{z_i} e_{i,hit}} \sum_{z_i} e_{i,hit}(z_i - z_i^*)^2} \qquad \lambda_{short} = \frac{\sum_{z_i} e_{i,short}}{\sum_{z_i} e_{i,short} z_i}$$

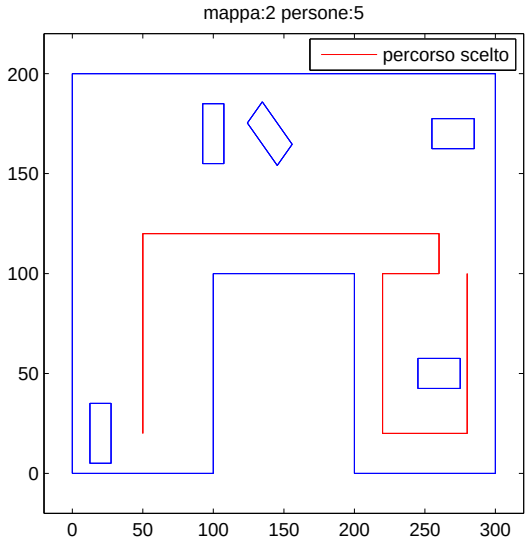
Appare evidente come l'algoritmo in esame non sia in forma chiusa, in quanto al suo primo passo calcola una serie di parametri p_1 assumendo noti altri parametri p_2, i quali però verranno modificati al passo successivo, richiedendo di fatto un nuovo calcolo dei parametri p_1. Tuttavia, iterando l'algoritmo un numero non eccessivamente elevato di volte, si nota una convergenza dei parametri più che sufficiente per l'uso richiesto. Nel seguito sono presentate le mappe e i percorsi scelti, nonchè la stima dei parametri ottenuta applicando l'algoritmo iterandolo 10 volte:



σ_{hit}	1.8437
λ_{short}	0.0222
z_{hit}	0.552
z_{max}	0.3146
z_{rand}	0.0849
z_{short}	0.0485



σ_{hit}	2.1919
λ_{short}	0.03
z_{hit}	0.4434
z_{max}	0.2996
z_{rand}	0.1363
z_{short}	0.1207



σ_{hit}	2.5266
λ_{short}	0.0328
z_{hit}	0.6065
z_{max}	0.0677
z_{rand}	0.157
z_{short}	0.1689

Un paio di esempi possono confermare quanto accennato precedentemente, ossia che la stima dei parametri varia in base alla struttura della mappa e al numero di oggetti presenti in essa. Nella terza simulazione, ad esempio, z_{max} ha un valore molto minore rispetto agli altri due casi, poichè le misurazioni oltre il raggio massimo del sensore sono meno probabili a causa delle dimensioni ridotte dell'ambiente; nella seconda simulazione, inoltre, z_{short} è più che triplicato rispetto alla simulazione effettuata con la stessa mappa ma con un numero minore di persone, semplicemente perchè saranno molto maggiori le misure troncate da elementi non previsti dalla mappa. Di seguito, si mostra la densità di probabilità $p(z_t|m, x_t)$ ottenuta con i parametri calcolati con i risultati della seconda simulazione:

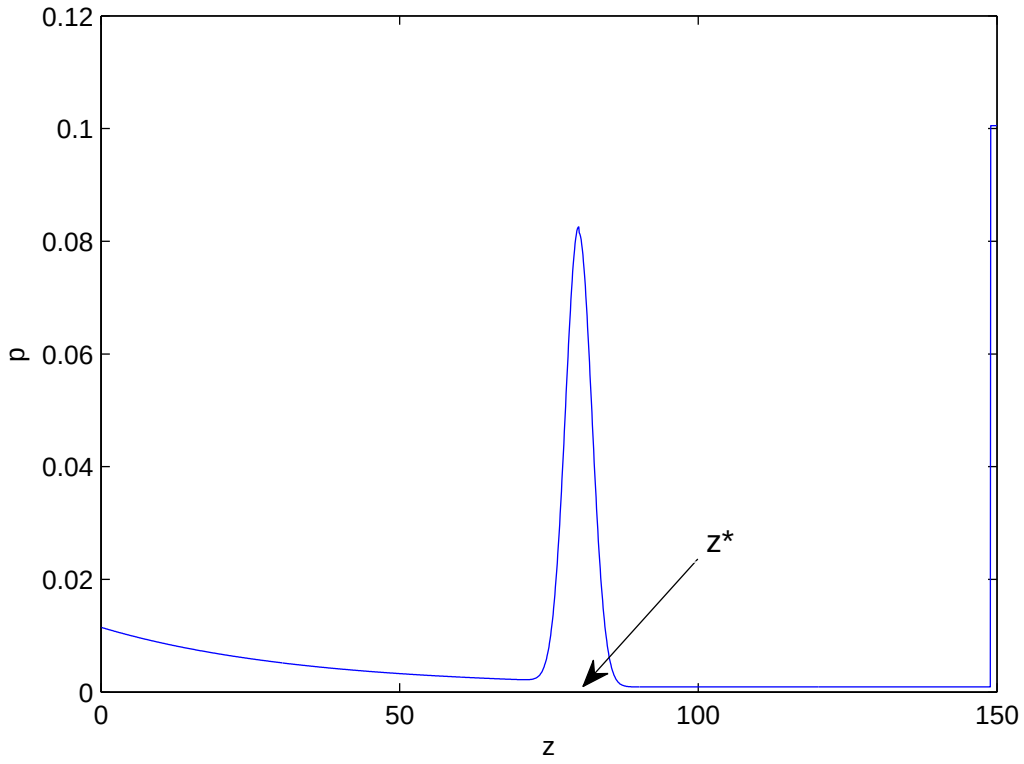


Figura 4.3: Distribuzione di probabilità utilizzata per il secondo sampling

Si è verificato sperimentalmente, inoltre, che non solo la convergenza dei parametri è assicurata in un numero relativamente breve di iterazioni dell'algoritmo di stima, ma anche che i valori di convergenza non dipendono in alcun modo dai valori iniziali utilizzati per la simulazione con cui si raccolgono le misure. A tale proposito, si allega l'andamento dei quattro parametri z_i della prima simulazione, nel caso in cui i loro valori iniziali siano rispettivamente $[0.4, 0.4, 0.1, 0.1]$, $[0.7, 0.1, 0.1, 0.1]$ e $[0.25, 0.25, 0.25, 0.25]$:

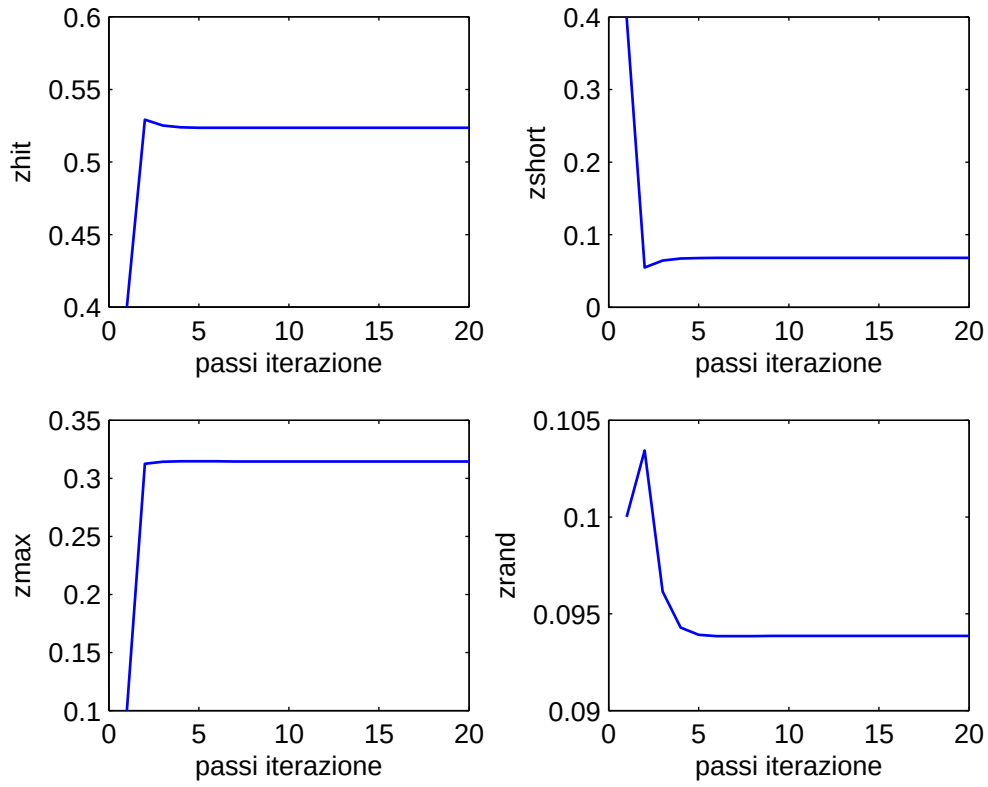


Figura 4.4: Convergenza dei parametri(1)

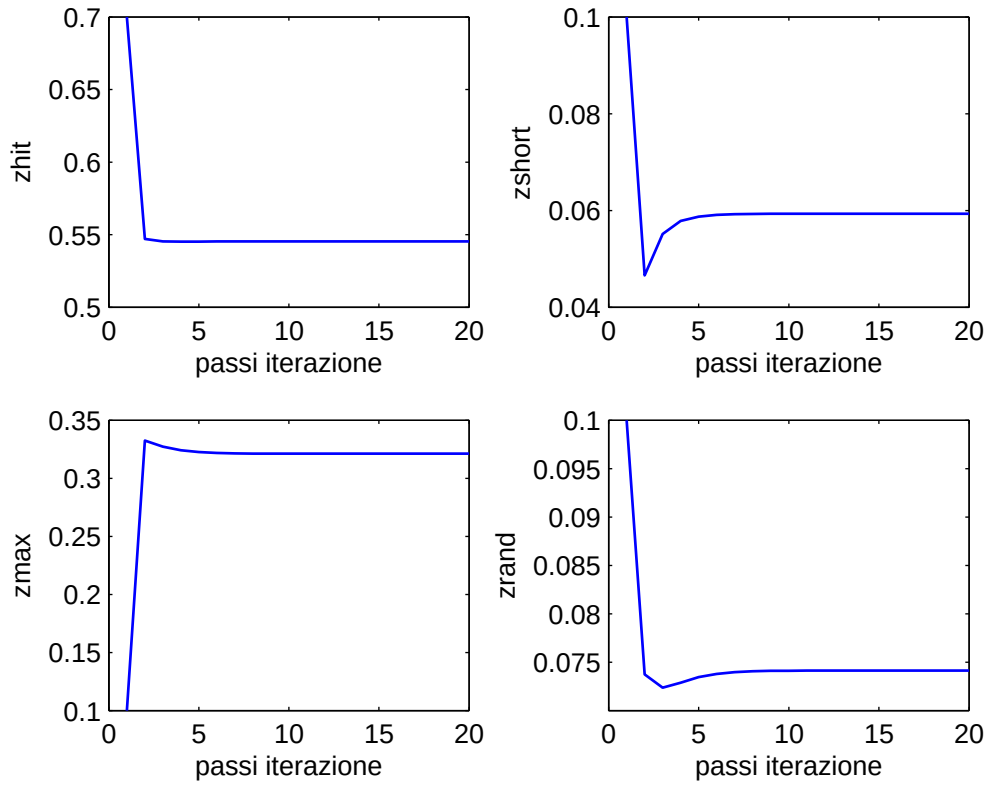


Figura 4.5: Convergenza dei parametri(2)

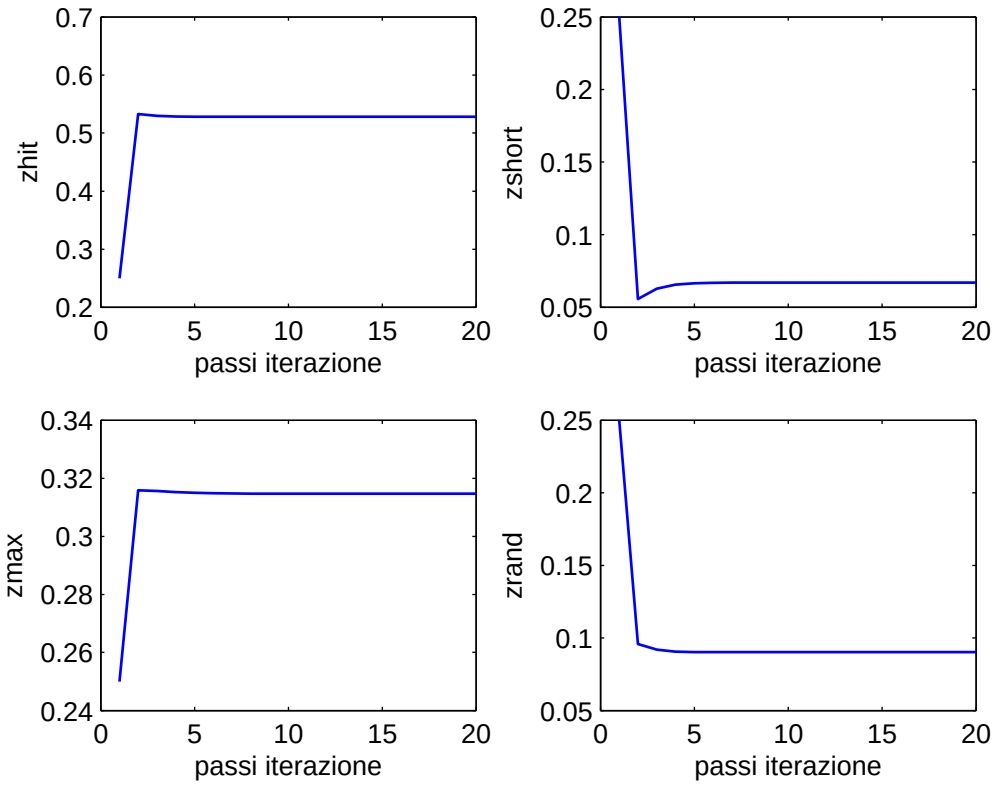


Figura 4.6: Convergenza dei parametri(3)

