UNIVERSITÀ DEGLI STUDI DI ROMA TOR VERGATA

FACOLTÀ DI INGEGNERIA

Corso di Laurea Specialistica in Ingegneria dell'Automazione

A.A. 2007/2008

Tesi di Laurea Specialistica

TECNICHE DI LOCALIZZAZIONE ATTIVA PER ROBOT MOBILI

RELATORE

Prof. Francesco Martinelli

CANDIDATO

Vittoria Piantelli

A mio nonno,
l'Ing. Vittorio Piazzini,
che fin dalla tenera età
mi ha fatto innamorare
delle scienze esatte.

Indice

Ringraziamenti

Desidero innanzitutto ringraziare il Prof. Francesco Martinelli per i preziosi insegnamenti, per la grande disponibilità dimostrata e per avermi introdotta all'affascinante e complesso mondo della robotica mobile.

Vorrei ringraziare anche l'intero corpo docente; rimarrà in me il piacevole ricordo di questi cinque anni di studio che ho trascorso a tempo pieno a Tor Vergata con professori disponibili al dialogo e a confrontarsi con le idee altrui, qualità non da tutti.

Come non ringraziare, inoltre, i compagni di corso con i quali ho prepatato numerosi esami condividendo fatiche e successi.

Un dolce pensiero è rivolto ai miei genitori e a mia sorella, senza i quali non avrei mai raggiunto questo traguardo.

Grazie a Federica, nonna Graziella e King, i miei tre angeli, per l'aiuto in tutti questi anni, e senza il cui affetto non sarebbe stato lo stesso. E ancora grazie ad Antonia, Lydia, Margherita e a tutti gli amici per il continuo sostegno.

Un grazie specialissimo a Francesco che mi è stato sempre vicino amorevolmente e se ho raggiunto questo traguardo lo devo anche a lui che mi ha incoraggiata nelle situazioni difficili, suggerendomi le modalità per raggiungere i miei obiettivi.

Introduzione

La robotica è nata dall'esigenza dell'uomo di essere sostituito da macchine nell'esecuzione di compiti gravosi, di precisione, noiosamente ripetitivi, in ambienti e situazioni difficili. Tale esigenza ha spinto l'uomo a creare macchine che, oltre ad eseguire lavori a comando, rispondano in mariera intelligente, autonoma ed efficiente nel raggiungimento dello scopo prefissato.

La robotica è un settore di ricerca in continua evoluzione ed è destinata a espandersi in qualsiasi settore. Sempre più nei film di fatascienza e nell'immaginario collettivo ci vediamo proiettati in un mondo completamente robotizzato dove coesistono insieme umani e androidi. Questi ultimi sono macchine all'avanguardia che rappresentano il nuovo e ormai prossimo futuro. Gli scienziati studiano automi che siano sempre più simili all'uomo e che rispondano autonomamente agli stimoli ambientali. Da qui nasce il crescente interesse nello studio di tecniche in grado di fornire al robot l'autonomia necessaria per poter prendere decisioni.

Per pianificare le operazioni per le quali il robot è stato realizzato è necessaria la conoscenza della posa (posizione e orientamento): per svolgere i compiti nella maniera più efficiente e intelligente deve sapere esattamente quale posizione occupa all'interno dell'ambiente di lavoro. Bisogna per questo risolvere il problema della localizzazione mobile, ovvero determinare la stima della posa di un robot mobile attraverso l'acquisizione e successiva rielaborazione delle informazioni ottenute dall'ambiente circostante,

dunque attraverso l'analisi di dati sensoriali.

La maggior parte degli approcci per risolvere tale problema si avvale di un tipo di localizzazione passiva che non sfrutta l'opportunità di scegliere il movimento durante la localizzazione: il percorso da compiere è infatti già assegnato.

L'obiettivo principale di questo elaborato riguarda, invece, la soluzione del problema da un punto di vista attivo. Si tratta di un obiettivo molto interessante: implica uno studio continuo dell'ambiente da parte del robot che dopo aver rilevato le informazioni acquisite con i dispositivi sensoriali di cui dispone, riutilizza tali dati per decidere quale è l'azione di controllo migliore da effettuare per localizzarsi nel modo più efficiente. Il criterio razionale di selezione della mossa più opportuna è scelto dal robot in modo da minimizzare l'Entropia associata alla stima della sua posa (W. Burgard, D. Fox, S. Thrun; *Active Mobile Robot Localization by Entropy Minimization*, 1997); tale tecnica ha come scopo principale quello di trovare la risposta a domande del tipo: 'dove andare per meglio posizionarsi?'. L'Entropia è un buon indicatore del grado di incertezza nella percezione della posa; una sua diminuzione comporta una maggior certezza di localizzazione. Questa tecnica, come verrà spiegato nell'elaborato, si basa su un approccio probabilistico passivo di tipo Markoviano alla localizzazione e sulla derivazione del filtro Bayesiano (S. Thrun, W. Burgard, D. Fox; *Probabilistic Robotics*, 2006). Il metodo permette di localizzare attivamente il robot tramite un algoritmo di selezione dell'azione dopo aver elaborato e analizzato le letture sensoriali relative alla presenza o assenza landmarks e i movimenti effettuati (S. Thrun; *Bayesian landmark learning for mobile robot localization*, 1998).

Il controllo implementato ha in ingresso la mappa metrica di un ambiente Mono/Bi-dimensionale nel quale il robot sarà libero di navigare, ma non ha alcuna informazione circa la posa iniziale: siamo nel caso della localizzazione globale.

I risulati sperimantali hanno dimostrato che la localizzazione è migliorata nettamente con il controllo attivo del movimento del robot minimizzando l'Entrolpia durante l'esplorazione.

Dopo una breve introduzione sul problema della localizzazione e sulle metodologie esistenti (Capitolo 1), il lavoro è stato inizialmente trattato da un punto di vista teorico (Capitolo 2) e solo successivamente sono state fornite spiegazioni riguardanti gli algoritmi implementati sperimentalmente durante la fase di progetto e di scrittura del codice (Capitolo 3), che è stato posto al vaglio di numerose verifiche di funzionamento (Capitolo 4).

Capitolo 1

Il problema della localizzazione robotica

Questo capitolo, prettamente introduttivo, illustra brevemente le metodologie esistenti per affrontare il problema della localizzazione di un robot mobile. Vengono introdotti i concetti principali dell'approccio probabilistico nella robotica e la nomenclaura utilizzata in tale settore scientifico.

1.1 Metodi di localizzazione

La Robotica mobile permette di percepire e manipolare il mondo fisico attraverso un controllo computerizzato.

Sempre più presente nel mondo essa opera in svariati campi della conoscenza umana; esempi di successo per sistemi robotici includono piattaforme mobili per l'esplorazione, per le linee di assembleggio, per il trasporto, e macchine che viaggiano autonomamente.

Incentrando l'attenzione sul problema della movimentazione del robot bisogna riferirsi a tutti quei sistemi di localizzazione che permettono al robot di percepire la propria posa.

Per **'posa'** del robot si intende l'insieme delle coordinate che descrivono la sua

posizione in un piano *(x,y)* e il suo orientamento *(θ)*.
La ricerca della configurazione assunta dal robot può essere ottenuta tramite diversi algoritmi di stima e utilizzando procedure in linea o fuori linea. Le prime avvengono durante la movimentazione e si basano su un metodo che seleziona l'azione più opportuna da effetture per meglio adattare la stima della posa; in questo caso quindi il percorso è scelto passo dopo passo in linea al movimento. Si differenziano invece dalle procedure fuori linea che utilizzano un livello simbolico che genera il percorso sulla base della mappa dell'ambiente e delle letture disponibili che vengono analizzate a posteriori e quindi dopo la movimentazione, ovvero non in linea.

La **Localizzazione mobile** consiste nella stima della posa attraverso l'acquisizione e successiva rielaborazione delle informazioni ottenute dall'ambiente circostante. Il termine *Localizzazione* dunque riguarda il problema più generale della determinazione della posizione di un robot mobile attraverso l'analisi di dati ottenuti da sensori.
La maggior parte degli approcci per risolvere tale problema si avvale di un tipo di *localizzazione passiva* dove la posizione del robot corrente viene rivelata tramite letture sensoriali senza sfruttare l'opportunità di controllare gli effettori del robot durante la localizzazione attraverso una serie di azioni scelte in modo opportuno.
Bisogna poi distinguere tra **sistemi di localizzazione assoluta** e **sistemi di localizzazione relativa**. La prima viene ottenuta considerando le misure disponibili istante per istante indipendenti e scorrelate tra loro e in questo caso, la stima della posizione corrente è ottenuta senza integrare quelle passate. Tale procedura non si verifica, invece, per la localizzazione relativa che dipende fortemente dalle posizioni assunte dal robot nel tempo.

Il problema relativo alla localizzazione può essere suddiviso in tre uteriori sotto-problemi, legati fondamentalmente alla località delle assunzioni.

- Position Tracking, ovvero tracciamento della posizione;
- Global Localization, ovvero localizzazione globale;
- Kidnapped Robot Problem, ovvero problema del rapimento.

Il **Position tracking** è la situazione nella quale il robot ha a disposizione una stima della posizione iniziale occupata in un ambiente e successivamente mantiene una traccia della posizione durante gli spostamenti. Esso considera una conoscenza pregressa della posizione e si occupa di tracciarla attraverso l'analisi dei dati sensoriali.
Il **Global Localization** è, invece, la situazione in cui il robot non ha una conoscenza a priori della propria posizione e cerca di stimare la posizione assoluta all'interno dell'ambiente.
Il **Kidnapped Robot Problem**, infine, si verifica quando il robot viene spostato all'interno dell'ambiente senza che tale spostamento sia registrato dai sensori, ad esempio a seguito di uno spostamento manuale da parte di un operatore umano.

Il presente elaborato propone un approccio attivo al problema della ***Localizzazione Globale*** avvalendosi di un criterio razionale di pianificazione del movimento che un robot deve effettuare per fare sì che questo determini la sua posizione nel modo più efficiente possibile e in piena autonomia. Infatti, tramite un controllo attivo per ogni spostamento, il robot riesce in modo intelligente e autonomo a capire quali sono le sequenze di movimenti elementari che deve compiere per meglio localizzare la sua posizione in un ambiente e senza una conoscenza a priori della posizione occupata

all'istante iniziale.

Si tratta di una navigazione basata su un ***approccio probabilistico*** di un robot libero di muoversi in modo indipendente all'interno di un *ambiente 'indoor'* strutturato noto partendo da una qualsiasi posizione iniziale appartenente allo spazio di tutte le configurazioni ammissibili.

La domanda **'Where to move for best positioning?'** è alla base della **Active Localization** e le routine per la localizzazione sono controllate completamente o parzialmente per permettere un metodo di localizzazione più efficiente e robusto.

Si tratta di algoritmi basati sull'intelligenza artificiale dove il controllo è attivo durante l'apprendimento: esso implica, quindi, un'interrogazione attiva con l'ambiente. Scegliere un'azione giusta sulla base di un'interrogazione attiva dell'ambiente durante l'esplorazione permette di raggiungere l'obiettivo riducendo la difficoltà di autolocalizzarsi.

L'Active Localization è un approccio valido empiricamente nel contesto di due problemi di localizzazione:

1. l'**Active Navigation**: risolve la questione di dove muoversi successivamente.
2. l'**Active Sensing**: risolve la questione di quali sensori usare e dove questi devono puntare.

Alla base dell'Active Localization vi è una ricerca euristica della posizione attuale del robot (stima della posizione) definita con il termine **Bel(l)** (*Belief*) e un controllo di apprendimento che permette al robot in esame di seguire una serie di azioni da effettuare per meglio autolocalizzarsi.

Per **Bel(l_k)** si intende la distribuzione di probabilità che esprime la densità di probabilità soggettiva del robot di occupare la posa 'l' all'istante k; **l** rappresenta lo stato

l nel caso unidimensionale o (l_x,l_y) in quello bidimensionale nello spazio di tutte le possibili configurazioni.

Sfruttando un modello metrico dell'ambiente, l'insieme appena definito costituisce una griglia di probabilità di occupare una certa posizione valutando in questo modo la posa assoluta attuale del robot. Il metodo è destinato a funzionare in maniera completamente indipendente dalla conoscenza del punto di partenza, ovvero delle condizioni iniziali.

Si tratta di un **metodo bayesano** basato su delle 'griglie di certezza': in ogni cella di tale griglia viene immagazzinata ad ogni istante durante la simulazione la probabilità Bel(l) che la posizione attuale del robot si trovi nella cella l-esima della griglia.

Queste probabilità sono ottenute integrando e aggiornando le probabilità delle letture del sensore con il passare del tempo e dopo ogni spostamento.

Nell'ambito dell'incertezza i modelli bayesiani discreti sono molto utili per la navigazione dei robot mobili. Questi modelli matematici discreti rappresentano infatti una soluzione ottimale al problema di dove muoversi per meglio percepire la propria posizione. Il problema viene riformulato come processo decisionale markoviano parzialmente osservabile. Tale metodo attivo di localizzazione si basa sulla localizzazione markoviana e sulla successiva elaborazione dell'azione migliore da effettuare durante l'esplorazione; quest'ultima consente di localizzare il robot nella maniera più efficiente possibile (**Active Navigation**).

I problemi di decisione markoviana (*MDPs*, ovvero Partially Observable Markov Decision Process) forniscono i fondamenti per un certo numero di problemi di **AI** (**Intelligenza Artificiale**) che studiano la progettazione e l'apprendimento automatizzato.

La convenienza del metodo implementato è dimostrata empiricamente attraverso nu-

merose simulazioni e confronti per un robot mobile per modelli semplificati.
I risultati esposti in questa tesi dimostrano che la tecnica è risultata robusta nel valutare in modo attendibile la posizione stimata del robot nei casi in cui si hanno a disposizione sensori rumorosi ed errori odometrici sullo spostamento.
Il procedimento di apprendimento attivo risulta particolarmente efficiente per l'autolocalizzazione in ambienti non completamente simmetrici e in presenza di pochi elementi che non permettono al robot di localizzarsi con altre tecniche di localizzazione. Nei casi in cui, invece, tali caratteristiche dell'ambiente vengono meno, il robot non può localizzarsi in modo non ambiguo anche durante la localizzazione attiva; per fare un esempio basta considerare un corridoio completamente simmetrico e con disposizione dei landmarks presenti sempre simmetrica.

1.2 Robotica probabilistica:

La robotica probabilistica ('probabilistic robotics') è un nuovo approccio che tiene conto dell'incertezza nella percezione del robot e delle sue azioni. L'idea chiave è di rappresentare in modo esplicito l'incertezza attraverso l'uso della teoria del calcolo della probabilità. Gli algoritmi probabilistici rappresentano l'informazione sulle distribuzioni di probabilità sull'intero spazio delle configurazioni e possono rappresentare le ambiguità in una via completamente matematica.
La scelta di controllo è fatta in modo tale da ridurre ad ogni spostamento voluto l'incertezza circa la posizione del robot. Gli algoritmi probabilistici permettono di ridurre l'incertezza (o Entropia) della posa assunta dal robot. Per Entropia si intende una quantità numerica associata alla probabilità di stimare correttamente la posizione assunta dal robot: se questa è elevata vuol dire che il robot non sa dove si

trova attualmente, se è prossima a zero, il robot potrebbe conoscere in modo esatto la posizione che occupa.
La Robotica probabilistica opera sulla percezione del robot e sulla pianificazione e controllo delle azioni.
Il problema relativo alla **Localizzazione Globale** (Global Localization) è uno specifico problema di localizzazione nel quale il robot è posizionato in qualsiasi posizione dell'ambiente conosciuto e deve localizzare se stesso. Il paradigma probabilistico si avvale della credenza momentanea del robot (la sopra citata Belief) che rappresenta una funzione di densità di probabilità su tutto lo spazio delle configurazioni ammissibili. Questa densità viene aggiornata ad ogni spostamento e ad ogni lettura sensoriale. Si farà riferimento a due tipi di Belief:

- ***Posterior Belief*** ($P_{posterior}$): ogni qualvolta la belief è aggiornata dopo aver acquisito letture laser o percepito landmark;
- ***Prior Belief*** (P_{prior}): ogni qualvolta la belief è aggiornata dopo aver eseguito uno spostamento, un'azione o subito prima dell'analisi delle letture laser.

Riferendosi ad un esempio semplice, si ha che il robot all'istante iniziale avrà una distribuzione di probabilità uniforme, il che implica la non conoscenza dello stato iniziale: il robot si trova in una qualsiasi posizione. Supponiamo poi che dopo una serie di movimenti il robot si trovi di fronte a una porta, percepisce così una 'feature' porta con una certa probabilità che dipende dal modello del sensore. Le tecniche di navigazione basate su approccio probabilistico sfruttano l'informazione percepita dal sensore landmark per aggiornare la propria Belief. Se nel caso in esame si hanno nell'ambiente tre porte la Posterior Belief si modifica con tre picchi con la stessa probabilità in prossimità delle posizioni delle tre porte e nelle altre posizioni si avrà

probabilità nulla di stare in quella posizione. Questo non significa che il robot conosca la sua posizione con certezza, ma, piuttosto, che ha solamente tre distinte posizioni ipotizzate. A questo punto, se l'ambiente non è simmetrico, le porte sono tra loro distanziate in maniera non omogenea e dunque, una volta che il robot si muove, se dopo una serie di altre azioni elementari percepisce un'altra porta, tenendo conto delle azioni effettuate e dell'aggiornameto delle probabilità, centrerà tutta la probabilità sulla posizione effettiva del robot autolocalizzandosi con certezza (vedi fig. 1.1).
Il problema della localizzazione globale per un robot mobile rappresenta l'idea base delle tecniche di localizzazione markoviana.

1.3 Sensori nei robot mobili

Il disporre di una buona dotazione sensoristica è di fondamentale importanza quando si lavora con piattaforme robotiche. In caso di compiti complessi che richiedono l'interazione con l'ambiente, l'avere a disposizione un'adeguata sensoristica è una precondizione essenziale al fine del successo dell'applicazione. Per fare ciò il robot deve essere a conoscenza dei suoi movimenti, un'informazione che gli viene fornita da una strumentazione capace di acquisire dati riguardo la cinematica e la dinamica del robot; inoltre deve essere anche capace di ottenere informazioni riguardo la struttura metrica e topologica dell'ambiente in cui lavora, in modo da percepire se il robot si trova in una configurazione ammissibile o inammissibile ed eventualmente notare la presenza di possibili ostacoli, come ad esempio i muri o oggetti all'interno dell'ambiente, permettendo così di individuare la propria posizione nell'ambiente.
I robot mobili sono agenti autonomi in grado di muoversi. Essi si sono evoluti nel

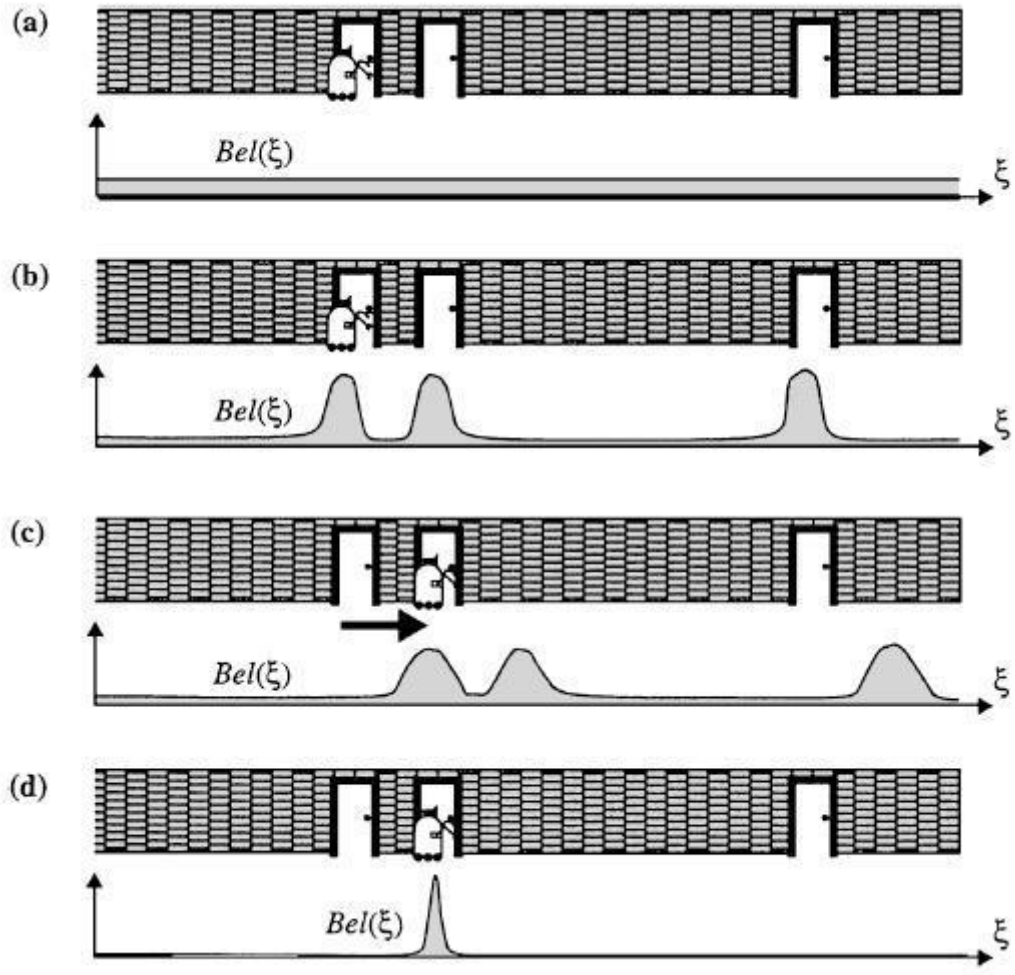

Figura 1.1: Esempio della simulazione di un problema di Localizzazione Globale.

tempo e, ad oggi, sono generalmente dotati di un equipaggiamento sensoriale che consente loro di tenere traccia dei propri spostamenti e di effettuare misure sull'ambiente in cui si muovono.

La parte sensoristica occupa pertanto un ruolo di fondamentale importanza per la soluzione di tutte quelle problematiche che richiedono la localizzazione del robot. Tutti i dati forniti dai sensori che equipaggiano i robot sono purtroppo affetti da errori, che dipendono da fattori non facilmente prevedibili. Questo comporta che tutte le rilevazioni sensoriali nelle simulazioni reali saranno soggette ad incertezze.

Si può fare una prima grande distinzione tra sensori che danno una misura degli spostamenti del robot (misure di posizione relative) e quelli che danno invece una misura di posizione assoluta. Nel primo caso, i dati dei sensori consentono di avere una misura della posizione assoluta del robot solo se è nota la posizione iniziale. L'errore di tale misura tende a divenire sempre più grande con lo spostamento effettuato dal robot. Nel caso di misure di posizione assoluta, invece, l'errore di misura rimane costante e indipendente da quanto il robot si è spostato.

È quindi possibile distinguere i sensori dei robot mobili in due grandi famiglie:

1. **Sensori propriocettivi**
2. **Sensori eterocettivi**

I ***Sensori propriocettivi*** sono quelli che danno informazioni sulla configurazione del robot e sono indipendenti dall'ambiente esterno. Essi riguardano tutti quei dispositivi in grado di fornire dati sulla posizione, sulla velocità e sull'orientamento del robot. In questa categoria rientrano gli encoder e tutti i sistemi di navigazione inerziale come giroscopi e accelerometri. In generale vengono anche definiti *Sensori di spostamento relativo* e rappresentano la classe di sensori che forniscono informazioni relative allo spostamento del robot in un determinato intervallo di tempo.

I ***Sensori eterocettivi***, invece, sono in grado di effettuare misurazioni esterne al robot e quindi fornire dati sulla configurazione rispetto all'ambiente esterno. Tra i sensori appartenenti a tale categoria ricordiamo quelli ad ultrasuoni, ad infrarossi, di visione e laser, a laser rangefinder, i sonar e le telecamere. Per riferirsi ai sensori di tale famiglia si usa anche la terminologia *Sensori di prossimità* per la loro caratteristica di

effettuare misurazioni metriche sugli oggetti presenti nell'ambiente, quale ad esempio la distanza che li separa dal robot.
Se il robot deve interagire con l'ambiente, oltre alle informazioni fornite dai sensori di spostamento relativo è possibile usare, ed è necessario per il mapping, alcune misure esterne: ad esempio la distanza e l'angolazione di ostacoli rispetto al robot.
Tali informazioni vengono di solito fornite come insieme di punti che rappresentano la distanza dell'ostacolo lungo raggi di angolazione differente.
I sensori esterocettivi vengono ulteriormente classificati come:

- attivi
- passivi

La maggior parte dei dispositivi sensoriali utilizzati in robotica appartiene alla prima classe; tale classe, per permettere le misurazioni, emette energia nell'ambiente e, al fine di ottenere le distanze precise tra il robot e gli oggetti, utilizza lo sfasamento o i tempi di eco tra la quantità emessa e quella riflessa.
Quelli definiti passivi, sfruttano invece l'energia che è già presente nell'ambiente. Sensori di questo tipo sono ad esempio i sistemi di visione.
È necessario porre particolare attenzione nella descrizione dei sensori di prossimità attivi, poiché saranno utilizzati nella fase di sperimentazione.

Sensori laser $\longrightarrow$ Sensori attivi:

Come accennato in precedenza i sensori attivi sfruttano una sorgente di energia per irradiare l'ambiente; l'energia riflessa viene analizzata per ottenere misure relative alla posizione occupata all'oggetto. Tali sensori possono essere classificati in base al tipo di energia che viene utilizzata.
Le fonti di energia maggiormente utilizzate in robotica mobile sono:

- quella ***luminosa*** che trova impiego negli scanner laser. In questi dispositivi tale forma di energia utilizzata è rappresentata da un diodo laser.

- quella ***sonora***, molto meno precisa e più lenta rispetto a quella dei laser, che è utilizzata dai sonar. Questi ultimi utilizzano come sorgente di energia un emettitore ad ultrasuoni.

- quella ***elettromagnetica*** utilizzata dai radar.

Nelle applicazioni robotiche che si svolgono all'interno di ambienti al chiuso vengono impiegati fondamentalmente sorgenti di energia appartenenti alle prime due categorie, tenendo conto che, a differenza dei sonar, gli scanner laser permettono di avere un numero di osservazioni molto elevato a parità di energia impiegata. Infatti si può applicare una coppia laser-diodo montata su uno specchio che ruota e se tale specchio ha due gradi di libertà è possibile ottenere informazioni tridimensionali relative all'ambiente di lavoro.
La misurazione effettiva della distanza viene poi ottenuta attraverso l'analisi dell'eco di risposta e si calcola tramite due principali tecniche di misurazione: la prima riguarda la misurazione del tempo di eco; la seconda, invece, la misurazione della differenza di fase tra segnale emesso e segnale riflesso.

Capitolo 2

Localizzazione Attiva

Il presente capitolo descrive, analizza e spiega la teoria alla base dell'algoritmo utilizzato per risolvere il problema relativo alla localizzazione globale di un robot mobile attraverso un controllo attivo; tale controllo permette di raggiungere la stima effettiva della configurazione del robot avvalendosi del principio di minimizzazione dell'Entropia associata alla stima della sua posa.

La localizzazione consiste nel determinare la posizione del robot mobile attraverso dei dati sensoriali. Come anticipato nel capitolo precedente molti approcci a tale problema sono passivi in quanto non sfruttano la possibilità di controllare gli effettori del robot durante la navigazione.

Il presente elaborato propone un approccio di Localizzazione attiva basato sulla minimizzazione dell'Entropia associata alla posa del robot; una tecnica che, attraverso un criterio razionale, si prefigge di controllare la direzione e l'azione del robot durante l'esplorazione, in maniera tale da permettere all'automa di autolocalizzarsi in modo efficiente riducendo l'incertezza associata alla sua posizione. Questo tipo di approccio, come vedremo, permette al robot di raggiungere il suo scopo anche utilizzando letture sensoriali rumorose e modelli approssimati.

L'appropriatezza dell'approccio attivo è dimostrata sperimentalmente usando modelli di robot mobile in ambienti strutturati noti. L'idea di base è di impiegare un ap-

proccio di Localizzazione Markoviana che provveda a stimare in modo accurato la posizione assunta dal robot in modo indipendente dalla conoscenza della sua posa iniziale assunta nell'ambiente.

L'idea Markoviana è quella di mantenere la densità di probabilità (detta Belief) ad ogni istante di tutte le possibili pose che il robot può occupare nel suo ambiente di manovra; tale approccio rappresenta lo spazio metrico sotto forma di una griglia i cui nodi o celle designano tutte le configurazioni che il robot può occupare all'interno dell'ambiente. Ad ogni nodo della griglia, che rappresenta una singola posizione della mappa, viene associata la densità di probabilià che la cella considerata sia occupata o meno dal robot. Questa metodologia che consente di associare la probabilità di occupare una certa posizione prende il nome di 'Position probability grid' e rappresenta una tecnica robusta per la stima della posizione assoluta di un robot mobile.

Passiamo ora alla decrizione dei temi fondamentali della robotica probabilistica che stanno alla base dell'algoritmo implementato.

2.1 Descrizione probabilistica della posa del robot

L'algoritmo assume che il robot abbia inizialmente a disposizione la mappa metrica dell'ambiente ma che non sappia quale posizione occupa.

Un fondamento chiave della robotica probabilistica è il concetto di ***Belief***; la ***Belief*** rappresenta la conoscenza interna del robot del proprio stato, ovvero la percezione di se stesso all'interno dell'ambiente di lavoro.

Spesso nei problemi reali di localizzazione il robot non ha una misura diretta della propria posizione e questa è spesso espressa rispetto ad un sistema di coordinate globali; a meno che il robot, nel caso reale, non abbia a disposizione un dispositivo GPS,

non può conoscere esattamente la sua posa all'istante iniziale e, in quelli successivi, può ottenere solo una stima probabilistica di dove si trova. In questo caso, il robot dovrà cercare la propria posa considerando i dati che ha a disposizione.
Dobbiamo fare una prima distinzione tra stato reale, ovvero posizione vera assunta dell'automa, e credenza interna rispetto allo stato, ovvero conoscenza personale della propria posa, dunque l'**Internal Belief** del robot.
In letteratura sono ***sinonimi di Belief*** altri termini quali **'State of Knowledge'**(Conoscenza dello Stato), o, ancora, **'Information State'** (Informazione dello Stato).
Le probabilità applicate al campo della robotica rappresentano il Belief attraverso distribuzioni di probabilità condizionata; la distribuzione Belief assegna la probabilità, o il valore della densità, ad ogni possibile ipotesi rispetto allo stato reale effettivo del robot. Le distribuzioni Belief rappresentano quindi le probabilità posteriori delle variabili di stato e sono condizionate, modificate e aggiornate, rispetto ad un qualsiasi dato disponibile il che implica che subiscono variazioni a seguito di acquisizione dati dall'ambiente da parte del robot. Nel caso considerato i dati disponibili si ottengono tramite un sensore di prossimità landmarks robot-porta e tramite letture odometriche e, dunque, a seguito di letture sensoriali e per ogni spostamento le Belief di ogni stato subiranno variazioni.
Si denota con il termine $Bel(l_k)$ la Belief, ovvero la densità di probabilità all'istante k su tutte le singole pose l_k appartenenti all'intero spazio di tutte le posizioni che il robot può assumere all'interno dell'ambiente; queste singole l_k, nel caso unidimensionale, assumono un valore x_k compreso nell'intervallo $x_k \in [1, L_x]$ per ogni possibile configurazione, mentre nel caso bidimensionale sono espresse, per ogni posa $l(k)$, dall'insieme di due coordinate $l_k = (x_k, y_k)$ con $x_k \in [1, L_x]$ e $y_k \in [1, L_y]$.

La l_k rappresenta la posa assunta dal robot espressa in un sistema di riferimento globale; fisicamente questo occupa un'unica posizione $l_r = \bar{l}_k$ all'interno dell'ambiente all'istante k e con Bel(l_k) esso tiene internamente conto solo della credenza di dove è localizzato. Il problema della localizzazione è di approssimare nel modo migliore possibile la posa vera del robot, mediante una distribuzione di probabilità che avrà un singolo picco in corrispondenza alla posa reale del robot e zero altrove. La densità di probabilità posteriore, a seguito di acquisizione dati, viene così ricalcolata:

$$Bel(l_k) = P(l_k|s_{1:k}, a_{1:k}) \quad \longrightarrow P_{posterior}(l_k) \tag{2.1.1}$$

Questa distribuzione di probabilità posteriore sull'intero stato delle possibili configurazioni l(k) assunte dal robot al tempo k è condizionata da tutte le $s_{1:k}$ misure sensoriali acquisite negli istanti da 1 a k e alle k azioni $a_{1:k}$ di controllo effettuate in precedenza. In essa si tiene conto di tutti i dati possibili fino all'istante k.
Per il calcolo di Bel(l_k) abbiamo assunto che il 'Belief state' è stato ottenuto incorporando tutte le k misure precedenti; occasionalmente, si può calcolare la P_{prior} prima di incorporare l'ultima lettura disponibile s_k e subito dopo aver effettuato il k-esimo movimento a_k.
Questa Belief assumerà la seguente forma:

$$\widehat{Bel}(l_k) = P(l_k|s_{1:k-1}, a_{1:k}) \quad \longrightarrow P_{prior}(l_k) \tag{2.1.2}$$

Quindi si ha la seguente distinzione: Bel_{prior} è la Belief prima di ottenere le letture sensoriali e la $Bel_{posterior}$ è la Belief solo dopo averle incorportate.
La Bel_{prior} è la distribuzione di probabilità che si riferisce spesso alla predizione della Belief nel contesto del filtraggio probabilistico[1]. Essa rappresenta la predizione Belief dello stato ('prior Belief'= $\widehat{Bel}(l_k)$) all'istante k basata sulla precedente Belief dello

[1]Verrà infatti utilizzata per l'algoritmo implementato nella fase di predizione.

stato all'istante $k-1$ ottenuta dalla 'posterior Belief'($Bel(l_{k-1})$) dopo un'azione a_k ma poco prima di incorporare l'ultima misurazione s_k. Calcolare $\widehat{Bel}(l_k)$ da $Bel(l_{k-1})$ è detta predizione; calcolare poi Bel(l_k) da $\widehat{Bel}(l_k)$ è detta *correzione* o *aggiornamento delle misure*.

