UNIVERSITÀ DEGLI STUDI DI ROMA
"TOR VERGATA"

FACOLTÀ DI INGEGNERIA

Tesi di Laurea in Ingegneria dell'Automazione

Risoluzione al problema dello SLAM tramite Unscented FastSLAM e Adaptive Selective Resampling

Relatore:
Ing. F. Martinelli

Laureando
Manuel Cugliari

Anno Accademico 2005-2006

Sommario

Questa tesi si propone di analizzare il complesso e articolato problema della SLAM, acronimo in lingua inglese di Simultaneous Localization And Mapping. Lo scopo iniziale del lavoro era capire cosa la letteratura robotica aveva precedentemente prodotto in merito e tentare di creare un software per lo SLAM (*environment e robot*) in cui implementare i vari algoritmi che risolvono il problema. Successivamente però, dopo un'attenta analisi degli scritti, l'attenzione si è rivoltà al FastSLAM, lo stato dell'arte dello SLAM. Parallelamente allo studio e al tentativo di capire nel profondo tutti i mattoni costituenti il FastSLAM (filtri particellari, state estimation problem, Extended Kalman Filter, Data Association Problem e Sistemi non-lineari), veniva portato avanti un lavoro di analisi e comprensione dei così detti 'Unscented Filtering' e 'Non-linear State Estimation'. Così, usufruenfo dei benefici delle tecniche di tipo Unscented, ampiamente trattate in questa tesi, si sono modificati alcuni aspetti problematici dell' algoritmo con soluzioni del tutto autonome e non reperibili ancora nella letteratura relativa allo SLAM. Innumerevoli simulazioni hanno confermato i miglioramenti apportati con questo tipo di approccio.

Indice

Capitolo 1

Introduzione

SLAM è l'acronimo in lingua inglese di **S**imultaneous **L**ocalization **A**nd **M**apping. Come si intuisce facilmente dal nome è un problema che appartiene alla branca della robotica mobile, in particolare alla navigazione autonoma di veicoli. Lo SLAM è essenzialmente la capacità di un robot, o, più in generale, di un veicolo, di muoversi in un ambiente non noto, localizzado la propria posizione e, simultaneamente appunto, generare una mappa dell'ambiente circostante. Problematiche quali il *motion plannig* o *l'obstacle avoidance*, esulano dagli obiettivi dello SLAM (anche se poi nella pratica realizzazione di un veicolo completamente autonomo andranno affrontate); in linea di principio lo SLAM dovrebbe funzionare anche se al robot fossero impartiti controlli e *task* da entità esterne più informate rispetto ad esso circa la mappa dell'ambiente e la sua posizione, come per esempio da un operatore umano. Esistono davvero molti tipi di approcci per risolvere questo problema, principalmente mediante tecniche che sfruttano il filtro di Kalman e la sua estensione a sistemi non-lineari. Questo approccio allo SLAM, detto appunto EKF-SLAM, verrà in seguito brevemente trattato, in quanto alcuni aspetti di questo tipo di soluzione caratterizzano il FastSLAM, l'oggetto di questa tesi che costituisce un superamento dello SLAM basato su EKF. Infatti, l'affidabilità dell'algoritmo nel suo complesso, la maggiore efficienza in termini di costi computazionali e di occupazione di memoria, la minore sensibilità ai rumori di misura e di moto e la spiccata robustezza nel risolvere le ambiguità nella delicata operazione del *data association*, lo rendono un

ottimo candidato ad un'applicazione in real-time.

Nel corso dello studio del FastSLAM, dopo aver acquisito tutti i molteplici e complessi strumenti matematici di cui tale approccio fa uso, si sono apportate delle profonde modifiche di natura teorica (quindi non semplicemente implementativa), come poi verrà estensivamente descritto. Queste modifiche hanno alla base quelle che impropriamente, ma molto efficacemente, si potrebbero definire tecniche di tipo Unscented, riferendosi con questo giro di parole, all'Unscented Transformation (UT) così come alla sua più immediata conseguenza, ossia all'Unscented Kalman Filter (UKF). Questo tipo di tecniche e il filtro che ne deriva, ideati da Julier et al., hanno riscosso molto successo nell'ambito delle problematiche riguardanti la stima dello stato di sistemi non-lineari soggetti a rumori nella dinamica di misura e di moto. L'idea alla base delle modifiche apportate è proprio quella di fondere i benefici di questo tipo di tecniche con il FastSLAM che, al di là dell'innovativa prospettiva da cui guarda il problema dello SLAM, utilizza ancora, seppur in maniera del tutto diversa dall' EKF-SLAM, il filtro di Kalman Esteso per ciò che riguarda l'update della mappa. In sostanza il FastSLAM viene radicalmente modificato nei suoi tre passi più importanti: calcolo della cosidetta *proposal distribution*, *resampling* particellare e aggiornamento delle *features*, ossia delle caratteristiche dell'ambiente. L'Unscented Transformation (UT) entra in gioco per quel che riguarda il calcolo della *proposal distribution*, ma anche per larga parte dell'andamento dell'algoritmo sotto certe condizioni e per l'aggiornamento della mappa, mentre per il delicatissimo processo di *resampling* particellare si è studiato un approccio di tipo adattivo che verrà poi nel dettaglio discusso in quanto incrementa in modo formidabile le prestazioni del FastSLAM.

Data la natura non sperimentale della tesi, si è proceduto ad una totale virtualizzazione dell'ambiente e del robot. Un'utility sviluppata in *Matlab* permette di creare mappe inserendo vari tipi di elementi (rette, circonferenze, *landmark* puntuali) in modo tale da realizzare qualsiasi tipo di *indoor environment*. Si specifica inoltre il *path* approssimativo che si vuole far seguire al robot, definendo in maniera estremamente intuitiva i *waypoint*, ossia i punti

attraverso i quali il robot deve passare e in funzione dei quali esso genera controlli. In questo ambiente virtuale, modellando opportunamente la dinamica di moto del robot, così come l'odometria e le acquisizioni sensoriali, si è potuto implementare l'algoritmo FastSLAM e sperimentare la consistenza e il reale ed innovativo apporto delle modifiche proposte.

Tutta la tesi è stata sviluppata su piattaforma scientifica *Matlab*. La scelta di tale ambiente è dovuta alla sua estrema flessibilità e potenza, che lo rendono un programma di sviluppo saldamente affermato nella comunità scientifica.

Le innumerevoli simulazioni effettuate provando moltissime mappe e variando i moltissimi paramentri che caratterizzano l'intero algoritmo in analisi, hanno dimostrato l'efficacia delle modifiche di cui sopra. Vengono riportate molteplici confronti tra le prestazioni del FastSLAM per così dire puro e dell'algoritmo con le modifiche proposte.

Si desidera ancora insistere sul grandissimo sforzo teorico compiuto per analizzare nel profondo questo tipo di approccio innovativo allo SLAM e successivamente, dopo averne scovato i potenziali punti fragili, tentare di porvi rimedio autonomamente, confrontandosi in maniera diretta con i più recenti sviluppi della ricerca in merito a questa tematica.

Capitolo 2

Simultaneous Localization And Map building

Lo SLAM ha come protagonisti un'ambiente di qualsiasi tipo, con entità dinamiche e non, e un veicolo che naviga in quest'ambiente, del quale non si dispone di alcuna informazione; appunto, per dirla con una terminologia tipica degli scritti scientifici che trattano di SLAM, in un *unknown environment*. In generale, i criteri, secondo i quali il robot naviga, esulano dal contesto dello SLAM; tuttavia lo SLAM costituisce l'inevitabile primo passo verso il *path plannig*, ossia l' intento di scoprire, nota la mappa -ecco quindi il ruolo dello SLAM- quali siano le traiettorie ottime, secondo criteri di varia natura quali tempo energia o minimo cammino, per raggiungere da un determinato punto della mappa un altro. In genere l'ambientazione che si ipotizza per lo studio di questo problema è di natura statica, ossia vi sono soggetti immobili; tuttavia, complicando ulteriormente il problema, si può pensare che il robot, come poi è in genere nella realtà , sia immerso in un ambiente semi-statico o del tutto dinamico. La Figura 2.1 mostra un tipico *indoor* SLAM planare con il robot e l'ambiente che lo circonda, insieme alla mappa ricostruita.

Lo SLAM negli ultimi anni sta ricevendo grandissima attenzione dalla comunità di ricerca robotica [6, 31]. Questo perchè diventano più frequenti le situazioni in cui è un solo dispositivo, ossia il robot, che affronta l'esplorazione di un ambiente. In genere questo è necessario quando non si dispone di un

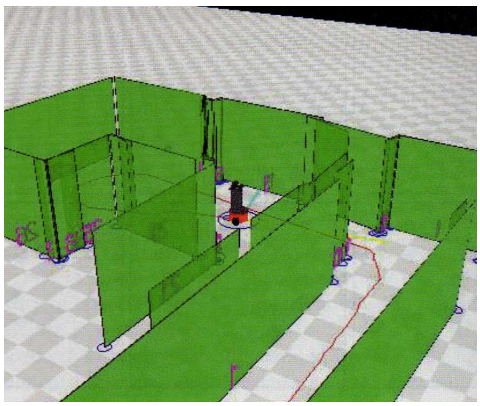


Figura 2.1: Situazione tipica di un *indoor* SLAM.

sistema del tipo Global Position System (GPS), il quale fornisce una stima della posa più o meno accurata del robot. Situazioni tipiche sono l'*indoor*, l'*underwater*, l'*underground* SLAM e le esplorazioni di pianeti. Tuttavia sempre in maniera più frequente i sofisticati algoritmi di SLAM vengono impiegati per contesti di tipo *outdoor*, ossia all'aperto, anche se si dispone di un sistema GPS, poichè il livello di precisione che un ben applicato algoritmo di SLAM può raggiungere è altissimo. Il problema dello SLAM è estremamente più complesso rispetto ad una situazione in cui si dispone dell'informazione circa la posa del robot. Quando appunto sistemi di tipo GPS, o che comunque forniscano la posizione assoluta del robot, non sono disponibili, esso inevitabilmente accumula errore circa la sua posa a causa del rumore a cui è soggetto il suo moto. Così le percezioni sensoriali, oltre ad essere affette dai caratterisitci rumori di misura, soffrono anche della non corretta stima circa la posizione del robot. Come si può facilmente intuire, il rishio di costruire una mappa errata e di commettere un forte errore per evoluzione delle pose del robot, è altissimo. Inoltre la possibilità che tutto l'esperimento fallisca, ossia che la posa del veicolo diverga totalmente dalla reale, con la creazioni per giunta di una mappa totalmente sbagliata, è sempre dietro l'angolo. La robustezza, intesa come capacità di mantenere sempre una corretta stima della posa e della mappa, diventa uno dei prerequisiti fondamentali per un buono, ma soprattutto efficace, SLAM. Può a volte non essere molto importante che la posa stimata si discosti -ovviamente entro certi limiti- dalla reale,

purchè il robot mantenga una buona idea di dove sia e dell'environment sino a lì esplorato.

Altro obbiettivo è, come si dice in gergo, il *closing loop*, ossia la capacità di chiudere il giro. Infatti se il robot compie una navigazione che lo riporta al punto di partenza, esso deve essere in grado di capire che, nel momento in cui si avvicina allo start point, quelle che osserva sono *features* già osservate e non interpretarle come nuovi elementi; deve essere in grado, inoltre, di correggere ulteriormente la sua posa nel momento in cui chiude il giro. Va detto inoltre che, nel caso di online SLAM, questa correzione viene fatta sulla cosiddetta *momentary pose*, ossia non si rimettono in discussione le precedenti pose, in quanto questo appesantirebbe di molto l'algoritmo mettendo a repentaglio la sua efficacia in un contesto real-time.

Come prima sottolineato, il problema dello SLAM è un problema di straordinaria complessità, a causa dell'intreccio che si instaura tra stima della posizione e stima dell'osservazione. Se infatti il percorso del robot fosse conosciuto in precedenza, la creazione della mappa sarebbe un lavoro immediato e sicuramente molto meno problematico. Analogo discorso nel caso in cui si conosca la mappa dell'ambiente, dalla quale è possibile estrarre la stima della posizione. Quest'ultimo è un problema di localizzazione. Nello SLAM, autolocalizzazione e mappatura sono, o almeno come vedremo sembrano, indistricabili. É proprio questo legame intrinsecamente connaturato allo SLAM che lo rende così sensibile all'errore di stima fino alla possibilità di una totale divergenza. La Figura 2.2 chiarisce questo punto: accumulare errore circa la posa del robot (gli ellissi grigi che si espandono) implica che aumenti anche l' incertezza circa la posizione delle *features* (gli ellissi rossi).

I metodi con cui si tenta di ovviare a questo problema sono molti e di vario tipo; sommariamente verranno trattati nel corso dello scritto, se non altro per mettere in luce l'innovativo spunto concettuale che è alla base del FastSLAM, il quale ha nella prospettiva da cui affronta la questione il suo punto più forte. Come si vedrà poi, FastSLAM in un certo senso cerca di violare la natura dello SLAM, tentando di separare, per quanto possibile, i

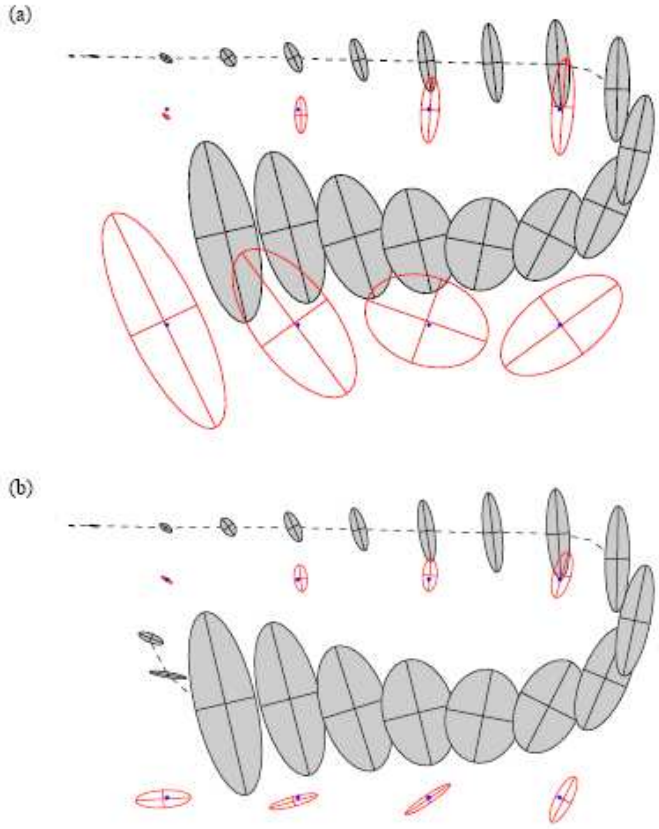


Figura 2.2: Il problema dello SLAM. (a) Il robot si muove in un ambiente non noto. Con il passare del tempo non solo l'errore sulla propria posizione aumenta, come indicato dagli ellissi grigi, ma anche l' errore circa la posizione delle *features* che il robot osserva, come indicato dagli ellissi rossi. (b) Loop closure: riosservando un *landmark* già precedentemente inglobato, il robot riesce a ridurre l'incertezza circa la posa.

problemi di localizzazione e mapping.

2.1 Notazioni dello SLAM

É utile cominciare l'analisi più dettagliata dello SLAM soffermandoci sulla notazione che verrà usata nel corso dello scritto. La posa del robot al tempo t è indicata come s_t. Nel caso di SLAM planare, come quello in esame, s_t è costituito da una terna di coordinate, ossia la coppia di coordinate cartesiane $x - y$ le quali definiscono la posizione assoluta del robot sul piano e l'orien-

tamento α, in genere noto come *heading*, anch'esso da intendersi rispetto ad un riferimento assoluto. L'evoluzione delle pose del robot, o *path*, è definita da s^t:

$$s^t = \{s_1, s_2, ..., s_t\} \tag{2.1}$$

Per qunato riguarda la mappa esistono molti modi per modellare le *feature* geometriche dell'ambiente, quali potrebbero essere muri, angoli, etc. L'approccio utilizzato, che oltre ad essere il più intuitivo e semplice, è anche il più utilizzato, è quello di considerare l'ambiente come un agglomerato di N *landmarks* puntuali immobili. Questi *landmarks* sono descritti da una coppia di coordinate nel piano di riferimento assoluto. Θ appunto definisce l'insieme di queste entità puntuali ed è definito come:

$$\Theta = \{\theta_1, \theta_2, ..., \theta_N\} \tag{2.2}$$

Ciò che il robot sfrutta, oltre alle misure sull'ambiente prelevate dai sensori, sono misure sul proprio moto e quindi sui propri spostamenti. Nel caso più comune un robot reale utilizza *encoders* calettati sull'asse di rotazione dei motori delle ruote. Mediante opportuni calcoli, ossia integrando i dati di tipo incrementale, si calcolano gli spostamenti. Generalmente, quando si affrontano dal punto di vista teorico problemi di SLAM, non si considera nel dettaglio il come o il perchè l'informazioni circa il moto siano ricavate. Il più delle volte si parla di *controls*, ossia di controlli. Analogamente a come si sono definite le altre grandezza, u_t denota il controllo al tempo t, mentre per la storia dei controlli:

$$u^t = \{u_1, u_2, ..., u_t\} \tag{2.3}$$

Comunemente, le informazioni circa la posizione dei *landmark*, vengono restituite in termini di una distanza e da un orientamento, entrambe riferite alla corrente posa del robot. Se l'osservazione al tempo t è indicata come z_t, la serie di tutte le osservazioni è indicata come:

$$z^t = \{z_1, z_2, ..., z_t\} \tag{2.4}$$

Nella realtà pratica, e anche in questa tesi, il robot, nel momento in cui importa nuovi dati sensoriali, non sa quale *landmark* ha osservato. Questo è il contesto del cosidetto *Unknown Data Association*, aspetto che complica notevolmente la questione. Ogni osservazione restituisce una misura circa un certo *landmark* θ_n. Risolvere il problema dell'associazione di dati, e il FastSLAM lo fa egregiamente, vuol dire ipotizzare proprio quale sia n, ossia quale era il *landmark* che si è osservato. Questa identità tuttavia non è mai certa, poichè i *landmarks* possono essere confusi. Si indichera con n_t l'associazione n-esimo *landmark*-osservazione z_t, dove $n_t \in \{1, 2, ..., N\}$. Si consideri per chiarezza, $n_5 = 7$: vuol dire che al time step 7 si è osservato il *landmark* 5. Come per le altre grandezze si indicherà con n^t il set del data association:

$$n^t = \{n_1, n_2, ..., n_t\} \tag{2.5}$$

Ad ogni time step, il robot riceve, se possibile, un set di misure z_t ed effettua un controllo u_t. Beninteso, z_t e u_t sono a loro volta dei vettori che raccolgono tutte le informazioni necessarie. Ad esempio z_t, nel caso in esame sarà un vettore che conterrà una scansione sensoriale del laser[1]. Per quanto riguarda invece u_t, gli elementi che lo compongono possono essere di vario tipo. Nel nostro caso, esso è composto dalla velocità traslazionale V e dalla quella angolare Ω.

Come facilmente si può intuire il numero N di *landmark* è determinante ai fini della complessità dell' algoritmo. Se in un contesto di associazione nota di dati il numero di *landmark* N è determinato *a priori*, in quanto conoscere le corrispondenze implica conoscere il numero di *landmark*, nel caso di *Unknown Data Association* si può immaginare che la mappa sia costituita da infinite entità puntuali, mentre N è il numero di *landmark* che l'algoritmo di SLAM, dopo opportuni stadi di filtering, decide di importare.

[1] L'apparato sensoriale ipotizzato per le simulazioni è costituito da un unico laser, ossia un sistema di misurazione senza contatto a lungo raggio. Laser di questo tipo, quali per esempio il SICK laser range finder, vengono frequentemente utilizzati nelle reali applicazioni dello SLAM.

2.2 Motion model e measurament model

Come in tutti i processi ingegneristici, è necessario modellare le dinamiche del problema e, successivamente, capire come poi queste dinamiche interagiscano tra loro. Per quel che riguarda la dinamica delle acquisizioni sensoriali:

$$p(z_t|s_t, \theta_{n_t}, n_t) = g(\theta_{n_t}, s_t) + \varepsilon_t \tag{2.6}$$

La (2.6) viene detta *measurament model*; in questa distribuzione di probabilità, z_t, cioè le misure, sono condizionate rispetto alla posa corrente, al *landmark* osservato e alla sua identità. Questa distribuzione di probabilità viene modellata dalla funzione (deterministica) non-lineare g che rappresenta come avvengono le misure distorte poi da un rumore di tipo additivo. Questo rumore al tempo t viene modellato da una variabile aleatoria gaussiana con valor medio nullo e matrice di covarianza R_t. L'assunzione di rumore di tipo gaussiano è in genere una buona approssimazione, che lavora bene soprattutto rispetto al range dei sensori[2]. La funzione g semplicemente modellizza la trigonometria delle misure ed è non-lineare nelle coordinate spaziali del robot e della *feature* percepita.

La seconda fonte di informazione del robot, come per le misure, viene descritta da una legge probabilistica, nota come *motion model*, la quale governa l'evoluzione delle pose:

$$p(s_t|u_t, s_{t-1}) = h(u_t, s_{t-1}) + \delta_t \tag{2.7}$$

La posa al tempo t è una funzione del controllo e della posa del robot allo step precedente; h è poi distorta dal rumore al tempo t, gaussiano a valor nullo e matrice di covarianza P_t. Come nel caso delle misure h è, in genere, non-lineare.

[2]Si ricorda che le misure sensoriale trattate si compongono di due elementi: *range* (distanza) del *landmark* colpito e *bearing* (angolo) rispetto al corrente *heading* del robot.

2.3 Formulazione dello SLAM

Il problema dello SLAM ha ragion d'esistere a causa della rumorosità che caratterizza qualsiasi applicazione ingegneristica. Più nel dettaglio, i controlli effettuati dal robot, così come le osservazioni, sono affette da errore come suesposto. Il progressivo accumularsi è ciò che le varie soluzioni allo SLAM tentano di controllare. Il movimento del robot è in genere affetto da errori di natura sistematica, ossia quegli errori causati dalla differenza tra il valore reale (in genere impossibile da conoscere) e quello misurato di una grandezza. Un approccio di tipo probabilistico, come già introdotto con i modelli di moto e di misura, sembra indispensabile. Infatti il cuore dello SLAM è il cosiddetto SLAM *posterior*, ossia la distribuzione probabilista soluzione al problema dello SLAM:

$$p(s_t, \Theta | z^t, u^t, n^t) \tag{2.8}$$

Questa distribuzione di probabilità stima tutte la possibili pose del robot e dei *landmark* da tutte le osservazioni e misure sui controlli. La (2.8) è una distribuzione di probabilità *posterior*, ossia una probabilità condizionata nella quale si tenga conto di dati empirici. Utilizzare una distribuzione di probabilità di tipo *posterior* permette non solo di ottenere la stima migliore della posa del robot e dei *landmark*, ma anche di quantificare l'incertezza di ogni quantità conosciuta attraverso la reciproca correlazione di tali grandezze. Inoltre definire quanto una soluzione sia possibile è la scelta più intelligente in ambienti rumorosi. La maggior parte degli algoritmi di SLAM è un'istanza del filtro di Bayes, ossia la soluzionie ottima per il calcolo ad ogni time step della (2.8). Se il calcolo di questa probabilità avviene in modo ricorsivo, ossia sfruttando in *posterior* al tempo $t-1$, quello che si ottiene è appunto un filtro di Bayes:

$$\begin{aligned} &p(s_t, \Theta | z^t, u^t, n^t) = \\ &\eta p(z_t | s_t, \theta_{n_t}, n^t) \int p(s_t | u_t, s_{t-1}) p(s_{t-1}, \Theta | z^{t-1}, u^{t-1}, n^{t-1}) ds_{t-1} \end{aligned} \tag{2.9}$$

Nella (2.9) η è una costante normalizzante ed in particolare è pari a $p(z_t|z^{t-1}, u^t, n^t)$. La costante normalizzante η non dipende da alcuna grandezza usata per il calcolo del *posterior*. Nel corso della tesi η verrà usata per indicare generiche costanti normalizzanti, indipendentemente dal valore da loro assunto. Dimostriamo ora come la (2.9) viene ottenuta a partire dallo SLAM *posterior* (per maggiori dettagli si veda [35]). Quest' ultimo viene sviluppato secondo la ben nota regola di Bayes:

$$p(s_t, \Theta|z^t, u^t, n^t) = \eta p(z_t|s_t, \Theta, z_{t-1}, u_t, n_t) p(s_t, \Theta|z_{t-1}, u_t, n_t) \tag{2.10}$$

Come successivamente si vedrà, lo SLAM può essere adeguatamente modellato da una catena di Markov. Sfruttando proprio alcune particolari indipendenze del problema, la (2.10) diventa:

$$p(s_t, \Theta|z^t, u^t, n^t) = \eta p(z_t|s_t, \Theta, n_t) p(s_t, \Theta|z_{t-1}, u_t, n_t), \tag{2.11}$$

in quanto, come si può evincere dal *measurement model*, z_t dipende unicamente dalla posa del robot s_t, dalla mappa Θ e dall'ultima identità del *landmark*. Condizionando il termine più a destra della (2.11) rispetto alla posa del robot al tempo $t-1$, e sfruttando il Teorema della Probabilità Totale, si ottiene:

$$\begin{aligned} &p(s_t, \Theta|z^t, u^t, n^t) = \\ &\eta p(z_t|s_t, \Theta, n_t) \int p(s_t, \Theta|s_{t-1}, z^{t-1}, u^t, n^t) p(s_{t-1}|z^{t-1}, u^t, n^t) ds_{t-1} \end{aligned} \tag{2.12}$$

Il termine $p(s_t, \Theta|s_{t-1}, z^{t-1}, u^t, n^t)$ nella (2.12) può essere espanso utilizzando la regola della Probabilità Condizionale ottenendo:

$$\begin{aligned} &p(s_t, \Theta|z^t, u^t, n^t) = \eta p(z_t|s_t, \Theta, n_t) \\ &\int p(s_t|\Theta, s_{t-1}, z^{t-1}, u^t, n^t) p(\Theta|s_{t-1}, z^{t-1}, u^t, n^t) p(s_{t-1}|z^{t-1}, u^t, n^t) ds_{t-1} \end{aligned} \tag{2.13}$$

Il termine più a sinistra dentro l'integrale della (2.13), può essere modificato

in quanto s_t dipende unicamente da s_{t-1} e u_t (si veda la (2.7)):

$$p(s_t, \Theta | z^t, u^t, n^t) = \\ \eta p(z_t | s_t, \Theta, n_t) \int p(s_t | s_{t-1}, u^t) p(\Theta | s_{t-1}, z^{t-1}, u^t, n^t) p(s_{t-1} | z^{t-1}, u^t, n^t) ds_{t-1} \tag{2.14}$$

Leggendo al contrario la regola della Probabilità Condizionale, la (2.14) diviene:

$$p(s_t, \Theta | z^t, u^t, n^t) = \\ \eta p(z_t | s_t, \Theta, n_t) \int p(s_t | s_{t-1}, u^t) p(s_{t-1}, \Theta | z^{t-1}, u^t, n^t) ds_{t-1} \tag{2.15}$$

Nel termine più a destra dell'integrale della precedente equazione, si possono sostituire u_t e n_t con, rispettivamente, u_{t-1} e n_{t-1}, in quanto il controllo e l'associazione di dati al tempo t non hanno influenza su s_{t-1}. Si ottiene così la formula ricorsiva del filtro di Bayes, funzione del *posterior* al time step precedente, del *motion model* e del *measurement model*:

$$p(s_t, \Theta | z^t, u^t, n^t) = \\ \eta p(z_t | s_t, \theta_{n_t}, n^t) \int p(s_t | u_t, s_{t-1}) p(s_{t-1}, \Theta | z^{t-1}, u^{t-1}, n^{t-1}) ds_{t-1} \tag{2.16}$$

Il filtro di Bayes costituisce oramai il cuore della maggior parte degli algoritmi di SLAM. Tutte le soluzioni sono un tentativo di approssimare l'integrale della (2.16), il quale non è risolvibile in forma chiusa. Approcci di tipo statistico, che fanno delle assunzioni circa la forma del *posterior*, non sono altro che approssimazioni più o meno efficaci del filtro di Bayes. Infatti nel caso in cui entrambe g e h siano lineari, la (2.16) equivale al filtro di Kalman. Nel caso di modello di moto e di misura non-lineari, l'Extended Kalman filter (EKF) linearizza con una espansione di Taylor al primo ordine entrambe g e h, applicando poi le consuete equazioni di *prediction* e *update* del filtro di Kalman.

