UNIVERSITÀ DEGLI STUDI DI TRIESTE

FACOLTÀ DI INGEGNERIA

Dipartimento di Elettrotecnica, Elettronica ed Informatica

Tesi di Laurea in

SISTEMI OPERATIVI

Fusione sensoriale con aritmetica a precisione finita mediante algoritmi di filtraggio lineare ricorsivo nella robotica mobile

Laureando:

Alberto CODOGNOTTO

Relatore:

Prof. Ing. Enzo MUMOLO

Correlatore:

Dott. Ing. Massimiliano NOLICH

Anno Accademico 2003–04

Ogni minuto è un'occasione
per sconvolgere tutta la vita!

A tutta la mia famiglia...

Indice

Capitolo 1

Introduzione

Il termine *robot* deriva dalla parola cèca *robota*, che significa lavoro[1]. I robot, che si presentano in forme bizzare, buffe, spaventose o più semplicemente curiose e molto spesso simili ad esseri viventi, hanno come scopo primario quello di compiere un lavoro.

La necessità di automatizzare processi industriali, civili e militari ha fornito l'occasione alla robotica di acquistare ogni giorno ruoli di importanza sempre maggiore.

I robot sono utilizzati in molti ambiti, diversi fra loro:

- nell'assemblaggio;
- nella verniciatura;
- nel controllo ambientale;
- in campo medico;
- come navette trasportatrici;
- come sonde in ambienti difficili;
- in sostituzione dell'uomo in situazioni pericolose;

[1] Con il termine *robot* nel 1920 lo scrittore cèco Karel Čapek denominò in una sua commedia (RUR, Rossum's Universal Robot) un androide, costruito da uno scienziato, in grado di compiere tutti i lavori normalmente eseguiti da un essere umano, ossia un tentativo di riproduzione artificiale della macchina umana. A tale finzione letteraria va ricondotto il significato più stretto del termine. [1]

- per compiere lavori ripetitivi e noiosi;
- nel gioco;
- nello spolverare;
- nel tagliare l'erba.

La capacità della macchina di spostarsi in modo autonomo è di fondamentale importanza in alcuni di questi casi. Non è difficile immaginare che per potersi muovere chiunque ha l'esigenza di conoscere la posizione di partenza e quella d'arrivo, nonchè dove si trova durante lo spostamento.

La localizzazione di un robot mobile rappresenta, ancora oggi, uno dei problemi di maggior interesse. Esistono molte tecnologie e tecniche per localizzare un oggetto in movimento. Ogni sistema produce una stima approssimata della posizione che occupa il veicolo in un dato istante, ognuna di queste stime non è precisa, quindi affetta da errore. In genere, più l'intervallo di confidenza della misura è limitato, più il sistema di rilevazione utilizzato è costoso.

La tendenza odierna è di utilizzare più sistemi di localizzazione contemporaneamente e fondere i risultati dei vari dispositivi al fine di diminuire l'incidenza dell'errore sulla misura complessiva. Questo permette di sfruttare sistemi di localizzazione poco costosi, ottendo risultati ragionevoli.

I sistemi per realizzare *fusione sensoriale* studiati sin ora si basano essenzialmente su reti neurali, algoritmi fuzzy e sul filtro di Kalman. Tuttavia la ricerca in questo campo non si può definire conclusa, ognuno di questi metodi nasconde dei problemi:

le reti neurali prevedono un addestramento ed hanno problemi di convergenza;

i filtri di Kalman sono contraddistinti da una complessità computazionale non trascurabile, hanno problemi in alcuni casi ad associare correttamente le informazioni acquisite e dimostrano un'incapacità di gestire le forti perturbazioni; inoltre la teoria dei filtri di Kalman prevede che il rumore filtrabile sia di tipo Gaussiano a media nulla;

gli algoritmi fuzzy prevedono una modellizzazione attraverso regole linguistiche.

Lo scopo del seguente elaborato è appunto quello di individuare nuovi algoritmi di fusione sensoriale, basandosi sulla tecnica di fusione che sfrutta il filtro di Kalman. Nel 1994 studi condotti da Ali H. Sayed e Thomas Kailath [3] hanno

dimostrato l'esistenza di una corrispondenza diretta tra i filtri di Kalman e i filtri RLS. Quest'ultimi possiedono capacità di convergenza maggiori del filtro di Kalman e necessitano di un costo computazionale inferiore. Nel 1999 Enzo Mumolo e Alberto Carini [5] hanno pubblicato degli algoritmi RLS in grado di convergere in aritmetica finita sfruttando un numero limitato di operazioni. Lo scopo di questa tesi è realizzare e studiare nuovi algoritmi di fusione sensoriale, basati sul filtro di Kalman, che sfruttino gli algoritmi RLS, nella speranza che ereditino da quest'ultimi i pregi di cui sopra. Quindi in generale, si vuole ottenere un filtro per fusione sensoriale con prestazioni migliori del semplice filtro di Kalman.

Nel robot utilizzato per lo studio degli algoritmi, la fusione avviene nel seguente modo: data la posizione iniziale, attraverso tecniche odometriche, si ricava la traiettoria del veicolo. Nello spostamento quest'ultimo utilizza dei sensori ad ultrasuoni per individuare gli ostacoli attorno a sè, queste rilevazioni permettono di correggere la posizione individuata dall'odometria attraverso gli algoritmi sudetti.

Non avendo a disposizione direttamente un robot operativo, tutti gli studi sono stati condotti servendosi di un simulatore, che rispecchia fedelmente il comportamento e le percezioni del robot reale.

Il lavoro di tesi svolto è stato condotto presso l'azienda A.I.B.S. lab - **A**rtificial **I**ntelligence **B**rain **S**torming (www.aibs-lab.com), con sede presso il B.I.C. di Trieste.

Capitolo 2

Sensori per la localizzazione di sistemi robotici mobili

In questo capitolo si presentano i diversi sensori utilizzabili nel processo di localizzazione, classificandoli in base al tipo di informazione che sono in grado di produrre:

- sensori di spostamento relativo: forniscono rilevazioni relative allo spostamento compiuto dal robot in un intervallo di tempo
- sensori di prossimità: effettuano misurazioni su oggetti presenti nell'ambiente circostante

La scelta, poi, di utilizzare un sensore al posto di un altro dipenderà da vari fattori: caratteristiche dell'ambiente, metodo di localizzazione, precisione dei dati sensoriali, costo dei sensori e tipo di compito da effettuare.

2.1 Sensori di spostamento relativo

A questa classe di sensori appartengono quei dispositivi che forniscono informazioni concernenti le variazioni d'orientazione o di posizione del robot, senza rilevare alcuna informazione topologica dall'ambiente circostante. Tutti i sensori di questo tipo, effettuano esclusivamente misure cinematiche o dinamiche rilevabili a bordo del robot.

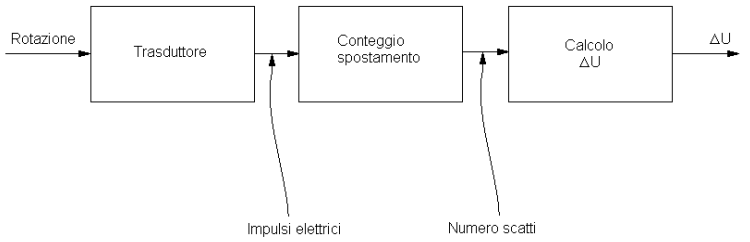

Figura 2.1: Sequenza del processo odometrico

2.1.1 Odometria

L'odometria è un metodo d'analisi dei dati sensoriali che, a fronte della conoscenza di

- cinematica del robot
- rotazione compiuta dalle ruote nell'intervallo di tempo $\Delta T_i = (t_i - t_{i-1})$

permette di ottenere lo spostamento compiuto dal robot nell'intervallo considerato. Esprimendo come $\Delta\rho_i$ la variazione di posizione lineare, con $\Delta\theta_i$ la variazione di orientazione, definiamo il vettore di spostamento relativo compiuto in ΔT_i come $\Delta U = (\Delta\rho_i \Delta\theta_i)_T$.
L'intero processo odometrico puo' essere schematizzato come in fig.2.1.
L'informazione relativa alla rotazione è convertita in grandezze elettriche, mediante il trasduttore di rotazione. Queste grandezze sono poi elaborate da uno stadio che ha il compito di determinare la rotazione relativa o assoluta compiuta da ogni ruota la cui informazione risulta rilevante. Infine in uno stadio di calcolo, nota la cinematica del robot, si è in grado di calcolare la grandezza ΔU.

2.1.2 Sensori di rotazione - encoder ottici

Ci sono due tipi di encoder ottici: incrementali ed assoluti. L'encoder incrementale misura la velocita' rotazionale dell'asse e può desumerne la relativa posizione angolare. Gli encoder assoluti misurano direttamente la posizione angolare e ricavano la velocità.
Gli encoder incrementali sono più semplici da interfacciare e forniscono soluzioni di basso costo.

Encoder incrementale tachimetrico

Gli encoder ottici sono costituiti da una ruota dentata (fig.2.2), a lato della quale è posta una sorgente luminosa, e dall'altro un fotodiodo (fig.2.3). A seguito

Figura 2.2: Ruota dentata

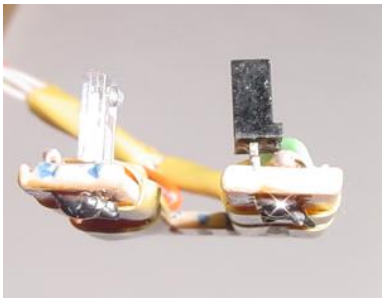

Figura 2.3: Coppia led - fotodiodo

di una rotazione, il segnale proveniente dal fotodiodo sarà un'onda quadra. Presenterà un numero di fronti di salita pari al numero di denti che interrompono il fascio luminoso. In caso di denti equispaziati il numero di fronti d'onda sarà proporzionale alla rotazione. Aumentando il numero di denti si aumenta la risoluzione. Questo tipo di encoder sono di costo relativamente basso e vengono

utilizzati nei sistemi di controllo della velocità. Hanno problemi di rumore per le basse velocità a causa degli errori di quantizzazione.

Encoder incrementale a quadratura di fase

Una configurazione composta da una coppia led-fotodiodo, non permette di conoscere il verso di rotazione. Tale informazione può essere ricavata usando due coppie led-fotodiodo.
In ogni istante i segnali all'uscita dei fotodiodi possono assumere una delle configurazioni descritte dalla tabella 2.1.2

Stato	Segnale 1	Segnale 2
S_0	1	0
S_1	1	1
S_2	0	1
S_3	0	0

Tabella 2.1: Stato dei segnali all'uscita dei fotodiodi

In caso di rotazione in un senso la sequenza di stati che si presenta in uscita è crescente, ($S_3 \rightarrow S_0 \rightarrow S_1 \rightarrow S_2 ...$). Nel caso la rotazione sia inversa la sequenza è decrescente ($S_3 \rightarrow S_2 \rightarrow S_1 \rightarrow S_0 ...$). Il numero di fronti d'onda sarà conteggiato da un opportuno contatore incrementato da una transizione $S_{(i)4} \rightarrow S_{(i+1)4}$ e decrementato da una transizione $S_{(i+1)4} \rightarrow S_{(i)4}$.
In questo genere di encoder la precisione è ancora influenzata dal numero di denti presenti sulla ruota. Per non saltare conteggi, occorre che la macchina a stadi, che analizza le due onde quadre, sia in grado di misurare transizioni che avvengono ad una distanza temporale di mezzo semiperiodo. Possiamo calcolare questa frequenza come:

$$F = 4VN \tag{2.1}$$

con F indichiamo la frequenza di refresh, V la velocità massima della ruota in giri al secondo, infine con N il numero di denti sulla ruota. Ad esempio avendo una ruota con 1000 denti, che gira a 10 giri al secondo è necessario avere una frequenza di aggiornamento di $40khz$.

Encoder ottici assoluti

Gli encoder incrementali non permettono di conoscere in modo assoluto la posizione angolare della ruota. Se si vuole disporre di questa informazione, anche

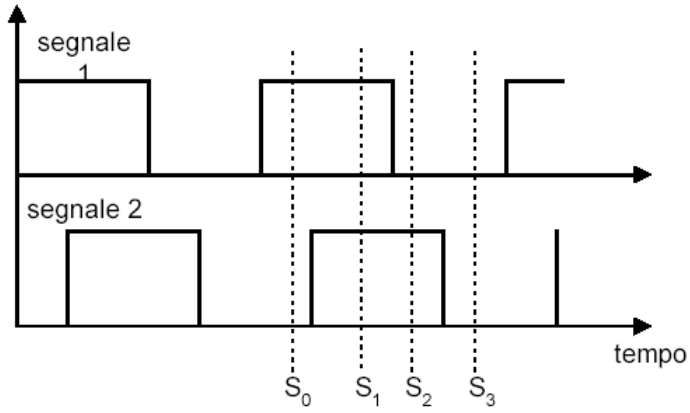

Figura 2.4: Segnali in uscita dei fotodiodi

a fronte dello spegnimento dell'impianto, è necessario utilizzare encoder ottici assoluti. Essi sono utilizzati essenzialmente in applicazioni con velocità di rotazione ridotte, là dove sia necessario conoscere la posizione angolare, come nel caso della misura di angoli di sterzo.

In questa tipologia di encoder (fig.2.5) la lettura è effettuata da una batteria di diodi - fotodiodi disposti in modo radiale, rilevando, per ogni posizione codificabile, una sequenza diversa: i raggi luminosi irradiano l'encoder e dove la luce è occlusa otteniamo per quel rilevatore un codice 0, dove la luce attraversa l'encoder otteniamo un codice 1. Un'opportuna rete combinatoria codificherà la configurazione in uscita dai fotodiodi nella posizione corrispondente, rappresentata da quell'angolo.

La minima frequenza ammissibile dalla rete combinatoria è pari a:

$$F = VN \tag{2.2}$$

con F la frequenza di refresh, V la velocità della ruota in giri/sec, N la risoluzione della ruota. Gli schemi più diffusi sono:

- codifica di Gray, fig. 2.6(a);
- codifica binaria, fig. 2.6(b);
- decimale codificato binario;

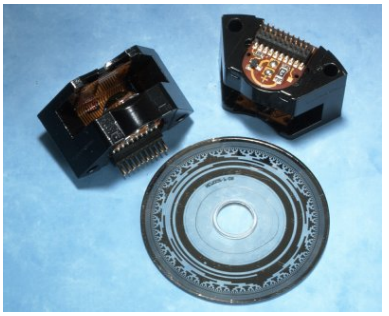

Figura 2.5: Encoder ottico assoluto

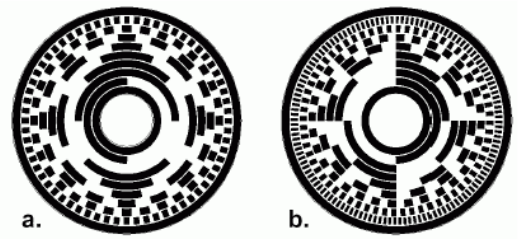

Figura 2.6: (a)Encoder con codifica di Gray (b) Encoder con codifica standard

La codifica di Gray è condizionata dal fatto che cambia un solo bit per volta, in questo modo vengono eliminate le ambiguità causate nel cambiamento di stato, dalle tolleranze dei componenti elettronici e dalle loro velocità di risposta.
Un possibile svantaggio degli encoder assoluti risiede nella complessità dell'interfaccia di codifica dell'output.

2.1.3 Modello cinematico

Per ottenere lo spostamento relativo a partire dalla posizione del robot è necessario conoscere la cinematica del sistema. Una delle più comuni nei robot mobili è quella a guida differenziale. Nel modello cinematico considerato siamo in presenza di due ruote motrici allineate tra loro ed una o più ruote indipendenti, a sostegno della base mobile fig.2.7 Possiamo calcolare lo spostamento lineare di

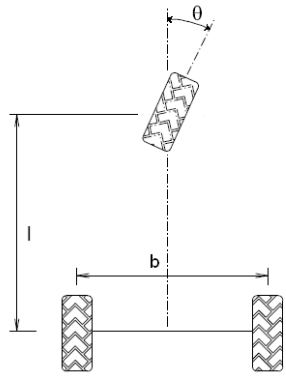

Figura 2.7: Modello cinematico

una ruota come:

$$\Delta U_i = c_m N_i \tag{2.3}$$

dove U_i è lo spostamento lineare della ruota nel periodo T_i, c_m fattore di conversione distanza percorsa / impulsi encoder, N_i numero di impulsi dell'encoder compiuti dall'ultima lettura.

Il fattore di conversione è ottenibile mediante la relazione 2.4

$$c_m = \pi D_n / n C_e \tag{2.4}$$

dove D_n è il diametro nominale della ruota, C_e rappresenta la risoluzione dell'encoder (numero dei denti della ruota), n è il rapporto di riduzione tra encoder e ruota.

Assumendo che il centro della base mobile si trovi nel punto medio dell'asse di collegamento tra le due ruote motrici, a seguito di uno spostamento lineare ΔU_{L_i} compiuto dalla ruota sinistra ed uno spostamento ΔU_{R_i} compiuto dalla destra, è possibile calcolare lo spostamento lineare del centro del robot come:

$$\Delta\rho_i = \frac{\Delta U_{L_i} + U_{R_i}}{2} \tag{2.5}$$

Lo spostamento angolare risulta:

$$\Delta\theta_i = \frac{\Delta U_{L_i} - \Delta U_{R_i}}{b} \tag{2.6}$$

dove b rappresenta la distanza tra i due punti di contatto di ruote e pavimento.

2.1.4 Sensori per l'angolo di orientazione

Questi sensori sono particolarmente importanti per il posizionamento dei robot mobili perchè possono aiutare a compensare la principale debolezza dell'odometria, ovvero l'errore di carattere incrementale, soprattutto nei confronti dell'orientazione. Per questo motivo è necessario correggere gli errori di posizionamento nel punto in cui si manifestano, per non consentire il loro propagamento.

A questo scopo entrano in aiuto gli strumenti che si descriverà di seguito.

Giroscopi

I giroscopi possono essere classificati in due categorie: giroscopi meccanici e giroscopi ottici.

Giroscopi meccanici si basano sulle proprietà innerziali di un rotore che gira rapidamente.
Il vettore momento della quantità di moto diretto nella direzione dell'asse di rotazione, deve rimanere invariato nel tempo nel sistema di riferimento inerziale adottato. La proprietà di orientazione del giroscopio è sfruttata in navigazione perchè fornisce una direzione nota e prescelta indipendente dai movimenti del veicolo (nave,aereo): al variare della posizione del veicolo muta la posizione del solido in rotazione rispetto al sostegno, ma

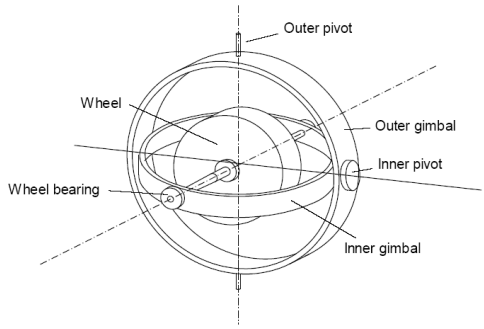

Figura 2.8: Giroscopio meccanico

la direzione dell'asse di rotazione resta invariata nel sistema inerziale di riferimento.

Giroscopi ottici non hanno parti in movimento, quindi sono meno sensibili alla forza di gravità e necessitano di meno manutenzione. Il dispositivo base consiste in due raggi laser che viaggiano in direzioni opposte lungo un percorso circolare chiuso. Le forme delle interferenze costruttive e distruttive, formate dalla divisione e dalla fusione dei due raggi, possono essere usati per determinare la velocità e la direzione di rotazione del dispositivo.

Sensori geomagnetici: le bussole

Il campo magnetico terrestre può essere rappresentato come un dipolo che varia sia nel tempo che nello spazio. Esso è decentrato di circa 440 km e inclinato di 11° rispetto all'asse di rotazione del pianeta. Questa differenza di posizione tra nord magnetico e geografico, detta inclinazione, varia nel tempo e in base alla posizione geografica.
I sensori geomagnetici, sensibili al campo magnetico terrestre, sono utilizzati per il calcolo della direzione di navigazione nei robot mobili.
L'utilizzo in ambiente chiuso di questa strumentazione è fortemente limitato dalle deformazioni del campo magnetico terrestre in presenza di linee elettriche e strutture metalliche.

2.2 Sensori di prossimità

I sensori di spostamento relativo forniscono un insieme di informazioni, insufficienti sotto certi aspetti, alla localizzazione. In aiuto a questo tipo di sensori si affiancano i sensori di prossimità che restituiscono un insieme di punti, in forma bidimensionale o tridimensionale, rappresentanti gli ostacoli individuati. Questo consente di integrare nella navigazione dati di natura topologica relativi all'ambiente di lavoro. (fig.2.9)

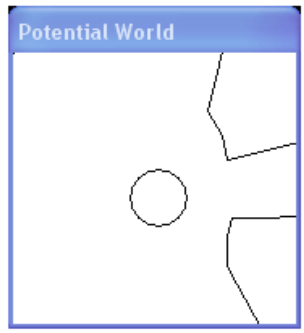

Figura 2.9: Informazioni topologiche tratte dai sensori di spostamento relativo

2.2.1 Sensori attivi

I sensori attivi sfruttano una sorgente di energia per irradiare l'ambiente. Quest'ultima, riflessa e captata, viene analizzata col fine di estrarre informazioni relative alla direzione, distanza e in alcuni casi forma dell'ostacolo.

È possibile classificare il trasduttore dei sensori attivi sulla base del tipo di energia emessa verso l'ambiente. Negli ambienti interni si utilizzano principalmente:

- sorgenti luminose;
- sorgenti acustiche.

Esistono anche sorgenti elettromagnetiche, impiegate nei radar.

Dalla direzionalità dell'energia impiegata e dalla velocità di propagazione nel mezzo dipendono risoluzione e frequenza di aggiornamento del sensore. In base a queste caratteristiche esistono due tecniche di analisi dell'eco:

- misurazione del tempo di eco;
- misurazione della differenza di fase tra segnale emesso e segnale riflesso.

Misurazione del tempo di eco (Time Of Flight)

La forma d'onda emessa dalla sorgente ha la forma di una serie di impulsi con periodo T. A seguito dell'iterazione con un oggetto parte dell'energia trasmessa viene riflessa. Misurando il tempo tra l'emissione dell'impulso e la ricezione del suo eco è possibile ottenere la distanza percorsa dall'impulso secondo la legge:

$$d = \frac{\nu t}{2} \tag{2.7}$$

dove d è la distanza percorsa, ν è la velocità di propagazione dell'energia nel mezzo. Il fattore peso $\frac{1}{2}$ è inserito perchè la distanza percorsa nel tempo t è doppia[1].

La distanza massima, per un sensore che utilizza questo tipo di tecnica, dipende dalla durata del periodo T del treno di impulsi:

$$d = \frac{\nu T}{2} \tag{2.8}$$

Oltre questa distanza è possibile confondere l'eco di due impulsi adiacenti. Quindi questa distanza costituisce il limite superiore della tecnica TOF. Le relazioni riportate per i sensori TOF sono valide se il segnale di ritorno è coassiale con

[1] Andata: dal sensore all'ostacolo. E ritorno: dall'ostacolo al sensore

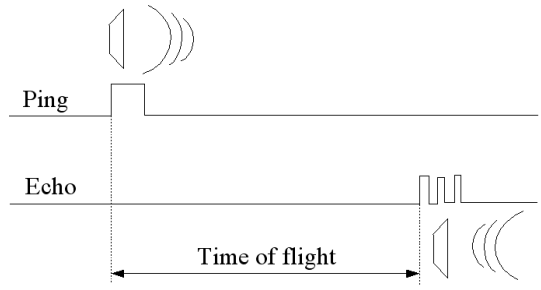

Figura 2.10: Onda diretta e onda riflessa in un sensore con tecnica TOF

quello trasmesso e se quindi il tragitto percorso è lo stesso. La distanza assoluta di un punto è disponibile direttamente senza la necessità di analisi complicate, la tecnica non è basata sulle proprietà planari o di orientazione della superficie osservata.

Possibili sorgenti di errore per i sistemi TOF sono:

- la variazione della velocità di propagazione. Riguardano in particolare i sistemi acustici che risentono delle variazioni di temperatura ed umidità.
- Le incertezze nella determinazione dell'esatto momento di arrivo dell'impulso riflesso. Sono causate dalle variazioni di intensità del segnale di ritorno a loro volta dovute alle diverse proprietà di riflessione dei materiali osservati.
- Le imprecisioni dei circuiti per il calcolo del tempo di volo dell'onda. Riguardano principalmente i sensori basati su raggi laser ed onde radio. Queste, più veloci delle onde acustiche, richiedono sistemi di controllo più sofisticati e costosi.
- Le iterazioni tra l'onda incidente e la superficie osservata. Il segnale captato dal ricevitore rappresenta solo una piccola parte dell'onda trasmessa. La restante energia viene riflessa in tutte le direzioni e può essere assorbita o passare attraverso l'oggetto osservato a seconda delle caratteristiche della superficie e dell'angolo d'incidenza del raggio. Se l'angolo con cui viene trasmessa l'onda supera un certo valore critico, l'energia riflessa non sarà più captata dal ricevitore.
- Le iterazioni tra più sensori dello stesso tipo. Negli ambienti con più robot mobili le onde emesse da un sensore possono essere riflesse da più oggetti e ricevute anche dagli altri sensori. Questo fenomeno è conosciuto come *crosstalk*.

I sensori TOF ad ultrasuoni sono ad oggi i più utilizzati nei robot mobili, per impieghi in ambienti chiusi o delimitati, a causa del loro basso costo e per il facile interfacciamento. I sensori TOF laser sono più costosi, e sono utilizzati in robotica in ambiti dove si richieda una precisione e un'accuratezza maggiore.

Misurazione della differenza di fase

La forma d'onda emessa è una sinusoide. Sfruttando il fatto che quando un'onda viene riflessa da una superficie la sua fase non risulta alterata, è possibile ricavare

la distanza da quest'ultima mediante la relazione:

$$d = \frac{\phi\lambda}{4\pi} = \frac{\phi\nu}{4\pi f} \tag{2.9}$$

dove λ è la lunghezza d'onda ν è la velocità di propagazione nel mezzo ed f la frequenza della sinusoide. La grandezza misurata direttamente è il seno della

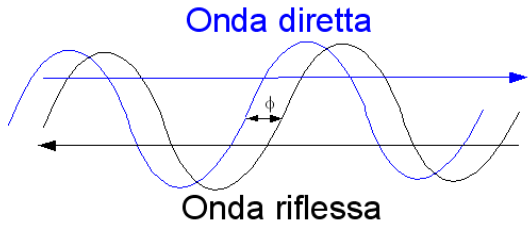

Figura 2.11: Sfasamento tra l'onda diretta e quella riflessa

fase, questa è calcolata con l'ambiguità di un periodo. L'intervallo d'incertezza che ne consegue è pari a

$$d_{\texttt{max}} = \frac{\nu}{2f} \tag{2.10}$$

La potenza della sorgente viene tarata in modo da non superare tale distanza. Con questa operazione è possibile lavorare in un intervallo di distanze non ambiguo. Rispetto ai sistemi TOF questi hanno il vantaggio di misurare anche la direzione e la velocità di un oggetto oltre che la sua distanza dalla sorgente.

2.2.2 Sonar

I sonar sono dei sensori di prossimità attivi che utilizzano come sorgente di energia un emittore di ultrasuoni. Ogni singolo dispositivo è in gradi di ottenere informazioni relative alla sola direzione secondo cui è orientato. Sono ampiamente usati in robotica per la loro semplicità e per il basso data rate che consente di ottenere rapidamente e a basso costo (computazionale ed economico) informazioni sensoriali che, per quanto rozze, possono risultare utili in molte applicazioni. I sonar soffrono di due problemi legati al tipo di sorgente utilizzata. L'impiego di ultrasuoni infatti non consente di avere una buona direzionalità del raggio di percezione (tipicamente di 30°) e a causa della scarsa direzionalità del suono nell'aria non è possibile aggiornare l'informazione con

frequenza. Per esempio volendo percepire un ostacolo alla distanza di 5m si ha invertendo l'equazione 2.8, un periodo minimo di aggiornamento di 1/30 s.

Il sistema radar più diffuso è il Polaroid ultrasonic ranging system basato sul tempo di volo. Il componente principale dell'apparecchio è il trasduttore ultrasonico. Funziona sia da altoparlante (sorgente) che microfono (rilevatore). Dato che il raggio a = 19 mm dell'apertura del trasmettitore è molto più grande della lunghezza d'onda acustica $\lambda = \frac{c}{f}$ (6.84mm) l'emissione forma un fascio in cui è concentrata l'energia. Modellizzando il trasmettitore come un pistone piatto di raggio a, chiuso in una membrana infinita, si ottiene un pattern di emissione che ha due regioni distinte: la zona vicina[2] e la zona lontana. Nella zona vicina, che si estende di fronte al trasduttore per una profondità di circa $\frac{a^2}{\lambda}$, il fascio è contenuto in un cilindro di raggio a. Nella zona lontana il fascio diverge con un angolo che dipende da a e λ. La figura 2.12 mostra la potenza

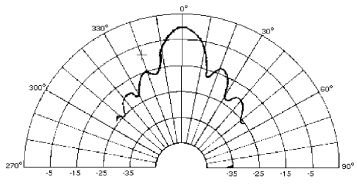

Figura 2.12: Diagramma di emissione radar

acustica in funzione dell'azimuth, nel caso monocromatico. Si nota un lobo centrale di circa $\theta_o = 20°$ più alcuni lobi secondari. Una prima approssimazione del sonar, che tuttavia produce buoni risultati consiste nel considerare il solo lobo principale di emissione.

Interpretazione dei dati sonar

Come evidenziato da Durrant - White, spesso in letteratura il sonar è considerato come una cattiva approssimazione di un sensore ideale (*ray-race scanner*) il quale ha la proprietà di emettere un fascio sottile e non divergente, passabile di riflessione diffusa. Con questo sensore è possibile misurare la distanza da qualunque bersaglio si trovi sull'asse del sensore indipendentemente dalla sua orientazione e dalla natura della superficie. In realtà un telemetro sonar possiede

[2]Zona di Fresnel

delle caratteristiche marcatamente diverse; in particolare differisce dal *ray-trace scanner* in due aspetti:

- apertura radiale del fascio;
- beam width e riflessione speculare.

Infatti come discusso nella precedente sezione il sonar emette un fascio con una apertura angolare non nulla (20°-30°) e la lunghezza d'onda (6mm) è tale per cui molti bersagli danno origine a riflessione speculare[3]. Questo da luogo ad una serie di problemi:

- Incertezza nella localizzazione del bersaglio. A causa dell'apertura del fascio l'eco ricevuta può provenire da un qualsiasi punto all'interno del cono di emissione;
- a causa della riflessione speculare è possibile ricevere l'eco solo da superfici ortogonali alla direzione di propagazione. Poichè il fascio emesso possiede una apertura angolare non nulla, si può ricevere l'eco da superfici illuminate con angoli di incidenza diversi da 0, diciamo compresi in $\pm\beta/2$. Questo è il semi-angolo di visibilità della superficie è in prima approssimazione si considera uguale a θ_o;
- un segnale spurio può arrivare al ricevitore a causa di riflessioni speculari multiple e come effetto viene sovrastimata la distanza del bersaglio dal sensore.

E possibile tuttavia interpretare correttamente i dati sonar mediante una corretta modellizzazione del sensore, come indicato da Kuc [18][19][20] e successivamente da Leonard e Durrant White [21], introducendo insiemi di punti TOF adiacenti aventi quasi la stessa profondità. Prendono il nome di *Regions of Constant Depth* e corrispondono ad un arco in coordinate cartesiane. Le RCD ottenute dal lobo principale sono le sole che danno informazione utile e possono essere distinte dalle altre (anche da quelle dovute a riflessioni multiple) grazie alla maggiore estensione angolare. Un algoritmo per l'estrazione e l'interpretazione delle RCD è presentato in [22].

[3] Un'onda incidente può essere riflessa in modo diffuso o speculare. La prima ha luogo quando la superficie incidente è ruvida rispetto la lunghezza d'onda del raggio. La seconda quando la superficie incidente è liscia rispetto la lunghezza d'onda.

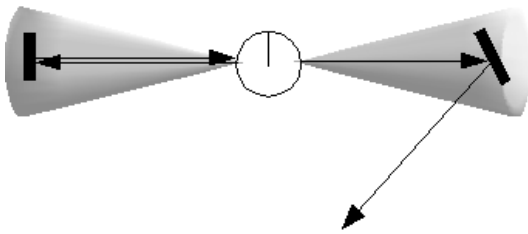

Figura 2.13: Riflessioni speculari del sistema radar. Vengono percepiti gli ostacoli con superfice incidente ortogonale.

2.2.3 Scanner laser

Uno scanner laser è un sensore che impiega come fonte di energia un diodo laser. Il raggio viene orientato verso la direzione da analizzare ed un sensore ad esso allineato ne misura la variazione di fase od il TOF. Poichè la velocità di propagazione della luce è molto più alta di quella del suono si possono avere cicli di aggiornamento molto più frequenti. L'impiego di una sorgente laser offre un'eccellente direzionalità.

Una singola coppia laser-diodo è montata su uno specchio rotante, in modo da effettuare un insieme di letture a ventaglio. Se lo specchio ha due gradi di libertà è possibile ottenere informazioni tridimensionali relativamente alla posizione di ambiente considerata.

2.2.4 Sensori passivi

I sensori passivi sono quelli che sfruttano l'energia presente nell'ambiente e tramite tecniche di elaborazione, dipendenti anche in questo caso dal tipo di energia analizzata, estraggono i bordi degli ostacoli. Questo tipo di sensori non soffre del problema dell'interferenza che affligge i sensori attivi, infatti non emettendo energia non possono influenzarsi reciprocamente.

Telecamera

La telecamera è forse il più semplice sensore di prossimità passivo. L'utilizzo di una immagine fornisce una grossa mole di informazione, che permette di affrontare il problema della localizzazione secondo diversi metodi:

- Navigazione con marcatori
- Localizzazione mediante informazioni topologiche

Altri tipi di sensori

Per quanto riguarda gli altri tipi di sensori si possono nominare, tra quelli a "basso livello":

sensori bumpers sensori di contatto;

sensori ad infrarossi sensori utilizzati in generale per scoprire la presenza di ostacoli nelle immediate vicinanze;

sensori termici e sniffer sensori di olfatto;

sensori di luminosità essenzialmente fotoresistori, fotodiodi e fototransistori.

Un'altra categoria di sensori è quella dei sensori di forza. Essi infatti hanno dimostrato di essere i più affidabili in termini di insensibilità al rumore e di facilità di interpretazione dei dati. Essi vengono in genere utilizzati per determinare quanto un robot entra in collisione con un altro oggetto e da la posizione relativa di quest'ultimo al robot.

Capitolo 3

Tecniche di localizzazione per sistemi robot mobili

Dato che nella nozione di localizzazione è insito il concetto di mappa, prima di descrivere il problema della localizzazione, è opportuno soffermarsi sulle possibili modalità di rappresentazione dell'ambiente.

3.1 Rappresentazione dell'ambiente

Esistono diverse tecniche per rappresentare la conoscenza che si ha del mondo circostante. La loro efficacia varia a seconda della tipologia dell'ambiente stesso. Alcune di queste tecniche si adattano meglio a piccoli ambienti di lavoro, altre invece ad ambienti interni, e così via. Una prima classificazione di mappe può essere quella che distingue le mappe metriche da quelle topologiche.