Inizialmente per il $k = 0$ la Belief che riflette l'incertezza nella posa del robot è detta $Bel_{prior}(l_0)$; il robot ha una prima credenza di quale dovrebbe essere la sua posizione. Questa, come accenato precedentemente, se il robot conosce con esattezza la sua posa, costituisce una distribuzione di probabilità tutta centrata in un punto il cui picco rappresenta la reale posizione assunta dal robot all'istante di partenza; in questo caso lo scopo della localizzazione è di compensare gli errori dovuti alla movimentazione risolvendo il classico problema della **Position Tracking**. Al contrario nel caso in cui il robot non conosca assolutamente la propria posizione iniziale, la $Bel_{prior}(l_0)$ è uniformemente distribuita; questa situazione corrisponde alla condizione iniziale che si ha in tutti i problemi relativi alla **Localizzazione Globale** e all'**Autolocalizzazione**. Più difficile di questi ultimi risulta il problema relativo al **Robot Kidnapping** in quanto, in questo caso, la densità ha un picco che però non è in corrispondenza della posizione vera.

2.2 Approccio Bayesiano e derivazioni matematiche del filtro per l'aggiornamento della probabilità

2.2.1 Inferenza Bayesiana

L'inferenza statistica è il procedimento per cui si traggono tramite induzione le caratteristiche di una popolazione dall'osservazione di una parte di essa detta campione, selezionata solitamente mediante un esperimento aleatorio. Alla base dell'inferenza vi

sono tutte quelle tecniche matematiche per quantificare il processo di apprendimento ottenuto tramite l'esperienza o da dati sperimentali.
L'inferenza statistica può essere suddivisa in due grandi famiglie, ciascuna delle quali è legata a diverse concezioni, o interpretazioni, del significato della probabilità:

1. Inferenza classica, detta anche Frequentista, legata a R.Fisher e K. Pearson;

2. Inferenza Bayesiana formulata in base al noto teorema di Bayes.

La principale differenza tra i due approcci sta nel fatto che mentre la prima fa riferimento ad affermazioni su quante volte si dice il vero, alla base della seconda, invece, si attribuisce la probabilità di verità di dire il vero, ovvero formalizza l'idea che si ha su come potrebbe essere probabilmente il vero. Inoltre l'approccio Bayesiano è in grado di utilizzare tutte le informazioni già in possesso in modo da modificare la probabilità che si ha a priori di un certo evento con la probabilità che si ottiene a posteriori solo dopo aver incorporato tutti i dati a disposizione.

Approfondiamo ora l'inferenza Bayesiana e le basi del teorema a cui si riferisce; questa verrà infatti utilizzata per affrontare il problema in esame.
Il ruolo del teorema di Bayes è di fondamentale importanza poichè tutta l'inferenza Bayesiana si fonda su di esso.

Teorema di Bayes:

Considerando un insieme di alternative $A_1, A_2, .., A_n$ che partiziona lo spazio di tutti gli eventi possibili, si trova la seguente espressione per la probabilità condizionata:

$$P(A_i|E) = \frac{P(E|A_i)P(A_i)}{P(E)} = \frac{P(E|A_i)P(A_i)}{\sum_{j=1}^{n} P(E|A_j)P(A_j)} \tag{2.2.1}$$

Dove i vari termini della formula esprimono:

- P(A) è la probabilità a priori o probabilità marginale di A. 'A priori' significa che non tiene conto di nessuna informazione riguardo E.
- P(A|E) è la probabilità condizionata di A, noto E. Detta probabilità a posteriori e dipendente dallo specifico valore di E.
- P(E|A) è la probabilità condizionata di E, noto A.
- P(E) è la probabilità a priori di E, e funge da costante di normalizzazione.

Intuitivamente il teorema descrive il modo in cui le opinioni nell'osservare A siano arricchite dall'aver osservato l'evento E; i due tipi di informazione menzionate, a priori e a posteriori, sono quantificate attraverso due distribuzioni di densità di probabilità, dette rispettivamente priori e verosimiglianza. Si noti che i parametri della formula, in quanto dotati di distribuzione, sono considerati variabili casuali.
Nelle procedure Bayesiane l'informazione espressa dalla priori è modificata dall'informazione contenuta nel campione osservato impiegando il teorema di Bayes. L'inferenza finale viene ottenuta dalla distribuzione posteriore del parametro aggiornata e condizionata dai dati osservati. La priori rappresenta invece la traduzione matematica dell'informazione extra-campionaria rilevante e disponibile; quest'ultima può considerare informazioni tecniche, opinioni, feed-back del processo, e includere risultati di esperimenti precedenti; essa implica una natura soggettivista. Dopo aver definito la priori ed avendo il campione disponibile si può ottenere la distribuzione a posteriori, la quale aggiorna l'informazione sul parametro contenuta nella priori attraverso l'informazione distributiva rilevata nel campione che dipende da dati indipendenti al campionare stesso (come può essere la percezione di un landamark). La posteriori fornisce una espressione completa dell'inferenza sul parametro e si basa spesso su una

o più misure riassuntive.

2.2.2 Formulazione del Filtro Bayesiano

Una volta introdotta la teoria alla base dell'approccio Bayesiano e definito nel primo paragrafo il concetto di credenza Belief, si passa ora alla descrizione dell'algoritmo implemento per la stima della posa ottenuta tramite il filtro Bayesiano discreto.
La maggior parte degli algoritmi per il calcolo della Belief utilizzano, infatti, l'approccio Bayesiano e le derivazioni matematiche alla base del filtro. L'algoritmo calcola la distribuzione di probabilità, Belief, detta per semplicità Bel, dalle misure e dalle azioni effettuate.
Si tratta di un filtro ricorsivo, il quale permette di calcolare al tempo k il Belief Bel(l_k) dal Bel(l_{k-1}) al tempo k-1. Viene qui riportato lo schema generale del filtro Bayesiano discreto in pseudocodice.

- **Algoritmo Bayesiano**$(Bel(l_{k-1}), a_k, s_k)$
- for all $l_k \in$[insieme degli stati possibili] do:
- $\widehat{Bel}(l_k) = \sum_{l_{k-1}} P(l_k|a_k, l_{k-1})Bel(l_{k-1})$
- $Bel(l_k) = P(s_k|l_k)\widehat{Bel}(l_k)$
- $Bel(l_k) = Bel(l_k)/p(s_k)$
- end for
- return $Bel(l_k)$

Il filtro Bayesiano discreto si applica in problemi con uno spazio di stato finito, infatti la variabile casuale l_k può assumere solo un numero finito di valori. Essa rappresenta uno stato individuale e quindi, se si suddivide l'ambiente di lavoro in un insieme di nodi o celle finito, l_k appartiene a tale insieme. In definitiva Belief al tempo k è la probabilità di stare in una certa posizione per ogni stato l_k la cui singola probabilità è appunto $Bel(l_k)$. Dall'algoritmo Bayesiano appena riportato si può vedere che tale filtro ha in ingresso la Belief all'istante k-1 e i soli dati più recenti, poi, dopo averli processati, fornisce in uscita l'aggiornamento della Belief secondo una regola ricorsiva. L'algoritmo del filtro Bayesiano possiede tre punti chiave: la fase di predizione, l'aggiornamento delle misure e la condizione al contorno.

1. Nella prima si processa l'azione di controllo a_k per la stima di $\widehat{Bel}(l_k)$ e lo si fa calcolando la Belief di tutti gli stati l_k basata sulla 'Prior Belief' $\widehat{Bel}(l_k)$ su tutti gli stati l_{k-1} e l'ultima azione di controllo. Per spiegare meglio, la $\widehat{Bel}(l_k)$ si ottiene integrando ($\sum$) la produttoria di due distinte distribuzioni: la probabilità $Bel(l_{k-1})$ per la probabilità che l'azione a_k porti lo stato l_{k-1} nello stato l_k. A livello teorico essa si riferisce al calcolo matematico della densità di probabilità spiegata nel paragrafo sull'inferenza Bayesiana.

2. Nella seconda, l'algoritmo moltiplica la probabilità $\widehat{Bel}(l_k)$ con la probabilità $P(s_k|l_k)$ che sia stata osservata la misura s_k nello stato l_k. Tale probabilità deve essere normalizzata perché spesso la sua produttoria se integrata non restituisce un valore unitario; in questo caso si fa riferimento al calcolo matematico della densità di probabilità a posteriori o di verosomiglianza del paragrafo precedente.

3. Per funzionare ricorsivamente l'algoritmo ha bisogno della condizione iniziale $Bel(l_0)$ all'istante $k = 0$. Se il robot conosce esattamente la propria posizione iniziale la $Bel(l_0)$ centra la sua distribuzione su di un unico valore che è quello corretto della posizione reale assunta dal robot mentre assegna a tutte le altre posizioni una probabilità nulla. Se, al contrario, il robot ignora completamente la conoscenza della propria posizione, la $Bel(l_0)$ all'istante $k = 0$ è uniformemente distribuita su tutto il dominio l_0. Una conoscenza parziale dello stato iniziale può essere espressa da una distribuzione non uniforme. Quest'ultimo caso è di poco interesse in quanto nella realtà è più frequente che il robot o conosce esattamente la propria posizione, o non la conosce affatto.

La derivazione del filtro dall'applicazione della regola di Bayes:

La regola di Bayes, per il caso corrente viene formalizzata nel seguente modo:

$$\begin{aligned} P(l_k|s_{1:k}, a_{1:k}) &= \frac{P(s_k|l_k, s_{1:k-1}, a_{1:k})P(l_k|s_{1:k-1}, a_{1:k})}{P(s_k|s_{1:k-1}, a_{1:k})} \\ &= \frac{P(s_k|l_k, s_{1:k-1}, a_{1:k})P(l_k|s_{1:k-1}, a_{1:k})}{P(s_k)} \end{aligned} \tag{2.2.2}$$

Assunzione 1: lo stato l_k è COMPLETO $\longrightarrow$ derivazione matematiche

Le derivazioni matematiche del filtro Bayesiano per l'aggiornamento di Belief richiedono che lo stato l_k sia completo, cioè che nessuna variabile prima di l_k possa influenzare l'evoluzione stocastica degli stati futuri; tale assunzione implica che se conosciamo lo stato l_k e siamo interessati nella predizione della misura s_k, nessuna misura e azione di controllo passata può fornire informazioni aggiuntive allo $Bel(l_{k-1})$. In termini matematici, questa condizione può essere espressa nella seguente condizione di indipendenza:

$$P(s_k|l_k, s_{1:k-1}, a_{1:k}) = P(s_k|l_k) \tag{2.2.3}$$

È un condizione che permette di semplificare la formula di Bayes in:

$$P(l_k|s_{1:k}, a_{1:k}) = \frac{P(s_k|l_k)P(l_k|s_{1:k-1}, a_{1:k})}{P(s_k)} \tag{2.2.4}$$

Ricordando che $Bel(l_k) = \frac{P(s_k|l_k)\widehat{Bel}(l_k)}{P(s_k)}$, si ha:

$$P(l_k|s_{1:k}, a_{1:k}) = \frac{P(s_k|l_k)P(l_k|s_{1:k-1}, a_{1:k})}{P(s_k)} = Bel(l_k) \tag{2.2.5}$$

$$\frac{P(s_k|l_k)P(l_k|s_{1:k-1}, a_{1:k})}{P(s_k)} = Bel(l_k) = \frac{P(s_k|l_k)\widehat{Bel}(l_k)}{P(s_k)} \tag{2.2.6}$$

Espandendo la $\widehat{Bel}(l_k)$ si ha:

$$\begin{aligned} \widehat{Bel}(l_k) = P(l_k|s_{1:k-1}, a_{1:k}) &= \sum_{l_{k-1}} P(l_k|l_{k-1}, a_k)Bel(l_{k-1}) \\ &= \sum_{l_{k-1}} P(l_k|l_{k-1}, a_k)P(l_{k-1}|s_{1:k-1}, a_{1:k}) \end{aligned} \tag{2.2.7}$$

Sfruttando ancora l'assunzione che lo stato sia completo questo implica che anche per l_{k-1} le misure e le azioni di controllo passate non influenzino le informazioni riguardanti l_k.

Dunque si ha:

$$P(l_k|l_{k-1}, s_{1:k-1}, a_{1:k}) = P(l_k|l_{k-1}, a_k) \tag{2.2.8}$$

Assunzione 2: le azioni di controllo sono scelte in modo indipendente dal l_k $\longrightarrow$ derivazione matematiche

Questa seconda assunzione permette di rendere ricorsivo l'algoritmo. Infatti considerando che l'azione a_k non può essere predetta dallo stato l_{k-1} ed è un'informazione inutile per lo stato l_{k-1}, la $\widehat{Bel}(l_k)$ diventa:

$$\widehat{Bel}(l_k) = \sum_{l_{k-1}} P(l_k|l_{k-1}, a_k) P(l_{k-1}|s_{1:k-1}, a_{1:k-1}) \tag{2.2.9}$$

[Equazione di aggiornamento ricorsiva.]

In poche parole, il filtro di Bayes calcola la Belief dello stato l_k condizionata dalle misure s_k e azioni a_k acquisite fino all'istante k. Tale derivazione matematica assume un modello Markoviano, cosa che è immediatamente verificata tramite l'assunzione che lo stato sia completo.

La definizione infatti di processo di Markov implica che, con riferimento ad un processo stocastico, la distribuzione di probabilità degli stati futuri di tale processo, noti gli stati passati, dipende solo dallo stato attuale considerato. Il fatto che lo stato l_k sia un processo Markoviano, implica quindi l'indipendenza dello stato dalle misure e dai controlli effettuati in precedenza, dunque rientra nel caso dell'assunzione 1 che ci ha permesso di ottenere le derivazioni matematiche del filtro.

Il modello discreto del fitro Bayesiano è molto popolare in diverse aree di processamento dei segnali, dove si supera di un passo in avanti l'HMM, ovvero l'Hidden Markov Model.

In definitiva ogni implementazione di questo filtro richiede la definizione di 3 distribuzioni di probabilità:

1. Belief iniziale: Bel(l_0)
2. La probabilità di acquisire le misure: $P(s_k|l_k)$
3. La probabilità della transizione tra gli stati: $P(l_k|l_{k-1}, a_k)$

2.3 Modello Markoviano per la transizione degli stati e Modello sensoriale utilizzato

In questo paragrafo affronteremo il problema relativo alla Localizzazione del robot mobile. Alla base di tale problema sta la ricerca della posa di un robot rispetto alla mappa metrica di un ambiente noto. Riferendosi a tale problema si usa spesso chiamarlo il problema della stima della posizione; per raggiungere tale scopo si hanno a disposizione i dati sensoriali che il robot percepisce dall'ambiente e i movimenti che esso compie.

Mentre le mappe vengono descritte in base ad un sistema di coordinate globale le letture sensoriali e i movimenti sono relativi alla posizione attualmente occupata dal robot.

Gli algoritmi di localizzazione probabilistica sono delle varianti del filtro Bayesiano. L'applicazione diretta di tale filtro la si ottiene per modelli di localizzazione di tipo Markoviano. Viene riportato di seguito l'algoritmo base del modello markoviano.

- **Algoritmo Localizzazione_Markoviana**(Bel(l_{k-1}),a_k,s_k,M)
- for all $l_k \in$[insieme degli stati possibili] do:
- $\widehat{Bel}(l_k) = \sum_{l_{k-1}} P(l_k|a_k, l_{k-1})Bel(l_{k-1})$
- $Bel(l_k) = P(s_k|l_k, M)\widehat{Bel}(l_k)$
- $Bel(l_k) = Bel(l_k)/p(s_k)$
- endfor
- return $Bel(l_k)$

Per applicare tale algoritmo è necessario che l'ambiente abbia le seguenti caratteristiche: deve essere statico ed avere oggetti statici. Inoltre il set dei possibili stati deve essere finito e lo deve essere anche il numero di azioni che il robot può compiere. Si può vedere come tale algoritmo sia molto simile all'algoritmo del filtro Bayesiano, l'unica differenza, infatti, si ha per l'introduzione di un elemento in più nell'algoritmo. In ingresso all'algoritmo è necessario il dato M; esso rappresenta la mappa dell'ambiente. M rappresenta un dato fondamentale nel modello sensoriale che descrive la $P(s_k|l_k, M)$ e, spesso, viene anche incorporato nel modello di movimentazione $P(l_k|a_k, l_{k-1}, M)$. Come avviene per il filtro di Bayes, la Localizzazione Markoviana trasforma e aggiorna la probabilità Belief Bel(l_{k-1}) al tempo k-1 nella Belief Bel(l_k) al passo successivo k. Tale tecnica di localizzazione risolve il problema della Global Localization, la Position Tracking e il Kidnapped Robot Problem in ambienti statici. Alla base di questa tecnica di localizzazione si ha l'aggiornamento della Belief.
La Bel(l_k) è aggiornata ogni volta che il robot percepisce delle letture sensoriali e ogni volta che compie un movimento.

Modello di movimentazione:

Il movimento del robot è modellato dalla probabilità condizionata $P(l_k|a_k, l_{k-1})$ che per semplicità chiameremo $P_a(l|l')$ con $l = l_k$, $l' = l_{k-1}$ e $a = a_k$.
La $P_a(l|l')$ denota la probabilità che se il robot effettua l'azione **'a'** quando si trova nella posizione l' tale azione eseguita lo porterà nel nuovo stato l.
Nel nostro caso, come vedremo nel terzo capitolo, **'a'** assume proprio il comando che il robot deve compiere, ovvero $a =$ 'Muoviti di ..'. $P_a(l|l')$ permette l'aggiornamento di Bel(l) rispetto a Bel(l') secondo la seconda linea dell'algoritmo della localizzazione di Markov appena riportato, dunque nel caso di **'l'** appartenente ad un numero discreto

e finito di posizioni si ha: $Bel(l) = \sum_{l'} P_a(l|l')Bel(l')$.

$P_a(l|l')$ è ottenuto dal modello cinematico del robot ed è rappresentabile nei casi esaminati, 1D e 2D, tramite un modello di tipo markoviano per la transizione degli stati. Per indenderci meglio, il modello cinematico del robot è stato pensato in entrambi i casi come una catena di Markov.

Considerata 'a =(vai 1 passo in avanti)' l'azione che il robot deve compiere rispetto ad un certo asse di riferimento; se **l'** è la posizione[2] che il robot occupa attualmente, si avranno, secondo il modello, le seguenti transizioni per lo stato di partenza **l'**:

1. $l_1 = l'$ con una certa probabilità p_1
2. $l_2 = l' + a$ con una certa probabilità p_2
3. $l_3 = l' + 2a$ con una certa probabilità p_3

Con il seguente vincolo valido: la sommatoria delle tre probabilità deve dare sempre 1 ($p_1 + p_2 + p_3 = 1$).

In questo caso si rispetta sempre l'assunzione relativa allo stato l, ovvero che sia completo; la catena di Markov rispetta tale proprietà, infatti è senza memoria. Inoltre è omogenea, infatti, ad ogni istante k estraiamo dal processo una variabile casuale discreta l_k nel quale la probabilità di transizione $l_{k-1} \longrightarrow l_k$ dipende unicamente dallo stato del sistema immediatamente precedente l_{k-1} e non anche dal tempo k, e tale probabilità di passare da uno stato ad un altro è costante nel tempo per ogni differente azione **'a'** considerata.

[2]Si tratta di uno stato che appartiene allo spazio di stato di tutte le possibili configurazioni che può assumere l'automa nell'ambiente

Modello Sensoriale:

Il modello di misura nell'ambiente rappresenta il secondo dominio specifico della robotica probabilistica. I modelli di misura descrivono come le informazioni percepite dall'ambiente vengono processate dal robot tramite l'acquisizione sensoriale. Il modello specifico si riferisce al tipo di sensore di cui si dispone.

Nella robotica probabilistica il rumore presente nelle letture sensoriali viene esplicitamente descritto dal modello. In maniera formale il modello di misurazione è descritto tramite la distribuzione di probabilità condizionata $P(s_k|l_k, M)$ che viene poi utilizzata nell'agoritmo per l'aggiornamento all'istante k della Bel(l_k). Abbiamo appena visto che la Belief Bel(l) viene aggiornata, a seguito dell'aquisizione delle letture sensioriali, dai dati percepiti dall'ambiente in cui il robot si trova ad operare. Incorporare tali dati nella Belief è molto importante per ottenere l'inferenza finale sulla densità di probabilità Bel(l) che descrive la probabilità di trovarsi in una certa posizione.

Per generare certe misure, bisogna avere una descrizione dell'ambiente nel quale tali misure sono effettuate.

La **MAPPA dell'ambiente** è una lista degli oggetti appartenenti all'ambiente e la posizione che occupano all'interno di esso e spesso riportante le loro caratteristiche.

$M = [m_1, m_2, .., m_N]$ ed N è il numero di oggetti-features considerati.

Ci sono due possibili strade per rappresentare la **mappa M**:

1. 'feature-based': in queste mappe la descrizione del singolo oggetto m_n è fatta rispetto all'indice n con riferimento al n-esimo landmark, mentre il valore m_n descrive le sue caratteristiche e la posizione che occupa in coordinate cartesiane.

2. 'location-based': in queste mappe, invece, l'indice n corrisponde alla specifica posizione che occupa il landmark all'interno dell'ambiente; nel caso bidimen-

sionale, ad esempio, si utilizza l'elemento $m_{x,y}$ al posto di m_n per esplicitare che l'oggetto si trova in posizione (x,y) rispetto ad un sistema di assi cartesiani (X,Y).

Nel caso 1D che verrà simulato, l'ambiente è visto secondo la feature-based: infatti la mappa dell'ambiente è vista come $M = [Porta_1, Porta_2, ..]$ e ogni elemento $Porta_n$ occupa una sua posizione (x_n) all'interno della mappa. Quindi il landmark n-esimo è visto come l'n-esima feature $Porta_n$ che occupa una certa posizione rispetto all'asse delle coordinate cartesiane X.
Sempre nel caso monodimensionale le misure rilevate dal sensore sono solo del tipo *'vedo'/'non vedo'* la porta, ON/OFF; infatti se indichiamo con s_k la lettura sensoriale all'istante k essa può assumere due unici valori $s_i \in [0, 1]$. Il robot avrà la probabilità di leggere una certa **'s'** in maniera corretta o scorretta secondo il seguente modello sensoriale:

- $P(s_k = 1|l_k \in M)$: associa la probabilità di vedere CORRETTAMENTE il landmark
- $P(s_k = 0|l_k \in M)$: associa la probabilità di NON vedere ERRONEAMENTE il landmark
- $P(s_k = 1|l_k \notin M)$: associa la probabilità di vedere ERRONEAMENTE il landmark
- $P(s_k = 0|l_k \notin M)$: associa la probabilità di NON vedere CORRETTAMENTE il landmark

Per il caso bidimensionale, il modello sensoriale è leggermente diverso. In effetti si potrebbe dire che già l'ambiente è visto secondo una **'Occupancy grid map'** che analizza ogni singola posizione e si basa su una mappa di tipo location-based; qui ogni elemento $Porta_n$ è visto come la presenza del landmark in un insieme di celle $(x_n, y_n) \in R_{n,t}$. Infatti il robot riesce a percepire la porta solo se si trova in una zona limitrofa a tale feature, ovvero in R_t, all'interno dell'ambiente. La dimensione di tale zona è determinata dalla potenza dallo scanner laser a disposizione. La probabilità $P(s_k|l_k, M)$ in questo caso decresce con la distanza robot-landmark $(dist_r)$; se ad esempio il robot è molto vicino alla porta si ha un'alta probabilità che esso ne percepisca l'esistenza, se si allontana tale probabilità diminuisce al crescere della distanza. Matematicamente, il modello sensoriale $P(s_k|l_k, M)$ che rappresenta la densità di probabilità di ottenere certe letture **'s'** è descritto da una *distribuzione esponenziale con parametro* $dist_r$.

2.4 Definizione di Entropia associata alla descrizione probabilistica della posa del robot

Il concetto di **'Entropia associata alla posa'** esprime l'incertezza che il robot ha sulla propria posa.

Viene detta Entropia della Belief e si ottiene con la seguente formula discreta[3]:

$$H_k = -\sum_{l_k=1}^{N_l} Bel(l_k) log(Bel(l_k)) \tag{2.4.1}$$

N_l è un numero finito e corrisponde alla dimensione dell'insieme di tutte le pose

[3]Il set del pose è discreto.

che il robot può assumere all'interno dell'ambiente. L'Entropia viene anche chiamata *'grado di incertezza'*.

Tramite il calcolo dell'Entropia H_k all'istante k si ha già una misura dell'incertezza della posizione del robot. Consideriamo il caso in cui si abbia ad un certo istante k della simulazione un valore nullo per H_k, ovvero $H_k = 0$, questa situazione implica che il robot non è più incerto della propria posa e, quindi, all'istante k la percepisce con assoluta certezza. In questo caso, facendo uno studio sulla distribuzione di probabilità Belief, si vede che Bel(l_k) è incentrata in un unico punto $\bar{l}$ che corrisponde ad una singola posa che il robot percepisce di se stesso, se poi questa corrisponde alla posizione reale corrente assunta dall'automa si ha una coincidenza esatta tra la posizione stimata e quella effettiva e l'errore di stima va a zero; il robot è sicuro di occupare una certa posizione all'interno dell'ambiente e questa è la posizione che occupa effettivamente nella realtà. È questa la condizione che si verifica più spesso nel caso in cui si ha una diminuzione dell'Entropia in quanto tale diminuzione è spesso associata ad una corrispondenza diretta tra informazioni acquisite dal robot durante la navigazione e posizione reale da lui assunta.

Se, al contrario, la H_k assume il valore massimo possibile, il robot non ha alcuna informazione circa la posa e non ha alcuna certezza di quale posizione possa occupare all'interno dell'ambiente. Tale circostanza si ha per Bel(l_k) distribuita in maniera uniforme sull'intero spazio delle configurazioni, infatti in questo caso il robot può occupare ciascuna posizione con uguale probabilità; questa situazione si verifica sempre all'istante iniziale $k = 0$ nel caso del problema della localizzazione globale perché il robot non ha alcuna certezza circa il proprio stato iniziale l_0.

Se con il passare del tempo l'Entropia aumenta vuol dire che si sta perdendo traccia della posizione reale del robot e questo ha una percezione della sua posa meno certa;

se al contrario l'Entropia diminuisce, il robot tende a percepire in modo più corretto la propria posa.

Se il robot non controlla gli attuatori in modo opportuno, ma effettua dei movimenti casuali, si avrà quasi sicuramente che l'Entropia ad ogni passo della simulazione può aumentare o diminuire anch'essa in modo aleatorio, senza portare alla stima della posizione certa. Per questo motivo si è cercato di risolvere il problema della localizzazione attraverso una selezione accurata dell'azione di controllo che rendesse possibile diminuire l'Entropia ad ogni istante k della simulazione permettendo così di avere ad ogni passo una maggiore certezza nella stima. Tale tecnica di controllo risolve il problema della Global Localization in modo attivo. Nel prossimo paragrafo illustreremo l'algoritmo di controllo attivo.

2.5 Selezione attiva dell'azione migliore e tipologia di controllo

L'**active localization** riguarda uno dei problemi fondamentali dell'esplorazione robotica: il robot investiga costantemente per determinare la sua posizione durante la localizzazione; lo scopo principale è dunque quello di massimizzare le informazioni sulla posizione assunta dall'automa nel tempo. A tale massimizzazione è associata una riduzione dell'incertezza che è rappresentabile dall'Entropia, precedentemente definita, la quale descrive infatti il grado di incertezza sulla posa del robot.

Il concetto di Entropia appena introdotto nel paragrafo precedente è alla base del metodo della localizzazione attiva basato sulla minimizzazione dell'Entropia. L'idea princiale di questa tecnica è di eliminare l'incertezza della posizione stimata attraverso

il calcolo di Bel(l), per fare ciò il robot deve scegliere delle azioni che lo aiutino a distinguere la posizione occupata tra le differenti configurazioni che potrebbe occupare all'interno dell'ambiente.

Il concetto chiave del problema può essere riformulato nella seguente maniera: *le azioni sono selezionate in modo da minimizzare la futura Entropia attesa dopo lo spostamento.*

Per ottenere la predizione dell'Entropia a seguito di un'azione di controllo, l'$E_a(H)$, si devono introdurre due variabili ausiliarie che erano già state utilizzate per implementare il filtro Bayesiano; questa volta però si tratta di Belief attese e non effettive e possono essere facilmente calcolate dalle equazioni di aggiornamento Markoviano della posizione[4]. Vengono qui riportate e specificate per comodità:

1. la $Bel_a(l)$[5] è la prima e corrisponde alla Belief che si otterebbe dopo aver eseguito l'azione **'a'**.

2. la $Bel_{a,s}$[6] è la seconda e corrisponde alla Belief che si avrebbe dopo aver eseguito l'azione **'a'** e percepito la lettura sensoriale **'s'**.

Una volta introdotte queste variabili si può passare alla stima dell'Entropia attesa $E_{a,s}(H)$ che dipende sia dall'azione **'a'** che dal dato sensoriale **'s'**, e trova la sua espressione analitica nella seguente formula matematica:

$$E_{a,s}(H) = -\sum_{l=1}^{N_l} Bel_{a,s}(l) log(Bel_{a,s}(l)) \tag{2.5.1}$$

[4]Si fa riferimento alla localizzazione Markoviana
[5]Si tratta della già nota $P_{prior}(l)$.
[6]Si tratta della già nota $P_{posterior}(l)$.

Con N_l si intende il numero delle posizioni, o celle, in cui è stata decomposta la mappa dell'ambiente.

Da questa si può ottenere l'espressione dell'Entropia attesa $E_a(H)$ dopo aver eseguito l'azione **'a'**, o per meglio dire, l'Entropia che ci si aspetterebbe se il robot decidesse di fare proprio quella azione **'a'**; tale grado di incertezza futura sulla posa del robot si ottiene integrando tutte le possibili letture sensoriali **'s'** pesate per la loro probabilità di accadere.

$$E_a(H) = \sum_s E_{a,s}(H)p(s) = -\sum_s \sum_{l=1}^{N_l} Bel_{a,s}(l)log(Bel_{a,s}(l))p(s) \tag{2.5.2}$$
$$= -\sum_s \sum_{l=1}^{N_l} P(s|l)Bel_a(l)log(\frac{P(s|l)Bel_a(l)}{p(s)})$$

Il metodo della localizzazione attiva si avvale del principio della minimizzazione dell'Entropia attesa $E_a(H)$ tramite una tecnica greedy.

'Active Navigation:'

La Navigazione attiva risolve il problema di dove muoversi per permettere una stima migliore della propria posizione.

Il controllo degli attuatori del robot è fatto in modo tale da far eseguire all'automa azioni base del tipo 'muoviti di 1 metro in avanti'.

La selezione dell'azione migliore da compiere è data dalla tecnica di *'Active Selection'*. Partendo dalla definizione dell'Entropia attesa già discussa e dal concetto di costo (C(a)) che viene associato alla distanza del cammino effettuato, si può esplicitare il criterio di selezione dell'azione nella localizzazione attiva. Tale politica, ad ogni istante di tempo, permette al robot di scegliere l'azione più intelligente da compiere

secondo il seguente criterio di selezione:

$$a^* = argmin_a(E_a(H) + \alpha C(a)) \tag{2.5.3}$$

La scelta di $\alpha \geq 0$ dipende dall'applicazione considerata; nel nostro caso è stata posta a zero perché non si vuole dare un peso al costo dell'azione e al cammino eseguito, questo per due motivi particolari: il primo è che l'obiettivo del controllo che si vuol eseguire ha come unico scopo nella funzione obiettivo la sola minimizzazione dell'Entropia e quindi non ci interessa il numero delle azioni o degli spostamenti necessari per raggiungere l'incertezza minima sulla posa; il secondo fatto è che l'azione che si compie, in questo caso greedy, è singola e quindi non avrebbe neanche senso calcolarne il costo dato che sarà unitario e uguale per l'insieme delle possibili azioni elementari che si possono scegliere nel controllo.

Il costo può assumere invece notevole importanza se si applica un controllo attivo basato sulla scelta di azioni che non sono più elemetari ma del tipo 'a = Muoviti di 10 passi in avanti e 2 a sinistra'. Questo tipo di controllo in alcuni casi risulta più efficiente di quello greedy in quanto riesce a superare il limite di quest'ultimo, cioè l'eventualità che l'Entropia abbia un punto di mininimo relativo e il robot tenda a tornare in una posizione remota, precedente, senza risolvere il problema.

La politica di controllo che seleziona le azioni del robot può essere considerata un'estensione dei classici problemi MDPs (Markov decision processes), anzi, per l'esattezza, dei POMDPs (Partially Observable MDPs); in questo caso, l'incertezza sullo stato è rappresentabile dalle Belief e la pianificazione del controllo è influenzata da misure rumorose ed incerte. Tali algoritmi prevedono, a loro volta, il calcolo del valore di una funzione obiettivo detta 'pay-off'; si tratta di effettuare una politica di selezione 'greedily' che massimizzi un certo valore. Se la funzione da valutare è ottima anche la

politica lo è; l'algoritmo itera il valore della funzione considerata migliorandolo tramite un aggiornamento ricorsivo. Se si lavora con modelli stocastici, anche per questi problemi, come avviene nel nostro caso, l'obiettivo è di incorporare nella politica l'incertezza relativa al movimento del robot e alle letture sensoriali rumorose. Il POMDP ha come goal quello di massimizzare le informazioni dell'ambiente rendendo massimo il guadagno di informazione; nel caso in esame, invece, il problema dell'esplorazione robotica ha lo scopo di diminuire l'Entropia associata alla posa assunta dal robot; è necessario sostituire la funzione obiettivo del POMDP con l'appropiata 'Pay-off function' prima riportata, cioè $U_a = E_a(H) + \alpha C(a)$.
L'algoritmo della minimizzazione dell'Entropia utilizzato può essere paragonato all'algoritmo MC-POMDP (Monte Carlo POMDP); si tratta di un algoritmo discreto per l'esplorazione probabilistica che ha come legge di controllo il 'trade-off' tra il guadagno di informazione, che si ottiene dalla differenza tra Entropia attuale H ed entropia attesa $E_a(H)$, e il costo associato allo spostamento. Per il calcolo della $E_a(H)$ si avvale anch'esso del filtro Bayesiano.