Va fatta un'ultima importante considerazione riguardo lo SLAM *posterior*. Come esplicitato dalla (2.8), il posterior è riferito alla cosiddetta *momentary pose*, e non all'intera evoluzione delle pose, ossia il *path*. Il FastSLAM,

come vedremo, lavora su una distribuzione più complessa che considera l'intero *path*. A prima vista sembrerebbe opinabile la scelta di considerare l'intero insieme delle pose, in quanto si lega il *posterior* in un certo senso alla durata dello SLAM, identificando intuitivamente questa grandezza con la lunghezza del *path* coperto. Al crescere della distanza coperta aumenta lo spazio su cui è definito il *posterior*. Tuttavia il FastSLAM permette di calcolare il *posterior* sul *path* con la stessa efficienza con cui si calcola il *posterior* sulla posa istantanea.

2.4 Associazione di dati robusta

Nelle applicazioni reali dello SLAM, il mapping tra osservazioni e *landmark* n^t raramente è noto. In particolare se l'incertezza circa la posizione dei *landmark* è abbastanza bassa rispetto alla distanza media tra i *landmark*, una semplice euristica può essere efficace nel determinare la corretta associazione di dati. Attualmente il più comune approccio all'associazione di dati è assegnare ogni osservazione al più probabile *landmark* che l'ha prodotta attraverso un criterio di massima verosimiglianza. Nel caso in cui non sia possibile assegnare l'osservazione a nessuno di essi, sotto certe condizioni che poi verranno analizzate nel dettaglio, viene generato un nuovo *landmark* nell'ambiente.

Nel caso dell'EKF la probabilità dell'osservazione può essere intesa come una funzione differenza tra la misura z_t e la misura attesa $\hat{z}_{n_t}$. Questa differenza è nota come innovazione:

$$\begin{aligned}\hat{n}_t &= \underset{n_t}{\operatorname{argmax}}\, p(z_t|n_t, s^t, z^{t-1}, u^t, \hat{n}^{t-1}) \\ &= \underset{n_t}{\operatorname{argmax}}\, \frac{1}{\sqrt{|2\pi Z_t|}} \exp\Big\{ -\frac{1}{2}(z_t - \hat{z}_{n_t})^\mathsf{T} Z_{-1} z_t - \hat{z}_{n_t})\Big\} \end{aligned} \tag{2.17}$$

dove Z_t è la matrice dell'innovazione della covarianza al tempo t. Questa euristica viene spesso riformulata in termini di *negative log likelihood*:

$$\hat{n}_t = \underset{n_t}{\operatorname{argmin}}[\ln Z_t + (z_t - \hat{z}_{n_t})^\mathsf{T} Z_t^{-1}(z_t - \hat{z}_{n_t})] \tag{2.18}$$

Il secondo termine di quest'equazione è la nota *distanza di Mahalanobis*, una metrica normalizzata dalle covarianze delle osservazioni e dalla stima del *landmark*. É per questa ragione che spesso l'associazione di dati che usa questa metrica è indicata come associazione di dati *nearest neighbor* o *nearest neighbor gating*.

L'associazione di dati a massima verosimiglianza lavora in genere bene quando la probabilità della corretta associazione di dati è significativamente più probabile di una associazione sbagliata. Se l'incertezza circa la posizione del *landmark* è alta, più di un'associazione di dati riceverà alta probabilità e se una o più associazioni sbagliate venissero confermate dall'algoritmo, questo indurrebbe catastrofici errori nell'accuratezza della mappa. Questo tipo di problema nell'associazione di dati si verifica facilmrente in presenza di sensori molto rumorosi. Un approccio per questo tipo di problema è incorporare solo quelle osservazioni che rispettano certe condizioni circa l'ambiguità nell'associazione (i.e. che stanno al di sotto della soglia di nearest neighbor). Quindi se lo SLAM è rumoroso nelle osservazioni, una larga percentuale di osservazioni non verrà associata e ciò condurrà ad una sovrastima della covarianza dei *landmark*; le future associazione di dati saranno allora anocora più ambigue. Si capisce allora come il data association sia uno dei passi più delicati di qualsiasi algoritmo di SLAM. Un numero molto alto di approcci sempre più sofisticati sono stati sviluppati proprio per contrastare l'ambiguità in contesti di SLAM molto rumorosi [35].

Capitolo 3

Unscented Filtering

Prima di addentrarci nel FastSLAM, è bene trattare nel dettaglio gli strumenti che stanno alla base delle modifiche apportate all'algoritmo. L'*Unscented Tranformation* (UT), ideata da J.S. Julier et al.[13, 15], è un metodo che nasce a causa della sempre più pressante esigenza di migliorare le prestazioni degli algoritmi di *target traking* o, più in generale, di non-linear state estimation e non-linear filtering. Infatti la stima dello stato in ambienti rumorosi è essenziale per qualsiasi tentativo di controllo di natura retroattiva. In un contesto lineare il filtro di Kalman costituisce la soluzione ottima, ossia quella che fornisce l'MMSE (Minimum Mean Squared Error). Quando la dinamica del sistema è invece di tipo non-lineare, il sistema viene linearizzato con una espansione Taylor al primo ordine e poi vengono applicate al sistema così linearizzato le equazione dello stesso filtro di Kalman. I problemi che si hanno con questo tipo di approccio hanno stimolato la comunità scientifica a cercare nuovi tipi di soluzione. Il problema principale è il delicatissimo passo di linearizzazione: quando l'assunzione di linearità locale è violata, il filtro potrebbere anche essere instabile e causare la divergenza della stima. Inoltre la linearizzazione implica il calcolo di Jacobiani, operazione non sempre di facile realizzazione. Applicazioni poi con un piccolo time step, ossia che lavorano ad alte frequenze, come in genere avviene nei problemi di *target traking* rapido, vengono, da un punto di vista computazionale, appesantite con la linearizzazione. Le operazioni per il calcolo dei Jacobiani e per la predizione della stima dello stato e della covarianza, diventano computazionalmente

poco efficienti, compromettendone l'effettiva realizzabilità in real-time.

3.1 Propagare l'incertezza

Il problema della predizione dello stato e delle osservazione di un qualsiasi sistema può essere formulato come segue. Supponiamo che x sia una variabile random con valor medio $\bar{\mathrm{x}}$ e matrice di covarianza P_x. Una seconda variabile random y è relazionata alla precedente attraverso la funzione non lineare:

$$\mathrm{y} = \mathrm{f}[\mathrm{x}] \tag{3.1}$$

Lo scopo è quello di calcolare il valor medio $\bar{\mathrm{y}}$ e la covarianza P_y di y. I primi due valori statistici di y vengono calcolati (i) provvedendo una distribuzione per y e (ii) valutando i due momenti rispetto a quest'ultima. Nel caso in cui $\mathrm{f}[\cdot]$ sia esattamente lineare, esistono soluzioni in forma chiusa. Comunque tale soluzione non esiste in generale e quindi, come spesso accade, si introducono alcune approssimazioni.

La trasformazione dei due momenti della x è consistente se sussiste la seguente diseguaglianza:

$$\mathrm{P}_y - E[(\mathrm{y} - \bar{\mathrm{y}})(\mathrm{y} - \bar{\mathrm{y}})^{\mathsf{T}}] \geq 0 \tag{3.2}$$

Questa condizione è estremamente importante e quantifica la bontà di un metodo di trasformazione dell'incertezza. Se il filtro di Kalman utilizza stime incostistenti di predizione dello stato e dell'osservazione, questo può produrre anche la divergenza del filtro. Quindi una trasformazione consistente dell'incertezza garantisce un filtro stabile. Tuttavia la trasformazione non solo deve essere consistente, deve essere anche efficente, nel senso che deve minimizzare il termine più a sinistra della (3.2). Infine la stima deve essere *unbiased*, ossia equa, nel senso che $\bar{\mathrm{y}} \simeq E[\mathrm{y}]$.

Il tentativo di realizzare un tale tipo di trasformazione può essere esaminato considerando l'espansione di Taylor della (3.1). Questa serie può essere espressa come:

$$\begin{aligned} \mathrm{f}[\mathrm{x}] &= \mathrm{f}[\bar{\mathrm{x}} + \delta_{\mathrm{x}}] \\ &= \mathrm{f}[\bar{\mathrm{x}}] + \nabla \mathrm{f} \delta x + \frac{1}{2} \nabla^2 \mathrm{f} \delta x^2 + \frac{1}{3!} \nabla^3 \mathrm{f} \delta x^3 + \frac{1}{4!} \nabla^4 \mathrm{f} \delta x^4 + ... \end{aligned} \tag{3.3}$$

dove δx è una variabile aleatoria di tipo Gaussiano con valor medio nullo e matrici di covarianza $\mathrm{P_x}$ e $\nabla^n \mathrm{f} \delta x^n$ è l'n-esimo termine dell'espansione multidimensionale di Taylor. Calcolando il valor atteso, si può dimostrare che per il valor medio e la covarianza valgono:

$$\bar{\mathrm{y}} = \mathrm{f}[\bar{\mathrm{x}}] + \frac{1}{2} \nabla^2 \mathrm{f} P_x + \frac{1}{2} \nabla^4 \mathrm{f} E[\delta x^4] + ... \tag{3.4}$$

$$\begin{aligned} \mathrm{P}_y = \nabla \mathrm{f} \mathrm{P}_x (\nabla \mathrm{f})^{\mathsf{T}} + \frac{1}{2 \times 4!} \nabla^2 \mathrm{f} \Big(E[\delta_x^4] - E[\delta_x^2 P_x] - E[P_x \delta_x^2] + P_x^2 \Big) (\nabla^2 \mathrm{f})^{\mathsf{T}} + \\ \frac{1}{3!} \nabla^3 \mathrm{f} E[\delta_x^4] (\nabla \mathrm{f})^{\mathsf{T}} \end{aligned} \tag{3.5}$$

In altre parole l'n-esimo termine nella serie di $\bar{\mathrm{x}}$ è funzione dell'n-esimo momento della x moltiplicato per la derivata di ordine n di $\mathrm{f}[\cdot]$ valutata in $\mathrm{x} = \bar{\mathrm{x}}$. Se i momenti e le derivate possono essere correttamente calcolati fino all'n-esimo ordine, allora il valor medio è corretto fino all'n-esimo ordine. Analoghe considerazioni valgono per la covarianza, sebbene la struttura di ogni termine sia molto più complicata. Poichè ogni termine nella serie è progressivamente moltiplicato per costanti sempre più piccole, i termini che hanno più influenza sulla corretta propagazione sono i primi. Quindi la predizione si calcolerà sui termini di ordine minore.

La linearizzazione assume che i termini dal secondo ordine in poi di δ_x nella (3.3) siano trascurati. Sotto questa assunzione:

$$\bar{\mathrm{y}} = \mathrm{f}[\bar{\mathrm{x}}] \tag{3.6}$$

$$\mathrm{P_y} = \nabla \mathrm{f} \mathrm{P}_x (\nabla \mathrm{f})^\mathsf{T} \tag{3.7}$$

Comparando queste espressioni con le (3.4) e (3.5), risulta chiaro che queste approssimazioni sono valide solo se i termini dal secondo ordine in poi per il valor medio e dal quarto ordine in poi per la covarinza, sono trascurabili. Nel caso in cui questo non sia verificato, la linearizzazione introduce significativi errori.

3.2 Unscented Transformation

L'*Unscented Transformation* è il mattone fondamentale di un nuovo tipo di filtro che eguaglia le prestazioni del filtro di Kalman in un contesto lineare, mentre migliora notevolmente le prestazioni di quest'ultimo con sistemi non-lineari. Questo nuovo approccio fornisce prestazioni simili a filtri Gaussiani del secondo ordine, pur essendo estremamente più semplice da un punto di vista implementativo. Questo tipo di trasformazione fa in modo che processo e osservazione posssano essere trattati come black boxes. Come il filtro di Kalman, così anche l'UT e l'*Unscented Kalman Filter* (UKF) che ne deriva, lavorano sui primi due momenti della completa *probability density function* (*pdf*) dello stato da stimare. Infatti la soluzione ottima (bayesiana, come prima visto nel caso dello SLAM) implica una totale descrizione della *pdf*, operazione il più delle volte impossibile in quanto essa incorpora caratteristiche quali multimodalità, asimmetrie e discontinuità. Inoltre data la forma non limitata della *pdf*, non la si può, in genere, descrivere con un set limitato di parametri. Le soluzioni utilizzate per ovviare a questo problema sono spesso computazionalmente improponibili o implicano speciali assunzioni circa la forma della *pdf*. Il filtro di Kalman, ancora il più utilizzato, ne costituisce un'approssimazione, lavorando sui primi due momenti della distribuzione, ossia sul valor medio e sulla covarianza. Quello che l'*Unscented Transformation* tenta di fare è propagare nella maniera migliore possibile questi due momenti attraverso sistemi non-lineari, migliorando quindi la stima dello stato delle grandezze del sistema.

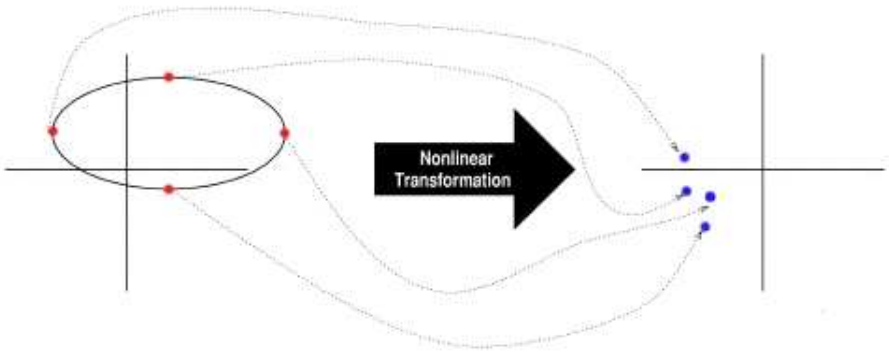


Figura 3.1: Il principio dell'Unscented Transformation

3.3 I Punti Sigma

L'UT si fonda sull'intuizione che è più facile approssimare una distribuzione di probabilità piuttosto che un'arbitraria trasformazione non-lineare [15]. La Figura 3.1 mostra graficamente come lavora l'UT. Un set di punti, detti *Sigma* points (punti sigma) e opportunamente scelti, vengono processati dal sistema non-lineare e, raccolti in uscita, composti secondo alcune regole. Conviene tuttavia, per una correta comprensione, cominciare proprio dai punti *sigma* e capire come e perchè questi vengano scelti in un determinato modo.

Si consideri un generico vettore x di dimensione n, dove n, come vedremo poi, non è la dimensione del sistema di partenza, ma di un sistema opportunamente aumentato. Un set di $p+1$ punti *sigma* $\mathcal{S} = \{\mathcal{X}^i, W^i\}$ è un insieme di vettori, i quali, per semplicità, si definiscono punti. Ogni punto è pesato, nel senso che ad esso è associato un peso. Questo set di punti si dice *deterministicamente* scelto se soddista un set di condizioni delle forma:

$$\mathbf{g}(S, p(\mathrm{x})) = 0 \tag{3.8}$$

dove $p(\mathrm{x})$ è la *pdf*, non necessariamente Gaussiana, e $\mathbf{g}(\cdot\,,\cdot\,)$ determina le informazioni che devono essere catturate riguardo a x. Per esempio, nel caso

Gaussiano, per catturare il valor medio:

$$g_1 = \sum_{i=0}^{p} \mathcal{X}^i W^i - \bar{\mathrm{x}} \tag{3.9}$$

dove appunto $\bar{\mathrm{x}}$ è l'attuale valor medio, mentre per la covarianza:

$$g_2 = \sum_{i=0}^{p} W^i (\mathcal{X}^i - \bar{\mathrm{x}})(\mathcal{X}^i - \bar{\mathrm{x}})^{\mathsf{T}} - \mathrm{P}_x \tag{3.10}$$

Continuando così si può catturare il terzo momento, ossia lo *skew*:

$$g_3 = \sum_{i=0}^{p} W^i (\mathcal{X}^i - \bar{\mathrm{x}})^3 \tag{3.11}$$

Un set di punti di $2n + 1$, *simmetricamente distribuito*, che soddisfa le condizioni di cui sopra, è il seguente [1]:

$$\begin{aligned}
\mathcal{X}^0 &= \bar{\mathrm{x}} \\
W^{0,(m)} &= W^{0,(c)} = \frac{\kappa}{n+\kappa} \\
\mathcal{X}^i &= \bar{\mathrm{x}} + \sqrt{(n+\kappa)\mathrm{P}_x} \\
\mathcal{X}^{(i+n)} &= \bar{\mathrm{x}} - \sqrt{(n+\kappa)\mathrm{P}_x} \\
W^{i,(m)} &= W^{(i+n),(m)} = \frac{1}{2(n+\kappa)} \\
W^{i,(c)} &= W^{(i+n),(c)} = \frac{1}{2(n+\lambda)}
\end{aligned}$$

[1]dove $(\sqrt{(n+\kappa)\mathrm{P}_x})_i$ è l'i-esima riga della decomposizione di Cholesky $\mathbf{A}$, $n\mathbf{P} = \mathbf{A}^{\mathsf{T}}\mathbf{A}$, W^i è il peso associato all'i-esimo punto. Per calcolare la radice della quantità $(n+\kappa)\mathbf{P}_x$ l'UT impiega la decomposizione di Cholesky, ossia la fattorizzazione di una matrice hermitiana e definita positiva in una matrice triangolare inferiore e nella sua trasposta coniugata. La scelta di impiegare

[1] $\mathcal{X}^i$ indica l'i-esimo punto *sigma*, mentre $W^{i,(m)}$ indica il peso dell'i-esimo punto *sigma* relativo al valor medio e $W^{i,(c)}$ il peso relativo alla covarianza

tale tipo di algoritmo è dovuta alla sua stabilità numerica. Il parametro κ serve per catturare terzo e ordini più alti di questo set; n è invece la dimensione del sistema aumentato del rumore (process+noise). Quando la *pdf* è assunta Gaussiana, euristicamente si è soliti fissare $n + \kappa = 3$. Se invece la *pdf* per $\mathbf{x}$ è diversa, bisognerà scegliere un altro valore per κ. Esistono inoltre molti altri tipi di set di punti *sigma* e nel corso della tesi alcuni verranno presi in considerazione, evidenziandone pregi e difetti rispetto al primo set proposto. La Figura 3.2 mostra graficamente i punti *sigma* per una variabile bi-dimensionale Gaussiana. L'altezza dei punti è proporzionale al loro peso. Inoltre, dato che $n = 2$, si contano 5 punti *sigma*.

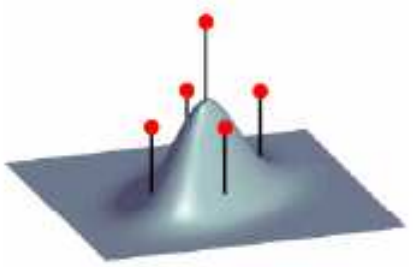


Figura 3.2: Punti *sigma* per una variabile bi-dimensionale Gaussiana

Ai punti *sigma* così calcolati vengono applicate le trasformazioni del sistema non-lineare, il quale per comodità viene indicato come una generica $\mathbf{f}(\mathbf{x})$ producendo una nuova nube di punti trasformati. In formule:

$$\mathcal{Y}_i = \mathbf{f}[\mathcal{X}^i] \tag{3.12}$$

Il nuovo valor medio è dato da una statistica pesata dei punti trasformati:

$$\bar{\mathbf{y}} = \sum_{i=0}^{p} W^{i,(m)} \mathcal{Y}_i \tag{3.13}$$

mentre la nuova matrice di covarianza è pari al prodotto punto pesato della nube dei punti trasformati:

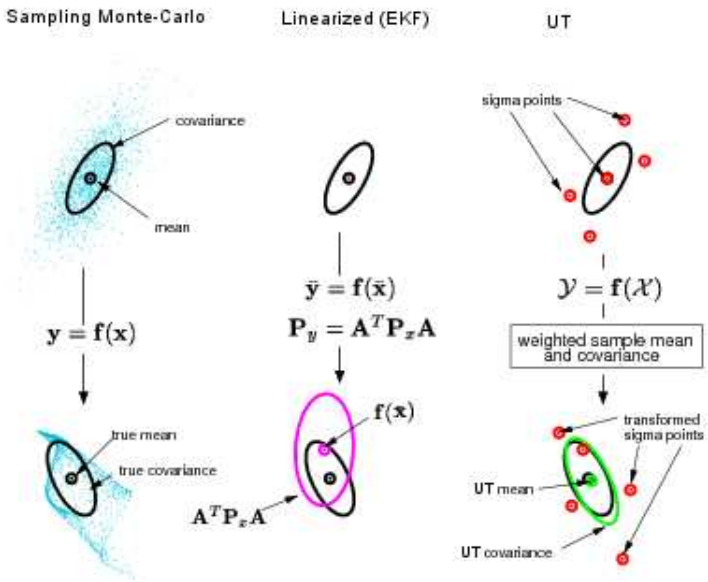


Figura 3.3: Esempi di propagazione del valor medio e della covarianza. a) valor medio e covarianza attuali calcolati con tecniche Monte-Carlo, b) approccio linearizzato del tipo EKF e c) Unscented Transformation

$$\mathbf{P}_y = \sum_{i=0}^{p} W^{i,(c)}(\mathcal{Y}_i - \bar{\mathbf{y}})(\mathcal{Y}_i - \bar{\mathbf{y}})^{\mathsf{T}} \tag{3.14}$$

La Figura 3.3 mette a contronto il risultato di una metodica linearizzante tipica dell'Extended Kalman Filter (EKF) con l'*Unscented Trasformation*. Questi due tipi di approccio vengono paragonati nell'approssimazione del reale valor medio e covarianza ottenuti con sampling Monte-Carlo, ossia eseguendo un numero abbastanza elevato di esperimenti, in modo da osservare come si distribuiscono i campioni nello spazio delle configurazioni (in questo caso uno spazio bidimensionale). L'errore che si compie con la soluzione EFK è abbastanza elevato, soprattutto per la covarianza. L'UT riesce, invece, bene nell'approssimazione con soli 5 punti *sigma*.

Poichè il valor medio e la covarianza di x sono calcolati precisamente fino al secondo ordine, il valor medio e la covarianza per y sono anch'essi correttamente stimati fino al second'ordine. Questo significa che il valor medio è calcolato con maggior precisione rispetto a quanto non faccia l'EKF. Poichè è la distribuzione di x che viene approssimata puittosto che f[·], la sua espansione in serie non è troncata ad un particolar ordine. Si può dimostrare che l'algoritmo Unscented è parzialmente in grado di incorporare informazioni circa termini di grado più elevato rispetto al secondo nell'espansione, garantendo quindi nel complesso maggiore accuratezza.

3.4 Tuning dei Punti Sigma

Come suddetto, quello presentato non è l'unico set di punti *sigma*. Ne esistono di diversi [12, 17]: ciò che li contraddistingue è il numero di punti *sigma*, quantità che indica in un certo senso quanto il set è in grado di approssimare valor medio e covarianza, ossia fino a che ordine questa approssimazione è precisa. Tuttavia un set con molti punti *sigma*, sebbene garantisca una miglior approssimazione, deve fare i conti con il costo computazionale che necessariamente ad esso si accompagna. D'altra parte minimizzare il numero di punti utilizzati non sembra una scelta ragionevole, in quanto contrasta con l'obbiettivo di maggior precisione nella propagazione dell'informazione e dell'incertezza ad essa associata.

Un set di punti [23], molto simile al precedente (ossia di $2n + 1$ entità e simmetricamente distribuito), è un utile esempio di come si possano maneggiare questi punti secondo alcune esigenze. Il set è molto simile al precedente e ne costituisce una specie di scompattamento e riformulazione introducendo, appunto, alcuni parametri che permettono una sorta di *tuning* dei punti. Il

set è il seguente:

$$\begin{aligned}
\mathcal{X}^0 &= \bar{\mathrm{x}} \\
W^{0,(m)} &= \frac{\lambda}{n+\lambda} \\
W^{0,(c)} &= \frac{\lambda}{n+\lambda} + (1-\alpha^2+\beta) \\
\mathcal{X}^i &= \bar{\mathrm{x}} + \sqrt{(n+\lambda)\mathrm{P}_x} \\
\mathcal{X}^{(i+n)} &= \bar{\mathrm{x}} - \sqrt{(n+\lambda)\mathrm{P}_x} \\
W^{i,(m)} &= W^{(i+n),(m)} = \frac{1}{2(n+\lambda)} \\
W^{i,(c)} &= W^{(i+n),(c)} = \frac{1}{2(n+\lambda)}
\end{aligned}$$

con $\lambda = \alpha^2(n+\iota) - n$. La costante α determina il cosiddetto *spread* dei punti *sigma*, ossia quanto la nube iniziale di punti sia schiacciata attorno all'attuale valor medio. In genere $1e{-}4 \leq \alpha \leq 1$. ι è un altro altro dei cosiddetti *scaling parametrer*, e viene poato uguale a zero per i problemi di stima dello stato, mentre viene fissato a $3-n$ per i problemi di stima dei parametri. Da ultimo, β è un parametro per includere nell'algoritmo una conoscenza *a priori* della distribuzione di x; per distribuzioni Gaussiane $\beta = 2$ costituisce la scelta ottimale.