Capitolo 5

Simulazioni e risultati

Una volta stimati i parametri di $p(z_t|m, x_t)$ l'implementazione dell'algoritmo di localizzazione con particle filter può dirsi conclusa ed è quindi possibile effettuare una serie di simulazioni che ne verifichino le prestazioni. A questo proposito si sono scelte le mappe e i percorsi utilizzati al capitolo precedente, questi ultimi rappresentati da una serie di punti: al generico passo k dell'algoritmo il robot effettua lo spostamento necessario per portarsi dalla posizione stimata $\overline{k}$ al punto $k+1$ del percorso. Si utilizzano $M = 1000$ campioni per il resampling del particle filter, ipotizzando un robot con un diametro di 20 cm e i cui otto sensori equispaziati angolarmente presentino un raggio massimo di misurazione r_{max} pari a 150 cm. Le misurazioni e l'applicazione dell'algoritmo sono stati effettuati dopo ogni spostamento del robot di circa 25 cm e le costanti α_i relative all'errore di spostamento sono state tutte assunte pari a 0.005. Per quanto riguarda la complessità computazionale dell'algoritmo, ricordiamo che un suo generico passo è composto da due sampling, dal calcolo di ω_t^m e dal calcolo delle misure dei sensori. Le prime tre fasi descritte hanno complessità computazionale dell'ordine di $O(M)$ mentre il calcolo delle misure ha complessità dell'ordine di $O(LM)$, con L numero di segmenti che compongono la mappa. Si può quindi concludere che la complessità dell'intero algoritmo sarà dell'ordine di $O(LM)$. Il tempo di calcolo

relativo all'applicazione dell'algoritmo è stato di circa 1.5 sec per iterazione, ma bisogna tener conto che in una eventuale implementazione pratica tale tempo risulterà minore, in quanto non sarà necessario simulare le misure reali, che verranno di fatto fornite dai sensori. È riportato di seguito un esempio della simulazione effettuata: nei riquadri a sinistra, nei quali il simbolo rosso indica la posizione reale del robot, è possibile osservare $\overline{bel}(x_{t-1})$ e $\overline{bel}(x_t)$ del robot e nei riquadri a destra sono rappresentate $bel(x_{t-1})$ e $bel(x_t)$:

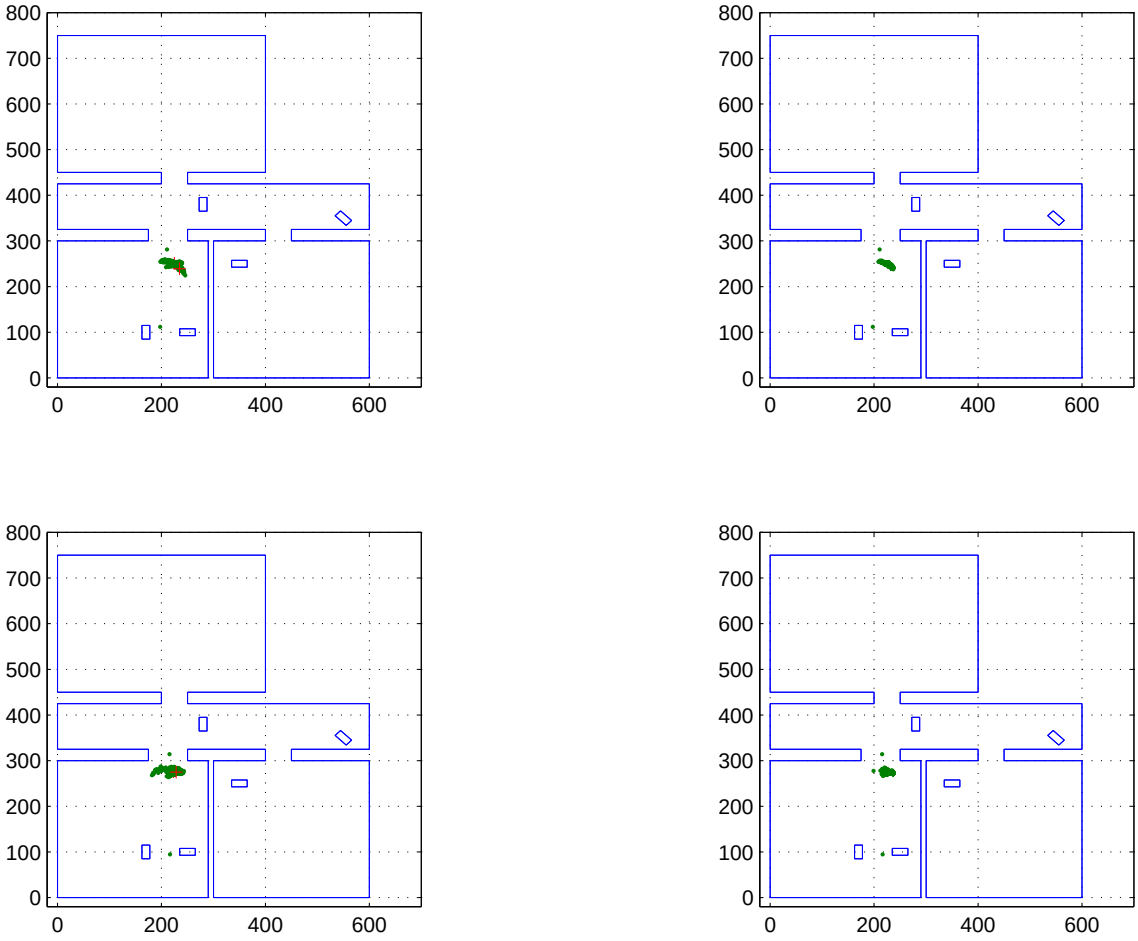


Figura 5.1: Esempio di simulazione

I risultati delle prime tre simulazioni, effettuate con posizione iniziale nota, sono riportati di seguito: di ciascuna di esse è fornito il tracciamento del percorso richiesto, di quello effettivamente realizzato e di quello stimato dal filtro, è presente inoltre un grafico dell'errore di posizione e di stima, pari rispettivamente alla distanza tra la posizione imposta e quella reale, e tra la posizione reale e quella stimata.