In conclusione, riassumendo in poche righe, la navigazione attiva nell'ambiente consiste nella scelta di azioni rispetto ad una politica di controllo considerata; tale insieme di azioni elementari o di 'target point' (sequenza di azioni elementari) che si possono effettuare devono essere selezionate in modo da minimizzare l'Entropia associata alla posa. Tali azioni, se eseguite secondo il criterio di selezione attiva, portano il robot da una certa posizione ad una nuova e, aggiornando la Belief, questo ottiene una nuova percezione della posa.

2.6 Stima della posizione effettiva del robot

Passiamo ora proprio a considerare la stima della posa del robot e descriviamo in che modo l'algoritmo la calcola.

Ricordiamo che ad ogni passo k della simulazione la localizzazione attiva implementata misura e aggiorna, tramite l'approccio Bayesiano, la Belief a priori (credenza sulla posa) e di verosomiglianza (a posteriori); inoltre la Belief altro non è che la probabilità del robot di occupare una certa posizione all'interno dell'ambiente. Da questa si può quindi ottenere un'idea, e dunque la stima, circa la posizione da questo occupata e tenere traccia della stima del percorso che compie durante la simulazione. Si avrà che il robot all'inzio avrà una Bel(l) uniformemente distribuita e dunque una stima della configurazione $x_s(0)$ in posizione centrale alla mappa; successivamente, a seguito delle azioni di controllo effettuate ottenute tramite selezione attiva l'Entropia associata alla posa tende a zero. A questo punto, la Belief avrà una densità di probabilità tutta concentrata in una certa zona relativa alla configuraziona che il robot occupa realmente; in caso contrario, ovvero l'Entropia non va a zero, la Belief fornirà una stima della posizione del robot in maniera errata e lontana dalla realtà.

La posizione stimata è calcolata nel file Stima_2D.m; esso fornisce la posizione stimata (x_s, y_s) nel caso bidimensionale con $x_r \in [1, L_x]$ e $y_r \in [1, L_y]$ ed è ottenuta nel seguente modo:

$$\begin{cases} x_s(k) &= \frac{1}{L_x * L_y} \sum_{x_k=1}^{L_x} \sum_{y_k=1}^{L_y} x_k Bel(x_k, y_k) \\ y_s(k) &= \frac{1}{L_x * L_y} \sum_{y_k=1}^{L_y} \sum_{x_k=1}^{L_x} y_k Bel(x_k, y_k) \end{cases} \tag{2.6.1}$$

Sempre in questo file si calcola l'errore di stima tra posizione effettiva del robot e

quella stimata.

$$\begin{cases} e_x(k) &= x_r(k) - x_s(k) \\ e_y(k) &= y_r(k) - y_s(k) \end{cases} \tag{2.6.2}$$

Per validare tale tecnica di localizzazione attiva, tutti i risultati ottenuti vengono graficati nel IV capitolo dove si riportano le simulazioni per i diversi casi analizzati.

Capitolo 3

Applicazione delle tecniche di localizzazione attiva ad alcuni modelli di robot mobili

In questo capitolo si illustreranno dettagliatamente la stesura del codice e il funzionamento dell'algoritmo di autolocalizzazione implementato in ambiente Matlab.

La localizzazione attiva descritta nel precedente capitolo è stata implementata e testata su alcuni modelli semplificati di robot mobile e per una tipologia via via più complessa e completa da un caso semplice Mono-dimensionale ad uno più complesso Bi-dimensionale.

La fase sperimentale è stata ottenuta dopo un'attenta e accurata analisi del problema. Si è dapprima scelto il modello del robot attraverso la definizione delle caratteristiche sonsoriali e di movimentazione possibili. Una volta concepito il modello si è analizzato l'ambiente: in primo luogo l'attenzione ha riguardato lo studio, l'analisi e l'implementazione del caso Mono-dimensionale, in questo modo l'ambiente lineare permetteva una più rapida soluzione e implementazione del problema.

A questo punto è stata fatta una prima verifica tramite controllo passivo per vedere se il robot fosse in grado di localizzarsi. Verificata la localizzazione è stato implemen-

tato il controllo attivo tramite la scelta dell'azione migliore, ossia quella in grado di minimizzare l'Entropia attesa. Quest'ultima è tenuta in memoria passo dopo passo ed è l'elemento fondamentale per il calcolo della funzione **Utilità** U(k) definita nel capitolo precedente nel paragrafo relativo al controllo.
Il controllo è *greedy* e le azioni elementari vengono selezionate in maniera intelligente tramite l'**action selection**: criterio di selezione che permette la scelta di un'azione che minimizza l'incertezza futura, ovvero l'Entropia attesa, sulla percezione della posizione occupata dal robot.
Il problema è stato successivamente evoluto al caso Bi-dimensionale dove si intende applicare il controllo attivo ad un robot libero di muoversi in un ambiente Bi-dimensionale lungo le coordinate degli assi x, y del piano e con un orientamento del robot che può assumere solo quattro angoli spaziati tra loro di 90°. Si tratta infatti di un robot dotato di una bussola che gli permette di posizionarsi nell'ambiente secondo quattro differenti orientamenti e può occupare qualsiasi posizione lineare all'interno dell'ambiente. Questo secondo caso è stato poi ulteriormente ampliato considerando non solo il controllo greedy ma anche un controllo basato su sequenze di azioni elementari scelte sempre in modo attivo che portino il robot in una configurazione lontana dalla precedente, ovvero un tipo di controllo a *target point* del tipo a =(5 passi avanti, 2 passi sopra).

Procediamo, quindi, all'esame e descrizione dettagliata in primo luogo del problema Mono-dimensionale per poi soffermarci su quello Bi-dimensionale.

3.1 Ambiente Mono-dimensionale

3.1.1 Ambiente di lavoro

In primo luogo si ha disposizione la mappa dell'ambiente dove il robot sarà libero di circolare durante le simulazioni.

Di essa si conoscono le 'features', ovvero le caratteristiche-oggetto necessarie per la localizzazione all'interno dello spazio di lavoro; si tratta di landmarks che il robot può riconoscere facilmente e utilizzare al fine della localizzazione.

Nel nostro caso sono definiti come 'landmarks PORTA' e il robot è capace di percepire la presenza o l'assenza di tali oggetti attraverso un sensore che ha a disposizione.

L'implementazione assume inizialmente che al robot è data una mappa metrica dell'ambiente ma non la sua posizione in esso.

L'ambiente è schematizzato da una lunga linea retta che potrebbe rappresentare un corridoio e poi suddiviso in tante celle esattamente L_x se L_x è la lunghezza fisica disponibile per la movimentazione, ovvero è il limite destro del corridoio a disposizione. Lo spazio delle possibili configurazioni appartiene all'intervallo compreso in $[1, L_x]$ ed è composto da un numero intero di celle ciascuna delle quali rappresenta una singola posa che il robot può occupare all'interno dell'ambiente; ad ognuna di esse viene, poi, associata una certa probabilità di essere occupata dal robot.

Il modello per la localizzazione è di tipo **Markoviano**: si tratta di un approccio probabilistico nel quale ad ogni istante k della simulazione, ad ogni possibile azione eseguita dal robot e per ogni acquisizione sensoriale, la localizzazione tiene conto di una densità di probabilità che il robot occupi una certa configurazione **l**, ovvero *Belief(l)*, all'interno dello spazio delle configurazioni del robot e ne tiene conto per $\forall l \in \{1, 2, .., L_x\}$.

Tale localizzazione però non si preoccupa di dare una risposta a come controllare gli attuatori del robot per meglio localizzarsi; quest'ultima si ottiene tramite poi una localizzazione attiva basata sulla selezione accurata dell'azione da compiere per meglio autolocalizzarsi attraverso il principio di minimizzazione dell'Entropia futura, cioè attesa a seguito di un'ipotesi di spostamento.

Nel caso esaminato l'ambiente è rappresentato dall'asse x le cui estremità sono situate in posizione:

- $x = 1$ limite sinistro;
- $x = L_x$ limite destro.

L'ambiente può essere concepito come un corridoio delimitato da pareti all'estremità, dove $l = 1$ rappresenta la posizione più a sinistra, $l = 2$ la penultima verso sinistra e così via.

I landmarks possono essere visti come degli oggetti riconoscibili dal robot; ipotizziamo infatti che il robot abbia a disposizione un sensore che rileva la presenza o l'assenza del 'feature porta' in una certa posizione $l \in [1, L_x]$. Ad esempio se la porta a_1 é in posizione $l = 7$, la a_2 in $l = 10$ e la porta a_i in $l = l_i$ si ottiene l'inizializzazione dei landmarks porta:

$$\begin{cases} a_1 & = 7 \quad \text{Posizione occupata dalla I porta.} \\ a_2 & = 10 \text{ Posizione occupata dalla II porta.} \\ a_i & = l_i \quad \text{Posizione occupata dalla i-esima porta.} \end{cases} \tag{3.1.1}$$

Nel caso implementato è possibile modificare velocemente l'ambiente. Per quanto riguarda l'estensione basta impostare semplicemente la grandezza desiderata attraverso la variabile L_x. Anche il numero e la posizione delle porte possono essere modificati con facilità attraverso l'inizializzazione del vettore Porta(:,1) desiderato. Entrambe si

settano nel file di Inizializzazione_dati_seq.m che è il file necessario all'inizializzazione di tutte le variabili necessarie all'algoritmo.
Sempre in questo file vengono infatti definite anche le $Bel(l_k)$, $Bel_a(l_k)$, $P_{prior}(l_k)$, $P_{posterior}(l_k)$, il numero di porte visitate[1] e gli errori presenti nel modello.
Nel caso pariticolare, trattandosi di un problema di **Localizzazione Globale**, il robot non ha alcuna informazione circa la sua x_0 e quindi la stima della $Bel(l_k)$ è uniformemente distribuita all'interno dell'intervallo $[1,L_x]$ e la probabilità di trovarsi in una qualsiasi posizione x_0 all'interno dello stesso è pari a $P(x_0) = \frac{1}{L_x}$ che coincide con $Bel(l_k)$ il quale è a sua volta uguale, per $k = 0$, a $Bel_a(l_k)$ $P_{prior}(l_k)$, $P_{posterior}(l_k)$.

Tornando all'ambiente, il robot è libero di muoversi all'interno di esso ma non ne può uscire. Infatti, i movimenti effettuati sono solo all'interno dell'insieme stabilito e se il robot si trova ad una delle due estremità, esso non ne è cosciente (non ha sensori che rilevino i fine corsa) e se decide di compiere un'azione che lo porterebbe in una posizione al di fuori del spazio consentito rimane sempre nella stessa posizione.
Tale situazione modifica l'aggiornamento della probabilità di stare in una certa posizione: la probabilità è calcolata in modo tale che all'esterno dell'insieme di tutte le possibili configurazioni $\Xi = \{1, 2, .., L_x\}$ la probabilità di occupare la posizione è costantemente nulla; all'interno, invece, è determinata dal modello di transizione degli stati e delle letture laser effettuate con particolare attenzione quando il robot si trova in posizione $l = 1$ o $l = L_x$ perché in questo caso la probabilità di stare in quella specifica posizione deve essere aggiornata tenendo conto che il robot può inibire una delle sue possibili azioni e restare fermo.

[1]Tale variabile non serve ai fini dell'algoritmo è stata utilizzata solo per verificarne la correttezza durante le simulazioni.

3.1.2 Modello sensoriale e di movimentazione

Nel caso considerato i sensori svolgono un ruolo importante ai fini della Localizzazione. Essi sono alla base del calcolo della densità di probabilità $Bel(l_k)$; le misure sensoriali acquisite durante la simulazione permettono l'aggiornamento da parte del robot della probabilità di percepire la propria posizione.
Ipotizziamo di avere a disposizione dati odometrici e letture laser.

Si ottengono, infatti, due aggiornamenti a seconda del dato sensoriale a disposizione:

- Il termine $Bel(l_k)$ indica l'aggiornamento della stima[2] a seguito di letture laser;
- Il termine $\widehat{Bel}(l_k)$ indica l'aggiornamento della stima[3] ottenuto dopo uno spostamento, dati odometrici.

I sensori sono dunque indispensabili per ottenere la stima della posizione.
I dati odometrici sono visti come letture sensoriali propriocettive e forniscono, dunque, misure relative[4] degli spostamenti, ovvero posizioni relative. Come tali sono sempre relazionate ad azioni precedenti ed è questo il motivo per cui, come vedremo, la probabilià $Bel(l_k)$ è un tipo di probabilità condizionata. L'errore di stima in questo caso tende a divenire sempre più grande con lo spostamento effettuato dal robot. Se, infatti, si avessero a disposizione i soli sensori odometrici e questi fossero rumorosi, il robot non riuscirebbe mai a far convergere la stima della propria posizione con quella reale.

[2]Corrisponde alle $P_{posterior}(l_k)$ durante le simulazioni effettive e alle $Bel(l_k)$ in caso della stima dell'Entropia.

[3]Corrisponde alle $P_{prior}(l_k)$ durante le simulazioni effettive e alle $Bel_a(l_k)$ in caso della stima dell'Entropia.

[4]Tali misure, come detto nel primo capitolo, possono considerarsi assolute solo se è nota la posizione iniziale, situazione che non si verifica mai nel caso della Localizzazione Globale.

Nel caso, invece, delle letture sensoriali laser[5] esse forniscono misure di posizione assoluta e l'errore di misura rimane costante, non vincolato, cioè, da quanto il robot si è spostato. Esse influenzano e aggiornano la $Bel(l_k)$ secondo la posizione assoluta assunta dal robot; dunque possono essere calcolate a priori per tutte le possibili configurazioni ammissibili e poi applicate alla simulazione per il caso particolare per la posizione corrente del robot.
In generale, poi, i sensori presentano alcuni problemi: i dati sono infatti spesso affetti da disturbi. Tali disturbi influiscono nell'azione di controllo perché agiscono in modo differente sull'aggiornamento della Belief e per l'analisi delle simulazioni si farà dunque riferimento a quattro possibili situazioni: il caso ideale, ovvero assenza di disturbi sia per il sensore che durante il movimento; i due casi intermedi dove è presente un solo tipo di disturbo alla volta; il caso reale nel quale sono presenti contemporaneamente i rumori sia sulle misure laser che sullo spostamento. Quest'ultimo è il caso più completo e realistico.

Passiamo dunque alla crezione del modello per il robot mobile basato sull'analisi delle letture sensoriali (misure laser) e della movimentazione (dati odometrici).

MODELLO SENSORIALE:

Il robot a disposizione per la simulazione è dotato di un sensore che rileva **'s'**, la presenza o l'assenza della porta.

[5] Appartengono alla famiglia dei sensori eterocettivi.

In particole **'s'** denota, quindi, una lettura laser e assume due unici valori: $s \in [0, 1]$:

$$\begin{cases} s & = 0 \quad \text{Non percepisce la porta} \\ s & = 1 \quad \text{Percepisce la porta} \end{cases} \tag{3.1.2}$$

- $P(s|l_k)$ è la probabilità di percepire la lettura **'s'** in posizione l all'istante k.

$P(s|l_k)$ si riferisce alla mappa dell'ambiente e si potrebbe calcolare fuori linea poiché rappresenta la probabilità di osservare una certa lettura **'s'** per $\forall l \in \{1, 2, .., L_x\}$.

Il modello probabilistico delle letture laser che aggiornano la stima della Bel(l) è il seguente:

- $P(s = 1|l_k = a_i) \longrightarrow$ Probabilità di vedere la porta quando il robot è davanti ad una porta (*Il sensore funziona correttamente*).
- $P(s = 0|l_k = a_i) \longrightarrow$ Probabilità di NON vedere la porta quando il robot è davanti ad una porta (*Il sensore NON funziona correttamente*).
- $P(s = 1|l_k \neq a_i) \longrightarrow$ Probabilità di vedere la porta quando il robot NON è davanti ad una porta (*Il sensore NON funziona correttamente*).
- $P(s = 0|l_k \neq a_i) \longrightarrow$ Probabilità di NON vedere una porta quando il robot NON è davanti ad una porta (*Il sensore funziona correttamente*).

Assenza di rumori sulle letture laser: caso ideale

$$\begin{aligned} P(s = 1|l_k = a_i) = 1 \qquad & P(s = 1|l_k \neq a_i) = 0 \\ P(s = 0|l_k = a_i) = 0 \qquad & P(s = 0|l_k \neq a_i) = 1 \end{aligned} \tag{3.1.3}$$

Corrisponde alle simulazioni in cui l'$errore_{sensore} = 0$ (OFF).

Presenza di rumori sulle letture laser: caso reale

$$\begin{aligned} &P(s=1|l_k=a_i)=0.7 \qquad P(s=1|l_k\neq a_i)=0.1 \\ &P(s=0|l_k=a_i)=0.3 \qquad P(s=0|l_k\neq a_i)=0.9 \end{aligned} \tag{3.1.4}$$

Corrisponde alle simulazioni in cui l'$errore_{sensore} = 1$ (ON).

Dopo aver ricevuto la lettura laser il robot aggiorna la sua densità di probabilità Belief seguendo la regola seguente:

$$Bel(l_k) \longleftarrow \frac{P(s|l_k)\widehat{Bel}(l_k)}{p(s_k)} \ \forall l_k \in \{1,2,..,L_x\} \tag{3.1.5}$$

Dove p(s) è un fattore normalizzante calcolato nel seguente modo:

$$p(s_k) = \sum_{l_k=1}^{L_x} P(s|l_k)\widehat{Bel}(l_k) \tag{3.1.6}$$

È fondamentale per fare sì che $\sum_{l_k=1}^{L_x} Bel(l_k) = 1$

Il termine $\widehat{Bel}(l_k)$ è la probabilità di occupare la posizione l all'istante k prima di acquisire la lettura sensoriale. Tale probabilità viene chiamata $P_{prior}(l_k)$. Il termine $Bel(l_k)$ è invece la probabilità di occupare la posizione l all'istante k subito dopo aver acquisito la lettura sensoriale. Tale probabilità viene chiamata $P_{posterior}(l_k)$.

MODELLO di MOVIMENTAZIONE:

Il movimento del robot in assenza di errori odometrici è modellato dalla seguente equazione cinematica:

$$\begin{cases} x(k+1) &= x(k)+a(k) \iff x(k+1)\in\{1,2,..,L_x\} \\ x(k+1) &= 1 \iff x(k+1)\leq 1 \\ x(k+1) &= L_x \iff x(k+1)\geq L_x \end{cases} \tag{3.1.7}$$

Si tratta dell'aggiornameto della posizione del robot nel caso ideale, ovvero per l'$errore_{spostamento} = 0$ (OFF).

L'azione $a(k)$ che effettua il robot per modificare la sua posizione attuale all'istante k permette al robot due soli spostamenti e in totale tre possibili configurazioni all'istante successivo.

$a(k) \in [-1, 0, +1]$ e implica rispettivamente:

$$\begin{cases} a(k) & = -1 \quad \text{Il Robot fa un passo indietro.} \\ a(k) & = \ 0 \quad \text{Il Robot resta fermo.} \\ a(k) & = +1 \quad \text{Il Robot fa un passo in avanti.} \end{cases} \tag{3.1.8}$$

In presenza di errori sulle misure odometriche l'$errore_{spostamento} = 1$ (ON) l'equazioni dell'aggiornamento di stato diventano:

$$\begin{cases} x(k+1) & = x(k) + a(k) + n(k) \iff x(k+1) \in \{1, 2, .., L_x\} \\ x(k+1) & = 1 \qquad \iff x(k+1) \leq 1 \\ x(k+1) & = L_x \qquad \iff x(k+1) \geq L_x \end{cases} \tag{3.1.9}$$

n(k) è il disturbo che agisce sullo spostamento e può assumere un valore intero compreso tra -1 e +1 ed è modellato in questa maniera: ogni volta che il robot si muove $a(k) \neq 0$ ha una certa probabilità di non muoversi (pari a 0.1), di muoversi correttamente (pari a 0.7) e di muoversi più del dovuto (pari a 0.2). Se sta fermo o in assenza di errori, ha rispettivamente la probabilità di restare fermo o di muoversi correttamete pari ad 1.

Quindi anche il movimento del robot è modellato da probabilità di spostamento. Si tratta in particolar modo di una probabilità condizionata $P_a(l|\widetilde{l})$ che denota la probabilità che l'azione $a = a(k)$ se eseguita in posizione $x(k) = \widetilde{l}$ porta $x(k)$ in una posizione $x(k+1) = l$ all'istante successivo.

Dopo aver eseguito un'azione il robot aggiorna la sua densità di probabilità Belief

seguendo questa regola:

$$\widehat{Bel}(x = l(k+1)) \longleftarrow \sum_{\widetilde{l}(k)=1}^{L_x} P_a(l(k+1)|\widetilde{l}(k))Bel(\widetilde{l}(k)) \;\; \forall\, l(k+1) \in \{1, 2, .., L_x\} \tag{3.1.10}$$

Il termine $\widehat{Bel}(l(k+1))$ è la probabilità di occupare la posizione l all'istante $(k+1)$ subito dopo aver eseguito l'azione a(k) ma poco prima di acquisire le successive letture sensoriali. Tale probabilità serve per aggiornare $P_{prior}(l(k+1))$.

3.1.3 Tipi di controllo utilizzati: passivo, attivo, casuale

Una volta messa a punto la procedura che calcola la stima della posizione del robot e le leggi che ne regolano l'aggiornamento, si passa alla costruzione degli algoritmi di controllo del moto che consentano al robot di spostarsi nell'ambiente per localizzarsi. Sono stati implementati diversi tipi di controllo.

Sequenza cronologica della stesura dell'algoritmo di controllo:

Il primo approccio è stato quello relativo al *controllo passivo greedy* che é servito nella prima fase del progetto perché ha permesso alcune verifiche circa la fattibilità dei modelli del robot utilizzati e per vedere se il robot era in grado di autolocalizzarsi.

Controllo PASSIVO:

Il controllo consisteva in una serie di azioni studiate ed analizzate a tavolino, che permette al robot di trovare la stima effettiva della propria posizione da una qualsiasi condizione iniziale per lo stato x all'interno dell'insieme delle configurazioni possibili. Il controllo passivo è stato implementato senza far uso di selezione attiva delle azioni e il programma non necessitava di calcolare l'Entropia futura associata allo sposta-

mento che si voleva effettuare per scegliere l'azione che la minimizzasse. In questo caso infatti l'algoritmo di localizzazione viene eseguito mandando in pasto all'algoritmo una sequenza di azioni elementari prestabilite ad ogni istante k dalla simulazione partendo da una certa posizione. Si è subito notato che l'andamento dell'Entropia associata alla sequenza di controllo andava a zero in un tempo finito portando alla stima della posizione reale del robot. Una volta verificata la capacità del robot di stimare la propria $x_s(k)$ per una certa posizione si è poi cercato di verificare se tale capacità fosse valida per qualsiasi condizione iniziale.
Fondamentalmente lo scopo di tale controllo passivo consisteva nel verificare che il robot idealizzato con le caratteristiche descritte nei paragrafi precedenti, quindi secondo quel modello cinematico, di movimento e sensoriale, fosse in grado di localizzarsi in un ambiente, anche questo concepito nel modo descritto all'inizio del capitolo.

Si è scoperto che per un tipo di 'controllo sequenziale significativo', applicando cioè una sequenza di movimenti uguali, ovvero tutti spostamenti in avanti (tutte $a_k = +1$) o indietro (tutte $a_k = -1$), il robot era in grado di stimare la propria posizione in presenza anche di errori sulle misure e sullo spostamento, diminuendo maggiormente il tempo neccessario per raggiungere la stima effettiva e con un andamento dell'Entropia che tendeva più velocemente a zero.

Controllo ATTIVO:

L'attenzione si è incentrata sull'ottimizzazione del problema attraverso l'applicazione di un tipo di controllo attivo che permettesse al robot di interrogarsi direttamente con l'ambinte e di operare in maniera autonoma attraverso un algoritmo di scelta ottima dell'azione da compiere.

Per meglio localizzarsi il robot ha bisogno di controllare gli attuatori in modo tale da effettuare delle azioni che permettano di trovare con maggior certezza la propria posizione occupata nello spazio di stato ammisibile. Tale posizione si ottiene dalla media della distribuzione della probabilità di occupazione cella ad un certo istante k che spesso coincide con quella che ha probabilità massima tra tutte le varie probabilità di occupazione di ogni configurazione possibile (di ogni nodo).

Il concetto base dell'approccio sviluppato è di controllare gli attuatori in modo da minimizzare l'incertezza futura che si aspetta a seguito di uno spostamento singolo $a_i \in$ [Insieme di tutte le azione che l'automa può effettuare]. L'incertezza è misurata dall'Entropia associata alla futura densità di probabilità (Belief).
Quella della **scelta di azioni ottime** a_k^* che minimizzano l'incertezza della posizione futura è una tecnica di **controllo attivo** che permette al robot di localizzarsi attivamente nell'ambiente. Tale approccio è valido empiricamente nel contesto di uno dei problemi fondamentali della localizzazione, ovvero l'**Active Navigation** che risolve la questione di dove muoversi successivamente. In questo caso l'azione viene scelta istante per istante durante la simulazione sulla base della funzione Utilità $U_{a_i}(1:3,k)$ di dimensione (3xk) e dove: 3 sono il numero di righe ciascuna delle quali è associata ad un movimento possibile[6] e k è il numero di colonne che è pari ai passi della simulazione poichè ad ogni istante si calcola la $U_{a_i}(k)$ associata ad ogni spostamento. Quindi ad un istante $k = 10$ l'algoritmo avrà calcolato il valore della funzione Utilità per (10x3)=30 volte.
Questa funzione è di fondamentale importanza per l'algoritmo poichè il valore che assume per ogni istante k, e per ogni riga, rappresenta l'Entropia attesa della posa del

[6]Nel caso Mono-dimensionale a può assumere solo 3 valori, $a_i(k) \in (-1, 0, +1)$.

robot che si avrà all'istante succesivo $(k+1)$ se si applica quella azione corrispondente alla riga considerata. Quindi la selezione si avvale di scegliere la a^* che renderà più piccola l'incertezza sulla stima $x_s(k+1)$.

Controllo CASUALE:

Si è poi implementata una selezione casuale delle tre possibili azioni ad ogni passo per generare una sequenza casuale delle azioni di controllo (**controllo casuale**) per verificare che tale controllo non fosse migliore di un algoritmo di controllo non casuale.

Per i vari controlli implementati sono stati poi confrontati gli errori di stima, il tempo necessario per l'esecuzione del compito e le simulazioni per diverse situazioni, condizioni iniziali, durata della simulzione, casi ideali, reali e per diversi ambienti simmetrici e asimmetrici.
In ogni circostanza l'algoritmo attivo è risultato il più efficiente, affidabile e robusto ad errori di misura e/o a guasti dei sensori.
Questi risultati simulativi verranno analizzati e commentati in modo dettagliato nel prossimo capitolo.

3.2 Ambiente Bididimensionale

Il caso Bi-dimensionale è stato concepito come estensione del caso Mono-dimensionale; si è quindi proceduto a seguire le tappe descrittive utilizzate per il caso precedentemente esposto adattandolo alla versione 2D del problema.

3.2.1 Ambiente di lavoro

Anche per il caso 2D si assume di avere a disposizione una mappa metrica dell'ambiente che viene opportunamente creata fuori linea dall'algoritmo. Si tratta sempre di un ambiente 'indoor' di tipo stazionario, ovvero non ci sono ostacoli o oggetti in movimento.
L'ambiente a disposizione è noto e il robot è libero di muoversi all'interno di esso e non può uscirne.
Si tratta di un ambiente schematizzato su un piano Bi-dimensionale di dimensione $(1 : L_x, 1 : L_y)$ e l'insieme di tutte le possibili configurazioni $(L_x * L_y)$ del robot sono i nodi appartenenti al piano, che si ottengono dall'intersezione delle rette parallele alle ascisse x_i e alle ordinate y_i per valori di x e y interi e appartenenti rispettivamente a $\{1, 2, .., L_x\}$ e a $\{1, 2, .., L_y\}$. I nodi appena definiti corrispondono all'insieme delle celle di cui ci si riferiva al caso mondimensionale, e, anche in questo caso, rappresentano l'insieme di tutte le configurazioni che il robot può occupare all'interno dell'ambiente di lavoro. Si tratta infatti di una rappresentazione topologica del 'Belief state', nel quale ogni possibile configurazione corrisponde a un nodo nella mappa topologica dell'ambiente.

I landmarks 'porta' che il robot riesce a riconoscere, ovvero le caratteristiche-oggetto necessarie per la localizzazione, in questo caso sono definiti attraverso un vettore 'Porta' di dimensione (Nx2) dove N è il numero di porte presenti nell'ambinete e 2 sono gli elementi numerici che servono per descrivere in maniera univoca la posizione che i landamarks occupano nello spazio di lavoro. Per ogni porta i-esima si hanno, infatti, due componenti e rispettivamente (x,y) che descrivono in maniera univoca il punto $porta_i(x_i, y_i)$ che la porta i-esima occupa nel piano.

Definizione porte:

$$\begin{cases} porta_1 & = (x_1, y_1) \quad \text{Posizione occupata dalla I porta.} \\ porta_2 & = (x_2, y_2) \quad \text{Posizione occupata dalla II porta.} \\ porta_N & = (x_N, y_N) \quad \text{Posizione occupata dalla N-esima porta.} \end{cases} \tag{3.2.1}$$

L'implementazione assume inizialmente che al robot è data una mappa metrica dell'ambiente ma non la sua posizione in esso. Si tratta, infatti, di risolvere, anche in questo caso, il noto problema di localizzare il robot globalmente.
Il modello per la localizzazine è di tipo **Markoviano** e si avvale di un approccio probabilistico per la determinazione della posizione del robot. Tale tecnica consiste, come visto precedentemente, nell'aggiornamento della densità di probabilità (*Belief*) dell'intero spazio delle configurazioni del robot ad ogni istante k della simulazione, dopo che l'automa ha eseguito un qualsiasi spostamento e dopo aver acquisito un qualsiasi dato sensoriale. La Belief a cui si fa riferimento, per il caso Bi-dimensionale, è la probabilità di occupare una certa posizione (x,y), ovvero, bisogna tener conto che per questo ambiente si tratta di un vettore appartente a $\Re^2$.

Tornando alla struttura metrica e topologica dell'ambiente, esso è rappresentato dallo spazio delimitato da 4 rette di cui due sono parallele tra loro e anche all'asse x e le altre due sono parallele sempre tra loro ma perpendicolari alle precedenti, dunque parallele all'asse y. L'insieme di tali rette costituisce un rettangolo nel caso in cui $L_x \neq L_y$ o un quadrato, $L_x \approx L_y$, il cui perimetro rappresenta il limite dello spazio di lavoro, quindi le estremità.

L'ambiente può essere concepito come una stanza di un ufficio delimitato da 4 pareti ciascuna delle quali può presentare una o più porte. Tali landmarks sono visibili

dal robot secondo un modello sensoriale che ne determina la probabilità di percepire o non percepire la porta da una certa distanza. Infatti percepire la feature-porta è inversamente proporzionale alla distanza del robot dall'elemento da rivelare; si vedrà nel successivo paragrafo che la possibilità di vedere una porta diminuisce secondo un andamento esponenziale.

Il sensore riesce a percepire una porta se, e soltanto se, il robot è in una certa posizione non troppo lontana dalla porta, ovvero all'interno di una certa regione R_t[7] di rivelamento.

Non c'è distinzione tra i vari landmarks infatti si tratta di tutte porte standard e il robot in ogni posizione ha la probabilità di percepire, o meno, la presenza di uno dei landmark presenti nell'ambiente.

Come si vedrà nel paragrafo successivo, infatti, la probabilità di vedere tali features-porta è una probabilità composta: ovvero la probabilità di vedere una porta è data dalla combinazione delle probabilità di vedere ciascuna porta all'interno dell'ambiente considerato.

Nel caso implementato è possibile modificare velocemente nel file contenente l'inizializzazione dei dati[8] la dimensione dell'ambiente impostando la grandezza desiderata per le variabili L_x e L_y e il numero e la posizione delle porte tramite il vettore Porta. Sempre in questo file vengono inizializzate le stesse quantità $Bel_a(x_0, y_0)$, $P_{prior}(x_0, y_0)$, $P_{posterior}(x_0, y_0)$, come avviene per il caso Mono-dimensionale. La scelta dello stesso nome per le variabili è stata fatta al fine di rendere più chiara l'implementazione per entrambi i casi e per permettere un facile confronto e notare le analogie esistenti tra il caso Mono-dimensionale e quello Bi-dimensionale.

[7] R_t rimandiamo la definizione nel paragrafo relativo al modello sensoriale.

[8] Si tratta dello omonimo file per caso 1D, ovvero Inizializzazioni_dati_seq.m.

Il problema 2D risulta, però, assai più complicato per l'utilizzo di matrici di dimensioni maggiori, di uno spazio di stato molto più grande dovuto alle molteplici possibili combinazioni (x,y)[9], di un numero di azioni possibili maggiore di quello Monodimensionale e per le risorse di memoria utilizzate; tralasciando questi particolari al livello più astratto il concetto alla base dell'algoritmo è il medesimo.