3.5 L'Unscented Kalman Filter

Utilizzare la metodica del filtro di Kalman, sfruttando però l'*Unscented Transformation*, permette di creare il cosiddetto *Unscented Filter* o *Unscented Kalman Filter* (UKF) [14, 24]. Si consideri il sistema a tempo-discreto:

$$\mathrm{x}_{k+1} = \mathrm{f}(\mathrm{x}_k, \mathrm{u}_k) + \delta_k \tag{3.5a}$$

$$\mathrm{z}_k = \mathrm{h}(\mathrm{x}_k) + \varepsilon_k \tag{3.5b}$$

dove x_k rappresenta lo stato del sistema al tempo t, u_k rappresenta l'ingresso e y_k è il vettore delle misure al tempo k. Il rumore sullo stato è descritto

dalla variabile aleatoria δ_k, mentre quello sulla misura da ϵ_k. Si fa la ben nota ipotesi che entrambe queste due variabili siano rumori di tipo bianco (*white noise*) e Gaussiani con matrici di covarianza, rispettivamente, Q_t e R_t. Come il filtro di Kalman, così anche l'Unscented Filter, lavora con i due ben noti passi di *prediction* e *update*. Tuttavia lo stato considerato dall'UKF è aumentato del rumore di processo e del rumore di misura. Ossia:

$$\mathrm{x}_k^a = \begin{bmatrix} \mathrm{x}_k \\ \delta_k \\ \varepsilon_k \end{bmatrix} \tag{3.15}$$

considerando quindi i rumori come una variabile di sistema e ristrutturando la f e la h:

$$\mathrm{x}_{k+1}^a = \tilde{\mathrm{f}}(\mathrm{x}_k^a, \mathrm{u}_k) \tag{3.16}$$

$$\mathrm{z}_{k+1} = \tilde{\mathrm{h}}(\mathrm{x}_k^a) \tag{3.17}$$

Una volta generato lo stato aumentato del sistema, si costruisce la matrice di covarianza di questo nuovo stato aumentato, utilizzando la matrice di covarianza dello stato di partenza P_k, e le due matrici di covarianza del rumore di processo e di misura, ossia rispettivamente Q_k e R_k:

$$\mathrm{P}_{k|k}^a = \begin{bmatrix} \mathrm{P}_{k|k} & \mathrm{P}_{k|k}^{x\delta} & \mathrm{P}_{k|k}^{x\varepsilon} \\ \mathrm{P}_{k|k}^{x\delta} & \mathrm{Q}_{k|k} & \mathrm{P}_{k|k}^{\delta\varepsilon} \\ \mathrm{P}_{k|k}^{x\varepsilon} & \mathrm{P}_{k|k}^{\delta\varepsilon} & \mathrm{R}_{k|k} \end{bmatrix} \tag{3.18}$$

I termini fuori diagonale della (3.18) rappresentano le reciproche correlazioni di stato e rumori di osservazione e di processo. Quando, come nel caso in esame, non esiste correlazione, la matrice diventa diagonale a blocchi. Come nel caso del EKF (nel resto della tesi, date le dinamiche non-lineari, ci riferiremo quasi sempre all'estensione a sistemi non lineari del filtro di Kalman), si considerano i valori iniziali:

$$\hat{\mathrm{x}}_0 = E[\mathrm{x}_0] \tag{3.19}$$

$$\mathrm{P}_0 = E[(\mathrm{x}_0 - \hat{\mathrm{x}}_0)(\mathrm{x}_0 - \hat{\mathrm{x}}_0)^{\mathsf{T}}] \tag{3.20}$$

e l'UKF viene inizializzato con i valori dello stato di aprtenza aumentato però nel seguente modo:

$$\hat{\mathrm{x}}_0^a = E[x^a] = E\left[\mathrm{x}_0^\mathsf{T} \quad \mathbf{0} \quad \mathbf{0}]^\mathsf{T}\right] \tag{3.21}$$

$$\mathrm{P}_0^a = E[(\mathrm{x}_0^a - \hat{\mathrm{x}}_0^a)(\mathrm{x}_0^a - \hat{\mathrm{x}}_0^a)^\mathsf{T}] = \begin{bmatrix} \mathrm{P}_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathrm{Q}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathrm{R}_0 \end{bmatrix} \tag{3.22}$$

[2]Successivamente, si calcolano i punti *sigma* rispetto allo stato aumentato, con le strategie di selezione suesposte:

$$\mathcal{X}_k = \left[\hat{\mathrm{x}}_k \quad \hat{\mathrm{x}}_k + \gamma\sqrt{\mathrm{P}_k} \quad \hat{\mathrm{x}}_k - \gamma\sqrt{\mathrm{P}_k}\right] \tag{3.23}$$

con $\gamma = \sqrt{n+\lambda}$ e $k \in \{1, ..., \infty\}$; n è la dimensione del sistema aumentato e per λ si veda la sezione (3.4). Usufruendo dell'algoritmo di selezione dei punti *sigma*esposto, si avranno quindi $2n+1$ punti. A questi punti viene applicata la trasformazione dello stato (non la f di partenza, bensì la $\tilde{\mathrm{f}}$):

$$\mathcal{X}^*_{k+1|k} = \tilde{\mathrm{f}}[\mathcal{X}_k, \mathrm{u}_k] \tag{3.24}$$

Si potrà quindi ottenere il valor medio *a priori* come:

$$\hat{\mathrm{x}}_k^- = \sum_{i=0}^{2n} W_i^{(m)} \mathcal{X}^*_{i,k+1|k} \tag{3.25}$$

mentre per la matrice di covarianza *a priori*

$$\mathrm{P}_k^- = \sum_{i=0}^{2n} W_i^{(c)} \left(\mathcal{X}^*_{i,k+1|k} - \hat{\mathrm{x}}_k^-\right)\left(\mathcal{X}^*_{i,k+1|k} - \hat{\mathrm{x}}_k^-\right)^\mathsf{T} \tag{3.26}$$

Si sostituiscono poi i punti *sigma* nel modello di misura, generando il set

$$\mathcal{Z}_{k+1|k} = \tilde{\mathrm{h}}[\mathcal{X}_{k+1|k}] \tag{3.27}$$

[2]Consideriamo d'ora in poi $\mathrm{x}_k^a = \mathrm{x}_k$ per non appesantire ulteriormente la scrittura ricordando che con x_k si intende lo stato aumentato. Analogamente per la covarianza $\mathrm{P}_k^a = \mathrm{P}_k$.

il quale, opportunamente pesato, genera la predizione della misura:

$$\hat{\mathrm{z}}_k^- = \sum_{i=0}^{2n} W_i^{(m)} \mathcal{Z}_{i,k+1|k} \tag{3.28}$$

Come nella nota struttura del KF, le equazioni di *prediction* vengono in un certo senso corrette dalle equazioni di *update*, sfruttando le nuove misure. Si calcolano prima le seguenti due matrici, che rappresentano, rispettivamente, la covarianza della misura e la correlazione incrociata della misura con lo stato:

$$\mathrm{P}_{\tilde{z}_k,\tilde{z}_k} = \sum_{i=0}^{2n} W_i^{(c)} \big(\mathcal{Z}_{i,k+1|k} - \hat{\mathrm{z}}_k^-\big)\big(\mathcal{Z}_{i,k+1|k} - \hat{\mathrm{z}}_k^-\big)^{\mathsf{T}} \tag{3.29}$$

$$\mathrm{P}_{x_k,z_k} = \sum_{i=0}^{2n} W_i^{(c)} \big(\mathcal{X}_{i,k+1|k} - \hat{\mathrm{x}}_k^-\big)\big(\mathcal{Z}_{i,k+1|k} - \hat{\mathrm{z}}_k^-\big)^{\mathsf{T}} \tag{3.30}$$

Dopo aver calcolato il guadagno di Kalman come:

$$\mathcal{K}_k = \mathrm{P}_{x_k,z_k} \mathrm{P}_{\tilde{z}_k,\tilde{z}_k}^{-1} \tag{3.31}$$

si calcola la stima e la covarianza *a posteriori*:

$$\hat{\mathrm{x}}_k = \hat{\mathrm{x}}_k^- + \mathcal{K}_k(\mathrm{z}_k - \hat{\mathrm{z}}_k^-) \tag{3.32}$$

$$\mathrm{P}_k = \mathrm{P}_k^- - \mathcal{K}_k \mathrm{P}_{\tilde{z}_k,\tilde{z}_k} \mathcal{K}_k^{\mathsf{T}} \tag{3.33}$$

Capitolo 4

Il FastSLAM

In questo capitolo viene presentato il FastSLAM [26, 27, 28, 29, 30], ossia un algoritmo innovativo che risolve in maniera eccezionale il problema dello SLAM. Prima di iniziare l'analisi dell'algoritmo, sembra utile spendere due parole sulla storia del FastSLAM. Questo algoritmo è stato portato ai massimi livelli da M. Montemerlo, grazie agli enormi passi compiuti nel campo dello studio dei filtri particellari presso la Standford University. Anche se il FastSLAM costituisce il superamento dell'EKF-SLAM, dell'EKF conserva ancora molti aspetti e metodi.

Il merito principale da riconoscere al FastSLAM è la capacità di rimettere in discussione l'intero modo di guardare al problema, scrollandosi di dosso l'impostazione data dell'EFK-SLAM e mettendosi in una nuova prospettiva, dalla quale trovare una soluzione più semplice ed efficace. Semplice nel senso che si vedrà poi, dato che in realtà l'implementazione del FastSLAM è tutt'altro che semplice. Ciò che il FastSLAM fa è sfruttare la *sparsità strutturale* dello SLAM per decomporre il problema in problemi più piccoli, fattorizzando lo SLAM *posterior* in stime di bassa dimensione. Ciò che rende l'EKF-SLAM computazionalmente ed strutturalmente inefficace è proprio il tentativo di imbrigliare in un'unica problematica questa sparsità strutturale.

Il primo paragrafo di questo capitolo è dedicato all'EKF-SLAM, il quale viene sommariamente descritto al fine di capire poi quale alternativa abbia

proposto il FastSLAM e cosa in esso rimane ancora dell'EKF-SLAM.

4.1 L'EKF-SLAM

L'EKF-SLAM [1] modella il *posterior* come una Gaussiana di dimensione elevata sulla posa del robot e su tutte *feature* della mappa. Gli elementi off-diagonal della Gaussiana rappresentano le correlazioni esistenti tra le variabili in gioco. A prima vista questa soluzione sembra calzare a pennello, soprattutto per la possibilità di modellare le interazioni tra stima della posa e *landmark* osservati. La variabile aleatoria dello stato complessivo si compone, quindi, della posa del robot e di tutte le *feature* attualmente note. In un contesto di *Unknow Data Association* la dimensione del vettore di stato varierà nel tempo. Questo rende l'EKF-SLAM adatto a SLAM in ambienti limitati nella dimensione, proprio a causa della limitatezza delle risorse computazionali di cui si dispone. Esistono tuttavia degli approcci che ottimizzano l'utilizzo di risorse dell'EKF-SLAM, ma non risolvono i problemi alla base.

Lo SLAM *posterior* è modellato da una multivariata Gaussiana come segue:

$$p(s_t, \Theta | z^t, u^t, n^t) = n(x_t; \mu_t, \Sigma_t) \tag{4.1}$$

dove la variabile complessiva di stato x_t è del tipo[1]:

$$x_t = \{s_t, \theta_t^1, \theta_t^2, ..., \theta_t^N\} \tag{4.2}$$

mentre il valor medio μ_t, che rappresenta la migliore stima che il filtro riesce a fare della posa reale al tempo t è pari a:

$$\mu_t = \{\mu_{s_t}, \mu_{\theta_t^1}, \mu_{\theta_t^2}, ..., \mu_{\theta_t^N}\} \tag{4.3}$$

e infine la matrice di covarianza Σ_t che modella le correlazioni tra le variabili

[1] Solo in questo caso l'n-esimo *landmark* al tempo t è indicato come θ_t^n

in gioco sempre al tempo t è data dalle seguente:

$$\Sigma_t = \begin{bmatrix} \Sigma_{s_t} & \Sigma_{s_t,\theta_t^1} & \cdots & \Sigma_{s_t,\theta_t^N} \\ \Sigma_{s_t,\theta_t^1} & \Sigma_{\theta_t^1} & \Sigma_{\theta_t^1,\theta_t^2} & \vdots \\ \vdots & \Sigma_{\theta_t^2,\theta_t^1} & \ddots & \vdots \\ \Sigma_{s_t,\theta_t^N} & \cdots & \cdots & \Sigma_{\theta_t^N} \end{bmatrix} \tag{4.4}$$

Il *posterior* considerato è sulla mappa e sulla posa istantanea s_t, non sull'intero *path* s^t. Come accennato, il punto debole dell'EKF-SLAM sta nel costo computazionale, sia in termini di occupazione di memoria che di tempo di esecuzione. Infatti, per uno SLAM planare, μ_t ha dimensione $2N + 3$, dove N è il numero totale di *landmark* e il 3 è dovuto al fatto che nel piano la posa del robot è definita dalle due coordinate cartesiane e dall'orientamento. La matrice di covarianza avrà quindi dimensione $\mathcal{O}(N^2)$ (precisamente sarà $2N + 3 \times 2N + 3$). Quindi la memoria cresce come N^2; inoltre ogni update sensoriale richiede tempo $\mathcal{O}(N^2)$ per essere processato, in quanto l'aggiornamento o l'aggiunta di un *landmark* influenza tutte le altre variabili. Questa complessità quadratica comporta l'esigenza di fissare a poche centinaia il numero di *landmark* importati, mentre in generale l'ambiente è formato da milioni di features. Da ultimo, come ben sappiamo, il cuore dell'EKF è la linearizzazione delle dinamiche di moto e di misura. Questa linearizzazione è un processo molto delicato: prima di tutto viene fatta sull'attuale stima dello stato (che potrebbe essere affetta da grave errore) ed inoltre è efficate solo se la dinamica del sistema non presenta spiccate caratteristiche di non-linearità. Questo in genere avviene se il time step dell'algoritmo è abbastanza piccolo, evitando in tal modo, proprio a causa dell'esiguità dell'intervallo di tempo, andamenti di tipo impulsivo che possono mettere a dura prova l'EKF.

4.2 Strutturare lo SLAM posterior

Il FastSLAM [29] decompone, in modo preciso e non approssimato, il problema dello SLAM in un ragionevole numero di sottoproblemi. Mentre altri tipi di approcci stimano il *posterior* sulla posa del robot e sulla mappa, il FastSLAM considera l'intera evoluzione delle pose, ossia il *path* e la mappa:

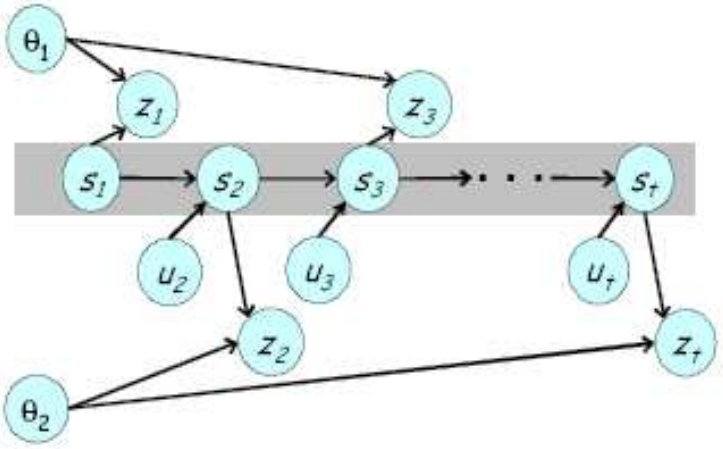


Figura 4.1: Lo SLAM come una catena di Markov. Il robot si muove dalla posa iniziale s_1 attraverso la serie di controlli $u_1, ..., u_t$. Al tempo $t = 1$ osserva il *landmark* θ_1 dei due *landmark* $\{\theta_1, \theta_2\}$ attraverso la misura z_1. Al tempo $t = 2$ osserva l'altro *landmark* θ_2, mentre al tempo $t = 3$ osserva di nuovo il *landmark* θ_1. Se il percorso del robot fosse noto (sctriscia grigia) le posizioni dei *landmark* $\{\theta_1$ e $\theta_2\}$ sarebbero indipendenti così come la loro stima.

$$p(s^t, \Theta | z^t, u^t, n^t) \tag{4.5}$$

Questa sottigliezza, che a prima vista può sembrare una complicazione, permette invece la cosiddetta fattorizzazione delo SLAM. Per dimostrarla, bisogna interpretare il problema dello SLAM come una catena di Markov o più in generale come una rete dinamica di Bayes (Dynamic Bayes Network), la quale permette di evidenziare le interdipendenze degli stati del sistema. A tal proposito si consideri la Figura 4.1, che appunto modella lo SLAM come una catena di Markov. La posa del robot al tempo t, s_t, è influenza solo dai controlli $u_1, ..., u_t$. Le osservazioni $z_1, ..., z_t$ sono invece condizionate dalla posizione θ_n del *landmark* e dalla posa del robot nell'istante di tempo in cui la misura è importata. Se, attraverso un oracolo, conoscessimo con precisione il *path* del robot, ossia riuscissimo ad estrapolare ciò che nella Figura 4.1 è evidenziato per chiarezza con la striscia grigia, potremmo considerare le misurazioni della posizione dei *landmark indipendenti*, ossia come N problemi di stima separati. Quindi ogni *landmark* sarebbe indipendente dall'altro, in quanto l'osservazione di uno non restituirebbe alcuna informazione circa la posizione degli altri. É quindi possibile *fattorizzare* lo SLAM *posterior*

nel prodotto esatto di termini più piccoli. Il FastSLAM decompone così il problema dello SLAM in un problema di localizzazione del robot e in un sottoinsieme di problemi indipendenti di stima dei *landmark*. Questa fattorizzazione, esatta e non approssimata proprio a causa delle indipendenze intrinseche nella natura dello SLAM, è stata per la prima volta ottenuta da Murphy [30]. In formule:

$$p(s^t, \Theta | z^t, u^t, n^t) = p(s^t | z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n | s^t, n^t, z^t) \tag{4.6}$$

Il termine più a sinistra della (4.6) è il *path posterior*, mentre quello più a destra rappresenta N *landmark posterior* condizionati sulla stima del percorso.

4.2.1 Dimostrazione della fattorizzazione dello SLAM posterior

Attraverso il teorema della probabilità condizionata lo SLAM *posterior* può essere descritto nel seguente modo:

$$\begin{aligned} p(s^t, \Theta | z^t, u^t, n^t) &= p(s^t | z^t, u^t, n^t) p(\Theta | s^t, u^t, z^t, n^t) \\ &= p(s^t | z^t, u^t, n^t) p(\Theta | s^t, z^t, n^t) \end{aligned} \tag{4.7}$$

notando che la seconda semplificazione è dovuta al fatto che il controllo al tempo t u_t non condiziona la mappa. Quindi per dimostrare la (4.6) è necessario dimostrare che:

$$p(\Theta | s^t, z^t, n^t) = \prod_{n=1}^{N} p(\theta_n | s^t, z^t, n^t) \tag{4.8}$$

La (4.8) può essere dimostrata per induzione. Prima però occorre distinguere tra due possibili casi, ossia se la *feature* θ_n sia stata o meno osservata al tempo t. In particolare se $n_t \neq n$ le più recenti misure z_t non hanno alcun effetto sul *posterior*, nè tantomeno sulla posa del robot s_t o sulla corrispondenza n_t: Quindi:

$$p(\theta_n | s^t, z^t, n^t) \overset{Markov}{=} p(\theta_n | s^{t-1}, z^{t-1}, n^{t-1}) \tag{4.9}$$

Se invece $n_t = n$, ossia se $\theta_n = \theta_{n_t}$ è stato osservato nelle più recenti osservazioni z_t allora:

$$p(\theta_{n_t}|s^t, n^t, z^t) \overset{Bayes}{=} \frac{p(z_t|\theta_{n_t}, s^t, n^t, z^{t-1})}{p(z_t|, s^t, n^t, z^{t-1})} p(\theta_{n_t}|s^t, n^t, z^{t-1})$$
$$\overset{Markov}{=} \frac{p(z_t|s_t, \theta_{n_t}, n^t)}{p(z_t|, s^t, n^t, z^{t-1})} p(\theta_{n_t}|s^{t-1}, n^{t-1}, z^{t-1}) \tag{4.10}$$

Risolvendo poi rispetto al termine più a destra otteniamo il valore di $p(\theta_{n_t}|s^{t-1}, n^{t-1}, z^{t-1})$:

$$p(\theta_{n_t}|s^{t-1}, n^{t-1}, z^{t-1}) = \frac{p(\theta_{n_t}|s^t, n^t, z^t) p(z_t|, s^t, n^t, z^{t-1})}{p(z_t|s_t, \theta_{n_t}, n^t)} \tag{4.11}$$

La (4.8) viene ora dimostrata per induzione matematica. Si suppone che il *posterior* al tempo $t-1$ sia già fattorizzato:

$$p(\Theta|s^{t-1}, z^{t-1}, n^{t-1}) = \prod_{n=1}^{N} p(\theta_n|s^{t-1}, n^{t-1}, z^{t-1}) \tag{4.12}$$

La (4.12) è banalmente vera al tempo 1, in quanto all'inizio il robot non conosce alcuna *feature* e quindi tutte le stime di quest'ultime sono indipendenti. Al tempo t invece il *posterior* assume la seguente forma:

$$p(\Theta|s^t, z^t, n^t) \overset{Bayes}{=} \frac{z_t|\Theta, s^t, n^t, z^{t-1}}{p(\Theta|s^t, n^t, z^{t-1})} p(z_t|, s^t, n^t, z^{t-1})$$
$$\overset{Markov}{=} \frac{p(z_t|s_t, \theta_{n_t}, n_t)}{p(\Theta|s^{t-1}, n^{t-1}, z^{t-1})} p(z_t|, s^t, n^t, z^{t-1}) \tag{4.13}$$

Sostituendo la (4.13) nell'ipotesi induttiva data dalla (4.11), si ottiene:

$$p(\Theta|s^t, z^t, n^t) \overset{Ind}{=} \frac{p(z_t|s_t, \theta_{n_t}, n_t)}{p(z_t|, s^t, n^t, z^{t-1})} \prod_{n=1}^{N} p(\theta_n|s^{t-1}, n^{t-1}, z^{t-1})$$
$$= \frac{p(z_t|s_t, \theta_{n_t}, n_t)}{p(z_t|, s^t, n^t, z^{t-1})} \underbrace{p(\theta_{n_t}|s^{t-1}, n^{t-1}, z^{t-1})}_{Eq.4.11} \prod_{n \not= t} \underbrace{p(\theta_{n_t}|s^{t-1}, n^{t-1}, z^{t-1})}_{Eq.4.8}$$
$$= p(\theta_{n_t}|s^t, n^t, z^t) \prod_{n \not= t} p(\theta_n|s^t, n^t, z^t)$$
$$= \prod_{n=1}^{N} p(\theta_n|s^t, n^t, z^t) \tag{4.14}$$

Va inoltre ancora sottolineata l'importanza di condizionare il *posterior* sull'intero *path* e non unicamente sulla posa istantanea, che dal punto di vista dell'intensità condizionante, è più debole.

4.3 L'algoritmo del FastSLAM

Attualmente esistono due versioni dell'algoritmo, ossia il FastSLAM 1.0 e il FastSLAM 2.0 (una trattazione estensiva si può trovare in [35]). Entrambi sfruttano appieno la fattorizzazione precedentemente dimostrata, tuttavia il secondo, essendo una evoluzione del primo, è estremamente più complicato da implementare. La nostra analisi vertèrà proprio su questo, evidenziando in alcuni punti l'approccio più semplice del FastSLAM 1.0 [30].

La struttura del FastSLAM è essenzialmente costituita da $N+1$ filtri di bassa dimensionalità attraverso i quali si stima il *posterior* strutturato; $N+1$ come il numero di termini da stimare che si contano nella (4.6). Entrambe le due versioni dell'algoritmo sfruttano un filtro particellare [3, 37] per la stima del *path posterior* $p(s^t|z^t, u^t, n^t)$ in modo simile a come si fa oramai da un pò nei problemi di localizzazione [5], di *mapping* e di *visual tracking*. La scelta del filtro particellare è dovuta al fatto che con questo tipo di scelta ogni aggiornamento, come vedremo, richiede tempo costante, indipendentemente dalla lunghezza del *path* al tempo t; inoltre il filtro particellare ben si adatta alla dinamica non-lineare del *motion model*. I restanti N filtri servono per la stima dei rimanenti N *posterior* (condizionati) circa la posizione delle *feature* presenti nella mappa $p(\theta_n|s^t, n^t, z^t)$. Prima di passare ad analizzare nel dettaglio il FastSLAM, dedichiamo il prossimo paragrafo alla descrizione del filtro particellare, che costituisce il cuore dell'algoritmo.

4.3.1 Il filtro particellare

Nella sostanza il problema dello SLAM è un problema di *dynamic state estimation*. Guardando il problema da una prospettiva Bayesiana si tenta di costruire la posterior *probability density function* (pdf) da tutte le informazioni disponibili, incluso l'ultimo set di misure ricevute. Trovare la corretta *pdf* vuol dire risolvere completamente il problema della stima dello stato. Ci sono tipologie di problema in cui è richiesta una stima in ogni istante in cui si riceve una misura. L'approccio migliore sembra allora essere quello di un filtro ricorsivo, che la stima in due fasi: *prediction* e *update*. Per la fase di predizione si utilizza il modello del sistema per predire la *pdf* dello stato da

un istante di tempo a quello successivo. Poichè la dinamica del processo è in genere soggetta a disturbi (modellati come variabili aleatorie), la predizione in genere trasla, deforma ed espande la *pdf*. L'operazione di *update* sfrutta l'ultimo set di misura proprio per correggere la predizione.

La soluzione ottima, come si è visto, è costituita dal filtro di Bayes. Tuttavia non esiste per questo filtro una soluzione in forma chiusa. Il filtro particellare, cosi come l'EKF o metodiche di tipo grid-based filter, costituiscono una famiglia di algoritmi sub-ottimi che tentano di approssimare la soluzione ottima. Mentre il filtro di Kalman e il filtro di Kalman esteso ipotizzano una *pdf* di tipo multivariata Gaussiana secondo quanto visto nel Capitolo 2, il problema dello SLAM si riduce o , per meglio dire, si concretizza nella stima del cosiddetto SLAM *posterior*. L'approccio di tipo EKF-SLAM stima appunto la posa del robot e tutti i *landmark* della mappa con un filtro di Kalman esteso. Lo SLAM *posterior*, nel caso del filtro particellare, non viene modellato come una distribuzione di probabilità multivariata Gaussiana, parametrizzata da valor medio e covarianza; quindi il filtro particellare non fa alcuna ipotesi circa la natura della *pdf* nè tanto meno linearizza i modelli di moto e di misura come fa l'EKF. Il filtro particellare appartiene alla famiglia dei metodi sequenziali Monte-Carlo [5] che si fondano su punti di massa o *particelle*. Regioni di alta probabilità contengono una gran numero di particelle, mentre regioni di bassa probabilità contengono un numero minore di particelle. Questa *nube* di particelle modella qualsiasi distribuzione di probabilità con tanta precisione quanto in genere è il numero di particelle che si usano. Intuitivamente, un numero di particelle infinito restituirebbe l'esatta *pdf*, con tutte le sue asimmetrie, discontinuità e multimodalità. Inoltre il filtro particellare non altera in alcun modo la natura del modello di moto e di misura, ossia non effettua alcun tipo di linearizzazione.

Il filtro particellare trova davvero molte applicazioni nel campo della robotica, a cominciare dalla cosiddetta MCL, ossia *Monte-Carlo Localization* [36]. In questo caso la mappa è nota *a priori*. L'incertezza circa la posizione del robot nella mappa è quantificata da uno sciame di particelle, ognuna delle quali rappresenta una possibile posizione del robot. Per fare un

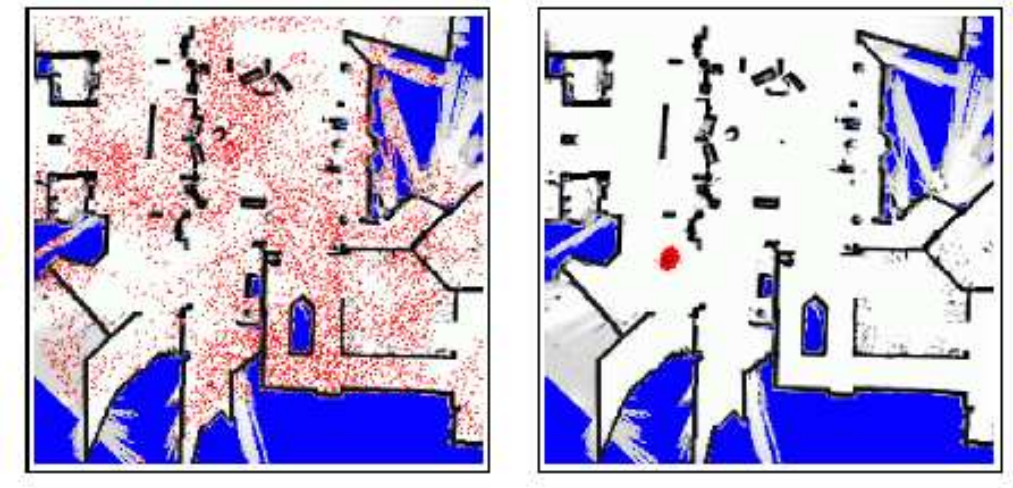


Figura 4.2: Problema dell'autolocalizzazione. All'inizio il robot, avendo inglobato poche osservazioni, può essere ovunque. Dopo che il filtro ha incorporato un buon numero di osservazioni il filtro converge ad un *posterior* unimodale.

esempio, se all'inizio, come è in genere, il robot non conosce la propria posizione, si vedrà che questa nube si dirada coprendo tutta l'area della mappa come si può vedere in Figura 4.2. Dopo aver inglobato un certo numero di informazioni attraverso le misure dei sensori, il *posterior* converge ad una distribuzione unimodale, ossia ad una distribuzione in cui vi è una sola moda, cioè un unico valore che è il più probabile tra tutti. Come prima accennato, la natura non strutturata del filtro, fa si che esso si possa confrontare con ogni tipo di dinamica, lineare e non.

4.3.2 L'architettura del FastSLAM

Il FastSLAM, sfruttando appieno la fattorizzazione dimostrata, si compone di $N+1$ filtri di bassa dimensionalità per la stima dell'evoluzione delle pose e per la locazione dei *landmark*. Una versione modificata del filtro particellare, ossia il filtro particellare Rao-Blackwellized [8, 10], costituisce l'oggetto che stima il *path posterior* e la posizione delle *feature* con N filtri di Kalman estesi. Quindi ogni EFK, di dimensione fissa (in particolare bi-dimensionale, in quanto vanno stimate le coordinate cartesiane del *landmark*), si occupa

di aggiornare la locazione di uno solo di essi. Il filtro particellare modella la posa del robot con un insieme di particelle (si veda paragrafo 4.3.1), ognuna delle quali rappresenta una possibile posa del robot. Dato che la stima dei *landmark* è condizionata sulla posa del robot, ogni particella possiederà il proprio insieme di EKFs per assolvere questo compito. Anche se con questo approccio il numero di EKFs è consistente, si ricorda la bassa e non-variabile dimensionalità di ogni filtro, i quali possono essere aggiornati velocemente e in tempo costante. Esistono in totale $N \cdot M$ filtri di Kalman, dove N, come noto, è il numero di *landmark* e M è il numero di particelle. La Figura 4.3 mostra la struttura del filtro particellare. Ogni particella ha la seguente struttura:

$$S_t^{[m]} = \left\langle s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, ..., \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\rangle \tag{4.15}$$

Con l'apice $[m]$ si indica l'indice della particella; $s^{t,[m]}$ rappresenta la stima del *path* dell'm-esima particella mentre con $\mu_{n,t}^{[m]}$ e con $\Sigma_{n,t}^{[m]}$ si indica il valor medio e la covarianza della distribuzione Gaussiana che modella la posizione dell'n-esimo *landmark* condizionato sul *path* $s^{t,[m]}$. Nel FastSLAM ci sono quindi in totale M strutture del tipo di quella presentata nella (4.15) che approssimano lo SLAM *posterior*. L'evoluzione dello SLAM *posterior* nel tempo

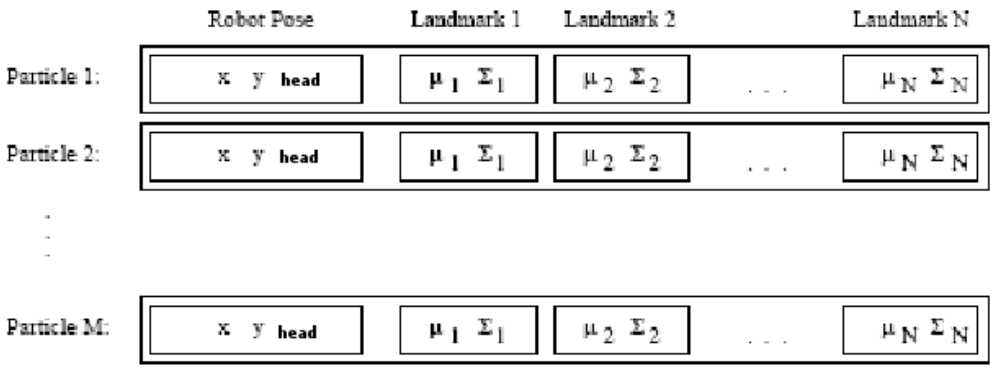


Figura 4.3: Struttura del filtro particellare.

viene approssimata dal set di particelle che si trasformano. Così propagare adeguatamente il *posterior* dal tempo $t-1$ al tempo t vuol dire generare un nuovo set di particelle tenendo conto dei controlli u_t, delle misure incorporate z_t con le corrispondenti associazioni di dati n_t. La Figura 4.4 mostra i

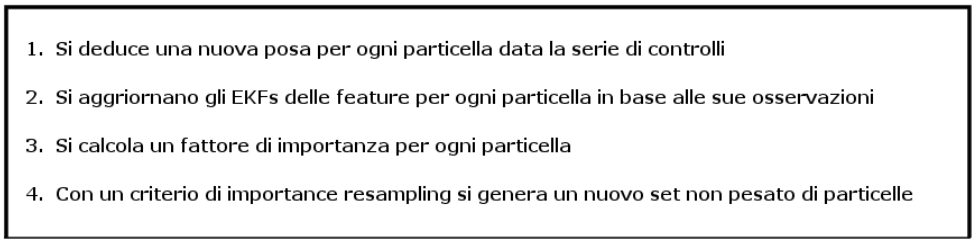


Figura 4.4: Algoritmo del FastSLAM.

passi fondamentali del filtro. Prima si deduce una nuova posa per ogni particella, tenendo conto della serie di controlli u_t e misure z_t al tempo t. Ogni posa della m-esima particella è aggiunta a $s^{t-1,[m]}$. Si procede poi all'update degli EKFs della m-esima particella; quest'aggiornamento per ogni EKF (e quindi per ogni *landmark*) viene fatto o no a seconda che il *landmark* sia stato osservato di nuovo. Dopo la fase di update delle *feature* le particelle vengono pesate con dei fattori di importanza (*importance weights*), proprio perchè esse approssimano il reale *path posterior*. Infine si genera un nuovo set S_t attraverso una procedura di *importance resampling*, ossia selezionando le particelle secondo il loro *importance weight*. Questo assicura che esse approssimino meglio il reale *path posterior*. I quattro passi dell'algoritmo vengono analizzati dettagliatamente nelle quattro sottosezioni che seguono.