3.1.1 Mappe topologiche

Nelle mappe topologiche l'ambiente è rappresentato come un grafo connesso. I nodi corrispondono a punti di particolare importanza, gli archi corrispondono alle connessioni esistenti tra i nodi. Per esempio, in una rete viaria i punti di particolare importanza potrebbero essere le città, mentre gli archi le strade. All'interno di un edificio i punti potrebbero essere rappresentati dalle stanze, mentre gli archi corrisponderebbero ai corridoi, alle scale eccetera.

A causa della forte dipendenza dell'ambiente nel modello, è impossibile dare una definizione precisa di punto di particolare importanza. Le mappe topologiche vengono organizzate in modo gerarchico. Per esempio, a basso livello un nodo può corrispondere ad una stanza, ad alto livello invece ad un edificio. Un altro vantaggio di tale rappresentazione è quella che, data la loro natura astratta, le mappe possono essere costruite senza avere l'esatta conoscenza geografica dei nodi, ma solo sulla base della relazione di connessione tra loro esistente. Il problema della localizzazione su una mappa topologica consiste nello stabilire in quale nodo o su quale arco si trova il robot. Per tale scopo è necessario che il robot, attraverso il suo sistema sensoriale, riesca a distinguere un vertice da un altro. I caratteri distintivi sono infatti funzione dei sensori impiegati: mediante l'uso di una telecamera è possibile distinguere due oggetti identici geometricamente, ma di colore diverso, cosa impossibile o molto difficile con dei sensori ad ultrasuoni.

Le mappe topologiche non memorizzano di solito informazioni metriche, esistono alcune varianti come quella presentata in [23], in cui i nodi vengono etichettati con informazioni relative alla loro lunghezza.

3.1.2 Mappe metriche

Le mappe metriche rappresentano gli oggetti del mondo circostante in base alla loro dimensione e alla posizione. I principali metodi per la rappresentazione dell'ambiente tramite mappe metriche sono:

- descrizione geometrica;
- metodi basati su estrazione di feature;
- griglie di occupazione.

Metodi basati su descrizione geometrica

Tali metodi rappresentano l'ambiente di lavoro utilizzando le informazioni direttamente provenienti dai sensori. L'idea è quella di memorizzare tali dati, presi da posizioni diverse, ottenendo per ogni posizione un'acquisizione caratteristica. Invertendo il procedimento, è possibile ricavare la posizione del robot.

L'insuccesso del metodo si manifesta in ambienti simmetrici[1], in questo caso per più posizioni corrispondono acquisizioni praticamente uguali. Questo incon-

[1] Un esempio di ambiente simmetrico è senz'altro un corridoio

veniente non permette che la funzione sia invertibile. Per porvi rimedio vengono fornite soluzioni ad hoc in ogni situazione.

Metodi basati sull'estrazione di feature

L'idea è di costruire una mappa dell'ambiente sfruttando i caratteri distintivi dello stesso. Leonard, Durrant-Whyte e Cox sono convinti che il metodo feature based sia la strada da seguire. Una mappa precisa e concisa è usata per generare, in modo efficiente, la predizione dell'acquisizione sensoriale del robot in una certa posizione [29]. John Leonard e Durrant-White hanno sviluppato dei metodi di localizzazione basati sul riconoscimento di oggetti naturali, facilmente osservabili e distinguibili nelle misure sensoriali [29]. La localizzazione attraverso landmark artificiali è molto più semplice[2] e robusta, ma richiede la presenza nell'ambiente dei landmark di riferimento[3], situazione non sempre possibile. La tecnica basata su features naturali invece, richiede l'adozione di un criterio per la scelta dei caratteri distintivi. In [36] viene presentata una rete neurale per la selezione automatica delle features. Thrun dimostra che la rete neurale ha performance superiori rispetto ai sistemi nei quali le features sono scelte in modo manuale.

Metodi basati su griglie

Il metodo occupancy grid prevede di partizionare lo spazio di lavoro in una griglia, ogni cella è una rappresentazione di una piccola porzione dell'ambiente [37]. Questo metodo è stato presentato da Moravec e Elfes [38]. Ogni cella contiene informazioni riguardo lo stato di occupazione, da parte di un ostacolo, della corrispondente porzione del mondo reale. I valori memorizzati in questa matrice corrispondono alla probabilità della presenza dell'ostacolo. Attraverso metodi basati su tecniche bayesiane viene aggiornata la matrice durante il moto del robot.

In ambienti molto grandi lo spazio occupato dalla struttura dati contenente la matrice ed il costo computazionale necessario per aggiornarla sono elevati. Un altro difetto consiste nella difficoltà di definizione della risoluzione di discretizzazione, anche se sono stati proposti metodi adattativi capaci di aumentare la risoluzione in zone che richiedono una conoscenza accurata dell'ambiente, e

[2] in termini implementativi

[3] Per esempio dei codici a barre o dei palloni colorati in posizioni predefinite.

ridurla in zone meno importanti[4][39]. Il problema della definizione della risoluzione è molto importante perchè influisce direttamente sulla dimensione della struttura dati che modella l'ambiente.

3.1.3 Metodi di localizzazione

Per realizzare in modo efficace il proprio compito nell'ambiente di lavoro, il robot deve conoscere con precisione la propria posizione. Per esempio un robot staffetta che prende un oggetto in una determinata posizione e lo ricolloca in un'altra, deve essere in grado di conoscere la propria posizione, istante per istante, per poter impostare la navigazione. Se il robot avesse un'informazione errata della propria posizione nello spazio, finirebbe in un punto sbagliato, compiendo l'operazione assegnatagli in modo insoddisfacente.

Per localizzazione di un robot mobile si intende il processo di individuazione e aggiornamento della posizione e dell'orientazione del robot attraverso l'utilizzo dei dati sensoriali [40]. Il termine autolocalizzazione sottolinea, invece, che il robot è autonomo nell'esecuzione di tale processo di localizzazione. La precisione con cui un robot può conoscere tale posizione non dipende soltanto dallo specifico metodo di localizzazione che si utilizza, ma anche dai sensori che forniscono i dati in ingresso al sottosistema che si occupa della localizzazione. Per migliorare la precisione con cui il robot si localizza è possibile, infatti, utilizzare tecniche più complesse, ma anche scegliere sensori più sofisticati.

Il metodo di localizzazione più semplice, studiato maggiormente, è quello del *position tracking*. Conoscendo la posizione iniziale si ricostruisce la posizione attuale integrando gli spostamenti del robot. Il problema in questo approccio si riduce nel compensare l'errore odometrico che si accumula all'aumentare della strada percorsa dal veicolo.

Più complesso è il metodo della localizzazione globale, nel quale il robot non conosce la sua posizione iniziale, e deve determinarla autonomamente.

Ancora più difficilmente da gestire è il problema del *rapimento del robot*. In questo caso il robot inizialmente è ben localizzato. Ad un certo punto viene spento, trasportato in un altro posto e riacceso. Questo problema differisce dal precedente, in quanto il robot deve essere in grado di riconoscere che l'ambiente circostante è cambiato, rilocalizzarsi e procedere con i suoi compiti. Questo problema viene spesso usato come test degli algoritmi di localizzazione, per

[4]Per esempio: in zone con una densita di ostacoli maggiori e zone prive di ostacoli, zone con pericoli (macchine operatrici, vincoli architettonici, eccetera) e zone non a rischio (stanze vuote, corridoi, prati, eccetera).

verificare la capacità di recupero dell'algoritmo, nel caso fallisca la localizzazione del robot.

Questi problemi si complicano notevolmente in ambienti dinamici, dove gli oggetti presenti, attorno al robot mobile, possono spostarsi. Anche in questo caso, non esiste l'algoritmo universale di localizzazione, anche perchè, così come per la navigazione, i problemi possono presentarsi in forme diverse:

- l'ambiente in cui opera il robot si muove(in ambienti interni e/o esterni);
- il sistema di locomozione del robot muta (ruote, gambe o altro);
- influiscono la tipologia e l'accuratezza dei suoi sensori (laser scanner, telecamere, eccetera);
- varia il tipo di mappa utilizzata (ad esempio su una mappa topologica, si deve determinare su quale arco o nodo si trovi il robot)

In linea di massima possiamo distinguere tra metodi di localizzazione relativi e assoluti e tra implementazioni passive ed attive. I metodi relativi calcolano la posizione del robot rispetto al punto di partenza, mentre quelli assoluti non hanno bisogno di questa informazione e valutano direttamente la posizione in un sistema di riferimento assoluto [41]. Quando le tecniche di localizzazione sono implementate in modo passivo il robot utilizza l'informazione che proviene dai sensori senza controllare gli attuatori. Contrariamente i metodi di localizzazione attivi hanno il controllo dei motori e possono decidere dove dirigere i sensori, quali utilizzare e dove puntarli in modo da ottenere la miglior stima possibile della posizione [42].

Le varie tecniche sono catalogabili in metodi di localizzazione relativi o assoluti. Delle buone soluzioni sono state realizzate combinando metodi di categorie diverse fra loro. Infatti, i metodi di localizzazione relativi sono soggetti all'accumulo di errore nel tempo e quelli di posizionamento assoluti sono sensibili al rumore presente nelle informazioni acquisite dall'ambiente, la combinazione di due o più tecniche può migliorare le prestazioni del sistema di localizzazione. Tuttavia questa tecnica di associazione presenta il problema dell'integrazione dei dati sensoriali. Esistono due tipologie di tecniche, quelle numeriche come il filtro di Kalman [43] e i modelli stocastici di Markov [44] e quelle simboliche basate su regole.

I metodi relativi ed assoluti si possono ulteriormente dividere in sottogruppi [14]:

Misure di posizione relativa

- Odometria;
- Navigazione innerziale;

Misure di posizione assoluta

- Guide attive;
- Riconoscimento di landmark naturali o artificiali;
- Navigazione tramite mappa.

3.1.4 Metodi basati sull'odometria

L'odometria è, come si è detto, il metodo di misura più utilizzato per il posizionamento e la navigazione dei robot mobili, in quanto garantisce una buona precisione nelle misure a breve tempo, è poco costosa e consente alte frequenze di campionamento. Tuttavia poichè l'idea di base di tale metodo è l'integrazione nel tempo di informazioni incrementali sul movimento, esso si presta ad un inevitabile accumulo di errori. In particolare, gli errori sull'orientazione causano discrepanze di posizione che aumentano con la distanza percorsa dal robot. Nonostante queste limitazioni l'odometria è sicuramente un elemento importante del sistema di navigazione. I motivi per i quali è implementata su quasi tutti i robot mobili sono diversi [14]:

- I suoi dati possono essere integrati con quelli provenienti dalle tecniche di posizionamento assoluto, in maniera tale da fornire una stima migliore e più accurata della posizione;
- può essere usata per fornire una valutazione corretta della posizione per un tempo sufficiente a garantire, alle tecniche basate su landmark e matching di mappe, l'elaborazione dei loro dati sensoriali;
- in alcuni casi l'odometria è l'unica fonte di informazione disponibile per la navigazione, soprattutto quando il robot si muove in ambienti del tutto sconosciuti e senza landmark o quando accade che altri tipi di sensori non riescano a fornire dati utilizzabili;

Errori sistematici e non sistematici dell'odometria

L'odometria è basata su semplici equazioni facilmente implementate, che utilizzano le informazioni provenienti dagli encoder incrementali delle ruote. Alla base di tutto si fa l'assunzione che le rotazioni delle ruote possano essere tradotte in spostamenti lineari relativi alla superficie su cui il robot si muove, ma questa assunzione ha validità limitata, basti pensare allo slittamento di una ruota. Le diverse sorgenti di errore si possono classificare come segue [14]:

Errori sistematici

- Diverso diametro tra le due ruote o valori differenti dal valore nominale;
- la misura reale della distanza tra le ruote è diversa dal valore nominale;
- disallineamento delle ruote;
- risoluzione e periodo di campionamento finiti per gli encoder.

Errori non sistematici

- Tragitto su terreno accidentato;
- passaggio su oggetti imprevisti sul terreno;
- slittamento sulle ruote dovuto a :
  - terreni scivolosi;
  - grande accelerazione;
  - sterzate rapide;
  - forze esterne (interazioni con corpi esterni);
  - forze generate internamente (vedi le forze castor, tipicamente fonti di tali problemi);
  - contatto non perfetto con il terreno.

Questa distinzione è utile allo scopo di ridurre gli errori. Ad esempio, gli errori sistematici sono particolarmente gravi poichè vanno costantemente accumulandosi e, su superfici abbastanza regolari, pesano molto più di quelli sistematici. Tuttavia, su superfici irregolari, gli errori del secondo tipo risultano essere dominanti ed in più possono difficilmente essere previsti soprattutto in ambienti poco noti. Molti ricercatori studiano algoritmi che stimano l'incertezza della posizione di un robot. Grazie a questo approccio ogni posizione calcolata dal

robot è contornata da un'elisse di errore che indica una regione di incertezza per la posizione attuale del robot. Tipicamente questi elissi crescono con la distanza percorsa finchè un metodo di posizionamento assoluto riduce la crescita di incertezza e quindi azzera la dimensione dell'elisse d'errore. Queste tecniche di stima dell'errore devono fare affidamento sui parametri derivanti dall'analisi delle prestazioni del veicolo. Chiaramente questi parametri considerano solo gli errori sistematici perchè la grandezza non sistematica non può essere predetta.

Riduzione degli errori odometrici

La precisione odometrica dipende anche da alcune dimensioni critiche e da grandezze del progetto cinematico [14]:

- veicoli con una piccola distanza tra le ruote sono soggetti ad errori di orientazione più grandi rispetto ai veicoli che hanno distanze maggiori;
- ruote castor sottoposte ad una significativa porzione del peso totale sono possibili fonti di slittamento quando invertono la direzione;
- le ruote usate per l'odometria dovrebbero essere sottili e non compressibili;
- la ruota ideale dovrebbe essere fatta di alluminio con un sottile strato di gomma per migliorare la trazionè; nella pratica le ruote odometriche sono di solito ruote motrici che richiedono una larga superficie di contatto con il terreno;
- per migliorare le misure odometriche è consigliabile usare ruote sottili, non motrici, montate su un carrellino effettuando una accurata calibrazione prima di ricorrere a dispositivi o sensori aggiuntivi.

3.1.5 Navigazione inerziale

Il principio base della navigazione inerziale consiste nel calcolo, mediante accelerometri, dell'accelerazione del robot lungo ognuno dei tre assi direzionali e nella conseguente determinazione della velocità e della posizione [14]. Una piattaforma di sensori stabilizzati mediante un giroscopio, ottico o meccanico, viene utilizzata per mantenere fisse le orientazioni dei tre assi.

Sebbene concettualmente semplice, l'implementazione di tale metodo non è immediata a causa di errori che influiscono sulla stabilità del giroscopio e a causa della necessità di utilizzare dispositivi di alta qualità. Inoltre gli alti costi

per la manutenzione hanno precluso la diffusione di questa metodologia nelle applicazioni di robotica mobile.

Ciò che rende la navigazione inerziale particolarmente interessante è la sua assoluta indipendenza da sorgenti di informazioni esterne per il calcolo della posizione e la sua capacità di fornire misure dinamiche ad alta frequenza.

Il principale svantaggio è che la determinazione della velocità e della posizione richiedono calcoli di integrazione di primo e di secondo grado; conseguentemente ogni piccolo errore nel calcolo dell'accelerazione comporta la crescita, senza limiti, degli errori nelle misure integrate.

Così come il metodo basato sull'odometria, anche il metodo della navigazione inerziale, è caratterizzato dall'accumolo di errore. Quindi l'utilizzo per periodi di tempo troppo lunghi non è consigliabile dato che comporta grossa imprecisione nella definizione della posizione del robot.

3.1.6 Sistemi di navigazione basati su guide attive

I sistemi di navigazione basati su guide attive costituiscono il più comune ausilio alla navigazione sulle navi e sugli aerei. Consentono di ottenere una grossa affidabilità nel posizionamento tramite calcoli minimi, ma risentono di alti costi di installazione e manutenzione.

I sistemi basati sulle guide attive si possono classificare in sistemi a trilaterazione e a triangolazione.

Sistemi a trilaterazione

I sistemi a trilaterazione sono basati sulla determinazione della posizione di un veicolo tramite misure della distanza di questo da sorgenti di energia [14]. In genere vi sono almeno tre di queste sorgenti (*beacons*), situate in posizioni di cui si conoscono le coordinate, ed un ricevitore a bordo del veicolo. Viceversa, potrebbe esserci un trasmettitore a bordo del veicolo e tre ricevitori in posizioni note, ma ciò risulta problematico nel caso in cui nello stesso ambiente si trovino ad agire robot diversi. Tramite informazioni sul tempo di volo il sistema calcola la distanza tra i trasmettitori e il ricevitore a bordo. Un esempio di tali dispositivi è il sistema GPS, mentre un altro esempio dotato di affidabilità medio alte, ma di basso costo è basato su sensori ad ultrasuoni.

A causa delle distanze relativamente brevi coperte dagli ultrasuoni ,questi ultimi sistemi sono adatti ad operare in aree piccole e sgombre da consistenti ostacoli alla propagazione delle onde. Esistono due implementazioni generali [14]:

- un singolo trasmettitore posto sul robot con più ricevitori in posizione fissa;
- un singolo ricevitore sul robot e più trasmettitori fissi.

I sistemi del primo tipo sono probabilmente più adeguati ad applicazioni che coinvolgono solo uno o al più un piccolo numero di robot, mentre le configurazioni del secondo tipo non sono influenzate dal numero di ricevitori passivi in azione.

Sistemi a triangolazione

Nei sistemi a triangolazione la posizione e l'orientazione del veicolo sono determinati a partire dagli angoli formati con le guide attive [14]. Ci sono almeno tre trasmettitori, solitamente ad infrarosso, situati in posizioni conosciute dell'ambiente e un sensore rotante rileva gli angoli λ_1,λ_2 e λ_3 rispetto ai quali vede i tre trasmettitori. Da queste tre misure possiamo calcolare la posizione (x, y) e l'orientazione θ del robot. Semplici sistemi di questo tipo sono poco costosi,ma un problema di questa configurazione è che i *beacon* attivi devono essere sufficientemente potenti da trasmettere in tutte le direzioni. Si può risolvere questo problema facendo in modo che i trasmettitori irradino solo in determinate direzioni. Così, però, i *beacon* non sono visibili in alcune aree e ciò costituisce un problema perchè servono almeno tre angoli per la triangolazione.

I metodi a triangolazione possono essere ulteriormente distinti in base alle caratteristiche delle loro implementazioni [14]:

- trasmettitore-ricevitore rotante, riflettori fissi: in questa implementazione c'è una sorgente laser rotante a bordo del veicolo e almeno tre riflettori sono situati in posizioni conosciute dell'ambiente;
- trasmettitore rotante, ricevitori fissi: qui il trasmettitore, solitamente una sorgente laser rotante,è montato sul veicolo e tre o più ricevitori fissi registrano i raggi incidenti.

In generale, si può mostrare che la triangolazione è sensibile a piccoli errori angolari quando o gli angoli osservati sono piccoli o il punto di osservazione è sul cerchio che contiene le tre guide (o nelle vicinanze del cerchio).

3.1.7 Navigazione tramite landmark

La parola *landmark* indica quegli elementi distintivi dell'ambiente che un robot può riconoscere grazie ai suoi sensori. I *landmark* devono avere posizioni fisse

e ben note ed essere facilmente individuabili; una volta riconosciuti sono usati per la navigazione.

Per semplificare il problema della ricerca dei *landmark* spesso si assume che la posizione e l'orientazione del robot siano noti con una certa approssimazione, così che il robot debba cercare i *landmark* in un'area limitata [14]. Per tale motivo una buona precisione del sistema odometrico è un requisito fondamentale per l'individuazione dei *landmark*.

La procedura generale per realizzare la localizzazione tramite *landmark* è articolata come segue:

1. acquisizione di informazioni sensoriali;
2. rilevamento dei *landmark*;
3. determinazione di corrispondenze con i modelli in memoria;
4. calcolo della posizione.

Parleremo dei due tipi di *landmark: naturali e artificiali.* I *landmark naturali* sono quegli elementi o caratteristiche che già si trovano nell'ambiente, che non viene modificato, e che hanno una loro funzione oltre quella di consentire la navigazione del robot; i landmark artificiali, al contrario, sono elementi progettati specificatamente per la navigazione.

Landmark naturali

Il problema principale nella navigazione tramite *landmark naturali* è nel rilevamento dei *landmark* stessi. Le caratteristiche da riconoscere sono generalmente angoli, spigoli verticali di porte od incroci tra corridoi, luci sul soffitto, ecc. La scelta delle caratteristiche da riconoscere è importante perchè determina la complessità delle attività di descrizione, di individuazione e di matching e può risolvere situazioni di ambiguità migliorando la precisione del metodo. Generalmente un sistema di posizionamento basato sul *landmark naturali* ha i seguenti componenti basilari [14]:

- i sensori per individuare i *landmark* separandoli dal resto della scena;
- un metodo per confrontare il riferimento osservato con una mappa dei *landmark* noti;
- un metodo per calcolare gli errori di localizzazione dopo il matching.

caratteristiche della navigazione basata su *landmark*

Possiamo riassumere le caratteristiche della navigazione basata su *landmark* come segue [14]:

- i metodi basati su landmark artificiali sono ben sviluppati ed affidabili;
- i *landmark naturali* offrono, per conto, molta flessibilità e non richiedono la modifica dell'ambiente (cosa difficilmente ottenibile in alcune applicazioni);
- la distanza massima tra robot e *landmark* è più piccola di quella nei sistemi con guide attive;
- la precisione di posizionamento dipende dalla distanza e dall'angolo tra robot e *landmark*;
- richiedono maggior tempo di calcolo rispetto ai sistemi basati sulle guide attive;
- le condizioni ambientali, come l'illuminazione possono essere problematiche e in situazioni al limite di visibilità alcuni *markers* potrebbero non venire riconosciuti o, peggio ancora, alcuni oggetti simili ai markers potrebbero essere confusi con essi determinando errori significativi nella valutazione della posizione;
- i *landmark* devono essere disponibili nell'ambiente di lavoro intorno al robot;
- tali sistemi richiedono di partire in posizione nota così che il robot sappia dove cercare i *landmark*; se ciò non è possibile il robot deve effettuare una manovra di ricerca in tutto l'ambiente;
- bisogna mantenere memoria dei *landmark* e della loro posizione.

3.1.8 Posizionamento basato su mappe

Il posizionamento basato su mappe,conosciuto come *map matching* è una tecnica in cui il robot usa i dati provenienti dai suoi sensori per creare una mappa dell'ambiente. Questa mappa viene poi confrontata con un modello in memoria e se viene trovata corrispondenza tra le due, il robot è in grado di calcolare la propria posizione ed orientazione. La mappa memorizzata che funge da mappa

di riferimento nel confronto, potrebbe essere stata costruita sia dal programmatore che attraverso un'attività di esplorazione preventiva dell'ambiente. La procedura base di questo metodo è articolata come segue:

- acquisizione di informazioni sensoriali;
- costruzione della mappa locale;
- determinazione di corrispondenze tra la mappa locale ed il modello - calcolo della posizione.

I principali vantaggi di questa tecnica di localizzazione sono [14]:

- l'utilizzo della naturale struttura ripetitiva tipica degli ambienti chiusi senza necessità di modificarli;
- la possibilità di generare e aggiornare la mappa dell'ambiente che può essere utile anche agli altri robot per compiti come la generazione di traiettorie o l'identificazione di punti in cui il robot potrebbe fermarsi, *local minima traps*;
- la possibilità di conoscere nuovi ambienti e di migliorarne il dettaglio mediante esplorazione.

Di contro,il metodo in questione richiede [14]:

- una notevole stazionarietà dell'ambiente per poter distinguere le caratteristiche utili al confronto;
- un'alta precisione del dettaglio nella costruzione della mappa;
- la disponibilità di un'elevata potenza di calcolo.

Costruzione della mappa locale e confronto

Un primo aspetto da considerare nel confronto tra la mappa corrente locale e quella globale presente in memoria è che le letture dei sensori ed il modello dell'ambiente possono essere in differenti formati [14]. Una soluzione tipica a questo problema consiste nell'utilizzare la posizione approssimata basata sull'odometria con la quale il robot, attraverso i suoi sensori, dovrebbe percepire. Successivamente si confronta questa scena con quanto il robot ha effettivamente acquisito e attraverso il calcolo delle differenze si identifica la reale posizione del robot. L'odometria è fondamentale per definire un sistema di posizionamento

basato su mappa, su di essa si basano sia l'individuazione delle caratteristiche dell'ambiente (mura ed angoli) che il calcolo della posizione.

Il problema del *map matching* è stato ampiamente studiato: ricordiamo le strategie per il confronto tra griglie di occupazione [25], il metodo dell'istogramma dell'angolo [26], l'approccio probabilistico [41], le tecniche per il confronto di scansioni [27] e i confronti sperimentali tra i vari metodi.

Schiele e Crowley [25] hanno studiato differenti tecniche per confrontare due griglie di occupazione: la prima è quella locale, centrata sul robot, la quale modella l'ambiente circostante utilizzando le più recenti letture dei sonar; la seconda è il modello di riferimento. Schiele e Crowley propongono di utilizzare, oltre alla griglia di occupazione, anche un'altra rappresentazione dell'ambiente, quella tramite primitive parametriche che descrivono i limiti dello spazio libero in termini di segmenti o superfici definite da una lista di parametri. Comunque il rumore nei dati del sonar può rendere il processo di raggruppamento delle letture del sensore, utile per formare primitive geometriche, poco affidabile. In particolare piccoli oggetti, come ad esempio gambe di tavoli, sono praticamente impossibili da distinguere dal rumore. Schiele e Crowley propongono di confrontare:

- segmenti e segmenti;
- segmenti e griglia;
- griglia e segmenti;
- griglia e griglia

Schiele e Crowley evidenziano l'importanza di tenere conto, nel processo di calcolo della nuova posizione, dell'incertezza della posizione della griglia locale. La correzione della posizione stimata del robot è molto importante, particolarmente durante l'esplorazione di ambienti sconosciuti.

Hinkel e Knieriemen [26] hanno sviluppato un metodo di modellazione dell'ambiente detto *ad istogramma dell'angolo*. Esso consiste, innanzitutto, in una scansione a 360° della stanza e poi nella memorizzazione, nella mappa locale, dei punti che rilevano la presenza di ostacoli.Dopo l'algoritmo misura l'angolo relativo d tra due coppie di punti adiacenti e costruisce l'istogramma degli angoli. La direzione uniforme dei muri principali è chiaramente visibile sotto forma di picchi nel diagramma. Quest'algoritmo è sicuramente robusto nei confronti di aperture nei muri, come porte e finestre.

Successivamente tutti gli angoli vengono normalizzati e quindi si costruiscono due ulteriori istogrammi, uno per la direzione x e l'altro per la y. A questo punto i picchi mostrano la distanza dei muri in x e y.

Renekenè classifica come ipotetica, incerta o confermata l'informazione, tipicamente non precisa, proveniente dai sonar. Quando un dato è confermato viene utilizzato per costruire la mappa. Prima di aggiornare la mappa però ogni nuovo dato deve essere associato con un piano, un angolo o un lato. Reneken costruisce un albero di ipotesi che costituisce una struttura per i dati e permette di seguire differenti ipotesi fino ad accumulare informazioni sufficienti per arrivare alla decisione finale.

Un elemento importante nella tecnica di Renekenè la visibilità degli oggetti stabilita tramite modelli interni. Il metodo può essere così riassunto:

1. predire la posizione del robot utilizzando l'odometria;
2. tra l'insieme di oggetti dati, testare quale oggetto è visibile e a quale sensore e predire le misure;
3. comparare le misure predette con quelle attuali;
4. usare l'errore rilevato per calcolare la posizione del robot.

3.1.9 Il filtro di Kalman

Il filtro di Kalman discusso in [55] [56] [57] [58] è un ottimo osservatore iterativo dello stato per sistemi lineari e linearizzabili. Ben si adatta alla soluzione del problema dell'integrazione, nel caso del tracciamento della posizione, in quanto opera su distribuzioni gaussiane (quindi aventi un solo massimo). Di solito il KF viene usato nell'ambito della localizzazione per fondere due flussi di dati: le rilevazioni di posizione e le rilevazioni di spostamento relativo. Si tiene traccia degli errori di queste rilevazioni, assunti gaussiani ed a media nulla, mediante opportune matrici di covarianza. Viene inoltre tenuto conto dell'errore di valutazione (anch'esso gaussiano ed a media nulla) mediante una matrice interna al sistema, che evolve in base agli ingressi. Quest'ultimi contengono l'informazione sullo spostamento e attraverso le leggi di evoluzione del sistema, basate sulla cinematica del robot, si calcola la configurazione del robot. Quest'ultima viene corretta sfruttando i riferimenti esterni che permettono il controllo del sistema. In questo caso è possibile distinguere i passi di proiezione e correzione che coincidono rispettivamente con le fasi di *time - update e measurement - update*. Durante il *time - update* viene generata la posizione stimata in base al passato e

allo spostamento, mentre durante *measurement - update* viene corretta in base alla rilevazione. Gutmann, Weigel, Nebel e Jensfel [43] [48] illustrano l'uso del filtro di Kalman nel processo di localizzazione.

Kalman riesce a fornire discretizzazioni di qualsiasi funzione di densità di probabilità. Lo svantaggio di queste implementazioni risiede nel loro costo computazionale. Ciò le rende spesso inapplicabili in situazioni in cui il robot abbia risorse computazionali ridotte.

Localizzazione attiva e passiva

La maggior parte delle implementazioni delle tecniche di localizzazione sono passive, cioè utilizzano l'informazione che proviene dai sensori del robot senza controllarne gli attuatori. Se il robot perde la sua posizione può accadere che i metodi di localizzazione passivi non riescano a ricalcolarla perchè non possono dirigere il robot nei punti dove ciò è possibile. Invece i metodi di localizzazione attivi hanno il controllo sui motori e sui sensori del robot e possono decidere dove dirigerlo, possono scegliere quali sensori usare e dove puntarli in modo da ottenere la miglior stima possibile della posizione [42]. In questo caso si parla anche di localizzazione basata sul contesto.

È importante considerare la localizzazione attiva come uno dei compiti che il robot è in grado di compiere. Tale compito da un lato ha l'effetto desiderato di rilocalizzare il robot e dall'altro quello indesiderato di cambiarne la configurazione. Così il sistema di controllo deve gestire le possibili interferenze tra le attività previste e l'azione di localizzazione quando decide di eseguirla. Per una buona implementazione di un sistema di localizzazione attiva abbiamo bisogno dei seguenti elementi [28]:

- Informazione sull'affidabilità della propria posizione. Per iniziare l'azione di rilocalizzazione il sistema di controllo deve disporre di informazioni sull'affidabilità della propria posizione. In questo modo quando il robot ritiene che la sua posizione non sia corretta interrompe il suo compito per avviare il processo di localizzazione.

- Capacità di interrompere i compiti e minimizzare i rischi. Il sistema di controllo memorizza lo stato dei compiti al momento dell'interruzione in modo da poterli riprendere dopo l'interruzione e inizia l'attività di localizzazione cercando di minimizzare i rischi di urti con gli ostacoli.

- Prosecuzione dei compiti. Dopo che il robot ha ricalcolato la sua posizione è in grado di riprendere le attività interrotte. La procedura di localizzazione generalmente porta il robot in un'altra posizione e la sua esecuzione

impiega un tempo relativamente alto, quindi il robot potrebbe dover ripianificare nuove attività. È fondamentale che nel sistema di controllo di alto livello si fornisca, per ogni attività, una descrizione delle azioni da eseguire al termine di un'interruzione. Per esempio nel caso della attività di navigazione bisogna ricalcolare il nuovo punto verso cui dirigersi.

3.2 SLAM

Un altro problema oggetto di numerosi studi da parte della comunità scientifica è quello della localizzazione e della simultanea costruzione della mappa, meglio noto come problema dello SLAM (Simultoneous Localization And Mapping). La possibilità di posizionare un veicolo autonomo in una posizione sconosciuta di un ambiente sconosciuto con l'abilità di autocostruirsi una mappa e simultaneamente di utilizzarla per la navigazione, renderebbe un tale robot realmente autonomo (autonomia intesa sempre nel contesto della navigazione e della localizzazione). Il più grande vantaggio dello SLAM è quello di poter eliminare la necessità dell'introduzione di infrastrutture artificiali nell'area di lavoro, o una conoscenza a priori della topologia del1'ambiente. Una soluzione al problema dello SLAM avrebbe quindi un valore inestimabile in una serie di applicazioni nelle quali mappe precise sono inottenibili [30][31] in questo contesto, essendo ignote sia la posizione del veicolo che la mappa, la stima della posizione del robot e il modello dell'ambiente sono fortemente correlati e non possono essere ottenuti uno indipendentemente dall'altro. Negli ultimi sei anni sono stati compiuti progressi significativi nella ricerca di una soluzione allo SLAM. In [32][33][34] viene presentata una soluzione probabilistica basata sull'uso del campionamento (in particolare viene utilizzato il metodo Monte Carlo) per 1'approssimazione di opportune funzioni di densità di probabilità. Questa soluzione è in grado di gestire distribuzioni multimodali, fornendo quindi una soluzione anche al problema del rapimento. A causa degli alti costi computazionali però, questo metodo non è applicabile in scenari in cui è richiesta la soluzione allo SLAM in real-time. Per questo genere di applicazioni, ipotizzando che 1'incertezza possa essere modellata come una gaussiana, il metodo più efficiente per risolvere lo SLAM è suggerito dall'uso del filtro di Kalman [55][56], che consente di costruire una soluzione ricorsiva al problema [45] [46] [47] [48] [49] [50] [51].

Capitolo 4

Filtri di Kalman

Nel 1960 R. E. Kalman presentò un nuovo approccio per i filtri digitali applicati ai problemi di predizione [2].
Un'importante classe di problemi teorici e pratici in telecomunicazioni ed in controlli automatici è di natura statistica:

- predizione di segnali random;
- separazione di segnali random in presenza di rumore;
- riconoscimento di segnali di forma nota (impulsi, sinusoidi) in presenta di rumore;

i filtri di Kalman gestiscono in modo efficiente queste problematiche.

Nel campo della robotica l'errore, con cui si ha a che fare maggiormente, concerne la rilevazione sensoriale del mondo che circonda il veicolo.

4.1 Introduzione ai filtri di Kalman

Essenzialmente il filtro di Kalman è un algoritmo di data processing ricorsivo, in grado fornire la stima di una grandezza, sulla base di tutte le informazioni che possono concorrere alla sua determinazione.

Il vettore di stato è composto da variabili misurabili direttamente o indirettamente e non misurabili. L'algoritmo, partendo da condizioni iniziali note e integrando tutte le informazioni in suo possesso, è in grado di stimare l'evoluzione corretta di tutte le variabili, basandosi sulle correlazione esistenti tra

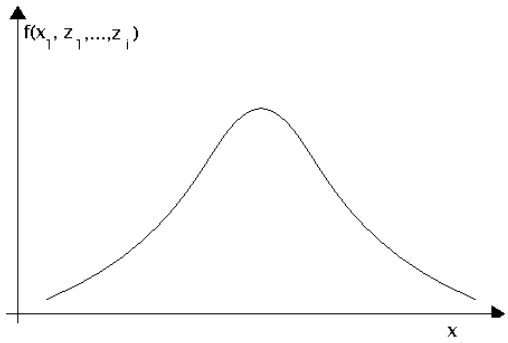

Figura 4.1: Esempio di densità di probabilità

quest'ultime. Il carattere ricorsivo permette che non vengano memorizzate tutte le informazioni raccolte dal sistema in quel momento e che non vengano rielaborate ad ogni passo.