Anche in questo caso, il robot è libero di muoversi all'interno dell'ambiente e non può uscirne: i movimenti effettuati sono quindi solo all'interno dell'insieme stabilito e se il robot si trova ad uno degli angoli della stanza non può avanzare oltre la parete; l'automa non ha a disposizione dei sensori che notino l'esistenza di un muro o di uno ostacolo e quindi, se il robot decide di compiere un'azione che lo porterebbe al di là della parete, ovvero fuori dal spazio di stato consentito, rimane fermo nella stessa posizione. Il robot pensa comunque di aver fatto il movimento ma la probabilità è aggiornata in modo tale da tener conto di questa situazione (il robot non può oltrepassare il confine consentito).

L'aggiornamento della probabilità di stare in una certa posizione (x_i, y_i) è fatto in modo tale che all'esterno del perimetro della stanza la probabilità di occupare la posizione è costantemente nulla, all'interno, invece, è determinata dal modello di transizione degli stati e delle letture laser effettuate ricordando il caso particolare nel quale il robot navighi in prossimità delle configurazioni limite. All'inizio della navigazione, il robot non ha nessuna informazione circa lo stato iniziale da lui occupato, si ha infatti che la probabilità di stare in ciascuna cella della stanza è uniformemente distribuita e vale $Bel(x, y) = \frac{1}{L_x * L_y}$.

[9]In particolare si avrano $L_x * L_y$ possibili posizioni occupate dal robot all'interno dell'ambiente.

3.2.2 Modello sensoriale e di movimentazione:

MODELLO SENSORIALE:

Il robot a disposizione per la simulazione è dotato di un sensore che rileva **'s'**, la presenza o l'assenza di una porta.

Per il caso Bi-dimensionale, si tratta di un sensore laser che permette di rilevare una 'feature-porta' compresa in un angolo di 360°. Infatti come abbiamo detto prima, in una certa posizione occupata dall'ambiente il robot ha la probabilià di vedere un qualcosa o di non vedere ma non determina quale landmark vede. Quindi si tratta, in particolare, di un sensore la cui lettura laser **'s'** può assumere due unici valori, $s \in [0, 1]$; è un dispositivo di tipo on/off, esso nota o non nota la presenza di una porta.

$$\begin{cases} s & = 0 \quad \text{Non percepisce la porta} \\ s & = 1 \quad \text{Percepisce la porta} \end{cases} \tag{3.2.2}$$

La probabilità di percepire un landmark da una certa posizione (x,y) è dato dalla sequente probabilità:

- $P(s|(x_k, y_k))$ è la probabilità di percepire la lettura **'s'**$\in [0, 1]$ all'istante k in posizione $l_k = (x_k, y_k)$.

Per il calcolo di tale probabilità, nel caso della predizione, ci si potrebbe avvalere di una procedura fuori linea, perché la probabilità non dipende dalla particolare posizione occupata durante la simulazione ma va calcolata per tutte le $(L_x * L_y)$ configurazioni in cui è stato discretizzato lo spazio di stato. Nel caso simulativo invece si ha che la matrice delle probabilità $P(s|(x_k, y_k))$ deve essere calcolata in linea perché dipende fortemente dalle letture laser correnti s_k. Essa dipende, inoltre, in modo diretto

dal modello metrico dell'ambiente e dal modello del sensore di prossimità utilizzato dall'automa.

La $P(s|(x,y))$ di percepire il dato **'s'** in posizione (x,y) dipende dalla combinazione delle singole probabilità di percepire **'s'** rispetto alla distanza robot/porta h-esima, cioè la quantità $P_s(porta_h|(x,y))$, calcolata per tutte le porte disponibili nell'ambiente.

In pratica la $P(s|(x_i,y_j))$ totale calcolata per ogni nodo (x_i,y_j) è data dalla seguente formula:

$$P(s|(x_i,y_j)) = 1-(1-P_s(porta_1|(x_i,y_j)))*...*(1-P_s(porta_N|(x_i,y_j)))\ \forall i,j \quad (3.2.3)$$

Con $i \in [1,L_x]$ e $j \in [1,L_y]$.

In definitiva si avranno 2 matrici $P(s|(x_i,y_j))\ \forall(i,j)$ di dimensione $(L_x \text{x} L_y)$ calcolabili fuori linea per ogni possibile lettura laser **'s'**$\in [0,1]$ e in particolare una sarà data da $P(s=1|(x_i,y_j))$ e l'altra è la sua complementare, ovvero, $P(s=0|(x_i,y_j)) = 1-P(s=1|(x_i,y_j))$.

Passiamo quindi allo studio e all'analisi del modello di percezione del sensore di una singola porta.

Lo spazio viene suddiviso in due zone fondamentali, R_t e R_{1-t}.

1. R_t definisce quell'insieme di celle limitrofe alla porta. Rientrano in essa tutti quei nodi che si trovano ad una distanza minima di d_{max} dalla porta;

2. R_{1-t} è lo spazio restante dunque lontano dalla porta ed è facilmete ottenibile dall'esclusione delle celle appartenenti a R_t dallo spazio delle configurazioni totali.

La distanza d_{max} è la distanza massima consentita al laser per percepire $(s=1)$ la

porta[10]. Nel nostro caso tale distanza è stata posta pari $d_{max} \approx 4.3$ per permettere, secondo il modello, di percepire la porta quando la probabilità di percepirla è maggiore di 0.3; ovvero quando la probabilità di non percepirla in maniera errata per mal funzionamento del dispositivo sensoriale ($s = 0$) è troppo elevata, cioè pari a 0.7.
Fuori dalla regione R_t si ha che la probabilità di percepire il landmark ($s = 1$) è costantemente pari a 0.1, mentre nel caso duale ($s = 0$) si ha che la probabilità di non percepirla è sempre pari a 0.9.

Caso Lettura laser s nella zona R_t di una porta:

Prendiamo in considerazione una porta qualunque **'h'** delle N porte disponibili. Ipotizziamo che si trova in posizione $porta_h = (x_p, y_p)$, la regione R_t è allora composta da tutte quelle celle che si trovano in posizione la cui distanza $r_{(i,j),h} \leq d_{max} = 4.3$.
Definendo con il termine $r_{(i,j),h}$ la distanza della posizione assunta dal robot mobile dalla porta h-esima; considerando, poi, **'h'** per ogni porta, si determinano tutte le $R_{i,j}$ distanze della posizione (x_i, y_j) con ciascun landmark.
Trattandosi di un sistema Bi-dimensionale, il calcolo delle distanze si è ottenuto con la norma euclidea e quindi riguarda il classico problema di determinare la distanza tra due punti. Quindi la quantità $r_{(i,j),h}$, nel caso particolare, rappresenta la distanza tra il punto (x_i, y_j) e il punto (x_p, y_p) e viene così calcolata:

$$r_{(i,j),h} = \sqrt{(x_i - x_p)^2 + (y_j - y_p)^2} \quad \forall\, i \in [1, L_x],\ \forall\, j \in [1, L_y],\ \forall\, h \in [1, N] \quad (3.2.4)$$

In questa regione, calcolata la distanza tra la posizione (x_i, y_j) e la $porta_h$ nel modo appena definito, la probabilità $P_s(porta_h|(x, y))$ di leggere s rispetto alla porta

[10]Distanza massima dalla porta superata la quale $s = 0$.

h-esima è dato dal seguente schema di condizioni su $r_{(i,j),h}$.

- $P_{s=1}(porta_h|(x,y)) \Longleftrightarrow r_{(i,j),h} \leq 4.3 \longrightarrow$ Funzionamento corretto[11].
- $P_{s=0}(porta_h|(x,y)) \Longleftrightarrow r_{(i,j),h} \leq 4.3 \longrightarrow$ Funzionamento scorretto[12].
- $P_{s=1}(porta_h|(x,y)) \Longleftrightarrow r_{(i,j),h} > 4.3 \longrightarrow$ Funzionamento scorretto[13].
- $P_{s=0}(porta_h|(x,y)) \Longleftrightarrow r_{(i,j),h} > 4.3 \longrightarrow$ Funzionamento corretto[14].

Il Modello probabilistico delle letture laser che aggiornano la stima della Bel(x,y) è il seguente:

CASO IDEALE: Assenza di rumori

Per le simulazioni è il caso $errore_{sensore} = 0$ (OFF).

$$\begin{cases} P_{s=1}(porta_h|(x,y)) & = 1 \quad \Longleftrightarrow r_{(i,j),h} \leq 4.3 \\ P_{s=0}(porta_h|(x,y)) & = 0 \quad \Longleftrightarrow r_{(i,j),h} \leq 4.3 \\ P_{s=1}(porta_h|(x,y)) & = 0 \quad \Longleftrightarrow r_{(i,j),h} > 4.3 \\ P_{s=0}(porta_h|(x,y)) & = 1 \quad \Longleftrightarrow r_{(i,j),h} > 4.3 \end{cases} \tag{3.2.5}$$

Il Modello probabilistico nel caso di letture laser soggette ad errori di misura aggiorna la stima della Bel(x,y) in due modi differenti e complementari.

CASO REALE: letture sensoriali RUMOROSE

Per le simulazioni è il caso $errore_{sensore} = 1$ (ON).

[11] Probabilità di vedere la porta quando il robot si trova nei pressi di una porta.
[12] Probabilità di vedere la porta quando il robot si trova nei pressi di una porta.
[13] Probabilità di vedere la porta quando il robot è lontano alla porta.
[14] Probabilità di NON vedere la porta quando il robot è lontano dalla porta.

Modello corrispondente alla lettura sensoriale $s = 1$.

$$\begin{cases} P_{s=1}(porta_h|(x,y)) & = 0.9 \iff r_{(i,j),h} < 1 \\ P_{s=1}(porta_h|(x,y)) & = 0.7 \iff r_{(i,j),h} \geq 1,\ r_{(i,j),h} < 2 \\ P_{s=1}(porta_h|(x,y)) & = 0.5 \iff r_{(i,j),h} \geq 2,\ r_{(i,j),h} < 3 \\ P_{s=1}(porta_h|(x,y)) & = 0.3 \iff r_{(i,j),h} \geq 3,\ r_{(i,j),h} \leq 4.3 \\ P_{s=1}(porta_h|(x,y)) & = 0.1 \iff r_{(i,j),h} > 4.3 \end{cases} \tag{3.2.6}$$

Modello corrispondente alla lettura sensoriale $s = 0$, si tratta del modello complementare al precedente.

$$\begin{cases} P_{s=0}(porta_h|(x,y)) & = 0.1 \iff r_{(i,j),h} < 1 \\ P_{s=0}(porta_h|(x,y)) & = 0.3 \iff r_{(i,j),h} \geq 1,\ r_{(i,j),h} < 2 \\ P_{s=0}(porta_h|(x,y)) & = 0.5 \iff r_{(i,j),h} \geq 2,\ r_{(i,j),h} < 3 \\ P_{s=0}(porta_h|(x,y)) & = 0.7 \iff r_{(i,j),h} \geq 3,\ r_{(i,j),h} \leq 4.3 \\ P_{s=0}(porta_h|(x,y)) & = 0.9 \iff r_{(i,j),h} > 4.3 \end{cases} \tag{3.2.7}$$

Andamento esponenziale per le letture laser

Per calcolare la probabilità $P_s(porta_h|(x,y))$ si è ipotizzato un comportamento del sensore le cui letture laser dispongono le probabilità di ottenere certi dati sensoriali, ovvero $s = 1$, seguendo una funzione esponenziale fortemente legata alla distanza robot/landmark. Essa mostra, volutamente, un andamento già da subito decrescente in modo molto rapido. Situazione molto vicina alla realtà perché il robot riesce a percepire la porta con elevata probabilità nei pressi di essa fin quando il raggio laser può scannerizzare; superata la distanza limite di lettura laser il robot ha una probabilità molto inferiore di vedere la porta. Si vede infatti che per i primi tre valori (0,1,2) che corrispondo rispettivamente alla distanze 'posa-porta' $r_{(i,j),h} = [0, 1, 2]$, e dunque in una situazione dove l'automa è molto prossimo alla porta o praticamente davanti ad essa, le probabilità che la percepisca sono molto elevate, con una media pari a 0.7. Superata una certa distanza la probabilità di percepire la porta diventa costantemente pari a 0.1 e diventa indipendente dal particolare valore $r_{(i,j),h}$ che ha

superato la soglia. La soglia corrispondente considerata è proprio la d_{max} introdotta precedentemente.

Uno stesso andamento esponenziale ma, in questo caso, crescente è stato utilizzato invece per descrivere l'andamento della probabilità di una lettura sensoriale pari a $s = 0$. In particolare si tratta dello stesso esponenziale nel caso precedente, $s = 1$, ma complementare, come è giusto che sia considerando che si tratta di un sensore ON/OFF. Questo significa che in una certa posizione dell'ambiente può leggere la porta con probabilità p mentre se non la legge lo fa con una probabilità $1 - p$. Anche qui superato il valore d_{max} rimane una probabilità pari a 0.9, ovvero da una certa distanza 'posa-porta' il robot non riesce più a vedere la porta e la probabilità di non vederla aumenta.

In figura 3.1 viene riportato un esempio del calcolo delle probabilità $P_s(porta_h|(x_i, y_j))$ per ogni singola porta 'h': le porte sono rappresentate da cerchi rossi, le pareti sono le line più grosse di colore verde, mentre le intersezioni delle linee leggere di colore verde perpendicolari tra loro rappresentano lo spazio delle configurazioni che il robot può assumere nell'ambiente (tre esempi di configurazione ammisibile: quadrati gialli).

Una volta calcolata la probabilità $P_s(porta_h|(x_i, y_j))$ di leggere **'s'** per la singola porta si passa alla probabilità globale di percepire almeno una porta in posizione (x,y), la $P(s|(x_k, y_k))$ già precedentemente esplicitata come composizione di tutte le singole probabilità 'posa-porta'.

Il modello sensoriale appena descritto è stato implementato in tre versioni: la prima è servita per il controllo attivo nel caso della predizione della $E_a(k)$ e del relativo calcolo della funzione di Utilità $U(k)$; le altre due per il calcolo dell'Entropia reale ($E(k)$) e di quella casuale ($E_c(k)$) a seguito di determinate azioni compiute.

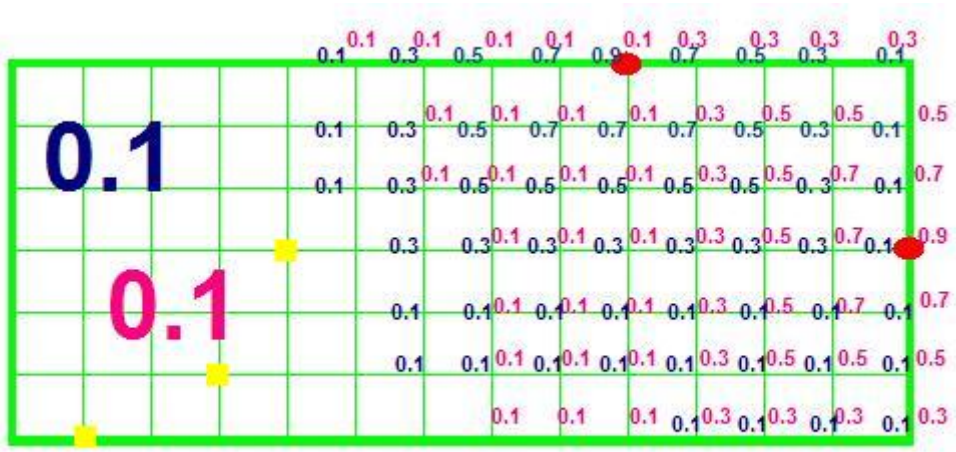

Figura 3.1: Calcolo delle probabilità di vedere un porta da una certa configurazione.

Il 'Modello di percezione' è stato implementato in ambiente matlab nel relativo foglio Perception_model_seq2D.m. Serve all'algoritmo per aggiornare come nel caso 1D la $Bel(l(k))$[15], detta anche $P_{posterior}(l(k))$, che ricordiamo essere la probabilità di occupare la posizione l all'istante k subito dopo aver acquisito le letture sensoriali. Mentre nel caso Mono-dimensionale si trattava di un aggiornamento di un vettore colonna, in questo caso si ha una matrice e la quantità $Bel(x_k, y_k)$ è ottenuta tramite due cicli for che esplorano lo spazio di tutte le configurazioni (x_i, y_i). Bisogna inoltre fare una precisazione, nel caso 1D il vettore Bel(l(k)) era un vettore colonna per il quale la posizione di ogni elemento ordinato per riga corrisponde in maniera lineare con la topologia dell'ambiente; ovvero la probabilità che il robot si trovi in posizione $x_i \in [1, L_x]$ è proprio $Bel(x_i)$. Ciò non accade per il caso 2D dove le righe x_i di $Bel(x_i, y_i)$ indicano la posizione occupata rispetto all'asse delle ascisse, mentre le colonne y_i quella relativa all'asse delle ordinate.

[15]Essa si ottiene dalla $\widehat{Bel}(l(k))$, quindi dalla $P_{prior}(l(k))$.

È di fondamentale importanza verificare che $\sum_{l_x=1}^{L_x} \sum_{l_y=1}^{L_y} Bel(l_x, l_y) = 1$, cosa che si ottiene applicando anche in questa circostanza il fattore di normalizzazione p(s).

MODELLO di MOVIMENTAZIONE:

Per quanto riguarda il modello di movimentazione del robot si parte dall'analisi di tutte le azioni che il robot è in grado di compiere. Si tratta di un modello di robot mobile dotato di una bussola capace di orientarlo con certezza[16] secondo quattro direzioni. Esplicitando gli angoli di posa ammissibili si ha che l'automa può avere un orientamento pari a $\theta \in [0, 90°, 180°, 270°]$ rispetto all'asse delle ascisse x e considerati crescenti in senso antiorario.

Il movimento lineare Δr in avanti consentito è pari all'unità per ogni spostamento ($\Delta r = 1$). Questo implica che, dato che si ha a disposizione con 4 orientamenti, l'insieme di azioni, distinte e disponibili, di cui l'automa può usufruire per navigare sono 5: le prime 4 sono relative ad uno spostamento unitario per ogni possibile orientamento consentito; l'ultima è relativa ad uno spostamento nullo, ovvero il robot sosta nella propria posizione senza effettuare alcun movimento.

Tale insieme di azioni permette al robot di portarsi in qualsiasi posizione all'interno dell'ambiente di lavoro e si tratta di navigazioni su traiettorie squadrate, quindi di traiettorie rettangolari.

Per meglio semplificare e schematizzare il problema si è effettuato un cambio di variabili per esprimere le 5 azioni rispetto al sistema di riferimento globale degli assi x ed y. In poche parole il robot può decidere se andare a destra, sinistra, sopra o sotto. Si definisce in questo modo un vettore (5x2), le cui righe rappresentano un comando mentre i 2 elementi per ognuna di esse rappresentano le componenti rispetto agli assi

[16] Implica che non ci sono errori di spostamento relativi a rotazioni del robot su se stesso.

x ed y del movimento; ogni singola azione è intesa come $a = (a_x, a_y)$.

Viene ora definito l'insieme a_seq(:,:) delle singole azioni a_i che il robot può compiere nel seguente modo:

$$\begin{array}{lll} a_1 = (+1,0) & \longrightarrow & \text{Destra.} \\ a_2 = (0,+1) & \longrightarrow & \text{Alto.} \\ a_3 = (-1,0) & \longrightarrow & \text{Sinistra.} \\ a_4 = (0,-1) & \longrightarrow & \text{Basso.} \\ a_5 = (0,0) & \longrightarrow & \text{Fermo.} \end{array} \tag{3.2.8}$$

Il movimento del robot si modella utilizzando anche in questo caso l'approccio Markoviano di localizzazione, che si basa sul calcolo della distribuzione di probabilità di tutte le possibili posizioni che il robot può assumere. Abbiamo appena visto che la Belief va aggiornata ad ogni lettura sensoriale; ora parleremo di come verrà aggiornata in caso di movimento. Esso viene modellato attraverso una probabilità condizionata chiamata $P_a((x,y)|(\widetilde{x},\widetilde{y}))$ che denota la probabilità che l'azione **'a'** una volta applicata nello stato di partenza $(\widetilde{x},\widetilde{y})$, porti il robot nel nuovo stato (x,y).

La $P_a((\text{x,y})|(\widetilde{x},\widetilde{y}))$ si ottiene sempre dal modello cinematico.

In assenza di errori odometrici il modello cinematico segue queste equazioni:

$$\begin{cases} x(k+1) = x(k) + a_x(k) & \iff x(k+1) \in \{1,2,..,L_x\} \\ y(k+1) = y(k) + a_y(k) & \iff y(k+1) \in \{1,2,..,L_y\} \end{cases} \tag{3.2.9}$$

Con le seguenti transizioni patologiche che si verificano quando il robot applica un'azione che lo porterebbe ad uscire dall'ambiente.

$$\begin{cases} x(k+1) = x(k) & \iff x(k+1) \notin \{1,2,..,L_x\} \\ y(k+1) = y(k) & \iff y(k+1) \notin \{1,2,..,L_y\} \end{cases} \tag{3.2.10}$$

Si consideri che le componenti a_x o a_y non possono essere non nulle contemporaneamente. Quindi si ha per l'evoluzione della componente x(k):

$$a = (-1,0) = \begin{cases} x(k+1) = x(k) - 1 & \iff x(k+1) \in \{1,2,..,L_x\} \\ x(k+1) = 1 & \iff x(k+1) \leq 1 \\ y(k+1) = y(k) \end{cases} \tag{3.2.11}$$

$$a = (+1, 0) \begin{cases} x(k+1) & = x(k) + 1 \quad \Longleftrightarrow x(k+1) \in \{1, 2, .., L_x\} \\ x(k+1) & = L_x \quad \Longleftrightarrow x(k+1) \geq L_x \\ y(k+1) & = y(k) \end{cases} \tag{3.2.12}$$

Per l'evoluzione della componente y(k):

$$a = (0, -1) = \begin{cases} x(k+1) & = x(k) \\ y(k+1) & = y(k) - 1 \quad \Longleftrightarrow y(k+1) \in \{1, 2, .., L_y\} \\ y(k+1) & = 1 \quad \Longleftrightarrow y(k+1) \leq 1 \end{cases} \tag{3.2.13}$$

$$a = (0, +1) \begin{cases} x(k+1) & = x(k) \\ y(k+1) & = y(k) + 1 \quad \Longleftrightarrow y(k+1) \in \{1, 2, .., L_y\} \\ y(k+1) & = L_y \quad \Longleftrightarrow y(k+1) \geq L_y \end{cases} \tag{3.2.14}$$

Il modello cinematico appena descritto aggiorna lo stato del robot nel caso ideale, ovvero per l'$errore_{spostamento} = 0$ (OFF).

Questi casi particolari sono stati implementati nel file 'Aggiornamento_stato_r.m'.

A questo modello cinematico, nel caso reale va poi aggiunto il disturbo odometrico. In presenza di errori sulle misure odometriche l'$errore_{spostamento} = 1$ (ON) le equazioni dell'aggiornamento di stato diventano.

$$\begin{cases} x(k+1) & = x(k) + a_x(k) + n_x(k) \quad \Longleftrightarrow x(k+1) \in \{1, 2, .., L_x\} \\ y(k+1) & = y(k) + a_y(k) + n_y(k) \quad \Longleftrightarrow y(k+1) \in \{1, 2, .., L_y\} \end{cases} \tag{3.2.15}$$

La componente n(k)=$[n_x(k); n_y(k)]$ è il disturbo che agisce sullo spostamento e può assumere un valore intero compreso tra -1 e +1 ed è modellato come nel caso Mono-dimensionale: vale a dire che ogni volta che il robot si sposta ha la probabilità 0.1 di restare fermo, 0.7 di muoversi correttamente e 0.2 di muoversi più del necessario.

Quindi anche il movimento del robot è modellato da un calcolo della probabilità. Si tratta in particalar modo di una probabilità condizionata $P_a(l|\widetilde{l})$ che denota la

probabilità che l'azione $a = a(k)$ se eseguita in posizione $x(k) = \widetilde{l}$ porta $x(k)$ in una posizione $x(k+1) = l$ all'istante successivo.

Ad ogni azione eseguita il robot aggiorna la sua densità di probabilià Belief seguendo la regola:

$$\widehat{Bel}(x,y) \longleftarrow \sum_{\widetilde{l}_x=1}^{L_x}\sum_{\widetilde{l}_y=1}^{L_y} P_a((x,y)|(\widetilde{l}_x,\widetilde{l}_y))Bel(\widetilde{l}_x,\widetilde{l}_y) \;\; \forall x \in [1,L_x] \;\; \forall y \in [1,L_y] \tag{3.2.16}$$

Il termine $\widehat{Bel}(x,y)$ è la probabilità di occupare la posizione (x,y) all'istante succesivo dopo aver eseguito l'azione a ma poco prima di acquisire le successive letture sensoriali. Tale probabilità serve per aggiornare $P_{prior}(x,y)$.

3.2.3 Tipi di controllo utilizzati: passivo, attivo, casuale

Considerando la parte relativa al controllo, anche per il caso Bi-dimensionale si è implementato dapprima un approccio passivo. Infatti, si è cercata una sequenza che permettesse al robot di autolocalizzarsi partendo da una configurazione qualsiasi dell'ambiente. Studiando tutte le varie combinazioni spazio di stato azioni che il robot può effettuare si è creata una sequenza di controllo passivo che permette al robot di portarlo verso una diminuzione dell'Entropia della posa per fare in modo che diminuisse l'incertezza legata alla stima della sua posizione.

Una volta notato che l'algoritmo passivo permette al robot di percepire dopo una serie di passi la propria configurazione si è passati all'implementazione del controllo attivo. Tale controllo si avvale ancora una volta della selezione dell'azione migliore da compiere sulla base di un'attenta selezione automatica del movimento compiuta dal robot.

Il vettore U delle funzioni di utilità in questo caso si riferisce alle 5 possibili azioni di

controllo che il robot può effettuare. Il vettore dei costi è stato annullato[17] in quanto la funzione obiettivo da minimizzare è quella relativa alla sola Entropia. Infatti si seleziona in maniera attiva, on line e per ogni istante k della simulazione, quella azione di controllo a_k^* che permette al passo successivo di ottenere un'incertezza nella posa sempre minore. Il tutto viene effettuato attraverso il seguente algoritmo, qui riassunto nei suoi passi principali.

INIZIO dell'algoritmo

Inizializzazione dati:

In primo luogo si inizializzano tutte le variabili utilizzate. Tra queste vi è anche lo stato iniziale del robot che, per tutti i tipi di controllo, è inizializzato in modo random attraverso la funzione disponibile in Matlab 'rand' e precisamente si ha:

- $x(0) = ceil(L_x. * rand(1,1)) \longrightarrow$ che restituisce un numero casuale intero tra $[1, l_x]$.
- $y(0) = ceil(L_y. * rand(1,1)) \longrightarrow$ che restituisce un numero casuale intero tra $[1, l_y]$.

Passiamo alla descrizione dei passi principali dell'algoritmo di controllo attivo.

$\forall k$, passo della simulazione, con k che va da 1 a t_{sim}:

Predizione dell'Entropia attesa $E_a(k)$ $\forall a$

$\forall a_i \in [a\_seq]$, possibile azione di controllo, con $i = \{1, 2, .., length(a\_seq)\}$ eseguire:

[17] Il coefficiente α è stato posto uguale a 0.

1. **Predizione** E. Primo passo per la stima dell'Entropia attesa futura $E_{a_i,tot}(k+1)$ dopo aver ipotizzato una azione di controllo $a_i(k) = a_i$.

   In questo passo si calcola la $Bel_{a_i}(x_{k+1}, y_{k+1})$[18] $\forall x \in [1 : L_x]$ e $\forall y \in [1 : L_y]$. Vale come aggiornamento della stima della distribuzione di probabilità dopo aver effettuato un singolo spostamento; il codice per il calcolo della $Bel_a(x, y)$ è stato implementato in Motion_model_seq2D.m. L'inizializzazione di $Bel_{a_i}(x_0, y_0)$ è uguale a quello per la $P_{prior}(x_0, y_0)$ e della $P_{posterior}(x_0, y_0)$ reale e vale $\frac{1}{L_x * L_y}$ per tutte le configurazioni possibili. La probabilità Belief di stare in una certa posizione all'istante $k = 0$ è uniformemente distribuita in tutto lo spazio delle possibili pose.

2. **Predizione** $E_{a_i,s}(k+1)$. Il secondo passo calcola la p(s) $\forall s \in [0, 1]$; determina l'aggiornamento della Belief dopo aver acquisito le letture laser. Tale operazione è eseguita nel file Perception_model_seq2D che aggiorna la probabilità ad ogni sensing, ovvero calcola la probabilità $Bel_{a_i,s}(l = x_{k+1}, y_{k+1})$ di trovarsi in posizione l all'istante $(k + 1)$ dopo aver letto il dato relativo al sensore porta s_{k+1}.

3. **Predizione** $E_{a_i}(k+1)$. Questo terzo passo calcola la $E_{a_i}(k+1)$ attesa dopo aver ipotizzato l'azione $a_i(k)$ considerata. In pratica in questo ultimo step si calcola l'Entropia che si avrebbe all'istante successivo $(k + 1)$ se si applica l'azione a(k) dalla posizione (x_k, y_k) all'istante corrente k. Si tratta di un valore $E_{a_i} = E_{a_i,tot}(k + 1)$ composto da due contributi: il primo è $E_{a_i,s=1}(k + 1)$dato dall'Entropia attesa al passo successivo se si avrà una lettura laser $s = 1$; l'altro se si avrà invece una lettura $s = 0$, quindi un predizione $E_{a_i,s=0}(k + 1)$. En-

[18]Corrisponde al calcolo di $\widehat{Bel}(x_{k+1}, y_{k+1})$ nella sezione relativa al modello di movimentazione utilizzato.

trambe dipendono dalla probabilità di percepire un certo dato dal dispositivo sensoriale.

4. **Aggiornamento della funzione Utilità per il controllo attivo**. Si tratta di inizializzare la funzione $U_{a_i,k} = E_{a_i,tot}(k+1) + \alpha C(r_{i,j})$. La prima parte è data dalla somma algebrica $E_{a_i,s=1}(k+1) + E_{a_i,s=0}(k+1)$, mentre la seconda è data dal costo associato ad uno spostamento $r_{i,j}$. In questo caso, il costo non influisce sulla scelta della soluzione ottima in quanto si tratta di selezione di tutti spostamenti di lunghezza unitaria. Si è posto $\alpha = 0$ volutamente nella funzione obiettivo da minimizzare anche per un'ulteriore considerazione: si tratta infatti di un algoritmo di stima basata solo sulla minimizzazione dell'Entropia e non interessa di diminuire il costo associato al percorso effettuato ma di trovare un percorso che permetta di stimare nel modo più veloce la posa del robot.

Controllo attivo: selezione della $a^*(k)$

Una volta calcolata la funzione obiettivo $U_{a_i}(k)$, vediamo la dipendenza diretta della azione $a_i(k)$ considerata; la parte relativa al controllo attivo è stata implementata in diverse versioni, le principali delle quali trovano l'implementazione nei seguenti file: Ottima_seq_sbocco_a02D, Ottima_seq_senza_a0_2D, Ottima_seq2D. Queste versioni sono state implementate fondamentalmente per raggirare i classici problemi di stallo del robot in una certa posizione, infatti svolgono il ruolo di riattivare il controllo se il robot non riesce più a selezionare l'azione migliore per il controllo.
Alla base di esso vi è la selezione dell'azione ottima all'istante k, $a^*(k)$ che permette di scegliere la $a^* = \arg\min_{a_i} U_i$ che minimizza la funzione obiettivo $\forall i$. Nel caso particolare nel quale più di un'azione $a_i \in [a\_seq]$ all'istante k se applicata porta ad

avere due valori uguali per la $U_{a_i}(k)$ la scelta di una di esse è completamente casuale e indipendente dalle precedenti. Tale situazione è molto importante nel caso simmetrico perché si ha sempre che all'istante di partenza per $k = 0$ tutte le funzioni Utilità per ogni spostamento non nullo hanno lo stesso valore; quindi se si simula per diverse volte il controllo per una stessa condizione inziale dello stato si hanno delle soluzioni differenti. Infatti alcune volte il robot deciderà di andare a destra e poi continuerà ad andarci, perché una volta che effettua uno spostamento in quella direzione si avvicina di più ad un landmark, il che implica una diminuzione netta dell'Entropia; altre volte invece decide di fare uno spostamento a destra e dopo di esso troverà che l'Entropia associata a spostamenti dello stesso tipo è minore.

Una volta selezionata la a(k) ottima si passa all'applicazione vera e propria dell'azione scelta.
Si ha l'aggiornamento della posizione effettiva del robot $(x_r(k), y_r(k))$ e della sua stima $(x_s(k), y_s(k))$; si simula la traiettoria SOGGETTA AL CONTROLLO ATTIVO. Per ogni azione e lettura laser disponibile, come abbiamo sempre detto, vi è un aggiornamento della Belief secondo il modello della localizzazione Markoviana descritto nel paragrafo precedente, dunque si devono calcolare le $P_{posterior,s}(k, (x, y))$ e le $P_{prior,a}(k + 1, (x, y))$ effettive, reali, per ogni configurazione possibile.

Rinizializzazione delle variabili per il passo successivo (k+1)

Inizializzazione della $Bel_{a,k+1}(x, y) = P_{prior}(k + 1)$ per l'istante successivo della simulazione. Si torna, dunque, al primo passo dell'algoritmo per un'altra iterazione relativa al passo (k+1) e successiva scelta dell'azione migliore a_{k+1} e questo si ripete fino a quando il robot non è in grado di trovare la sua posizione con certezza; ovvero

fino a quando l'Entropia associata alla posa non è prossima ad un valore nullo, il che implica una buona certezza circa la configurazione dell'automa.

FINE dell'algoritmo.