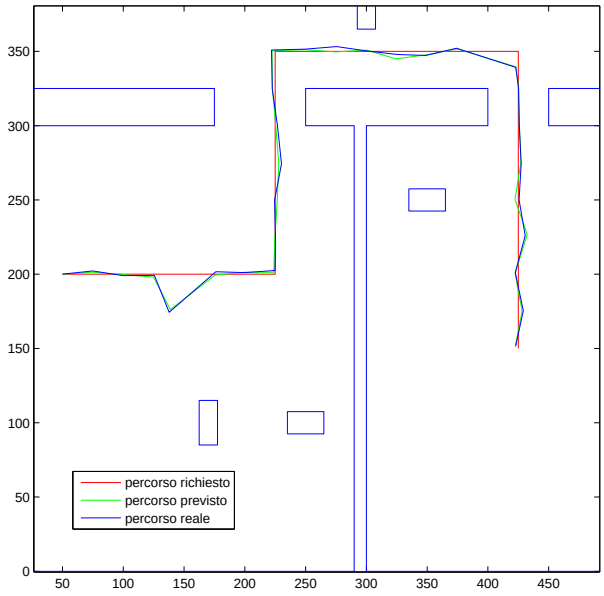


Figura 5.2: Simulazione 1: percorso seguito

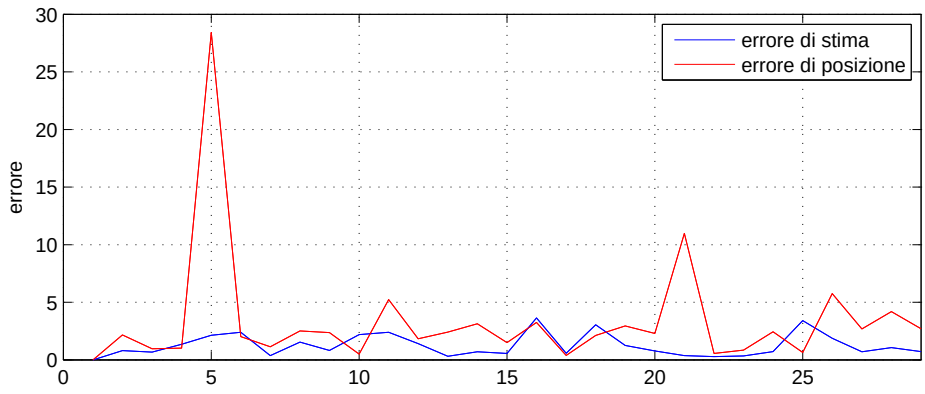


Figura 5.3: Simulazione 1: errore di posizione e stima

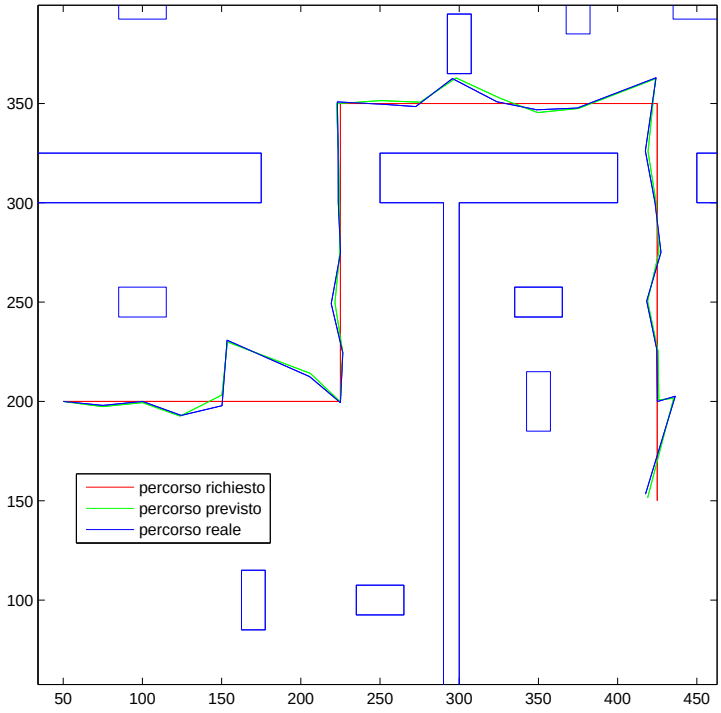


Figura 5.4: Simulazione 2: percorso seguito

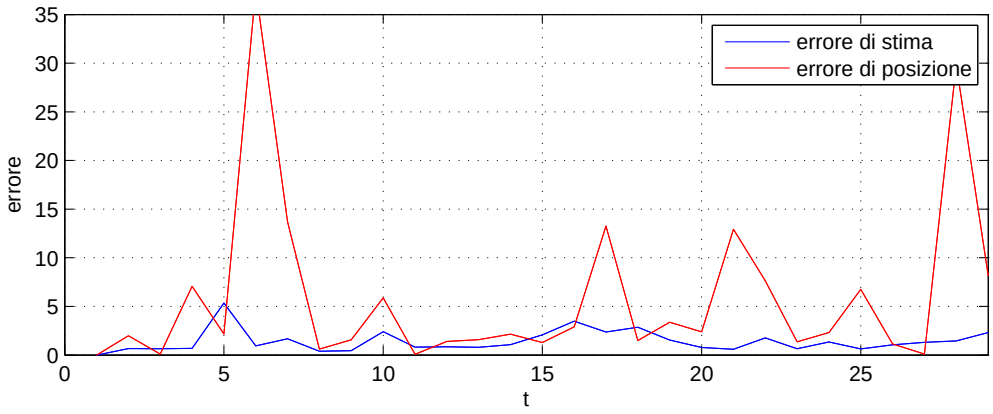


Figura 5.5: Simulazione 2: errore di posizione e stima

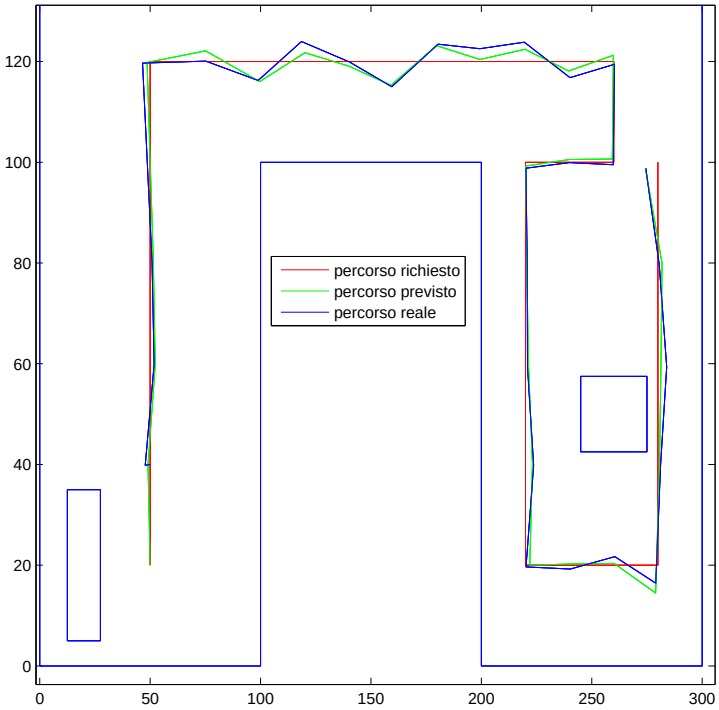


Figura 5.6: Simulazione 3: percorso seguito

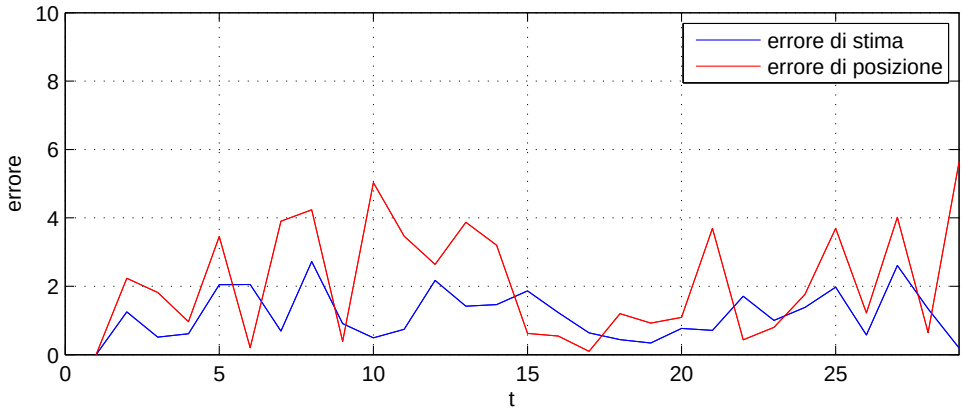


Figura 5.7: Simulazione 3: errore di posizione e stima

Appare evidente come nei casi in esame l'algoritmo di localizzazione utilizzato permetta di seguire una traiettoria assegnata con errore piú che accettabile. Si può notare come tale errore possa essere leggermente maggiore nel caso in cui ci si trovi in una zona fittamente popolata da oggetti estranei alla mappa, ma tale errore viene ridotto prontamente nei passi successivi dell'algoritmo. L'algoritmo é stato testato anche nel caso in cui la posizione iniziale del robot sia ignota, specificatamente la densità di probabilità iniziale sia distribuita uniformemente su un intera stanza della mappa 1. I risultati sono i seguenti:

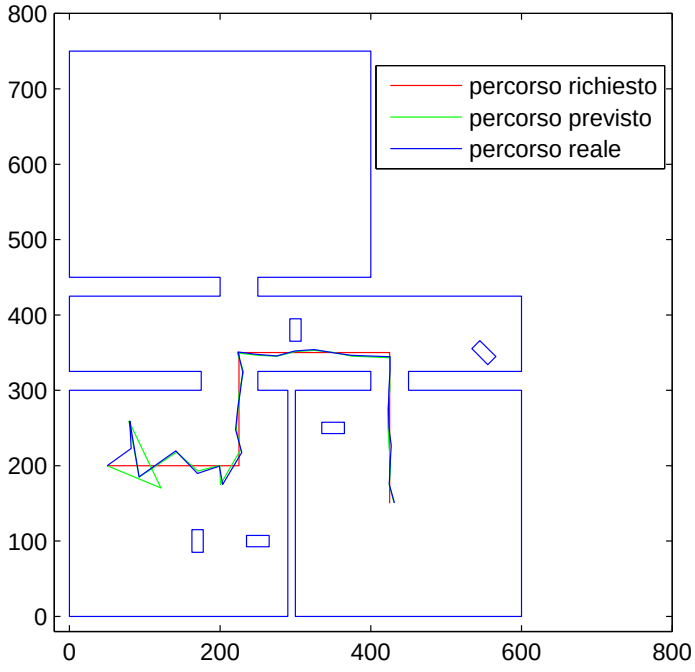


Figura 5.8: Simulazione 4: percorso seguito

Si può notare come l'errore di stima iniziale, inevitabilmente molto elevato, viene praticamente azzerato con un solo passo dell'algoritmo, che di fatto si rivela adeguato anche per problemi di localizzazione globale.

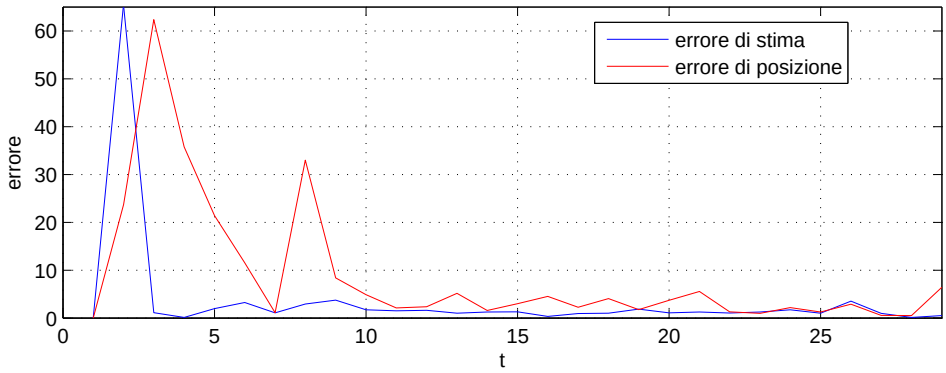


Figura 5.9: Simulazione 4: errore di posizione e stima

Capitolo 6

Conclusioni e sviluppi futuri

L'algoritmo sviluppato si è dimostrato funzionante e, nell'ambito delle simulazioni, piuttosto performante. La scelta di utilizzare un particle filter si è rivelata più che soddisfacente e ha consentito di realizzare un algoritmo versatile sotto molteplici punti di vista:

- la modellazione usata consente la sua applicabilità per un'ampia varietà di ambienti e, con lievi modifiche, per numerosi modelli di robot mobili.
- la possibilità di operare sul numero di campioni che descrivono la stima dello stato garantisce di poter individuare il compromesso desiderato tra potenza di calcolo e approssimazione nella stima.
- il tipo di parametrizzazione utilizzato per il secondo resampling garantisce l'applicabilità del filtro a un'ampia gamma di ambienti e situazioni.