Il filtro di Kalman fornisce la soluzione ottima,[1] utilizzando i dati provenienti da un sistema di misura affetto da errore. Dal punto di vista statistico si pone il fine di propagare la densità di probalità condizionata, dei dati provenienti dagli strumenti di misura, della variabile casuale desiderata.

Si consideri la figura 4.1. Descrive il profilo della densità di probabilità della variabile aleatoria x. L'andamento è condizionato dalla conoscenza delle misurazioni fatte sul sistema sino all'istante i. La sua forma dipende dal grado di incertezza che possiede la variabile aleatoria x. Tanto più è stretta e alta, tanto più il valore di x è concentrato in quella regione. Una volta determinata la densità di probabilità della variabile aleatoria si possono definire i seguenti stimatori:

la media: il valor medio dei possibili valori;

la moda: il valore di x con probabilità maggiore, ovvero quello in corrispondenza del picco più alto;

la mediana: il valore di x che lascia alla sua destra e alla sua sinistra il 50% della probabilità totale.

[1] Per ottima si intende la stima ottenuta minimizzando gli errori. Oltretutto Kalman è la soluzione del problema di Wiener.

Considerando un sistema descritto da un modello lineare, se il rumore del sistema e delle misurazioni sono descritti da variabili aleatorie non correlate e di carattere gaussiano, allora scegliere come criterio di ottimalità per gli stimatori la media, la moda o la mediana è equivalente.

4.2 Un semplice esempio

Si vuole descrivere un esempio banale dell'applicazione del filtro di Kalman col fine di chiarire eventuali perplessità. Si consideri il problema della localizzazione. Al tempo t_1 il robot acquisisce la misura z_1 della distanza fra se stesso ed un riferimento fisso nell'ambiente (per esempio un muro). La misura acquisita è affetta da una imprecisione con deviazione standard pari a σ_{z_1}. Sulla base di questa rilevazione è possibile creare una funzione di probabilità condizionata rispetto la misura, che stima la posizione del robot all'istante t_1. Questa funzione restituisce la probabilità che il robot si trovi in posizione x, sulla base delle rilevazioni effettuate. La stima migliore è senz'altro:

$$\widehat{x}(t_1) = z_1 \tag{4.1}$$

con una varianza dell'errore nella stima data da:

$$\sigma_x^2(t_1) = \sigma_{z_1}^2 \tag{4.2}$$

Il valore di $\widehat{x}$ corrisponde sia alla media, che alla moda, che alla mediana.
Se al tempo $t_2 = t_1$ viene effettuata un'altra osservazione, da un altro sistema di misura, con varianza $\sigma_{z_2}^2$, dando un valore z_2, la stima migliore della variabile x è:

$$\hat{x}(t_2) = \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_2 \tag{4.3}$$

con varianza σ^2:

$$\sigma^2 = \frac{\sigma_{z_1}^2 \sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \tag{4.4}$$

La densità di probabilità condizionata da z_1 e z_2 della variabile aleatoria che descrive la posizione del robot risulta essere gaussiana con media $\hat{x}(t_2)$ e varianza σ^2.
L'espressione 4.3 può essere riscritta come:

$$\hat{x}(t_2) = z_1 + \left[\frac{\sigma_{z_1^2}}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \right] [z_2 - z_1] \tag{4.5}$$

quindi:

$$\hat{x}(t_2) = \hat{x}(t_1) + G_2[z_2 - z_1] \tag{4.6}$$

$$G_2 = \frac{\sigma_{z_1^2}}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \tag{4.7}$$

dove G_2 è definito come il guadagno di Kalman.
La relazione riportata corrisponde al filtro di Kalman, si noti la caratteristica ricorsiva, che può essere applicata anche alla varianza:

$$\sigma_x^2(t_2) = \sigma_x^2(t_1) - G_2\sigma_x^2(t_1) \tag{4.8}$$

4.3 Il filtro di Kalman-Bucy

Si consideri il sistema descritto dalla seguente equazione di stato [8]:

$$\begin{cases} \dot{X}(t) = & AX(t) + v_1(t) \\ Z(t) = & CX(t) + v_2(t) \end{cases} \tag{4.9}$$

dove A è la matrice di transizione di stato all'istante t, X è il vettore di stato, B è la matrice di ingresso, C è la matrice di misurazione, $0 \leq t \leq T$ è la variabile temporale, il vettore v_1 rappresenta il rumore introdotto nella transizione di stato, v_2 rappresenta il rumore introdotto nell'operazione di misurazione. Sono definiti come processi di rumore bianco, con le seguenti caratteristiche:
media nulla:

$$E[v_1(t)] = 0; \quad E[v_2(t)] = 0; \quad \forall t \tag{4.10}$$

e varianza:

$$E[v_1(t)v_1^T(t+\tau)] = \Psi_{v_1}(t)\delta(\tau) \tag{4.11}$$

$$E[v_2(t)v_2^T(t+\tau)] = \Psi_{v_2}(t)\delta(\tau) \tag{4.12}$$

non correlati:

$$E[v_1(t)v_2^T(t+\tau)] = 0 \tag{4.13}$$

non correlati anche con lo stato iniziale:

$$E[X(0)v_1^T(t)] = 0; \quad E[X(0)v_2^T(t)] = 0; \quad \forall t \tag{4.14}$$

Lo stato iniziale ha media nota e matrice di covarianza nota:

$$E[X(0)] = X_0; \quad E\{[X(0) - X_0][X(0) - X_0]^T\} = \Psi_0 \tag{4.15}$$

Lo scopo dell'algoritmo è di calcolare lo stato, sfruttando la conoscenza di z nell'intervallo $0 - T$, al fine di minimizzare la funzione indice:

$$J[\hat{X}(t|T)] = E\{[X(t) - \hat{X}(t|T)]^T Q(t)[X(t) - \hat{X}(t|T)]\} \tag{4.16}$$

dove $\hat{X}(t|T)$ indica lo stato stimato.

Si cerca una stima che dipenda direttamente da Z. Il legame lineare viene supposto del tipo:

$$\hat{\dot{X}}(t|t) = F(t)\hat{X}(t|t) + G(t)Z(t) \tag{4.17}$$

L'equazione riportata rappresenta la predizione dello stato. Le condizioni iniziali $\hat{X}(0|0)$ si suppongono note ed uguali a X_0.
Applicando la seguente notazione:

$$R(t) = D(t)\Psi_v(t)D^T(t) \tag{4.18}$$

$$S(t) = B(t)\Psi u(t)B^T(t) \tag{4.19}$$

$$P(t|t) = E[\hat{e}(t|t)\hat{e}^T(t|t)] = \Psi_e(t|t) \tag{4.20}$$

dove l'errore, una volta acquisita l'osservazione, è calcolabile come:

$$\hat{e}(t|t) = X(t) - \hat{X}(t|T)$$

La stima ottima si ottiene scegliendo:

$$F(t) = A(t) - G(t)C(t) \tag{4.21}$$

$$G(t) = P(t|t)C^T(t)R^{-1}(t) \tag{4.22}$$

dove la $P(t|t)$ è la soluzione dell'equazione differenziale di Riccati:

$$\dot{P} = A(t)P + PA^T(t) - PC^T(t)R^{-1}P + B(t)\Psi_u(t)B^T(t) \tag{4.23}$$

con $P(0|0) = 0$.

4.4 Il filtro di Kalman discreto

Si consideri il seguente sistema di stato:

$$\begin{cases} X(k+1) = & A(k+1,k)X(k) + v_1(k) \\ Z(k) = & C(k)X(k) + v_2(k) \end{cases} \tag{4.24}$$

dove A è la matrice di transizione di stato dal punto k al punto $k+1$, X è il vettore di stato, C è la matrice di misurazione, il vettore v_1 rappresenta il rumore introdotto nella transizione di stato, v_2 rappresenta il rumore introdotto nell'operazione di misurazione.
Sia v_1 che v_2 sono processi di rumore bianco definiti come:

$$E[v_1(n)v_1^T(k)] = \begin{cases} Q_1(n) & n = k \\ 0 & n \neq k \end{cases} \tag{4.25}$$

$$E[v_2(n)v_2^T(k)] = \begin{cases} Q_2(n) & n = k \\ 0 & n \neq k \end{cases} \tag{4.26}$$

i due processi non sono correlati con lo stato iniziale $X(0)$ e sono incorrelati fra loro per ogni $k \geq 0$:

$$\forall n, k \quad E[v_1(n)v_2^T(k)] = 0 \tag{4.27}$$

Considerando tutte le variabili osservabili in ingresso, date dal vettore $Z(1)$, $Z(2)$, ... , $Z(k)$, il filtro di Kalman, formalmente, stima le componenti dello stato $X(i)$ col fine di minimizzare l'errore quadratico medio. Questo procedura è chiamata:

problema di filtraggio se $i = k$,

problema di predizione se $i > k$,

problema del perfezionamento se $1 \leq i < k$.

Essenzialmente si tratterà il problema della predizione.

4.4.1 Il processo di innovazione

Date le misurazioni $Z(k|Z_{k-1})$ costituenti la stima ai minimi quadrati del vettore $Z(k)$ al tempo k basata sulle osservazioni passate partendo dal passo $k = 1$ al passo $k - 1$, si definisce innovazione:

$$\alpha(n) = Z(k) - Z(k|Z_{k-1}), \quad k = 1, 2, 3... \tag{4.28}$$

che rappresenta l'informazione raccolta nelle osservazioni $Z(k)$.

L'innovazione eq. 4.28 gode delle seguenti proprietà:

- l'innovazione $\alpha(k)$ associata al vettore $Z(k)$ al tempo k è ortogonale alle osservazioni passate $Z(1), Z(2), ..., Z(k-1)$:

$$E[\alpha(k)Z^T(n)] = 0, \quad 1 \leq n \leq k-1 \tag{4.29}$$

- Il processo di innovazione consiste in una sequenza di vettori di variabili aleatorie ortogonali fra loro:

$$E[\alpha(k)\alpha^T(n)] = 0, \quad 1 \leq n \leq k-1 \tag{4.30}$$

- Esiste una corrispondenza uno-uno tra la sequenza di vettori di variabili aleatorie $\{Z(1), Z(2), ..., Z(k-1)\}$ rappresentante i dati osservati e la sequenza di vettori di variabili aleatorie $\{\alpha(1), \alpha(2), \alpha(3), ..., \alpha(k)\}$ rappresentante l'innovazione ad ogni passo. Alla luce di questa corrispondenza ogni sequenza può essere ricavata dall'altra attraverso un'operazione lineare, senza perdita di dati:

$$\{Z(1), Z(2), ..., Z(k-1)\} \Leftrightarrow \{\alpha(1), \alpha(2), \alpha(3), ..., \alpha(k)\} \tag{4.31}$$

4.4.2 Matrice di correlazione del processo di innovazione

Per determinare la matrice di correlazione del processo di innovazione $\alpha(k)$ consideriamo la forma ricorsiva della equazione 4.24:

$$X(k) = A(k,0)X(0) + \sum_{i=1}^{k-1} A(k, i-1)v_1(i) \tag{4.32}$$

alla quale si devono aggiungere le seguenti condizioni:

- Il valore iniziale del vettore di stato X(0) è noto;
- Le osservazioni per $k \leq 0$ sono nulle, così come il vettore di rumore;
- La matrice di transizione di stato ha la seguente proprietà:

$$A(k, k-1)A(k-1, k-2)...A(i+1, i) = A(k, i)$$

e

$$A(k, k) = I$$

dove I è la matrice identica.
Si tenga presente che per un sistema tempo invariante vale:

$$A(k+1, k) = A(k+1-k) = A(1) = \text{costante}$$

Date le osservazioni acquisite nei passi precedenti $\{Z(1), Z(2), ..., Z(k-1)\}$ la stima ai minimi quadrati che minimizza l'errore è:

$$\hat{Z}(k|Z_{k-1}) = C(k)\hat{X}(k|Z_{k-1}) \tag{4.33}$$

quindi il calcolo dell'innovazione è dato dalla relazione:

$$\alpha(k) = Z(k) - C(k)\hat{X}(k|Z_{k-1}) \tag{4.34}$$

sostituendo $Z(k)$ coll'equazione di misurazione si ottiene:

$$\alpha(k) = C(k)\epsilon(k, k-1) + \nu_2(k) \tag{4.35}$$

dove $\epsilon(k, k-1)$ è l'errore sulla stima dello stato al passo k.

$$\epsilon(k, k-1) = X(k) - \hat{X}(k|Z_{k-1}) \tag{4.36}$$

La matrice di correlazione dell'innovazione è definita come:

$$S_\alpha(k) = E[\alpha(k)\alpha^T(k)] \tag{4.37}$$

Riformulandola utilizzando la 4.35 e sapendo che l'errore ϵ è ortogonale al rumore di transizione di stato e al rumore di misurazione, si ottiene:

$$S_\alpha(k) = C(k)P(k, k-1)C^T(k) + Q_2(k) \tag{4.38}$$

dove Q_2 è la matrice di correlazione del processo di rumore v_2. La matrice $P(k, k-1)$ è la matrice di correlazione dell'errore sullo stato, definita come:

$$P(k, k-1) = E[\epsilon(k, k-1)\epsilon^T(k, k-1)] \tag{4.39}$$

dove $\epsilon(k, k-1)$ è l'errore sul vettore di stato. La matrice P è utilizzata come descrizione statistica dell'errore, nella stima dello stato.

4.4.3 Stima dello stato usando il processo di innovazione

Quindi ora consideriamo il problema di ricavare la stima dello stato dal processo di innovazione. Da quanto trattato fino ad ora si deduce che la stima può essere espressa come una combinazione lineare della sequenza di vettori di innovazione: $\{\alpha(1), \alpha(2), ..., \alpha(k)\}$

$$\hat{X}(i|Z_k) = \sum_{n=1}^{k} B_i(n)\alpha(k) \tag{4.40}$$

dove $B_i(n)$ con $n = 1, 2, ..., k$ è un'insieme di matrici da determinare. In accordo al principio di ortogonalità ϵ è ortogonale a $\alpha(k)$:

$$E[\epsilon(i,k)\alpha^T(m)] = E[X(i) - \hat{X}(i|Z_k)]\alpha^T(m) = 0 \tag{4.41}$$

$$m = 1, 2, ...k$$

Quindi sfruttando le ultime due relazioni e la proprietà di ortogonalità si ricava:

$$E[X(i)\alpha^T(m)] = B_i(m)E[\alpha(m)\alpha^T(m)] = B_i(m)S_\alpha(m) \tag{4.42}$$

Da cui si ricava facilmente che:

$$B_i(m) = E[X(i)\alpha^T(m)]S_\alpha^{-1}(m) \tag{4.43}$$

Sostituendo la 4.40 nella 4.43 si ottiene la stima ai minimi quadrati:

$$\hat{X}(i|Z_k) = \sum_{n=1}^{k} E[X(i)\alpha^T(k)]S_\alpha^{-1}(k)\alpha(k) \tag{4.44}$$

dove per $k+1$, si deduce:

$$\hat{X}(k+1|Z_k) = \sum_{n=1}^{k-1} E[X(k-1)\alpha^T(k)]S_\alpha^{-1}(k)\alpha(k) \tag{4.45}$$

Lo stato al momento $k+1$ è correlato con lo stato al momento k, dall'equazione 4.24, quindi per $0 \leq n \leq k$ si ottiene:

$$\begin{array}{rcc} E[X(n+1)\alpha^T(k)] & = & E[A(k+1,k)X(n) + v_1(n)]\alpha^T(k) \\ & = & A(k+1,k)E[X(n)\alpha^T(k)] \end{array} \tag{4.46}$$

dove si è sfruttato il fatto che $\alpha(k)$ dipende solo dai dati osservati $Z(1)$, $Z(2)$, ... , $Z(k)$. Sapendo che $Z(k)$ e $\alpha(k)$ sono ortogonali per $0 \leq k \leq n$ si ottiene dalla equazione 4.45:

$$\begin{array}{c} \sum_{n=1}^{k-1} E[X(k+1)\alpha^T(n)]S_\alpha^{-1}(n)\alpha(n) = \\ = A(k+1,k)\sum_{n=1}^{k-1} E[X(k)\alpha^T(n)]S_\alpha^{-1}(n)\alpha(n) \\ = A(k+1,k)\hat{X}(k|Z_{k-1}) \end{array} \tag{4.47}$$

4.4.4 Il guadagno di Kalman

Si definisce guadagno di Kalman la matrice:

$$G(k) = E[X(k+1)\alpha^T(k)]S_\alpha^{-1}(k) \tag{4.48}$$

Le equazioni precedenti possono essere riscritte sfruttando l'equazione 4.48 nel modo:

$$\hat{X}(k+1|Z_k) = A(k+1,k)\hat{X}(k|Z_{k-1}) + G(k)\alpha(k) \tag{4.49}$$

Il guadagno di Kalman è di fondamentale importanza, infatti dimostra che la stima dello stato può essere calcolata sfruttando la stima precedente, la matrice di transizione di stato e l'innovazione.
Rimane da definire una forma per il calcolo del guadagno di Kalman computazionalmente conveniente. Sfruttando l'equazione del calcolo dell'innovazione e la 4.46 si ottiene:

$$\begin{aligned} E[X(k+1)\alpha^T(k)] &= A(k+1,k)E[X(k)\alpha^T(k)] \\ &= A(k+1,k)E[X(k)(C(k)\epsilon(k,k-1) + v_2(k))^T] \\ &= A(k+1,k)E[X(k)\epsilon^T(k,k-1)]C^T(k) \end{aligned} \tag{4.50}$$

dove si sfrutta che lo stato $X(k)$ ed il rumore $v_2(k)$ sono non correlati. L'errore di previsione sullo stato $\epsilon(k,k-1)$ è ortogonale alla stima stessa. Quindi il loro prodotto è nullo. Ciò permette di riscrivere l'equazione precedente nella forma:

$$E[X(k+1)\alpha^T(k)] = A(k+1,k)E[\epsilon(k,k-1)\epsilon^T(k,k-1)]C^T(k) \tag{4.51}$$

dove $E[\epsilon(k,k-1)\epsilon^T(k,k-1)] = P(k,k-1)$, quindi si ottiene:

$$E[X(k+1)\alpha^T(k)] = A(k+1,k)P(k,k-1)C^T(k) \tag{4.52}$$

Questa relazione ci permette di ridefinire il guadagno di Kalman con la seguente espressione ricorsiva:

$$G(k) = A(k+1,k)P(k,k-1)C^T(k)S_\alpha^{-1}(k) \tag{4.53}$$

dove la $S_\alpha(k)$ è la matrice di correlazione dell'innovazione. La figura 4.2 riporta uno schema a blocchi di come calcolare il guadagno di Kalman.

4.4.5 L'equazione di Riccati

Come visto per il caso in tempo continuo la relazione iterattiva del calcolo della matrice di covarianza corrisponde all'equazione di Riccati. La matrice di

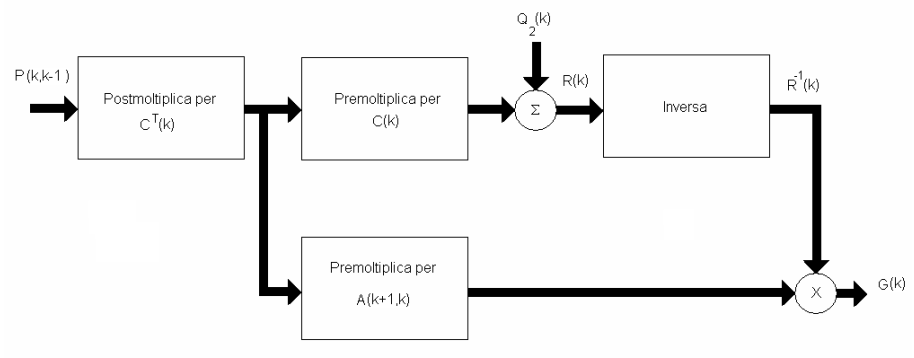

Figura 4.2: Schema a blocchi per il calcolo del guadagno di Kalman

covarianza è indispensabile per il calcolo del guadagno di Kalman. Per giungere alla relazione ricorsiva, che ricava la matrice P si procede come segue:
si consideri l'errore commesso nel valutare la stima dello stato:

$$\epsilon(k+1,k) = X(k+1) - \hat{X}(k+1|Z_k) \tag{4.54}$$

sostituendo l'equazione di transizione di stato (eq. 4.24) e l'equazione di stima-aggiornamento di Kalman (eq. 4.49) all'equazione precedente e utilizzando la funzione per il calcolo dell'innovazione si ottiene:

$$\epsilon(k+1,k) = A(k+1,k)[X(k)-\hat{X}(k|Z_{k-1})]-G(k)[Z(k)-C(k)\hat{X}(k|Z_{k-1})]+v_1(k) \tag{4.55}$$

Poi utilizzando l'equazione di misurazione al posto di $Z(k)$, si ottiene la seguente equazione alle differenze per il calcolo ricorsivo dell'errore sullo stato:

$$\epsilon(k+1,k) = [A(k+1,k) - G(k)C(k)]\epsilon(k,k-1) + v_1(k) - G(k)v_2(k) \tag{4.56}$$

Sapendo che la matrice di correlazione è definita come:

$$P(k+1,k) = E[\epsilon(k+1,k)\epsilon^T(k+1,k)] \tag{4.57}$$

dove sostituendo e considerando che i processi di rumore sono incorrelati si ottiene:

$$P(k+1,k) = [A(k+1,k) - G(k)C(k)]P(k,k-1)[A(k+1,k) - G(k)C(k)]^T + Q_1(k) + G(k)Q_2(k)G^T(k) \tag{4.58}$$

dove $Q_1(k)$ e $Q_2(k)$ sono le matrici di correlazione dei processi di rumore $v_1(k)$ e $v_2(k)$.
Espandendo l'equazione e inserendo la relazione per il calcolo del guadagno e la relazione per il calcolo della covarianza dell'innovazione si ottiene l'equazione alle differenze di Riccati, per il calcolo ricorsivo della matrice di correlazione dell'errore sulla stima dello stato:

$$P(k+1,k) = A(k+1,k)P(k)A^T(k+1,k) + Q_1(k) \tag{4.59}$$

che sfruttando la proprietà della matrice di transizione di stato:

$$A(k+1,k)A(k,k+1) = I$$

può essere espressa meglio nel seguente modo:

$$P(k+1,k) = P(k,k-1) - A(k,k+1)G(k)C(k)P(k,k-1) \tag{4.60}$$

La matrice di correlazione offre una descrizione statistica dell'errore nella stima dello stato, grazie alla quale è possibile correggere quest'ultima.

Le equazioni:

$$G(k) = A(k+1,k)P(k,k-1)C^T(k)S_\alpha^{-1}(k)$$

$$S_\alpha(k) = C(k)P(k,k-1)C^T(k) + Q_2(k)$$

$$\alpha(k) = Z(k) - C(k)\hat{X}(k|Z_{k-1})$$

$$\hat{X}(k+1|Z_k) = A(k+1,k)\hat{X}(k|Z_{k-1}) + G(k)\alpha(k)$$

$$P(k+1,k) = P(k,k-1) - A(k,k+1)G(k)C(k)P(k,k-1)$$

$$P(k+1,k) = A(k+1,k)P(k)A^T(k+1,k) + Q_1(k)$$

rappresentano un passo dell'algoritmo di predizione di Kalman.

4.4.6 L'operazione di filtraggio

Il passo successivo è l'operazione di filtraggio per il calcolo della stima di $\hat{X}(k|Z_k)$. Si noti che lo stato $X(k)$ ed il rumore $\hat{v}_1(k|Z_k)$ al passo k sono non correlati. Dall'equazione di stato si ricava che la stima migliore dello stato al momento $k+1$ è:

$$\hat{X}(k+1|Z_k) = A(k+1,k)\hat{X}(k|Z_k) + \hat{v}_1(k|Z_k) \tag{4.61}$$

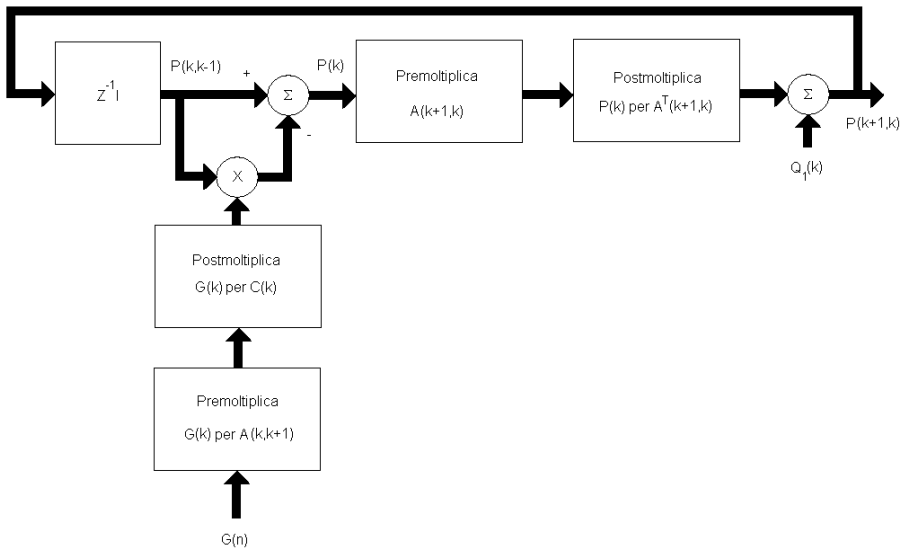

Figura 4.3: Schema a blocchi del calcolo della matrice di covarianza

Il vettore di rumore $v_1(k)$ è indipendente dai dati osservati, questo comporta che dalla stima dello stato precedente, il rumore possa essere considerato nullo:

$$\hat{X}(k+1|Z_k) = A(k+1,k)\hat{X}(k|Z_k) \tag{4.62}$$

da cui si ricava che la stima dello stato al passo k è:

$$\hat{X}(k|Z_k) = A^{-1}(k+1,k)\hat{X}(k+1|Z_k) \tag{4.63}$$

Usando la proprietà:

$$A^{-1}(k+1,k) = A(k,k+1) \tag{4.64}$$

Quindi:

$$\hat{X}(k|Z_k) = A(k,k+1)\hat{X}(k+1|Z_k) \tag{4.65}$$

Questo dimostra che conoscendo la soluzione del passo di predizione, si ottiene la stima dello stato moltiplicando per la funzione di transizione.

4.4.7 L'errore della stima di filtraggio

L'operazione di filtraggio, inevitabilmente, non potrà essere precisa. L'errore commesso è quantificabile come segue:

$$e(k) = Z(k) - C(k)\hat{X}(k|Z_k) \tag{4.66}$$

La definizione è simile a quella data per l'innovazione, cambia il termine $\hat{X}(k|Z_k)$ che per l'innovazione è la stima a priori $\hat{X}(k|Z_{k-1})$.
Utilizzando la funzione per il calcolo del guadagno e la 4.65, si ottiene:

$$\begin{aligned} e(k) &= Z(k) - C(k)\hat{X}(k|Z_{k-1}) - C(k)A(k,k+1)G(k)\alpha k \\ &= \alpha(k) - C(k)A(k,k+1)G(k)\alpha(k) \\ &= [I - C(k)A(k,k+1)G(k)]\alpha(k) \end{aligned} \tag{4.67}$$

La quantità:

$$r^{-1} = I - C(k)A(k,k+1)G(k) \tag{4.68}$$

è definita come fattore di conversione. Fornisce una formula per convertire il valore della innovazione nel valore di errore di stima.

Utilizzando la relazione (eq. 4.53) che cancella il guadagno di Kalman possiamo scrivere:

$$e(k) = Q_2(k)S_\alpha^{-1}(k)\alpha(k) \tag{4.69}$$

dove Q_2 è la matrice di correlazione del processo di rumore $v_2(k)$.

4.5 Il filtro di Kalman esteso - EKF

I sistemi reali sono quasi sempre non lineari, mentre il filtro di Kalman tratta sistemi di natura lineare. Per applicare l'algoritmo, bisogna creare una condizione in cui i sistemi non-lineari siano considerabili come lineari. Lo scopo dei filtri estesi di Kalman è risolvere quanto appena accennato.

Prima di procedere nello sviluppo di questa nuova serie di algoritmi, riportiamo, per cortesia al lettore, l'insieme di relazioni che governano il filtro:

$$\begin{cases} \hat{X}(k+1|Z_{k-1}) = & A(k+1,k)\hat{X}(k|Z_{k-1}) + v_1(k) \\ \widehat{Z}(k) = & C(k)\hat{X}(k|Z_{k-1}) + v_2(k) \end{cases} \tag{4.70}$$

$$\alpha(k) = Z(k) - C(k)\hat{X}(k|Z_{k-1}) \tag{4.71}$$

$$S_\alpha(k) = C(k)P(k,k-1)C^T(k) + Q_2(k) \tag{4.72}$$

$$P(k+1,k) = A(k+1,k)P(k)A^T(k+1,k) + Q_1(k) \tag{4.73}$$

$$G(k) = A(k+1,k)P(k,k-1)C^T(k)S_\alpha^{-1}(k) \tag{4.74}$$

$$P(k+1,k) = P(k,k-1) - A(k,k+1)G(k)C(k)P(k,k-1) \tag{4.75}$$

$$\hat{X}(k|Z_k) = \hat{X}(k|Z_{k-1}) + G(k)\alpha(k) \tag{4.76}$$

Si è detto che i sistemi reali sono tradizionalmente non lineari:

$$\begin{cases} \hat{X}(k+1|Z_{k-1}) = & f(\hat{X}(k|Z_{k-1}), U(k), v_1(k)) \\ \hat{Z}(k) = & h(\hat{X}(k), v_2(k)) \end{cases} \tag{4.77}$$

L'idea di base del filtro di Kalman esteso è di linearizzare il modello di stato attorno al punto della stima più recente, che potrebbe essere $\hat{X}(k|Z_k)$ oppure $\hat{X}(k|Z_{k-1})$, a seconda della funzione considerata. Ottenuto un modello di stato lineare si può applicare il filtro di Kalman.

L'approssimazione procede in due passi:

Passo 1 Si costruiscono le seguenti matrici:

$$A(k+1,k) = \left.\frac{\delta f(k,X)}{\delta X}\right|_{X=\hat{X}(k|Z_k)} \tag{4.78}$$

$$C(k) = \left.\frac{\delta C(k,X)}{\delta X}\right|_{X=\hat{X}(k|Z_{k-1})} \tag{4.79}$$

Quindi l'elemento (i,j) della matrice A(k+1,k) è uguale alla derivata parziale della componente i-esima della funzione $f(k,X)$ rispetto la variabile j-esima di X. Analogamente per la matrice $C(k)$

Passo 2 Calcolate le matrici nei punti d'interesse espandiamo le funzioni non lineari di partenza in serie di Taylor:

$$f(k,X(k)) \simeq f(k,\hat{X}(k|Z_k)) + A(k+1,k)[X(k) - \hat{X}(k|Z_k)] \tag{4.80}$$

$$C(k,X(k)) \simeq h(n,\hat{X}(k|Z_{k-1})) + C(k)[X(k) - \hat{X}(k|Z_{k-1})] \tag{4.81}$$

Con l'approssimazione introdotta procediamo con la linearizzazione delle equazioni di stato:

$$X(k+1) \simeq A(k+1,k)X(k) + v_1(k) + l(k) \tag{4.82}$$

$$\overline{Z}(k) \simeq C(k)X(k) + v_2(k) \tag{4.83}$$

dove sono state introdotte due nuove quantità:

$$\begin{aligned} \overline{Z}(k) &= Z(k) - [C(k, \hat{X}(k|Z_{k-1})) - C(k)\hat{X}(k|Z_{k-1})] \quad (4.84) \\ l(k) &= f(k, \hat{X}(k|Z_{k-1}) - A(k+1,k)\hat{X}(k|Z_k) \quad (4.85) \end{aligned}$$

che sono note ad ogni passaggio.

Da quanto visto si ricava il seguente insieme di equazioni:

$$\begin{aligned} \hat{X}(k+1|Z_k) &= A(k+1,k)\hat{X}(k|Z_k) + l(k) \\ &= A(k+1,k)\hat{X}(k|Z_k) + f(k, \hat{X}(k|Z_{k-1}) - A(k+1,k)\hat{X}(k|Z_k) \\ &= f(k, \hat{X}(k|Z_k)) \\ \hat{X}(k|Z_k) &= \hat{X}(k|Z_{k-1}) + G(k)\alpha(k) \\ \alpha(k) &= \overline{Z}(k) - C(k)\hat{X}(k|Z_{k-1}) \\ &= Z(k) - [C(k, \hat{X}(k|Z_{k-1})) - C(k)\hat{X}(k|Z_{k-1})] \\ &\quad -C(k)\hat{X}(k|Z_{k-1}) \\ &= Z(k) - C(k, \hat{X}(k|Z_{k-1})) \end{aligned} \quad (4.86)$$

Da qui si ricava facilmente l'insieme di equazioni che implementano il filtro esteso di Kalman:

$$\hat{X}(k|Z_{k-1}) = f(\hat{X}(k-1|Z_{k-1}), U(k)) \quad (4.87)$$

$$P(k|Z_{k-1}) = \nabla_x f(k) P(k-1|Z_{k-1}) \nabla_x f^T(k) + Q(k) \quad (4.88)$$

$$\hat{Z}(k) = h(\hat{X}(k|Z_{k-1})) \quad (4.89)$$

$$\alpha(k) = Z(k) - \hat{Z}(k) \quad (4.90)$$

$$S(k) = \nabla_x h(k) P(k|Z_{k-1}) \nabla_x h^T(k) + R(k) \quad (4.91)$$

$$G(k) = P(k|Z_{k-1}) \nabla_x h^T(k) S_\alpha^{-1}(k) \quad (4.92)$$

$$\begin{aligned} \hat{X}(k|Z_k) &= \hat{X}(k|Z_{k-1}) + G(k)\nu(k) \quad (4.93) \\ P(k|Z_k) &= P(k|Z_{k-1}) - G(k)S_\alpha(k)G^T(k) \quad (4.94) \end{aligned}$$

Capitolo 5

Recursive Least Square Algorithm

I filtri RLS, come il nome lascia intuire[1], sono dei filtri adattativi di tipo ricorsivo. Grazie a questa caratteristica le stime offerte dall'algoritmo dipendono dall'istante precedente e dall'ingresso al tempo attuale. Quindi non è necessario immagazzinare molti dati e processarli di volta in volta.

5.1 Introduzione ai filtri RLS

Nella implementazione ricorsiva [4] del metodo ai minimi quadrati, partendo da condizioni iniziali note, si utilizza l'informazione contenuta nei campioni in ingresso per correggere la stima precedente.
Lo scopo di questo genere di filtri è di minimizzare la funzione di errore:

$$J(k) = \sum_{n=0}^{k} \lambda^{k-n} |d(n) - H(n)^T X(n)|^2 \tag{5.1}$$

dove $d(n)$ e l'uscita desiderata del sistema,

$$H(k)^T = [a_0(k), a_1(k), ..., a_{N-1}(k)] \tag{5.2}$$

[1] analogamente ai filtri di Kalman

è la funzione di trasferimento del filtro all'istante k e

$$X(k)^T = [x(k), x(k-1), ..., x(k-N+1)] \tag{5.3}$$

è il vettore di ingresso all'istante k.