Dopo diverse simulazioni, soprattutto nel caso reale con presenza degli errori sia sulle misure odometriche che su quelle sensoriali, ci si è resi conto che il robot, alcune volte, nel controllo *greedy* tendeva a tornare in posizioni precedentemete assunte. Si è allora implementato un controllo a *target-point* ovvero un tipo di controllo basato sull'applicazione e la selezione di una sequenza di azioni elementari al fine di ridurre l'Entropia in modo più veloce. Si tratta di effettuare gli stessi passi dell'algoritmo appena descritto dove però la selezione del movimento migliore che il robot deve effettuare per meglio localizzarsi non è basato sulla scelta multipla di singole azioni elementari ma di un'insieme di azioni elementari del tipo $a_i(k) = ((+1,0),(+1,0),(+1,0),(+1,0),(+1,0),(0,-1),(0,-1) \longrightarrow (+5,-2)$ che corrisponde ad un comando del tipo a='vai avanti di 5 e sotto di 2 rispetto alla posizione attuale'.
In conclusione, si è visto che in molti casi, nonostante il controllo *greedy* sia comunque efficiente, quello a *terget-point* migliora, anche se di poco, la soluzione.

Per quanto riguarda la simulazione casuale, quest'ultima si è ottenuta tramite una scelta random dell'azione ad ogni passo k della simulazione. Fondamentalmente si è implementata come verifica che il controllo con una selezione accorta dell'azione da far fare al robot fosse migliore, e di quanto lo fosse, rispetto ad azioni scelte casualmente. Si utilizza lo stesso schema per il controllo, ma si evita la parte della selezione attiva dell'azione (nel caso del controllo attivo) o dell'applicazione di una seguenza presta-

bilità (caso del controllo passivo); in questo caso l'evoluzione della traiettoria casuale è data da $(x_c(k), y_c(k))$, tramite le seguenti equazioni:

- $x_c(k+1) = x_c(k) + a_{x,c}(k) \longrightarrow$ la cui stima è poi restituita in $x_{c,s}(k)$.
- $y_c(k+1) = y_c(k) + a_{y,c}(k) \longrightarrow$ la cui stima è poi restituita in $y_{c,s}(k)$.

Le componenti dell'azione $a_c(k) = (a_{x,c}(k), a_{y,c}(k))$ sono le azioni scelte in modo aleatorio per ogni istante k della simulazione all'interno di tutti i possibili movimenti del robot, vale a dire $a_c(k) \in [a\_seq]$.

I vari controlli implementati, come per il caso 1D, sono stati poi simulati, vagliati a verifiche di ogni tipo per notarne la validità. I risultati vengono illustrati nel prossimo capitolo.

Capitolo 4

Risultati simulativi

Questo capitolo affronta il problema dell'Autolocalizzazione da un punto di vista completamente sperimentale-simulativo; si vuole dimostrare come la scelta di una particolare azione ***'a'****, ottenuta tramite l'algoritmo di selezione basato sulla minimizzazione dell'Entropia proposta dal controllo attivo implementato, migliori in modo significativo i risultati della 'Global Localization'. Gli esperimenti simulativi descritti qui di seguito sono stati implementati per modelli di robot semplificati liberi di navigare all'interno di ambienti Mono/Bi-dimensionali.*

I modelli considerati Mono/Bi-dimensionale dell'ambiente e del robot, e l'insieme delle variabili utilizzate, è stato già introdotto e descritto nel capitolo precedente; nei prossimi paragrafi si farà dunque riferimento direttamente alle simulazioni eseguite, agli specifici casi considerati, a quelli più significativi, al particolare ambiente dove vengono effettuate e al set di pose iniziali scelte in modo randomico per dimostrare la correttezza dell'algoritmo. Va inoltre ricordato che il codice implementato è molto versatile, si possono infatti effettuare tanti tipi diversi di simulazione per varie configurazioni iniziali, per tipologia dell'ambiente, per definizione dei landmarks, per durata della simulazione, per casi ideali (assenza di rumori) e reali (presenza di errori di misura). Queste variazioni si ottenegono inizilizzando nel modo opportuno e desiderato le variabili in gioco nel file Inizializzazione_dati_seq.m.

4.1 Caso Mono-dimensionale

4.1.1 Descrizione e discussione delle simulazioni

L'insieme delle simulazioni effettuate per l'analisi del caso Mono-dimensionale vengono ora descritte seguendo lo schema logico-temporale che ha portato alla soluzione del problema e alle sue conclusioni.
Il primo passo è stato quello di affrontare il problema relativo al caso più semplice, cioè quello ideale; una volta implementato l'algoritmo le prime simulazioni relative all'autolocalizzazione sono state effettuate in assenza di qualsiasi disturbo sui dati percepiti dal robot. Una volta verificata la bontà del codice scritto per il caso ideale l'attenzione è stata rivolta allo studio e all'analisi dei casi in cui sono presenti i disturbi. Le simulazioni, infatti, sono state effettuate successivamente per tre casi: nei primi due gli errori inficiano o solo sul modello sensoriale o solo su quello di movimentazione del robot, ma non coesistono contemporaneamente; nel terzo, il caso reale, si considerano sia letture sensoriali rumorose che spostamenti del robot soggetti ad errore. Quest'ultimo è oggetto di studio di maggior interesse in quanto, in natura, durante la localizzazione il movimento che compie il robot e i dati che egli percepisce saranno entrami soggetti a fenomeni aleatori, quindi è importante considerarli contemporaneamente.

Analisi dell'Entropia e di come viene influenzata dalla struttura dell'ambiente e dal modello del robot considerato.

Si può in prima battuta notare come la presenza o l'assenza di tali errori sensore/movimento influisca enormemente nel caso del calcolo dell'Entropia associata alla posa; infatti, il grado di incertezza che tale grandezza esprime, cresce enormemente

per i casi rumorosi rispetto a quelli ideali in cui l'Entropia va velocemente a zero e la posa viene stimata in maniera esatta in pochi istanti. Bisogna poi fare una distinzione riguardo al calcolo dell'Entropia ottenuto rispetto al tipo di errore considerato nella simulazione; anche il tipo di disturbo esaminato ha un diverso effetto su tale valore: infatti, in presenza dei soli rumori sul sensore[1], l'Entropia associata risulta, per condizioni iniziali randomiche, sempre inferiore a quella calcolata con disturbi odometrici, ovvero nei casi in cui il disturbo **'n'** può assumere valori diversi da zero secondo una certa percentuale considerata dal modello di movimentazione del robot, quindi quando $n \neq 0$[2]. In tale situazione l'Entropia non si azzera mai completamente, e in molte delle simulazioni tale valore oscilla spesso attorno ad un valore medio prossimo all'unità; gli errori di stima che ne conseguono non si annullano mai e dunque la coincidenza tra valore stimato e posizione effettiva della posa del robot è difficilmente raggiungibile.

L'Entropia e l'errore di stima tendono ad azzerarsi durante la simulazione nei casi ideali e anche, con una convergenza molto meno rapida, nei casi in cui sono presenti errori esclusivamente sulle letture sensoriali. In questa situazione, il robot, nonostante percepisca i dati dall'ambiente esterno in modo non perfetto, riesce, comunque, ad usarli per localizzarsi se tali incertezze non sono combinate con incertezze sul movimento che compie. Nei rimanenti casi considerati, ovvero quelli relativi a situazioni dove sono presenti errori di tipo odometrico, o dove coesitono sia letture sensoriali rumorose che movimenti soggetti ad errori di spostamento, oltre al fatto che il valore della stima in pratica non coincide mai con quello reale, può anche accadere che dopo un certo istante della simulazione il valore dell'Entropia associata alla posa cresca in maniera non controllata. Quest'ultimo fenomeno si verifica perché l'errore sul mo-

[1] $Errore_{spostamento} = 0$; $Errore_{sensore} = 1$.
[2] $Errore_{spostamento} = 1$.

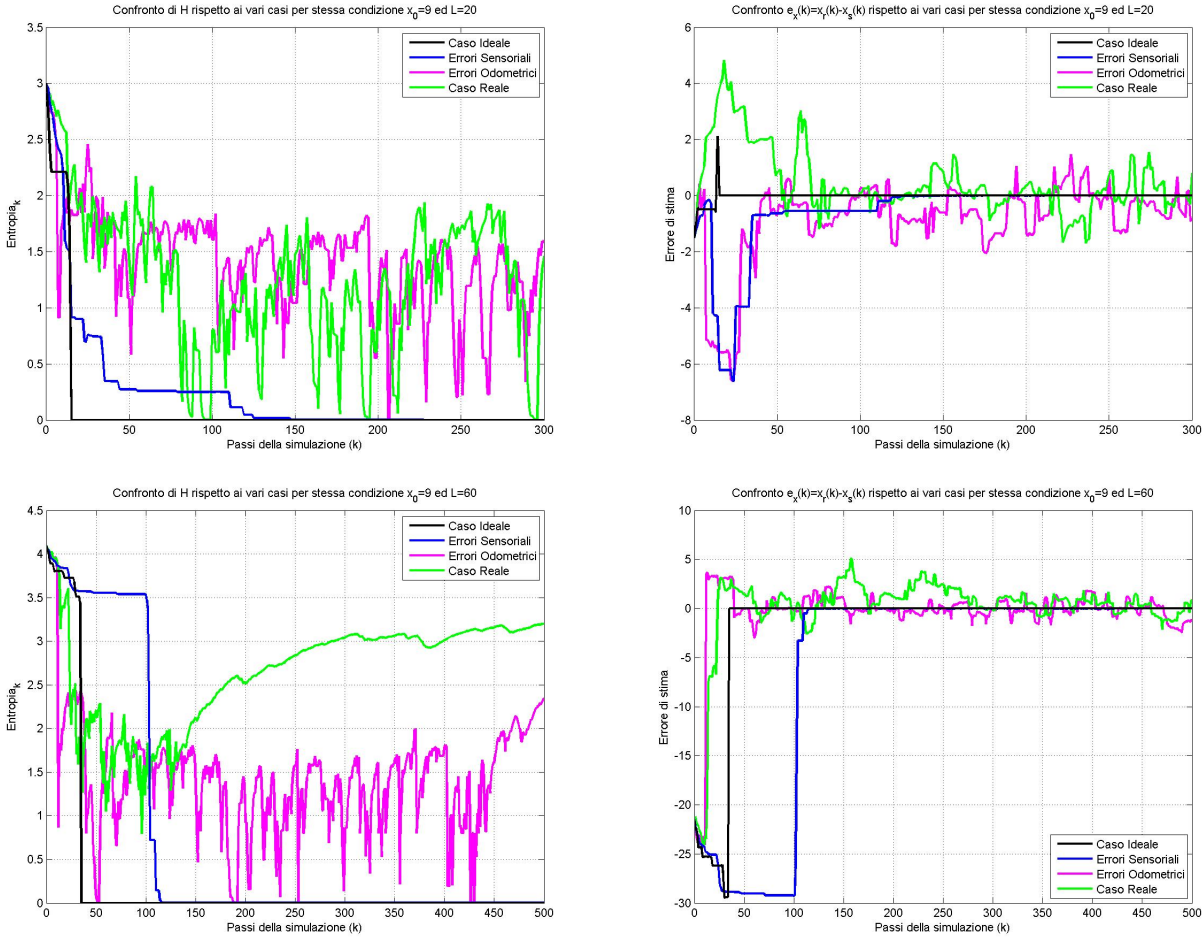

Figura 4.1: Studio della funzione Entropia per i vari casi analizzati.

vimento inficia la stima della posa del robot, ovvero, una volta che questo riesce a scovarla con esattezza poi può accadere che ne perda comunque la traccia, o che non la stimi più perfettamente.

I grafici riportati in Fig. 4.1 sono relativi a due esempi simulativi in cui l'ambiente ha per entrambe le situazioni una configurazione molto simile per quanto riguarda la posizione dei landmarks-porta, ma una differente misura per la lunghezza del corridoio che nel primo caso misura $L = 20$[m], mentre nel secondo $L = 60$[m]. Questo

perché si vuol far notare come l'Entropia calcolata dipenda anche dalla dimensione dello spazio di stato relativo alle possibili configurazioni che il robot può assumere.
Nelle figure 4.1 si vede subito che l'analisi appena fatta per il caso ideale vale ancora in entrambe le simulazioni: si ha infatti che l'Entropia associata all'incertezza della posa durante la simulazione tende ad andare sempre a zero con il trascorrere del tempo anche per ambienti più complicati, si verifica cioè che l'Entropia $H \longrightarrow 0$ con $k \longrightarrow \infty$. Per ambienti più 'complicati' si intendono quegli spazi di lavoro per i quali è associata un'Entropia iniziale più elevata. Nel caso Mono-dimensionale, dove l'ambiente è visto come un corridoio statico all'interno del quale troviamo solo delle features-porta, l'Entropia iniziale associata è legata alla lunghezza L del corridoio. Questo perché lo studio si fa rispetto alla posizione x_k assunta dal robot all'istante k della simulazione considerando noto l'orientamento, ovvero si ipotizza che esso sia rivolto verso la parete che delimita il corridoio a destra, in questo modo l'Entropia associata alla posa dipende solo dall'incertezza di occupare una certa posizione $x \in [1, L]$. Nel caso del corridoio più lungo, le possibili pose che il robot può assumere sono in numero maggiore rispetto a quelle che si avrebbero con un corridoio di dimensione ridotta; essendo la probabilità di occupare una certa posa all'istante iniziale inversamente proporzionale alla lunghezza L del corridoio si ha quindi che la probabilità singola Bel(x) che ogni cella x, in cui è suddiviso il corridoio, ha di essere occupata dal robot, per $k = 0$, è $Bel(x_i) = \frac{1}{L}$. Detto ciò, l'incertezza iniziale che si ottiene dal calcolo delle singole probabilità $Bel(x_i)$ secondo l'argomento dell'Entropia '$Bel(x_i) * log(Bel(x_i))$' è maggiore se si tratta di ambienti con dimensioni più grandi. Dalle simulazioni si vede, inoltre, che per un corridoio lungo $L = 20$[m] l'Entropia iniziale è circa 3, se si triplica la sua grandezza, $L = 60$[m], tale valore aumenta di 1/3.
Per un ambiente di dimensione L=60[m] sono state riportate altre simulazioni casuali

per i quattro casi precedentemente considerati, dall'ideale al reale; questo permette di fare un'altra costatazione. Qui per lunghe simulazioni si possono verificare le seguenti situazioni: o il robot inconsapevolmente va verso una zona[3] e vi permane, o si trova già in tale zona con una condizione iniziale randomica 'fortunata', oppure ci entra in modo transitorio, per poi uscirne. Per tutte le situazioni si vede che il grafico dell'Entropia relativo al caso ideale balza a zero in pochi istanti; anche il caso con errori solo sul sensore l'andamento entropico converge a zero ma è più lento. In presenza di disturbi sull'odometria si ha, invece, che per la prima situazione l'Entropia oscilla rispetto ad un valore medio costante ma non presenta divergenze, mentre per la seconda l'Entropia o oscilla rispetto ad un valore medio costante per poi divergere o diverge da subito. Questi vengono riportati nelle figure 4.2.

Ricordiamo che lo studio dell'Entropia è stato effettuato rispetto ad azioni del robot scelte in maniera randomica; le considerazioni evidenziate dalle simulazioni sono state fatte, infatti, per robot liberi di vagare e non soggetti ad alcun tipo di controllo specifico, ma con azioni selezionate in modo del tutto aleatorio (controllo casuale); infatti, lo scopo era verificare se il robot fosse in grado di localizzarsi in modo autonomo all'interno dell'ambiente. È dunque interessante applicare un controllo che piloti la scelta delle azioni in modo intelligente secondo una politica che minimizzi l'Entropia e permetta di conseguire i seguenti obiettivi: velocizzare la stima della posa per le situazioni 'ideali' dove comunque il robot riesce ad autolocalizzarsi; permettere al robot soggetto a disturbi e per il quale sarebbe impossibile autolocalizzarsi anche per tempi infiniti, di riuscirci nel più breve tempo possibile.

L'attenzione ora sarà rivolta al controllo attivo implementato per vedere se tale

[3]Zona dei landmarks permette al robot di localizzarsi in maniera migliore

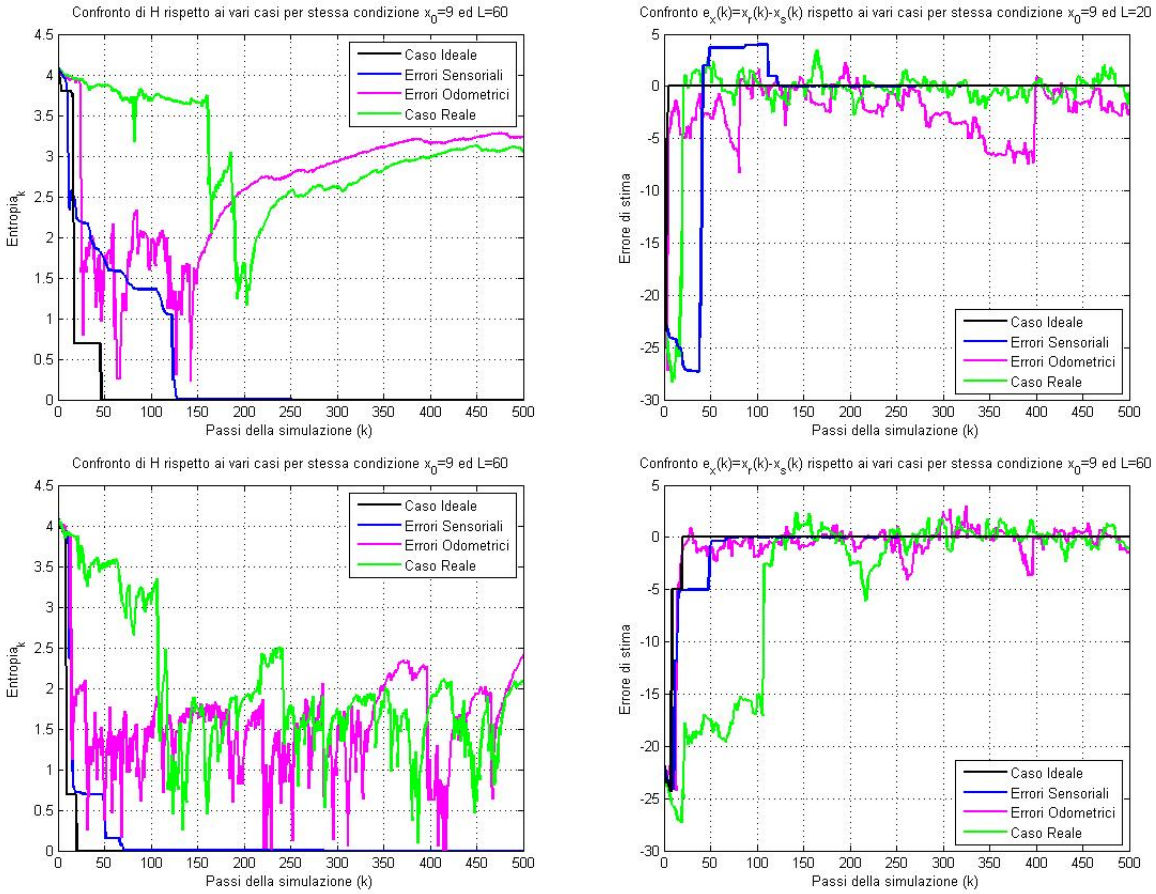

Figura 4.2: Entropia per due situazioni distinte con $L = 60$[m].

controllo adempie in pieno lo scopo prefissato, ovvero, l'autolocalizzazione del robot in maniera autonoma ed efficiente.

L'implementazione del controllo attivo on-line permette la stima della posa in qualsiasi situazione.

L'implementazione del CONTROLLO ATTIVO on-line, che consiste nella selezione autonoma da parte del robot di un'azione, tra quelle consentite, rispetto ad una politica di controllo stabilita, ha permesso di ottenere ottimi risultati per la localizzazione; si è ottenuta per tutti i casi la minimizzazione dell'Entropia associata alla posa e

nei casi in cui essa già andava a zero ha permesso di aumentare la rapidità di tale convergenza.

Nelle simulazioni effettuate, e qui riportate, vi sono esempi di come la politica di controllo implementata permetta al robot di selezionare l'azione più opportuna nel set delle possibili scelte di controllo. Il robot può scegliere tre mosse: può decidere di restare fermo ($a = 0$), di fare un passo in avanti ($a = +1$[m]), di farne uno indietro ($a = -1$[m]). La selezione dipende dalla funzione 'Utilità' $U_a = E_a(H) + \alpha C(a)$; essa è alla base del calcolo della $a_{star} = a^*$.

Sono state implementate tre diverse versioni di processamento della funzione U_a per la selezione dell'azione ottima a^*:

1. Ottima_seq (tipo 1);
2. Ottima_seq_sblocco_a0 (tipo 2);
3. Ottima_seq_senza_a0 (tipo 3).

Istante per istante per il primo e il secondo caso a^* è scelta come argomento minimo tra $[U_{a=-1}, U_{a=0}, U_{a=+1}]$ con la seguente differenza: mentre nel primo caso il robot può decidere di stare sempre fermo in una certa posizione, se l'azione ottima risulta costantemente $a = 0$; nel secondo non può sostare troppo a lungo in una certa posa perché l'algoritmo di controllo per la selezione viene riattivato.

Tralasciando questa differenza, la selezione delle azioni avviene per entrambi in questo modo: se ad un certo istante k della simulazione la $U_a(k)$ assume il valore uguale per tutte e tre le azioni possibili, ognuna di esse avrà la probabilità $p = 0.\overline{333}$ di essere scelta come azione ottima dal robot; se invece il più piccolo valore per $U_a(k)$ è attribuibile contemporaneamente per due sole azioni, il robot sceglierà ognuna di queste con una probabababilità $p = 0.5$; se invece $U_a(k)$ ha tre valori distinti, l'algoritmo di

controllo farà fare al robot l'azione che minimizza la U_a, quindi $a_k^* = argmin_a(U_{a,i}(k))$. Dalle simulazioni effettuate, il primo approccio non funziona mai in modo efficiente per tutte le possibili situazioni, pertanto va abbandonato immediatamente. Questo, infatti, non considera la possibilità di trovarsi in un minimo relativo per la funzione $U_a(k)$ e dunque sceglie correttamete $a = 0$ come soluzione ottima, anche se in realtà lo è solo localmente, piantando il robot in una posa. Come si vede nelle figure 4.3 l'algoritmo smette di selezionare correttamente e il robot si ferma e stima una posizione con un errore di stima tanto più grande quanto più sono presenti i disturbi nel modello simulativo, o per la complicanza dell'ambiente.

La seconda politica aggira il problema e lo evita 'risvegliando il robot', ovvero la politica di controllo implementata controlla gli ultimi movimenti del robot, se questo sta per troppo tempo in una posizione, riattiva il robot facendolo muovere nuovamente durante la simulazione fino a quando l'Entropia non si azzera completamete. Questa politica risulta migliore della precedente ma ha il difetto che se il robot trova effettivamente il minimo globale dell'Entropia, prima che lo riconosca come tale, riattiva la selezione e movimenta il robot anche se non sarebbe neccessario. Tale situazione si può notare nella seconda simulazione riportata in figura 4.4 dove si hanno gli spikes per aver costretto il robot a muoversi.

In definitiva si è deciso di elimare la possibilità di scegliere la $a = 0$ e nel caso in cui all'istante k $U_{a=-1}$ risulti uguale a $U_{a=+1}$ il robot ha la probabilità 0.5 di decidere l'una o l'altra mossa. Dalle varie simulazioni, un esempio è riportato negli ultimi due riquadri in basso della figura 4.4, questa politica è risultata la migliore, e di tanto, tra le tre implementate; permette rapide convergenze a zero dell'Entropia associata alla stima della posa sia in presenza di disturbi che in loro assenza, nessuno spike dovuto alla riattivazione e leggeri overshot relativi all'errore di stima. La posa percepita

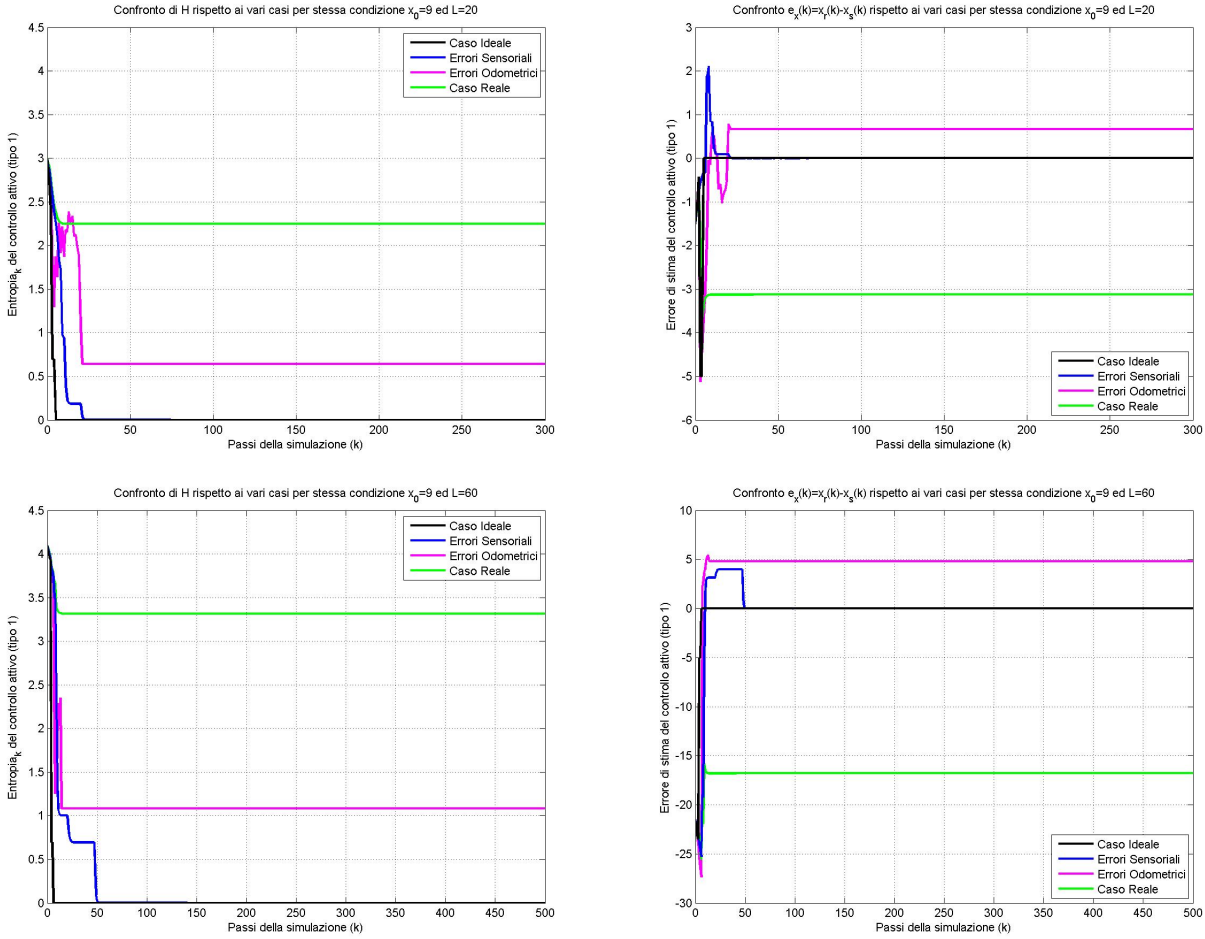

Figura 4.3: Simulazione per i vari tipi di controllo attivo implementati.

dal robot in pochi istanti risulta coincidente con quella effettivamente occupata per tutte le simulazione e in tutti i casi. Analizzando la scelta delle azioni di controllo selezionate nel tempo dall'algoritmo, si è visto che esse corrispondevano il più delle volte a sequenze elementari tutte di valore -1 oppure di valore +1 una volta che è stata scelta la prima azione a_0 ripettivamente -1 o +1; interpretando tale significato il risultato si poteva già intuire, una volta che il robot decide di fare ad esempio un passo avanti, non potendo restare fermo potrà solo decidere di proseguire in avanti o tornare indietro, se va avanti acquisisce nuovi dati sensoriali, se torna inditro non

ha informazioni aggiuntive e in più ha un'incertezza ulteriore dovuta ad errori di tipo odometrico, quindi l'Entropia associata alla posa assunta precedentemete è maggiore rispetto a quella che si otterebbe con un successivo passo in avanti.
Questa politica, come si vedrà tra poco, è risultata anche la migliore tra i casi considerati per l'ambiente Mono-dimensionale.

Tutte le simulazioni relative ai tre tipi di controllo sono state confrontate per la stessa condizione iniziale $x_0 = 9$ e per una situazione $k = 0$ che avesse $U_{a=-1} = U_{a=+1}$, così il robot era libero di scegliere se andare a destra o a sinistra in modo indifferente.

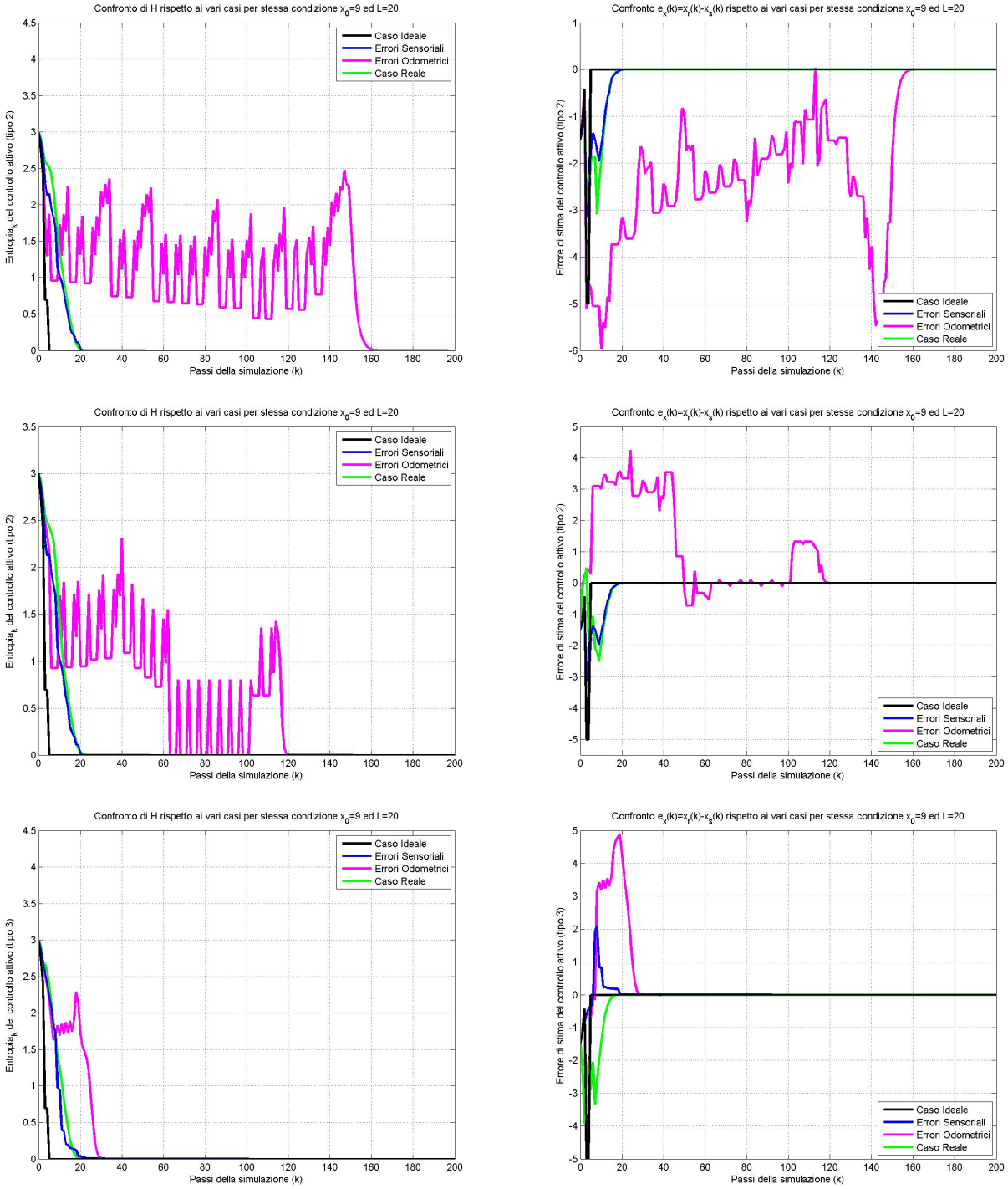

Figura 4.4: Simulazione per i vari tipi di controllo attivo implementati.

Confronto tra controllo Attivo e Passivo:

Per verificare la bontà del controllo attivo di tipo 3, bisogna chiedersi se tale politica di controllo sia effettivamente migliore di un controllo passivo scelto off-line, ad esempio da un operatore esperto. Bisogna confrontare i due tipi di controllo.

La scelta di un buon controllo passivo è stata effettuata dopo un'analisi molto accurata sulle possibili politiche ottime che si possono far eseguire al robot per permettergli di autolocalizzarsi all'interno dell'ambiente; sono stati esaminati tanti casi e le politiche migliori sono risultate delle strategie nelle quali si hanno sequenze significative di azioni possibili. Vanno dunque escluse sequenze contenenti $a = 0$. Lo studio è stato effettuato per 5 possibili sequenze di controllo passivo:

1. 'Seq di controllo 1' (Seq_1): $a_k = -1$ per ogni passo k della simulazione;
2. 'Seq di controllo 2' (Seq_2): $a_k = +1$ per ogni passo k della simulazione;
3. 'Seq di controllo 3' (Seq_3): $a_k = +1$ per 10 volte e $a_k = -1$ per 30 volte ogni 40 passi;
4. 'Seq di controllo 4' (Seq_4): $a_k = +1$ per 30 volte e $a_k = -1$ per 30 volte ogni 60 passi;
5. 'Seq di controllo 5' (Seq_5): $a_k = -1$ per i primi 20 passi poi tutti $a_k = +1$.