Eventuali sviluppi al lavoro svolto potrebbero essere l'implementazione pratica dell'algoritmo di localizzazione e di stima dei parametri per verificare l'attendibilità dei risultati simulativi, oppure un'integrazione con un algoritmo probabilistico di mapping che permetta di realizzare lo SLAM

Appendice A
Listato

Rappresentazione della mappa

```
%specifiche della mappa
global npunti tipo ymax ymin xmax xmin xfissa yfissa m q p_ascissa p_ordinata


%inizializzazione variabili

npunti=8;
tipo=zeros(1,npunti);
ymax=zeros(1,npunti);
ymin=zeros(1,npunti);
xmax=zeros(1,npunti);
xmin=zeros(1,npunti);
xfissa=zeros(1,npunti);
yfissa=zeros(1,npunti);
m=zeros(1,npunti);
q=zeros(1,npunti);


%definizioni punti della mappa

p_ascissa(1)=0;
p_ascissa(2)=100;
p_ascissa(3)=100;
p_ascissa(4)=200;
p_ascissa(5)=200;
p_ascissa(6)=300;
p_ascissa(7)=300;
p_ascissa(8)=0;

p_ordinata(1)=0;
p_ordinata(2)=0;
p_ordinata(3)=100;
p_ordinata(4)=100;
p_ordinata(5)=0;
p_ordinata(6)=0;
p_ordinata(7)=200;
p_ordinata(8)=200;


%conversione in segmenti

for i=1:1:npunti
    if i<npunti
        temp=i+1;
```

```
    else temp=1;
    end

    if (p_ascissa(i)==p_ascissa(temp))
        tipo(i)=2; %retta parallela all'asse delle ordinate
        ymax(i)=max(p_ordinata(i),p_ordinata(temp));
        ymin(i)=min(p_ordinata(i),p_ordinata(temp));
        xfissa(i)=p_ascissa(i);
    else

    if (p_ordinata(i)==p_ordinata(temp))
        tipo(i)=1; %retta parallela all'asse delle ascisse
        xmax(i)=max(p_ascissa(i),p_ascissa(temp));
        xmin(i)=min(p_ascissa(i),p_ascissa(temp));
        yfissa(i)=p_ordinata(i);

    else
        tipo(i)=0;
        m(i)=(p_ordinata(i)-p_ordinata(temp))/(p_ascissa(i)-p_ascissa(temp));
        q(i)=((p_ascissa(i)*p_ordinata(temp))-(p_ordinata(i)*p_ascissa(temp)))
                /(p_ascissa(i)-p_ascissa(temp));
        xmax(i)=max(p_ascissa(i),p_ascissa(temp));
        xmin(i)=min(p_ascissa(i),p_ascissa(temp));
    end
    end
end
```

Oggetti estranei alla mappa

```
%data posizione e orientamento della persona, genera i relativi segmenti
%che la modellizzano, in modo da poter effettuare le misurazioni reali

function [punti]=genera_persona(coordinate,indice)
global npunti tipo ymax ymin xfissa xmax xmin yfissa m q

lunghezza=7.5;
larghezza=15;
theta=atan(larghezza/lunghezza);
raggio=sqrt(lunghezza^2+larghezza^2);
%theta1=2*asin(lunghezza/raggio)
%theta2=2*asin(larghezza/raggio)

punti(1,1)=coordinate(1)+raggio*cos(coordinate(3)+theta);
punti(1,2)=coordinate(2)+raggio*sin(coordinate(3)+theta);
punti(2,1)=coordinate(1)+raggio*cos(coordinate(3)+(pi-theta));
punti(2,2)=coordinate(2)+raggio*sin(coordinate(3)+(pi-theta));
punti(3,1)=coordinate(1)+raggio*cos(coordinate(3)+(pi+theta));
punti(3,2)=coordinate(2)+raggio*sin(coordinate(3)+(pi+theta));
punti(4,1)=coordinate(1)+raggio*cos(coordinate(3)+(2*pi-theta));
punti(4,2)=coordinate(2)+raggio*sin(coordinate(3)+(2*pi-theta));


for j=1:1:4
tipo(npunti+j+(indice-1)*4)=0;
ymax(npunti+j+(indice-1)*4)=0;
ymin(npunti+j+(indice-1)*4)=0;
xfissa(npunti+j+(indice-1)*4)=0;
xmax(npunti+j+(indice-1)*4)=0;
xmin(npunti+j+(indice-1)*4)=0;
yfissa(npunti+j+(indice-1)*4)=0;
```

```
m(npunti+j+(indice-1)*4)=0;
q(npunti+j+(indice-1)*4)=0;
end

for i=1:1:4
    if i<4
        temp=i+1;
    else temp=1;
    end
nuovoindice=npunti+i+(indice-1)*4;

    if (punti(i,1)==punti(temp,1))
        tipo(nuovoindice)=2; %retta parallela all'asse delle ordinate
        ymax(nuovoindice)=max(punti(i,2),punti(temp,2));
        ymin(nuovoindice)=min(punti(i,2),punti(temp,2));
        xfissa(nuovoindice)=punti(i,1);
    else

    if (punti(i,2)==punti(temp,2))
        tipo(nuovoindice)=1; %retta parallela all'asse delle ascisse
        xmax(nuovoindice)=max(punti(i,1),punti(temp,1));
        xmin(nuovoindice)=min(punti(i,1),punti(temp,1));
        yfissa(nuovoindice)=punti(i,2);

    else
        tipo(nuovoindice)=0;
        m(nuovoindice)=(punti(i,2)-punti(temp,2))/(punti(i,1)-punti(temp));
        q(nuovoindice)=((punti(i,1)*punti(temp,2))-(punti(i,2)*punti(temp,1)))
                /(punti(i,1)-punti(temp,1));
        xmax(nuovoindice)=max(punti(i,1),punti(temp,1));
        xmin(nuovoindice)=min(punti(i,1),punti(temp,1));
    end
    end
end
```