L'obbiettivo è di calcolare $H(k)$ al fine di minimizzare appunto $J(k)$. La soluzione è data dalla relazione:

$$H(k) = R_{XX}^{-1}(k) R_{Xd}(k) \tag{5.4}$$

dove

$$R_{XX}(k) = \sum_{n=0}^{k} \lambda^{k-n} X(n) X(n)^T \tag{5.5}$$

rappresenta la matrice di autocorrelazione e

$$R_{Xd}(k) = \sum_{n=0}^{k} \lambda^{k-n} X(n) d(k) \tag{5.6}$$

rappresenta la matrice di crosscorrelazione.
λ è una costante positiva $0 \ll \lambda \leq 1$, l'inverso di $1 - \lambda$ rappresenta, in parole povere, una misura della memoria dell'algoritmo. Tanto più λ si avvicina ad 1 tanto più i campioni passati saranno tenuti in considerazione. Per $\lambda = 1$ si ha memoria infinita.

5.2 RLS standard

Dalle equazioni 5.5 e 5.6 si ricavano le seguenti formulazioni ricorsive:

$$R_{XX}(k) = \sum_{n=1}^{k} \lambda^{k-n} X(n) X^T(n)$$

$$R_{XX}(k) = \lambda \left[\sum_{n=1}^{k-1} \lambda^{k-1-n} X(n) X^T(n) \right] + X(k) X^T(k)$$

$$R_{XX}(k) = \lambda R_{XX}(k-1) + X(k) X^T(k) \tag{5.7}$$

e procedendo analogamente per la seconda si ricava:

$$R_{Xd}(k) = \sum_{n=1}^{k} \lambda^{k-n} X(n)d(n)$$

$$R_{Xd}(k) = \lambda[\sum_{n=1}^{k-1} \lambda^{k-1-n} X(n)d(n)] + X(k)d(k)$$

$$R_{Xd}(k) = \lambda R_{Xd}(k-1) + X(k)d(k) \tag{5.8}$$

Lo scopo di questa trattazione è di calcolare una stima di $H(k)$. In accordo con l'equazione 5.4 si deve determinare l'inversa della 5.5, ovvero della matrice di autocorrelazione di ingresso. In pratica, come sempre, si cerca di evitare simili procedimenti col fine di limitare il costo computazionale dell'algoritmo, specialmente se il vettore $H(k)$ è di grandi dimensioni. Quindi si vuole determinare la stima ai minimi quadrati di $H(k)$ attraverso una forma ricorsiva.
Per un simile intento si sfrutta un postulato basilare dell'algebra, meglio conosciuto come il lemma di inversione delle matrici.

5.2.1 Lemma di inversione delle matrici

Ipotesi:
Date A e B matrici $M \times M$ definite positive in relazione da

$$A = B^{-1} + CD^{-1}C^H \tag{5.9}$$

dove D è un'altra matrice definita positiva di dimensioni $N \times M$ e C è una matrice $M \times N$;
Tesi:
Si può esprimere l'inversa della matrice A come:

$$A^{-1} = B - BC(D + C^H BC)^{-1} C^H B \tag{5.10}$$

La dimostrazione si ottiene moltiplicando la 5.9 per la 5.10 e verificando che il prodotto dia la matrice identica.

5.2.2 Calcolo di $H(k)$

Con la conoscenza acquisita dall'identità di Woodbury[2] si è in grado di invertire la matrice di correlazione d'ingresso in modo efficiente.

[2]In letteratura il lemma di inversione delle matrici è conosciuto anche come identità di Woodbury. Le origini esatte di questo lemma non sono note, Householder (1964) ne attribuì

Ponendo:

$$\begin{aligned} A &= R_{XX}(k) \\ B^{-1} &= \lambda R_{XX}(k-1) \\ C &= X(k) \\ D &= 1 \end{aligned}$$

e sostituendo queste definizioni nel lemma di inversione delle matrici, otteniamo la seguente relazione:

$$R_{XX}^{-1}(k) = \lambda^{-1} R_{XX}^{-1}(k-1) - \frac{\lambda^{-2} R_{XX}^{-1}(k-1) X(k) X^T(k) R_{XX}^{-1}(k-1)}{1 + \lambda^{-1} X^T(k) R_{XX}^{-1}(k-1) X(k)} \quad (5.11)$$

Per convenienza poniamo

$$K(k) = R_{XX}^{-1}(k)$$

e definiamo

$$C(k) = \frac{\lambda^{-1} K(k-1) X(k)}{1 + \lambda^{-1} X^T(k) K(k-1) X(k)} \quad (5.12)$$

quindi dall'equazione 5.11 si ricava:

$$K(k) = \lambda^{-1} K(k-1) - \lambda^{-1} C(k) X^T(k) K(k-1) \quad (5.13)$$

$K(k)$ rappresenta l'inversa della matrice di correlazione di ingresso, $C(k)$ rappresenta il vettore del guadagno di Kalman. L'equazione 5.13 rappresenta l'equazione di Riccati legata all'algoritmo RLS. Arrangiando la eq. 5.12 si otteniene:

$$C(k) = [\lambda^{-1} K(k-1) - \lambda^{-1} C(k) X^T(k) K(k-1)] X(k) \quad (5.14)$$

dove la parte compresa fra le parentesi della 5.14 rappresenta la $K(k)$:

$$C(k) = K(k) X(k) \quad (5.15)$$

ovvero:

$$C(k) = R_{XX}^{-1}(k) X(k) \quad (5.16)$$

il merito a Woodbury (1950). Le prime applicazioni di elaborazione digitale, riportate in letteratura, di questo metodo furono condotte da Kailath, egli usò il lemma di inversione delle matrici per provare l'equivalenza tra i filtri di Wiener e la procedura di massima probabilità per la stima dell'uscita di un canale di trasmissione, lineare tempo-invariante, sotto l'azione di un rumore bianco(Kailath, 1960).

Gettate le basi fondamentali, si può quindi ricavare la relazione ricorsiva di $H(k)$. Considerando le equazioni 5.4, 5.8 e ricordando l'equivalenza $K(k) = R_{XX}^{-1}(k)$ si ricava:

$$H(k) = R_{XX}^{-1} R_{Xd}(k)$$

$$H(k) = K(k) R_{Xd}(k)$$

$$H(k) = \lambda K(k) R_{Xd}(k-1) + K(k) X(k) \hat{d}(k) \tag{5.17}$$

quindi applicando la 5.13 per il primo addendo a secondo membro della eq. 5.17 si ottiene:

$$\begin{aligned} H(k) &= K(k-1)R_{Xd} - C(k)X(k)K(k-1)R_{Xd}(k-1) + K(k)X(k)\hat{d}(k) \\ &= R_{XX}^{-1}(k-1)R_{Xd} - C(k)X(k)R_{XX}^{-1}(k-1)R_{Xd}(k-1) + K(k)X(k)\hat{d}(k) \\ &= H_{k-1} - C(k)X^T(k)H(k-1) + K(k)X(k)\hat{d}(k) \end{aligned}$$

Infine considerando che $K(k)X(k)$ è uguale al vettore guadagno $C(k)$, si è in grado di fornire l'equazione per il calcolo ricorsivo di $H(k)$:

$$H(k) = H(k-1) + C(k)[\hat{d}(k) - X^T(k)H(k-1)] = H(k-1) + C(k)\xi(k) \tag{5.18}$$

dove $\xi(k)$ è la stima dell'errore a priori definita come:

$$\xi(k) = d(k) - H(k-1)^T X(k) \tag{5.19}$$

basata sulla stima precedente del vettore H all'istante $k-1$. La stima a priori dell'errore è differente, in generale, dalla stima a posteriori:

$$e(k) = d(k) - H(k)^T X(k) \tag{5.20}$$

5.3 RLS SQR

Uno dei problemi di maggior rilievo evidenziati nella trattazione finora illustrata degli algoritmi RLS è senz'altro l'instabilità numerica, dovuta al calcolo ricorsivo dell'equazione di Riccati eq.5.13.
L'analogo problema è valido per il filtro classico di Kalman.

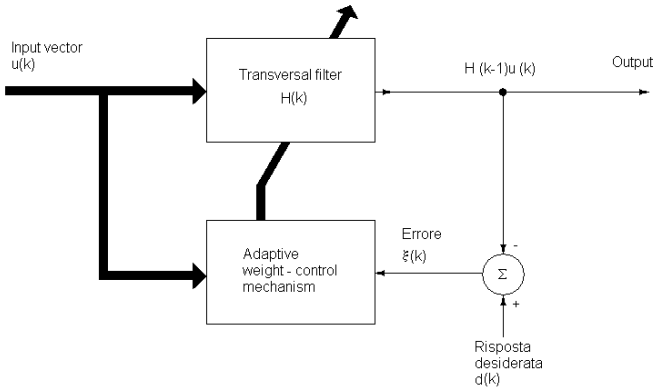

Figura 5.1: Schema a blocchi algoritmo RLS

Una soluzione al problema è costituita dall'utilizzo di una variante del filtro che frutta la decomposizione square-root.

$$R_{XX}^{-1}(k) = K(k) = S_k S_k^T \tag{5.21}$$

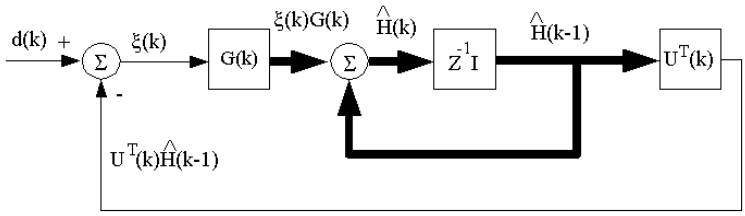

Figura 5.2: Flusso dei segnali dell'algoritmo RLS

Dalla quale si possono ricavare le seguenti relazioni[12]:

$$F(k) = S_{k-1}^T H(k) \tag{5.22}$$
$$L(k) = S_{k-1} F(k) \tag{5.23}$$
$$\beta(k) = \lambda + F^T(k) F(k) \tag{5.24}$$
$$\alpha(k) = \frac{1}{\beta(k) + \sqrt{\lambda \beta(k)}} \tag{5.25}$$
$$S_k = \frac{1}{\sqrt{\lambda}} \left[S_{k-1} - \alpha(k) L(k) F^T(k) \right] \tag{5.26}$$
$$C(k) = \frac{L(k)}{\beta(k)} \tag{5.27}$$
$$H(k) = H(k-1) + C(k) \xi(k) \tag{5.28}$$

5.4 Fast RLS SQR

Si deve sviluppare una versione iterativa dell'equazione 5.4 con il minor numero di operazioni possibili per iterazione, in modo che sia numericamente stabile [5].

5.4.1 Introduzione dell'algoritmo

Considerando il vettore $X(k)$, per convenienza di notazione, si definisce $r_{k-1} = x(k-N)$ e $v_k = x(k)$ dove il primo rappresenta l'ultimo elemento del vettore di ingresso all'istante $k-1$ ed il secondo rappresenta il primo elemento del vettore di ingresso all'istante k. Si definisce predittore a priori la matrice A_k, e predittore a posteriori la matrice B_k. Le predizioni a priori e a posteriori dell'errore sono date dalle relazioni:

$$f_k(k) = v_k + A_k^T X(k-1) \tag{5.29}$$

$$b_k(k) = r_{k-1} + B_k^T X(k-1) \tag{5.30}$$

Il guadagno di Kalman è definito come:

$$C(k) = R_{XX}^{-1}(k) X(k) \tag{5.31}$$

Dalla equazione

$$H(k) = R_{XX}^{-1}(k) R_{Xd}(k)$$

si può vedere il guadagno di Kalman come un predittore per la sequenza di ingresso $X(k)$. L'errore di predizione corrispondente è definito:

$$\gamma_k = 1 - C_k^T X(k) \tag{5.32}$$

La matrice A_k può essere calcolata in modo ricorsivo, sfruttando il guadagno di Kalman:

$$A_k = A_{k-1} - C_{k-1} f_{k-1}(k) \tag{5.33}$$

Analogamente per la matrice B_k:

$$B_k = B_{k-1} - C_{k-1} b_{k-1}(k) \tag{5.34}$$

Si definisce inoltre il vettore di ingresso esteso $\overline{X}(k)$ come visibile nella seguente equazione:

$$\overline{X}(k) = \begin{bmatrix} v_k \\ X(k-1) \end{bmatrix} = \begin{bmatrix} X(k) \\ r_{k-1} \end{bmatrix} \tag{5.35}$$

Per concludere questa introduzione si riportano le espressioni per il calcolo delle autocorrelazioni dell'errore a priori e a posteriori:

$$\alpha_k = \sum_{n=0}^{k} \lambda^{k-n} f_k^2(n) \tag{5.36}$$

$$\beta_k = \sum_{n=0}^{k} \lambda^{k-n} b_k^2(n) \tag{5.37}$$

Il filtro fast RLS si basa sulle relazioni appena descritte.

5.4.2 Risultati preliminari

Proposizione 1

Dati la predizione a priori dell'errore $f_k(k)$, la predizione a posteriori $b_k(k)$ e l'ingresso $X(k-1)$, valgono le seguenti relazioni:

$$\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} f_k(n) = 0$$

$$\sum_{n=0}^{k} \lambda^{k-n} X_n b_k(n) = 0$$

La dimostrazione è riportata in appendice.

Proposizione 2

Date le autocorrelazioni della predizione dell'errore α_k, β_k e i corrispondenti predittori d'errore A_k, B_k, valgono le seguenti relazioni:

$$\sum_{n=0}^{k} \lambda^{k-n} v_n^2 = \alpha_k + A_k^T R_{XX}(k-1) A_k$$

$$\sum_{n=0}^{k} \lambda^{k-n} r_{n-1}^2 = \beta_k + B_k^T R_{XX}(k) B_k$$

La dimostrazione è riportata in appendice.

Proposizione 3

Dati l'ingresso $X(k)$ e le matrici A_k e B_k valgono le due relazioni:

$$\sum_{n=0}^{k} \lambda^{k-n} X(n-1) v_n = -R_{XX}(k-1) A_k$$

$$\sum_{n=0}^{k} \lambda^{k-n} X(n)^T r_{n-1} = -B_k^T R_{XX}(k)$$

La dimostrazione è riportata in appendice.

5.4.3 Fattorizzazione square root della matrice di correlazione

L'algoritmo descritto come già detto frutta la decomposizione square root dell'inversa della matrice di autocorrelazione:

$$R_{XX}^{-1}(k) = S_k S_k^T \tag{5.38}$$

dove S_k è la radice quadrata della matrice R_{XX}^{-1}.

Si riporta di seguito la forma estesa della matrice di correlazione:

$$\overline{R}_{XX}(k) = \sum_{n=0}^{k} \lambda^{k-n} \overline{X}_n \overline{X}_n^T \tag{5.39}$$

e la corrispondente decomposizione:

$$\overline{R}_{XX}^{-1}(k) = \overline{S}_k \overline{S}_k^T \tag{5.40}$$

La fattorizzazione è unica se la matrice S_k è triangolare superiore con diagonale positiva[3]. Il lemma seguente descrive la forma della matrice radice quadrata.

Lemma 1

Si consideri la matrice radice quadrata della inversa della matrice di correlazione dell'ingresso, la matrice radice quadrata estesa è calcolabile con la forma:

$$\overline{S}_k = \begin{pmatrix} S_k & \beta_k^{-1/2B_k} \\ \mathbf{0}^T & \beta_k^{-1/2} \end{pmatrix} \tag{5.41}$$

dove il vettore (0) è il vettore nullo di N elementi, β_k è l'autocorrelazione dell'errore a posteriori, B_k è il predittore a posteriori e S_k è la radice quadrata di R_{XX}^{-1}.
La dimostrazione è riportata in appendice.

Lemma 2

Si consideri la fattorizzazione della inversa della matrice di autocorrelazione:

$$\overline{R}_{XX}{-}1(k) = \widehat{S}_k \widehat{S}_k^T \tag{5.42}$$

Quindi una possibile forma per la matrice di $\widehat{S}_k$ è la seguente:

$$\widehat{S}_k = \begin{pmatrix} \alpha_k^{-1/2} & \mathbf{0}^T \\ \alpha_k^{-1/2} A_k & S_{k-1} \end{pmatrix} \tag{5.43}$$

dove A_k è l'errore a priori e S_k è la decomposizione della matrice di correlazione. Per la dimostrazione si rimanda all'appendice.

Si possiedono ora due possibili decomposizioni della matrice di autocorrelazione. Nel lemma seguente si propone un'altra forma della matrice triangolare superiore, decomposizione della matrice di correlazione. La matrice seguente è equivalente alla decomposizione proposta nel lemma 1.

[3]Fattorizzazione di Cholesky[15]

Lemma 3

Si consideri la radice quadrata dell'inversa della matrice di autocorrelazione estesa. Quindi la matrice radice quadrata triangolare superiore $\overline{S}_k$ è la seguente:

$$\overline{S}_k = \begin{pmatrix} \sqrt{\frac{\alpha_k^{-1}}{1+\alpha_k^{-1} Z_k^T Z_k}} & \frac{\alpha_k^{-1}}{1+\alpha_k^{-1} Z_k^T Z_k} Z_k^T L_k \\ \mathbf{0} & S_{k-1} L_k \end{pmatrix} \tag{5.44}$$

dove Z_k è definita come

$$Z_k = S_{k-1}^{-1} A_k \tag{5.45}$$

e L_k è la seguente matrice triangolare superiore data dalla decomposizione:

$$(I + \alpha_k^{-1} Z_k Z_k^T) = L_k L_k^T \tag{5.46}$$

Per la dimostrazione si rimanda all'appendice.

L'approccio ai filtri RLS attraverso la decomposizione della matrice di correlazione aumenta la stabilità numerica dell'algoritmo. L'inversa della matrice di correlazione è fattorizzata secondo l'equazione 5.38. Se le equazioni ricorsive sono espresse in termini della matrice S_k, la matrice $R_{XX}^{-1}(k)$ è definita sicuramente positiva. Consideriamo il vettore:

$$D_k = S_k^T X^k \tag{5.47}$$

dove X_k è il vettore di ingresso. Aggiornando ricorsivamente la matrice D_k è implicitamente garantito che la matrice R_{XX}^{-1} è definita positiva. Ogni relazione ricorsiva può essere tratta usando la fattorizzazione ricavata nei lemmi 1 e 3.

Teorema 1

Dato il filtro lineare $d_k(k) = H_k^T X_k$, un algoritmo stabile numericamente che minimizza la funzione costo 5.1 in $O(N)$ operazioni è quello descritto nelle

equazioni 5.48 - 5.57.

$$
\begin{aligned}
f_{k-1}(k) &= v_k + D_{k-1}^T(\sqrt{\lambda}T_{k-1}^{-1}Z_{k-1}) && (5.48)\\
f_k(k) &= \gamma_{k-1}f_{k-1}(k) && (5.49)\\
Z_k &= \sqrt{\lambda}T_{k-1}^{-1}Z_{k-1} - D_{k-1}f_{k-1}(k) && (5.50)\\
\alpha_k &= \lambda\alpha_{k-1} + f_k(k)f_{k-1}(k) && (5.51)\\
\sigma_k &= \lambda\sigma_{k-1} + v_k^2 && (5.52)\\
\overline{D}_k &= \begin{bmatrix} D_k \\ \beta_k^{-1/2}b_k(k) \end{bmatrix} = \begin{bmatrix} v_k\sigma_k^{-1/2} \\ L_k^T(v_k\sigma_k^{-1}Z_k + D_{k-1}) \end{bmatrix} && (5.53)\\
\gamma_k &= \gamma_{k-1} - \alpha_k^{-1}f_k^2(k) + \beta_k^{-1}b_k^2(k) && (5.54)\\
e_k(k) &= \gamma_k e_{k-1}(k) && (5.55)\\
e_{k-1} &= d(k) - D_k^T(\sqrt{\lambda}T_k^{-1}Y_{k-1}) && (5.56)\\
Y_k &= \sqrt{\lambda}T_k^{-1}Y_{k-1} + D_k e_{k-1}(k) && (5.57)
\end{aligned}
$$

Dimostrazione

Le dimostrazioni delle equazioni 5.49, 5.51 e 5.55 possono essere trovate in [4]. Usando la definizione di $R_{XX}(k)$ 5.5 e 5.21 si può ricavare la seguente relazione:

$$S_k^{T^{-1}}S_k^{-1} = \lambda S_{k-1}^{T^{-1}}S_{k-1}^{-1} + X_k X_k^T \tag{5.58}$$

Quindi si introduce la seguente fattorizzazione:

$$I - D_k D_k^T = T_k^T T_k \tag{5.59}$$

con la matrice T_k triangolare superiore. Invertendo l'equazione precedente ed applicando il lemma di inversione delle matrici, possiamo scrivere:

$$I - \gamma_k^{-1}D_k D_k^T = T_k^{-1}T_k^{T^{-1}} \tag{5.60}$$

Si ottiene così una forma simile alla precedente semplicemente dalla inversa. Successivamente si vedranno degli algoritmi efficienti per questa decomposizione. Applicando la 5.47 e la 5.59, la forma 5.58 diventa:

$$\lambda S_{k-1}^{T^{-1}}S_{k-1}^{-1} = S_k^{T^{-1}}T_k^T T_k S_k^{-1} \tag{5.61}$$

dove da quest'ultima equazione si ricava che:

$$\sqrt{\lambda}S_{k-1}^{-1} = T_k S_k^{-1} \tag{5.62}$$

quindi:

$$S_k^{-1} = \sqrt{\lambda} T_k^{-1} S_{k-1}^{-1} \tag{5.63}$$

invertendo l'equazione ad ambo i membri:

$$S_k = \frac{1}{\sqrt{\lambda}} S_{k-1} T_k \tag{5.64}$$

Oltretutto dalla equazione 5.29 si ottiene:

$$f_{k-1}(k) = v_k + X_{k-1}^T A_{k-1} \tag{5.65}$$

Pre-moltiplicando A_{k-1} per $S_{k-1} S_{k-1}^{-1}$ e utilizzando l'equazione 5.47, 5.63 e la $Z_k = S_{k-1}^{-1} A_k$ si ottiene la (eq. 5.48):

$$f_{k-1}(k) = v_k + D_{k-1}^T (\sqrt{\lambda} T_{k-1}^{-1} Z_{k-1})$$

Considerando la 5.33 si ottiene facilmente:

$$A_k = A_{k-1} - S_{k-1} D_{k-1} f_{k-1}(k).$$

Da qui si ricava una nuova espressione per la matrice Z_k:

$$Z_k = S_{k-1}^{-1} A_{k-1} - D_{k-1} f_{k-1}(k) \tag{5.66}$$

utilizzando l'equazione 5.63 l'ultima relazione scritta diventa (eq. 5.50):

$$Z_k = \sqrt{\lambda} T_{k-1}^{-1} Z_{k-1} - D_{k-1} f_{k-1}(k) \tag{5.67}$$

Usando la definizione di $\overline{D}_k$ e usando il lemma 1 e l'equazione 5.30 si ottiene:

$$\overline{D}_k = \overline{S}_k^T \begin{bmatrix} X_k \\ r_{k-1} \end{bmatrix} = \begin{bmatrix} D_k \\ \beta_k^{-1/2} b_k(k) \end{bmatrix} \tag{5.68}$$

In linea di principio, dall'ultima relazione riportata si vede che D_k è il "vettore errore di predizioni a posteriori normalizzate"[4]. I suoi elementi corrispondono all'errore a posteriori per i filtri di diversi ordini. Usando il lemma 3, il vettore $\overline{D}_k$ può essere espresso anche come:

$$\overline{D}_k = S_k^T \begin{bmatrix} v_k \\ X_{k-1} \end{bmatrix} = \begin{bmatrix} v_k \sqrt{\frac{\alpha_k^{-1}}{1+\alpha_k^{-1} Z_k^T Z_k}} \\ L_k^T \left(\frac{\alpha_k^{-1} v_k}{1+\alpha_k^{-1} Z_k^T Z_k} Z_k + D_{k-1} \right) \end{bmatrix} \tag{5.69}$$

Naturalmente la 5.68 è equivalente alla 5.69. L'equazione 5.69 può essere semplificata considerando che il primo coefficiente di D_k è la predizione a posteriori dell'errore normalizzata del predittore di ordine zero. In altre parole il campione v_k diviso per l'energia dell'errore a posteriori $\sqrt{\sigma_k}$, dove

$$\sigma_k = \sum_{n=0}^{k} \lambda^{k-n} v_n^2 = \lambda \sigma_{k-1} + v_k^2 \tag{5.70}$$

dimostra l'equazione 5.52. Quindi sfruttando la 5.69 si può ricavare che:

$$\frac{v_k}{\sqrt{\sigma_k}} = v_k \sqrt{\frac{\alpha_k^{-1}}{1 + \alpha_k^{-1} Z_k^T Z_k}} \tag{5.71}$$

dalla quale si ottiene facilmente la relazione 5.53 per D_k e $\beta_k^{-1/2} b_k(k)$.

Dalle equazioni 5.32, 5.31 e 5.47, si ottiene:

$$\begin{aligned} \gamma_k &= 1 - C_k^T X_k \\ &= 1 - X_k^T R_{XX}^{-1}(k) X_k \\ &= 1 - D_k^T D_k \end{aligned} \tag{5.72}$$

Quindi:

$$\overline{\gamma}_k = 1 - \overline{D}_k^T \overline{D}_k \tag{5.73}$$

Considerando la 5.68 può essere espressa come:

$$\overline{\gamma}_k = 1 - D_k^T D_k - \beta_k^{-1} b_k^2(k) \tag{5.74}$$

che combinata con l'equazione 5.72 diventa:

$$\gamma_k = \overline{\gamma}_k + \beta_k^{-1}(k) b_k^2(k) \tag{5.75}$$

Quindi da $\gamma_k = \overline{\gamma}_k - \alpha_k^{-1}(k) f_k^2(k)$ come descritto in [4] si ottiene la 5.54. È importante notare che la 5.54 non è sufficiente per realizzare la stabilità numerica, specialmente in condizioni di precisione limitata o in caso di vettore di ingresso molto lungo[4]. Il problema sorge dagli effetti di un errore accumulato dalla 5.54. Una soluzione al problema consiste nel monitorare γ_k, quando il suo valore esce dal range $[0 - 1]$ si ricalcola il suo valore usando la relazione $\gamma_k = 1 - D_k^T D_k$, che non essendo ricorsiva non introduce il problema dell'instabilità del lungo periodo.

[4] In questo caso ci sono problemi di stabilità a lungo periodo[5]

L'errore accumulato in γ_k diventa rilevante dopo un certo numero di cicli. Un altro approccio consiste nel ricalcolare $\gamma_k = 1 - D_k^T D_k$ ad intervalli regolari, per esempio ogni L campioni. Le equazioni 5.48 e 5.54 descrivo la parte di predizione dell'algoritmo. Si descrive di seguito la parte di filtraggio:

$$e_k(k) = d(k) - H(k)^T X(k) \tag{5.76}$$

È facile dimostrare che:

$$H(k) = H(k-1) + S_k D_k e_{k-1}(k) \tag{5.77}$$

e

$$e_{k-1}(k) = d(k) - H(k-1)^T X(k) = d(k) - D_k^T S_k^{-1} H(k-1) \tag{5.78}$$

Definendo:

$$Y_k = S_k^{-1} H(k) \tag{5.79}$$

quindi usando la 5.58 e la 5.21 dalla 5.78 si ottiene la 5.56. Usando la 5.79 e la 5.77 si ottiene in fine la 5.57.

5.4.4 Algoritmi di fattorizzazione efficienti

Come accennato precedentemente si vuole ottenere un algoritmo di fattorizzazione efficiente per il calcolo della seguente fattorizzazione:

$$\Pi_n = I + cD_n D_n^T = \Gamma_n \Gamma_n^T \tag{5.80}$$

dove c è una costante positiva, D_n è un vettore n-dimensionale[5], Γ_n è la radice quadrata della matrice Π_n, che consiste in una matrice quadrata, triangolare superiore. Come si nota in 5.48, 5.53 e in 5.56, piuttosto che calcolare Γ, si devono calcolare $\Gamma^T X$ e ΓX, dove X è un vettore noto. Si consideri la forma quadratica $X^T \Pi X$. Può essere espressa nel seguente modo:

$$X_n^T \Gamma \Gamma^T X_n = X_n^T (I_n + c_n D_n D_n^T) X_n = y_n^2 + [X_{n-1}^T (I_{n-1} + c_{n-1} D_{n-1} D_{n-1}^T) X_{n-1}] \tag{5.81}$$

dove I_n è la matrice identica $n \times n$, X_n e D_n sono vettori n-dimensionali:

$$X_n^T = [x_1 \quad x_2 \quad ... \quad x_n]$$

$$D_n^T = [d_1 \quad d_2 \quad ... \quad d_n]$$

[5]Per evitare confusione, si tenga presente che in questa sezione il pedice nei vettori o nelle matrici indica la dimensione dei medesimi.

e $c_n = c$, $c_{n-1} = c_n/e_n$, dove $e_n = 1 + c_n d_n^2$. Oltretutto

$$y_n = e_n^{-1/2} \left[x_n + c_n d_n \sum_{i=1}^{n} x_i d_i \right] \tag{5.82}$$

Quindi applicando ricorsivamente la 5.81 si ottiene:

$$X_n^T \Gamma \Gamma^T X_n = y_n^2 + y_{n-1}^2 + y_{n-2}^2 + ... + y_1^2 \tag{5.83}$$

dove gli y_i sono dati da 5.82. Se si definisce:

$$\upsilon = \Gamma^T X = [v1 \quad v2 \quad ... \quad vn] \tag{5.84}$$

dalla 5.81 si ha $v_i = y_i$. Il prodotto ΓX può essere calcolato dalla 5.82 e la matrice Γ è data dalla somma della matrice triangolare superiore T_n e dalla matrice diagonale con elementi diagonali

$$(e_1^{-1/2} \quad e_2^{-1/2} \quad ... \quad e_n^{-1/2})$$

L'elemento generico della T_n può essere calcolato da:

$$T(i,j) = d_i d_j c_j e_j^{-1/2} \tag{5.85}$$

Quindi il prodotto $\mu = \Gamma X = [u_1 \; u_2 \; ... \; u_n]$ è dato da:

$$u_i = e_i^{-1/2} x_i + d_i \sum_{k=i}^{n} c_k d_k e_k^{-1/2} x_k \tag{5.86}$$

Quanto descritto sopra può essere riassunto nei seguenti due algoritmi.

Algoritmo 1

Si consideri la seguente fattorizzazione square-root:

$$\Gamma \Gamma^T = I + cDD^T$$

dove Γ è triangolare superiore, c è una costante positiva e D è un vettore dato di dimensioni N. Il suo generico elemento è d_i. Anche X è un vettore N-dimensionale, il cui elemento generico è x_i. Un algoritmo efficiente per $\upsilon = \Gamma^T X$ è il seguente:

```
c(n) = c; z(0) = 0;

for i = 1 to n do
    z(i) = z(i-1);
od;

for i=n down to 1 do
    e(i) = 1 + c(i) * d(i) * d(i);
    v(i) = 1/sqrt(e(i)) * [x(i) + c(i) * d(i) * z(i)];
    c(i-1) = c(i) / e(i);
od;
```

Algoritmo 2

Nelle stesse ipotesi precedenti si calcola $\mu = \Gamma X$ col seguente algoritmo:

```
z(n+1)=0; c(n)=c;

for j = n down to 1 do
    e(j) = 1 + c(j) * d(j) * d(j);
    z(j) = c(j) * d(j) * x(j) * (1/sqrt(e(j))) + z(j+1);
    c(j-1) = c(j)/e(j);
    u(j) = x(j) * (1/sqrt(e(j))) + d(j) * z(j);
od;
```

Al fine del calcolo della matrice T è necessario aggiungere ai precedenti algoritmi un ciclo in grado di calcolare i coefficienti di T. Si sa che $T(i,j) = d_i d_j c_j e_j^{-1/2}$, quindi nell'algoritmo 2 si inserirà all'interno del ciclo principale il seguente:

```
for i = j down to 1
    T(i,j) = d(i)d(j)c(j)(1/sqrt(e(j)))
od;
```

Si vede che vengono calcolati i termini superiori della matrice, questo perche T è una matrice triangolare superiore.

5.5 Condizioni iniziali

5.5.1 Algoritmo RLS

L'algoritmo RLS per essere utilizzabile richiede che tutti i parametri calcolati ricorsivamente vengano inizializzati al valore più opportuno. In particolare di fondamentale importanza è il valore iniziale della matrice $K(0)$, la quale non deve portare alla condizione di non singolarità la matrice di correlazione $R_{XX}(k)$. Questo si può verificare calcolando l'inversa

$$\left[\sum_{i=-k_0}^{0} \lambda^{-i} X(i) X^T(i)\right]^{-1} \tag{5.87}$$

dove il vettore $X(i)$ è ottenuto dal blocco di dati compreso in $-k_0 \leq i \leq 0$.

La procedura più semplice è di calcolare:

$$R_{XX}(k) = \sum_{i=-k_0}^{0} \lambda^{-i} X(i) X^T(i) + \delta \lambda^k I \tag{5.88}$$

dove I è la matrice identica , δ è una piccola costante positiva. Ponendo $n = 0$ si ottiene:

$$R_{XX}(0) = \delta I$$

e quindi sapendo che $R_{XX}^{-1}(k) = K(k)$ si ottiene:

$$K(0) = \delta^{-1} I$$

Il vettore dei coefficienti del filtro viene usualmente inizializzato:

$$H(0) = [0\ 0\ ...\ 0]$$

corrispondente al vettore nullo.

L'inizializzazione presentata è relativamente semplice. L'unico parametro da scegliere è δ. Si raccomanda[4] di scegliere

$$\delta \sim 0.01 \sigma_X^2$$

dove σ_X^2 è la varianza dei dati di ingresso.

5.5.2 Algoritmo RLS SQR

L'inizializzazione dell'algoritmo RLS SQR[12] viene effettuata in genere come segue:

- $H(0) = 0$
- $S(0) = mI$
- m è un valore molto grande

Considerando che gli algoritmi tendono (in modo diverso) agli stessi risultati, si può creare un collegamento tra le condizioni iniziali dei vari algoritmi. Si è visto che

$$K(0) = \frac{100}{\sigma_X^2} I$$

sapendo che $K(n) = S(k)S^T(k)$ si può porre come condizione iniziale il valore:

$$S(0) = \sqrt{\frac{100}{\sigma_X^2}} I$$

5.5.3 Algoritmo fast RLS SQR

L'algoritmo va inizializzato come segue:

- $\gamma = 1$;
- Z e D vanno inizializzati a zero;
- α può essere inizializzata con una piccola costante positiva;
- $\sigma = 0$.

I parametri riportati sono stati tratti da [5]. Tuttavia per l'algoritmo di fusione sensoriale è fondamentale che Z e D non siano inizializzati a zero, altrimenti nel primo passo la matrice T sarebbe la matrice nulla. Considerando che l'aggiornamento della matrice S viene compiuto nel modo:

$$S_k = \lambda^{-1} S_{k-1} T_k$$

renderemmo S matrice nulla, senza possibilità di aggiornamento. Per questo i vettori D e S sono inizializzati con valori molto piccoli. La matrice S è inizializzata alla matrice identica.