4.3.3 Sampling di una nuova posa per particella

Come già accennato il *filtering* consiste nel generare un nuovo set di particelle S_t al tempo t dal precedente set S_{t-1} al tempo $t-1$ sfruttando i nuovi controlli u_t, cosi come le misurazioni z_t. Il primo passo del FastSLAM, sia nella sua versione 1.0 che nella versione 2.0, consiste nell'espandere il *path posterior* considerando le nuove grandezze quali u_t e z_t e n_t per generare un nuovo set di particelle. Il FastSLAM 1.0 campiona probabilisticamente dal *motion model* e prende quindi unicamente in considerazione i controlli al tempo t:

$$s_t^{[m]} \sim p(s_t | s_{t-1}^{[m]}, u_t) \tag{4.16}$$

Ossia si applica ad ogni particella il *motion model* e si considera il valore che ne deriva come la più probabile posa della particella al tempo t. Per l'm-esima

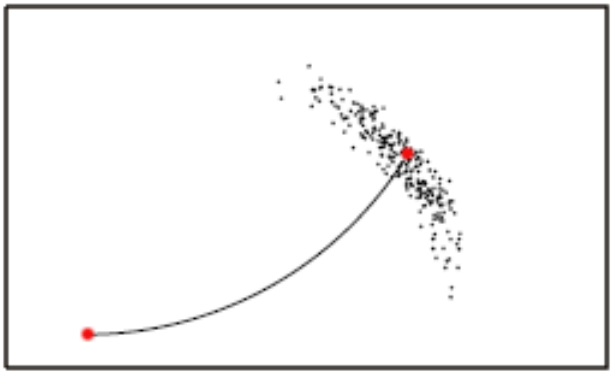


Figura 4.5: Campioni dedotti applicando il *motion model* al set iniziale di particelle.

particella questa ipotesi di posa è aggiunta al percorso $s^{t-1,[m]}$. Ipotizzando un numero di particelle infinito il set S_{t-1} è distribuito secondo il *posterior* al tempo $t-1$, ossia secondo $p(s^{t-1}|z^{t-1}, u^{t-1}, n^{t-1})$ (questa ipotesi è appunto asintoticamente corretta), le nuove particelle si distribuiscono secondo:

$$p(s^t|z^{t-1}, u^t, n^{t-1}) \tag{4.17}$$

la quale è nota come *proposal distribution*. Alla m-esima particella viene applicato il *motion model*, in genere non-lineare. La nuova posa per la particella viene calcolata in tempo costante indipendentemente dalla dimensione della mappa. Questo step di *sampling* è graficamente riportato nella Figura 4.5, la quale raffigura campioni (*samples*) dedotti dal *motion model*. In questa figura si può osservare come l'errore traslazionale sia relativamente basso (le particelle sono più o meno tutte sparse attorno alla posa reale), mentre l'errore sulla velocità angolare è più forte causando appunto l'apertura a ventaglio delle particelle.

Il FastSLAM 2.0 [27] invece tiene conto nel calcolo della *proposal distribution* delle misura z^t. Questo approccio risolve alcuni problemi che possono sorgere nel FastSLAM 1.0. In particolare, il FastSLAM 1.0 campiona le pose tenendo solamente in considerazione i controlli u_t e utilizzando le z_t come metrica per il *resampling* di queste pose. Questa metodica comincia a vacillare quando l'accuratezza dei controlli è molto bassa rispetto all'accu-

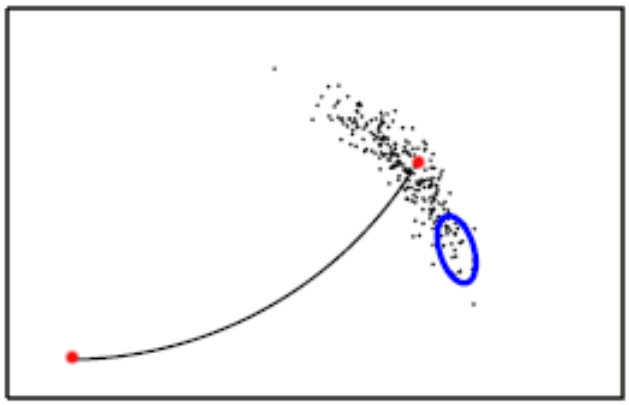


Figura 4.6: FastSLAM 1.0: samples dedotti dal *motion model* e samples (cerchiati dall'ellisse blu) i quali sopravvivono al processo di *resampling* che tiene in conto delle ultime misure.

ratezza delle misure. Questa situazione è illustrata in Figura 4.6 nella quale si pùo vedere che la *proposal* genera un largo spettro di campioni, mentre solo un piccolo sottoinsieme, quelli nell'ellisse in blu, sopravvive al processo di *resampling* dopo aver incorporato le z_t in quanto ha un alta probabilità. Sembra quindi intuitivo tenere in considerazione le attuali acquisizioni sensoriali per generare la *proposal*. Questo è ciò che fa il FastSLAM 2.0, il quale sarà in grado di espandere in maniera più consona il *path posterior*, rendendo tuttavia più difficile la sua derivazione matematica e la sua implementazione.

La posa $s_t^{[m]}$ viene quindi calcolata dal *posterior*:

$$s_t^{[m]} \sim p(s_t | s^{t-1,[m]}, u^t, z^t, n^t) \tag{4.18}$$

Come si può vedere la (4.18) differisce dalla (4.16) proprio perchè $s_t^{[m]}$ è condizionato anche dalle misure z^t, insieme alle corrispondenze n_t. Si ricorda che $s^{t-1,[m]}$ è il *path* dell'm-esima particella fino al tempo $t-1$. Il meccanismo per campionare dalla (4.18) richiede qualche analisi. Si riscrive, prima di tutto, la (4.18) in termini di distribuzioni note come i modelli di misura e di moto e la distribuzione Gaussiana che rappresenta la stima delle *feature*

dell'm-esima particella:

$$\begin{aligned} & p(s_t|s^{t-1,[m]}, u^t, z^t, n^t) \\ & \overset{Bayes}{=} \frac{p(z_t|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)p(s_t|s^{t-1,[m]}, u^t, z^{t-1}, n^t)}{p(z_t|s^{t-1,[m]}, u^t, z^{t-1}, n^t)} \\ & = \eta^{[m]} p(z_t|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)p(s_t|s^{t-1,[m]}, u^t, z^{t-1}, n^t) \end{aligned} \tag{4.19}$$

La posa del robot s_t nel secondo termine della (4.19) dipende solo dalle pose precedenti $s^{t-1,[m]}$ e dal controllo corrente u_t: le restanti variabili condizionanti possono essere eliminate e la (4.19) diventa:

$$\overset{Markov}{=} \eta^{[m]} p(z_t|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)p(s_t|s^{t-1,[m]}, u^t) \tag{4.20}$$

Si utilizza poi il teorema della Probabilità Totale per condizionare il primo termine della precedente sul *landmark* attualmente osservato θn_t. La (4.20) diviene:

$$\begin{aligned} & = \eta^{[m]} \int p(z_t|\theta n_t, s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)p(\theta n_t|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t)d\theta n_t \\ & p(s_t|s^{t-1,[m]}, u^t) \end{aligned} \tag{4.21}$$

e si applica a quest'ultima la regola di Markov:

$$\begin{aligned} & s_t^{[m]} \sim p(s^t|s^{t-1}, z^t, u^t, n^t) \overset{Markov}{=} \eta^{[m]} \\ & \int \underbrace{p(z_t|\vartheta_{n_t}, s_t, n_t)}_{\sim\mathcal{N}(z_t|g(\vartheta_{n_t}, s_t), R_t)} \underbrace{p(\vartheta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1})d\vartheta_{n_t}}_{\sim\mathcal{N}(\vartheta_{n_t}; \mu^{[m]}_{n_t,t-1}, \Sigma^{[m]}_{n_t,t-1})} \underbrace{p(s_t|s^{[m]}_{t-1}, u_t)}_{\sim\mathcal{N}(s_t|h(s^{[m]}_{t-1}, u_t), P'_t)} \end{aligned} \tag{4.22}$$

[2]La (4.22) mostra che si può esprimere la distribuzione da cui campionare in termini della convoluzione di due distribuzioni Gaussiane moltiplicate per una terza distribuzione. Tuttavia nella maggior parte dei casi questa distribuzione non possiede una soluzione in forma chiusa. Infatti il problema sta nella non-linearità della funzione g, ossia della dinamica di osservazione: se la funzione g fosse lineare, allora la (4.22) sarebbe una distribuzione Gaussiana.

[2] $P_t^{'}$ rappresenta l'attuale incertezza circa la posa della particella ed è diverso dalla semplice matrice di covarianza del rumore di moto.

In generale però questo integrale non ha soluzione in forma chiusa, rendendo difficoltoso, se non impossibile, il *sampling* dalla (4.18). Sembra naturale allora tentare di espandere la funzione g con un'espansione in serie di Taylor troncata al prim'ordine:

$$g(\theta n_t, s_t) \approx \hat{z}_t^{[m]} + G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) + G_s(s_t - \hat{s}_t^{[m]}) \tag{4.23}$$

dove $\hat{s}_t^{[m]}$ è la posa predetta della m-esima particella al tempo t e $\hat{z}_t^{[m]}$ è la predetta osservazione sempre al tempo t. Queste quantità vengono calcolate come:

$$\hat{s}_t^{[m]} = h(s_{t-1}^{[m]}) \tag{4.24}$$

$$\hat{z}_t^{[m]} = g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]}) \tag{4.25}$$

Le matrici G_θ e G_s sono i Jacobiani di g, ossia le derivate di g rispetto a θ_{n_t} e s_t rispettivamente, valutate rispetto agli attuali valori dei loro argomenti. In formule:

$$G_\theta = \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} \tag{4.26}$$

$$G_s = \nabla_{s_t} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} \tag{4.27}$$

Con queste approssimazioni, la distribuzione (4.22) è approssimabile da una Gaussiana parametrizzata dai seguenti valor medio e covarianza:

$$\Sigma_{s_t}^{[m]} = \left[G_s^{\mathsf{T}} H_t^{[m]-1} G_s + P_t'^{-1}\right]^{-1} \tag{4.28}$$

$$\mu_{s_t}^{[m]} = \Sigma_{s_t}^m G_s^{\mathsf{T}} H_t^{[m]-1}(z_t - \hat{z}_t^{[m]}) + \hat{s}_t^{[m]} \tag{4.29}$$

con la matrice $H_t^{[m]}$ definita come segue:

$$H_t^{[m]} = R_t + G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^{\mathsf{T}} \tag{4.30}$$

Dimostriamo come vengono ottenute la (4.28) e la (4.29). Data la linearizzazione esposta, l'integrale della (4.22) è risolubile in forma chiusa attraverso il Teorema della Convoluzione. In particolare è pari a:

$$\mathcal{N}(z_t; \hat{z}_t^{[m]} + G_s s_t - G_s \hat{s}_t^{[m]}, H_t^{[m]}) \tag{4.31}$$

Ora la distribuzione di *sampling* nel suo complesso è data dal prodotto della precedente normale distribuzione per il termine più a destra della (4.22), il quale è anch'esso normalmente distribuito $\mathcal{N}(s_t; s_{t-1}^{[m]}, P_t')$. Eseguendo il prodotto delle due distribuzioni attraverso alcuni semplici passaggi si ottiene per la *proposal distribution*:

$$p(s_t|s^{t-1,[m]}, u^t, z^t, n^t) = \eta \; exp\Big\{ - y_t^{[m]} \Big\} \tag{4.32}$$

con

$$y_t^{[m]} = \frac{1}{2}\Big[(z_t - z_t^{[m]} - G_s s_t + G_s \hat{s}_t^{[m]})^\mathsf{T} H_t^{[m]-1} (z_t - z_t^{[m]} - G_s s_t + G_s \hat{s}_t^{[m]})\Big]$$
$$(s_t - \hat{s}_t^{[m]})^\mathsf{T} P_t^{'[m]} (s_t - \hat{s}_t^{[m]}) \tag{4.33}$$

L'espressione è quadratica nella variabile s_t, quindi $p(s_t|s^{t-1,[m]}, u^t, z^t, n^t)$ è Gaussiana. Il valor medio e la covarianza sono equivalenti al minimo di $y_t^{[m]}$ e dalla sua curvatura. Questo equivale a calcolare la derivata prima e seconda di $y_t^{[m]}$ rispetto a s_t:

$$\begin{aligned}\frac{\partial y_t^{[m]}}{\partial s_t} &= -G_s^\mathsf{T} H_t^{[m]-1}(z_t - z_t^{[m]} - G_s s_t + G_s \hat{s}_t^{[m]}) + P_t^{'-1}(s_t - \hat{s}_t^{[m]}) \\ &= (G_s^\mathsf{T} H_t^{[m]-1} G_s + P_t^{'-1}) s_t - G_s^\mathsf{T} H_t^{[m]-1} z_t - z_t^{[m]} + G_s \hat{s}_t^{[m]}) - P_t^{'-1} \hat{s}_t^{[m]}\end{aligned} \tag{4.34}$$

$$\frac{\partial^2 y_t^{[m]}}{\partial s_t^2} = G_s^\mathsf{T} H_t^{[m]-1} G_s + P_t^{'-1} \tag{4.35}$$

La (4.28) si ottiene quindi invertendo la derivata seconda:

$$\Sigma_{s_t}^{[m]} = \Big[G_s^\mathsf{T} H_t^{[m]-1} G_s + P_t^{'-1} \Big]^{-1} \tag{4.36}$$

mentre per il valor medio si pone uguale a zero la derivata prima, cioè:

$$\begin{aligned}\mu_{s_t}^{[m]} &= \Sigma_{s_t}^{[m]} \Big[G_s^\mathsf{T} H_t^{[m]-1}(z_t - z_t^{[m]} + G_s \hat{s}_t^{[m]}) + P_t^{'-1} \hat{s}_t^{[m]} \Big] \\ &= \Sigma_{s_t}^{[m]} G_s^\mathsf{T} H_t^{[m]-1}(z_t - z_t^{[m]}) + \Sigma_{s_t}^{[m]} \Big[G_s^\mathsf{T} H_t^{[m]-1} G_s + P_t^{'-1} \Big] \hat{s}_t^{[m]} \\ &= \Sigma_{s_t}^{[m]} G_s^\mathsf{T} H_t^{[m]-1}(z_t - z_t^{[m]}) + \hat{s}_t^{[m]}\end{aligned} \tag{4.37}$$

Questa Gaussiana è l'approssimazione della *proposal distribution* nel FastSLAM 2.0. Questa *proposal* come si è visto è notevolmente più complessa

rispetto a quella del FastSLAM 1.0, ma permette di diminuire drasticamente il numero di particelle necessarie pad ottenere un determinato livello di accuratezza; oppure, a parità di particelle, la *proposal* è estremamente più consistente rispetto alle successive osservazioni.

4.3.4 Update della stima dei landmark

Il FastSLAM aggiorna il *posterior* dei *landmark* condizionandolo sul set di misure z_t e sulla posa campionata $s_t^{[m]}$. L'aggiornamento di $p(\theta_n|s^t, z^t, u^t, n^t)$ viene effettuato attraverso dei filtri di Kalman di bassa dimensionalità. Poichè questa stima è condizionata sulla posa del robot, gli N filtri di Kalman che stimano gli N *landmark* della mappa sono relativi ad un'unica particella. Quindi il *posterior* totale, cioè quello sul *path* e sulla locazione dei landmark è dato dal set di campioni:

$$S_t^{[m]} = \left\langle s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, ..., \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\rangle \tag{4.38}$$

La stima al tempo $t-1$ è rappresentata dal valor medio $\mu_{n,t-1}^{[m]}$ e dalla covarianza $\Sigma_{n,t-1}^{[m]}$, mentre l'aggiornamento della stima è rappresentato dai nuovi valor medio $\mu_{n,t}^{[m]}$ e covarianza $\Sigma_{n,t}^{[m]}$. Nello SLAM planare si ricorda che $\mu_{n,t}^{[m]}$ è il vettore delle coordinate della *feature*, mentre $\Sigma_{n,t}^{[m]}$ è una matrice 2×2. Il *posterior* sull'n-esimo *landmark* θ_n è facilmente ottenibile. L'update dipende dall'informazione ottenuta dall'associazione di dati, ossia dal fatto che sia verificato o no che $n = n_t$, cioè che al tempo t sia stato effettivamente osservato il *landmark* n. Se questo è vero, si sviluppa il *posterior* attraverso la regola di Bayes ottenendo:

$$p(\theta_{n_t}|s^t, z^t, u^t, n^t) \overset{Bayes}{\propto} p(z_t|\theta_{n_t}, s^t, z^{t-1}, u^t, n^t) p(\theta_{n_t}|s^t, z^{t-1}, u^t, n^t) \tag{4.39}$$

Si applica poi la regola di Markov per semplificare i termini della precedente. Infatti z_t dipende solo da θ_{n_t} s^t e n^t, mentre θ_{n_t} è condizionato solo da s^t u^t e n^t senza l'osservazione z_t:

$$p(\theta_{n_t}|s^t, z^t, u^t, n^t) \overset{Markov}{=} \eta p(z_t|\theta_{n_t}, s^t, n^t) p(\theta_{n_t}|s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \tag{4.40}$$

Nel caso il cui $n \neq n_t$, il *posterior* rimane inalterato:

$$p(\theta_{n \neq n_t}|s^t, z^t, u^t, n^t) = p(\theta_{n \neq n_t}|s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \tag{4.41}$$

Il FastSLAM implementa l'aggiornamento presente nell'equazione (4.40) attraverso EKF [38]. Il filtro sfrutta una linearizzazione del *measurement model* $p(z_t|s_t, \theta_{n_t}, n^t)$. In questo il FastSLAM è simile all'EKF-SLAM, nel senso che entrambi approssimano la dinamica delle misurazioni con una funzione lineare Gaussiana. Si noti che approssimando il modello di osservazione con una funzione lineare Gaussiana, la distribuzione $p(\theta_{n_t}|z^t, s^t, u^t, n^t)$ è esattamente Gaussiana anche se il *motion model* non è lineare. Questa è una diretta conseguenza del fatto che si utilizza un'approccio di tipo particellare per la stima della posa del robot.

L'uso che il FastSLAM e l'EKF-SLAM fanno del filtro di Kalman è differente. Infatti il primo utilizza il filtro per aggiornare Gaussiane bidimensionali (per i due parametri della locazione dei *landmark*). Nel secondo la Gaussiana da aggiornare ha dimensione $2N+3$ (N *landmark* e tre parametri per la posa del robot). Quindi se il FastSLAM aggiorna la stima in tempo costante, per l'EKF-SLAM il tempo di aggiornamento è proporzionale a N^2.

Prima di analizzare come venga aggiornata la stima dei *landmark*, analizziamo la trigonometria delle acquisizioni sensoriali e ricaviamo per esteso la forma della g per lo SLAM planare. In tali circostanze il robot misura *range* (distanza) r e *bearing* (orientamento) ϕ, come mostrato in Figura 4.7. Questi dati, ossia la distanza e l'orientamento del bersaglio sono riferiti al sistema di assi descritto dalla terna $\{s_{t,x}, s_{t,y}, s_{t,\alpha}\}$, ossia centrato in $(s_{t,x}, s_{t,y})$, cioè nell'attuale posa del robot rispetto ad un sistema di riferimento immobile, e orientato come $s_{t,\alpha}$. Le correnti posizioni dei *landmark* vengono descritte invece da $(\theta_{n_t,x}, \theta_{n_t,y})$. La funzione $g(\theta_{n_t}, s_t)$ sarà:

$$g(\theta_{n_t}, s_t) = \begin{bmatrix} r(\theta_{n_t}, s_t) \\ \phi(\theta_{n_t}, s_t) \end{bmatrix} = \begin{bmatrix} \sqrt{(\theta_{n_t,x} - s_{t,x})^2 + ((\theta_{n_t,y} - s_{t,y})^2} \\ \tan^{-1} \frac{(\theta_{n_t,y} - s_{t,y})}{(\theta_{n_t,x} - s_{t,x})} \end{bmatrix} \tag{4.42}$$

Riscriviamo ora la distribuzione $p(z_t|s_t, \theta_{n_t}, n^t)$ considerando l'indice della particella m e visualizzando le ipotesi circa la normalità delle distribuzioni del modello di misura:

$$p(\theta_{n_t}|s^t, z^t, u^t, n^t) = \eta \underbrace{p(z_t|\vartheta_{n_t}, s_t, n_t)}_{\sim\mathcal{N}(z_t|g(\vartheta_{n_t}, s_t), R_t)} \underbrace{p(\vartheta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1})}_{\sim\mathcal{N}(\vartheta_{n_t}; \mu^{[m]}_{n_t,t-1}, \Sigma^{[m]}_{n_t,t-1})} \tag{4.43}$$

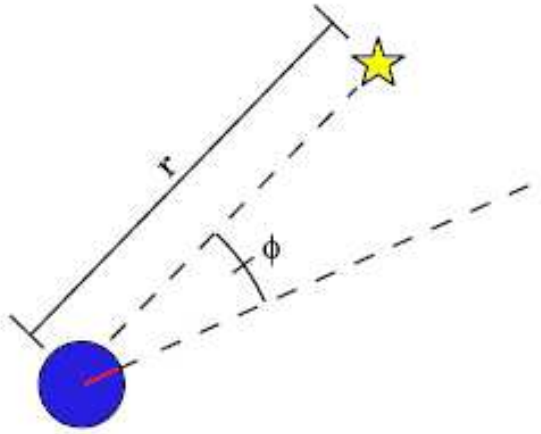


Figura 4.7: I sensori effettuano misure di distanza e orientamento del *landmark*.

La non-linearità di g, come nella (4.22), causa la non-gaussianità della distribuzione che contrasta con l'assunzione del FastSLAM circa la normalità della distribuzione della stima delle *feature*. Quindi il modello di misurazione non-lineare $g(\theta_{n_t}, s_t)$ viene linearizzato con uno sviluppo in serie di Taylor troncato al primo ordine:

$$\begin{aligned} \hat{z}_t^{[m]} &= g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]}) \\ G_{\theta_{n_t}} &= \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} \\ g(\theta_{n_t}, s_t) &\approx \hat{z}_t^{[m]} + G_{\theta_{n_t}}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \end{aligned} \tag{4.44}$$

[3]utilizzando questa approssimazione il primo termine a destra della (4.43) è effettivamente pari a:

$$p(z_t|\vartheta_{n_t}, s_t^{[m]}, n_t) \sim \mathcal{N}(z_t; \hat{z}_t^{[m]} + G_{\theta_{n_t}}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}), R_t) \tag{4.45}$$

Analogamente il secondo termine a destra della (4.43) può essere approssimato come:

$$p(\vartheta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1}) \sim \mathcal{N}(\vartheta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]}) \tag{4.46}$$

[3]Si noti come nella linearizzazione sia possibile omettere il termine più a destra della (4.23) poichè s_t non è una variabile, in quanto la stima dei *landmark* è condizionata ad una sola posa del robot.

Questa approssimazione rende la (4.43) Gaussiana nella variabile ϑ_{n_t}. Eseguendo il prodotto delle Gaussiane nella (4.43) si ottiene:

$$\begin{aligned} & p(\theta_n|s^t, z^t, u^t, n^t) \\ & \quad = \eta \exp\Big\{ -\frac{1}{2}(z_t - \hat{z}_t^{[m]} - G_{\theta_{n_t}}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}))^\mathsf{T} R_t^{-1} \\ & \quad (z_t - \hat{z}_t^{[m]} - G_{\theta_{n_t}}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})) - \frac{1}{2}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]})^\mathsf{T} \Sigma_{n_t,t-1}^{[m]-1}(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \Big\} \end{aligned} \tag{4.47}$$

Lo Jacobiano $G_{\theta_{n_t}}$ è pari a:

$$G_{\theta_{n_t}} = \begin{bmatrix} \frac{\theta_{n_t,x} - s_{t,x}}{sqrtq} & \frac{\theta_{n_t,y} - s_{t,y}}{sqrtq} \\ -\frac{\theta_{n_t,x} - s_{t,x}}{sqrtq} & \frac{\theta_{n_t,x} - s_{t,x}}{sqrtq} \end{bmatrix} \tag{4.48}$$

con $q = (\theta_{n_t,x} - s_{t,x})^2 + ((\theta_{n_t,y} - s_{t,y})^2$

Il nuovo valor medio e la nuova covarianza possono essere ottenuti con le equazioni del filtro di Kalman [20]:

$$\mathcal{K}_t^{[m]} = \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}} Q_t^{[m]-1} \tag{4.49}$$

$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + \mathcal{K}_t^{[m]}(z_t - \hat{z}_t^{[m]}) \tag{4.50}$$

$$\Sigma_{n_t,t}^{[m]} = (I - \mathcal{K}_t^{[m]} G_{\theta_{n_t}}) \Sigma_{n_t,t-1}^{[m]} \tag{4.51}$$

4.3.5 L'importance factor della particella

Le particelle generate non modellano ancora in maniera ottimale il desiderato *posterior*. Il problema è la costante normalizzante $\eta^{[m]}$ in (4.22) che è differente per diverse particelle. Questa differenza viene compensata grazie ad un procedimento chiamato *importance sampling*, una tecnica per ottenere campioni da funzioni per le quali non esiste una diretta procedura di *sampling* [3]. Invece di fare *sampling* direttamente dalla funzione, alla quale ci si riferisce come *target distribution*, si campiona da una distribuzione più semplice, la *proposal distribution*. Ad ogni campione viene assegnato un peso, pari al rapporto della target sulla *proposal* calcolato in quel particolare punto. Si genera poi un nuovo set non pesato di particelle scegliendo una o l'altra particella proporzionalmente al suo peso. Questo procedimento altro non è che un'istanza del *Sampling Importance Resampling* (SIR) di Rubin [33]. Un

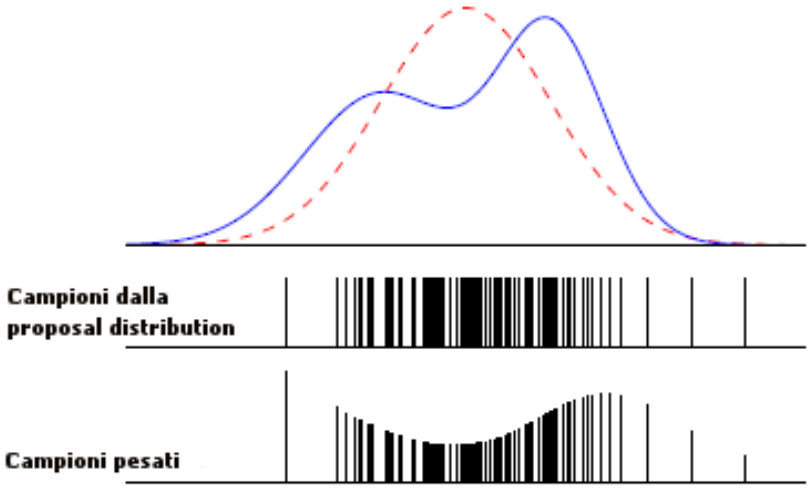


Figura 4.8: I campioni non possono essere ottenuti direttamente dalla target distribution (linea blu continua). Si campiona da una distribuzione più semplice, in questo caso una Gaussiana (linea tratteggiata in rosso). In basso vengono riportati i campioni pesati,con un lunghezza proporzionale al loro importance factor.

esempio di questa tecnica viene riportato in Figura 4.8. Invece di campionare direttamente dalla *target distribution* (disegnato con la linea continua in blu), si campiona una Gaussiana (linea tratteggiata in rosso). Nelle regioni in cui la target è più grande della *proposal*, i campioni riceveranno un peso maggiore. Come conseguenza i campioni in questa regione verrano scelti con più frequenza. Viceversa, nelle regioni in cui la target è minore della *proposal*, i pesi riceveranno un peso minore e verranno quindi scelti con meno probabilità. In basso vengono riportati i campioni con una lunghezza proporzionale al loro peso e si può osservare come con buona approssimazione modellino la *target distribution*. Asintoticamente, ossia campionando infinitamente la *proposal* e ottenendo quindi da essa infiniti *samples*, questa tecnica permette di riprodurre fedelmente la *target distribution*.