Calcolo misure ideali

```
%calcola la misura ideale effettuata dal sensore, dato il vettore posizione
%(x,y,th) del robot e la mappa


function [misura,valido]=misideale(posizione)
global npunti tipo m q xmin xmax ymin ymax xfissa yfissa

while(posizione(1,3)<0)
    posizione(1,3)=posizione(1,3)+(2*pi);
end

while(posizione(1,3)>=(2*pi))
    posizione(1,3)=posizione(1,3)-(2*pi);
end

distanza=zeros(1,npunti);
for i=1:1:npunti
  if(tipo(i)==0)
    if (posizione(1,3)==0)
        xint(i)=(posizione(1,2)-q(i))/m(i);
        if ((xint(i)-posizione(1,1))>0 && xint(i)<=xmax(i) && xint(i)>=xmin(i))
          distanza(i)=xint(i)-posizione(1,1);
        else distanza(i)=-1;
```

```
        end
    end
    if((posizione(1,3)>0 && posizione(1,3)<(pi/2))
                          || (posizione(1,3)>(3*pi/2) && posizione(1,3)<(2*pi)))
       mm(i)=tan(posizione(1,3));
       qq(i)=posizione(1,2)-(mm(i)*posizione(1,1));
       if (mm(i)~=m(i))
         xint(i)=(q(i)-qq(i))/(mm(i)-m(i));
         if (xint(i)>posizione(1,1) && xint(i)>=xmin(i) && xint(i)<=xmax(i))
           distanza(i)=sqrt((xint(i)-posizione(1,1))^2
                                    +((m(i)*xint(i)+q(i))-posizione(1,2))^2);
         else distanza(i)=-1;
         end
       else distanza(i)=-1;
       end
    end
    if(posizione(1,3)==pi/2)
      yint(i)=m(i)*posizione(1,1)+q(i);
      if (yint(i)>posizione(1,2) && posizione(1,1)>=xmin(i) && posizione(1,1)<=xmax(i))
        distanza(i)=yint(i)-posizione(1,2);
      else distanza(i)=-1;
      end
    end
    if(posizione(1,3)==pi)
      xint(i)=(posizione(1,2)-q(i))/m(i);
      if ((posizione(1,1)-xint(i))>0 && xint(i)<=xmax(i) && xint(i)>=xmin(i))
        distanza(i)=posizione(1,1)-xint(i);
      else distanza(i)=-1;
      end
    end
    if(posizione(1,3)==3*pi/2)
      yint(i)=m(i)*posizione(1,1)+q(i);
      if (yint(i)<posizione(1,2) && posizione(1,1)>=xmin(i) && posizione(1,1)<=xmax(i))
        distanza(i)=posizione(1,2)-yint(i);
      else distanza(i)=-1;
      end
    end
    if((posizione(1,3)>pi/2 && posizione(1,3)<pi)
                              || (posizione(1,3)>pi && posizione(1,3)<(3*pi/2)))
     mm(i)=tan(posizione(1,3));
      qq(i)=posizione(1,2)-(mm(i)*posizione(1,1));
      if (mm(i)~=m(i))
        xint(i)=(q(i)-qq(i))/(mm(i)-m(i));
        if (xint(i)<posizione(1,1) && xint(i)>=xmin(i) && xint(i)<=xmax(i))
          distanza(i)=sqrt((xint(i)-posizione(1,1))^2
                                  +((m(i)*xint(i)+q(i))-posizione(1,2))^2);
        else distanza(i)=-1;
        end
     else distanza(i)=-1;
     end
  end
end

if(tipo(i)==1)
  if (posizione(1,3)==0 || posizione(1,3)==pi)
    distanza(i)=-1;
  end
  if (posizione(1,3)==pi/2)
    if(posizione(1,1)>=xmin(i) && posizione(1,1)<=xmax(i) && yfissa(i)>posizione(1,2))
      distanza(i)=yfissa(i)-posizione(1,2);
    else distanza(i)=-1;
    end
  end
  if (posizione(1,3)==3*pi/2)
    if(posizione(1,3)>=xmin(i) && posizione(1,3)<=xmax(i) && yfissa(i)<posizione(1,2))
```

```
    distanza(i)=posizione(1,2)-yfissa(i);
  else distanza(i)=-1;
  end
end
if((posizione(1,3)>0 && posizione(1,3)<(pi/2))
                        || (posizione(1,3)>(3*pi/2) && posizione(1,3)<(2*pi)))
  mm(i)=tan(posizione(1,3));
  qq(i)=posizione(1,2)-(mm(i)*posizione(1,1));
  xint(i)=(yfissa(i)-qq(i))/mm(i);
  if (xint(i)>posizione(1,1) && xint(i)>=xmin(i) && xint(i)<=xmax(i))
    distanza(i)=sqrt((xint(i)-posizione(1,1))^2
                        +((mm(i)*xint(i)+qq(i))-posizione(1,2))^2);
  else distanza(i)=-1;
  end
end
if((posizione(1,3)>pi/2 && posizione(1,3)<pi)
                        || (posizione(1,3)>pi && posizione(1,3)<(3*pi/2)))
  mm(i)=tan(posizione(1,3));
  qq(i)=posizione(1,2)-(mm(i)*posizione(1,1));
  xint(i)=(yfissa(i)-qq(i))/mm(i);
    if (xint(i)<posizione(1,1) && xint(i)>=xmin(i) && xint(i)<=xmax(i))
      distanza(i)=sqrt((xint(i)-posizione(1,1))^2
                        +((mm(i)*xint(i)+qq(i))-posizione(1,2))^2);
    else distanza(i)=-1;
    end
  end
end

if(tipo(i)==2)
  if (posizione(1,3)==pi/2 || posizione(1,3)==3*pi/2)
    distanza(i)=-1;
  end
  if (posizione(1,3)==0)
    if(posizione(1,2)>=ymin(i) && posizione(1,2)<=ymax(i) && xfissa(i)>posizione(1,1))
      distanza(i)=xfissa(i)-posizione(1,1);
    else distanza(i)=-1;
    end
  end
  if (posizione(1,3)==pi)
    if(posizione(1,2)>=ymin(i) && posizione(1,2)<=ymax(i) && xfissa(i)<posizione(1,1))
      distanza(i)=posizione(1,1)-xfissa(i);
    else distanza(i)=-1;
    end
  end
  if((posizione(1,3)>0 && posizione(1,3)<(pi/2))
                                  || (posizione(1,3)>(3*pi/2) && posizione(1,3)<(2*pi)))
    mm(i)=tan(posizione(1,3));
    qq(i)=posizione(1,2)-(mm(i)*posizione(1,1));
    yint(i)=(xfissa(i)*mm(i))+qq(i);
    if (xfissa(i)>posizione(1,1) && yint(i)>=ymin(i) && yint(i)<=ymax(i))
      distanza(i)=sqrt((xfissa(i)-posizione(1,1))^2
                                  +((mm(i)*xfissa(i)+qq(i))-posizione(1,2))^2);
    else distanza(i)=-1;
    end

  end
  if((posizione(1,3)>(pi/2) && posizione(1,3)<pi)
                                  || (posizione(1,3)>pi && posizione(1,3)<(3*pi/2)))
    mm(i)=tan(posizione(1,3));
    qq(i)=posizione(1,2)-(mm(i)*posizione(1,1));
    yint(i)=xfissa(i)*mm(i)+qq(i);
      if (xfissa(i)<posizione(1,1) && yint(i)>=ymin(i) && yint(i)<=ymax(i))
        distanza(i)=sqrt((xfissa(i)-posizione(1,1))^2
                                      +((mm(i)*xfissa(i)+qq(i))-posizione(1,2))^2);
      else distanza(i)=-1;
```

```
        end
    end
  end
end

if(distanza==-1*ones(1,npunti))
    valido=0;
else valido=1;
end

temp=max(distanza);
for i=1:1:npunti
    if(distanza(i)==-1)
        distanza(i)=temp+1;
    end
end

misura=min(distanza);
```