5.6 Analisi della convergenza di RLS

Si assume che la risposta desiderata $d(k)$ e il vettore degli ingressi sono in relazione attraverso un modello di regressione lineare multipla:

$$d(k) = e_0(k) + H_0^T X(k) \tag{5.89}$$

dove il vettore $H_0 = H(0)$ contiene i parametri di regressione lineare del modello, e $e_o(k)$ è l'errore di misurazione. Esso è un processo di rumore bianco a media nulla e varianza σ^2. Il vettore H_0 è costante. L'ultima ipotesi è $\lambda = 1$, condizione che corrisponde a considerare il filtro come un ambiente stazionario.

5.6.1 Convergenza dell'algoritmo RLS

Si consideri il vettore di ingresso iniziale $X(0) = 0$, al quale corrisponde $R_{XX}(0) = 0$. Il vettore dei coefficienti del filtro $H(k)$ calcolati dall'algoritmo RLS sono:

$$H(k) = R_{XX}^{-1}(k) R_{Xd}(k) \tag{5.90}$$

dove per $\lambda = 1$:

$$R_{XX}(k) = \sum_{n=0}^{k} X(n) X^T(n) \tag{5.91}$$

$$R_{Xd}(k) = \sum_{n=0}^{k} X(n) d(n) \tag{5.92}$$

Sostituendo la 5.89 nella 5.92 si ottiene:

$$\begin{array}{rl} R_{Xd}(k) = & \sum_{n=0}^{k} X(n) X^T(n) H_0 + \sum_{n=0}^{k} X(n) e_0^n \\ = & R_{XX}(k) H_0 + \sum_{n=0}^{k} X(n) e_0^*(n) \end{array} \tag{5.93}$$

Questo significa che si può riscrivere l'equazione 5.90 come:

$$\begin{array}{rl} H(k) = & R_{XX}^{-1}(k) R_{XX}(k) H_0 + R_{XX}^{-1}(k) \sum_{n=0}^{k} X(n) e_0^*(n) \\ = & H_0 + R_{XX}^{-1}(k) \sum_{n=0}^{k} X(n) e_0^*(n) \end{array} \tag{5.94}$$

Invocando la proprietà delle variabili aleatorie:

$$E[x] = E[E[x|y]] \tag{5.95}$$

dove $E[x|y]$ è la speranza condizionata della variabile aleatoria x, data la variabile y. Alla luce di questa proprietà si usa la 5.95 per esprimere la speranza matematica di $H(k)$ come segue:

$$E[H(k)] = H_0 + E\left[R_{xx}^{-1}(k)\sum_{n=0}^{k} X(n)e_0^*(n)\right] \tag{5.96}$$

$$= H_0 + E\left[E\left[R_{xx}^{-1}(k)\sum_{n=0}^{k} X(n)e_0^*(n)|X(n), n = 0, 1, 2, ..., k\right]\right]$$

Considerando che:

- La matrice di correlazione $R_{XX}(k)$ è unicamente definita dalla sequenza dei vettori d'ingresso $X(0), X(1), ..., X(k)$;
- L'errore di misurazione $e_0(k)$ è indipendente dal vettore di ingresso $X(k)$ per ogni k;
- L'errore di misurazione $e_0(k)$ ha media zero;

si può ridurre la 5.96 in:

$$E[H(k)] = H_0 \quad k \geq M \tag{5.97}$$

L'equazione 5.97 dimostra che l'algoritmo RLS converge al valore medio con $k \geq M$, dove M è il numero dei coefficienti del filtro.

5.6.2 Rendimento dell'algoritmo RLS

Nell'algoritmo RLS si devono considerare due tipi di errore: l'errore di stima a priori $\xi(k)$ e l'errore di stima a posteriori $e(k)$.

Date le condizioni iniziali si verifica,al passo k, che i momenti di ordine 2 delle variabili errore sono molto diversi fra loro. Al passo $k = 1$ la media del quadrato di $\xi(k)$ raggiunge un valore molto grande, uguale al valore della media del quadrato di $d(k)$, e decresce al crescere di k. Il momento di secondo ordine di $e(k)$, tende dall'altra parte ad un valore molto piccolo, aumentando al crescere di k. L'errore su $\xi(k)$ produce la curva di rendimento dell'algoritmo RLS. Si può esprimere [4] l'errore a priori nella forma:

$$\xi(k) = e_0(k) - [H(k-1) - H_0]^T X(k) \tag{5.98}$$

$$= e_0(k) - \epsilon^T(k-1)X(k)$$

dove $\epsilon^T(k-1)$ è il vettore dell'errore sui coefficienti al passo $n-1$. Come indice di rendimento statistico, nella valutazione di RLS, è conveniente utilizzare la stima a priori dell'errore $\xi(k)$ per definire la media del quadrato:

$$J'(k) = E\left[|\xi(k)|^2\right] \tag{5.99}$$

dove il segno di primo in $J'(k)$ è per distinguere la media del quadrato di $\xi(k)$ dalla media del quadrato di $e(k)$. Sostituendo la 5.98 nella 5.99 ed espandendone i termini si ottiene:

$$\begin{aligned} J'(k) = E[|e_0(k)|^2] + E[X^T(k)\epsilon(k-1)\epsilon^T(k-1)X^T] \\ -E[\epsilon(k-1)X(k)e_0^*(k)] - E[e_0X^T(k)\epsilon(k-1)] \end{aligned} \tag{5.100}$$

dove $e_0(k)$ ha media zero. La prima media nella parte destra dell'equazione 5.100 è semplicemente la varianza di $e_0(k)$. Per le rimanenti tre medie si considerino le seguenti osservazioni:

1. la stima $H(k-1)$ e quindi il vettore errore $\epsilon(k-1)$ sono indipendenti dal vettore d'ingresso $X(k)$, quest'ultimo è un processo stazionario a media zero. Quindi sfruttando l'indipendenza statistica e l'algebra matriciale si può esprimere la seconda media dell'equazione 5.100 nel seguente modo:

$$\begin{aligned} & E\left[X^T(k)\epsilon(k-1)\epsilon^T(k-1)X(k)\right] = \\ = \; & E\left[tr\left\{X^T(k)\epsilon(k-1)\epsilon^T(k-1)X(k)\right\}\right] \\ = \; & E\left[tr\left\{X(k)X^T(k)\epsilon(k-1)\epsilon^T(k-1)\right\}\right] \\ = \; & tr\left\{E\left[X(k)X^T(k)\epsilon(k-1)\epsilon^T(k-1)\right]\right\} \\ = \; & tr\left\{E\left[X(k)X^T(k)\right]E[\epsilon(k-1)\epsilon^T(k-1)]\right\} \end{aligned} \tag{5.101}$$

2. L'errore di misurazione $e_0(k)$ dipende dal vettore di ingresso $X(k)$, da ciò segue un semplice arrangiamento del modello di regressione lineare. Il vettore d'errore $\epsilon(k-1)$ è perciò indipendente da $X(k)$ e $e_0(k)$. In accordo a quanto evidenziato si ricava che la terza media nella parte destra dell'equazione 5.100 è zero. È visibile riformulandola nel seguente modo:

$$E[\epsilon^T(k-1)X(k)e_0^*(k)] = E[\epsilon^T(k-1)]E[X(k)e_0^*(k)] \tag{5.102}$$

Ricorrendo inoltre al principio di ortogonalità, dove ogni elemento del vettore $X(k)$ è ortogonale all'errore di misurazione $e_0(k)$, si ricava:

$$E[\epsilon^T(k-1)X(k)e_0^*(k)] = 0 \tag{5.103}$$

3. La quarta e ultima media dell'equazione 5.100 ha la stessa forma della media considerata al punto 2. Anche questa media è uguale a zero:

$$E[e_0 X^T(k)\epsilon(k-1)] = 0 \tag{5.104}$$

Quindi sapendo che $E[|e_0(k)|] = \sigma^2$, e usando i risultati tratti ai punti 1-3 nella 5.100 si ottiene:

$$J'(k) = \sigma^2 + tr\left\{E\left[X(k)X^T(k)\right]E[\epsilon(k-1)\epsilon^T(k-1)\right]\right\} \tag{5.105}$$

Sapendo che [4] :

$$tr\left\{E\left[X(k)X^T(k)\right]E[\epsilon(k-1)\epsilon^T(k-1)\right]\right\} = \frac{M\sigma^2}{n-M-1}$$

con

$$n > M+1.$$

Sostituendo la precedente nella 5.105 (per $\lambda = 1$) si ottiene:

$$J'(k) = \sigma^2 \frac{M\sigma^2}{n-M-1}, + \quad n > M+1 \tag{5.106}$$

Basandosi sui risultati riportati si possono trarre le seguenti conclusioni:

1. Dalla curva di rendimento dell'algoritmo RLS, si deduce che l'algoritmo converge in circa $2M$ iterazioni, dove M è il numero di coefficienti del filtro.

2. Per un numero di iterazioni *prossimo all'infinito*, la media del quadrato dell'errore $J'(k)$ tende alla varianza σ^2 dell'errore di misurazione $e_0(n)$. In altre parole l'algoritmo RLS produce un errore quadratico medio maggiore di zero.

3. La convergenza di RLS è indipendente dagli autovalori presenti mediamente nella matrice di correlazione dell'ingresso.

Capitolo 6

Equivalenza tra i due algoritmi

In questo capitolo si tratterà l'equivalenza tra i filtri di Kalman e i filtri RLS. Essenziamente ci si basa sulla trattazione effettuata da Ali H. Sayed e Thomas Kailath [3].

6.1 Caso speciale del modello del sistema di stato

Un caso speciale del modello del sistema di stato gioca un ruolo fondamentale nella derivazione di una corrispondenza fra i filtri di Kalman e i filtri RLS.
Si consideri il seguente modello di sistema di stato (eq.6.1):

$$\begin{cases} X(k+1) = & \lambda^{-1/2} X(k) \\ z(k) = & U^T(k) X(k) + \nu(k) \end{cases} \tag{6.1}$$

dove $X(k)$ è il vettore di stato, $z(k)$ è l'uscita, $U(k)$ è la matrice di misurazione, $\nu(k)$ è un processo di rumore bianco di media nulla e varianza unitaria. Il parametro λ è una costante reale positiva.

6.2 Comparazione tra il modello stocastico e quello deterministico

Dal modello di stato si ricava facilmente che:

$$X(k) = \lambda^{-k/2} X(0) \tag{6.2}$$

dove X(0) rappresenta la condizione iniziale del sistema.
Sfruttando l'equazione d'uscita della 6.1 e l'equazione 6.2 si ricava il seguente sistema stocastico di equazioni lineari simultanee:

$$\begin{aligned} z(0) &= U^T(0)X(0) + \nu(0) \\ z(1) &= \lambda^{-1/2} U^T(1)X(0) + \nu(1) \\ &\vdots \\ z(k) &= \lambda^{-k/2} U^T(k)X(0) + \nu(k) \end{aligned} \tag{6.3}$$

Allo stesso modo le eq.6.3 possono essere scritte:

$$\begin{aligned} z(0) &= U^T(0)X(0) + \nu(0) \\ \lambda^{1/2} z(1) &= U^T(1)X(0) + \lambda^{1/2}\nu(1) \\ &\vdots \\ \lambda^{k/2} z(k) &= U^T(k)X(0) + \lambda^{k/2}\nu k \end{aligned} \tag{6.4}$$

Il sistema di equazioni 6.4 rappresenta una caratterizzazione stocastica del modello dinamico non forzato, pertinente dei filtri di Kalman.

Si consideri ora una formulazione deterministica del problema, affine al punto di vista RLS. In accordo con il modello di regressione lineare [4], si riporta il seguente sistema deterministico di equazioni lineari:

$$\begin{aligned} d^*(0) &= U^T(0)H(0) + e_0^*(0) \\ d^*(1) &= U^T(1)H(0) + e_0^*(1) \\ &\vdots \\ d^*(k) &= U^T(k)H(0) + e_0^*(k) \end{aligned} \tag{6.5}$$

dove $H(0)$ è la condizione iniziale del vettore dei coefficienti del filtro RLS, $U(k)$ è il vettore di ingresso, $d(k)$ è la risposta desiderata e $e_0(k)$ è l'errore di misurazione.

I due sistemi di equazioni lineari simultanee 6.4 e 6.5 risolvono essenzialmente lo stesso problema. Uno dei sistemi è stocastico, centrato sulla teoria dei filtri di Kalman, l'altro è deterministico, centrato sulla teoria dei filtri RLS.
Ci si aspetta, intuitivamente, che entrambi i sistemi conducano alla stessa soluzione del problema considerato. Considerando che i due sistemi di equazioni hanno la medesima formulazione è ragionevole porre:

$$X(0) = H(0) \tag{6.6}$$

Inoltre confrontando i due sistemi di equazioni 6.4 e 6.5 si ricava immediatamente la seguente corrispondenza uno-a-uno:

$$\begin{aligned} z(k) &= \lambda^{-k/2} d^*(k) \\ \nu(k) &= \lambda^{-k/2} e_0^*(k) \end{aligned}$$

dove l'asterisco indica la complessa coniugata, le variabili di sinistra si riferiscono al modello di stato, le variabili di destra si riferiscono al modello di regressione lineare.

6.3 Comparazione tra i filtri di Kalman basati sulla covarianza e l'algoritmo RLS

La corrispondenza tra RLS e Kalman finora trattata può essere espansa ai restanti termini dei due algoritmi. Comparando le equazioni, dei due algoritmi, per simmetria si ricava:

$$P(k-1) = \lambda^{-1} K(k-1) \tag{6.7}$$

$$g(k) = \lambda^{-1/2} C(k) \tag{6.8}$$

$$\alpha(k) = \lambda^{-k/2} \xi(k) \tag{6.9}$$

$$\widehat{X}(k+1|z_k) = \lambda^{-(k+1)/2} \widehat{H}(k) \tag{6.10}$$

Inoltre comparando il fattore di conversione dell'algoritmo di Kalman e il fattore di conversione dell'algoritmo RLS si verifica che entrambi rappresentano la medesima quantità:

$$r^{-1}(k) = \gamma(k) \tag{6.11}$$

6.3.1 Calcolo dell'ingresso di RLS

Ali H. Sayed e Thomas Kailath [3] nella loro trattazione trascurano di indicare a cosa corrisponde nel filtro di Kalman l'ingresso RLS. Sfruttando le corrispondenze da loro fornite si vede che:

$$z(k) = \lambda^{-\frac{k}{2}} d(k)$$

$$X(k+1|z_k) = \lambda^{-\frac{k+1}{2}} H(k) = \lambda^{-\frac{1}{2}} X(k)$$

$$\begin{aligned} d(k) &= u(k)H(k) \\ z(k) &= h(k)X(k) \end{aligned}$$

Sostituendo a quest'ultima relazione l'uscita e lo stato:

$$\lambda^{-\frac{k}{2}} d(k) = h(k)\lambda^{-\frac{k}{2}} H(k)$$

Si ottiene che:

$$\begin{aligned} d(k) = h(k)H(k) &\Longleftrightarrow d(k) = u(k)H(k) \\ &\Rightarrow h(k) = u(k) \end{aligned} \tag{6.12}$$

6.4 Sommario della corrispondenza fra i due algoritmi

Si riportano nella tabella 6.4 tutte le relazioni di equivalenza tra i due algoritmi.

Descrizione	Variabile	Variabile	Descrizione
Vettore di stato	$X(k)$	$\lambda^{-k/2}H(0)$	Versione esponenziale della regressione del vettore dei coefficienti
Segnale di osservazione	$z(k)$	$\lambda^{-k/2}d(k)$	Risposta desiderata
Rumore di misurazione	$\nu(k)$	$\lambda^{-k/2}e_0(k)$	Errore di misurazione
Predizione dello stato	$X(k+1\|z_k)$	$\lambda^{-(k+1)/2}H(k)$	Stima dei coefficienti del filtro
Matrice di correlazione dello stato	$P(k)$	$\lambda^{-1}K(k)$	Inversa della matrice di correlazione dell'ingresso
Kalman gain	$g(k)$	$\lambda^{-1/2}C(k)$	Guadagno di Kalman
Innovazione	$\alpha(k)$	$\lambda^{-k/2}\xi(k)$	Stima a priori dell'errore
Fattore di conversione	$r^{-1}(k)$	$\gamma(k)$	Fattore di conversione
Condizioni iniziali	$X(1\|z_0)=0$	$H(0)=0$	Condizioni iniziali
	$P(0)$	$\lambda^{-1}K(0)$	
	$X(0)$	$H(0)$	

Tabella 6.1: Sommario equivalenza tra Kalman e RLS

Capitolo 7

L'ambiente di simulazione

In questo capitolo si descriverà l'ambiente utilizzato per acquisire i dati da elaborare con gli algoritmi di fusione e per rappresentare i risultati ottenuti.

Il simulatore utilizzato si basa su ERASMUS (Expert-based Robot Architecture for Sensing and Moving in Unknown Surrounding), un'architettura di controllo per robot mobili. L'ERASMUS è stato sviluppato grazie ad una collaborazione tra l'Università di Genova e l'Università di Trieste, al fine di studiare i robot mobili in ambienti civili e/o industriali. L'obiettivo dei programmatori era di creare un sistema in grado di simulare il movimento di un robot in navigazione autonoma, in un ambiente non noto a priori, in presenza d'ostacoli, dandogli la capacità di cooperare con altri robot o con un elaboratore centrale detto supervisore. Il ruolo del supervisore è di pianificare compiti complessi e di aiutare ogni singolo robot nella risoluzione di problemi locali, come uscire da un punto morto della navigazione, o stimare la propria posizione. Nello stesso tempo, l'architettura prevede che ogni robot sia indipendente in caso di blackout delle comunicazioni.

7.1 Il simulatore

L'architettura del simulatore (fig. 7.1) si divide principalmente in due parti:

L'interfaccia utente: offre all'operatore i comandi per interagire con il nucleo del simulatore, quindi permette di avviare e terminare la simulazione, comandare il robot durante la marcia, visualizzare i risultati della simu-

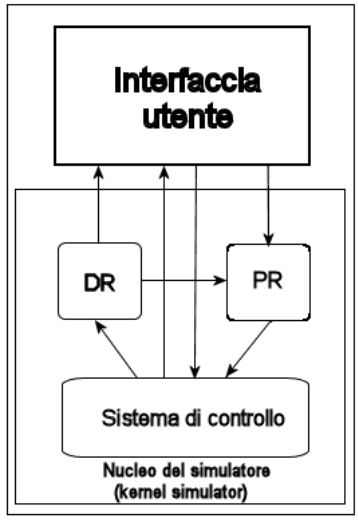

Figura 7.1: Schema a blocchi dell'architettura del simulatore

lazione. Contiene inoltre la rappresentazione (sotto forma di mappa di pixel) dell'ambiente che circonda il robot (fig. 7.2).

Il nucleo del simulatore: Il nucleo del simulatore si occupa della gestione del robot: simulazione sensori, navigazione, eccetera. È diviso principalmente in tre parti:

- DR che simula la dinamica del robot;
- PR che simula le percezioni del robot;
- Il sistema di controllo che si basa sull'ambiente di sviluppo ETHNOS,

PR, DR ed il sistema di controllo agiscono secondo uno schema ad anello. I sensori simulati da PR eseguono una scansione dell'ambiente simulato (i dati vengono forniti dall'interfaccia utente), le informazioni ottenute vengono passate al sistema di controllo, che a sua volta invia i comandi per lo spostamento del robot a DR. Quest'ultimo simula lo spostamento e trasmette i dati all'interfaccia utente, che rappresenterà il robot nella mappa. In base a questa nuova posizione vengono acquisite nuove rilevazioni sensoriali, ed il ciclo si ripete. L'interfaccia

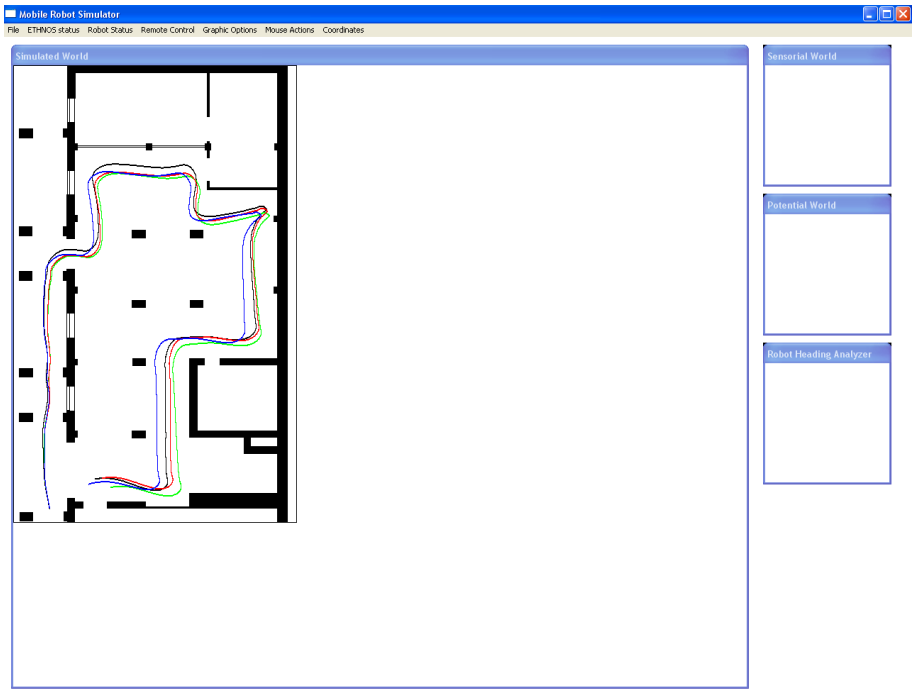

Figura 7.2: Schermata principale del simulatore

utente è collegata direttamente con il sistema di controllo, questo permette in un senso di inviare comandi per la navigazione direttamente al sistema di controllo, e nell'altro senso di acquisire i valori dei sensori per ricostruire la mappa ambientale.

7.2 DR, dinamica del robot

Questo blocco serve a simulare il movimento del robot e i rilevamenti dei sensori odometrici. In ingresso al sistema si hanno i comandi di velocità traslazionale (speed) e rotazionale (jog) forniti dalla routine, che nel sistema di controllo teoricamente comanda i motori. Ad intervalli di tempo prefissati (il perio-

do di campionamento è di 1/10 s) si calcolano le nuove coordinate del robot $(x(k), y(k), \theta(k))$ secondo le relazioni odometriche.

7.3 PR, percezioni del robot

PR fornisce le letture simulate dei sensori ad ultrasuoni. La posizione e la direzione del robot è ricavabile dal modulo DR, in questo modo PR individua la giusta parte di mondo simulato da analizzare. Il robot simulato possiede 14 sensori ad ultrasuoni disposti nella parte antistante. La portata dei sensori è di 2 m, l'angolo di dispersione del sensore è di $\frac{4}{9}\pi$. Il simulatore scanna il cono d'apertura del sensore, se in questa zona individua un ostacolo (quindi nella mappa trova dei pixel di colore nero), rende la minima distanza dal sensore all'ostacolo.

7.4 L'ambiente di sviluppo ETHNOS

ETHONS (Expert Tribe in a Hybrid Network Operating System) è un sistema basato sulle specifiche real time POSIX e fornisce le primitive per la creazione di thread (esperti), per la loro schedulazione e la loro interoperabilità, fornendo un prefissato protocollo di comunicazione tra i processi. È un ambiente di sviluppo pensato per rispondere alle seguenti esigenze:

- semplificare lo sviluppo in parallelo di diversi progetti fornendo un prefissato protocollo di comunicazione;
- facilitare lo sviluppo di sistemi real-time gestendo la schedulazione dei processi;
- distribuire il calcolo computazionale su diversi calcolatori senza dover gestire direttamente la comunicazione.

Il sistema è stato sviluppato interamente in ANSI C++ in ambiente Linux, utilizzando primitive per la gestione dei processi, in particolare quelle relative ai trhead.

L'ambiente di sviluppo è composto da due unità fondamentali: gli *esperti* ed il *kernel*.
Un *esperto* è una porzione di codice preposta ad un compito specifico, eseguita ripetutamente.
Il *kernel* è un programma che si interfaccia con il sistema operativo e ha il

compito di gestire le comunicazioni e la schedulazione dei processi. Una volta effettuata l'analisi di schedulabilità il sistema passa in esecuzione, in particolare viene dato il controllo ai singoli *esperti*, i quali eseguono in modo sequenziale il codice specificato nel proprio corpo.

Per quanto riguarda la gestione delle comunicazioni, il *kernel* si occupa di gestire in modo affidabile e trasparente lo scambio di messaggi tra gli esperti, effettuando un broadcast dell'informazione. L'ambiente di sviluppo ETHNOS consente di realizzare un sistema distribuito, semplicemente stanziando più *kernel* su differenti calcolatori.

7.4.1 Il kernel

Il *kernel* è un oggetto derivato dalla classe `ETDispacth`, che fornisce le primitive necessarie per la gestione degli esperti, cioè le funzioni per inserire gli esperti nel sistema e per l'esecuzione degli stessi. La funzione `DoYourDuty` del *kernel* una volta chiamata consegna il controllo del sistema al kernel stesso, il quale esegue le seguenti operazioni:

1. effettua un'analisi approssimata del tempo di esecuzione di ogni esperto attraverso ripetute chiamate alla funzione `ETExpert::DoYourDuty`;
2. verifica la possibilità di schedulare l'insieme di esperti tramite l'algoritmo `Rate Monotomic`, in base al tempo di esecuzione e al periodo degli *esperti* periodici;
3. manda l'insieme di *esperti* in esecuzione, se le condizioni di schedulabilità sono verificate, altrimenti il controllo viene restituito al sistema operativo;
4. una volta in esecuzione il *kernel* si occupa anche di gestire in modo efficiente lo scambio di messaggi, rendendo trasparente agli *esperti* il fatto di trovarsi in un sistema distribuito o meno.

La comunicazione tra gli *esperti* è di tipo broadcast: ogni messaggio inviato al *kernel* viene reso accessibile a tutti gli *esperti* che ne hanno fatto richiesta, aggiungendolo alla loro lista locale. Per evitare inutili occupazioni di memoria, in realtà, ETHNOS utilizza una zona di memoria condivisa piuttosto che effettuare numerose copie dello stesso messaggio. Per questo gli *esperti* possono solo leggere i messaggi e non possono effettuare loro cambiamenti. Se un esperto ha necessità di apportare delle modifiche ad un messaggio, dovrà utilizzare una copia locale creata appositamente. La *lista globale* dei messaggi funge da memoria condivisa, quando il *kernel* riceve un messaggio da un *esperto* da inserire nella

lista, verifica se vi sono richieste per quel tipo di messaggio. In caso di successo il messaggio viene salvato, in caso contrario viene scartato.

Nella *lista globale* i messaggi sono memorizzati in modo non omogeneo, cioè senza tener conto dei tipi associati ad ogni messaggio.
Se il *kernel* riceve, ad esempio, messaggi di tipo A, verifica che ci siano richieste per quel tipo di messaggio e in caso affermativo lo inserisce nella *lista globale*. Successivamente si preoccupa di aggiornare le *liste locali* inserendo un puntatore che individui il messaggio nella lista globale. Gli *esperti* potranno accedere al messaggio per la solo lettura vedendolo come un elemento della propria *lista locale*. Attraverso un contatore si tiene traccia di quanti esperti stanno attualmente accedendo al messaggio; questo viene fatto in quanto verranno eliminati dalla lista globale i messaggi privi di interesse. Il *kernel* assegna ad ogni messaggio un contatore (*reference counter*) che viene inizializzato ad un valore pari al numero di esperti che utilizzano l'informazione. Ogni volta che un *esperto* accede ad un messaggio, il *kernel* decrementa il *reference counter*. Quando quest'ultimo assume valore zero il messaggio viene distrutto. Il *kernel* costruisce per ogni tipo di messaggio un oggetto di tipo *mailbox* contenente le seguenti informazioni:

- tipo del messaggio
- numero di richieste avanzate per tipo
- lista di puntatori agli *esperti* che hanno avanzato richiesta
- flag associato al tipo

L'informazione contenuta nel secondo punto viene utilizzata per inizializzare il *reference counter* di ogni nuovo messaggio aggiunto alla *lista globale*. La lista di puntatori viene scandita ogni volta che è necessario accedere alle *liste locali* per rendere disponibile i messaggi agli *esperti* interessati. Il *kernel* preleva il puntatore ad ogni *esperto*: `Pesperto`, quindi effettua una chiamata a `Pesperto::AddToLocalList`. Infine il flag associato al tipo viene utilizzato per il ciclo di sospensione/risveglio di ogni *esperto*.

7.4.2 Gli esperti

Il sistema considera esperto un qualunque oggetto appartenente a una classe derivata dalla classe astratta `ETExpert`. Quest'ultima contiene tre funzioni virtuali che devono essere definite dal programmatore nella classe derivata, in fase di creazione dell'esperto:

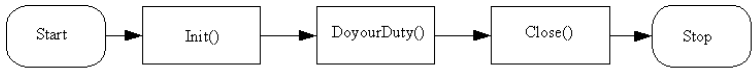

Figura 7.3: Schema a blocchi degli *esperti*

`Init` è la funzione che contiene la procedura di inizializzazione, che viene eseguita all'avvio dell'esecuzione dell'esperto;

`DoYourDuty` è la funzione che contiene il comportamento specifico dell'esperto, ovvero il codice che deve essere eseguito ad ogni attivazione dello stesso;

`Close` è la funzione che contiene la parte di codice eseguita una sola volta nella fase terminale dell'intervento dell'*esperto*.

Quindi ogni esperto è organizzato secondo lo schema a blocchi in figura 7.3. L'ambiente di sviluppo permette l'implementazione di tre categorie di *esperti*: periodici, sporadici e di background. L'appartenenza ad una o all'altra categoria è decisa nel momento in cui si dichiara la classe. Al costruttore della classe `ETExpert` vengono passati i valori opportuni per ogni categoria.

Le caratteristiche dei tre tipi di esperti sono:

- *esperti periodici*: eseguono il proprio compito secondo una scadenza temporale prefissata; nel costruttore della classe è necessario specificare il periodo di attivazione;
- *esperti sporadici*: eseguono il proprio compito in corrispondenza di un evento determinato. Nel costruttore della classe è necessario specificare il tempo minimo che deve intercorrere tra due successive attivazioni; inoltre all'interno della funzione `init()` bisogna specificare i tipi di messaggio alla cui ricezione si vuole subordinare l'esecuzione dell'esperto;
- *esperti di background*: eseguono il proprio compito nel tempo in cui il processore non esegue *esperti periodici* o *sporadici*. Nel costruttore si specifica che all'*esperto* è assegnata la priorità più bassa.

Una volta stanziato l'*esperto*, è necessario informare il *kernel* di inserirlo nel sistema. L'operazione è fatta tramite le funzioni `AddExpert` e `ActivateExpert`. A questo punto il *kernel* inserisce nella schedulazione anche il nuovo esperto e ne gestisce le comunicazioni.

Gli *esperti* comunicano tra loro tramite dei messaggi. È compito del programmatore utilizzare i messaggi per coordinare le attività degli *esperti*. Il principio di comunicazione è il seguente:

- ogni esperto che vuole ricevere un determinato tipo di messaggio ne fa richiesta al kernel;
- ogni esperto che decide di produrre e condividere un messaggio, lo passa al kernel senza preoccuparsi del destinatario.

Il kernel garantisce che ogni messaggio prodotto sia ricevuto da tutti gli esperti che ne facciano richiesta. Tutti i messaggi sono divisi in tipi, il protocollo di comunicazione è basato sui tipi di messaggio e sulle richieste del tipo di messaggio. Ogni *esperto* possiede una *lista locale*, dove accoda i messaggi recapitati dal *kernel*, per ogni tipo di messaggio richiesto. In ogni momento la *lista locale* contiene tutti i messaggi disponibili. Sarà l'*esperto* stesso che decide i tempi ed i modi per disporre dei vari messaggi disponibili.

Ogni messaggio è costituito da un header e da un campo riservato ai dati. Nell'header sono specificate la dimensione ed il tipo di messaggio. La classe `ETMessage` fornisce tutte le primitive per la creazione, la scrittura e la lettura dei messaggi.

Il protocollo descritto non sempre è adatto alla comunicazione tra *esperti*, infatti la sua natura burocraticamente complessa intralcia, quando è necessario, lo scambio di dati rapido ed immediato (per esempio: in comunicazioni tra esperti caratterizzate da un flusso continuo di dati a frequenza molto elevata). Esiste a tal fine una comunicazione di *tipo privilegiato*[1], che consente di scavalcare il protocollo di comunicazione.

A seconda del compito da svolgere esistono diverse classi di *esperti*:

esperti simbolici: lavorano accedendo ad una base comune di conoscenza per svolgere le loro attività come, per esempio, l'aggiornamento delle informazioni immagazzinate, il ragionamento simbolico o lo scambio di messaggi con altri esperti simbolici.

esperti diagrammatici: lavorano su rappresentazioni diagrammatiche[2] di cui possono esistere differenti istanze. Alcuni esperti hanno il compito di modificare tali rappresentazioni, altri le usano per ricavare informazione ed altri ancora possono fare entrambe le cose.

[1] La comunicazione privilegiata è eseguita tramite puntatori. Quindi si ha un accesso diretto ai dati.

[2] per esempio bitmap o diagrammi

esperti reattivi: interagiscono con la parte hardware del sistema come sensori ed attuatori. A causa della loro natura, richiedono maggior velocità di esecuzione e sono quindi soggetti a vincoli real-time restrittivi. Di conseguenza usano spesso chiamate di funzione dirette piuttosto che messaggi per comunicare tra di loro.

Lo scambio diretto di informazioni avviene solo tramite esperti della stessa classe, mentre la comunicazione tra esperti di diverse classi avviene tramite messaggi.

7.5 L'interfaccia utente

L'interfaccia utente offre la possibilità di interagire con l'ambiente di simulazione e di visualizzare i risultati prodotti. Come si può vedere in figura 7.4,

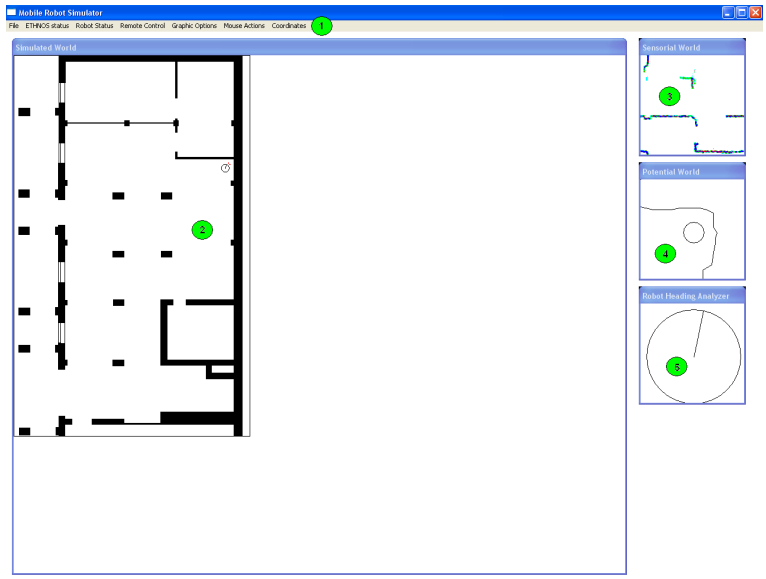

Figura 7.4: Interfaccia utente del simulatore. 1- la barra dei comandi, 2- visuale del mondo simulato, 3- visuale del mondo sensoriale, 4- visuale del mondo potenziale, 5- visuale dell'orientazione del robot

l'interfaccia utente è composta da:

1. la barra dei comandi;
2. la finestra del mondo simulato;
3. la finestra del mondo sensoriale;
4. la finestra del mondo potenziale;
5. la finestra che mostra la direzione del robot.