Come già detto, l'*importance factor* è pari al quoziente:

$$w_t^{[m]} = \frac{\text{target distribution}}{\text{proposal distribution}} \tag{4.52}$$

La *target distribution* che noi vogliamo le nostre particelle catturino è data dal *path posterior* $p(s^{t,[m]}|z_t, u_t, n_t)$. Nell'ipotesi, asintoticamente corretta, che i percorsi in $s^{t-1,[m]}$ vengano generati secondo la *target distribution* al

passo precedente, $p(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1})$, la *proposal distribution* è data dal prodotto:

$$p(s^{t,[m]}|z_t, u_t, n_t) = p(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1})p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t) \quad (4.53)$$

Il secondo termine a destra della (4.53) è il campionamento della distribuzione che cattura la posa del robot (4.22). Il peso della particella è allora ottenuto come:

$$w_t^{[m]} = \frac{p(s^{t,[m]}|z_t, u_t, n_t)}{p(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1})p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t)} \quad (4.54)$$

Il numeratore della precedente viene sviluppato con il Teorema della Probabilità Condizionale:

$$= \frac{p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t)p(s^{t-1,[m]}|z_t, u_t, n_t)}{p(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1})p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t)} \quad (4.55)$$

$$= \frac{p(s^{t-1,[m]}|z_t, u_t, n_t)}{p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t)} \quad (4.56)$$

Il numeratore della (4.55) viene sviluppato con la regola di Bayes:

$$\overset{Bayes}{=} \eta \frac{p(z_t|s^{t-1,[m]}, z_{t-1}, u_t, n_t)p(s^{t-1,[m]}|z^{t-1}, u^t, n^t)}{p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t)} \quad (4.57)$$

Nel secondo termine del numeratore u^t e n^t possono essere eliminati in quanto non condizionano in alcun modo $s^{t-1,[m]}$:

$$\begin{aligned} &\overset{Markov}{=} \eta \frac{p(z_t|s^{t-1,[m]}, z_{t-1}, u_t, n_t)p(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1})}{p(s^{t,[m]}|s^{t-1,[m]}, z_t, u_t, n_t)} \\ &= \eta p(z_t|s^{t-1,[m]}, u^t, z^{t-1}, n^t) \end{aligned} \quad (4.58)$$

Si può notare come questa espressione sia l'inversa della costante normalizzante $\eta^{[m]}$ nella (4.22). Attraverso altre trasformazioni, in particolare applicando due volte il Teorema della Probabilità Condizionale per condizionare la (4.58) su s_t e su θ_{n_t} e due volte la regola di Markov, otteniamo una nuova

forma per $w_t^{[m]}$:

$$\begin{aligned} w_t^{[m]} = &\ \eta \int p(z_t|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t) p(s_t|s^{t-1,[m]}, u^t, z^{t-1}, n^t) ds_t \\ \overset{Markov}{=} &\ \eta \int p(z_t|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t) p(s_t|s^{t-1,[m]}, u^t) ds_t \\ = &\ \eta \iint p(z_t|\theta_{n_t}, s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t) p(\theta_{n_t}|s_t, s^{t-1,[m]}, u^t, z^{t-1}, n^t) d\theta_{n_t} \\ &\ p(s_t|s^{t-1,[m]}, u^t) ds_t \\ \overset{Markov}{=} &\ \eta \iint \underbrace{p(z_t|\vartheta_{n_t}, s_t, n_t)}_{\sim \mathcal{N}(z_t|g(\vartheta_{n_t}, s_t), R_t)} \underbrace{p(\vartheta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1}) d\vartheta_{n_t}}_{\sim \mathcal{N}(\vartheta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]})} \underbrace{p(s_t|s_{t-1}^{[m]}, u_t)}_{\sim \mathcal{N}(s_t|h(\hat{s}_{t-1}^{[m]}, u_t), P_t')} \end{aligned} \tag{4.59}$$

Si può dimostrare che questa espressione può essere approssimata da una Gaussiana a patto di linearizzare la dinamica delle misura g. Applicando quindi il teorema della convoluzione si ottiene che $w_t^{[m]}$ distribuito normalmente con valor medio $\hat{z}_t$ e matrice di covarianza pari a:

$$L_t^{[m]} = G_s P_t' G_s + G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta + R_t \tag{4.60}$$

Montemerlo è riuscito ad ottenere una forma matriciale per l'*importance weight*:

$$w_t^{[m]} = |2\pi L_t^{[m]}|^{-\frac{1}{2}} \exp\Big\{ -\frac{1}{2}(z_t - \hat{z}_t)^\intercal L_t^{[m]-1}(z_t - \hat{z}_t) \Big\} \tag{4.61}$$

4.3.6 L'Importance Resampling

L'ultimo step del FastSLAM è l'*importance resampling*. Una volta che al set provvisorio di particelle sono stati assegnati i pesi, si genera un nuovo set di particelle. Questa esigenza nasce dal fatto che il set temporaneo non è distribuito secondo il desiderato *posterior*; questo discostamento è modellato per ogni particella dal suo peso e in base a questo peso una particella entra o no a far parte del nuovo set S_t. Il *resampling* avviene con sostituzione, ossia se una particella non viene selezionata essa viene rimpiazzata da una con un peso maggiore. Esistono svariate tecniche per estrarre S_t dal set temporaneo. L'algoritmo utilizzato in qusta tesi è il Madow's *Systematic Sampling*

Algorithm [21], in quanto la sua implementazione è abbastanza semplice e l'algoritmo è molto efficace. Nella sua implementazione originale lo step di *resampling* richiede un tempo proporzionale al numero di *landmark*, ossia $\mathcal{O}(N)$ in quanto ogni particella deve essere copiata nel nuovo set e la lunghezza di ogni particella è pari a N. Forme più complesse di implementazione permettono di ridurre il costo computazionale a $\mathcal{O}(\log(N))$, evitando copie inutili, in quanto ad ogni time step solo un piccolo numero di *landmark* viene osservato. Nel capitolo successivo si tratterà più nel dattaglio il *resampling*, per esempio come e quando questo processo è eseguito. Infatti se i primi tre stage del FastSLAM vengono lanciati ad ogni time step dell'algoritmo, non vale la stessa cosa per il *resampling*.

4.4 Convergenza del FastSLAM per LG-SLAM

La potenza del FastSLAM è tale che l'algoritmo converge anche con una sola particella. Nel filtro particellare standard, lo step di *resampling* è l'unica fase in cui i campioni vengono rigenerati in accordo alle osservazioni. Il *resampling* diventa inutile in questo caso poichè, essendovi un unica particella, essa non può essere rimpiazzata e quindi il filtro divergerà. Ciò non accade nel FastSLAM 2.0, il quale costruisce la *proposal distribution* anche sulla base delle osservazioni e quindi minimizza l'errore della mappa proponendo una nuova posa in accordo alle osservazioni; è quindi in grado di convergere anche con una sola particella.

In questa sezione si dimostra la convergenza del FastSLAM con una singola particella appunto per lo speciale caso di SLAM lineare e Gaussiano (LG-SLAM). La dimostrazione della convergenza con una unica particella implica la convergenza con M particelle. Questa dimostrazione ha due importantissime conseguenze: la prova della convergenza di questo approccio allo SLAM con tempo computazionale lineare e il superamento dell'idea che mantenere l'intera matrice di correlazione tra i *landmark* è indispensabile per la convergenza. Per la dimostrazione si assume noto il mapping tra le osservazioni e i *landmark*. Nell'LG-SLAM il *motion model* e il *measurement*

model sono del tipo:

$$g(\theta_{n_t}, s_t) = \theta n_t - s_t \tag{4.62}$$

$$h(u_t, s_t) = u_t + s_{t-1} = \tag{4.63}$$

Un tipico LG-SLAM pùo essere quello di un robot che si muove in uno spazio cartesiano equipaggiato con una bussola senza rumore e sensori che misurano la distanza delle *feature* sugli assi coordinati. La formulazione di convergenza è la seguente:

Teorema 1. *Il FastSLAM lineare-gaussiano converge alla mappa corretta con M=1 particelle se tutte le feature vengono osservate infinite volte e se è nota a priori la posizione di una feature.*

Se non si conosce in anticipo la posizione di una *feature*e, la mappa sarà corretta in senso relativo, ossia nel senso che un offset costante andrà sommato a tutte le *feature* della mappa.

4.4.1 Prova di convergenza del FastSLAM per LG-SLAM

La prova di convergenza viene effettuata attraverso una serie di lemmi. Prima di tutto formuliamo per chiarezza gli elementi del FastSLAM 2.0 per LG-SLAM. Il *motion model* e il *measurement model* già sono definiti dalle (4.63) e (4.62). Inoltre:

$$\hat{s}_t^{[m]} = h(s_{t-1}^{[m]}, u_t) = s_{t-1}^{[m]} + u_t \tag{4.64}$$

$$\hat{z}_t^{[m]} = g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]}) \tag{4.65}$$

$$G_\theta = \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} = I \tag{4.66}$$

$$G_s = \nabla_{s_t} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}} = -I \tag{4.67}$$

$$H_t^{[m]} = R_t + G_\theta \Sigma_{n_t,t-1}^{[m]} G_\theta^\mathsf{T} = R_t + \Sigma_{n_t,t-1}^{[m]} \tag{4.68}$$

Le (4.29) e (4.28) per il *sampling* della posa di ogni particella al tempo t viene riformulata come segue:

$$\begin{aligned}\mu_{s_t}^{[m]} &= \Sigma_{s_t}^{m} G_s^{\mathsf{T}} H_t^{[m]-1}(z_t - \hat{z}_t^{[m]}) + \hat{s}_t^{[m]} \\ &= -\Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\hat{z}_t^{[m]} - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) + s_{t-1}^{[m]} + u_t \quad (4.69)\end{aligned}$$

$$\Sigma_{s_t}^{[m]} = \left[G_s^{\mathsf{T}} H_t^{[m]-1} G_s + P_t'^{-1}\right]^{-1} = \left[(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1} + P_t'^{-1}\right]^{-1} \quad (4.70)$$

$$s_t^{[m]} = \sim \mathcal{N}(s_t^{[m]}; \mu_{s_t}^{[m]}, \Sigma_{s_t}^{[m]}) \quad (4.71)$$

Analogamente le equazioni di aggiornamento delle posizioni dei *landmark* vengono riscritte come:

$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\hat{z}_t^{[m]} - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) \quad (4.72)$$

$$\Sigma_{n_t,t}^{[m]} = (I - \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1})\Sigma_{n_t,t-1}^{[m]} \quad (4.73)$$

Si può tranquillamente ignorare l'equazione per il calcolo dei pesi, in quanto non ha senso pesare un'unica particella. Per la dimostrazione del teorema è necessario invece introdurre le due variabili di errore:

$$\alpha_t^{[m]} = s_t^{[m]} - s_t \quad (4.74)$$

$$\beta_{n,t}^{[m]} = \mu_{n,t}^{[m]} - \theta_n \quad (4.75)$$

Queste variabili misurano l'errore assoluto sulla stima della singola particella della posa del robot e delle *feature*. Nel seguito della dimostrazione si indicherà come *anchoring feature* la *feature* nota a priori. Il primo lemma descrive la correlazione tra gli errori sulla mappa β con quelli sulla posa del robot α.

Lemma 1. *Se l'errore $\beta_{n_t,t}^{[m]}$ della feature z_t osservata al tempo t è più piccolo dell'errore sulla posa $\alpha_t^{[m]}$, allora $\alpha_t^{[m]}$ diminuisce nel suo valore atteso come risultato di questa misurazione. Se viceversa l'errore $\beta_{n_t,t}^{[m]}$ è maggiore dell'errore sulla posa, quest'ultimo può crescere anche se nel suo valore atteso non eccederà mai $\beta_{n_t,t}^{[m]}$.*

Prova del Lemma 1. Il valore atteso dell'errore sulla posa campionata del robot al tempo t è dato da:

$$E[\alpha_t^{[m]}] = E[s_t^{[m]} - s_t] = E[s_t^{[m]}] - E[s_t] \quad (4.76)$$

Il primo termine è ottenuto dalla (4.69) mentre per il secondo si utilizza il *motion model* della (4.63)

$$\begin{aligned} E[\alpha_t^{[m]}] &= E[-\Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\hat{z}_t^{[m]} - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t)] + E[s_{t-1}^{[m]} + u_t] \\ &= -\Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(E[\hat{z}_t^{[m]}] - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) + \underbrace{s_{t-1}^{[m]} + s_{t-1}}_{\alpha_{t-1}^{[m]}} \end{aligned} \tag{4.77}$$

L'ultima trasformazione sfrutta la linearità dell'operazione di calcolo del valore atteso. Ora per l'LG-SLAM $Ez_t = \theta_{n_t} - E[s_t] = \theta_{n_t} - u_t - s_{t-1}$. L'espressione tra parentesi nella (4.77) diventa:

$$\begin{aligned} E[\hat{z}_t^{[m]}] - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t &= \theta_{n_t} - u_t - s_{t-1} - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) \\ &= s_{t-1}^{[m]} - s_{t-1} + \theta_{n_t} - \mu_{n_t,t-1}^{[m]} \\ &= \alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]} \end{aligned} \tag{4.78}$$

Sostituendo quest'ultima nella (4.77) e successivamente sustituendo $\Sigma_{s_t}^{[m]}$ secondo la (4.69) si ottiene:

$$\begin{aligned} E[\alpha_t^{[m]}] &= \alpha_{t-1}^{[m]} + \Sigma_{s_t}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\beta_{n_t,t-1}^{[m]} - \alpha_{t-1}^{[m]}) \\ &= \alpha_{t-1}^{[m]}\left[(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1} + P_t'^{-1}\right]^{-1}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\beta_{n_t,t-1}^{[m]} - \alpha_{t-1}^{[m]}) \\ &= \alpha_{t-1}^{[m]} + \left[I + (R_t + \Sigma_{n_t,t-1}^{[m]})^{-1} + P_t'^{-1}\right]^{-1}(\beta_{n_t,t-1}^{[m]} - \alpha_{t-1}^{[m]}) \end{aligned} \tag{4.79}$$

Poichè R_t, $P_t'^{-1}$ e $\Sigma_{n_t,t-1}^{[m]}$ sono tutte semidefinite positive, l'inversa di $I + (R_t + \Sigma_{n_t,t-1}^{[m]})^{-1} + P_t'^{-1}$ sarà anch'essa semidefinita positiva con autovalori tutti minori di uno: questo prova il Lemma 1. In particolare il valore atteso di $\alpha_{t-1}^{[m]}$ diminuisce se $\beta_{n_t,t}^{[m]}$ è più piccolo di $\alpha_{t-1}^{[m]}$. Viceversa se $\alpha_{t-1}^{[m]}$ è più piccolo di $\beta_{n_t,t}^{[m]}$, la (4.79) implica che il valor atteso di $\alpha_{t-1}^{[m]}$ aumenterà, ma di una quantità pari alla loro differenza. Questo assicura che $\alpha_{t-1}^{[m]}$ non supererà nel valor atteso $\beta_{n_t,t}^{[m]}$.

Di particolare interesse è la situazione che si ha quando il robot osserva un *anchoring feature*. Senza perdità di generalità si assume che la *feature* conosciuta *a priori* sia θ_1.

Lemma 2. *Se il robot osserva l'anchoring feature, il valor atteso sulla posa diminuirà.*

Prova del Lemma 2. Per l'*anchoring feature* θ_1 è possibile sfruttare il fatto che $\Sigma_{1,t}^{[m]} = \beta_{1,t}^{[m]} = 0$. La dimostrazione del Lemma 2 segue allora direttamente dalla (4.79),

$$\begin{aligned} E[\alpha_t^{[m]}] &= \alpha_{t-1}^{[m]} + \left[I + (R_t + 0) + P_t'^{-1}\right]^{-1}(0 - \alpha_{t-1}^{[m]}) \\ &= \alpha_{t-1}^{[m]} - \left[I + R_t P_t'^{-1}\right]^{-1}(\alpha_{t-1}^{[m]}) \end{aligned} \tag{4.80}$$

Quindi se il robot osserva l'*anchoring feature* l'errore sulla posa $\alpha_t^{[m]}$ diminisce. L'unica eccezione si ha quando l'errore è già zero, nel qual caso esso rimane tale.

Un Lemma molto simile al Lemma 1 viene trattato per mostrare l'influenza dell'errore sulla posa α rispetto a quello sulla mappa β.

Lemma 3. *Se l'errore sulla posa del robot $\alpha_{t-1}^{[m]}$ è più piccolo rispetto all'errore sulla feature osservata, z_t fa diminuire il valore attteso su $\beta_{n_t,t}^{[m]}$. Viceversa se $\alpha_{t-1}^{[m]}$ è più grande dell'errore sulla feature $\beta_{n_t,t}^{[m]}$, quest'ultimo può incrementare, ma nel valor atteso non eccederà $\alpha_{t-1}^{[m]}$.*

Prova del Lemma 3. La dimostrazione di questo Lemma è del tutto analoga a quella del Lemma 1. Dalla (4.72) segue che il valor atteso dell'errore sulla *feature* dopo l'*update*:

$$\begin{aligned} E[\beta_{n,t}^{[m]}] &= E[\mu_{n,t}^{[m]} - \theta_n] = E[\mu_{n,t}^{[m]}] - \theta_n \\ &= E[\mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\hat{z}_t^{[m]} - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t)] - \theta_n \\ &= \mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(E[\hat{z}_t^{[m]}] - \mu_{n_t,t-1}^{[m]} + s_{t-1}^{[m]} + u_t) - \theta_n \end{aligned} \tag{4.81}$$

L'equazione (4.80) ci permette di riscrivere quest'ultima equazione come segue:

$$\begin{aligned} E[\beta_{n,t}^{[m]}] &= \mu_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]}) - \theta_n \\ &= \beta_{n_t,t-1}^{[m]} + \Sigma_{n_t,t-1}^{[m]}(R_t + \Sigma_{n_t,t-1}^{[m]})^{-1}(\alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]}) - \theta_n \\ &= \beta_{n_t,t-1}^{[m]}(\Sigma_{n_t,t-1}^{[m]-1} R_t)^{-1}(\alpha_{t-1}^{[m]} - \beta_{n_t,t-1}^{[m]}) \end{aligned} \tag{4.82}$$

Come nella dimostrazione del Lemma 1, entrambe R_t e $\Sigma_{n_t,t-1}^{[m]-1}$ sono semidefinite positive; l'inversa di $(\Sigma_{n_t,t-1}^{[m]-1} R_t)$ è anch'essa semidefinita positiva con tutti autovalori minori di uno. Resta quindi dimostrato il Lemma.

La veridicità del Teorema segue dai tre lemmi di cui sopra e dalle equazioni che modellano l'evoluzione del valor medio dell'errore nel tempo.

Prova del Teorema 1. Si consideri $\hat{\beta}_t^{[m]}$ che denota l'errore più grande nella stima della posizione di tutte le *feature* al tempo t.

$$\hat{\beta}_t^{[m]} = \underset{\beta_{n,t}^{[m]}}{\operatorname{argmax}} |\beta_{n,t}^{[m]}| \tag{4.83}$$

il Lemma 3 indica come il valore atteso dell'errore sulla mappa possa incrementare, ma solo se è minore del valore assoluto dell'errore sulla posa del robot $\alpha_{t-1}^{[m]}$. Tuttavia, questo sarà vero solo per poche iterazioni. In particolare, il Lemma 1 garantisce che, se questo accade, $\alpha_{t-1}^{[m]}$ può solo diminuire (nel suo valore atteso). Inoltre il Lemma 2 dimostra che ogni volta che l'*anchoring feature* è osservata, questo diminuirà di una quantità finita l'errore $\alpha_{t-1}^{[m]}$ indipendentemente da $\hat{\beta}_t^{[m]}$. Quindi $\alpha_{t-1}^{[m]}$ diventerà più piccola nel suo valore atteso del più grande errore sulla posizione delle *feature*. Detto questo, il Lemma 3 mostra come il valore atteso di quest'ultimo diminuisce ogni volta che la *feature*, a cui è associato l'errore più grande, è osservata; si può quindi osservare come $\hat{\beta}_t^{[m]}$ e $\alpha_{t-1}^{[m]}$ convergano a zero. L'osservazione dell'*anchoring feature* induce una riduzione finita (si veda la (4.79)). Per incrementare $\alpha_{t-1}^{[m]}$ al suo precedente valore atteso, l'errore complessivo atteso sulle *feature* decresce secondo la (4.82); così l'errore sulle *feature* si porta asintoticamente a zero. Dato che questo errore è un *upper bound* per l'errore di posa atteso (Lemma 1), si dimostra la convergenza del valore atteso dell'errore sulla posa.

Il teorema appena dimostrato è esteso al caso di più particelle ($M > 1$) dal seguente corollario:

Corollario 1. *Il FastSLAM converge nei valori attesi per l'LG-SLAM se tutte le feature sono osservate infinite volte e se la posizione di una feature è conosciuta a priori.*

4.5 FastSLAM con Unknown Data Association

Tutte le considerazioni riguardo i punti fondamentali dell'algoritmo del FastSLAM fin qui descritte sono state fatte ipotizzando di conoscere il vettore delle associazione di dati n^t. Nella realtà, come si è già detto, questo vettore non è noto in quanto la conoscenza dell'associazione di dati presuppone la conoscenza della mappa, che è invece uno degli obbiettivi dello SLAM. Questa sezione estende l'algoritmo del FastSLAM al caso di associazione di dati non nota. La soluzione più tipica per risolvere questo tipo di problema è di scegliere l'n_t che massimizza un criterio di massima verosimiglianza rispetto alle misurazioni z_t data tutte le precedenti misurazioni:

$$\hat{n}_t = \underset{n_t}{\operatorname{argmax}}\, p(z_t|n_t, s^t, z^{t-1}, u^t, \hat{n}^{t-1}) \tag{4.84}$$

Il termine $p(z_t|n_t, s^t, z^{t-1}, u^t, \hat{n}^{t-1})$ è la funzione di verosimiglianza e questo approccio è un esempio di stimatore del tipo *maximum likelihood* (ML). L'associazione di dati ML è anche indicata come associazione di dati *nearest neighbor*, interpretando il logaritmo negativo di likelihood come una funzione di distanza. Per esempio nel caso di modelli Gaussiani, il *log negative likelihood* altro non è che la distanza di Mahalanobis, e lo stimatore usa questa metrica per selezionare le associazioni di dati.

É chiaro ora il perchè l'EKF-SLAM abbia come punto debole l'associazione di dati, che può anche causare la divergenza del filtro. Infatti EKF-SLAM utilizza una singola associazione di dati per l'intero filtro; così una serie di associazioni di dati sbagliate scatenano una reazione a catena di altre associazioni errate, mettendo quasi sempre a repentaglio l'intera convergenza. Capire come l'incertezza nello SLAM *posterior* conduca ad associazioni di dati errate dimostrerà implicitamente come questa semplice euristica spesso fallisca.

4.5.1 Cause dell'incertezza nell'associazione di dati

I fattori che contribuiscono ad accrescere l'incertezza circa lo SLAM *posterior* e quindi indirettamente causa delle ambiguità nell'associazione di dati sono il

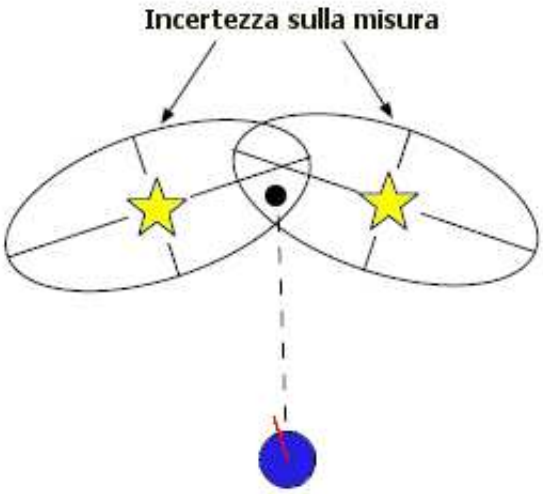


Figura 4.9: Ambiguità di misura: un forte disturbo sulla misura conduce ad ambiguità circa l'osservazione di *landmark* vicini.

rumore di moto e il rumore di misura (si vedano a proposito le (2.6) e (2.7)). Quando il rumore sulle osservazioni aumenta, la posizione dei *landmark* diventa sempre più incerta. A patto di considerare un rumore di misura sufficientemente alto, la distribuzione di incertezza circa l'osservazione di landmark vicini si sovrapporrano causando appunto la cosiddetta *measurement ambiguity*, ossia quando l'ambiguità sull'associazione di dati è causata da rumore di misura. Un esempio di ambiguità di misura è riportato in Figura 4.9. I due ellissi indicano il *range* di probabili osservazioni causate dai *landmark* mentre il pallino nero indica l'osservazione corrente: come si vede l'osservazione può essere stata indifferentemente generata da uno o dall'altro *landmark*.

L'attribuzione di una osservazione al *landmark* sbagliato causata da incertezza di misura incrementerà sicuramente l'errore sulla mappa e sulla posa del robot, ma il suo impatto sull'andamento generale dell'algoritmo sarò abbastanza ridotto. Questo perchè l'osservazione potrebbe essere stata generata da uno dei due *landmark* con altà probabilità, l'effetto circa la stima della posizione del *landmark* e della posa del robot sarà ridotto; ciò che si avrà sarà una leggera sottostima della covarianza di un *landmark*, mentre per la

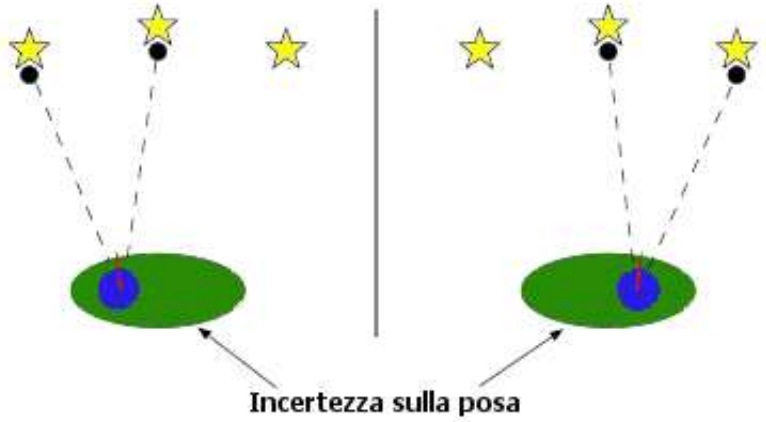


Figura 4.10: Ambiguità di associazione di dati causata dal rumore sul moto.

covarianza dell'altro *landmark*k si avrà una leggera sovrastima. Un metodo per ovviare immediatamente a questo problema è di incorporare nello stesso time step in cui si ha un solo controllo, osservazioni multiple.

L'ambiguità nell'associazione di dati causata da rumori di moto può invece avere conseguenze molto più gravi sull'accuratezza della stima. Infatti un rumore consistente sul moto conduce ad una forte incertezza sulla posa dopo avere incorporato un nuovo controllo. Se questa incertezza sulla posa è abbastanza consistente, assumere differenti valori in questa distribuzione per la posa del robot condurrà a differenti ML ipotesi di associazioni per le seguenti osservazioni. Una tipica circostanza in cui questo avviene è se c'è un significativo errore nella velocità angolare, come mostrato in Figura 4.10. Inoltre se si sceglie di incorporare più misure per un unico controllo, la posa del robot verrà correlata per l'associazione di dati con molte più osservazioni. Se l'algoritmo di SLAM fallisce l'associazione di dati per una singola osservazione, in genere, con altissima probabilità, fallisce anche le successive associazioni di dati. Un numero abbastanza consistente di associazione di dati errata può condurre alla divergenza del filtro in uno SLAM su base EKF.