I diversi tipi di controllo passivo sono stati graficati in figura 4.5. Da questa si può notare che le sequenze di controllo del terzo e del quarto tipo erano da eliminare; queste infatti non solo non permettevano la stima della posizione del robot corretta, ma l'Entropia, ad intervalli regolari, diminuiva fino ad annullarsi per poi assumere nuovamente valori positivi; gli spikes relativi al cambio dell'azione sono inevitabili e

chiaramente visibili nell'errore sulla stima. La periodicità dell'Entropia dipende dalla lunghezza delle sequenze base considerate, ovvero 40 per la Seq_3 e 60 per la Seq_4. Per le altre tre sequenze si è visto che le prime due sono risultate le migliori anche se pure l'utima ha un andamento soddisfacente.
Si passa allora alla scelta di quella ottima tra le prime due sequenze; questa verrà settata per il controllo passivo.
La scelta è stata effettuata considerando tre ambienti differenti con medesima lunghezza $L = 60[m]$ stessa condizione iniziale, ma diversa distribuzione dei landamark nella mappa. Le scelte sono tali da permettere di analizzare le seguenti situazioni:

- Si posiziona il robot nel punto centrale del corridoio con distribuzione delle features-porta all'estrema sinistra in posizione $Porta = [7; 15; 18]$;
- Si posiziona il robot nel punto centrale del corridoio con distribuzione simmetrica delle porte delle features-porta in posizione $Porta = [15; 30; 45]$;
- Si posiziona il robot nel punto centrale del corridoio con distribuzione delle features-porta all'estrema sinistra in posizione $Porta = [42; 45; 53]$;

Dai grafici simulativi riportati nelle figure 4.6 per il primo ambiente, si può notare che per sequenze di '+1' l'andamento a zero dell'Entropia non presenta 'gobbe' cosa che invece avviene per sequenze di controllo con tutti spostamenti all'indietro '-1': questo fenomeno si verifica proprio in corrispondenza delle features porta, si vede infatti una prima risalita dell'Entropia associata al 12-esimo passo ovvero $30 - 12 = 18$, l'altra relativa al 15-esimo passo e infine all'istante $k = 23$. Tali gobbe sono dovute anche al modello considerato: il robot quando non è davantati alla porta ha una probablità maggiore di non percepirla in maniera corretta pari a 0.9 rispetto alla situazione complementare, 0.7. Quando il robot trova le features l'Entropia diminuisce

drasticamente anche per la seconda sequenza per poi a andare a zero seguendo lo stesso andamento che si ottiene per la prima. Stessa situazione, ma complementare, si ha per la terza simulazione.
La seconda risulta molto significativa. Nel caso di ambiente simmetrico il robot dà risultati simmetricamente opposti, scegliere di andare a destra o a sinistra è equivalente.

In conclusione le azioni migliori per autolocalizzarsi che il robot può fare risultano o sequenze di tutti +1 o tutti -1. Qui subentra il controllo attivo e la sua validità, in quanto esso permette da solo, on-line e per lo specifico caso simulato, di scegliere la prima o la seconda sequenza. Il controllo attivo da subito seleziona la sequenza di -1 o +1 ottima in base alla mappa e ai landmarks, invece quello passivo sceglie a priori, off-line, di andare o sempre avanti o sempre indietro.

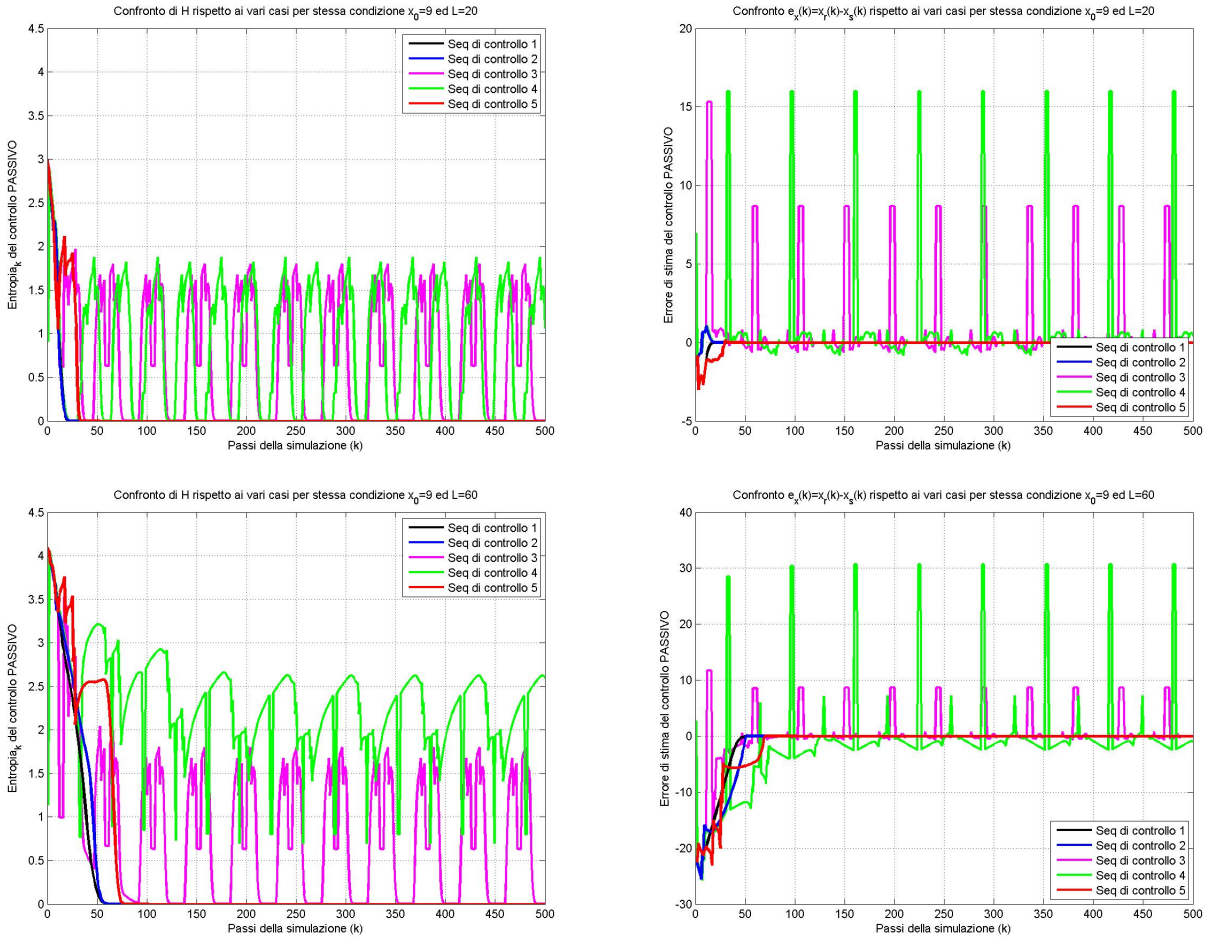

Figura 4.5: Grafico completo dei controlli passivi a confronto.

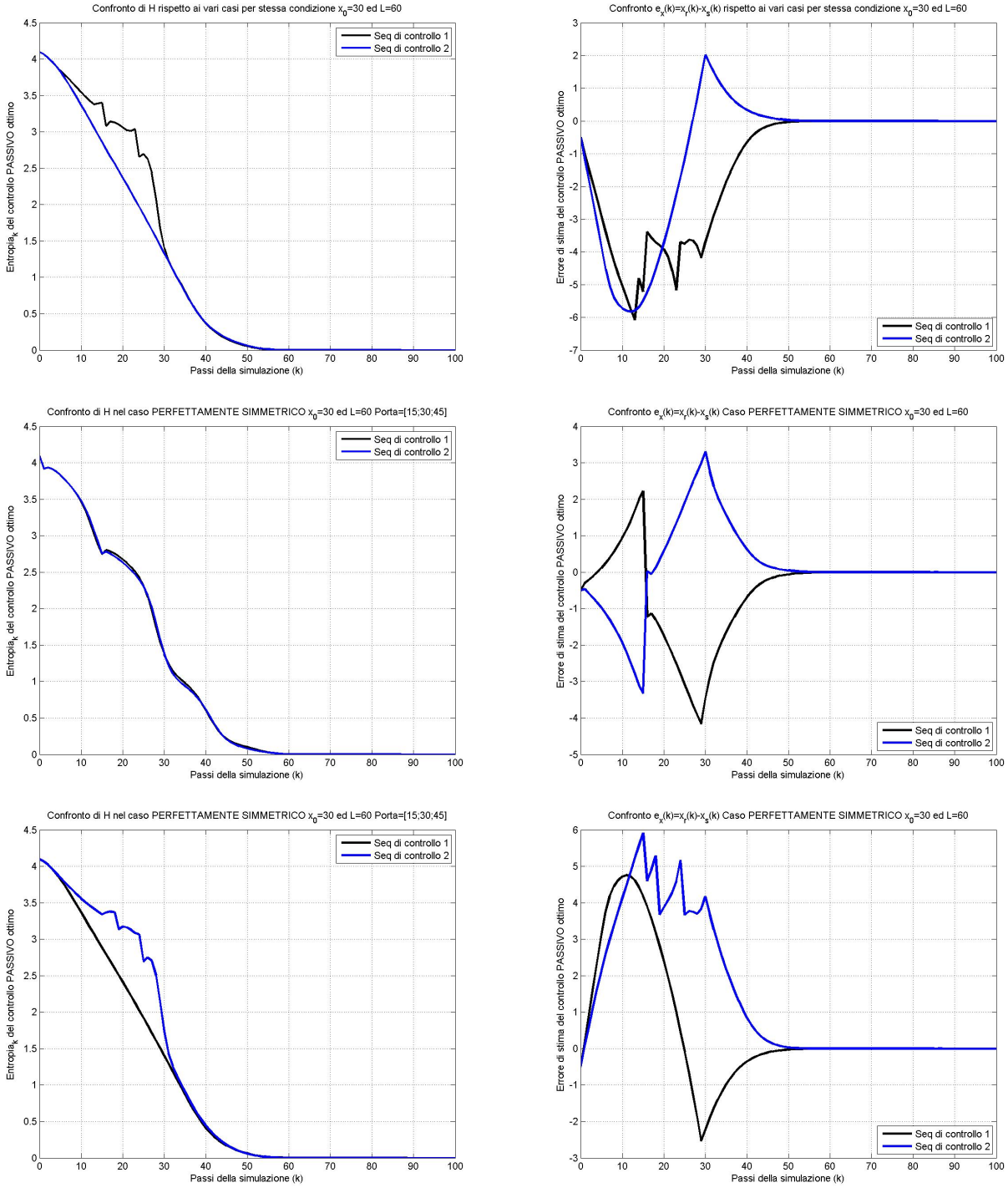

Figura 4.6: Grafico delle sequenze ottime per il controllo passivo a confronto.

4.1.2 Conclusioni dei risultati simulativi

Per mostrare la bontà dell'Active Localization basata sulla minimizzazione dell'Entropia si è deciso di portare tre particolari simulazioni.

Le simulazioni sono state effettuate con posizione reale $x_r(0)$ del robot scelta in maniera aleatoria tramite la funzione di matlab **ceil(L.*rand(1,1))** che permette di ottenere un numero casuale all'interno della mappa, ovvero fornisce un valore intero tra 1 e L.

Si tratta di tre simulazioni nelle quali si hanno 3 condizioni iniziali differenti e per le quali è applicato dapprima il controllo attivo e casuale, e, successivamente, un controllo passivo selezionato tra i migliori possibili per validare quello attivo.

Per tutte le situazioni sono stati graficati gli andamenti dell'Entropia associata ai diversi tipi di controllo implementati, quelli dell'errore di stima ad essi associato e quelli delle Belief calcolate per gli istanti significativi. Ricordiamo che la Belief per ogni passo k della simulazione è la probabilità del robot di percepire se stesso in posizione 1,2,3,...,L; si tratta dunque della probabilità di trovarsi in una certa posizione ad un certo istante. Quelli riportati di seguito sono i grafici essenziali per verificare se si sono raggiunti gli obiettivi dell'elaborato.

In conclusione qui si vuole effettuare un'analisi sul controllo attivo implementato; per giustificare l'uso di tale algoritmo ai fini della problematica relativa alla Localizzazione Globale esso deve risultare il migliore rispetto ai vari controlli considerati. Le simulazioni qui riportate servono a dimostrare che il controllo attivo permette di ottenere una soluzione più rapida al problema della localizzazione e, se consideriamo un controllo passivo più buono(tutti -1 o +1), l'algoritmo attivo per prestazione è migliore in quasi tutti i casi o al massimo uguale. Per induzione diventa migliore soltanto perché la scelta del movimeto è fatta in maniera automatica.

Entriamo nello specifico di ogni simulazione.

1. Simulazione A: controllo attivo scelta più efficiente (controllo tipo 3), controllo Passivo fatto off-line con $a_k = -1 \ \forall k \ (x_r(0) = 20)$.

2. Simulazione B: controllo attivo scelta più efficiente (controllo tipo 3), controllo Passivo fatto off-line con $a_k = +1 \ \forall k \ (x_r(0) = 5)$.

3. Simulazione C: controllo attivo scelta più efficiente (controllo tipo 3), controllo Passivo fatto off-line con $a_k = +1 \ \forall k \ (x_r(0) = 8)$.

Nella simulazione A si hanno i seguenti risultati:

- Il controllo attivo sceglie di effettuare tutte azioni $a_k = +1$ come si evince dall'andamento della funzione di utilità riportato nel riquadro in alto a destra dove si mostra la riduzione che si ha se si sceglie di volta in volta un'azione piuttosto che l'altra. In questo caso per comodità è stata portata anche la $U_{a=0}$ anche se poi questa opzione non viene mai scelta (fig. 4.7).

- Il controllo attivo permette di minimizzare l'Entropia associata alla posa in maniera molto più veloce dell'algoritmo passivo che presenta tutti -1. Dalla posizione reale di partenza il robot soggetto a controllo attivo si sposterà fino al muro destro del corridoio, mentre il robot soggetto a quello passivo, dalla stessa posizione si porterà al muro sinistro.

- La stima della posizione per il controllo attivo converge alla posizione reale in modo molto dolce e definitivo entro molto prima dei 20 secondi della simulazione, cosa che si riperte in modo simile ma con un andamento meno lineare e più lento circa a 20 secondi per il controllo passivo; la stima per il controllo casuale presenta sempre un errore di stima che tende a zero ma non ci va del tutto.

- Gli ultimi grafici mettono a confronto le Belief per i tre controlli per alcuni istanti della simulazione $k \in [0, 4, 8, 10, 50, 100]$; in questi si può vedere come il robot modifichi la sua Belief tramite l'algoritmo di aggiornamento utilizzando un estenzione del filtro di Bayes. Al primo istante tutte le Bel_0 sono uniformemente distribuite e, essento $L = 20$, la probabilità di trovarsi in una singola cella $x_i(0)$ vale proprio $Bel(x_i(0)) = \frac{1}{20} = 0.05$. Poi la probabilità si modifica a seguito delle azioni effettuate e delle letture sensoriali percepite dal robot; le Belief per i tre controlli vengono distribuite in modo differente. Quando una di esse presenta un valore prossimo all'unità tutte le altre Belief si annullano e il robot conosce con esattezza la sua posa.

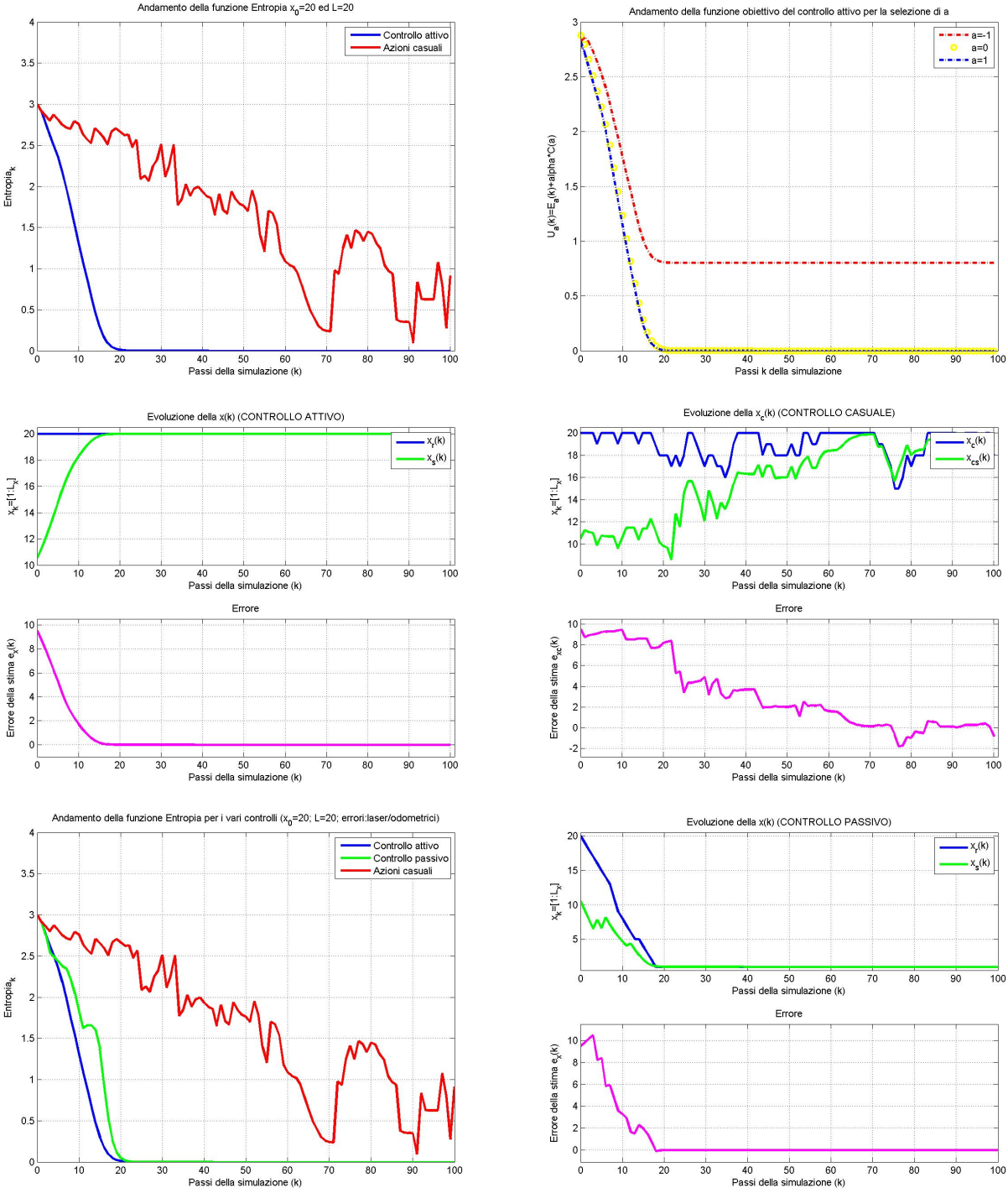

Figura 4.7: Esempio di una simulazione A(caso reale).

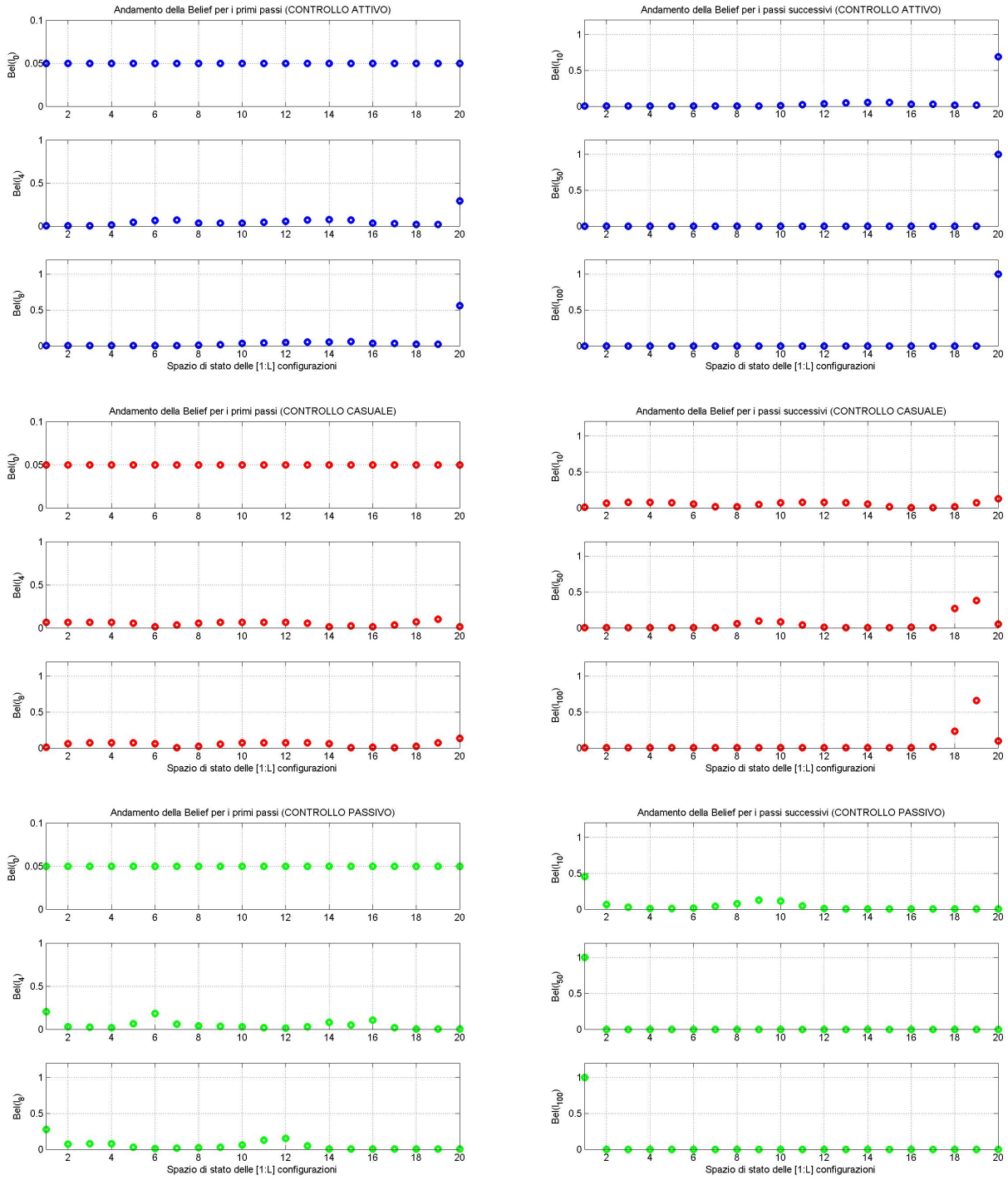

Figura 4.8: Belief a confronto per alcuni istanti k (Simulazione A).

Nella simulazione B si hanno i seguenti risultati:

- Il controllo ottimo sceglie di effettuare tutte azioni $a_k = -1$ come si evince dall'andamento della funzione di utilità riportato in questo caso, sempre nel riquadro in alto a destra; qui si vede chiaramente che la linea rossa è sempre al di sotto di quella blu (fig. 4.9).

- Il controllo ottimo permette di minimizzare l'Entropia associata alla posa in maniera molto più veloce dell'algoritmo passivo che presenta tutti +1. Si tratta di un caso complementare al precedente ed ancora una volta il controllo ottimo batte quello off-line scelto dall'operatore. Nella simulazione, invece, dalla posizione iniziale $x_r(0)$ il robot soggetto al controllo attivo si sposterà fino al muro sinistro del corridoio, mentre il robot soggetto a quello passivo dalla stessa posizione si porterà al muro destro.

- La stima della posizione per il controllo attivo converge alla posizione reale in modo molto dolce e definitivo ancora una volta più rapidamente di quella ottenuta tramite controllo passivo e questa presenta anche una piccola sovraelongazione; la stima per il controllo casuale presenta sempre un errore di stima che tende a zero ma non ci va del tutto.

- Gli ultimi grafici confrontano le Belief per i tre controlli per alcuni istanti della simulazione $k \in [0, 4, 8, 10, 50, 100]$ e servono ancora una volta per studiare l'aggiornamento della Belief.

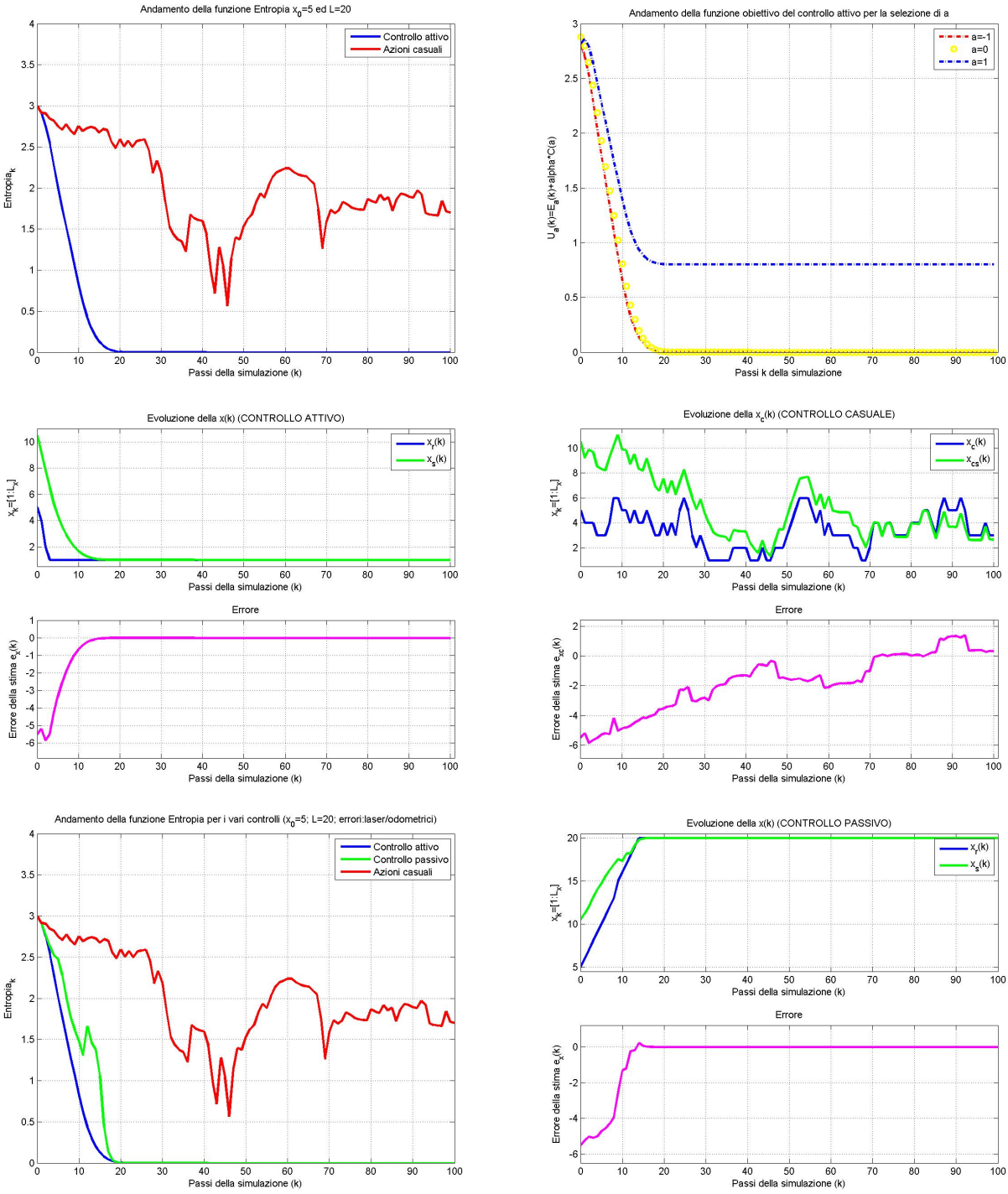

Figura 4.9: Esempio di una simulazione B (caso reale).

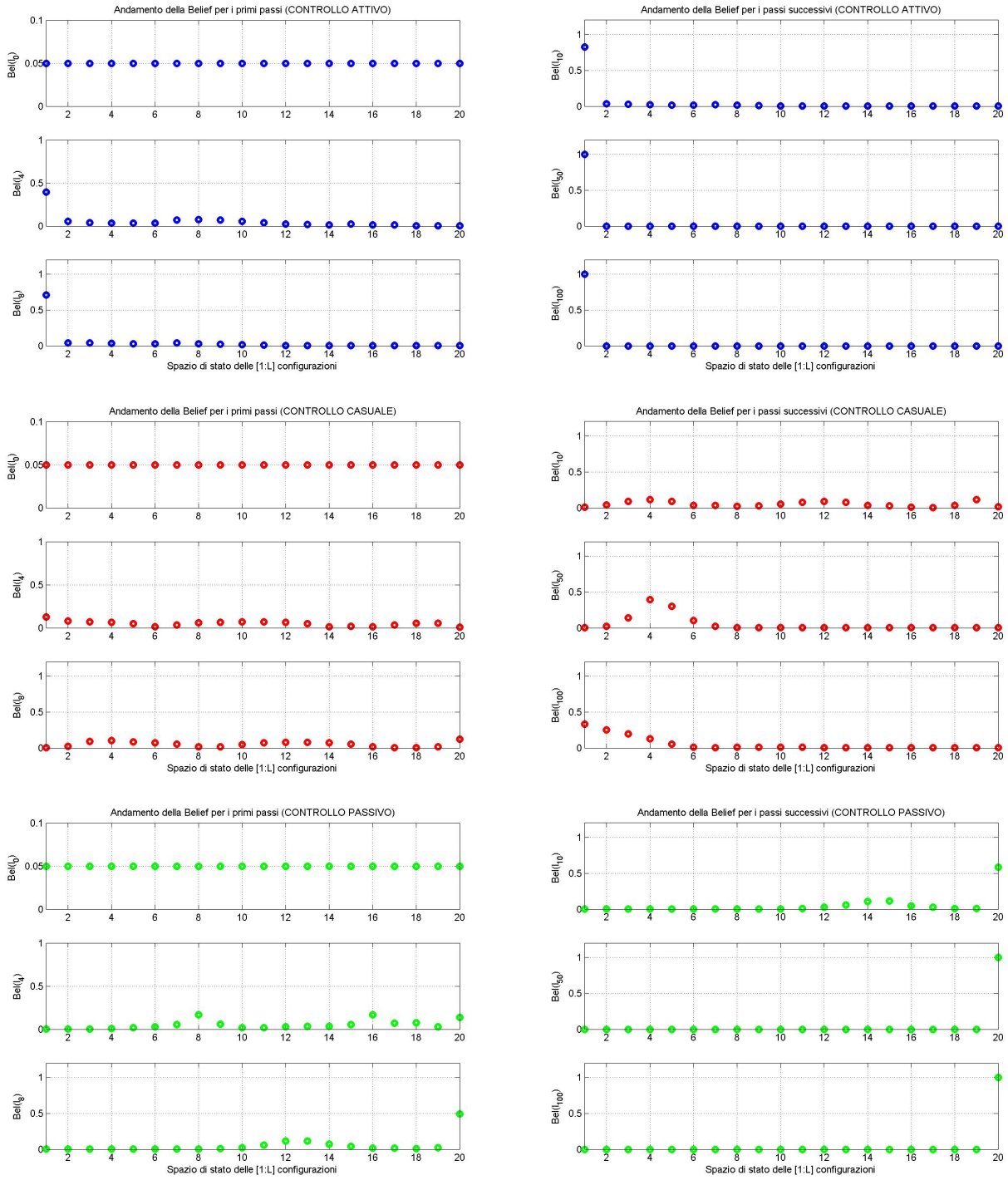

Figura 4.10: Belief a confronto per alcuni istanti k (Simulazione B).

Nella simulazione C si hanno i seguenti risultati:

- Il controllo ottimo sceglie di effettuare tutte azioni $a_k = +1$; tale scelta attiva risulta la stessa per il controllo passivo dove, anche per questo, si ha una lunga sequenza off-line di tutte $a_k = +1$.

- In questo caso si ha praticamente una coincidenza tra controllo ottimo attivo e controllo passivo; l'unica variante[4] è data all'istante $k = 0$ dove nel controllo passivo si applica subito l'azione di controllo $+1$, situazione che invece non si verifica per quello attivo, dove al primo passo dell'algoritmo si pone l'azione $a_0 = 0$; questa inizializzazione serve per il primo passo dell'iterazione dove il robot è fermo e subito dopo decide di fare la sua prima mossa.

- La stima della convergenza è istantanea in entrambi i casi anche se quella attiva sembra leggermente più pronta.

- Gli ultimi 2 grafici in basso alla figura (4.11) dimostrano che la Belief per controllo attivo e passivo per i primi istanti viene aggiornata dall'algoritmo in maniera pressochè uguale.

[4] È visibile nel piccolo shift in alto della funzione Entropia del controllo passivo rispetto a quella attiva.