All'avvio del programma compare la finestra di **Simulator Window**. Egli è responsabile della creazione di tutti gli altri oggetti: PR, DR, kernel ed esperti di ERASMUS. Tuttavia non interagisce direttamente con il nucleo della simulazione, ma attraverso gli oggetti in figura 7.4.

Il **Simulator Window** gestisce la barra comandi per le interazioni con l'utente: avvio della simulazione, cambiamento del modo di navigazione, variazione della velocità del robot, etc.

La prima operazione da fare, per avviare la simulazione, è caricare il mondo simulato[3], che verrà immagazzinato in **Simulated Word**. Si prosegue poi avviando il *kernel* di ERASMUS, il quale viene gestito da un'altro thread in modo da rendere indipendente il tempo del simulatore dal tempo di sistema di controllo. Gli oggetti con cui **Simulator Window** interagisce sono:

Simulator Interface: gestisce l'interazione con il mondo simulato, in particolare è possibile specificare una destinazione per il robot ed aggiungere o rimuovere ostacoli[4]. Inoltre rappresenta il robot nella mappa.

Sensorial Interface: gestisce il mondo sensoriale, in pratica mostra graficamente la mappa ricostruita dai sensori del robot. La presenza di un ostacolo è indicata da diversi colori a seconda di quante volte l'ostacolo viene individuato. Le informazioni sono ricavate dalla struttura di controllo e in particolare dalla rappresentazione diagrammatica della mappa.

Potential Interface: gestisce il mondo potenziale, ovvero la rappresentazione grafica del fronte d'onda potenziale, ricavata sempre dalla struttura di controllo da un'altra rappresentazione diagrammatica. Questa volta del campo potenziale artificiale.

[3]Il mondo simulato consiste in un file in formato bitmap, con una profondità di colore di 8bit (256 livelli), rappresentante la mappa di un ambiente.

[4]viene modificata la mappa memorizzata

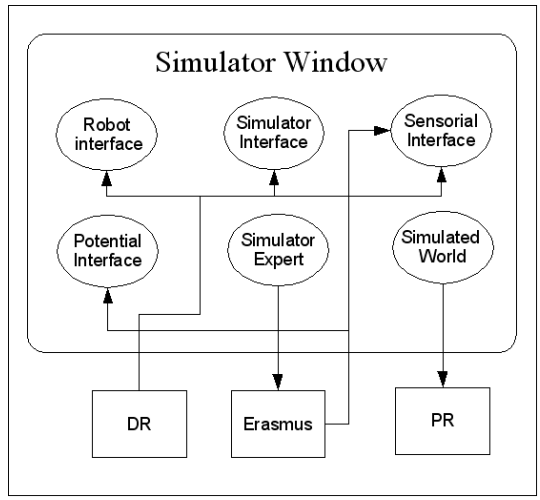

Figura 7.5: Schema a blocchi dell'interfaccia utente

Robot Interface: mostra semplicemente la direzione corrente del robot in modo più preciso della rappresentazione fornita sulla mappa.

Simulator Expert: è un esperto che ha il compito di tradurre i comandi, impartiti dall'utente, in messaggi da inviare ad ERASMUS. I comandi consistono essenzialmente in:

NEWSPEED: cambia la velocità massima del robot;

NEWTARGET: cambia la destinazione finale del robot;

STOPETHNOS: ferma il kernel, terminando così la simulazione.

Simulated World: possiede una copia della mappa usata dall'oggetto PR per simulare le percezioni sensoriali. La mappa è sostanzialmente memorizzata come una tabella contenente i valori dei pixel, in cui il valore zero[5] indica un ostacolo. La mappa è in scala 1:50 ovvero 1 pixel corrisponde a 50mm.

[5] Il valore zero corrisponde al colore nero.

7.6 ERASMUS

La versione di ERASMUS implementata nel simulatore non è uguale a quella presentata da Piaggio[16]. Sono stati implementati solo gli esperti che costitui-

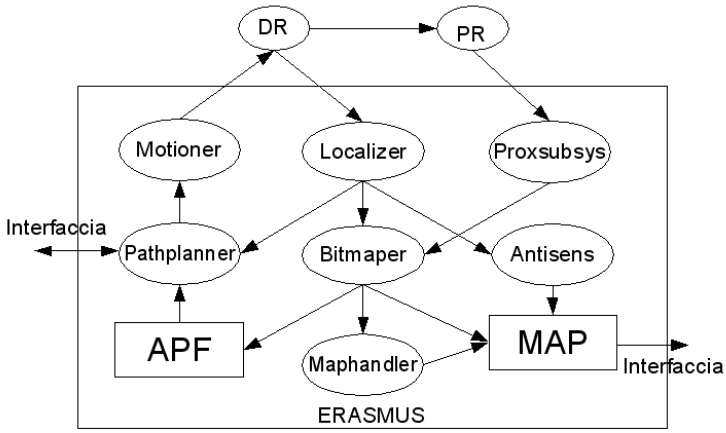

Figura 7.6: Schema a blocchi del simulatore ERASMUS

scono il minimo indispensabile per il corretto funzionamento del programma[6]. Ci sono tre esperti reattivi, detti anche esperti di basso livello:

- Motioner
- Localizer
- Proxsubsys

E quattro esperti diagrammatici, detti di alto livello, che gestiscono le rappresentazioni diagrammatiche APF e MAP (fig. 7.6).:

- Pathplanner
- Bitmaper

[6]Come anche nel robot reale

- Maphandler
- Antisens

Gli esperti di basso livello traducono i comandi forniti dal livello superiore in comandi per gli attuatori. Oppure trasformano le informazioni fornite dai sensori, o dagli oggetti che li simulano, in messaggi utilizzabili dagli esperti di alto livello. Nel caso del robot reale costituiscono l'interfaccia tra ERASMUS e l'hardware. Gli esperti di basso livello acquisiscono una notevole importanza in quanto, dal loro corretto funzionamento, dipendono gli altri esperti. Per questo la loro esecuzione è schedulata con frequenza maggiore[17].

Capitolo 8

SLAM - Simultaneous Localization And Mapping

In questo capitolo affrontiamo il problema dello SLAM: *Simultaneous Localization And Mapping*. L'approccio adottato per questa tecnica è di tipo *feature based*, con l'utilizzo del filtro di Kalman per valutare lo stato del robot e delle feature nella mappa. Si riportano di seguito le due trattazioni, quella relativa allo SLAM con il filtro di Kalman classico e quella relativa allo SLAM con la variante RLS.

8.1 EKF SLAM

Il filtro di Kalman è un algoritmo di data processing ricorsivo. Questa caratteristica permette che tutte le informazioni utili all'istante $k+1$ per il calcolo della stima non derivino da dati memorizzati nei precedenti k passi, ma semplicemente dalla stima precedente. Quindi l'algoritmo tiene *memoria intrinsecamente di ciò che è avvenuto nel passato.*

Il filtro integra tutte le informazioni disponibili acquisite dal sistema con la conoscenza dello stato iniziale del sistema, per stimare le grandezze incognite, in modo tale da minimizzare statisticamente l'errore, ovvero minimizzare l'indice di qualità [8]:

$$J[X(k|k-1)] = E\{[X(k) - X(k|k-1)]^T Q(k)[X(k) - X(k|k-1)]\} \quad (8.1)$$

dove $X(k|k-1)$ è la stima dello stato al passo k basata sui passi precedenti,

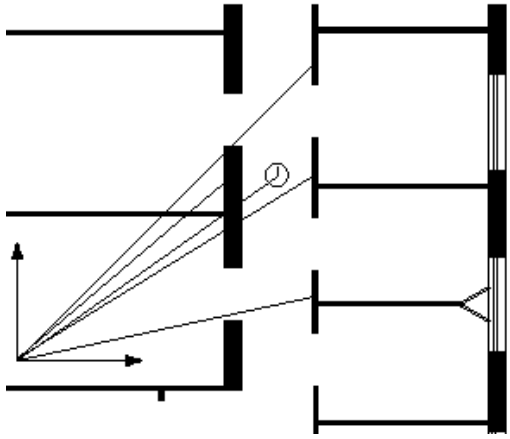

Figura 8.1: Il sistema considerato

$X(k)$ è lo stato reale al momento k e $Q(k)$ è una matrice peso definita positiva. In altre parole, se avessimo un insieme di algoritmi di filtraggio che stimano la stessa grandezza, sfruttando gli stessi dati in ingresso, il risultato fornito dal filtro di Kalman avrebbe un errore medio inferiore rispetto tutti gl'altri algoritmi in esame. Essenzialmente quindi si definisce che la stima prodotta dal filtro di Kalman per una data grandezza è ottima[1], utilizzando i dati in ingresso di un sistema affetto da errore.

8.1.1 Il modello di stato del sistema

Il modello di stato del sistema ambiente più veicolo descrive essenzialmente la condizione della figura 8.1. Il robot viaggia attraverso l'ambiente, i sensori di cui è dotato analizzano il mondo circostante. Lo stato del sistema all'istante k descrive la posizione del robot:

coordinata x Posizione lungo le ordinate del baricentro del veicolo

coordinata y Posizione lungo le ascisse del baricentro del veicolo

[1]Per stima ottima si intende che è stata ottenuta minimizzando gli errori di misura

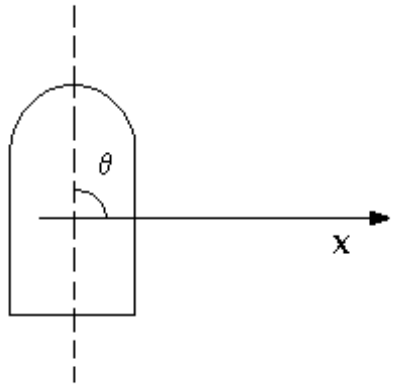

Figura 8.2: Orientazione robot

orientazione θ Angolo sotteso tra l'asse di simmetria del robot e l'asse delle ascisse (fig.8.2).

Inoltre descrive la posizione dei landmark individuati.

Il vettore $X(k)$ quindi conterrà nei primi n_ν stati le grandezze che caratterizzano il veicolo e nei successivi stati n_m le grandezze che descrivono le *feature* individuate nell'ambiente.

$$X(k) = [{}^G X_\nu(k), {}^G x_1(k), ..., {}^G x_m(k)]^T \tag{8.2}$$

dove ${}^G X_i(k)$ indica lo stato rispetto il sistema di riferimento $\Im_G$

Possiamo esprimere meglio il vettore di stato nel seguente modo:

$$X(k) = \begin{bmatrix} {}^G X_\nu(k) \\ {}^G X_m(k) \end{bmatrix} \tag{8.3}$$

8.1.2 I modelli del sistema

Il modello del processo descrive come gli stati del sistema evolvono nel tempo. Questi può essere descritto, nel tempo continuo, da un'equazione lineare del primo ordine.

$$\dot{x}(t) = f(x(t), u(t), t) + \nu(t) \tag{8.4}$$

dove $x(t) \in \Re^n$ è il vettore di stato al momento t, $u(t)$ è un ingresso noto al momento t, $f(...)$ è la funzione di cambiamento di stato, $\nu(t)$ è un vettore, di variabili aleatorie, che descrive sia il rumore causato in movimento, sia le incertezze del modello di stato.

Per l'elaborazione in un calcolatore il modello continuo deve essere discretizzato. La discretizzazione del modello avviene tramite l'operazione di campionamento:

$$x(t_k) = f(x(t_{k-1}), u(t_k), t_k) + \nu(t_k) \tag{8.5}$$

dove, come precedentemente, $x(t_k)$ è il vettore di stato all'istante t_k, $x(t_{k-1})$ è il vettore di stato all'istante precedente, ovvero t_{k-1}, $u(t_k)$ è l'ingresso e $\nu(t_k)$ è il rumore.
L'intervallo di tempo:

$$\Delta t(k) = t_k - t_{k-1} \tag{8.6}$$

è definito intervallo di campionamento. Nelle simulazioni che seguiranno questa trattazione, $\Delta t(k)$ sarà costante ad ogni k, in particolare varrà $0,1s$.

Applicando alla relazione 8.5 una notazione priva della variabile temporale, si esprimere di nuovo il modello nella forma discreta:

$$x(k) = f(x(k-1), u(k)) + \nu(k) \tag{8.7}$$

Il modello del veicolo

Il modello del veicolo descrive l'evoluzione dello stato $X_\nu(k)$ in funzione dello stato precedente $X_\nu(k-1)$ e dell'ingresso di controllo $u(k)$:

$$x_\nu(k) = f(x_\nu(k-1), u(k)) + \nu_\nu(k) \tag{8.8}$$

I modelli di navigazione, che descrivono l'andamento del robot in modo preciso, dipendono essenzialmente dall'attendibilità con cui descrivono lo spostamento in base alle rilevazioni dei sensori. Possono avere diversi gradi di complessità in funzione dei parametri considerati. Si riprenderà la trattazione sul modello che descrive il veicolo più avanti.

Il modello dei landmark

Nel contesto dello SLAM, i *landmark* sono delle *feature* nell'ambiente che possono essere osservate direttamente dai sensori in modo attendibile e consistente. In questa trattazione l'algoritmo di SLAM utilizza un approccio AMF *Absolute*

Map Filter, nel quale i *landmark* sono considerati nel sistema di riferimento globale.

Nell'algoritmo implementato si suppone che i landmark siano tempo invarianti, ovvero che la loro posizione sia stabile nel tempo.

$$X_m(k) = X_m(k-1) \tag{8.9}$$

dove $X_m(k)$ è lo stato dei landmark nel momento k.

Il modello del sensore

Il modello, che descrive nel sistema stato spazio il processo di osservazione, è ancora una funzione non lineare:

$$z(t) = h(x(t), u(t), t) + w(t) \tag{8.10}$$

dove la $z(t) \in \Re^m$ è l'osservazione fatta al momento t, il $h(...)$ è il modello di osservazione degli stati del sistema e $w(t)$ è una variabile aleatoria che descrive sia il rumore di misura, sia l'incertezza del modello.

Come per il caso precedente si discretizza la relazione 8.10 attraverso il processo di campionamento:

$$z(k) = h(x(k), u(k)) + w(k) \tag{8.11}$$

8.1.3 Il processo di stima dello stato

Sfruttando i modelli descritti, il processo di SLAM consiste nel determinare la stima ottima dello stato. Quest'operazione può essere fatta grazie al filtro di Kalman.
Il filtro di Kalman è uno stimatore ricorsivo strutturato in tre passi: previsione, osservazione e aggiornamento. Data la sequenza dei dati osservati in ingresso:

$$Z_k = \{z(0), z(1), ..., z(k)\}$$

fornisce la stima dello stato con minima varianza d'errore. Per semplicità di esposizione si adotta la seguente notazione:

$$\hat{x}^+(k) = \hat{x}(k|k) \tag{8.12}$$

dove $\hat{x}^+(k)$ rappresenta la stima a posteriori dello stato, ovvero la stima sfruttando le precedenti k osservazioni, rappresenta inoltre l'aggiornamento del sistema, ovvero la stima dello stato all'istante k sfruttando la k-esima osservazione;

$$\hat{x}^-(k) = \hat{x}(k|k-1) \tag{8.13}$$

dove $\hat{x}^-(k)$ è la stima a priori dello stato, ovvero la stima sfruttando le precedenti $k-1$ osservazioni, nonchè la previsione dello stato fatta al passo precedente, per il passo successivo.

L'algoritmo di mappatura e localizzazione simultanea, utilizza il filtro di Kalman per valutare la posizione del veicolo $\hat{X}^+_\nu(k)$ e dei *landmark* $\hat{X}^+_m(k)$. Il vettore di stato utilizzato per questo algoritmo è stato descritto precedentemente (eq. 8.3).

$$X(k) = \begin{bmatrix} X_\nu(k) \\ X_m(k) \end{bmatrix} \tag{8.14}$$

La matrice di correlazione associata allo stato è:

$$P^+(k) = E[(X(k) - \hat{X}^+(k))(X(k) - \hat{X}^+(k))^T | Z_k] \tag{8.15}$$

Questa matrice rappresenta la varianza dell'errore e la correlazione d'errore per ogni stima di stato.

Nel caso trattato dell'*Absolute Map Filter*, la matrice di covarianza si presenta nella seguente forma:

$$P^+(k) = \begin{pmatrix} P^+_{\nu\nu}(k) & P^+_{\nu 1}(k) & ... & P^+_{\nu m}(k) \\ P^{+T}_{\nu 1}(k) & P^+_{11}(k) & ... & P^+_{1m}(k) \\ ... & ... & ... & ... \\ P^{+T}_{\nu m}(k) & P^{+T}_{1m}(k) & ... & P^k_{mm} \end{pmatrix} \tag{8.16}$$

dove:

$$P^+_{\nu\nu}(k) = E[(X_\nu(k) - \hat{X}^+_\nu(k))(X_\nu(k) - \hat{X}^+_\nu(k))^T | Z_k] \tag{8.17}$$

rappresenta la covarianza del veicolo,

$$P^+_{ii}(k) = E[(X_i(k) - \hat{X}^+_i(k))(X_i(k) - \hat{X}^+_i(k))^T | Z_k], \quad i \in \{1, ..., m\} \tag{8.18}$$

rappresenta la covarianza dei *landmark* osservati nell'ambiente,

$$P^+_{\nu i}(k) = E[(X_\nu(k) - \hat{X}^+_\nu(k))(X_i(k) - \hat{X}^+_i(k))^T | Z_k], \quad i \in \{1, ..., m\} \tag{8.19}$$

rappresenta la cross-covarianza fra il veicolo e i *landmark*

$$P^+_{ij}(k) = E[(X_i(k) - \hat{X}^+_i(k))(X_j(k) - \hat{X}^+_j(k))^T | Z_k], \quad \{i, j\} \in \{1, ..., m\} \tag{8.20}$$

rappresenta la cross-covarianza fra i *landmark*. La matrice di covarianza può essere scritta in modo più consistente:

$$P^+(k) = \begin{pmatrix} P^+_{\nu\nu}(k) & P^+_{\nu m}(k) \\ P^+_{\nu m}(k) & P^+_{mm}(k) \end{pmatrix} \tag{8.21}$$

dove $P^+_{mm}(k)$ rappresenta la matrice di covarianza dei landmark , $P^+_{\nu m}(k)$ e $P^+_{\nu m}(k)$ rappresentano la cross-covarianza fra robot e landmark.

Fase di predizione

Il filtro, utilizzando il modello di movimento del veicolo (eq.8.8), fornisce una stima sulla posizione del robot, basandosi sulla stima a posteriori precedente, quindi corretta:

$$X_{\nu}^{-}(k) = f(\hat{X}_{\nu}^{+}(k-1), u(k)) \tag{8.22}$$

Per quanto riguarda i *landmark*, il modello di previsione è descritto dall'equazione 8.9:

$$\hat{X}_{m}^{-}(k) = \hat{X}_{m}^{+}(k-1) \tag{8.23}$$

Questi due modelli insiemi completano la funzione di previsione del filtro:

$$\begin{pmatrix} \hat{X}_{\nu}^{-}(k) \\ \hat{X}_{m}^{-}(k) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_{\nu}^{+}(k-1), u(k)) \\ \hat{X}_{m}^{+}(k-1) \end{pmatrix} \tag{8.24}$$

La funzione che rappresenta il modello di transizione di stato per il veicolo è non lineare, quindi si deve utilizzare la versione estesa del filtro di Kalman (EKF). Si tratta di linearizzare nel punto la funzione $f(...)$. Quindi la previsione anche della matrice di covarianza:

$$P^{-}(k) = \nabla_x f(k) P^{+}(k-1) \nabla_x f^{T}(k) + Q(k) \tag{8.25}$$

L'incertezza dell'ingresso di controllo può essere rappresentata usando l'incertezza di controllo $U(k)$ e lo Jacobiano $\nabla_x f(k)$ della funzione di transizione di stato, valutato attorno l'ingresso di controllo, ottenendo la seguente forma:

$$P^{-}(k) = \nabla_\nu f(k) P^{+}(k-1) \nabla_\nu f^{T}(k) + \nabla_u f(k) U(k) \nabla_u f^{T}(k) + Q(k) \tag{8.26}$$

La 8.26 può essere semplificata considerando che i *landmark* sono tempo invarianti. Questo permette di ridurre la complessità computazionale dell'algoritmo. Solo i termini associati al veicolo vengono aggiornati durante la fase di predizione.

$$\begin{aligned} &\begin{pmatrix} P_{\nu\nu}^{-}(k) & P_{\nu m}^{-}(k) \\ P_{\nu m}^{-T}(k) & P_{mm}^{-}(k) \end{pmatrix} = \\ &= \begin{pmatrix} \nabla_\nu f(k) P_{\nu\nu}^{+}(k-1) \nabla_\nu f^{T}(k) + Q_{\nu\nu}(k) & \nabla_\nu f(k) P_{\nu m}^{+}(k-1) \\ \nabla_\nu f^{T}(k) P_{\nu m}^{+}(k-1) & P_{mm}^{+}(k-1) \end{pmatrix} \end{aligned} \tag{8.27}$$

dove il termine di rumore $Q_{\nu\nu}(k)$, rappresenta sia il rumore di processo, sia il rumore di ingresso di controllo.

La fase di osservazione

La correzione dello stato è realizzabile se e solo se il sistema è in grado di apportare informazioni aggiuntive sullo stato. Ovvero se lo stato, o parte di quest'ultimo, è osservabile.

La fusione sensoriale essenzialmente comincia qui. Grazie allo stato predetto $\hat{X}_{\nu}^{-}(k)$ si calcola un'osservazione predetta:

$$\hat{Z}^{-}(k) = h(\hat{X}^{-}(k)) \tag{8.28}$$

sfruttando il modello di misurazione (eq.8.11). Si acquisiscono le osservazioni per mezzo dei sensori ultrasonici intorno al robot. Tali osservazioni sono legate alle *feature* dell'ambiente.
La differenza tra stima della posizione della *feature* e la misura diretta è un valore che esprime l'errore commesso dal processo di stima. Questo valore si chiama innovazione:

$$\alpha(k) = Z(k) - \hat{Z}^{-}(k) \tag{8.29}$$

Rilevato questo parametro è importante definire la covarianza dell'innovazione $S_{\alpha}(k)$. Si ottiene sfruttando la stima della covarianza dello stato corrente $P^{-}(k)$, dallo Jacobiano del modello di osservazione $\nabla_x h(k)$ e dalla covarianza del modello di osservazione $R(k)$:

$$S_{\alpha}(k) = \nabla_x h(k) P^{-}(k) \nabla_x h^{T}(k) + R(k) \tag{8.30}$$

Nella sezione relativa all'inizializzazione e al controllo delle *feature* in questa parte, dell'algoritmo di fusione sensoriale, si utilizzerà l'innovazione $\alpha(k)$ e la sua matrice di correlazione $S_{\alpha}(k)$ per verificare che l'informazione acquisita non sia soggetta ad un errore molto grave.

L'equazione 8.30 può essere semplificata considerando che ciascuna osservazione è soltanto una funzione della *feature* osservata.
Lo Jacobiano della funzione di osservazione, $\nabla_x h(k)$, è una matrice sparsa nella forma:

$$\nabla_x h(k) = [\nabla_{\nu} h(k) \;\; 0 \;\; K \;\; 0 \;\; \nabla_i h(k) \;\; 0 \;\; K] \tag{8.31}$$

Sostituendo la eq.8.31 nella eq.8.30 si ottiene:

$$\begin{aligned} S_{\alpha}(k) = {} & \nabla_{\nu} h(k) P_{\nu\nu}^{-}(k) \nabla_{\nu} h^{T}(k) + \nabla_i h(k) P_{\nu i}^{-}(k) \nabla_{\nu} h^{T}(k) + \\ & + \nabla_{\nu} h(k) P_{\nu i}^{-}(k) \nabla_i h^{T}(k) + \nabla_i h(k) P_{ii}^{-}(k) \nabla_i h^{T}(k) + R(k) \end{aligned} \tag{8.32}$$

La fase di aggiornamento

La fase di aggiornamento è l'ultima fase del processo di fusione. Individuata una *feature*, acquisita la misurazione associata e calcolata l'innovazione si può aggiornare la stima precedentemente effettuata. L'algoritmo di aggiornamento è il seguente:

$$\hat{X}^+(k) = \hat{X}^-(k) + G(k)\nu(k) \tag{8.33}$$
$$P^+(K) = P^-(K) - G(k)S_\alpha(k)G^T(k) \tag{8.34}$$

La matrice $G(k)$ è definita come il guadagno di Kalman. Essa fornisce dei coefficienti, pesati, da moltiplicare all'innovazione per ottenere il valore di correzione per ogni variabile di stato. La matrice guadagno di Kalman è calcolabile dalla relazione:

$$G(k) = P^-(k)\nabla_x h^T(k)S_\alpha^{-1}(k) \tag{8.35}$$

8.1.4 Inizializzazione dei *landmark*

La fusione sensoriale avviene grazie all'integrazione tra le rilevazioni odometriche e le misure effettuate sulle *feature* individuate nell'ambiente. Quando un nuovo *landmark* viene aggiunto al sistema, si deve inizializzare lo stato e la matrice di correlazione. Nel robot e nel modello esaminati da questa trattazione la posizione del *landmark* è riportata nello stato in coordinate assolute, mentre l'osservazione effettuata corrisponde alla distanza fra il landmark ed il sensore. È necessario utilizzare una funzione che, dato lo stato del robot e la misura effettuata, fissi la posizione del *landmark* nel sistema di coordinate mondo.

$$x_i^+(k) = g_i(\hat{x}_\nu^-(k), y(k)) \tag{8.36}$$

dove le stime $x_i^+(k)$, relative ai *landmark* della mappa, saranno incorporate nel vettore di stato . Anche la covarianza della stima delle nuove *feature* deve essere inizializzata correttamente, perchè la stima iniziale dipende dalla stima corrente del veicolo e quindi è correlata al robot e agli stati della mappa. Se si ignorano le correlazioni, tra le nuove stime dello stato ed il resto dell mappa, si arriva ad una inconsistenza dei dati durante il processo di fusione [10]. La matrice di covarianza inizialmente viene ampliata dal termine di covarianza di osservazione, poi vengono calcolati i termini di cross-covarianza con lo stato, quindi si procede con la fase di osservazione e aggiornamento. Si riporta di seguito il metodo descritto:

Data la previsione a priori della matrice di correlazione 8.25:

$$P^{-}(k) = \begin{pmatrix} P^{-}_{\nu\nu}(k) & P^{-}_{\nu m}(k) \\ P^{-}_{\nu m}(k) & P^{-}_{mm}(k) \end{pmatrix} \tag{8.37}$$

si aggiunge il termine della covarianza d'osservazione $R(k)$:

$$P^{*-}(k) = \begin{pmatrix} P^{-}_{\nu\nu}(k) & P^{-}_{\nu m}(k) & 0 \\ P^{-}_{\nu m}(k) & P^{-}_{mm}(k) & 0 \\ 0 & 0 & R(k) \end{pmatrix} \tag{8.38}$$

La matrice di covarianza finale è calcolata attraverso lo Jacobiano della funzione di inizializzazione g_i, rispetto ai nuovi stati:

$$P^{-}(k) = \nabla_x g(k) P^{*-}(k) \nabla_x g^T(k) \tag{8.39}$$

dove lo Jacobiano della matrice di inizializzazione di ingresso vale:

$$\nabla_x g(k) = \begin{pmatrix} I_\nu & 0 & 0 \\ 0 & I_m & 0 \\ \nabla_\nu g(k) & 0 & \nabla_z g(k) \end{pmatrix} \tag{8.40}$$

Il calcolo di questo aggiornamento comporta un costo computazionale pari a $O(n^3)$. L'operazione può essere semplificata, considerando la natura sparsa delle matrici interessate.

$$P^{-}(k) = \tag{8.41}$$

$$\begin{pmatrix} P^{-}_{\nu\nu}(k) & P^{-}_{\nu m}(k) & P^{-}_{\nu\nu}(k)\nabla_\nu g^T(k) \\ P^{-T}_{\nu m}(k) & P^{-}_{mm}(k) & P^{-}_{\nu m}(k)\nabla_\nu g^T(k) \\ \nabla_\nu g(k) P^{-}_{\nu\nu}(k) & \nabla_\nu g(k) P^{-}_{\nu m}(k) & \nabla_\nu g(k) P^{-}_{\nu\nu}(k) \nabla^T_\nu g(k) + \nabla_z g(k) R(k) \nabla_z g^T(k) \end{pmatrix}$$

Con questa inizializzazione della nuova *feature* si mantiene la consistenza e si genera la corretta cross-correlazione tra il *landmark* e lo stato.

8.1.5 Gestione del filtro

Per rendere affidabile l'algoritmo di fusione è necessario disporre di osservazioni che rispecchino concretamente il mondo osservato. A questo scopo è necessario includere, alla trattazione, una serie di considerazioni. Queste includono i metodi impiegati per l'estrazione e l'identificazione delle *feature* e i metodi necessari a compiere l'associazione di dati una volta ottenute le nuove osservazioni. Questa sezione esamina in dettaglio alcuni di questi problemi.

Estrazione delle *feature*

Nella navigazione basata sul metodo *feature based*, riveste una particolare importanza la capacità dei sensori di individuare *landmark* appropriati e affidabili, grazie ai quali è possibile costruire la mappa dell'ambiente. Il sistema di rilevazione delle *feature* dipende fortemente dall'ambiente in cui il robot è in esercizio e dai sensori a disposizione della macchina, intenti nella rilevazione dell'ambiente.

Negli ambienti interni è possibile individuare le linee descritte dai muri, gli angoli eccetera, grazie a sensori come il laser, il sonar e i sistemi di visione.

Negli ambienti esterni tipicamente non strutturati è difficile individuare riferimenti affidabili. Quindi l'osservazione delle feature non sempre porterebbe giovamenti nel sistema di tracciamento della traiettoria.

La tipologia di sensori, installati nel robot considerato nella seguente trattazione (sensori ultrasonici con portata di circa 2m), consiglia l'utilizzo di ambienti interni (uffici, magazzini, ...).

Controllo dei landmark

Le osservazioni ricevute dal filtro vengono associate ai corrispondenti *landmark* individuati nel percorso. È molto importante che l'osservazione sia affidabile, ovvero che sia associata al landmark giusto. Per risolvere questo problema, si sfruttano metodi statistici. Il metodo utilizzato è quello *nearest neighbor*, ovvero l'associazione più vicina in senso statistico.

Sfruttando l'innovazione $\alpha(k)$ (eq.4.28), l'osservazione $y(k)$ (eq. 4.9) e la covarianza dell'innovazione (eq. 8.32), si può calcolare la norma di Mahalanobis:

$$d_{fi} = \alpha^T(k) S_\alpha^{-1}(k) \alpha(k) \tag{8.42}$$

Il valore calcolato viene confrontato con una valore limite che ne stabilisce la validità. Sperimentalmente se la norma di Mahalanobis e inferiore a 3[2], il landmark associato a questa osservazione è da ritenersi valido.

Si presenta un esempio dove l'indice calcolato "salva" il sistema da associazioni fra dati errate fig. 8.3. In questo caso l'osservazione tra un punto e l'altro varia di pochi centimetri, a causa della diversa distanza tra il muro nei due casi. Se il sistema non rilevasse il cambiamento come un errore, il robot prenderebbe una direzione errata. In questo caso l'utilizzo dell'algoritmo di controllo del landmark, attraverso la norma di Mahalanobis, avvisa il sistema di non considerare l'osservazione. Questo approccio ci consente di evitare grossolani errori di associazione dei dati.

[2] Il valore presentato è stato ricavato sperimentalmente

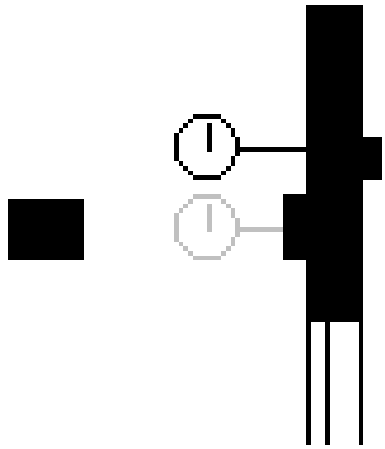

Figura 8.3: Esempio critico per l'associazione dei dati

8.1.6 Proprietà dell'algoritmo SLAM

L'algoritmo descritto possiede le seguenti proprietà che ne caratterizzano il funzionamento:

- la convergenza della stima dello stato nella mappa;
- la necessità di mantenere la consistenza nel processo di stima;
- la complessità di calcolo invocata nel mantenimento della matrice di covarianza dell'AMF.

Convergenza

Newman [11] dimostrò per primo le tre proprietà importanti per lo SLAM. Successivamente furono sviluppate nel lavoro di Csorba [10]ed essenzialmente sono:

- la determinazione di ogni sotto matrice della matrice di covarianza decresce monotonicamente dopo ogni osservazione;
- con l'aumento del numero di osservazioni al limite, le stime del *landmark* diventano completamente correlate;

- il limite più basso sull'incertezza della mappa è funzione dell'incertezza iniziale sulla posizione del veicolo, quando il primo *landmark* viene individuato.

Queste proprietà sono fondamentali nelle applicazioni nel mondo reale.

Mantenimento della consistenza di SLAM

Nell'approccio AMF, per mantenere la consistenza dell'algoritmo, è necessario aggiornare la matrice di covarianza con ogni osservazione usando l'equazione 8.34. Questo è indispensabile per evitare l'inconsistenza nel processo di stima [10]. Dato che le osservazioni delle *feature* sono anticipate rispetto il veicolo, qualsiasi errore nella stima del veicolo sarà fortemente correlata agli errori nella stima della mappa. Le osservazioni sono fatte in funzione della posizione

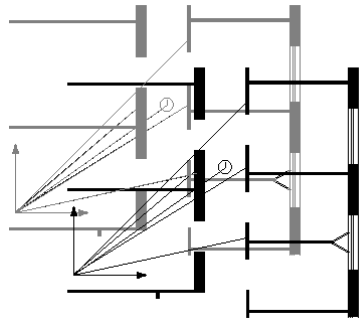

Figura 8.4: Errore sulla stima iniziale della posizione del veicolo

stimata del robot e, commettendo un errore nella posizione iniziale del robot, fig.8.4, questo verrà propagato in tutte le posizioni successive. Se non si dispone d'informazioni esterne circa le posizioni dei landmark o del veicolo, è essenziale mantenere la correlazione fra gli stati per contenere l'errore nella stima.

Complessità computazionale

Il costo computazionale per mantenere le informazioni di correlazione fra le *feature* e il veicolo, rappresenta uno degli ostacoli più significativi per implementare l'algoritmo su larga scala. L'aggiornamento della matrice di covarianza, nell'algoritmo di Kalman diretto, possiede una complessità $O(n^3)$. Nel caso dello

SLAM, la complessità può essere ridotta a $O(n^2)$ grazie alla natura sparsa delle matrici. Ma questo implica che il costo computazionale aumenta, in modo quadratico, col numero di feature individuate nella mappa. Per le mappe ricche di feature, la laboriosità di calcolo renderà rapidamente insolvibile il calcolo dell'aggiornamento. Quindi è necessaria una tecnica efficace nell'amministrazione della mappa per poter controllare questa complessità. Nel nostro caso il numero di feature consente di ignorare questa problematica.