4.5.2 Associazione di dati con euristica ML su base particellare

A differenza di molti approcci su base EKF, il FastSLAM considerà più ipotesi per l'associazione di dati. Infatti ogni particella rappresenta un'ipotesi di *path* per il robot e come conseguenza l'associazione di dati viene fatta su base particellare. Quelle particelle che colgono la corretta associazione di dati riceveranno un alto *importance factor* in quanto giustificano bene le correnti osservazioni e avranno quindi una vita più lunga nel filtro, essendo con alta probabilità selezionate nello step di *resampling*. Particelle che falliscono l'associazione di dati verranno invece pesate con un *importance factor* minore e sostituite in futuro da altre nel *resampling*. L'associazione di dati per-particella presenta molti vantaggi rispetto alla associazione di dati ML. L'incertezza sul moto viene resa indipendente rispetto al problema di associazione di dati. Dato che l'ambiguità di moto è ciò che mette maggiormente a repentaglio l'associazione corretta, affrontare quest'ultima con un approccio di tipo multi-ipotesi per il *path* del robot sembra la scelta più ragionevole. Sempre facendo riferimento alla Figura 4.10 alcune particelle modelleranno la nuova posa in maniera consistente con l'ipotesi di associazione di dati sulla sinistra, mentre altre con quella sulla destra.

L'approccio di associazione di dati per-particella inoltre rende l'intero problema molto più semplice. Con l'Extended Kalman Filter l'incertezza circa la posizione di un *landmark* è dovuta sia all'incertezza nella posa del robot sia all'incertezza dovuta all'errore di misura. Nel FastSLAM, l'incertezza circa la posa del robot viene modellata dall'intero set di particelle. Così i filtri di una singola particella non sono affetti dall'errore di moto in quanto sono condizionati su uno specifico *path* del robot. Questa caratteristica è particolarmente utile nel caso in cui i rumori sull'odometria siano molto più consistenti rispetto ai rumori di misura. Un altra conseguenza dell'associazione di dati su base particellare è implicita. Se in un dato istante di tempo una frazione di particelle ricevesse una plausibile ma sbagliata associazione di dati, nel futuro il robot incoporerebbe sicuramente osservazioni che entrerebberp in contrasto con la precedente associazione di dati; così le par-

ticelle riceverebbero un peso basso e non sopravviverebbero al *resampling*. Quindi un'errata associazione dei dati nel passato può chiaramente essere rimossa. Non bisognerà inoltre trovare alcuna euristica per rimuovere associazioni sbagliate fatte nel passato. Questo processo viene statisticamente espletato come una diretta conseguenza del *resampling*.

Nel FastSLAM l'associazione di dati viene risolta fondendo l'euristica ML con l'approccio su base particellare. Quindi la (4.84) diventa:

$$\hat{n}_t^{[m]} = \underset{n_t}{\operatorname{argmax}}\, p(z_t|n_t, s^{t,[m]}, z^{t-1}, u^t, \hat{n}^{t-1,[m]}) \tag{4.85}$$

Quindi i campioni vengono ottenuti dalla seguente probabilità:

$$\hat{n}_t^{[m]} \sim \eta p(z_t|n_t, s^{t,[m]}, z^{t-1}, u^t, \hat{n}^{t-1,[m]}) \tag{4.86}$$

La probabilità nella (4.86) viene calcolata come segue:

$$\begin{aligned}
&p(z_t|n_t, s^{t,[m]}, z^{t-1}, u^t, \hat{n}^{t-1,[m]}) \\
&= \int p(z_t|\theta_{n_t}, n_t, s^{t,[m]}, z^{t-1}, u^t, \hat{n}^{t-1,[m]}) p(\theta_{n_t}|n_t, s^{t,[m]}, z^{t-1}, u^t, \hat{n}^{t-1,[m]}) d\theta_{n_t} \\
&= \int \underbrace{p(z_t|\theta_{n_t}, n_t, s^{t,[m]})}_{\sim\mathcal{N}(z_t|g(\vartheta_{n_t}, s_t^{[m]}), R_t)} \underbrace{p(\theta_{n_t}|s^{t-1,[m]}, z^{t-1}, \hat{n}^{t-1,[m]})}_{\sim\mathcal{N}(\vartheta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]})}
\end{aligned} \tag{4.87}$$

Linearizzare g permette di ottenere una forma chiusa per l'integrale precedente:

$$\begin{aligned}
&p(z_t|n_t, s^{t,[m]}, z^{t-1}, u^t, \hat{n}^{t-1,[m]}) \\
&= |2\pi H_t^{[m]}|^{-\frac{1}{2}} \exp\Big\{ -\frac{1}{2}(z_t - g(\mu_{n_t,t-1}^{[m]}, s_t^{[m]}))^\intercal H_t^{[m]-1}(z_t - g(\mu_{n_t,t-1}^{[m]}, s_t^{[m]})) \Big\}
\end{aligned} \tag{4.88}$$

dove $H_t^{[m]}$ è definita nella (4.30). Se il valore di questa probabilità scende sotto una fissata soglia p_0, allora un nuovo *landmark* viene aggiunto alla particella. Quindi ogni particella conteggia le proprie *feature* attraverso la variabile $N_t^{[m]}$. L'associazione di dati ML tende a lavorare meglio nel FastSLAM piuttosto che nell'EKF-SLAM. La ragione di questa maggiore efficienza è che la componente più forte della ambiguità viene dal rumore sulla dimanica di moto.

4.6 Aggiunta di un nuovo landmark

L'aggiunta di nuovo *landmark* è un procedimento molto delicato. Questo è particolarmente vero nel caso in cui una singola misurazione è in grado di definire un *landmark* in tutte le sue dimensioni. Se la funzione di misura $g(\theta_{n_t}, s_t)$ è invertibile, una singola misurazione è sufficiente ad inizializzare un *landmark*. Ogni osservazione definisce una Gaussiana:

$$\mathcal{N}(z_t; \hat{z}_t + G_{\theta_{n_t}}(\theta_{n_t} - \mu^{[m]}_{n_t,t-1}, R_t)) \tag{4.89}$$

Questa Gaussiana può esplicitamente essere riscritta come:

$$\frac{1}{\sqrt{2\pi R_t}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_t - G_{\theta_{n_t}}(\theta_{n_t} - \mu^{[m]}_{n_t,t-1}))^{\mathsf{T}} R_t^{-1}(z_t - \hat{z}_t - G_{\theta_{n_t}}(\theta_{n_t} - \mu^{[m]}_{n_t,t-1}))\right\} \tag{4.90}$$

Si definisce poi una funzione J pari all'esponente della precedente Gaussiana cambiato di segno:

$$J = \frac{1}{2}(z_t - \hat{z}_t - G_{\theta_{n_t}}(\theta_{n_t} - \mu^{[m]}_{n_t,t-1}))^{\mathsf{T}} R_t^{-1}(z_t - \hat{z}_t - G_{\theta_{n_t}}(\theta_{n_t} - \mu^{[m]}_{n_t,t-1})) \tag{4.91}$$

La derivata seconda di J rispetto a θ_{n_t} è l'inversa della matrice di covarianza della Gaussiana nelle coordinate del *landmark*:

$$\frac{\partial J}{\partial d\theta_{n_t}} = -(z_t - \hat{z}_t - G_{\theta_{n_t}}(\theta_{n_t} - \mu^{[m]}_{n_t,t-1}))^{\mathsf{T}} R_t^{-1} G_{\theta_{n_t}} \tag{4.92}$$

$$\frac{\partial J^2}{\partial d\theta^2_{n_t}} = G^{\mathsf{T}}_{\theta_{n_t}} R_t^{-1} G_{\theta_{n_t}} \tag{4.93}$$

Quindi l'osservazione invertibile vien usata per creare un nuovo *landmark* come segue:

$$\mu^{[m]}_{n_t,t} = g^{-1}(s^{[m]}_t, z_t) \tag{4.94}$$

$$\Sigma^{[m]}_{n_t,t} = (G^{\mathsf{T}}_{\theta_{n_t}} R_t^{-1} G_{\theta_{n_t}})^{-1} \tag{4.95}$$

Capitolo 5

Unscented FastSLAM

In questo capitolo vengono riportate le modifiche di natura puramente teorica apportate all'algoritmo del FastSLAM [4]. Queste modifiche riguardano in qualche modo tutti i passi dell'algoritmo standard e si basano sull'Unscented Filtering trattato ampiamente nel Capitolo 3. Tecniche di tipo Unscented già sono state sfruttate per risolvere lo SLAM [1, 2, 22], ma mai introdotte nel FastSLAM (il quale, rispetto ai precedenti algoritmi, utilizza i filtri particellari). Inoltre per quel che riguarda il resampling, questo processo appartiene al filtro particellare addetto alla stima del *path*; in genere il tipo di resampling adottato dal FastSLAM è di tipo selettivo, ma non necessariamente. In una delle sezioni di questo capitolo verrà introdotto e giustificato un tipo di approccio innovativo al resampling e ne verrà dimostrata la consistenza.

5.1 Estensione del path posterior attraverso Unscented Proposal

Come visto nel Capitolo 4 il FastSLAM ottiene una nuova posa per ogni particella campionando dalla (4.18), che qui riportiamo per comodità:

$$s_t^{[m]} \sim p(s_t | s^{t-1,[m]}, u^t, z^t, n^t) \tag{5.1}$$

Come si è visto nella sezione 4.3.3, sviluppando la precedente e utilizzando distribuzioni di probabilità note, come il *measurement model*, il *motion model*

e la posizione di un landmark modellato da una Gaussiana, si ottiene la (4.22) che viene anche qui riportata per comodità:

$$s_t^{[m]} \sim p(s^t|s^{t-1}, z^t, u^t, n^t) = \eta^{[m]} \int \underbrace{p(z_t|\vartheta_{n_t}, s_t, n_t)}_{\sim\mathcal{N}(z_t|g(\vartheta_{n_t},s_t),R_t)} \underbrace{p(\vartheta_{n_t}|s^{t-1,[m]}, z^{t-1}, n^{t-1})d\vartheta_{n_t}}_{\sim\mathcal{N}(\vartheta_{n_t};\mu^{[m]}_{n_t,t-1},\Sigma^{[m]}_{n_t,t-1})} \underbrace{p(s_t|s_{t-1}^{[m]}, u_t)}_{\sim\mathcal{N}(s_t|h(s^{[m]}_{t-1},u_t),Q_t)} \tag{5.2}$$

Il termine più a destra della precedente costituisce l'oggetto dell'analisi; esso rappresenta l'evoluzione delle particelle in accordo con il solo controllo rumoroso al tempo t, e Q_t[1] modella l'attuale incertezza nella posa della particella, mentre i due termini dentro l'integrale permettono alla *proposal* di tener conto delle più recenti osservazioni z_t. Definiamo allora Ξ_t come l'evoluzione del set di particelle S_{t-1} dovuto al nuovo controllo. Ξ_t ha la seguente struttura:

$$\Xi_t = \{\langle\xi_t^{[1]}, Q_t^{[1]}\rangle, ..., \langle\xi_t^{[M]}, Q_t^{[M]}\rangle\} \tag{5.3}$$

dove $\xi_t^{[m]}$ è la posa per l'm-esima particella e $Q_t^{[m]}$ è la nuova matrice di covarianza associata all'm-esima particella. Il FastSLAM calcola questa quantità semplicemente sostituendo l'attuale valore della posa della particella $s_{t-1}^{[m]}$ nel modello di moto h e guardando al nuovo valore ottenuto come al valor medio di una Gaussiana che modella la distribuzione in accordo alla quale è posizionata la nuova particella. In formule:

$${}^s\xi_t{}^{[m]} = h(s_{t-1}^{[m]}, u_t) \tag{5.4}$$

[2]mentre per la matrice di covarianza che, lo ricordiamo, modella la dispersione dei valori attorno al valor medio, si ha:

$${}^sQ_t^{[m]} = G_{s_{t-1}} Q_{t-1}^{[m]} G_{s_{t-1}}^T + G_{u_t} P_{t-1} G_{u_t}^T \tag{5.5}$$

dove $G_{s_{t-1}}$ e G_{u_t} sono i Jacobiani di h, cioè le derivate di h rispetto a s_{t-1} e a u_t. Questo tipo di approccio, come si può vedere, considera solo il primo

[1]Nel Capitolo 4 questa quantità era indicata come $P_t^{'}$.

[2]Da adesso in poi denoteremo il calcolo standard del FastSLAM di $\xi_t^{[m]}$ con ${}^s\xi_t{}^{[m]}$, mentre con ${}^u\xi_t{}^{[m]}$ il calcolo di tipo Unscented. Analogamente per la matrice di covarianza e per le altre quantità che verranno introdotte. Si capisce ora come il $P_t^{'}$ del Capitolo 4 nella (4.22) era pari a ${}^sQ_t^{[m]}$.

ordine dell'espansione di Taylor per il calcolo di $h(s_{t-1}^{[m]}, u_t)$ (si veda la sezione 3.1 per maggior chiarezza); ignorando i termini di ordine maggiore, a volte il FastSLAM produce un valore non coerente per ${}^s\xi_t{}^{[m]}$ e per ${}^sQ_t^{[m]}$ e quindi un set non preciso ${}^s\Xi_t$. Di conseguenza la risoluzione della (5.2) conduce ad una *proposal distribution* da cui campionare le nuove pose non consistente con il reale *posterior* nonostante questa sia notevolmente raffinata nel considerare anche z_t. Un set più accurato Ξ_t può essere calcolato sfruttanto l'*Unscented Transformation* (UT) (si veda il capitolo 3).

L'*Unscented Transformation* è un metodo sviluppato per propagare correttamente informazioni circa i momenti di distribuzioni (in genere i primi due, ossia il valor medio e la covarianza) attraverso sistemi non-lineari. Questo metodo si fonda sull'intuizione che «is easier to approximate a Gaussian distribution than it is to approximate an arbitrary non-linear function or transformation »[11]: cioè, è più facile approssimare una Gaussiana rispetto ad un arbitraria funzione non-lineare. Il calcolo di Ξ_t, che è una sorta di *prediction step*, dovuto solo al moto imposto, viene effettuato attraverso il modello del sistema, in questo caso h. Questo permette di predire la *pdf* (probability density function) della posa del robot. Tuttavia poichè quest'ultima è in genere soggetta a disturbi non noti, la predizione in genere trasla, deforma e sparpaglia la sua *pdf*. Così la forte non-linearità della dinamica del moto unita ai disturbi non noti può produrre una non corretta propagazione dei due momenti della Gaussiana che modella l'incertezza circa la posizione e l'orientamento del robot. Ecco perchè molto spesso il FastSLAM genera un set Ξ_t non molto accurato.

L'UT in un certo senso protegge il valor medio e la covarianza quando attraversano il sistema non-lineare $h(s_{t-1}^{[m]}, u_t)$. Così ad ogni time step è possibile produrre nel complesso una più compatta nube di particelle attorno al reale valore della posizione del robot insieme all'estrazione di più compatti gruppi di *feature* (questo è dovuto alla minore sparsità particellare nel complesso). I Frames della Figura 5.1 mostrano l'evoluzione del set S delle particelle (in blu). Come si può vedere, alcune volte il FastSLAM produce set inconstenti rispetto alla reale posizione del robot in rosso. Con l'UT,

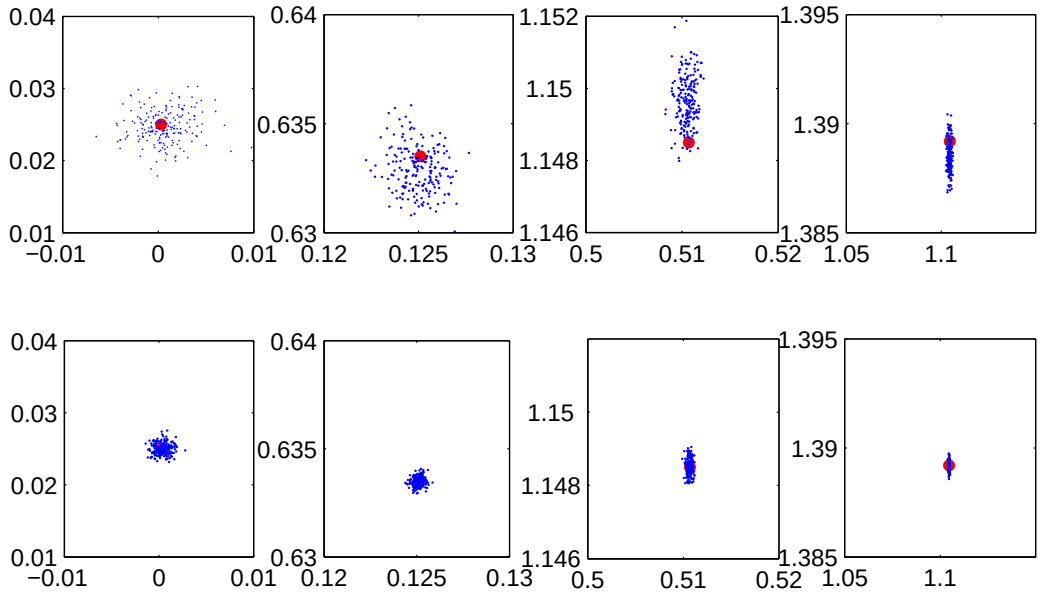


Figura 5.1: Evoluzione delle nubi di particelle calcolate attraverso il FastSLAM standard (in alto) e attraverso il FastSLAM con l'UT (in basso). Come si può vedere il FastSLAM qualche volta genera un set di particelle inconsistente.

invece, il set è sempre consistente e la particella che ha peso maggiore e che viene considerata come la miglior stima che il filtro possa fare circa la posa del robot, si discosta sempre di pochissimo dal valore reale. L'UT permette quindi di sottostimare la matrice di covarianza della posa della particella, non mettendo però a repentaglio la convergenza del filtro. Inoltre con questo tipo di approccio è possibile ridurre drasticamente il fenomeno del *particles depletion* o impoverimento dei campioni (vedi sezione 6.1).

Applicare l'UT per la propagazione dei due momenti richiede aumentare il vettore di stato (in questo caso la posa del robot) con il rumore sul controllo. Quindi il nuovo vettore di stato ha dimensione $n^a = 3 + n^u$, dove appunto n^u è la dimensione del vettore che modella il rumore sul controllo. Quindi il nuovo vettore di posa è dato da $[s_{t-1}^{[m]}\ \delta_{t-1}]$, mentre la matrice di covarianza

aumentata è data da:

$$Q_{t-1}^{[m]'} = \begin{bmatrix} Q_{t-1}^{[m]} & P_{cross,t-1} \\ P_{cross,t-1} & P_{t-1} \end{bmatrix} \tag{5.6}$$

Da adesso in poi il vettore aumentato $[s_{t-1}^{[m]}\ \delta_{t-1}]$ sarà denotato semplicemente come $s_{t-1}^{[m]}$. In questo caso, inoltre, $P_{cross,t-1} = 0$, in quanto si fa l'ipotesi di correlazione nulla tra posa e rumore di controllo. Un set di *punti sigma* $\mathcal{S} = \{\mathcal{X}^{i,[m]}, W^{i,[m]}\}$, $i = 0, 1, \ldots, p$ è scelto per l'm-esima particella in modo da soddisfare una set di condizioni della forma:

$$\mathbf{g}(S, p(s_{t-1}^{[m]})) = \mathbf{0} \tag{5.7}$$

dove $p(s_{t-1}^{[m]})$ è la *pdf* di $s_{t-1}^{[m]}$, Gaussiana nel nostro caso, e $\mathbf{g}(\cdot\,,\cdot\,)$ fissa le proprietà che devono essere catturate circa $s_{t-1}^{[m]}$. Le quantità $W^{i,[m]}$ rappresentano i pesi, i quali, per produrre una stima corretta, devono soddisfare la seguente condizione di normalizzazione:

$$\sum_{i=0}^{p} W^{i,[m]} = 1 \tag{5.8}$$

Quindi per catturare il valor medio deve essere che:

$$g_1 = \sum_{i=0}^{p} \mathcal{X}^i, W^i - \bar{s}_{t-1}^{[m]} \tag{5.9}$$

dove $\bar{s}_{t-1}^{[m]}$ è l'attuale valor medio di $s_{t-1}^{[m]}$. Mentre per la covarianza:

$$g_2 = \sum_{i=0}^{p} W^i (\mathcal{X}^i - \bar{s}_{t-1}^{[m]})(\mathcal{X}^i - \bar{s}_{t-1}^{[m]})^T - Q_{t-1}^{[m]'} \tag{5.10}$$

Successivamente il *motion model* viene applicato a tutti i punti sigma ($i = 0, ..., p$) per l'm-esima particella in modo da restituire una nuvola di punti trasformati:

$$\mathcal{Z}^{i,[m]} = h(\mathcal{X}^{i,[m]}) \tag{5.11}$$

Ora il valor medio, ossia la *prediction* per la posa dell'm-esima particella è dato dalla media pesata dei punti trasformati:

$${}^{u}\xi_t^{[m]} = \sum_{i=0}^{p} W^{i,[m]} Z^{i,[m]} \tag{5.12}$$

mentre la nuova covarianza è il prodotto punto pesato dei punti trasformati:

$$ {}^{u}Q_t = \sum_{i=0}^{p} W^i (\mathcal{Z}^{i,[m]} - {}^{u}\xi_t^{[m]})(\mathcal{Z}^{i,[m]} - {}^{u}\xi_t^{[m]})^T \tag{5.13} $$

Poichè il valor medio e la covarianza di $s_t^{[m]}$ sono catturati in maniera precisa fino al second'ordine, è possibile ottenere un set molto accurato per Ξ_t. La distribuzione $p(s_t|s_{t-1}^{[m]}, u_t)$ viene approssimata in maniera più precisa da $\mathcal{N}({}^{u}\xi_t{}^{[m]}, {}^{u}Q_t)$ piuttosto che da $\mathcal{N}({}^{s}\xi_t{}^{[m]}, {}^{s}Q_t)$ e il calcolo della proposal risulta più accurato. Inoltre questo metodo evita il calcolo dei Jacobiani.

5.2 Aggiornamento della stima dei landmark tramite Unscented Kalman Filter

Dopo aver stimato il path del robot, il FastSLAM condiziona la stima della posizione dei landmark sulla posa del robot ottenuta appunto con il filtro particellare. Proprio perchè $p(\theta_{n_t}|s^t, z^t, u^t, n^t)$ è condizionata sulla posa del robot, ogni particella, che costituisce un'ipotesi di *path*, dispone del proprio gruppo di stimatori, che, nel caso del FastSLAM, sono Extended Kalman Filters. Dato che i landmark della mappa sono N, ad ogni particella sono associati N EKFs, appunto uno per ogni landmark. Riportiamo qui per comodità la struttura completa del filtro che stima il *path* e la posizione dei landmark (4.15):

$$ S_t^{[m]} = \left\langle s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, ..., \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\rangle \tag{5.14} $$

Si ricorda inoltre che nel caso in cui $n_t = n$, ossia il landmark osservato al tempo t è n, allora il suo posterior diventa:

$$ p(\vartheta_n|s^t, z^t, u^t, n^t) = p(z_t|\vartheta_n, s^t, n^t)\, p(\vartheta_n|s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \tag{5.15} $$

mentre per $n_t \neq n$ rimane inalterato:

$$ p(\vartheta_n|s^t, z^t, u^t, n^t) = p(\vartheta_n|s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \tag{5.16} $$

il FastSLAM stardard per l'aggiornamento di queste equazioni attraverso l'EKF, linearizza la dinamica di osservazione con tutti i ben noti problemi

che questo tipo di approccio comporta. L'UKF, oltre ad evitare la linearizzazione, permette di ottenere una stima migliore della posizione del landmark. Come nel caso dell'*Unscented proposal*, si aumenta lo stato, in questo caso la posizione del landmark descritta dalle coordinate $x - y$, del rumore di misura; quindi $n^a = 2 + n^m$, con n^m appunto dimensione del rumore di misura. Quindi il nuovo vettore che stima la posizione del landmark è del tipo $[\mu^{[m]}_{n_t,t-1})\ \varepsilon_{t-1}]$, mentre la covarianza sarà pari a:

$$\Sigma^{[m]'}_{t-1} = \begin{bmatrix} \Sigma^{[m]}_{n_t,t-1} & \Sigma_{cross,t-1} \\ \Sigma_{cross,t-1} & R_{t-1} \end{bmatrix} \tag{5.17}$$

Anche in questo caso $\Sigma_{cross,t-1} = 0$ in quanto non esiste correlazione tra lo stato del landmark e il rumore sulla misura. Inoltre $\mu^{[m]}_{n_t,t-1}$ indicherà lo stato del landmark aumentato del rumore di misura. Si genera quindi il set di *punti sigma* $\mathcal{L} = \{\mathcal{L}^{i,[m]}, W^{i,[m]}\}$, $i = 0, 1, \ldots, p$ con le strategie di selezione viste nel Capitolo 3:

$$\mathcal{L} = \begin{bmatrix} \mu^{[m]}_{n_t,t-1} & \mu^{[m]}_{n_t,t-1} + \gamma\sqrt{\Sigma^{[m]'}_{t-1}} & \mu^{[m]}_{n_t,t-1} - \gamma\sqrt{\Sigma^{[m]'}_{t-1}} \end{bmatrix} \tag{5.18}$$

Questi punti sigma soddisfano il set di equazioni per l'n_t-esimo landmark dell'm-esima particella:

$$\mathbf{f}(L, p(\mu^{[m]}_{n_t,t-1})) = \mathbf{0} \tag{5.19}$$

dove $p(\mu^{[m]}_{n_t,t-1})$ è la distribuzione di probabilità di $\mu^{[m]}_{n_t,t-1}$, Gaussiana nel nostro caso, e $\mathbf{f}(\cdot\,,\cdot\,)$ determina le informazioni da catturare per $\mu^{[m]}_{n_t,t-1}$. Il valor medio si ottiene come:

$$f_1 = \sum_{i=0}^{p} \mathcal{L}^i W^i - \bar{\mu}^{[m]}_{n_t,t-1} \tag{5.20}$$

dove $\bar{\mu}^{[m]}_{n_t,t-1}$ è l'attuale valor medio di $s^{[m]}_{t-1}$. Per la matrice di covarianza:

$$f_2 = \sum_{i=0}^{p} W^i(\mathcal{L}^i - \bar{\mu}^{[m]}_{n_t,t-1})(\mathcal{L}^i - \bar{\mu}^{[m]}_{n_t,t-1})^{\mathsf{T}} - \Sigma^{[m]'}_{t-1} \tag{5.21}$$

La dinamica di misura $g(\theta_{n_t}, s^{[m]}_t)$ viene riscritta in modo da essere applicata ad ogni punto $(i = 0, ..., p)$ per ottenere:

$$\mathcal{Z}^{i,[m]} = \tilde{g}(\mathcal{L}^{i,[m]}) \tag{5.22}$$

Prendendo la sommatoria pesata dei punti precedentemente ottenuti si calcola la predizione di misura come:

$$z_t^{-[m]} = \sum_{i=0}^{p} W^{i,[m]} Z^{i,[m]} \tag{5.23}$$

mentre la covarianza della misura:

$$\Sigma_{zz} = \sum_{i=0}^{p} W^{i}(\mathcal{Z}^{i,[m]} - z_t^{-[m]})(\mathcal{Z}^{i,[m]} - z_t^{-[m]})^{\mathsf{T}} \tag{5.24}$$

Si calcola poi la matrice di correlazione incrociata dello stato con la misura:

$$\Sigma_{z\mu} = \sum_{i=0}^{p} W^{i}(\mathcal{Z}^{i,[m]} - z_t^{-[m]})(\mathcal{L}^{i,[m]} - \bar{\mu}_{n_t,t-1}^{[m]})^{\mathsf{T}} \tag{5.25}$$

Il guadagno di Kalman è dato da:

$$\mathcal{K} = \Sigma_{z\mu}\, \Sigma_{zz}^{\mathsf{T}} \tag{5.26}$$

mentre la nuova stima per il landmark è pari a:

$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + \mathcal{K}(z_t^{[m]} - z_t^{-[m]}) \tag{5.27}$$

Per quel che riguarda la nuova matrice di covarianza, essa è pari a:

$$\Sigma_t^{[m]'} = \Sigma_{t-1}^{[m]'} - \mathcal{K}\Sigma_{zz}\mathcal{K}^{\mathsf{T}} \tag{5.28}$$

Capitolo 6

L'Adaptive Selective Resampling

In questo capitolo viene introdotto un nuovo tipo di *resampling* per il filtro particellare addetto alla stima del *path* del robot. L'algoritmo del FastSLAM non prevede un ben definito metodo di *resampling*, ma in genere l'approccio è di tipo selettivo, cioè il *resampling* viene effettuato quando una certa quantità che modella la dispersione delle particelle scende sotto una determinata soglia.