Spostamento del robot

```
%sampling con odometria

function [nuova_pos]=sampleodom(robin,robfin,pos)
global alpha1 alpha2 alpha3 alpha4

drot1=atan2(robfin(1,2)-robin(1,2),robfin(1,1)-robin(1,1))-robin(1,3);
dtrans=sqrt((robin(1,1)-robfin(1,1))^2+(robin(1,2)-robfin(1,2))^2);
drot2=robfin(1,3)-robin(1,3)-drot1;

ddrot1=normrnd(drot1,alpha1*drot1+alpha2*dtrans);
ddtrans=normrnd(dtrans,alpha3*dtrans+alpha4*(drot1+drot2));
ddrot2=normrnd(drot2,alpha1*drot2+alpha2*dtrans);

nuova_pos(1,1)=pos(1,1)+ddtrans*cos(pos(1,3)+ddrot1);
nuova_pos(1,2)=pos(1,2)+ddtrans*sin(pos(1,3)+ddrot1);
nuova_pos(1,3)=pos(1,3)+ddrot1+ddrot2;
```

Calcolo $p(z|x_t, m)$

```
function [p]=get_prob(z,zideale)
global rmax sigma lambda delta zhit zshort zmax zrand

picco=0;
if(z>=(rmax-delta) && z<=rmax)
    picco=(1/delta);
end

p=zhit*gaussiana(z,zideale)+zshort*esponenziale(z,zideale)+zrand*(1/rmax)+zmax*picco;
```

Secondo resampling

```
%nuova distribuzione di punti in seguito alle misure
function [distr_rid]=ridistribuisci(distr,prob,somma)
global M

counter=0;
distrprob=0;
posizione=0;
for i=1:1:M
    if (prob(i)~=0)
        counter=counter+1;
        distrprob(counter)=prob(i)/somma;
        posizione(counter)=i;
    end
end

j=1;

while(j<=M)
    ascissa=unidrnd(counter);
    ordinata=unifrnd(0,1); %invece di max(prob)
        if(ordinata<=distrprob(ascissa))
        distr_rid(j,:)=distr(posizione(ascissa),:);
        j=j+1;
    end
end
```

Stima dei parametri

```
%algoritmo per la stima dei parametri

numeromisure=28;
numerosensori=8;


for i=1:1:numeromisure
    for j=1:1:numerosensori
        picco=0;
        if((z(i,j)>=(rmax-delta)) && (z(i,j)<=rmax))
            picco=(1/delta);
        end
        phit(i,j)=gaussiana(z(i,j),zstar(i,j));
        pshort(i,j)=esponenziale(z(i,j),zstar(i,j));
        prand(i,j)=1/rmax;
        pmax(i,j)=picco;
        ni(i,j)=1/(phit(i,j)+pshort(i,j)+prand(i,j)+pmax(i,j));
        ehit(i,j)=ni(i,j)*phit(i,j);
        eshort(i,j)=ni(i,j)*pshort(i,j);
        emax(i,j)=ni(i,j)*pmax(i,j);
        erand(i,j)=ni(i,j)*prand(i,j);
    end
end

zhit=sum(sum(ehit))/(numeromisure*numerosensori);
zshort=sum(sum(eshort))/(numeromisure*numerosensori);
zmax=sum(sum(emax))/(numeromisure*numerosensori);
zrand=sum(sum(erand))/(numeromisure*numerosensori);
sommatoria1=0;
```

```
sommatoria2=0;

for i=1:1:numeromisure
    for j=1:1:numerosensori
        sommatoria1=sommatoria1+(ehit(i,j)*(z(i,j)-zstar(i,j))^2);
        sommatoria2=sommatoria2+(eshort(i,j)*z(i,j));
    end
end


sigma=sqrt(sommatoria1/sum(sum(ehit)));
lambda=sum(sum(eshort))/(sommatoria2);
```

Esempio di simulazione

```
global persona1 persona2 persona3 persona4 persona5

rob(1,:)=[50,200,0];
rob(2,:)=[75,200,0];
rob(3,:)=[100,200,0];

persona1=genera_persona([50,100,0],1);
persona2=genera_persona([250,250,3*pi/2],2);
persona3=genera_persona([280,380,0],3);
persona4=genera_persona([550,350,pi/4],4);
persona5=genera_persona([350,250,3*pi/2],5);

stima(1,:)=rob(1,:);

disp11=odometria(rob(1,:),rob(2,:),distr_in);
indice=unidrnd(M);
x(1,:)=disp11(indice,:);
z(1,:)=misrealrobot(x(1,:));
zstar(1,:)=misidrobot(x(1,:));
[prob,somma,limite]=calcolo_w(disp11,z(1,:));
disp12=ridistribuisci2(disp11,prob,somma,limite);
figure(1)
subplot(2,2,1)
plotdistr(disp11)
hold on
scatter(x(1,1),x(1,2),'r','+')

subplot(2,2,2)
plotdistr(disp12)

%si spostano le persone
persona1=genera_persona([70,100,0],1);
persona2=genera_persona([250,230,3*pi/2],2);

stima(2,:)=creastima(disp12);
disp21=odometria(stima(2,:),rob(3,:),disp12);
indice=unidrnd(M);
x(2,:)=disp21(indice,:);
z(2,:)=misrealrobot(x(2,:));
zstar(2,:)=misidrobot(x(2,:));
[prob,somma,limite]=calcolo_w(disp21,z(2,:));
disp22=ridistribuisci2(disp21,prob,somma,limite);
subplot(2,2,3)
plotdistr(disp21)
hold on
```

```
scatter(x(2,1),x(2,2),'r','+')

subplot(2,2,4)
plotdistr(disp22)
stima(3,:)=creastima(disp22);

%si spostano le persone
persona1=genera_persona([85,100,0],1);
persona2=genera_persona([250,220,3*pi/2],2);
```

Elenco delle figure

Bibliografia

[1] S. Thrun - W. Burgard - D. Fox *"Probabilistic Robotics"*, The MIT press, 2005.

[2] G. Galati - G.Pavan, *"Teoria dei fenomeni aleatori"*, TexMat, 2004.

[3] B. Siciliano - L. Sciavicco - L. Villani - G.Oriolo *"Robotica:Modellistica, pianificazione e controllo"*, McGraw-Hill, 2008.

[4] J. Borenstein, H.R. Everett, L. Feng *"Navigating Mobile Robots: Systems and Techniques"*, A K Peters, 1996.

[5] H.R. Everett *"Sensors for mobile robots:Theory and applications"*, A K Peters, 1995.

[6] R. Featherstone *"Robot dynamics algorithm"*, Kluwer, 1987.

[7] J.L. Jones, A.M. Flynn *"Mobile robots: Inspiration to implementation"*, A K Peters, 19932.