8.1.7 Fallimento EKF-SLAM

L'algoritmo non riesce ad apportare correzioni significative nei seguenti casi:

- divergenza dovuta ad errori di associazione dei dati
- slittamento della mappa
- perturbazione non prevedibile del robot

Fallimento dell'associazione dei dati

Questo tipo di fallimento è dovuto ad una osservazione associata ad una *feature* non affidabile. Generalmente succede quando due feature sono molto vicine fra loro e l'incertezza sulla posizione del veicolo è elevata, rendendo ambigua l'associazione dei dati. Questo tipo di fallimento conduce a errori molto gravi, perchè l'algoritmo sbaglia in modo grossolano, senza accorgersi di quello che succede. L'errore commesso si espande nelle rilevazioni successive. Un modo per risolvere il problema è quello di confrontare la stima data della posizione effettuata con Kalman con la stima odometrica. Se l'entità dell'errore è elevata, la stima fornita potrebbe essere affetta da questo tipo di problema. Purtroppo, questo approccio è attuabile se la stima odometrica non è affetta da errori di grande entità.

Slittamento della mappa

Questo tipo di errore avviene quando la posizione del robot è vicina al limite di errore. A causa della linearizzazione delle funzioni non lineari tutte le feature sono mappate nelle nuove posizioni e un po' spostate rispetto la mappa originale. In caso di ripetizione di questo fenomeno, la mappa slitterà ancora di più, causando una stima inconsistente della mappa.

Fallimento dovuto ad una imprevista perturbazione

Questa eventualità è senz'altro quella meno gestibile fra le tre problematiche. Essenzialmente dipende da sistemi esterni che intervengono a perturbare fortemente il sistema robot-ambiente. Per esempio una buca nel terreno, che crea uno slittamento di una ruota.

8.2 Implementazione dello EKF-SLAM

Finora si è mantenuta l'esposizione in ambito generale. In questa sezione si descrive l'implementazione dell'algoritmo relativamente al robot e al simulatore a disposizione.

8.2.1 I modelli del sistema

Il nucleo del simulatore contiene un sistema di controllo definito **ERASMUS**[3], che simula la dinamica del veicolo (DR) e la percezione sensoriale (PR). I compiti di ERASMUS sono: pianificare la traiettoria, costruire la mappa sulla base delle rilevazioni sensoriali, gestire i sensori e gli attuatori, simulare il comportamento di sensori e attuatori.

Il simulatore della dinamica DR

In ingresso al sistema che descrive le rilevazione odometriche abbiamo i comandi di velocità traslazionale (speed s) (eq.2.5) e rotazionale (jog ω)(eq.2.6) forniti dall'esperto che nel robot reale comanda i motori. Ad intervalli (time-step) di un decimo di secondo si calcola la nuova posizione e il nuovo orientamento $[x(k), y(k), \theta(k)]$ dalle seguenti relazioni:

$$x(k+1) = \begin{cases} x(k) + \frac{s}{\omega}[\sin(\omega\Delta t + \theta(k)) - \sin\theta(k)], & \omega \neq 0 \\ x(k) + s\cos[\theta(k)]\Delta t, & \omega = 0 \end{cases} \tag{8.43}$$

$$y(k+1) = \begin{cases} y(k) + \frac{s}{\omega}[cos(\omega\Delta t + \theta(k)) - \cos\theta(k)], & \omega \neq 0 \\ y(k) + s \cdot \sin[\theta(k)]\Delta t, & \omega = 0 \end{cases} \tag{8.44}$$

$$\theta(k+1) = \omega\Delta t + \theta(k) \tag{8.45}$$

I valori ottenuti vengono passati al modulo di gestione PR e alle finestre che li utilizzano.

[3]Si è trattata in modo più approfondito la costituzione del simulatore nel capitolo 7

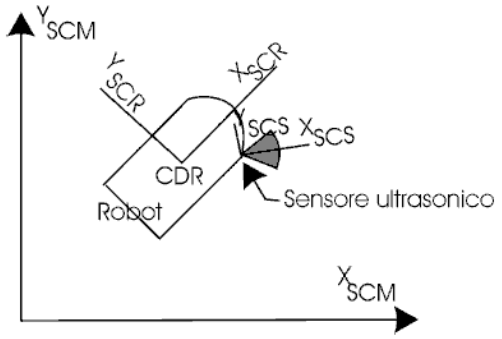

Figura 8.5: Sistemi di coordinate usati nel simulatore, SCM *Sistema di Coordinate Mondo*, SCR *Sistema di Coordinate del Robot*, SCS *Sistema di Coordinate Sensore*

Il simulatore delle percezioni del robot PR

Il sistema fornisce delle letture simulate dei sensori ad ultrasuoni. Per individuare la parte di mondo simulato da analizzare, si sfrutta la posizione e la direzione del robot ricavati da DR (fornito dall'interfaccia utente). Il SCM *Sistema di Coordinate Mondo* è il sistema di coordinate visto dall'utente. La posizione degli ostacoli, la posizione e la direzione del robot, come anche la sua destinazione, sono definite in questo sistema di coordinate. Queste coordinate sono anche usate dalla finestra del mondo simulato per rappresentare l'ambiente.

Il SCR *Sistema di Coordinate del Robot* ha la sua origine in un punto chiamato *Centro Del Robot* (CDR). L'asse x positivo segue la direzione frontale del robot. La posizione di tutto l'equipaggiamento è specificata in queste coordinate. I sensori ultrasonici sono caratterizzati dai seguenti parametri: $(x_{Si}, y_{Si}, \theta_{Si})$, rispettivamente posizione e orientamento dei sensori in SCR, lr che rappresenta la larghezza del raggio o meglio l'angolo di dispersione del sensore ed infine la *portata* del sensore. Questo modo di rappresentare l'insieme dei sensori è conveniente perchè la posizione degli stessi non varia nel SCR. Per ottenere la posizione e l'orientamento del sonar nel SCM si utilizza la seguente trasformazione:

$$\begin{pmatrix} x_{MS}(k) \\ y_{MS}(k) \\ \theta_{MS}(k) \end{pmatrix} = \begin{pmatrix} x(k) \\ y(k) \\ \theta(k) \end{pmatrix} + \begin{pmatrix} \cos\theta(k) & -\sin\theta(k) & 0 \\ \sin\theta(k) & \cos\theta(k) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{Si}(k) \\ y_{Si}(k) \\ \theta_{Si}(k) \end{pmatrix} \tag{8.46}$$

dove $(x(k), y(k))$ è la posizione e $\theta(k)$ è l'orientamento del robot nel SCM. Il SCS, *Sistema di Coordinate Sensore*, ha il sensore come origine e il centro del raggio come asse x positivo. La distanza di un punto dall'origine del sensore nel SCS è uguale alla distanza nel SCM, poichè essa è invariante a roto-traslazioni di coordinate. I confini di acquisizione del sensore sono posti simmetricamente rispetto all'asse x (SCS) nel semipiano positivo. L'angolo di apertura del cono è pari a $lr/2$. Qualunque oggetto che si trovi in quest'area (in grigio in fig.8.5) verrà rilevato dal sensore. Nel simulatore questa operazione corrisponde a controllare se in quest'area compaiono pixel di colore nero. La scansione viene fatta a partire dall'origine e si procede allargandosi verso l'esterno. Il sensore, simulato, tornerà la distanza fra il centro del suo sistema di coordinate e l'oggetto. Questa operazione viene fatta per tutti i sensori. I vari valori di ritorno vengono ceduti all'esperto che si occupa della gestione di quest'ultimi.

Il robot considerato è dotato di 14 sensori ad ultrasuoni disposti lungo un arco nella parte frontale del veicolo. La portata è pari a 2m. L'angolo di dispersione di un sensore è pari a $lr = 4\pi/9$. Il simulatore è in grado di simulare un errore, nella lettura del sensore, sommando un errore di tipo gaussiano di media 0 e varianza del 10%.

Il modello del robot

Per rappresentare il moto del robot in modo reale nel simulatore, si introducono due tipi di errori:

- errori di misura
- errori di controllo

L'errore di misura influenza le grandezze di uscita dell'odometro $(\Delta\rho, \Delta\theta)$ e rappresenta l'effetto degli errori non sistematici. Supponiamo che questo tipo di errore abbia una distribuzione di tipo gaussiano.

$\sigma_\rho^2 = k_p|\Delta\rho(k)|$, la varianza dell'errore della distanza percorsa $\Delta\rho(k)$ è proporzionale alla distanza stessa;

$\sigma_\theta^2(k) = k_\theta^{\Delta\rho}|\Delta\rho(k)| + k_\theta^{\Delta\theta}|\Delta\theta(k)|$, la varianza dell'errore nella varianza dell'angolo di orientamento $\Delta\theta$ dipende sia dalla distanza percorsa, sia dalla variazione dell'angolo di orientamento $\Delta\rho$.

L'errore di controllo influenza i comandi di velocità traslazionale e rotazionale forniti dall'esperto, che nel robot reale controlla i motori. Anche questo tipo di errore può essere di tipo gaussiano con media nulla.

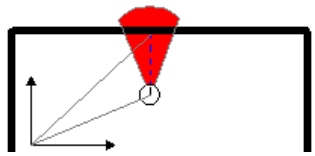

Figura 8.6: Misura della distanza con un sensore ultrasonico

Il modello di misura del sensore

Di seguito si descrive il modello utilizzato per i sensori di tipo ultrasonico.

Prima di iniziare la trattazione si deve fare la seguente ipotesi: si suppone di operare in ambienti con pareti ortogonali fra loro e ortogonali con il pavimento. Una parete, nel sistema di coordinate SCM, è rappresentabile come una terna (P_r, P_n, P_ν), dove P_r è la distanza assoluta tra il piano e l'origine, P_n è l'angolo fra la parete e l'asse x del sensore ultrasonico, mentre P_ν è un valore booleano $P_\nu \in \{1, -1\}$ che rappresenta la faccia riflettente del piano. La distanza $y_i(k)$, misurata del sonar i-esimo di posizione $(x_{MS}(k), y_{MS}(k))$ e orientamento $\theta_{MS}(k)$, nel piano (P_r, P_n, P_ν) è esprimibile:

$$z_i(k) = P_\nu[P_r - x_{MS}(k)\cos(P_n) - y_{MS}(k)\sin(P_n)] \tag{8.47}$$

considerando l'angolo di dispersione $P_n \in [\theta_M - lr/2, \theta_m + lr/2]$, dove lr è l'angolo di dispersione del sensore. Inserendo la eq.8.46 nella eq.8.47, si ottiene:

$$z_i(k) = P_\nu[P_r - (x(k) + x_{Si}\cos\theta(k) - y_{Si}\sin\theta(k))cos(P_n) + \tag{8.48}$$

$$-(y(k) + x_{Si}sin\theta(k) + y_{Si}cos\theta(k))sin(P_n)]$$

Considerando anche il modello di errore gaussiano introdotto dal simulatore:

$$z_i(k) = P_\nu[P_r - (x(k) + x_{Si}\cos\theta(k) - y_{Si}\sin\theta(k))cos(P_n) + \tag{8.49}$$

$$-(y(k) + x_{Si}sin\theta(k) + y_{Si}cos\theta(k))sin(P_n)] + w_r(k)$$

Considerando un ambiente con pareti ortogonali, la distanza fra una parete verticale[4], rappresentata dalla tripla (P_r, P_n, P_ν) con $P_n = 1/2\pi$, e il robot è:

$$z_{iv}(k) = P_\nu[P_r - y(k) - x_{Si}\sin\theta(k) - y_{Si}\cos\theta(k)] + w_r(k) \tag{8.50}$$

[4]Si considera parete verticale un muro ortogonale all'asse y in coordinate SCM

mentre la distanza fra una parete orizzontale[5], rappresentata dalla tripla (P_r, P_n, P_ν) con $P_n = 0$ è:

$$z_{io}(k) = P_\nu[P_r - x(k) - x_{Si}\sin\theta(k) + y_{Si}\cos\theta(k)] + w_r(k) \tag{8.51}$$

8.2.2 Algoritmo dello SLAM

Si riporta in forma di pseudo codice l'algoritmo EKF-SLAM:

```
lettura dei sonar
estrazione delle feature
inizializzazione delle feature
while(fine punti da esaminare){
    lettura dell'odometro
    predizione
    lettura dei sonar
    estrazione delle feature
    if (nuova feature)
        inizializzazione della feature
    osservazione
    associazione dei dati
        if (feature gia esistente)
            aggiornamento con Kalman
        else
            inizializzazione della feature
}fine while
```

Lettura dei sonar

I sensori ad ultrasuoni sono posizionati a bordo del robot nella parte antistante il veicolo. Possiamo scegliere qualsiasi sensore per determinare le pareti (*feature*). La tecnica di navigazione è wall following per cui è preferibile scegliere i sonar laterali o frontali per determinare la distanza fra le pareti a destra a sinistra e di fronte al veicolo. Il numero di sonar da utilizzare per estrarre le feature è limitato solo dal costo computazionale. Il limite superiore è pari al numero di sonar a disposizione nel robot.

[5]Per parete orizzontale si intende un muro ortogonale all'asse x in coordinate SCM

Estrazione delle feature

Una volta ottenuta la lettura del sonar, si può associare la lettura ad una parete (*landmark*) verticale o orizzontale attraverso le seguenti relazioni:

$$P_{rv} = P_v z_{iv}(k) + y(k) + x_{Si}\sin\theta(k) + y_{Si}\cos\theta(k) \quad (8.52)$$

$$P_{ro} = P_v z_{io}(k) + x(k) + x_{Si}\sin\theta(k) - y_{Si}\cos\theta(k) \quad (8.53)$$

dove P_{rv} rappresenta la parete verticale ($P_n = 1/2\pi$), P_{ro} rappresenta la parete orizzontale ($P_n = 0$), $(x_{Si}, y_{Si}, \theta_{Si})$ sono la posizione e l'orientamento dei sensori e $(x(k), y(k), \theta(k))$ sono la posizione e l'orientamento del robot.
Per determinare l'orientamento delle pareti (verticale o orizzontale) consideriamo la seguente ipotesi:

Se l'orientamento dell'asse acustico del sonar (θ_{MS}) ***è contenuto in uno degli intervalli*** $[-1/4\pi, 1/4\pi]$ ***o*** $[3/4\pi, -3/4\pi]$***, allora la parete è verticale. In caso contrario è orizzontale.***

Nel caso quest'ipotesi non sia vera l'orientamento viene associato incorrettamente, tuttavia non ci sono problemi perchè tali associazioni errate vengono eliminate nei passi successivi.

Inizializzazione delle feature

Determinata la posizione è l'orientamento del nuovo *landmark* è necessario aggiungerlo al vettore di stato e inizializzare la nuova matrice di correlazione con il nuovo elemento (eq.8.77,8.39) mediante la:

$$\Delta_\nu g(k) = \begin{cases} 0 & 1 & x_{Si}\cos\theta(k) - y_{Si}\sin\theta(k) & \text{se parete verticale} \\ 1 & 0 & -x_{Si}\cos\theta(k) - y_{Si}\sin\theta(k) & \text{se parete orizzontale} \end{cases} \quad (8.54)$$

e

$$\Delta_\nu g(k) = [P_\nu] \quad (8.55)$$

Predizione

Conoscendo lo stato del robot $(x(k), y(k), \theta(k))$, la velocità traslazionale $s(k)$ (speed) e quella rotazionale $\omega(k)$ (jog), si applicano le relazioni odometriche (eq. 8.43, 8.44, 8.45) per ottenere la predizione dello stato del veicolo, mentre

lo stato dei landmark rimane invariato (eq. 8.9).
In questa fase viene anche aggiornata la matrice di covarianza (eq.8.26):

$$P^{-}_{\nu\nu}(k) = \nabla_\nu f(k) P^{+}_{\nu\nu}(k-1) \nabla_\nu f^T(k) + \nabla_u f(k) U(k) \nabla_u f^T(k) + Q(k) \quad (8.56)$$

dove:

$$\Delta_\nu f(k) = \begin{cases} \begin{pmatrix} 1 & 0 & \frac{s}{\omega}[\cos(\omega\Delta t + \theta(k)) - \cos(\theta(k))] \\ 0 & 1 & -\frac{s}{\omega}[\sin(\omega\Delta t + \theta(k)) - \sin(\theta(k))] \\ 0 & 0 & 1 \end{pmatrix} & \omega \neq 0 \\ \begin{pmatrix} 1 & 0 & -s\sin(\theta(k))\Delta t \\ 0 & 1 & s\cos(\theta(k))\Delta t \\ 0 & 0 & 1 \end{pmatrix} & \omega = 0 \end{cases} \quad (8.57)$$

$$\Delta_u f(k) = \begin{cases} \begin{pmatrix} \frac{1}{\omega}(\sin(\omega\Delta t + \theta(k)) - \sin\theta(k)) & 0 \\ \frac{1}{\omega}(\cos(\omega\Delta t + \theta(k)) - \cos\theta(k)) & 0 \\ 0 & \Delta t \end{pmatrix} & \omega \neq 0 \\ \begin{pmatrix} cos(\theta(k))\Delta t & 0 \\ sin(\theta(k))\Delta t & 0 \\ 0 & \Delta t \end{pmatrix} & \omega = 0 \end{cases} \quad (8.58)$$

con

$$U(k) = \begin{pmatrix} \sigma_s^2(k) & 0 \\ 0 & \sigma_\omega^2(k) \end{pmatrix} \quad (8.59)$$

e

$$Q(k) = \begin{pmatrix} \sigma_x^2(k) & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{pmatrix} \quad (8.60)$$

Osservazione

La fusione sensoriale si realizza nel seguente modo:

1. si crea una stima dell'osservazione;
2. la si confronta con l'osservazione acquisita;
3. si utilizza la differenza per correggere tutto lo stato.

Usando il modello di misura del sensore, si calcola la stima della misura nel seguente modo:

$$\hat{z}^-(k) = \begin{cases} P_\nu(x_i^- - y(k) - x_{Si}\sin\theta(k) - y_{Si}\cos\theta(k)) & \text{parete verticale} \\ P_\nu(y_i^- - x(k) - x_{Si}\sin\theta(k) + y_{Si}\cos\theta(k)) & \text{parete orizzontale} \end{cases} \tag{8.61}$$

Dopo l'acquisizione dai sensori ultrasonici, si deve associare l'osservazione alla feature corrispondente nell'ambiente. La differenza tra l'osservazione attuale $z(k)$, ricevuta dai sensori di sistema e l'osservazione prevista $\hat{z}^-(k)$ si chiama innovazione $\alpha(k)$:

$$\alpha(k) = z(k) - \hat{z}^-(k) \tag{8.62}$$

Conoscendo l'innovazione si procede col calcolo della covarianza dell'innovazione $S(k)$ (eq.8.30):

$$S_\alpha(k) = \nabla_x h(k) P^-(k) \nabla_x h^T(k) + R(k) \tag{8.63}$$

dove $\Delta_\nu h(k)$ è lo Jacobiano del modello di misura rispetto gli stati della mappa (eq. 8.31) dato dalle relazioni:

$$\Delta_\nu h(k) = \begin{cases} 0 & -P_\nu & -x_{Si}\cos\theta(k)P_\nu + y_{Si}\sin\theta(k)P_\nu & \text{parete verticale} \\ -P\nu & 0 & x_{Si}\sin\theta(k)P_\nu + y_{Si}\cos\theta(k)P_\nu & \text{parete orizzontale} \end{cases} \tag{8.64}$$

$$\Delta_i h(k) = \begin{cases} 0 & P_\nu & \text{parete verticale} \\ P_\nu & 0 & \text{parete orizzontale} \end{cases} \tag{8.65}$$

Associazione dei dati

L'associazione dei dati sfrutta:

la tecnica di *nearest neighbur* , trattata nel sotto paragrafo 8.1.5, nella parte di controllo del *landmark*.

il vincolo odometrico , dove si paragona lo stato del robot, dopo l'aggiornamento, con la predizione dello stato ottenuta per mezzo dell'odometria. Se la stima dello stato è fuori dal vincolo di errore odometrico, la nuova feature viene inizializzata.

Aggiornamento

Quando l'osservazione acquisita appartiene ad una feature già esistente si può procedere con l'aggiornamento, come visto nel sotto paragrafo 8.1.3, nella parte relativa all'aggiornamento.

8.2.3 Cambiamento delle coordinate e applicazione dello SLAM

A causa della non linearità del modello odometrico e del modello di misura si utilizza l'algoritmo Extended Kalman Filter, piuttosto che il Kalman Filter. L'EKF prevede di linearizzare nel punto i modelli. Quest'operazione introduce dell'errore, compromettendo la convergenza del filtro e rendendo il sistema instabile. Al fine di risolvere questo problema, si propone un altro metodo: il cambiamento delle coordinate. Lo si riporta per completezza espositiva, ma non è stato implementato nelle simulazioni, dati i risultati modesti che comporta. Per ulteriori informazioni si veda [7].

Per ottenere un modello lineare di predizione odometrica ed un'esatta predizione dello stato e della matrice di covarianza, si esegue il seguente cambiamento di coordinate:

$$\begin{array}{rcl} \xi_1 & = & x\cos(\theta) + y\sin(\theta) \\ \xi_2 & = & y\cos(\theta) - x\sin(\theta) \\ \xi_3 & = & -\theta \end{array} \tag{8.66}$$

applicandole si ottiene il modello lineare dell'odometria:

$$\xi(k) = A(k)\xi(k)\xi + u(k) \tag{8.67}$$

dove:

$$A(k) = \begin{cases} \begin{pmatrix} \cos(\omega\Delta t) & \sin(\omega\Delta t) & 0 \\ -\sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} & \omega \neq 0 \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \omega = 0 \end{cases} \tag{8.68}$$

e

$$u(k) = \begin{cases} \begin{pmatrix} \sin(\Delta t\omega(k))s(k)/\omega(k) \\ (\cos(\Delta t\omega(k)) - 1)s(k)/\omega(k) \\ -\Delta t\omega(k) \end{pmatrix} & \omega \neq 0 \\ \begin{pmatrix} s(k)\Delta t \\ 0 \\ -\Delta t\omega(k) \end{pmatrix} & \omega = 0 \end{cases} \tag{8.69}$$

Anche il modello di misura deve essere modificato sostituendo le nuove coordinate (ξ_1, ξ_2, ξ_3) alle (x, y, z) tramite la trasformazione 8.66. Si noti che gli stati della mappa non vengono modificati con le nuove coordinate.

L'algoritmo non viene cambiato con le nuove coordinate, ma non c'è più bisogno di calcolare lo Jacobiano $\Delta_\nu f(k)$ in fase di predizione. Si sostituisce con la matrice lineare $A(k)$ dato che il modello dell'odometria è adesso lineare. Inoltre c'è ancora bisogno di calcolare gli Jacobiani di $\Delta_\nu g(k), \Delta_z g(k), \Delta_\nu h(k)$ e $\Delta_i h(k)$ rispetto le nuove coordinate.

8.3 RLS SLAM

L'algoritmo di SLAM basato su RLS è derivato dall'EKF-SLAM. Lo scopo della tesi è di fornire un algoritmo di fusione sensoriale secondo la teoria fondamentale di Kalman basato sull'algoritmo RLS.

La tecniche di lettura dei sonar, l'estrazione delle *feature* e la lettura dell'odometria sono le stesse viste nell'algoritmo precedentemente esposto. La predizione dello stato, l' associazione dei dati e l'aggiornamento si devono ridefinire in base al nuovo algoritmo. Dalle equivalenze riportate nel capitolo 6 si intuisce facilmente che tale operazione non è al quanto complicata. Tuttavia l'equivalenza è valida solo in un specifico caso dell'algoritmo di Kalman (par. 6.1). Si dovranno compiere alcune considerazioni per porsi in tale situazione.

8.3.1 Equivalenza tra i due sistemi

Il modello di stato che descrive il sistema (8.70) non è uguale al caso di equivalenza (8.71).

$$\begin{cases} X(k) = & f(X(k-1), u(k)) + \nu(k) \\ Z(k) = & h(X(k), w(k)) \end{cases} \tag{8.70}$$

$$\begin{cases} X(k+1) = & \lambda^{-1/2} X(k) \\ z(k) = & U^T(n) X(k) + \nu(k) \end{cases} \tag{8.71}$$

Per applicare il filtro di Kalman si linearizza nel punto il modello di stato. Per rientrare nel caso di equivalenza ciò non è sufficiente. La matrice di transizione

di stato utilizzata nel programma di simulazione è la seguente:

$$A(k) = \begin{cases} \begin{pmatrix} 1 & 0 & \frac{s}{\omega}[\cos(\omega\Delta t+\theta)-\cos\theta] & 0 & 0 & 0 \\ 0 & 1 & \frac{s}{\omega}[\sin(\omega\Delta t+\theta)-\sin\theta] & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \omega \neq 0 \\ \begin{pmatrix} 1 & 0 & -s\sin(\theta)\Delta t & 0 & 0 & 0 \\ 0 & 1 & s\cos(\theta)\Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \omega = 0 \end{cases} \tag{8.72}$$

È stato previsto di utilizzare tre landmark.

Si vede che l'equivalenza fra i due sistemi (eq.8.72,8.71) è vera se $s = 0$, $\omega = 0$ e $\lambda = 1$. La situazione corrisponde nell'avere il robot fermo e memoria dell'algoritmo RLS infinita. Questa affermazione molto probabilmente avrà scosso il lettore. Tuttavia ciò corrisponde a iterare l'algoritmo nel punto e non da punto a punto come nel caso EKF. L'associazione dei dati rimane valida, perchè si suppone che da un punto a quello adiacente, l'ambiente e lo stato non varino eccessivamente.

8.3.2 Le dimensioni del problema

Un altro ostacolo alla corrispondenza fra i due algoritmi lo pongono le dimensioni del problema. Per avere la congruenza, l'uscita del filtro non può essere multipla. Si deve stringere il problema ad una sola osservazione. Ciò non costituirà una limitazione in quanto basterà duplicare le strutture dati, una per ogni osservazione e fondere infine tutte le rilevazioni (fig. 8.7). Nel caso specifico, questo approccio coincide col filtrare attraverso Kalman le stime dei vari sensori. Infatti nel simulatore la varianza delle misurazioni acquisite è la stessa, per cui la media è la stima migliore.

In un'implementazione reale del sistema, la varianza della rilevazioni ultrasoniche dipenderebbe dal tipo di superficie individuata dal sensore. Quindi la stima ottima sarebbe calcolabile dalla relazione:

$$\hat{X}(n) = \frac{\sigma_{sx}^2\sigma_{fr}^2}{\sigma_{sx}^2\sigma_{dx}^2\sigma_{fr}^2}\hat{X}_{dx} + \frac{\sigma_{dx}^2\sigma_{fr}^2}{\sigma_{sx}^2\sigma_{dx}^2\sigma_{fr}^2}\hat{X}_{sx} + \frac{\sigma_{sx}^2\sigma_{dx}^2}{\sigma_{sx}^2\sigma_{dx}^2\sigma_{fr}^2}\hat{X}_{fr} \tag{8.73}$$

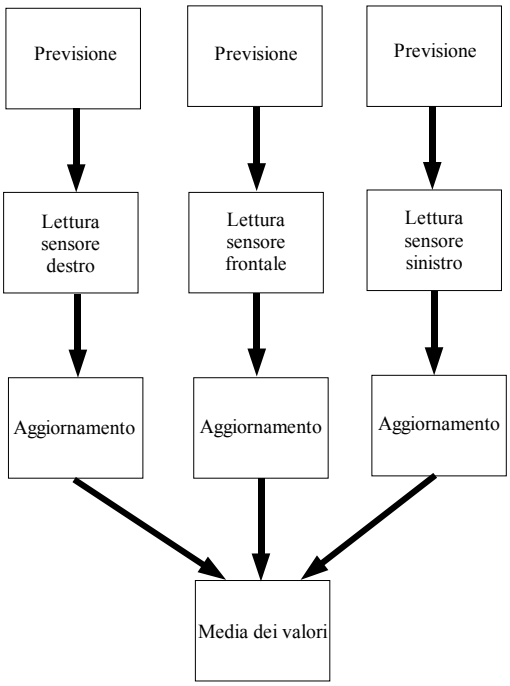

Figura 8.7: Schema a blocchi dell'algoritmo RLS-SLAM a più sensori

dove $\hat{X}_{dx}$ è la stima fornita dal sensore destro, $\hat{X}_{sx}$ è la stima fornita dal sensore sinistro, $\hat{X}_{fr}$ è la stima fornita dal sensore frontale. Le varianze delle stime sono calcolate in base all'errore:

$$\sigma_{ii}^2(n) = \sum_{j=1}^{n} \frac{[\hat{X}(j) - \hat{X}_{ii}(j)]^2}{n-1} \tag{8.74}$$

Il calcolo della varianza tuttavia deve essere fatto in modo ricorsivo per consentirne l'utilizzo in tempo reale:

$$\sigma_{ii_n}^2 = \frac{n-2}{n-1}\sigma_{ii_{n-1}}^2 + \frac{(\hat{X}_{ii_n} - \hat{X})^2}{n-1} \tag{8.75}$$

8.3.3 Utilizzo della radice quadrata della matrice di correlazione

L'implementazione dell'algoritmo RLS-SQR prevede l'utilizzo della decomposizione (eq.8.76) della matrice di covarianza. Quindi tutte le relazioni, che implementano la fusione sensoriale attraverso la matrice di correlazione, devono essere ridefinite.

$$R_{XX}^{-1}(k) = S_k S_k^T = \lambda P(k) \tag{8.76}$$

Si ricorca l'equivalenza:

$$\lambda^{-1} K(k) \Longleftrightarrow P(k)$$

dove $K(k)$ è la matrice di correlazione dell'ingresso in RLS.

Le operazioni di lettura dei sonar, l'estrazione delle *feature*, la lettura odometrica e l'osservazione sono le medesime del filtro di Kalman. Tutto il resto cambia in funzione del nuovo algoritmo.

Inizializzazione del landmark

Individuato un nuovo *landmark*, la matrice di covarianza va ampliata con la covarianza dell'osservazione acquisita (eq.8.77). Avendo ridotto la dimensione del modello di stato e ponendo come variabile di stato l'ultimo elemento del vettore, si verifica facilmente che:

$$S_k^{*-} = \begin{pmatrix} S_{k_{\nu\nu}}^- & S_{k_{\nu m}}^-(k) \\ 0 & R'(k) \end{pmatrix} \tag{8.77}$$

dove $R'(k)$ è pari a:

$$R'(k) = \sqrt{R(k)} \tag{8.78}$$

dove $R(k)$ rappresenta la matrice di covarianza dell'osservazione.
Quindi si deve moltiplicare la matrice di transizione di stato per la funzione di inizializzazione:

$$P^-(k) = \nabla_x g(k) P^{*-}(k) \nabla_x g^T(k) \tag{8.79}$$

Non avendo la matrice di correlazione, ma una sua decomposizione. Si deve riscrivere la relazione per la matrice S.

$$S_k S_k^T = \nabla_x g(k) S_k^{*-} S_k^{T*-} \nabla_x g^T(k) \tag{8.80}$$

La decomposizione della matrice è una decomposizione di Cholesky. Da cui si vede che:

$$S_k = \nabla_x g(k) S_k^{*-}$$

$$S_k^T = S_k^{-*T} \nabla_x g^T(k)$$

Infatti facendo la trasposta membro a membro della seconda relazione si ottiene:

$$S_k = \nabla_x g(k) S_k^{*-}$$

Quindi la funzione di inizializzazione per S_k:

$$S_k = \nabla_x g(k) S_k^{*-}(k) \tag{8.81}$$

dove la matrice $\nabla_x g(k)$ è la stessa dell'algoritmo EKF-SLAM.

Verifica del *landmark*

Così come per l'algoritmo di Kalman, l'osservazione raccolta è valida solo se la *feature* alla quale è associata è affidabile. Il controllo della *feature* anche in questo algoritmo è fatta attraverso la tecnica *nearest neighbor* con la norma di Mahalanobis:

$$d_{fi} = \alpha^T(k) S_\alpha^{-1}(k) \alpha(k)$$

Il calcolo della matrice S_α^{-1} è fatto questa volta attraverso la relazione:

$$S_\alpha(k) = \nabla_x h(k) S_k^- S_k^{T-} \nabla_x h^T(k) \tag{8.82}$$

dove S_k è la decomposizione della matrice di covarianza.

Quindi, come per EKF-SLAM, quando il valore d_{fi} sale, oltre un certo limite, la feature non viene più considerata affidabile.

Cancellazione del *landmark*

Quando la *feature* non è più affidabile si deve cancellarla sia dal vettore di stato, sia dalla matrice di covarianza. Come nel caso precedente, si ha a disposizione solo la decomposizione della matrice di covarianza. Quindi la cancellazione del landmark deve avvenire in base alla matrice S.

Come per il caso precedente si parte dalla matrice P dell'algoritmo EKF-SLAM.

$$R_{XX}^{-1}(k) = S_k S_k^T = \lambda P(k)$$

quindi sapendo che la cancellazione del *landmark* nella matrice $P(n)$ comporta l'azzeramento dei termini nella colonna e nella riga corrispondente al landmark,

si può calcolare quali termini azzerare in S_k risolvendo il sistema di equazioni:

$$\begin{pmatrix} \star & \star & \star & 0 \\ \star & \star & \star & 0 \\ \star & \star & \star & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ 0 & e & f & g \\ 0 & 0 & h & i \\ 0 & 0 & 0 & j \end{pmatrix} \begin{pmatrix} a & 0 & 0 & 0 \\ b & e & 0 & 0 \\ c & f & h & 0 \\ d & g & i & j \end{pmatrix} \tag{8.83}$$

Il sistema trova facilmente soluzione per $j = 0$. Quindi ponendo $j = 0$ si cancella l'influenza del landmark nella matrice S_k.

Predizione

La fase di predizione in RLS-SLAM avviene in modo completamente diverso dalla fase di predizione in Kalman. Si rammenta che RLS itera nel punto e non da punto a punto. Questo comporta che lo stato nel punto non vari[6]:

$$X(k+1) = \lambda^{1/2} X(k) \tag{8.84}$$

con $\lambda = 1$. Lo spostamento in base all'odometria è gestito da punto a punto così come visto per l'EKF-SLAM.

Aggiornamento

La fase di aggiornamento è simile all'algoritmo EKF-SLAM, ma è applicata in modo diverso. La relazione è quella vista nel capito di descrizione RLS-SQR e vale:

$$X(k) = \lambda^{-1/2} X(k-1) + \lambda^{-1/2} C(k) \alpha(k) \tag{8.85}$$

Il calcolo dell'innovazione $\alpha(k)$ utilizza la stessa funzione dell'algoritmo di Kalman ed è riportata in RLS secondo l'equivalenza esistente fra i due algoritmi:

$$\alpha(k) = \lambda^{-k/2} \xi(k)$$

Il calcolo del guadagno di Kalman è implementato attraverso l'algoritmo RLS-SQR:

$$C(k) = S_k D_k$$

dove S_k e D_k appartengono all'algoritmo RLS.