Questo capitolo tratta più nel dettaglio il *resampling* e i motivi che ne rendono necessario l'utilizzo, ossia il fenomeno del *particles depletion* o impoverimento dei campioni. In seguito verrà quindi descritto il nuovo tipo di approccio insieme ad alcune considerazioni che ne validano la consistenza. Nel capitolo successivo si misureranno invece quantitativamente gli effettivi miglioramenti introdotti.

6.1 Impoverimento delle particelle

I due passi chiave del filtro particellare sono il sampling dalla *proposal distribution* e l'*importance resampling*. Campionare dalla *proposal* permette di estendere il *path*, generando nuove traiettorie mentre il *resampling* elimina tutte quelle particelle, con le rispettive traiettorie, improbabili. Dall'equilibrio tra queste due operazioni dipende l'andamento della stima del *path*. Chiaramente un filtro con molte traiettorie distinte avrà un sampling più

vario del *posterior* di un filtro particellare con molte traiettorie duplicate. Da queste affermazioni sembrerebbe quasi che lo step di *resampling*, il quale elimina alcune particelle rimpiazzandole con delle altre, sia inutile. In realtà il filtro particellare, senza *resampling*, è destinato a degenerare in poche iterazioni; infatti dopo un numero basso di steps solo una particella avrà un peso grande, mentre tutte le altre avranno peso trascurabile. Viene dimostrato in [xx] che la varianza degli *importance factor* non può che aumentare, rendendo quindi inevitale il fenomeno della degenerazione dei campioni. Si sta allora compiendo un forte sforzo computazionale per aggiornare alcune particelle che non danno in alcun modo contributo all'approssimazione del *posterior*. Il *resampling* evita questa degenerazione assicurata eliminando da subito quei campioni che non stanno contribuendo ad una corretta approssimazione.

6.2 La soglia dinamica

Un modo di misurare gli effettivi campioni N_{eff} utili alla modellazione del *posterior* è stato introdotto da Liu [18]:

$$N_{eff} = \frac{1}{\sum_{m=1}^{M} (w^{[m]})^2} \tag{6.1}$$

L'intuizione che c'è dietro alla (6.1) è abbastanza chiara: se i campioni fossero ottenuti dal reale *posterior*, i fattori di importanza sarebbero tutti uguali e N_{eff} sarebbe pari ad uno. Una cattiva approssimazione comporta un'alta varianza dei campioni e quindi un valore basso per il loro numero effettivo. N_{eff} è una sorta di grandezza che misura la dispersione dei fattori di importanza dei campioni ed è indice di come i *samples* stanno approssimando il vero *posterior*. La tecnica maggiormente impiegata per il *resampling* step è quella introdotta da Doucet [7], il quale avvia il *resampling* non appena N_{eff} cade sotto una fissata soglia, la quale, nella maggior parte dei casi, viene fissata euristicamente (in genere questa soglia va dal 50% al 90% delle particelle). Questo tipo di approccio è definito *selective resampling*.

Utilizzare tuttavia un tipo di soglia statica introduce forti costrizioni nello step di *resampling* e tiene conto solo del valore istantaneo di N_{eff} e non

della sua storia passata. L'evoluzione di N_{eff} contiene invece importanti informazioni su come il filtro stia approssimando il *posterior*, e non guardare a questo putrebbe causare *resampling* poco tempestivi. Una situazione molto tipica si verifica quando il valore di N_{eff} oscilla costantemente intorno ad un valore vicino alla soglia di *resampling*. In questo caso l'andamento costante è sintomo del fatto che il filtro particellare sta dando una buona approssimazione del *posterior*, in quanto è come se ci fosse accordo tra uno step e l'altro circa i pesi delle particelle; e se N_{eff} cade sotto la soglia viene avviato il *resampling*, con la conseguente cancellazione di *samples* utili. In questo caso, allora, sarebbe meglio muovere la soglia, in particolare diminuirne il valore, fino ad un minimo valore accettabile dopo il quale il *resampling* viene comunque avviato. Così come la variabilità di N_{eff} è un indice di come il filtro stia approssimando il *posterior*, così anche il numero medio di particelle nell'intervallo e la soglia dinamica dovrebbe tenerne conto. L'idea quindi è di consisderare, al posto di una soglia statica, una soglia dinamica; questa soglia deve tener conto della variabilità di N_{eff} e del numero medio attuale di particelle sempre nel senso della (6.1).

6.3 Formulazione matematica dell'Adaptive Selective Resampling

Formalmente, per tener conto del passato recente di N_{eff}, viene euristicamente fissata una finestra di osservazione. Ragionando su questa finestra sarà possibile fissare una soglia opportuna per il *resampling* dei successivi step dell' algoritmo. Definiamo questa finestra di osservazione come τ. Denotiamo invece come $v_{\tau-1}$ il valore della soglia nella finestra $\tau - 1$ e con N_{min} e N_{max}, rispettivamente, il minimo valore accettato per N_{eff} (dopo il quale il *resampling* viene comunque avviato) e il massimo valore della soglia. Una euristica funzionante consiste nel fissare $N_{min} = 0.6M$ e $N_{max} = 0.9M$. Da notare che N_{max} deve comunque essere minore di M, cioè del numero massimo di particelle del filtro; questo perchè se la soglia assumesse come valore M, l'algoritmo entrerebbe in un *loop* di infiniti *resampling* dal quale non potrebbe più uscire. In accordo con la variabilità media nella finestra

di osservazione e con il contenuto medio di particelle in essa, viene generata una nuova soglia υ_τ in base alla quale effettuare le successive verifiche per il *resampling*.

Per il calcolo del numero medio di particelle si sommano i valori che N_{eff} assume in $\tau - 1$ e tale sommatoria viene normalizzata rispetto al massimo contenuto particellare ottenibile:

$$\alpha = \frac{1}{Mk} \sum_{t=1}^{k} \frac{N_{eff}(t+1) + N_{eff}(t)}{2} \tag{6.2}$$

dove k e il numero di time step nella finestra e $N_{eff}(t)$ è l'effettivo numero di particelle al t-esimo step. Per cogliere come la varianza dei pesi stia cambiando si calcola invece la derivata media:

$$\beta = \frac{1}{(k-1)(M - N_{min})} \sum_{t=1}^{k-1} [N_{eff}(t+1) - N_{eff}(t)] \tag{6.3}$$

Successivamente si calcolano le due quantità di particelle, ossia un'ipotesi di soglia che tenga conto solo del contenuto di particelle ed una che tenga invece conto di come N_{eff} stia cambiando:

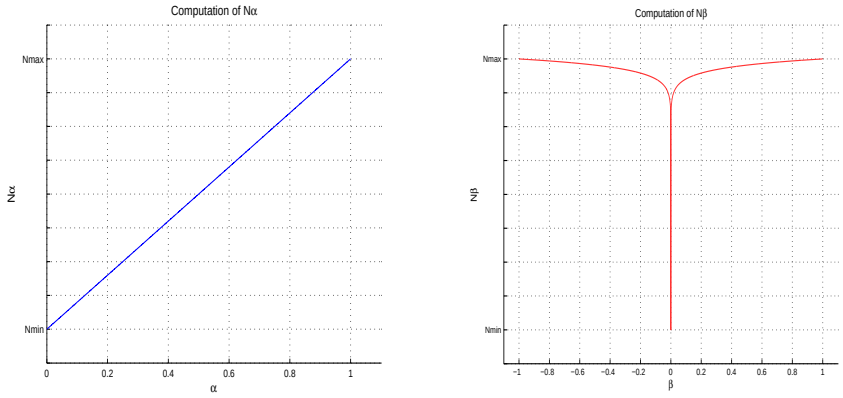
$$N_\alpha = \alpha(N_{max} - N_{min}) + N_{min} \tag{6.4}$$

$$N_\beta = (N_{max} - N_{min})|\beta|^{\frac{1}{N_{min}}} + N_{min} \tag{6.5}$$

La nuova soglia υ_τ viene calcolata come una media pesata di N_α e N_β:

$$\upsilon_\tau = aN_\alpha + bN_\beta \tag{6.6}$$

L'intuizione che sta dietro ad N_α è chiara: quando α tende ad uno, ossia in media ci sono molte particelle effettive, allora N_α tende a N_{max}, con una legge di tipo lineare, e υ_τ cresce in accordo alla (6.6). Viceversa quando ci sono poche particelle in $\tau - 1$ è giusto che la soglia si adatti e decresca. La funzione non-lineare N_β richiede invece qualche considerazione in più. β rappresenta come N_{eff} varia in $\tau - 1$ ($-1 \leq \beta \leq 1$). Quando β è prossimo allo zero, vuol dire che essendo N_{eff} abbastanza costante, il filtro sta fornendo una buona



(a) (b)

Figura 6.1: Funzioni per il calcolo di N_α (a sinistra) e di N_β (a destra).

approssimazione del *posterior*. Così la soglia viene diminuità in maniera più o meno consistente in accordo all'esponente di β nella (6.5); viceversa se β registra un'alta variabilità di N_{eff} sarà meglio alzare la soglia di *resampling*. Le due funzioni per N_α e N_β sono graficamente riportate in Figura 6.1. Come si può vedere per N_α la legge che muove la soglia è lineare, mentre per N_β è non-lineare. In particolare la scelta di quest'ultima funzione è data dalla possibilità, variando l'esponente di β (che qui viene fissato a $\frac{1}{N_{min}}$), di controllare come la soglia debba essere decrementata quando β tende a zero; infatti l'esponente di β rende più o meno accentuata la strozzatura della cuspide e quindi la relativa sensibilità della soglia ad un andamento costante di N_{eff}. Infine per i due parametri a e b che compaiono nella (6.6), l'euristica qui adottata è di fissare $a = 0.3$ e $b = 0.7$; viene quindi tenuta in maggiore considerazione la variabilità di N_{eff}.

In questo modo si capisce perchè il *resampling*, pur mantenendo la sua natura selettiva, è adattativo. L'idea su cui si basa questo tipo di approccio è abbastanza semplice, ma efficacissima. Nel Capitolo 8 verrà mostrata, nel corso delle simulazioni, la consistenza e il grande contributo in termini di miglior stima che il filtro riesce a dare della reale posa del robot.

Capitolo 7

Implementazione dell'Unscented FastSLAM

La scelta in questa tesi è di non implementare l'algoritmo con le modifiche, definito appunto non semplicemente come FastSLAM, ma come Unscented FastSLAM, su un sistema reale. Infatti il tipo di soluzione che il FastSLAM e l'Unscented FastSLAM offrono per il problema dello SLAM, richiede un forte sforzo prima teorico e poi pratico, al fine di implementare in maniera corretta ed efficace l'algoritmo, per non parlare degli innumerevoli fattori esterni che in un contesto reale farebbero allontanare dalla pura analisi teorica.

7.1 Virtualizzazione dello SLAM

Data la scelta di non considerare un sistema reale, si è quindi dovuto procedere ad una totale virtualizzazione dello SLAM, creando un *environment* e un *robot*. Per quel che riguarda la scelta del software in cui testare il FastSLAM prima e l' Unscented FastSLAM poi, si è scelto *Matlab*, una piattaforma matematica di sviluppo oramai universalmente accettata dalla comunità scientifica, grazie alla sua estrema potenza e flessibilità.

7.1.1 Environment e robot

Per quel che riguarda l'environment, ossia le mappe in cui far muovere il robot, la scelta è ricaduta su l'*indoor* SLAM di grandi dimensioni, il quale è dibattutissimo nel mondo delle ricerca che concerne lo SLAM; infatti in questo tipo di contesto indoor strutturato, il robot non può disporre, essendo all'interno di un luogo chiuso, di alcun supporto di tipo GPS con funzione almeno correttiva. Si è quindi creata *ex novo* un utility in Matlab che permette di generare qualsiasi tipo di ambiente. É infatti possibile costruire le mappe bidimensionali inserendo segmenti, rette, cerchi, entità puntuali (ossia oggetti di piccole dimensioni). Viene inoltre stabilito *a priori* il percorso che il robot deve eseguire, in quanto un algoritmo di SLAM dovrebbe funzionare anche se il robot venisse comandato da entità esterne, quali per esempio un utente o un altro robot, che dispongono di informazioni circa l'ambiente e in funzione di queste comandano il robot che sta effettuando lo SLAM. Il percorso che il robot deve eseguire viene fissato attraverso una serie di *waypoints* (in italiano punti di via), ossia delle coordinate della mappa che il robot deve raggiungere. La traiettoria tra uno *waypoint* e l'altro viene stabilita dal robot attraverso alcune semplici funzioni che controllano il moto. Il robot in genere viene fatto partire da $\langle 0, 0, \alpha_0 \rangle$, dove α_0 è l'*heading* iniziale che viene variato a seconda delle circostanze.

Riguardo al robot, esso viene modellato come un'entità geometrica priva di massa in grado di eseguire controlli (in funzione dei waypoints) che può misurare (sensori interocettivi) e capace di percepire, in qualche modo, l'ambiente che lo circonda (sensori esterocettivi). In particolare per questi ultimi, si immagina che il robot sia dotato di un sistema di misurazione laser che fornisce misure discretizzate dell'orizzonte visibile, quindi in grado di restituire la distanza (range) e l'orientamento (bearing) del bersaglio colpito. I parametri di questo laser virtuale, come la massima distanza (20-30 metri) e la precisione per orientamento e range vengono modellati in modo da imitare gli attuali laser presenti sul mercato e utilizzati per SLAM su sistemi reali. Da ultimo il laser è a 180 gradi field-of-view, ossia si può immaginare che davanti al robot ci sia un angolo piano, o meglio una semicirconferenza con

raggio pari al range; gli oggetti che appunto cadono in quest'ultima vengono percepiti nella loro distanza e orientamento rispetto all'attuale valore dell'*heading* del robot.

7.1.2 Modellazione del moto e della misura

Il robot virtuale si muove su una mappa bidimensionale di moto uniforme in traslazione rettilinea, mentre si muove di un moto rotazionale e traslazionale uniforme quando ruota. Così si riesce a modellare bene un robot reale. Il controllo u_t sarà allora funzione della velocità traslazionale e angolare, ossia:

$$u_t = u_t(v_t, \omega_t, waypoints) \tag{7.1}$$

L'ultimo termine tra le parentesi della precedente indica che il controllo è funzione dei *waypoints* da seguire. Per rendere la simulazione efficace, ossia simile ad un contesto reale, questi valori vengono disturbati con un rumore Gaussiano a valor medio nullo e matrice di covarianza P_t; in genere però la matrice di covarianza è ipotizzata costante e nel seguito ci si riferirà ad essa come P.

Le misure sono di tipo range/bearing dell'obbiettivo colpito. Anche in questo caso esse vengono sporcate con un rumore Gaussiano a media nulla e matrice di covarianza costante R.

7.1.3 Feature e Landmark

Le *features* dell'ambiente sono appunto le caratteristiche che esso presenta, visibili anche all'occhio umano. Si può trattare di muri, angoli, recessi. Queste caratteristiche vengono percepite anche dai sensori esterocettivi, in questo caso dal laser virtuale.

Il landmark è invece l'entità o l'informazione che il robot ingloba circa la mappa mentre esegue lo SLAM. Inoltre sarà sulla base di questi landmark che il robot correggerà la sua posa. L'approccio seguito in questa tesi è quello di

utilizzare le misure ottenute dal laser per generare i landmark della mappa; quindi il landmark si presenta come un'entità geometrica puntuale. In particolare l'equazione del semicerchio del laser viene opportunamente intersecata con le equazioni dell'ambiente (note perchè siamo in un contesto simulativo). Le entità geometriche ottenute (segmenti, archi di cerchio) vengono poi discretizzate e filtrate con un algoritmo di *z-buffering* per renderle consistenti con la realtà, dato che il robot non può per esempio percepire un oggetto coperto da qualcos'altro. Successivamente le misure vengono relativizzate al sistema di riferimento centrato nel robot e orientato con l'attuale valore dell'*heading* α.

7.2 Unscented FastSLAM in MATLAB

Viene qui di seguito riportato l'elenco delle funzioni necessarie per la realizzazione delle metodiche di SLAM suesposte. Questa breve descrizione schematica servirà per comprendere poi come i vari step dell'algoritmo sono stati realizzati. Per brevità vengono riportate alcune delle funzioni (che in totale superano il centinaio). Le funzioni non vengono riportate in ordine alfabetico, ma cecando più o meno di seguire il flusso dell'algoritmo:

particels_inizialization : questa funzione si occupa di inizializzare il filtro.

compute_controls : questa funzione calcola i controlli da impartire al robot in base ai *waypoints*.

predict_true : viene calcolata in base ai controlli la posizione reale del robot.

add_control_noise : i controlli da impartire vengono sporcati con un rumore Gaussiano a media nulla e matrice di covarianza P.

predict_UT : questa funzione calcola l'evoluzione della posa delle particelle in funzione dei controlli rumorosi, ossia il set Ξ_t) sfruttando l'Unscented Transformation (si veda a proposito la sezione 5.1).

Unscented_Transformation : funzione che implementa l'UT.

get_observation : funzione che simula una scansione laser dell'ambiente circostante e restituisce un insieme discreto di punti, tutti potenziali landmark che l'algoritmo potrebbe incorporare.

add_observation_noise : sporca le misurazioni laser in distanza e orientamento con un rumore Gaussiano a media nulla e matrice di covarianza R.

DAU : DAU sta per Data Association Unknown. Questa funzione ha il delicatissimo compito di effettuare le corrette associazioni tra misure e landmark.

compute_importance_factors : questa funzione implementa l'*importance sampling*, cioè calcola i pesi da assegnare ad ogni particella per quantizzare la differenza tra target distribution e proposal distribution.

compute_resmpling_treshold : calcola la soglia di resampling.

add_landmark : effettua se necessario, quindi secondo il nuovo valore di soglia, il resampling delle particelle.

sample_proposal : calcola la proposal distribution e ottiene nuovi campioni da essa.

multivariate_gauss : funzione rilasciata dalla comunità scientifica per il sampling da una Gaussiana.

feature_UKF_update : con le equazioni dell'Unscented Kalman Filter (UKF) viene aggiornata la posizione dei landmark riosservati.

Unscented_Update : Unscented Transformation per l'aggiornamento della posizione dei landmark.

add_landmark : aggiunge un landmark ad una determinata particella.

z_sensor_organizer : filtra le misure del sensore, nel senso che elimina quelle simili e le organizza in sequenze in modo che dopo possano essere congiunte al fine di costituire una mappa chiara e continua; infatti riportanto unicamente le misure dei sensori sulla base della posa stimata

conduce ad una mappa estremamente pesante e poco maneggevole per una successiva opera di analisi (che comunque esula dallo SLAM).

map_building : creazione della mappa sulla base delle misure sensoriali relativizzate alla particella che al tempo t ha maggior peso.

7.2.1 Inizializzazione del filtro

Ogni particella è una struttura dati con posa $\langle x, y, \alpha \rangle$, matrice di covarianza per la posa, landmark $\langle \theta_{n_x}, \theta_{n_y} \rangle$ e la relativa matrice di covarianza associata al landmark. Tutte le particelle del filtro vengono inizializzate nella posa come $\langle 0, 0, 0 \rangle$ e a ciascuna viene assegnato un peso pari a $\frac{1}{M}$.

7.2.2 Predizione via UT

Il calcolo corretto dell'evoluzione delle particelle in accordo alla posa al tempo $t-1$ $s_{t-1}^{[m]}$ è fondamentale per tutto l'andamento dell'algoritmo. Come già spiegato nella sezione 5.1, questo calcolo sfrutta l'UT. La posa di ogni particella viene estesa inglobando anche il rumore di moto. Dato che quest'ultimo è a valor medio nullo, cioè:

$$E[\delta_{t-1}] = E[\delta_{t-1}^{v}, \delta_{t-1}^{a}] = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{7.2}$$

$$s_{t-1}^{[m]} = \begin{bmatrix} s_{t-1}^{x,[m]} & s_{t-1}^{y,[m]} & s_{t-1}^{\alpha,[m]} & 0 & 0 \end{bmatrix}^{\mathsf{T}} \tag{7.3}$$

Ogni punto *sigma* calcolato dalla (7.3) viene fatto passare attraverso la dinamica di moto riscritta come[1]:

$$s_t^{x,[m]} = s_{t-1}^{x,[m]} + (v_t + \delta_t^v) dt \cos(\omega_t dt + \delta_t^a + s_{t-1}^{\alpha,[m]}) \tag{7.4}$$

$$s_t^{y,[m]} = s_{t-1}^{y,[m]} (v_t + \delta_t^v) dt \sin(\omega_t dt + \delta_t^a + s_{t-1}^{\alpha,[m]}) \tag{7.5}$$

$$s_t^{\alpha,[m]} = s_{t-1}^{\alpha,[m]} + (v_t + \delta_t^v) dt \sin(\frac{\omega_t dt + \delta_t^a}{L}) \tag{7.6}$$

Poi come già detto si calcola il nuovo valor medio e covarianza. Questa funzione è di estrema importanza ed è l'unica che interviene sempre nell'algoritmo. Infatti non è detto che il robot osservi sempre (la qual cosa

[1] L corrisponde alla lunghezza delle asse del veicolo.

dipende dal range e dall'ambiente), mentre sicuramente esso sarà sempre in moto. É anche per questo motivo che la corretta propagazione della particelle attraverso il *motion model* ha un grande impatto sull'andamento generale dell'algoritmo.

7.2.3 Data Association Unknown

Il problema dell'associazione dati non nota viene risolto con una metodica *simple gated nearest-neighbour particle-based*. Questa metodica si fonda essenzialmente su uno stimatore ML su base particellare. La complessità di questo tipo di soluzione è $\mathcal{O}(N_t^{[m]})$, dove $N_t^{[m]}$ rappresenta il numero di landmark della particella m al tempo t. Per ogni osservazione vengono calcolate due distanze tra questa osservazione e la posizione dei landmark associati alla particella. Questi valori vengono comparati con una doppia soglia in modo da decidere se l'osservazione va associata ad un landmark già presente nel filtro, va ignorata oppure può creare un nuovo landmark. Le due distanze che vengono calcolate sono la distanza normalizzata e la distanza di Mahalanobis. In formule:

$$d_{Mahalanobis} = (z_t - \hat{z}_t)^{\mathsf{T}} S_f^{-1} (z - \hat{z}_t) \tag{7.7}$$

$$d_{Normalizzata} = d_{Mahalanobis} + \log(\det(S_f)) \tag{7.8}$$

dove S_f è pari a:

$$S_f = G_\theta \; P_f \; G_\theta^{\mathsf{T}} + R \tag{7.9}$$

con P_f pari alla matrice di covarianza del landmark. Considerando una particella con la sua posa $s_t^{[m]}$ e un landmark associato ad essa θ_n, si definisce la seguente funzione che rappresenta la predizione di misura:

$$z = g(\theta_n, s_t^{[m]}) = \begin{bmatrix} \sqrt{(\theta_{n,x} - s_t^{x,[m]})^2 + (\theta_{n,y} - s_t^{y,[m]})^2} \\ \tan^{-1} \frac{(\theta_{n,y} - s_t^{y,[m]})}{(\theta_{n,x} - s_t^{x,[m]})} \end{bmatrix} \tag{7.10}$$

Quindi $z - \hat{z}_t$ esprime l'innovazione di misura; G_θ è lo Jacobiano relativo alla distanza della *feature* selezionata rispetto agli stati del landmark:

$$\begin{aligned} G_\theta &= \begin{bmatrix} \frac{\partial g_1}{\partial \theta_n} \\ \frac{\partial g_2}{\partial \theta_n} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial g_1}{\partial \theta_{n,x}} & \frac{\partial g_1}{\partial \theta_{n,y}} \\ \frac{\partial g_2}{\partial \theta_n} & \frac{\partial g_2}{\partial \theta_{n,y}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{-(s_t^{x,[m]} - \theta_{n,x})}{\sqrt{(\theta_{n,x} - s_t^{x,[m]})^2 + (\theta_{n,y} - s_t^{y,[m]})^2}} & \frac{-(s_t^{y,[m]} - \theta_{y,x})}{\sqrt{(\theta_{n,x} - s_t^{x,[m]})^2 + (\theta_{n,y} - s_t^{y,[m]})^2}} \\ \frac{(s_t^{y,[m]} - \theta_{n,y})}{\sqrt{(\theta_{n,x} - s_t^{x,[m]})^2 + (\theta_{n,y} - s_t^{y,[m]})^2}} & \frac{-(s_t^{x,[m]} - \theta_{n,x})}{\sqrt{(\theta_{n,x} - s_t^{x,[m]})^2 + (\theta_{n,y} - s_t^{y,[m]})^2}} \end{bmatrix} \end{aligned} \tag{7.11}$$

Incorporata un'osservazione per l'n-esimo landmark, dopo aver fissato due soglie, GATE_ABS e GATE_NEW, viene calcolata la distanza di Mahalanobis. Se questa è minore di GATE_ABS allora il landmark viene associato. Se tale valore è compreso tra le due soglie il l'osservazione non viene associata al landmark, mentre se la distanza è maggiore di GATE_NEW allora l'osservazione genera un nuovo landmark. L'associazione di dati viene localizzata all'attuale posizione del robot e del campo visivo. É inutile infatti provare ad associare landmark lontani dal robot.

7.2.4 Sampling dalla proposal distribution

Per definire nuova posa e matrice di covarianza per ogni particella si campiona da una Gaussiana avente come valor medio appunto $s_t^{[m]}$ e matrice di covarianza $\Sigma_t^{[m]}$; tali valor medio e matrice di covarianza vengono calcolate in maniera incrementale sulla base delle *feature* riosservate per l'm-esima particella. Si calcolano quindi per l'n-esimo landmark riosservato, l'innovazione di misura, i Jacobiani J_θ e G_s, la matrice $H_t^{[m]}$. Si campiona la nuova posa da una Gaussiana a valor medio matrice di covarianza pari a:

$$\mu_{s_t}^{[m]} = \Sigma_{s_t}^m G_s^\mathsf{T} H_t^{[m]-1} (z_t - \hat{z}_t^{[m]}) + \hat{s}_t^{[m]} \tag{7.12}$$

$$\Sigma_{s_t}^{[m]} = \left[G_s^\mathsf{T} H_t^{[m]-1} G_s + P_t^{-1} \right]^{-1} \tag{7.13}$$

7.2.5 Calcolo degli Importance Factor

Il calcolo dei pesi, per ogni particella, viene effettuato prima calcolando la predizione di misura, S_f come nella (7.9) e poi G_s, ossia lo Jacobiano di g rispetto alla posa, come:

$$\begin{aligned} G_s &= \begin{bmatrix} \frac{\partial g_1}{\partial s_t^{[m]}} \\ \frac{\partial g_2}{\partial s_t^{[m]}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial g_1}{\partial s_t^{x,[m]}} & \frac{\partial g_1}{\partial s_t^{y,[m]}} & \frac{\partial g_1}{\partial s_t^{\alpha,[m]}} \\ \frac{\partial g_2}{\partial s_t^{x,[m]}} & \frac{\partial g_2}{\partial {s_t}^{y},[m]} & \frac{\partial g_2}{\partial s_t^{\alpha,[m]}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{s_t^{x,[m]}-\theta_{n,x}}{\sqrt{(\theta_{n,x}-s_t^{x,[m]})^2+(\theta_{n,y}-s_t^{y,[m]})^2}} & \frac{s_t^{y,[m]}-\theta_{y,x}}{\sqrt{(\theta_{n,x}-s_t^{x,[m]})^2+(\theta_{n,y}-s_t^{y,[m]})^2}} & 0 \\ \frac{-(s_t^{y,[m]}-\theta_{n,y})}{\sqrt{(\theta_{n,x}-s_t^{x,[m]})^2+(\theta_{n,y}-s_t^{y,[m]})^2}} & \frac{s_t^{x,[m]}-\theta_{n,x}}{\sqrt{(\theta_{n,x}-s_t^{x,[m]})^2+(\theta_{n,y}-s_t^{y,[m]})^2}} & -1 \end{bmatrix} \end{aligned} \tag{7.14}$$

Si calcola poi la seguente quantità, nota come covarianza dell'innovazione, che include l'incertezza sulla posizione del robot e sulle *feature*:

$$S = G_s \Sigma_{s_t}^{[m]} G_s^{mathsfT} + S_f \tag{7.15}$$

Il peso di ogni particella sarà pari a:

$$w = w_{t-1} p(z_t | s_{t-1}) \tag{7.16}$$

dove:

$$p(z_t | s_{t-1}) = \frac{\exp(\frac{-1}{2}(z_t - \hat{z}_t)^{\mathsf{T}} S^{-1}(z - \hat{z}_t))}{2\pi\sqrt{\det S}} \tag{7.17}$$

7.2.6 Flusso dell'algoritmo

Il FastSLAM entra in funzione solo quando si ha a disposizione una nuova z_t, ossia quando il laser importa nuove misurazioni. Se questo non accade, è la funzione di predizione, oggetto di una delle modifiche fatte in questa tesi (**predict_UT**), che si occupa di stimare la posizione del robot, non essendovi informazioni circa l'ambiente esterno. Proprio la maggiore accuratezza con questa funzione stima il *path* del robot permette al FastSLAM di non divergere qualora non siano disponibili misure per un certo intervallo di tempo. Nel codice questa sempre non possibile possibilità di osservare,

viene modellata con un contatore fisso, ma che può essere anche variabile. Sulla base del valore di questo contatore, il robot osserva e viene avviato il processamento vero e proprio del FastSLAM. La mappa viene costruita mentre il robot è in moto, senza alcun tipo di aggiustamento dopo che il robot ha terminato lo SLAM. Infatti algoritmi di analisi successive sulle misure importate esulano dal vero e proprio scopo del FastSLAM, che è quello di costruire una mappa attendibile mentre il robot sta perlustrando l'ambiente e non dopo. La mappa viene quindi costruita semplicemente filtrando le misure del laser (in maniera opportuna con la funzione **z_sensor_organizer)** e generando la mappa sulla base della particella che al tempo t ha maggior peso.