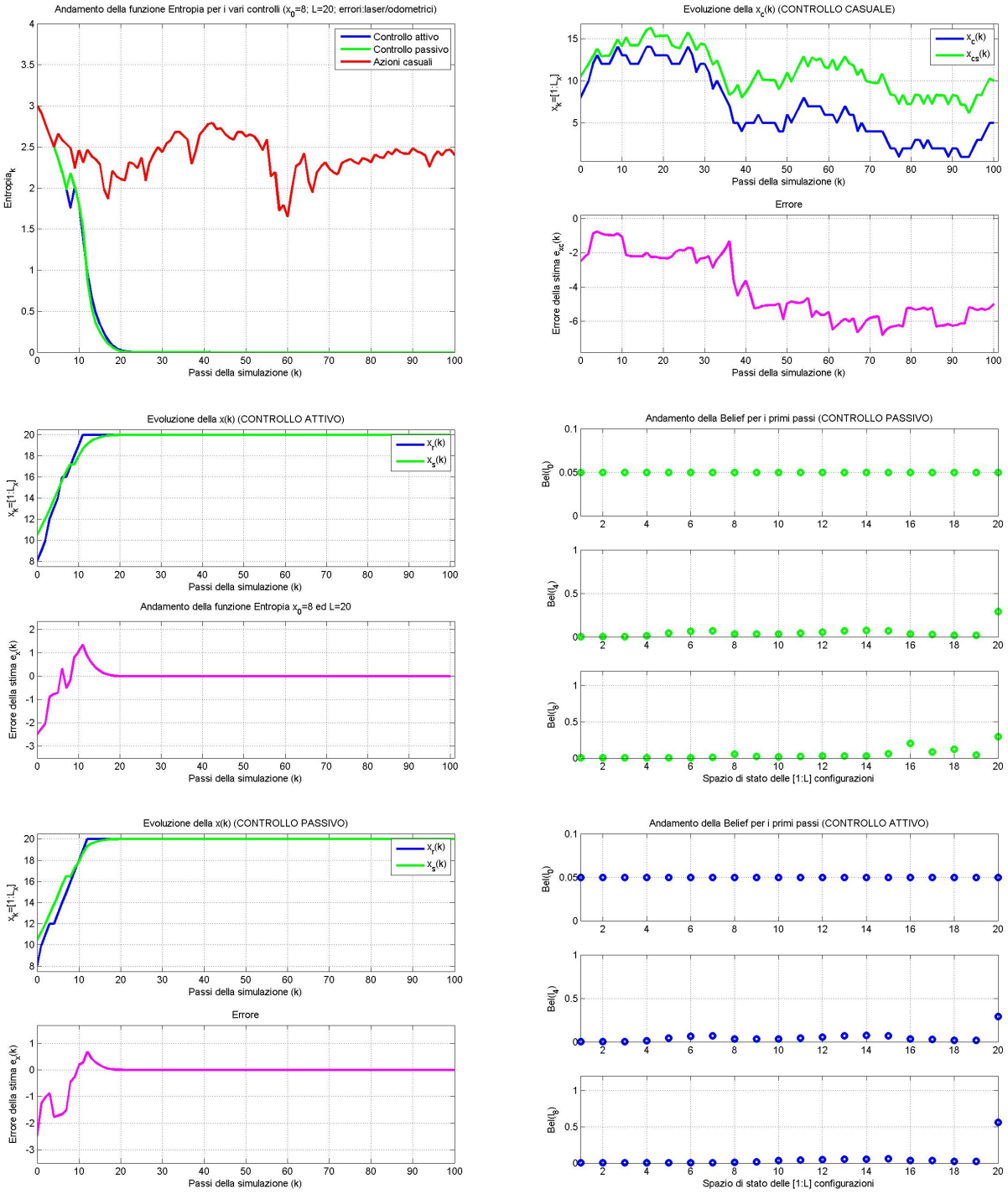

Figura 4.11: Esempio di una simulazione C (caso reale).

4.2 Caso Bi-dimensionale

4.2.1 Descrizione e discussione delle simulazioni

In questo paragrafo vengono riportati, come nel caso 1D, i risultati simulativi analizzati e commentati in modo dettagliato, ovvero, vengono mostrati e confrontati gli errori di stima, il tempo necessario per l'esecuzione del compito, e le simulazioni per diverse situazioni, condizioni iniziali, durata della simulzione, casi ideali, reali e per diversi ambienti simmetrici e asimmetrici. Tutto questo per permettere di trarre numerose conclusioni circa l'algoritmo utilizzato e circa la difficoltà di implemetazione e l'importanza di tale approccio probabilistico attivo rispetto agli altri due tipi di controllo analizzati, ovvero quello passivo e quello casuale.

In ogni circostanza, infatti, l'algoritmo attivo è risultato il più efficiente, affidabile e robusto ad errori di misura e/o a guasti dei sensori.

Il precedente paragrafo oltre a descrivere il caso Mono-dimensionale, ha svolto un ruolo anche introduttivo per diversi motivi, primo tra tutti la semplificazione del modello che permetteva di evidenziare le relazioni tra Belief e posizioni, calcolo delle stesse, e, in generale, l'approccio del controllo, e quindi il funzionamento base dell'algoritmo. Nel presente paragrafo si tralasciano le descrizioni introduttive e si parte già dai risultati ottenuti in precedenza. Inoltre, in questo caso, la mappa dell'ambiente nel quale il robot naviga è facilmente rappresentabile su un figura stilizzata a 2 dimensioni su piano (X,Y); si rimanda direttamente al grafico relativo. Quest'ultimo rappresenta la stanza, i cui limiti (le pareti) sono linee verdi dritte, e le cui porte sono piccoli cerchi di colore rosso (nelle figure 'AMBIENTE' dalla 4.15 in poi).

Per l'analisi del caso Bi-dimensionale si segue dunque la linea guida dell'1D, ovviamente riadattata alle esigenze del caso 2D.

Analisi dell'Entropia associata alla posa del robot:

In primo luogo, per vedere se non sia necessario effettuare un controllo particolare per permettere al robot di autolocalizzarsi, viene studiata l'Entropia associata alla posa senza applicare alcuna specifica politica di controllo. I risulti raggiunti e dedotti nel paragrafo relativo al caso Mono-dimensionale non devono essere dimenticati; si era visto che il problema di localizzazione si dimostrava molto difficile nel caso reale, ovvero, per tutti quei modelli che presentavano disturbi sia sulle letture sensoriali, sia sui dati odometrici; dunque in accordo con questo, tutta l'analisi verrà realizzata per il caso più significativo. Una volta trovata la soluzione al problema di localizzazione essa sarà valida anche per gli altri casi.

Dal grafico relativo all'andamento entropico della possa per diverse condizioni iniziali si evince un andamento molto irregolare con errori di stima molto elevati; essi sono lontani dall'origine per $e_y(k)$ di addirittura 20 unità. Va allora applicato il controllo, altrimenri il robot non si localizzerà mai in maniera esatta all'interno dell'ambiente di lavoro.

Una volta analizzata l'Entropia, l'attenzione è stata rivolta al controllo attivo volendosi dimostrare come questo possa permettere di ottenere un'ottima soluzione al problema della localizzazione globale. Se la sua bontà è assicurata per il caso più complicato, ovvero quello reale, come detto pocanzi, essa sarà sempre una valida soluzione per i problemi più semplici.

Per il caso Bi-dimensionale, come per quello Mono-dimensionale, sono state implementate delle varianti all'algoritmo di selezione. Il controllo è stato implementato secondo una prima versione base ($Ottima\_seq\_2D$) che prevedeva la scelta di tutte le azioni possibili del set a_seq; come nel caso 1D tale strategia di controllo non sembra

essere una stretegia ottima in quanto spesso l'algoritmo si pianta non permettendo al robot di autolocalizzarsi in maniera corretta. Si riporta qualche esempio nelle figure 4.20.

È stata così implementata nel file $Ottima\_seq\_senza\_a0\_2D$ una strategia di controllo che elimina tra le possibili scelte di movimento la possibilità di stare fermo; questa dà ottimi risultati, visibili dai grafici, nella maggior parte delle simulazioni. In rarissimi casi, ovvero quando l'ambiente presenta forti simmetrie o per qualche combinazione posa iniziale, ambiente e distribuzione landmark, è capitato che l'algoritmo di selezione si sia piantato scegliendo sempre la stessa azione di controllo; tale situazione di stallo è da paragonare alla stessa che si verifica nel caso Mono-dimensionale, cioè situazione di minimo relativo dell'Entropia. Per aggirare questa situazione, benchè rara, si è provata una tecnica di controllo basata sull'Active Navigation, ovvero la selezione di un insieme di azioni elementari, non più in modo *greedy* ma a *target point*, eliminando i limiti della applicazione greedy[5] e la possibilità quindi di restare fermo. Questa però risulta migliore della precedente solo in quei casi particolari di stallo, dove le azioni sono effettuate in maniera tale che il robot non rimanga più di tanto fermo in una posizione. Risulta una buona strategia.

Vengono riportate le seguenti simulazioni:

1. 2 Simulazioni 'intere' A/B (da fig. 4.15 a 4.18): si confrontano i vari tipi di controllo e l'andamento delle stime; dalla simulazione si vede il buon funzionamento del controllo attivo rispetto a scelte passive.

2. Simulazione C: altro caso di buon funzionamento.

3. Simulazione D, 4 esempi di controllo applicati: 2 greedy; 2 target point.

[5]Come visto nel caso 1D, il robot durante la localizzazione potrebbe occupare posizioni precedenti.

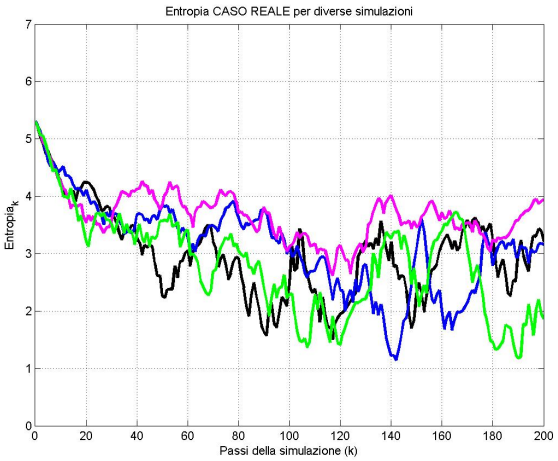

Figura 4.12: Funzione Entropia per robot libero di muoversi casualmente.

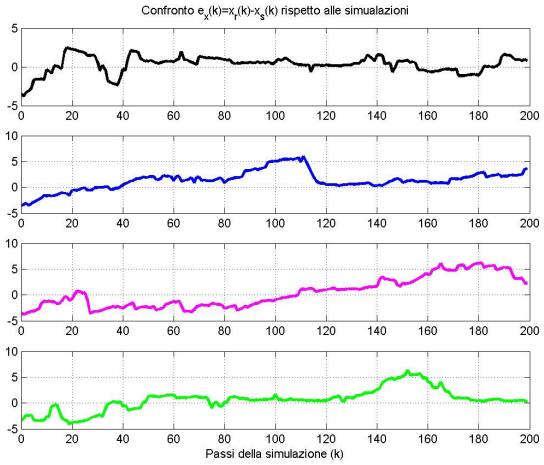

Figura 4.13: Errori relativi alla stima della posa: evoluzione x(k) (CASO REALE).

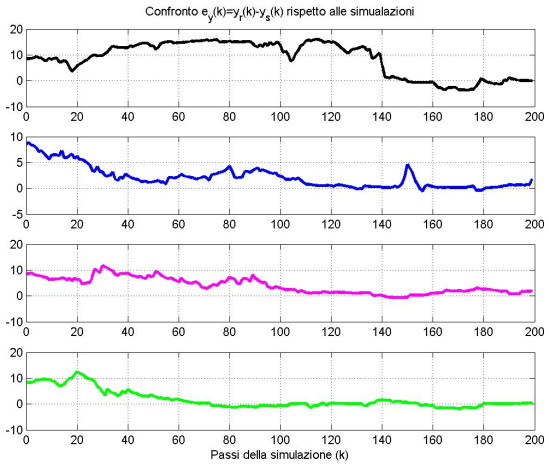

Figura 4.14: Errori relativi alla stima della posa: evoluzione y(k) (CASO REALE).

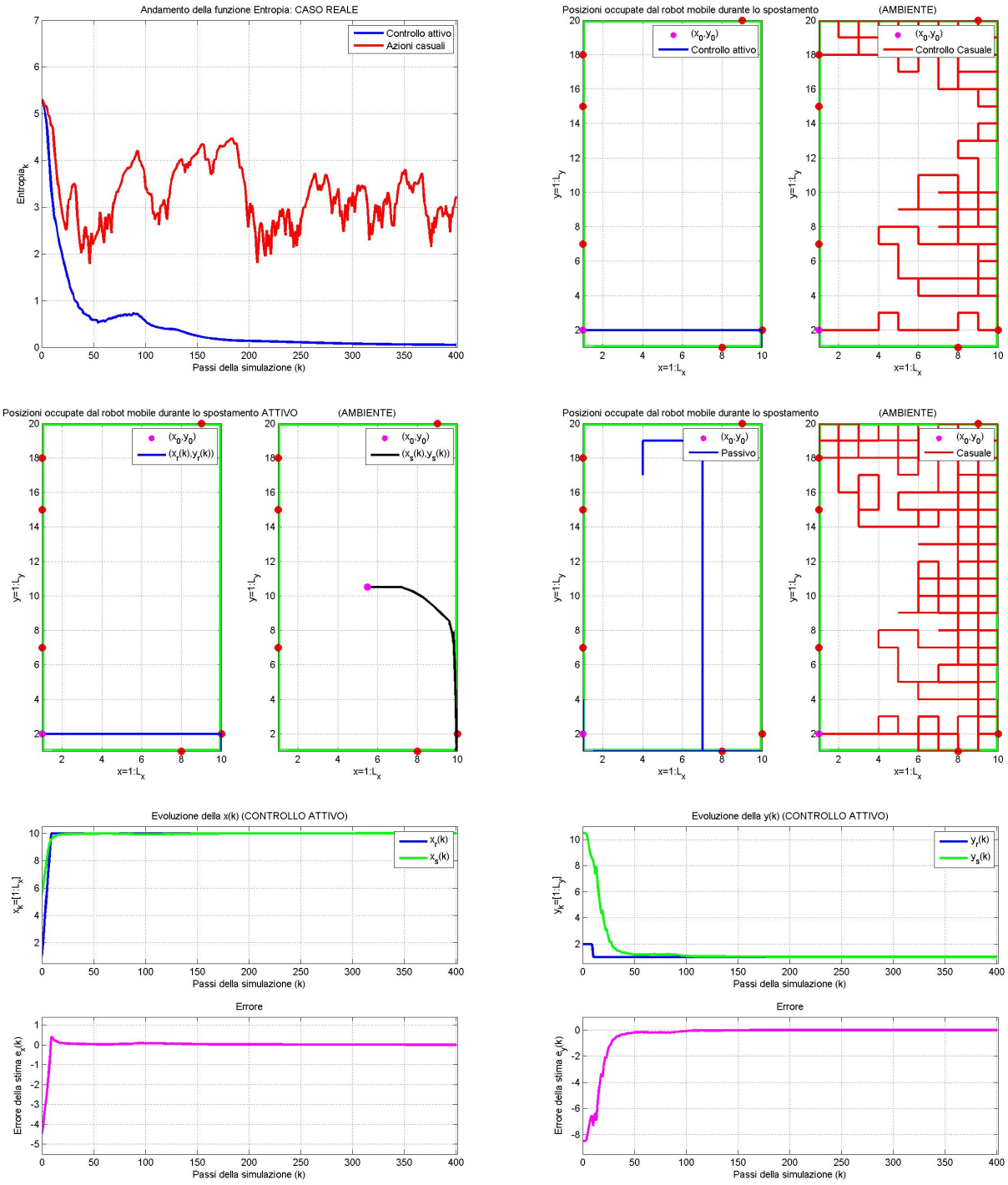

Figura 4.15: Esempio di simulazione A, prima parte (CASO REALE).

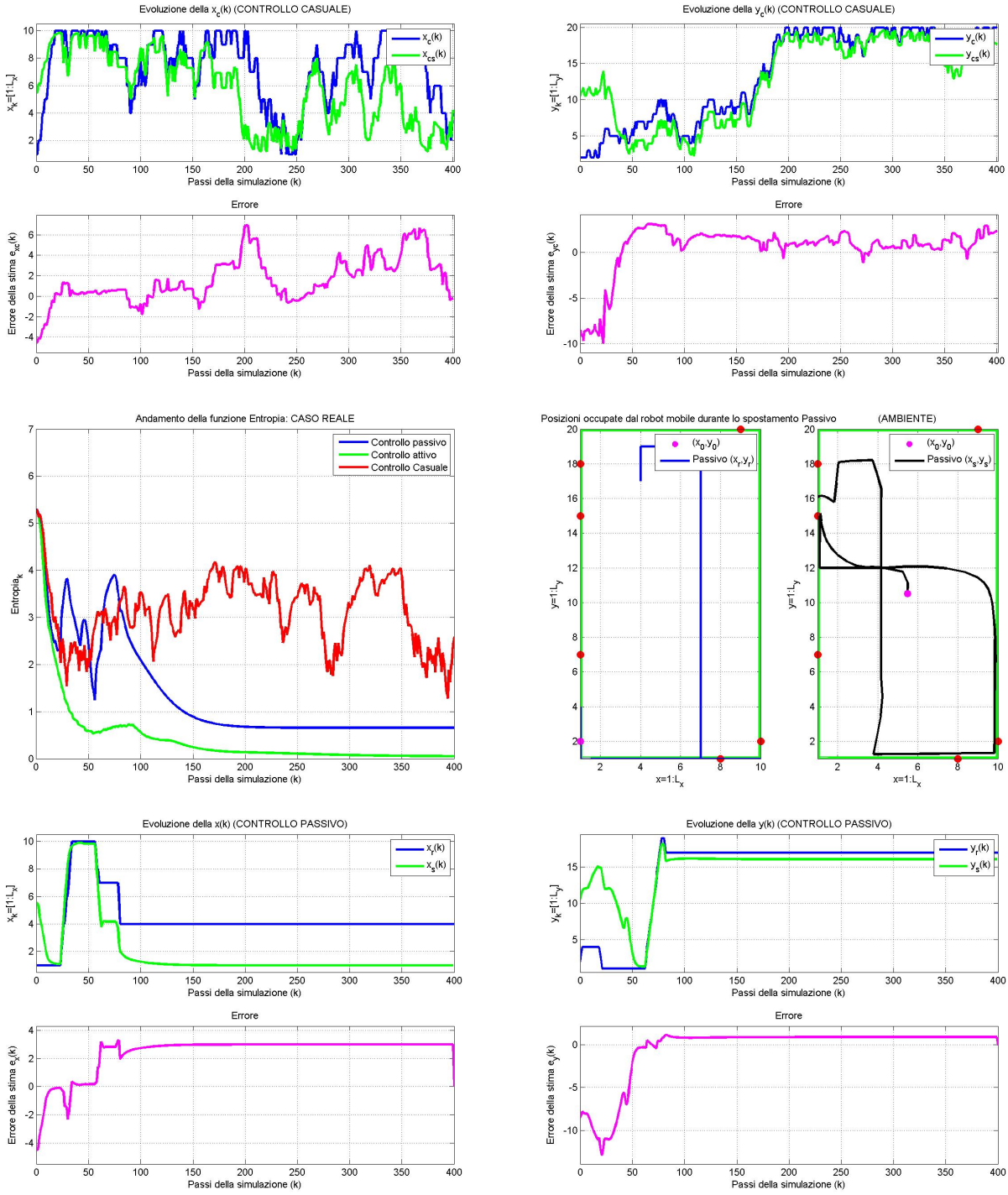

Figura 4.16: Esempio di simulazione A, seconda parte (CASO REALE).

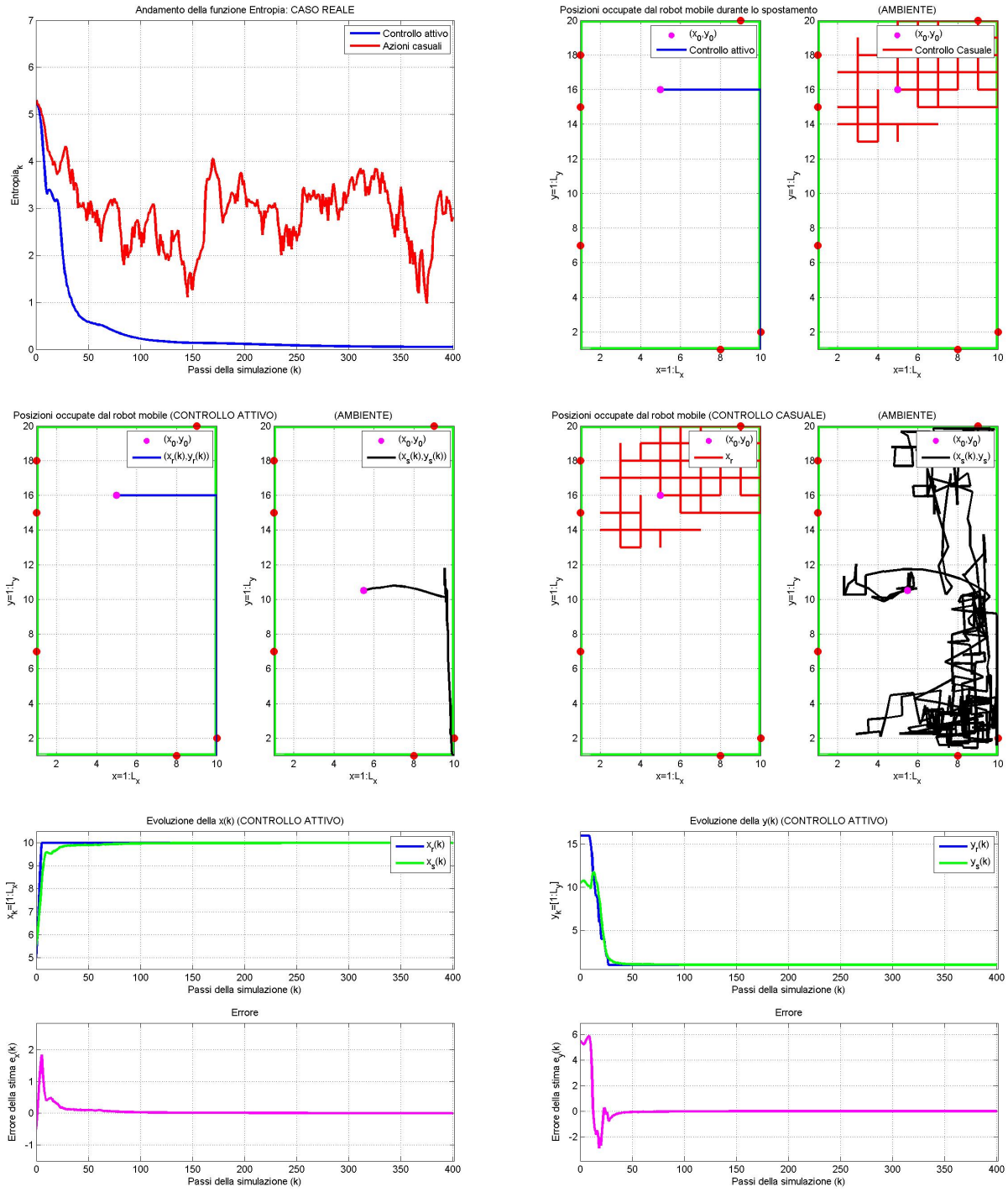

Figura 4.17: Esempio di simulazione B, prima parte (CASO REALE).

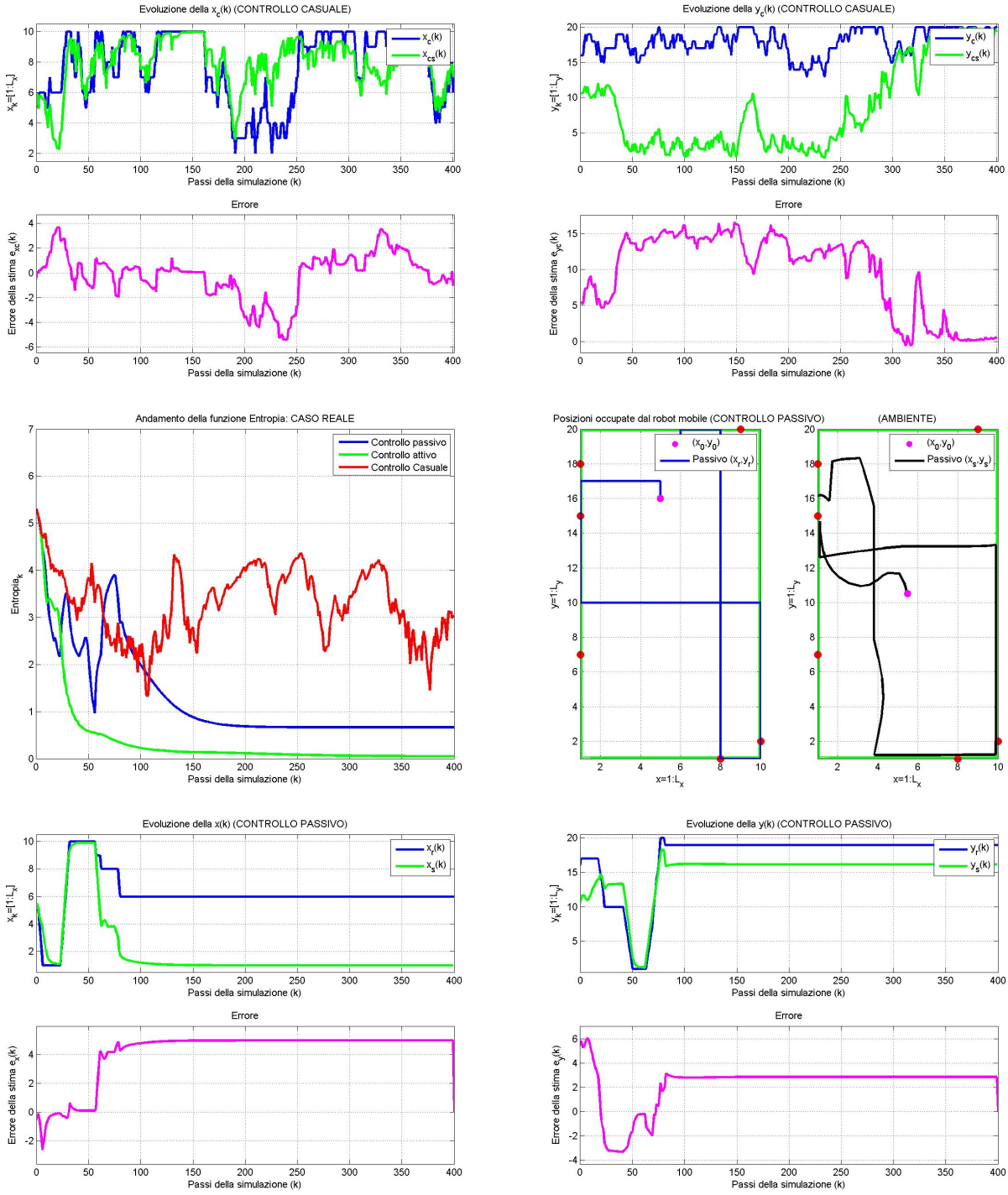

Figura 4.18: Esempio di simulazione B, seconda parte (CASO REALE).

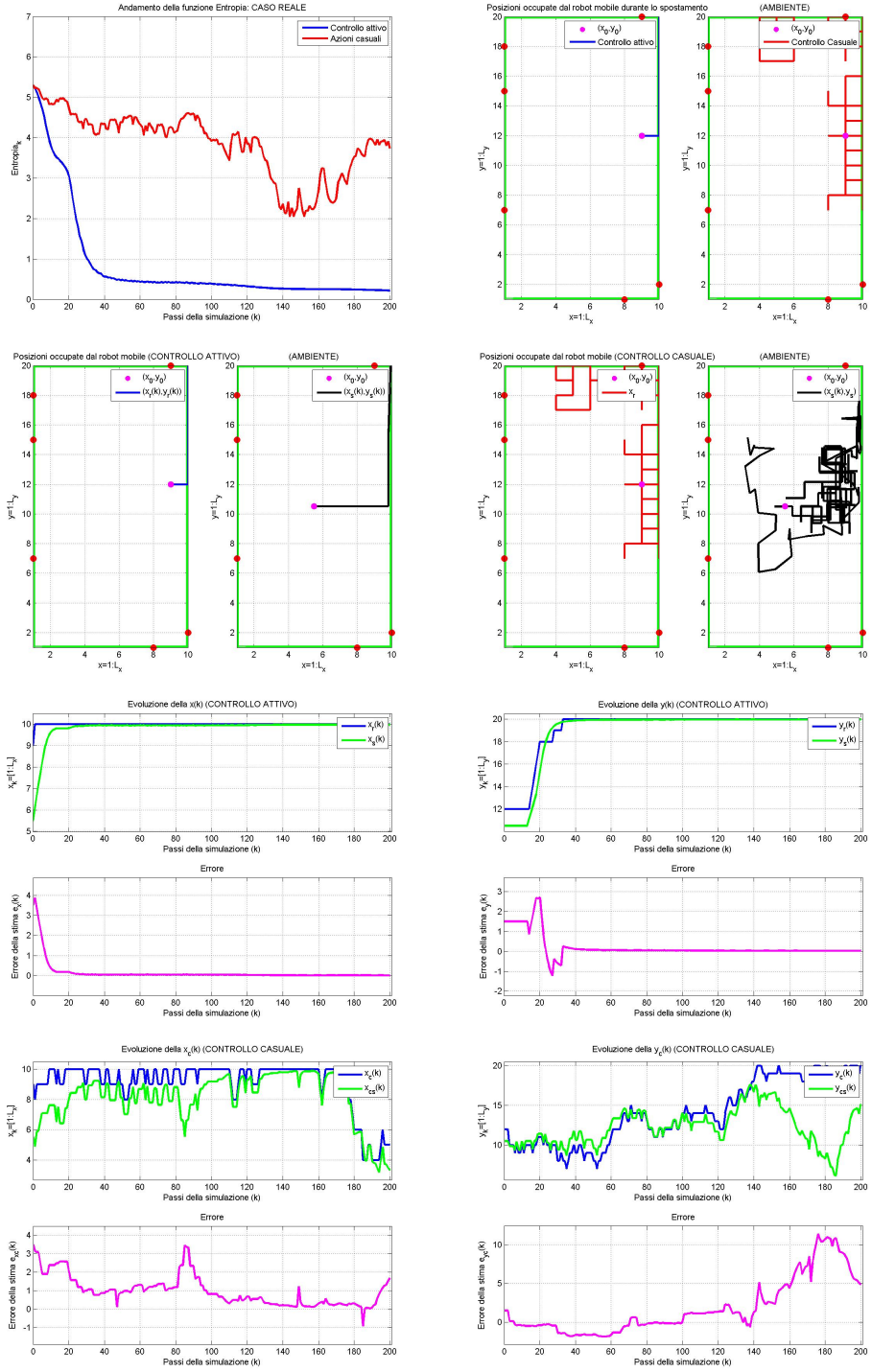

Figura 4.19: Simulazione C (altro caso di buon funzionamento).

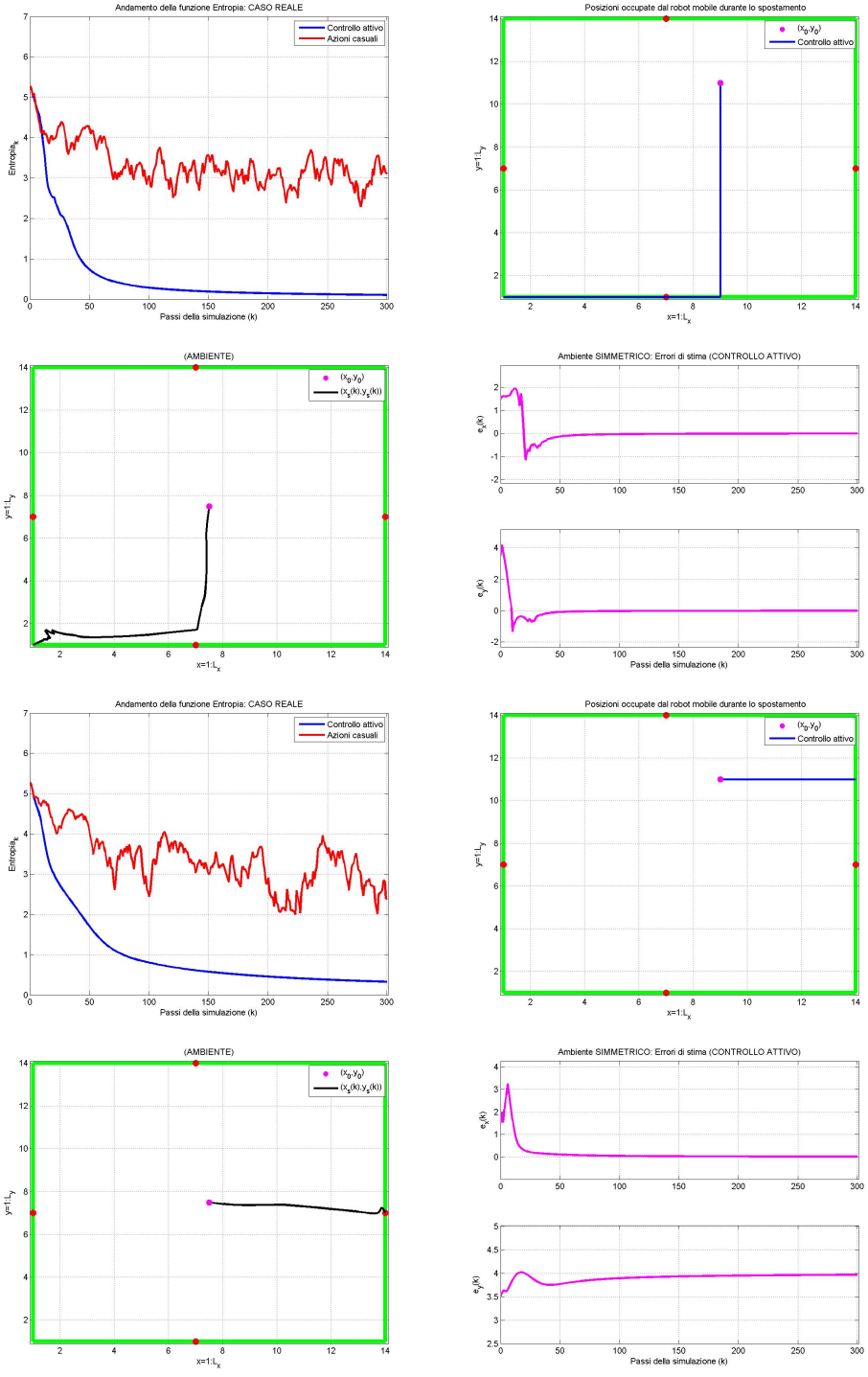

Figura 4.20: Simulazione D (Stallo).

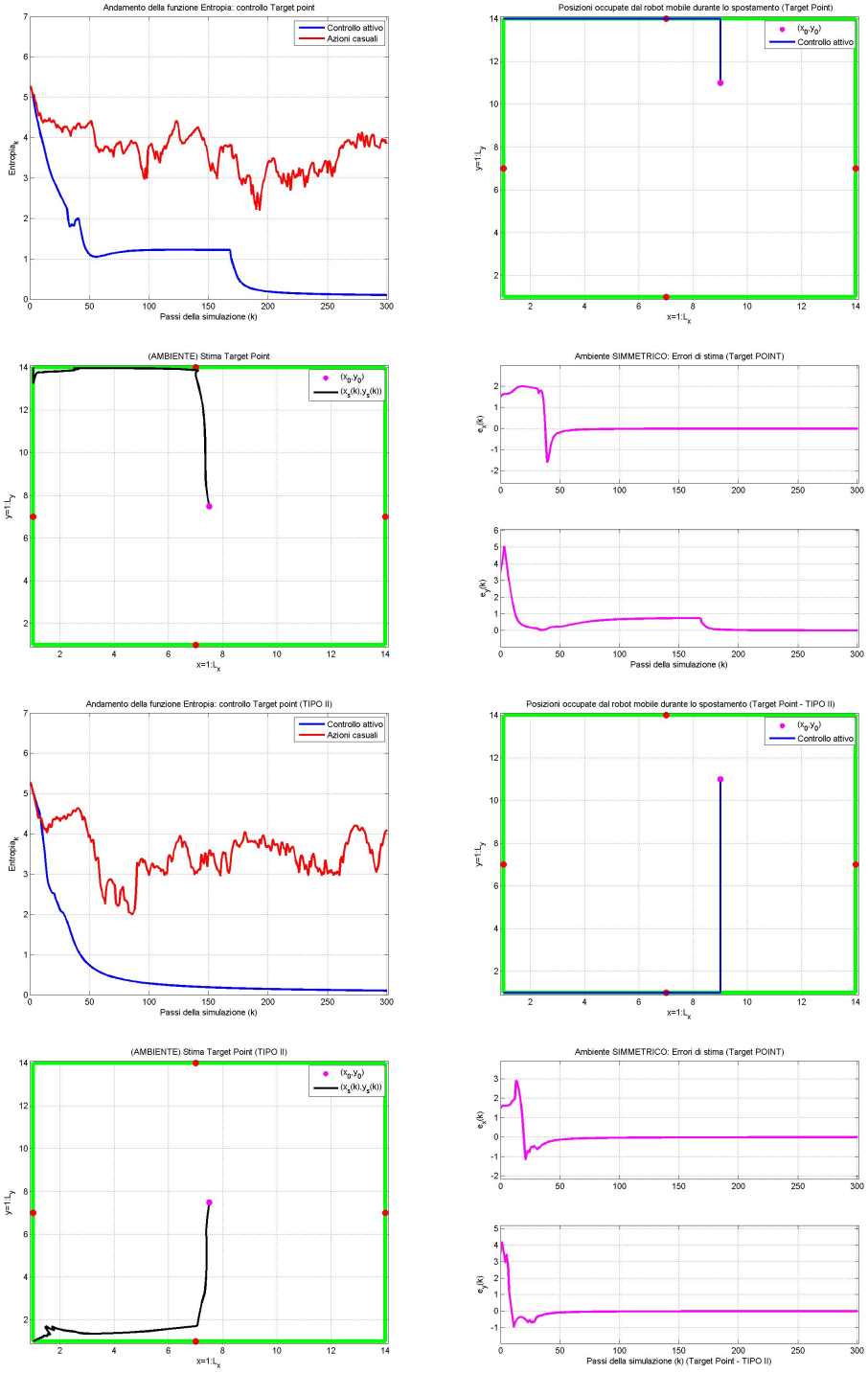

Figura 4.21: Simulazione D bis (Soluzione Target Point).

Confronto tra controllo Attivo e Passivo:

Il controllo passivo, come abbiamo già visto nel paragrafo relativo al caso Mono-dimensionale, dipende fortemente dalla scelta fatta off-line. L'operatore manualmente decide fuori linea la particolare sequenza che deve far effettuere al robot per permettergli di localizzarsi; per trovare le azioni ottime deve essere analizzata molto bene la mappa topologica dell'ambiente in cui l'automa sarà libero di circolare perché il robot, per meglio localizzarsi, deve andare in quella zona con più informazioni (regione con maggior presenza delle features-porta). Quindi, un certo set di azioni selezionate per ottimizzare la localizzazione per un certo ambiente, potrebbe risultare addirittura inefficiente per un'altro di differente grandezza, o con la medesima ma con distribuzione diversa dei landmarks. Questi, infatti, influiscono molto per la localizzazione. Ad esempio, per l'ambiente considerato precedentemente, sono state effettuate tante simulazioni per diversi tipi di controllo passivo; di queste, le sequenze di azioni che permettevano una migliore soluzione del problema erano quelle che spingevano il robot verso le pareti, e là dove c'è una maggiore concentrazione dei landmarks; se l'automa passa vicino ad un numero più elevato di porte, questo riesce prima ad autolocalizzarsi all'interno dell'ambiente. Nelle figure riportate, vediamo come le sequenze di controllo 'Seq 2' e 'Seq 3', in questo caso siano le migliori.

In particolare la prima delle due, nella realtà $(x_r(k), y_r(k))$, muove il robot verso il muro con più porte passando prima vicino ad una di queste e poi davanti alle altre due proseguendo lungo lo stesso lato. A tale sequenza di azioni è associato un errore di stima della posizione quasi nullo, infatti, alla fine della simulazione, il robot conosce quasi perfettamente la sua posizione. Ad essa è associato un errore di stima a regime[6] nullo per la posizione x, mentre pari a circa 0.8 per la posizione y. Questa piccola

[6]Ci interessa a regime per sapere se il robot si è autolocalizzato alla fine della simulazione.

imprecisione era prevedibile anche dal grafico nel riquadro più in alto a sinistra nel quale l'andamento entropico associato alla posa non si annulla; si sarebbe annullato solo se il robot avesse avuto la certezza assoluta, probabilità pari ad 1, di occupare un'unica posa, quindi quando l'Entropia vale $H = 1 * log(1) = 0$. Il fatto che il robot riesca comunque a stimare una posa molto prossima a quella reale implica comunque che l'aggiornamento delle Belief sia tale da concentrare la distribuzione di probabilità di occupare una certa posa proprio nei pressi della posa effettiva; dunque, si tratta di una buona approssimazione della posa.

Nel caso di 'Seq 5', il robot effettua una serie di azioni che lo portano a visitare ben 3 porte, per tale sequenza di controllo gli errori associati alla stima, e anche l'Entropia, si annullano; l'automa ottiene una coincidenza tra posa effettiva e quella percepita e quest'ultima conta di una probabilità Belief pari ad 1. In questa simulazione, inoltre, gli andamenti della stima presentano degli spikes dovuti ad inversioni di orientamento che il robot percepisce in modo errato, ma questi si manifestano solo nel transitorio, mentre successivamente a regime si stabilizzano.

Riguardo alle altre sequenze di controllo è interessante notare che le 'Seq 1 ' e 'Seq 4' presentano caratteristiche simili. Il robot segue un set di azioni che lo portano prima vicino a due landmarks poi lo costringono ad allontanarsi; il robot si avvicina alla parete con le porte, riesce quasi ad autolocalizzarsi con una buona stima, quando cambia rotta e si allontana da queste perdendo informazioni. Dall'andamento sulle stime, gli errori e_x e e_y tendono ad annullarsi all'inzio, per poi allontanarsi dall'origine; a questo comportamento è associata la seguente situazione: il robot prima percepisce con buona approssimazione la traccia reale della propria traiettoria in prossimità delle features, poi la perde mentre se ne allontana.

Infine per l'ultima sequenza 'Seq 3' il robot si avvicina e si allontana per ben tre volte

da tre porte differenti, e i grafici dell'errore di stima che ne conseguono hanno un andamento che cerca di annullarsi per poi ricrescere; a regime non si raggiunge una buona stima sulla posizione nè si ottiene l'annullamento dell'Entropia.

Applichiamo ora lo stesso controllo passivo ritenuto il migliore, ovvero la sequenza di controllo 'Seq 5' per un'altra tipologia di mappa che cambia per la sola distribuzione dei landmarks (non per dimensione) e per una diversa condizione iniziale. È facile vedere come tale controllo passivo non risulti poi così soddisfacende. Dalla simulazione in figura 4.22 notiamo che il controllo off-line effettua una stima approssimativa della posa finale; il controllo attivo ne permette una stima perfetta.
Si può notare inoltre che l'Active Localization non sfrutta solo il fatto di andare verso il landmark per autolocalizzarsi, politica di controllo spesso associata ad una diminuzione dell'incertezza, ma, a seconda della condizione iniziale e della distribuzione dei landmark, potrebbe scegliere come azione ottima una che sia opposta alla precedente, ovvero di allontanamento dai landmarks, alla quale si attribuisce, anche per questo caso, una diminuzione dell'Entropia. Al robot potrebbe, infatti, risultare che la mossa greedy migliore da fare sia di allontanarsi dalla zona-features e, tenendo conto che non si percepisce l'esistenza di alcuna porta, esclude dalla probabilità Belief la probabilitià di troversi vicino ad una di esse, e quindi, dopo aver navigato per un pò, riesce ad autolocalizzarsi nel modo opportuno.

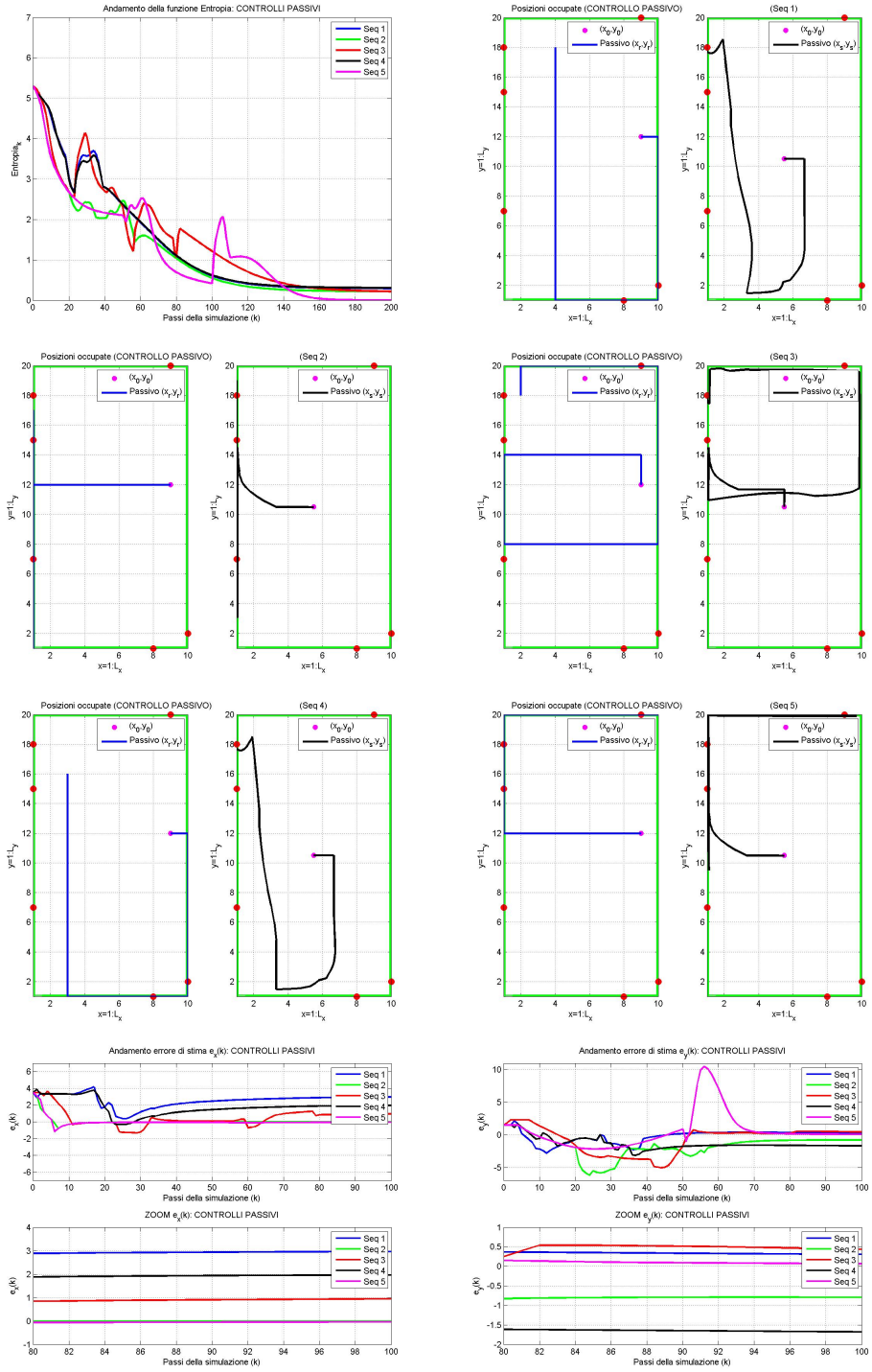

Figura 4.22: Studio del controllo passivo: scelta della sequenza ottima 'Seq 5'.

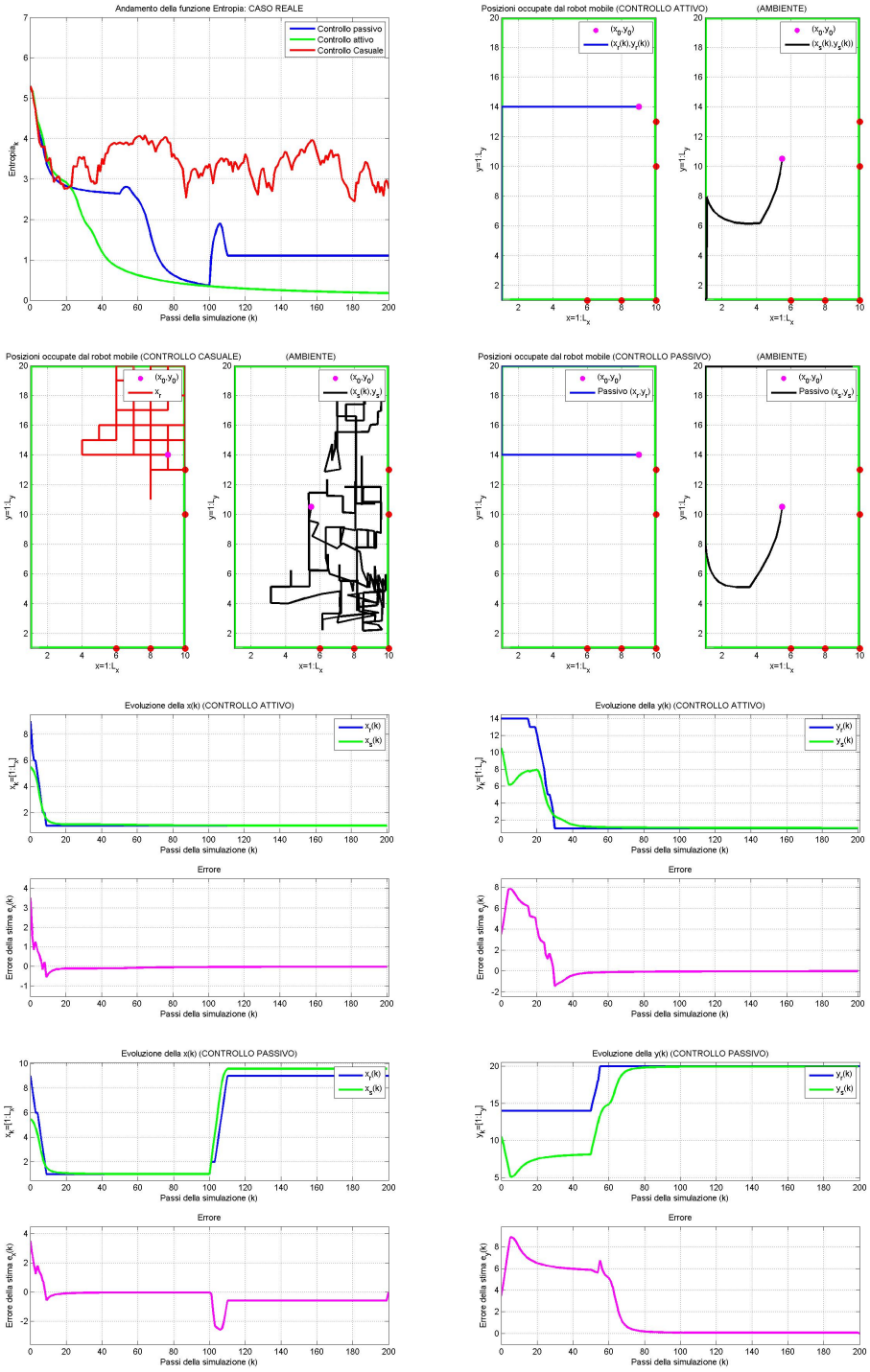

Figura 4.23: Validità della 'Seq 5'(controllo passivo): diverso ambiente e condizione iniziale.

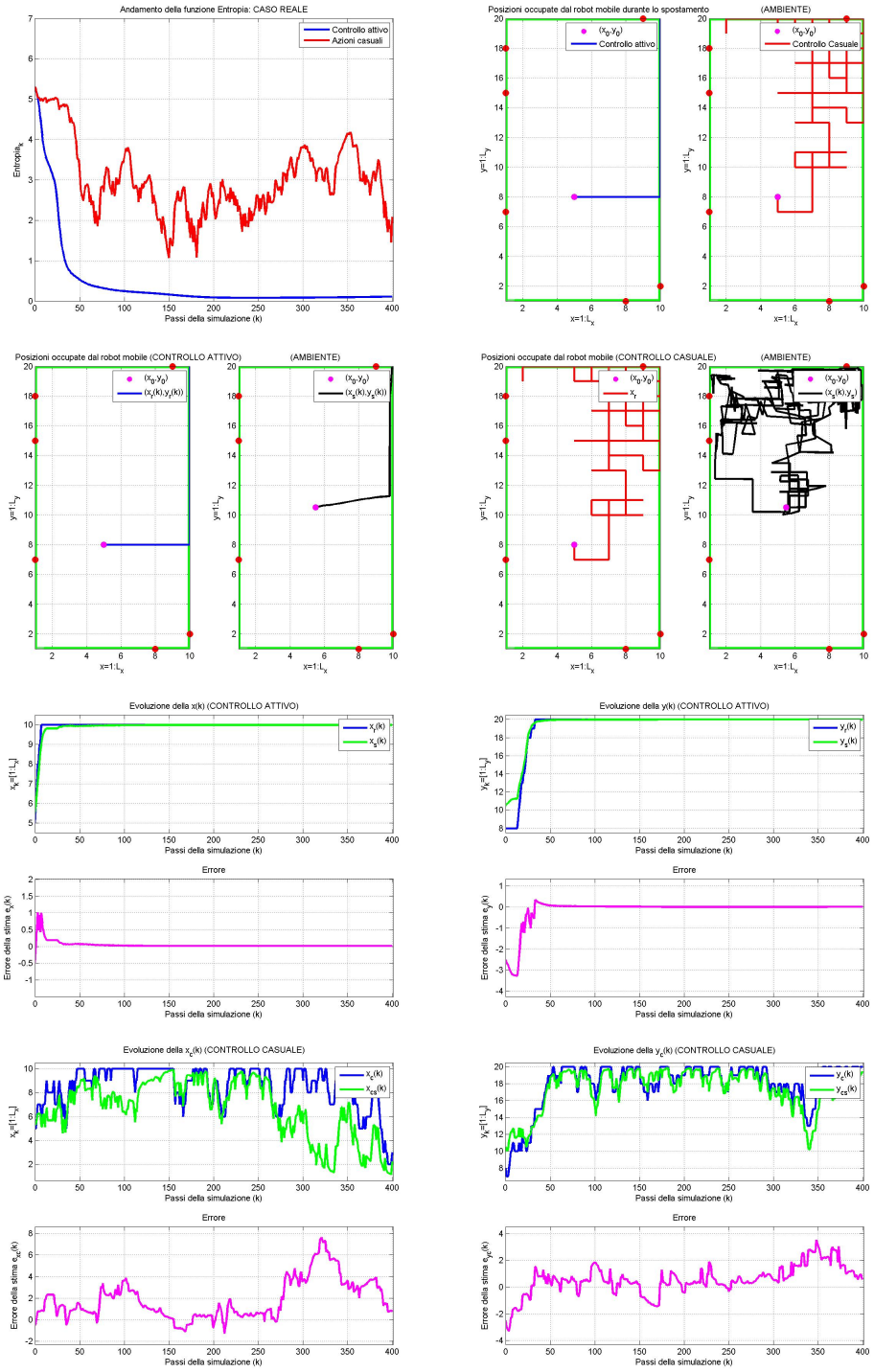

Figura 4.24: Validità del controllo attivo: stesso ambiente; diversa condizione iniziale (I parte).

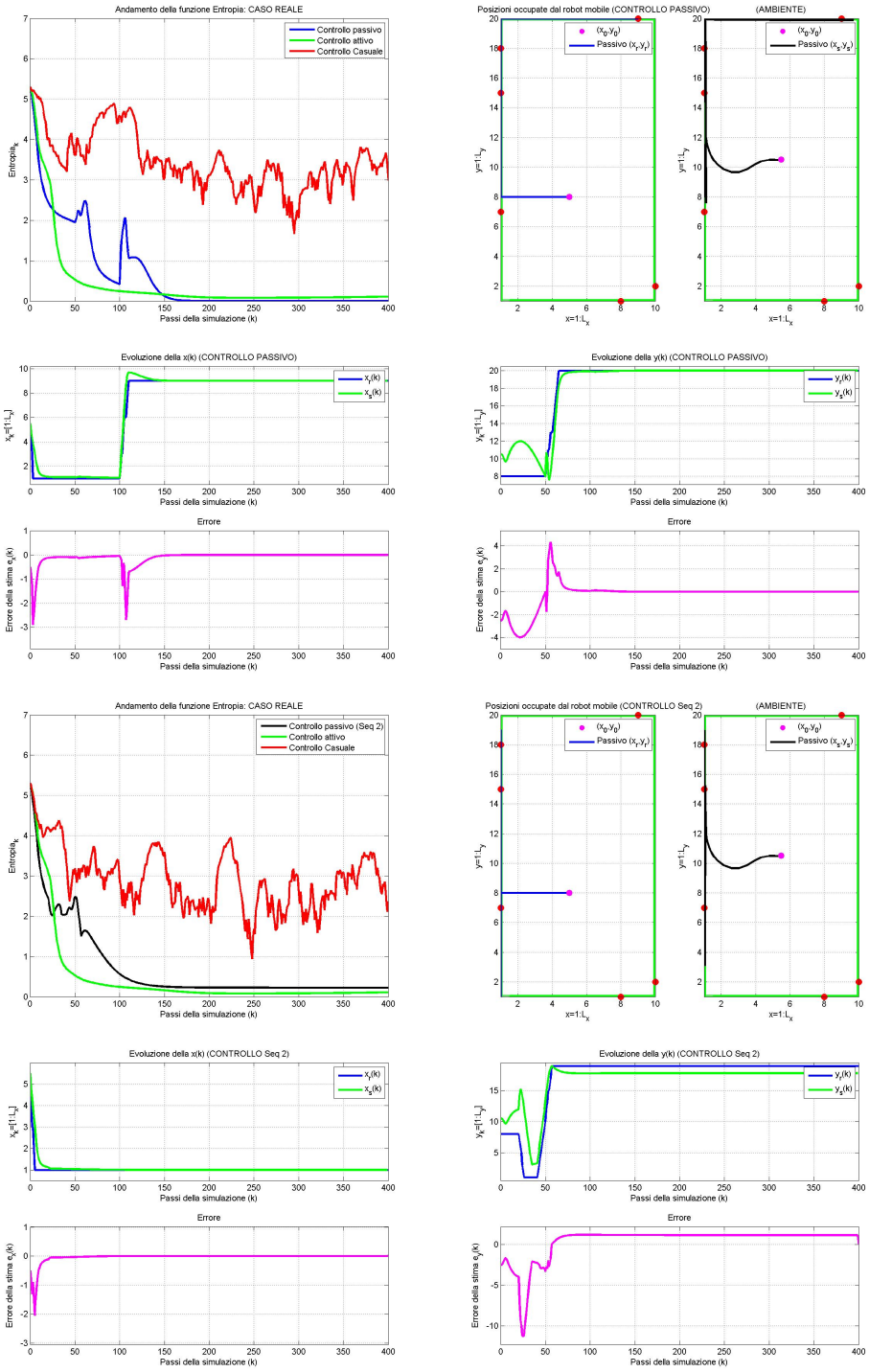

Figura 4.25: Validità delle 'Seq 5' e 'Seq 2' (controllo passivo): stesso ambiente; diversa condizione iniziale (II parte).

4.2.2 Conclusioni dei risultati simulativi

In conclusione dai risultati simulativi si evince che l'algoritmo della localizzazione attiva basato sulla minimizzazione dell'Entropia è molto efficiente; tale controllo, se fatto su un caso a due dimensioni, nonstante sia molto più complesso del caso Mono-dimensionale da un punto di vista computazionale, risulta molto buono e presenta situazioni di stallo dell'algoritmo in casi rarissimi. La selezione migliore dipende sia dall'ambiente particolare nel quale viene a trovarsi il robot, sia dalla distribuzione dei landmarks. Questi sono molto utili al fine della localizzazione.
L'Active Localization fallisce qualche volta nei casi di ambiente simmetrico, ma è possibile aggirare il problema utilizzando una selezione attiva su una sequenza di azioni elementari che definiscono il controllo a target point: ovvero a= 'spostati di..'.

Per quanto riguarda la validità del controllo on-line rispetto a ipotetici controlli effettuati off-line, esso è risultato un approccio nettamente superiore; questo risolve infatti in maniera efficiente il problema della localizzazione globale per qualsiasi tipo di ambiente considerato, di posa iniziale e distibuzione dei landamarks, adattandosi in manera automatica ad ogni cambiamento. Al contrario quello passivo deve essere studiato in modo approfondito per un certo ambiente e spesso non è adattabile ad altri tipi di ambiente per i quali non era stato progettato. Quest'ultimo, inoltre, non risulta neanche così buono per ogni condizione iniziale nell'ambiente di partenza. Questo discorso non è valido per il caso Mono-dimensionale, dove le uniche strategie migliori provate risultano andare o sempre a destra o sempre a sinistra e la preferibile tra le due è determinata dalla mappa dell'ambiente[7]. Nel caso 2D, c'è più libertà di

[7]La validità è data dal fatto che le uniche due azioni che il robot può compiere sono 'avanti' o 'indietro'.

movimento e quindi possiamo avere tante combinazioni di controlli off-line che possono dare buoni risultati, da questi però non si riesce a ricavarne uno che sia ottimo per ogni condizione iniziale e mappa metrica dell'ambiente in generale, pertanto per ogni situazione bisognerebbe calcolare quello ottimo per il caso particolare. Preso uno 'buono' tra questi, i risultati simulativi che ne derivano mettono in luce che in alcuni casi esso risulta buono almeno quanto il controllo attivo implementato, ma in altri nettamente inferiore con errori sulla posizione che superano anche le 2 unità.

Capitolo 5

Conclusioni

L'obiettivo principale del lavoro svolto è stato quello di affrontare i problemi relativi alla localizzazione di un robot mobile attraverso l'applicazione di un approccio attivo alla localizzazione.

Abbiamo visto come nella localizzazione attiva il robot, durante l'esplorazione, controlli i suoi attuatori per meglio localizzare se stesso all'interno dell'ambiente. Alla base delle tecniche utilizzate vi è la localizzazione Markoviana che è un tipo di approccio probabilistico passivo alla localizzazione e permette al robot di aggiornare la stima di occupare una certa posa, ovvero la distribuzione Belief; questo però non provvede a dare una risposta su come controllare gli attuatori.

Lo scopo principale dell'approccio attivo studiato in questa tesi controlla on-line gli attuatori per meglio localizzare il robot, sceglie le azioni in modo intelligente per permettere di stimare la sua posa il prima possibile e in modo corretto. Si avvale di una politica di controllo che muove il robot in modo da minimizzare l'incertezza futura sulla sua posa che è misurata dall'Entropia, dunque decide le azioni cercando di minimizzare quest'ultima. Questo principio di funzionamento è stato applicato al problema attivo della localizzazione relativo al *Active Navigation*, quel problema che risolve la scelta dell'azione futura da compiere ad ogni spostameto. L'algoritmo

implementato determina il percorso ottimo per un robot soggetto a incertezze sulla posizione e lo fa utilizzando una versione modificata della programmazione dinamica. Tale tecnica della localizzazione attiva tramite minimizzazione dell'Entropia si è dimostrata una soluzione valida per risolvere la localizzazione di un robot in modo molto efficiente e in particolare in ambienti con pochi landmarks. Questo controllo ha permesso, molte volte, di raggiungere la localizzazione per alcuni ambienti per i quali l'approccio passivo falliva; è risultato, inoltre, molto efficiente anche utilizzando sistemi di rilevamento sensoriale rumorosi o con disturbi sul movimento, offrendo una buona reiezione a questi e portando ad una sostanziale coincidenza tra lo stato stimato e quello effettivo.

Questa tecnica si è però dimostrata gravosa da un punto di vista computazionale per ambienti indoor di grandi dimensioni, il limite deriva dalla complessità di calcolo dell'algoritmo di controllo nel calcolare l'Entropia futura.

Un'altro limite proviene dal modo greedy con cui si selezionano le azioni: potrebbe capitare infatti che il robot torni in una posizione precedentemente occupata non risolvendo la localizzazione; questa situazione è visibile solo in pochi casi e non sempre ad esempio per ambienti simmetrici. In questa circostanza una strategia di controllo target-point, che consiste nella selezione attiva di una sequenza di azioni elementari, risulta una soluzione più opportuna.

Elenco delle figure

Bibliografia

[1] J. Borenstein, L.Feng; Transactions on Robotics and Automation 12 (IEEE), *Measurement and correction of systematic odometry errors in mobile robots*, 1996.

[2] J. Borenstain, H.R. Everett, L. Feng; University of Michigan, *Where am I? Sensors and Methods for Mobile Robot Positioning*, 1996.

[3] J. Borenstein, B. Everett, L. Feng; A. K. Peters, Ltd., Wellesley (MA), *Navigating Mobile Robots: Systems and Techniques*, 1996.

[4] W. Burgard, A. Derr, D. Fox, A. Cremers; Proc. of IROS-98, *Integrating global position estimation and position tracking for mobile robots: the DynamicMarkov Localization approach*, 1998.

[5] W. Burgard, D. Fox, D. Hennig, T. Schmidt; Issue, pp. 2-9, in IEEE Xplore, *Position Tracking with Position Probability Grids*, 1996.

[6] W. Burgard, D. Fox, D. Hennig, T. Schmidt; Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-96), pp. 896-901, *Estimating the absolute position of a mobile robot using position probability grids*, 1996.

[7] W. Burgard, D. Fox, S. Thrun; Issue, pp. 155-162, in IEEE Xplore, *Active Mobile Robot Localization by Entropy Minimization*, 1997.

[8] W. Burgard, D. Fox, S. Thrun; Issue, in IEEE Xplore, *Active Mobile Robot Localization for Mobile Robots*, 1997.

[9] F. Dellaert, D. Fox, W. Burgard, S. Thrun; Journal of Artificial Intelligence Research, *Monte Carlo localization for mobile robots*, 1999.

[10] D. Fox; Ph.D. Diss, University of Bonn (GE), *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Naviagation*, 1998.

[11] L. Kaelbling, A. Cassandra, J. Kurien; In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (Proc. of IROS-96), *Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation*, 1996.

[12] L. Kaelbling; Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence, *On the complexity of solving markov decision problems*, 1995.

[13] J. Kleinberg; Proc. of the 35th IEEE Symposium on Foundations of Computer Science, *The localization problem for mobile robots*, 1994.

[14] B. Kuipers, Y.T. Byun; Robotics and Autonomous Systems 8, *A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations*, 1981.

[15] J. C. Latombe; Kluwer Academic Publishers, *Robot Motion Planning*, 1991.

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

[17] D. Rubin; Oxford University Press, emphUsing the SIR algorithm to simulate posterior distributions. Bayesian Statistics 3, 1998.

[18] R. Simmons and S. Koenig; Proc. International Joint Conference on Artificial Intelligence, *Probabilistic robot navigation in partially observable environments*, 1995.

[19] A. Singhal; *Issues in Autonomous Mobile Robot Navigation*, 1997.

[20] S. Thrun, W. Burgard, D. Fox; Massachussetts Institute of Technology(MIT), Cambridge (MA-UK), *Probabilistic Robotics*, 2006.

[21] S. Thrun, W. Burgard, D. Fox; Robotics and Autonomous Systems 25:3-4, *Active markov localization for mobile robots*, 1999.

[22] S. Thrun; Machine Learning, *Bayesian landmark learning for mobile robot localization*, 1998.

Per la ricerca del materiale sono risultati di fondamentale importanza i siti web:

Google Italia - *www.google.it*

Laboratorium - *www.laboratorium.dist.unige.it*

Citeseer - Scientific Digital Library - *http://citeseer.ist.psu.edu*

IEEE - *www.ieee.com*