Capitolo 8

Risultati simulativi

In questo Capitolo vengono mostrati i risultati delle simulazioni che confrontano le prestazioni del FastSLAM classico con l'Unscented FastSLAM oggetto di questa tesi. Dopo una breve introduzione ai criteri che misurano la bontà di un algoritmo di SLAM, vengono appunto mostrati i risultati ottenuti e la ricostruzione di un ambiente. Le mappe in cui si è provata la robustezza e la consistenza delle modifiche di cui si è parlato, sono state innumerevoli e, per ognuna di esse, si sono eseguite moltissimie simulazioni, in cui venivano cambiati variati i parametri quali, per esmpio, quelli per l'associazione di dati o per l'errore di moto e di misura. Inoltre viene quantitativamente mostrata l'efficacia e i benefici dell'Adaptive Selective Resampling.

8.1 Mappa traiettoria e loop

La mappa costruita per le simulazioni imita quella reale di un piano del CMU-Newell-Simon Hall (si veda a proposito radish.sourceforge.net). Questo per la particolare complessità e grandezza della mappa. Infatti come si può vedere dalla Figura 8.1 l'ambiente in cui il robot si muove è molto complesso e ricco di recessi, angoli, zone in ombra, spigoli. Inoltre la dimensione della mappa è molto grande come si può vedere e al robot vengono fatti percorrere circa 0.7 km. I cerchi rossi sono i *waypoint*; la traiettoria sarà simile a quella in blu nella mappa, ma non del tutto, in quanto la linea blu serve solo a collegare i

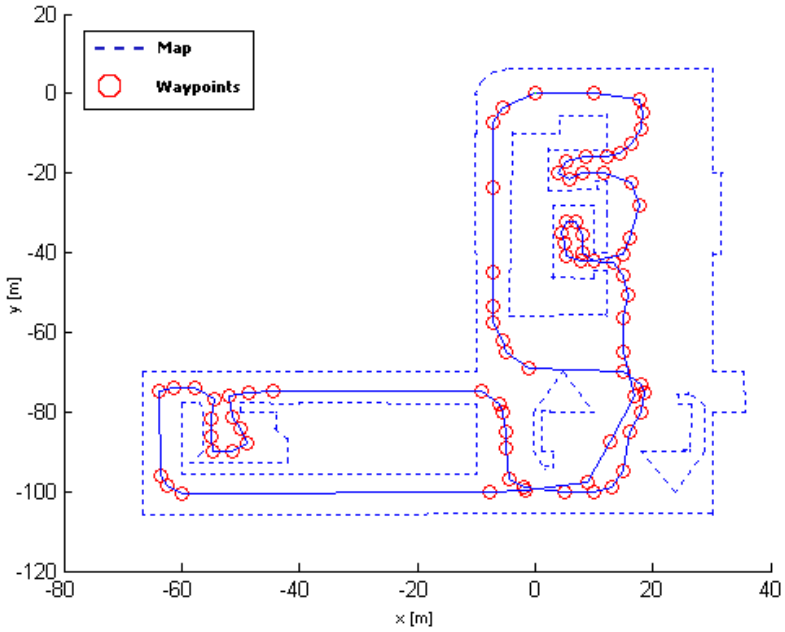
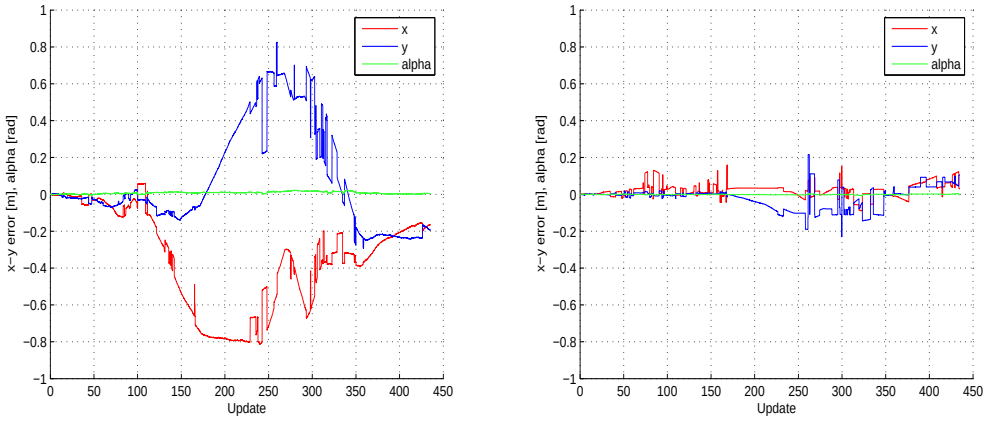


Figura 8.1: Mappa (linea blu tratteggiata) e waypoints (cerchietti rossi). La linea blu continua collega semplicemente i punti di via in modo da disegnare una traiettoria molto simile a quella che verrà imposta al robot.

punti di via.

Il robot parte dalla posizione $s_0 = \langle 0, 0, 0 \rangle$ e torna in questa posizione alla fine del suo giro. Come infatti si può vedere dalla Figura 8.1 la traiettoria non è aperta, ma è un loop. Infatti uno dei primi requisiti di un buon algoritmo di SLAM è che il robot riesca a chiudere il *loop*, ossia a tornare nella stessa posizione di partenza dopo aver esplorato l'ambiente e riosservare le *feature* che aveva osservato all'inizio. Questo è possibile solo se l'errore sulla posa si è mantenuto limitato; infatti solo in questo caso le feature riosservate non saranno interpretate come nuove e il robot potrà ulteriormente correggere la sua posa.



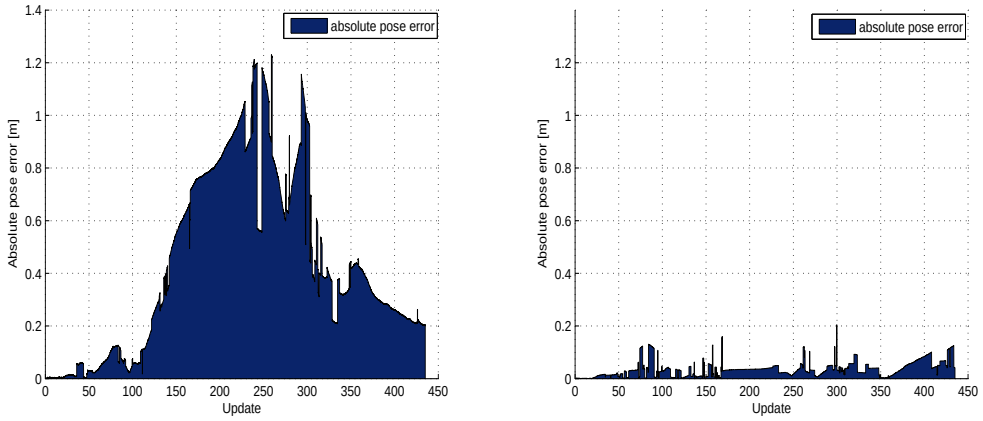
(a) (b)

Figura 8.2: Confronto tra l'errore su $\langle x, y, \alpha \rangle$ del FastSLAM classico (a) e dell'Unscented FastSLAM implementato con l'Adaptive Selective Resampling (b).

8.2 Standard FastSLAM vs Unscented FastSLAM

I due algoritmi sono stati testati nella mappa di Figura 8.1 con $M = 60$ particelle e con gli stessi paramentri per altre grandezze. I criteri con cui vengono comparati i due algoritmi sono l'errore sulla posa in tutte le sue componenti $\langle x, y, \alpha \rangle$, ossia l'errore che si sta commettendo stimando ciascuna componente dello stato (la posa). Viene poi considerato l'errore assoluto tra la posizione stimata e la posizione reale (absolute error) e il quadrato della precedente quantità (il quadrato permette di penalizzare maggiormente i grandi errori compiuti dal filtro). Si calcolano poi il MAE (Mean Absolute Error) e l'RMSE (Root Mean Square Error) per i due algoritmi. Da ultimo viene considerata come misura significativa a sè stante l'errore assoluto di posizione quando il robot chiude il loop. Questi criteri nel loro insieme hanno dimostrato come la tecnica del FastSLAM, perfezionata in questo lavoro, dia ottimi risultati in termini di precisione e maggiore robustezza.

La Figura 8.2 mostra il confronto tra i due algoritmi per quel che riguarda l'errore sulla stima della terna $\langle x, y, \alpha \rangle$. Come si può vedere i miglioramen-



(a) (b)

Figura 8.3: Errore in valore assoluto tra posizione reale e posizione stimata con il FastSLAM (a) e con l'Unscented FastSLAM (b).

ti introdotti con l'Unscented FastSLAM sono significativi. La successiva Figura 8.3 mostra invece in confronto tra l'errore in modulo sulla distanza euclidea tra posizione reale e posizione stimata. Anche qui si può vedere come l'Unscented FastSLAM riduca l'errore, rispetto al semplice FastSLAM, in maniera drastica; in particolare l'errore è ridotto di circa un decimo di grandezza. L'errore più grande compiuto è di circa 0.2 cm per l'Unscented FastSLAM, mentre di circa 1.4 m per lo standard FastSLAM. In circostanze come questa, mentre l'Unscented FastSLAM costruisce una mappa consistente con quella reale, il semplice FastSLAM compie un forte errore sulla mappa. Il robot, nel caso dell'algoritmo modificato chiude il loop con una MAE di 0.0349 m e con un RMSE di 0.0755. Con il FastSLAM standard invece il robot chiude il loop con MAE 0.4289 m e un RMSE pari a 0.2716.

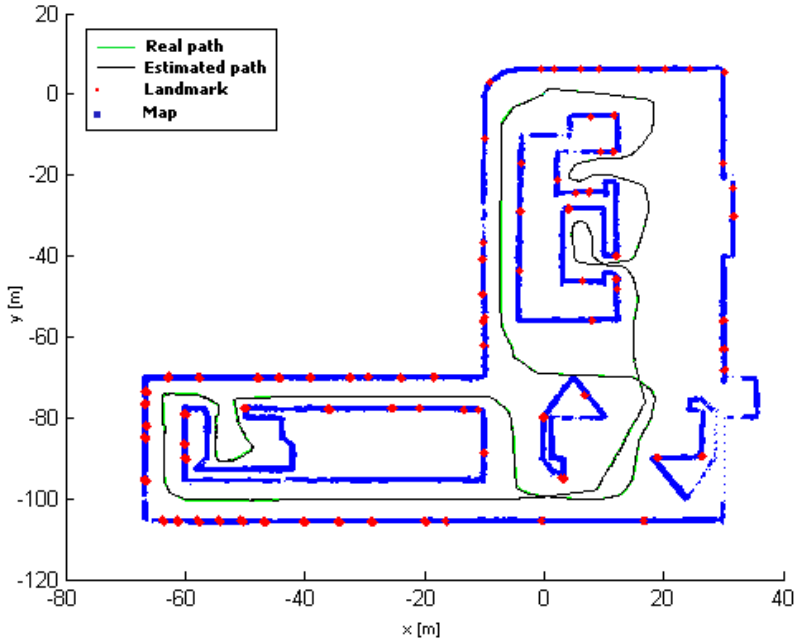
Le successive due Figure mostrano la mappa e il *path* (reale e stimato) del robot che esegue lo SLAM nell'ambiente descritto dalla Figura 8.1. Vengono riportati anche i landmark estratti dall'algoritmo (in rosso), mentre la mappa viene costruita relativizzando le misure sensoriali (in blu) sulla particella di maggior peso (che rappresenta quindi la miglior stima che il filtro riesce a fare circa la posizione del robot). Le misure dei sensori non vengono

semplicemente riportate cosi come sono, ma vengono filtrate da un algoritmo (**z_sensor_organizer**)che le organizza nel campo visivo, filtra quelle inutili (che sono innumerevoli se il robot si muove lentamente) e vengono interpolate, ossia unite (dopo essere state organizzate in sequenza) per generare una mappa meno caotica e più continua.

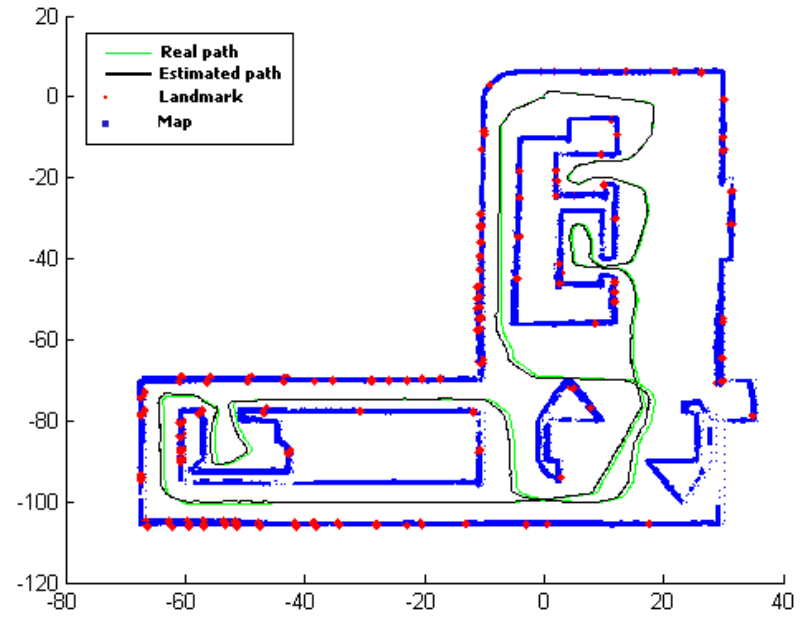
Come si vede dalle figure nel caso dell'Unscented FastSLAM il *path* reale (in verde) e il *path* stimato (in nero) sono quasi sempre sovrapposti e il robot torna nella posizione di partenza con un errore assoluto di 0.0414 m; il FastSLAM classico a volte fallisce sensibilmente nella stima del *path* e nonostante la robustezza (comunque chiude il loop) torna nella posizione di partenza con circa un errore assoluto di 0.2040. Come si vede entrambi gli algoritmi all'opera sono estremamente potenti e generano una mappa consistente dell'ambiente facendo si che il robot mantenga sempre con elevata precisione l'idea di dove è nella parte di mappa che ha già costruito. Tuttavia nel caso dell'Unscented FastSLAM la mappa può essere ingrandita fino ad un 1cm di risoluzione senza notare significativi, mentre nella mappa generata dal semplice FastSLAM ci si può spingere fino ad 1 m circa e poi l'errore diventa molto evidente.

Vengono riportati degli ingrandimenti della mappa ricostruita dagli algoritmi. L'Unscented FastSLAM è notevolmente più preciso nella stima del *path* come si evince dalla Figura 8.5 (Frames (a) e (c) a confronto). Inoltre l'Unscented FastSLAM stima meglio le *feature* della mappa (Frames (b) e (d) sempre della 8.5) e produce gruppi di *landmark* più compatti, dato che le particelle sono molto più vicine nella loro evoluzione. La mappa riportata in blu è semplicemente il set di misure al tempo t relativizzate alla posa del robot stimata (posa della particella di maggior peso); data la maggiore precisione dell'Unscented FastSLAM nell'approssimazione del *posterior*, ad ogni time step la particella di maggior peso sarà molto più vicina alla posa del reale del robot, rispetto a quanto non faccia il semplice FastSLAM. Così la mappa ricostruita è molto più precisa, come si osserva dagli ingrandimenti.

Per quel che riguarda i costi computazionali, quelli del FastSLAM sono

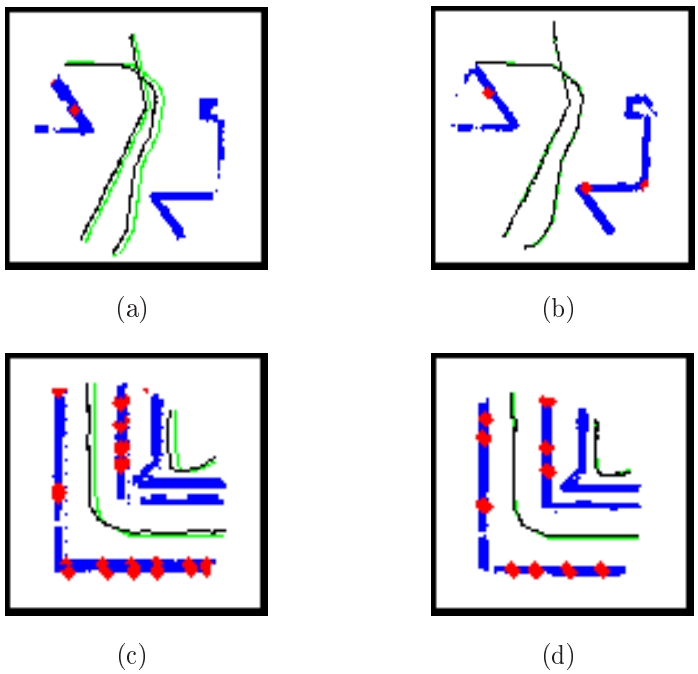


(a)



(b)

Figura 8.4: Mappa finale 2-D generata con il FastSLAM classico e con l'Unscented FastSLAM insieme a l'Adaptive Selective Resampling.

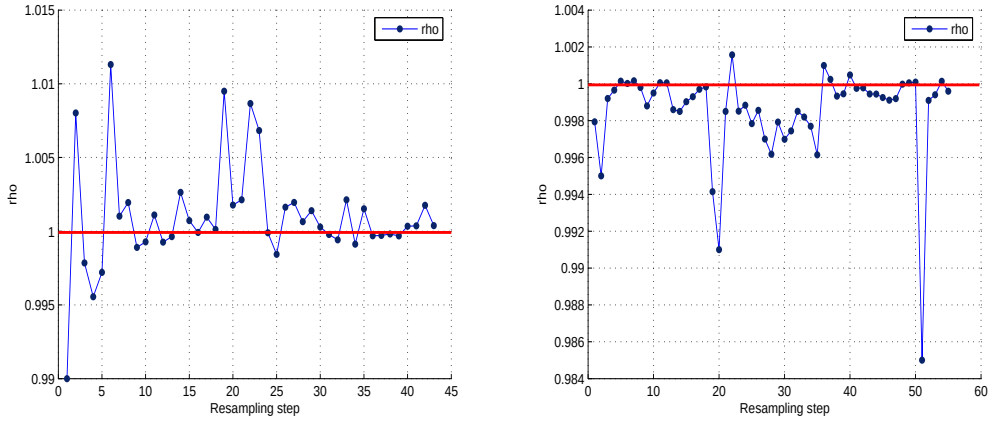


(a) (b)

(c) (d)

Figura 8.5: Zoom su due zone significative della mappa ricostruita dagli algoritmi. Nei Frames (a) e (c) si può vedere l'errore copmpiuto dal FastSLAM, mentre nei Frames (b) e (d) ciò che riesce a fare l'Unscented FastSLAM. Si noti la maggior accuratezza di quest'ultimo, sia nella stima del *path* che è notevolmente più precisa che della mappa.

ben noti ed è inutile qui ripeterli di nuovo. L'Unscented FastSLAM invece implica aggiungere per ogni particella un insieme di punti sigma (che vanno inoltre di volta in volta calcolati) che rendeno l'algoritmo notevolmente più pesante. In particolare infatti, se la semplice predizione nel FastSLAM ha costo $\mathcal{O}(M)$, nell'Unscented FastSLAM essa ha costo $\mathcal{O}(\sigma_p \cdot M)$, dove σ_p è il numero di punti sigma e M è il numero di particelle. Tuttavia si è visto è meglio diminuire il numero di particelle nel filtro, ma continuare a proteggerle nella predizione con l'UT.



(a) (b)

Figura 8.6: Confronto dell'andamento di ρ nel caso di resampling solo selettivo e nel caso di Adaptive Selective Resampling.

8.3 Efficacia dell'Adaptice Selective Resampling

Ci sono vari modi per quantificare l'efficacia e la consistenza dello step di resampling di un filtro particellare. L'approccio in questa tesi è quello di considerare il rapporto tra l'errore assoluto tra posa reale e posa stimata nello step successivo al resampling rispetto a quello nello step immediatamente precedente al resampling. In formule:

$$\rho = \frac{e_{tr}}{e_{tr-1}} \tag{8.1}$$

Con l'Adaptive Selective Resampling tale rapporto è sempre minore di uno, in quanto vi è una riduzione dell'errore sulla posa. Il semplice *selective resampling* spesso è inconsistente e poco tempestivo, incrementando l'errore sulla stima della posa. L'Adaptive Selective Resampling riduce il rischio del cosiddetto *ill-timed resampling*. La Figura mostra appunto l'andamento della grandezza ρ della (8.1) durante lo SLAM sulla mappa di Figura 8.1. In ascissa viene riportato il tempo di *resampling*, mentre in ordinata la grandezza ρ. Come si può vedere nel caso dell'Adaptive Selective Resampling ρ è sempre sotto la linea rossa continua che segna l'unità, quindi la maggior parte (quasi

tutti) i resampling sono effficaci. Nel caso del solo resampling selettivo molti valori di ρ sono sopra l'unità. I picchi più alti indicano che oltre ad un aumento di errore, quest'ultimo è molto forte, mentre nel caso delle Figura a sinistra questi picchi sono minori dell'unità, e indicano quindi forti correzioni di posa.

Conclusioni e sviluppi futuri

Questa tesi nasce con l'intento di modificare alcuni punti deboli di un algoritmo estremamente potente che costituisce lo stato dell'arte dello SLAM. Tecniche di tipo Unscented hanno reso più consistente l'evoluzione delle particelle che modellano il reale *path posterior* del robot. L'EKF (Extended Kalman Filter) dell'algoritmo classico è stato sostituito con l'UKF (Unscented Kalman Filter), che permette di evitare metodiche di linearizzazione in alcuni casi estremamente poco significative in termini di approssimazione. Si è inoltre introdotto un innovativo tipo di *resampling*, considerando in generale il grande impatto che questo step ha su tutto l'andamento dell'algoritmo. Inoltre, attraverso la virtualizzazione dell'ambiente e del robot, ci si è messi in un contesto in cui era possibile studiare le caratteristiche dei due algoritmi a confronto al di là di tutti quei disturbi imprevedibili e non modellabili che intervengono in un esperimento reale.

Nel futuro bisognerà testare il nuovo algoritmo, funzionante da un punto di vista teorico-simulativo, su un sistema reale, cercando di ottimizzarlo nell'ottica di una implementazione in real-time.

Ringraziamenti

In collaborazione con il Relatore della qui presente tesi, Ing Francesco Martinelli, si è scritto un articolo proprio su queste modifiche e lo si è sottomesso all'ICRA2007 (IEEE International Conference on Robotics and Automation). Colgo quindi l'occasione per ringraziare il Relatore della fiducia e dell'aiuto.

Bibliografia

[1] J. Andrade-Cetto, A. Sanfeliu. Environment Learning for Indoor Mobile Robots. A Stochastic State Estimation Approach to Simultaneous Localization and Map Building Series: Springer Tracts in Advanced Robotics, Vol. 23, 2006.

[2] J. Andrade-Cetto, T. Vidal-Calleja, A. Sanfeliu. Unscented Transformation of Vehicle States in SLAM Robotics and Automation. ICRA 2005. In *Proceedings of the 2005 IEEE International Conference on* 18-22 April 2005 Page(s):323 - 328

[3] M.S. Arulampalam, S. Maskell, N. Gordon, T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking Signal Processing. In *IEEE Transactions on Acoustics, Speech, and Signal Processing* Volume 50, Issue 2, Feb. 2002 Page(s):174 - 188.

[4] M. Cugliari, F. Martinelli. A FastSLAM algorithm based on the Unscented Filtering with adaptive selective resampling (*submitted* ICRA2007).

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[6] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. In *IEEE Trans. on Robotics and Automation* 17(3):229-241, 2001.

[7] A Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Departement of Engeneering, University of Cambridge, 1998.

[8] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, Stanford, CA, USA, 2000.

[9] G. Galati, G. Pavan. Teoria dei fenomeni aleatori, volume 1 & 2, Texmat 2002.

[10] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling, In *IEEE International Conference on Robotics & Automation* (ICRA), 2005.

[11] J.S. Julier, J.K. Uhlmann. Unscented filtering and non linear estimation, *Proc. IEEE*, vol. 92, no. 3, pp. 401-422, Mar. 2004.

[12] J.S. Julier. The spherical simplex Unscented Transformation. In*Proc. American Control Conf.*, Denver, Jun. 2003.

[13] J.S. Julier, J.K. Uhlmann, and H. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Trans. on Automatic Control*, 45(3):477-482, March

[14] J.S. Julier, J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems, In *Proc. 11th SPIE Int. Sym. Aerospace/Defense Sensing, Simulation, Controls*, I. Kadar, Ed. Orlando: SPIE, Apr. 1997, pp. 182-193.

[15] J.S. Julier, J.K. Uhlmann and H. F. Durrant-Whyte. A New Approach for Filtering Nonlinear Systems. In*Proceedings of the American Control Conference*, Seattle, Washington., pages 1628-1632, 1995.

[16] J.S. Julier, J.K. Uhlmann. Corrections to Unscented Filtering and Nonlinear Estimation. In *Proceedings of the IEEE* Volume 92, Issue 12, Dec 2004 Page(s):1958 - 1958.

[17] J.S. Julier, J.K. Uhlmann. Reduced sigma point filters for the propagation of means and covariances through non linear transformations. In *Proceedings of the American Control Conference*, 2002. Volume 2, 8-10 May 2002 Page(s):887 - 892 vol.2

[18] J.S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statist. Comput., 6:113-119, 1996.

[19] T. Lefebvre, H. Bruyninckx, Joris De Schutter. Comment on A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators, In *IEEE Transactions on Automatic Control.*

[20] R. Kalman. A new approach to linear filtering and prediction problems. In *Transactions of the ASME-Journal of Basic Engineering*, 82(Series) D9:35-45, 1960.

[21] W. Madow. On the theory of systematic sampling, ii. Annals of Mathematical Statistics, 20:333-354, 1949.

[22] R. Martinez-Cantin, J.A. Castellanos. Unscented SLAM for Large-Scale Outdoor Environments. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on* 2-6 Aug. 2005 Page(s): 3427-3432.

[23] R. Van der Merwe, E. Wan. The square-root unscented kalman filter for state and parameter estimation, In *Proceedings of the International Conference on Accoustics, speech, and Signal Processing*(ICASSP), 2001.

[24] R. Van der Merwe, E. Wan. The unscented kalman filter for nonlinear estimation. In *Proceedings of IEEE Symposium*, 2000.

[25] R. Van der Merwe, N. De Freitas, A. Doucet, E. Wan. The Unscented Particel Filter. In *Advances in Neural Information Processing Systems 13*, 2001.

[26] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. AAAI-02.

[27] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence*(IJCAI), 2003.

[28] M. Montemerlo, S. Thrun. Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM (2003). In *IEEE International Conference on Robotics and Automation*, vol. 2, September 2003.

[29] M. Montemerlo. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association, Ph.D. Thesis, School of Computer ScienceCarnegie Mellon University, Pittsburgh.

[30] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems* (NIPS). MIT Press, 1999.

[31] P. Newman. On the Structure and Solution of the Simultaneous Localisation and Map Building Problem. PhD thesis, Univ. of Sydney, 2000.

[32] F. Romanelli. Filtri particellari per la localizzazione e navigazione di robot mobili in ambienti non noti privi di landmark, Tesi di Laurea in Ingegneria Informatica, Università degli Studi di Roma, Tor Vergata.

[33] D. B. Rubin. Bayesian Statistics 3. Oxford University Press, 1988.

[34] L. Spinello. Risoluzione al problema dello SLAM tramite filtri particellari e filtro di Kalman Esteso, Tesi di Laurea in Ingegneria dell'Automazione, Università degli Studi di Roma, Tor Vergata.

[35] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association, *Journal of Machine Learning Research*, 2004. in press.

[36] S. Thrun, D. Fox, W. Burgard and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 2001. 128:99-141.

[37] S. Thrun, Particle Filters in Robotics. In *Proc. of the the 17th Annual Conf. on Uncertainty in AI* (UAI), 2002.

[38] G. Welch and G. Bishop. An introduction to the kalman filter. ACM SIGGRAPH Tutorial, 2001.