[6] robot virtualmente fermo

8.4 Riassunto algoritmi

8.4.1 EKF-SLAM

Si riportano di seguito le relazioni che governano l'algoritmo:
Predizione:

$$\begin{pmatrix} \hat{X}_\nu^-(k) \\ \hat{X}_m^-(k) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu^+(k-1), U(k)) \\ \hat{X}_m^+(k-1) \end{pmatrix} \tag{8.86}$$

$$P^-(k) = \nabla_x f(k) P^+(k-1) \nabla_x f^T(k) + Q(k) \tag{8.87}$$

$$\hat{Z}^-(k) = h(\hat{X}^-(k)) \tag{8.88}$$

Osservazione:

$$\alpha(k) = Z(k) - \hat{Z}^-(k) \tag{8.89}$$

$$S_\alpha(k) = \nabla_x h(k) P^-(k) \nabla_x h^T(k) + R(k) \tag{8.90}$$

Inizializzazione feature:

$$x_i^+(k) = g_i(\hat{x}_\nu^-(k), y(k)) \tag{8.91}$$

$$P^-(k) = \nabla_x g(k) P^{*-}(k) \nabla_x g^T(k) \tag{8.92}$$

Controllo feature:

$$d_{fi} = \alpha^T(k) S_\alpha^{-1}(k) \alpha(k) \tag{8.93}$$

Aggiornamento:

$$G(k) = P^-(k) \nabla_x h^T(k) S_\alpha^{-1}(k) \tag{8.94}$$

$$\hat{X}^+(k) = \hat{X}^-(k) + G(k)\nu(k) \tag{8.95}$$

$$P^+(k) = P^-(K) - G(k) S_\alpha(k) G^T(k) \tag{8.96}$$

8.4.2 RLS-SLAM

Nelle relazioni dell'algoritmo n rappresenta il numero di punti, k il numero di iterazioni:
Spostamento da punto a punto:

$$\begin{pmatrix} \hat{X}_\nu(n) \\ \hat{X}_m(n) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu(n-1), U(n)) \\ \hat{X}_m(n-1) \end{pmatrix} \tag{8.97}$$

Predizione:

$$\hat{X}^-(k) = \hat{X}^+(k-1) \tag{8.98}$$

Filtraggio nel punto n:

$$K(k=0) = K(n-1)$$

$$\hat{X}(k=0) = \hat{X}(n-1)$$

$$\hat{Z}^-(k) = h(\hat{X}^-(k)) \tag{8.99}$$

$$\alpha(k) = Z(k) - \hat{Z}^-(k) \tag{8.100}$$

$$C(k) = \frac{\lambda^{-1}K(k-1)H_k}{1+\lambda^{-1}H_k^T K(k-1)H_k} \tag{8.101}$$

$$\hat{X}(k) = \lambda^{-1/2}\hat{X}(k-1) + \lambda^{-1/2}C(k)\alpha(k) \tag{8.102}$$

$$K(k) = \lambda^{-1}K(k-1) - \lambda^{-1}C(k)H_k K(k-1) \tag{8.103}$$

8.4.3 RLS-SQR-SLAM

Nelle relazioni dell'algoritmo n rappresenta il numero di punti, k il numero di iterazioni, in questo caso al posto della matrice K si utilizza la matrice S:
Spostamento da punto a punto:

$$\begin{pmatrix} \hat{X}_\nu(n) \\ \hat{X}_m(n) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu(n-1), U(n)) \\ \hat{X}_m(n-1) \end{pmatrix} \tag{8.104}$$

Predizione:

$$\hat{X}^-(k) = \hat{X}^+(k-1) \tag{8.105}$$

Filtraggio nel punto n:

$$S(k=0) = S(n-1)$$

$$\hat{X}(k=0) = \hat{X}(n-1)$$

$$\hat{Z}^-(k) = h(\hat{X}^-(k)) \tag{8.106}$$

$$\alpha(k) = Z(k) - \hat{Z}^-(k) \tag{8.107}$$

$$
\begin{aligned}
F(k) &= S^T(k-1)H_k && (8.108)\\
L(k) &= S(k-1)F(k) && (8.109)\\
\beta(k) &= \lambda + F^T(k)F(k) && (8.110)\\
\alpha(k) &= \frac{1}{\beta(k) + \sqrt{\lambda\beta(k)}} && (8.111)\\
S(k) &= \frac{1}{\sqrt{\lambda}}\left[S(k-1) - \alpha(k)L(k)F^T(k)\right] && (8.112)\\
C(k) &= \frac{L(k)}{\beta(k)} && (8.113)
\end{aligned}
$$

$$\hat{X}(k) = \lambda^{-1/2}\hat{X}(k-1) + \lambda^{-1/2}C(k)\alpha(k) \tag{8.114}$$

8.4.4 fast RLS-SQR-SLAM

Nelle relazioni dell'algoritmo n rappresenta il numero di punti, k il numero di iterazioni:
Spostamento da punto a punto:

$$\begin{pmatrix} \hat{X}_\nu(n) \\ \hat{X}_m(n) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu(n-1), U(n)) \\ \hat{X}_m(n-1) \end{pmatrix} \tag{8.115}$$

Predizione:

$$\hat{X}^-(k) = \hat{X}^+(k-1) \tag{8.116}$$

Filtraggio nel punto n:

$$S(k=0) = S(n-1)$$

$$\hat{X}(k=0) = \hat{X}(n-1)$$

$$v_n = H_k(k\%N)$$

dove N rappresenta l'ordine del filtro;

$$
\begin{aligned}
f_{k-1}(k) &= v_n + D_{k-1}^T(\sqrt{\lambda}T_{k-1}^{-1}Z_{k-1}) && (8.117)\\
f_k(k) &= \gamma_{k-1}f_{k-1}(k) && (8.118)\\
Z_k &= \sqrt{\lambda}T_{k-1}^{-1}Z_{k-1} - D_{k-1}f_{k-1}(k) && (8.119)\\
\alpha_k &= \lambda\alpha_{k-1} + f_k(k)f_{k-1}(k) && (8.120)\\
\sigma_k &= \lambda\sigma_{k-1} + v_k^2 && (8.121)\\
\overline{D}_k &= \begin{bmatrix} D_k \\ \beta_k^{-1/2}b_k(k) \end{bmatrix} && (8.122)\\
\gamma_k &= \gamma_{k-1} - \alpha_k^{-1}f_k^2(k) + \beta_k^{-1}b_k^2(k) && (8.123)\\
T(i,j) &= d_i d_j c_j e_j^{-1/2} && (8.124)\\
S_k &= \frac{1}{\sqrt{\lambda}}S_{k-1}T_k && (8.125)
\end{aligned}
$$

$$\hat{Z}^-(k) = h(\hat{X}^-(k)) \tag{8.126}$$

$$\alpha(k) = Z(k) - \hat{Z}^-(k) \tag{8.127}$$

$$\hat{X}(k) = \lambda^{-1/2}\hat{X}(k-1) + \lambda^{-1/2}C(k)\alpha(k) \tag{8.128}$$

Capitolo 9

Risultati ottenuti

Sono state eseguite numerose prove degli algoritmi di fusione presentati, al fine di testare ogni possibile caratteristica.

9.1 Simulazioni off-line

Le simulazioni off-line sono state condotte seguendo il seguente schema:

1. Simulazione della navigazione del robot, con acquisizione dei dati sensoriali ad ogni passo;
2. Aggiunta di rumore gaussiano, nei dati raccolti, a seconda del tipo di variabile;
3. Ricostruzione a posteriori della traiettoria secondo i vari metodi, sfruttando i dati raccolti affetti da errore.

Con questo procedimento si vuole ricostruire il percorso tracciato dal robot, eliminando gli errori introdotti durante la simulazione. Lo scopo prefissato è quello di costruire una traccia il più prossima possibile alla traccia reale.

Gli algoritmi coinvolti sono tutti quelli visti nel capitolo 8, ovvero:

- Odometria;
- Algoritmo di fusione sensoriale basato solo sul filtro di Kalman;
- Algoritmo di fusione sensoriale basato sul filtro di Kalman ed il filtro RLS classico;

- Algoritmo di fusione sensoriale basato sul filtro di Kalman ed il filtro SQR-RLS;
- Algoritmo di fusione sensoriale basato sul filtro di Kalman ed il filtro fast SQR-RLS.

Nelle simulazioni seguenti vengono testati tutti gli algoritmi citati. In ogni immagine vengono riprodotte le traccie relative al percorso reale, a quello ricostruito tramite l'odometria, al percorso ricostruito tramite il filtro di Kalman ed al percorso ricostruito tramite il filtro di Kalman attraverso RLS. In questo modo possono essere testati i singoli algoritmi RLS.

Il rumore gaussiano aggiunto ai dati d'ingresso (speed, jog) è stato dimensionato in base al modello, già studiato nella tesi precedente da Ervin Čeperić [7], riassunto di seguito [54]:

$$err = gauss(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{9.1}$$

dove μ e σ^2 corrispondono alla media e alla varianza della distribuzione normale d'errore. Nelle prove che seguono, per entrambe le variabili, la media dell'errore è nulla. Nel paragrafo successivo vengono presentate anche delle prove a media non nulla. La varianza della distribuzione normale è diversa per ogni variabile di ingresso:

$$\sigma_\rho^2(k) = k_\rho |\Delta\rho(k)| \tag{9.2}$$

$$\sigma_\theta^2(k) = k_\theta^{\Delta\rho} |\Delta\rho(k)| + k_\theta^{\Delta\theta} |\Delta\theta(k)| \tag{9.3}$$

L'equazione 9.2 rappresenta la varianza dell'errore compiuto nella valutazione di speed dal sensore odometrico al passo k. Il parametro dipende essenzialmente dallo spostamento traslazionale ($\Delta\rho(k)$) del veicolo nel passo k. L'equazione 9.3 rappresenta la varianza dell'errore compiuto nella valutazione di jog dal sensore odometrico al passo k. In questo caso il parametro dipende dallo spostamento roto-traslazionale (rotatorio $\Delta\theta(k)$, traslazionale $\Delta\rho(k)$) del veicolo all'istante k. Gli spostamenti roto-traslazionali sono proporzionati dalle costanti k_ρ, $k_\theta^{\Delta\rho}$ e $k_\theta^{\Delta\theta}$.

Per le simulazioni off-line l'entità di queste costanti è stata maggiorata, rispetto le costanti utilizzate in [7], per testare in condizioni peggiori gli algortmi. I valori adottati sono:

$k_\rho =$ 0.0002;

$k_\theta^{\Delta\rho} =$ 0.001;

$k_\theta^{\Delta\theta} = 0.1$.

Per ulteriori informazioni sull'errore di odometria si rimanda la trattazione in appendice.

9.1.1 Simulazioni sulla mappa denominata sala macchine

Questa mappa descrive la sala macchine presente nei sotterranei del dipartimento di elettronica, elettrotecnica ed informatica. La sala rappresenta un tipico ambiente di carattere industriale: ampio, spazioso, con ostacoli sparsi.

Come percorso campione si è scelto un percorso che attraversasse la maggior parte della stanza. Una volta fissato sono state effettuate delle prove ripetute. Si riportano di seguito le tracce ottenute in una di queste prove. Le traccie si riferiscono a simulazioni a precisione massima (quindi 64 bit 9.1) e a precisione limitata (16 bit 9.2, 8 bit 9.3). In supporto alla rappresentazione delle traccie sulla mappa, sono stati riportati anche i diagrammi dell'andamento dell'errore. In particolare, si considera la distanza euclidea tra la posizione del baricentro del robot e la posizione presunta dello stesso. Si è scelto di utilizzare questo metodo per interpretare i risultati ottenuti per avere subito, con due soli parametri, una stima dell'errore commesso dagli algoritmi di fusione (e non) sulla posizione. La norma euclidea è stata calcolata come segue (eq.9.4):

$$e(k) = \|X_{reale} - X_{presunta}\|$$

$$= \sqrt{(x_{reale} - x_{presunta})^2 + (y_{reale} - y_{presunta})^2 + (\theta_{reale} - \theta_{presunta})^2} \tag{9.4}$$

con X stato del sistema[1]. Il comportamento medio degli algoritmi nelle varie prove è stato riassunto nelle tabelle 9.1, 9.2, 9.3, 9.4, 9.5, 9.6.

A precisione massima, il comportamento medio del filtro per fusione sensoriale che sfrutta il filtro fast SQR-RLS presenta caratteristiche di funzionamento simili e addirittura in alcuni casi superiori al filtro di Kalman. Limitando il numero di bit a disposizione per la computazione l'ago della bilancia pende inesorabilmente verso l'algoritmo fast SQR-RLS SLAM. Infatti la media dell'errore e la varianza rimangono entro limiti accettabili.

Il comportamento degl'altri due filtri basati su RLS tuttavia non è altrettanto efficiente. Il filtro SQR-RLS dimostra una forte capacità di commettere

[1] In realtà lo stato del sistema comprende anche la posizione dei landmark. Si è definito $X = (x, y, \theta)$ come variabile di stato per brevità di notazione, nella speranza che ciò non tragga in inganno.

errori d'associazione dati. Il filtro RLS invece assume caratteristiche intermedie tra l'odometria ed il filtro di Kalman. Questi due comportamenti diversi sono da ricondurre alle proprietà di convergenza dei vari algoritmi. L'algoritmo RLS converge "dolcemente", ciò è visibile dalla figura 9.7. L'errore medio diminuisce con il numero di iterazioni, questo comporta che l'intervento del filtro è graduale, quindi non è troppo invasivo per il sistema. Grazie a questa "delicatezza", nell'intervento, il filtro è in grado di convergere senza commettere gravi errori nell'associazione dati. Contrariamente l'algoritmo SQR-RLS ha capacità di convergenza più rapide, a scapito tuttavia di essere caratterizzato da oscillazioni attorno al punto di convergenza molto ampie. Il tutto è visibile nelle figure 9.8 e 9.9 che rappresentano l'errore di posizionamento in aritmetica a 16 bit, dove questo problema risulta accentuato. L'algoritmo fast SQR-RLS invece ha il pregio, nella sua implementazione, di essere tarabile attraverso variabili interne. Questa caratteristica consente di controllare l'azione di filtraggio, calibrando l'intervento dell'algoritmo. La taratura cambia a seconda della precisione di lavoro. Diminuendo il numero di bit a disposizione, diminuisce il peso dei coefficienti. In questo modo l'intervento del filtro non apporta cambiamenti troppo rapidi che possono condurre il sistema in instabilità. Come ulteriore considerazione si tenga presente che i numeri rappresentabili, essendo più piccoli, necessitano di un numero di bit minore, e l'errore di troncamento interessa quantità più piccole.

9.1.2 Considerazioni riassuntive sulla convergenza degli algoritmi RLS

Nel capitolo 8 si è visto che gli algoritmi di fusione sensoriale basati su RLS evolvono nel punto. È fondamentale studiare in quanti cicli si ha convergenza. Per esempio, nella figura 9.5 si noti che l'algoritmo fast SQR-RLS non raggiunge la convergenza con soli 8 cicli. Tuttavia utilizzando 10 cicli l'algoritmo converge subito al valore ottimale. Una caratteristica di fast SQR-RLS è di convergere immediatamente al valore ottimale non appena il numero di cicli supera una certa soglia. Nelle simulazioni effettuate tale soglia risulta variabile e compresa tra 8 e 12 cicli. Mentre nel diagramma di figura 9.7 sembra che RLS necessiti di un numero elevato di iterazioni per raggiungere la convergenza. L'algoritmo SQR-RLS converge entro un numero di iterazioni ristretto e presenta oscillazioni molto ampie attorno al punto di equilibrio. L'effetto di questo comportamento è visibile nel diagramma di figura 9.8. Durante il compimento di questo lavoro la convergenza degli algoritmi è stata materia di studio per molto tempo. I tentativi di limitare gli interventi di SQR-RLS hanno portato sempre esiti negativi.

N° Prova	**Algoritmo**	**Media errore** $[m]$	**Varianza errore** $[m^2]$
1	fast SQR-RLS	0.598	0.487
	SQR-RLS	3.218	5.9
	RLS	2.257	3.06
	Kalman	1.195	0.601
	Odometria	2.570	4.35
2	fast SQR-RLS	0.52	0.041
	SQR-RLS	3.135	3.372
	RLS	1.988	1.670
	Kalman	1.22	0.464
	Odometria	2.484	2.432
3	fast SQR-RLS	0.511	0.03
	SQR-RLS	1.256	0.26
	RLS	2.21	3.262
	Kalman	0.558	0.125
	Odometria	0.476	0.081
4	fast SQR-RLS	0.594	0.056
	SQR-RLS	1.072	0.625
	RLS	4.073	4.398
	Kalman	0.517	0.111
	Odometria	1.133	0.622
5	fast SQR-RLS	0.799	0.312
	SQR-RLS	2.682	8.339
	RLS	2.759	6.418
	Kalman	1.93	6.768
	Odometria	3.15	7.719

Tabella 9.1: Media e varianza dell'errore, sulla posizione presunta. Simulazioni effettuate con 20 cicli nel punto per gli algoritmi RLS. Risultati conseguiti con aritmetica a precione massima: 64bit.

N° Prova	**Algoritmo**	**Media errore** $[m]$	**Varianza errore** $[m^2]$
6	fast SQR-RLS	0.435	0.033
	SQR-RLS	3.975	6.24
	RLS	2.482	3.225
	Kalman	1.56	1.041
	Odometria	2.952	4.13
7	fast SQR-RLS	0.496	0.043
	SQR-RLS	3.231	3.61
	RLS	1.99	1.702
	Kalman	1.136	0.404
	Odometria	2.479	2.436
8	fast SQR-RLS	1.874	0.065
	SQR-RLS	1.533	0.56
	RLS	1.108	0.439
	Kalman	1.149	0.626
	Odometria	1.874	1.33
9	fast SQR-RLS	0.513	0.053
	SQR-RLS	4.247	9.323
	RLS	2.94	6.259
	Kalman	2.588	3.721
	Odometria	3.579	7.527
10	fast SQR-RLS	0.894	0.081
	SQR-RLS	1.901	0.558
	RLS	2.386	1.033
	Kalman	1.664	0.487
	Odometria	2.172	0.933

Tabella 9.2: Media e varianza dell'errore, sulla posizione presunta. Simulazioni effettuate con 20 cicli nel punto per gli algoritmi RLS. Risultati conseguiti con aritmetica a precione massima: 64bit.

N° Prova	**Algoritmo**	**Media errore** $[m]$	**Varianza errore** $[m^2]$
1	fast SQR-RLS	0.621	0.058
	SQR-RLS	3.31	6.11
	RLS	2.258	3.07
	Kalman	2.187	5.183
	Odometria	2.570	4.35
2	fast SQR-RLS	0.511	0.041
	SQR-RLS	3.138	3.378
	RLS	1.984	1.664
	Kalman	1.09	0.307
	Odometria	2.484	2.432
3	fast SQR-RLS	0.513	0.028
	SQR-RLS	1.369	0.287
	RLS	1.152	0.231
	Kalman	4.741	51.741
	Odometria	0.476	0.081
4	fast SQR-RLS	0.61	0.062
	SQR-RLS	1.338	0.88
	RLS	4.157	4.626
	Kalman	1.598	10.96
	Odometria	1.133	0.622
5	fast SQR-RLS	0.798	0.333
	SQR-RLS	4.066	9.086
	RLS	2.76	6.418
	Kalman	1.484	1.956
	Odometria	3.15	7.719

Tabella 9.3: Media e varianza dell'errore, sulla posizione presunta. Simulazioni effettuate con 20 cicli nel punto per gli algoritmi RLS. Risultati conseguiti con aritmetica a precione limitata: 16bit.

N° Prova	**Algoritmo**	**Media errore** $[m]$	**Varianza errore** $[m^2]$
6	fast SQR-RLS	0.457	0.036
	SQR-RLS	3.966	6.212
	RLS	2.47	3.22
	Kalman	1.532	0.979
	Odometria	2.952	4.13
7	fast SQR-RLS	0.524	0.046
	SQR-RLS	3.231	3.609
	RLS	1.985	1.697
	Kalman	1.349	0.772
	Odometria	2.479	2.436
8	fast SQR-RLS	0.626	0.057
	SQR-RLS	1.548	0.539
	RLS	1.111	0.436
	Kalman	2.097	4.318
	Odometria	1.874	1.33
9	fast SQR-RLS	0.524	0.062
	SQR-RLS	4.301	9.204
	RLS	3.741	6.891
	Kalman	11.912	154.35
	Odometria	3.579	7.527
10	fast SQR-RLS	0.939	0.089
	SQR-RLS	1.924	0.552
	RLS	2.389	1.038
	Kalman	2.27	2.115
	Odometria	2.172	0.933

Tabella 9.4: Media e varianza dell'errore, sulla posizione presunta. Simulazioni effettuate con 20 cicli nel punto per gli algoritmi RLS. Risultati conseguiti con aritmetica a precione limitata: 16bit.

N° Prova	**Algoritmo**	**Media errore** $[m]$	**Varianza errore** $[m^2]$
1	fast SQR-RLS	0.656	0.058
	SQR-RLS	2.47	1.04
	RLS	3.096	3.814
	Kalman	10.449	60.632
	Odometria	2.570	4.35
2	fast SQR-RLS	1.31	1.24
	SQR-RLS	1.541	0.261
	RLS	12.412	77.349
	Kalman	7.697	19.158
	Odometria	2.484	2.432
3	fast SQR-RLS	0.887	0.62
	SQR-RLS	3.715	2.66
	RLS	1.361	0.341
	Kalman	8.34	23.807
	Odometria	0.476	0.081
4	fast SQR-RLS	0.612	0.699
	SQR-RLS	3.03	0.996
	RLS	2.74	0.925
	Kalman	8.431	21.53
	Odometria	1.133	0.622
5	fast SQR-RLS	2.748	7.577
	SQR-RLS	12.029	68.704
	RLS	10.776	43.43
	Kalman	8.961	17.169
	Odometria	3.15	7.719

Tabella 9.5: Media e varianza dell'errore, sulla posizione presunta. Simulazioni effettuate con 20 cicli nel punto per gli algoritmi RLS. Risultati conseguiti con aritmetica a precione limitata: 8bit.

N° Prova	**Algoritmo**	**Media errore** $[m]$	**Varianza errore** $[m^2]$
6	fast SQR-RLS	1.875	2.81
	SQR-RLS	1.738	0.869
	RLS	2.17	2.748
	Kalman	6.909	21.806
	Odometria	2.952	4.13
7	fast SQR-RLS	1.4	1.345
	SQR-RLS	1.612	0.286
	RLS	12.389	77.713
	Kalman	4.191	10.037
	Odometria	2.479	2.436
8	fast SQR-RLS	0.947	0.255
	SQR-RLS	2.271	0.538
	RLS	1.123	0.463
	Kalman	15.99	136.31
	Odometria	1.874	1.33
9	fast SQR-RLS	2.343	3.084
	SQR-RLS	3.022	3.685
	RLS	9.922	36.641
	Kalman	9.737	24.898
	Odometria	3.579	7.527
10	fast SQR-RLS	2.259	1.386
	SQR-RLS	5.038	5.29
	RLS	3.046	2.624
	Kalman	8.801	17.555
	Odometria	2.172	0.933

Tabella 9.6: Media e varianza dell'errore, sulla posizione presunta. Simulazioni effettuate con 20 cicli nel punto per gli algoritmi RLS. Risultati conseguiti con aritmetica a precione limitata: 8bit.

L'algoritmo RLS ha dimostrato di mantere un'efficienza nel suo stato originario. Le condizioni iniziali delle variabili interne di fast SQR-RLS sono state ricavate empiricamente come segue:

γ =1;

α =10;

Z ogni elemento del vettore vale 0.001;

Infine il vettore D ha valori diversi a seconda della precisione di calcolo adottata:

	64 bit	16 bit	8 bit
Elementi vettore D	0.08	0.06	0.04

9.1.3 Casi particolari

Il filtro fast SQR-RLS-SLAM ha dimostrato caratteristiche migliori del filtro EKF-SLAM.

Simulazioni con errore a media non nulla

Si sono sperimentati gli algoritmi anche con un errore di tipo gaussiano a media non nulla. Mediamente si verifica che l'algoritmo fast SQR-RLS ha un comportamento migliore dell'algoritmo basato su Kalman. Una prova è visibile nelle figure 9.10 e 9.11.

Simulazioni con errori non gaussiani

Sono state condotte simulazioni con tipologie d'errore non gaussiane. L'errore introdotto è di tipo casuale, di entità smisurata. Lo scopo è quello di ricreare artificialmente la condizione di errore dovuta a forti perturbazioni del veicolo. Per esempio a seguito di urto con oggetti non captati dai sensori, da buche nel pavimento, etc.

Nelle figure 9.12, 9.13, 9.14 e 9.15 si vede chiaramente che la capacità d'assorbimento della perturbazione di fast-SQR-RLS è superiore all'altro algoritmo.

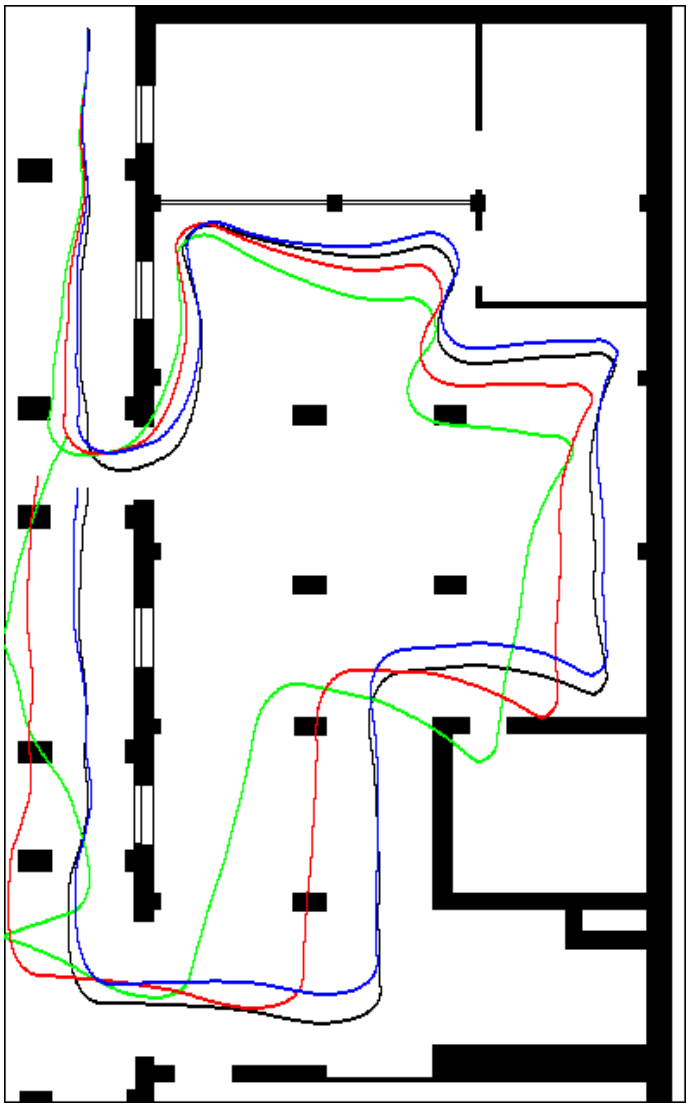

Figura 9.1: Simulazione nella sala macchine. Precisione a 64 bit. Le traccie rappresentano: **nera** percorso reale del robot, **rossa** traccia ricavata dalla fusione con filtro di Kalman, **verde** traccia ricavata dall'odometria, **blu** traccia ricavata dall'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

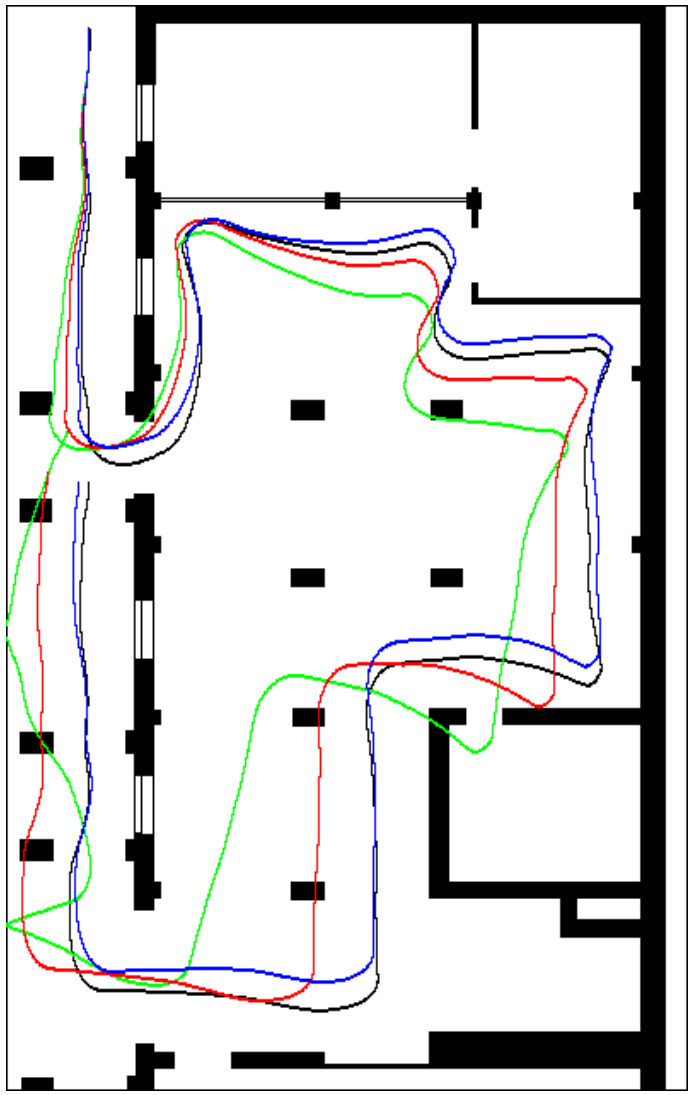

Figura 9.2: Simulazione nella sala macchine. Precisione a 16 bit. Le traccie rappresentano: **nera** percorso reale del robot, **rossa** traccia ricavata dalla fusione con filtro di Kalman, **verde** traccia ricavata dall'odometria, **blu** traccia ricavata dall'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

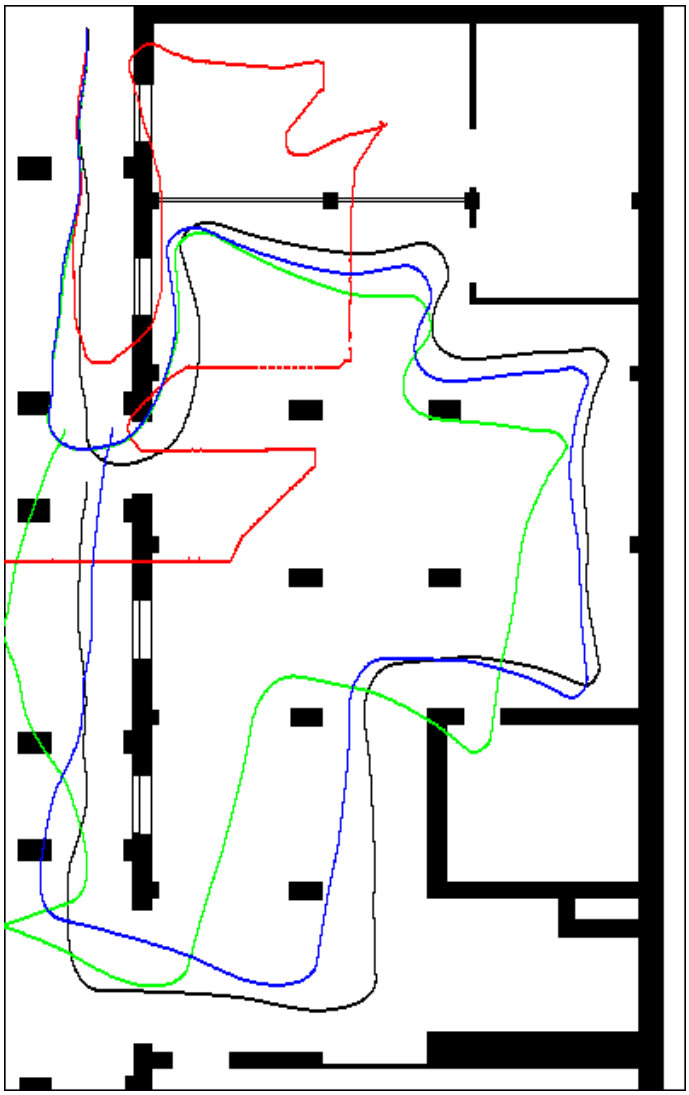

Figura 9.3: Simulazione nella sala macchine. Precisione a 8 bit. Le traccie rappresentano: **nera** percorso reale del robot, **rossa** traccia ricavata dalla fusione con filtro di Kalman, **verde** traccia ricavata dall'odometria, **blu** traccia ricavata dall'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

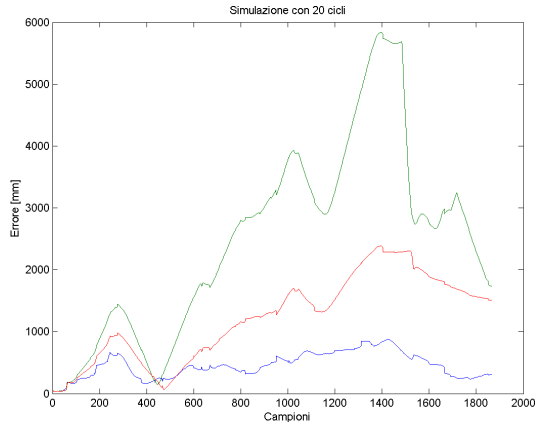

Figura 9.4: Simulazione nella sala macchine. Andamento della distanza euclidea tra posizione reale e presunta. Le curve rappresentano: **rossa** errore prodotto dal metodo dalla fusione con il filtro di Kalman, **verde** errore prodotto col metodo dell'odometria, **blu** errore prodotto servendosi dell'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

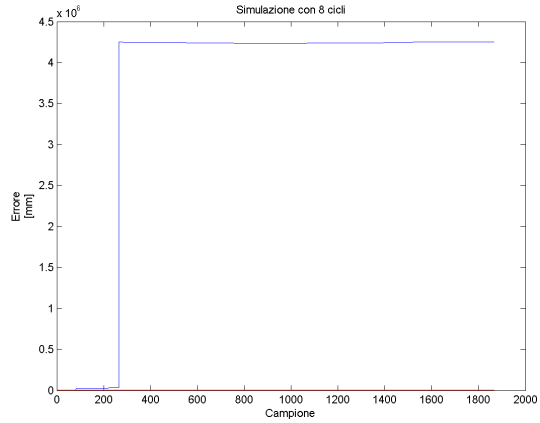

Figura 9.5: Simulazione nella sala macchine. Andamento della distanza euclidea tra posizione reale e presunta. Le curve rappresentano: **rossa** errore prodotto dal metodo dalla fusione con il filtro di Kalman, **verde** errore prodotto col metodo dell'odometria, **blu** errore prodotto servendosi dell'algoritmo di fusione basato su fast RLS-SQR, 8 cicli nel punto.

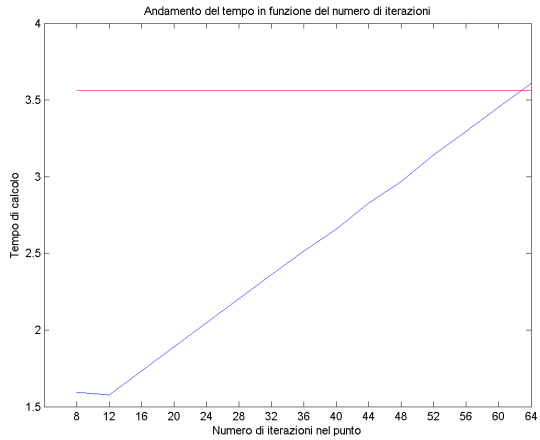

Figura 9.6: Simulazione nella sala macchine. Tempo di calcolo per l'esecuzione dei vari algoritmi, in funzione del numero di iterazioni nel punto. Le curve rappresentano: **rossa** tempo di calcolo per la fusione con il filtro di Kalman, **blu** tempo di calcolo per l'algoritmo di fusione basato su fast RLS-SQR.

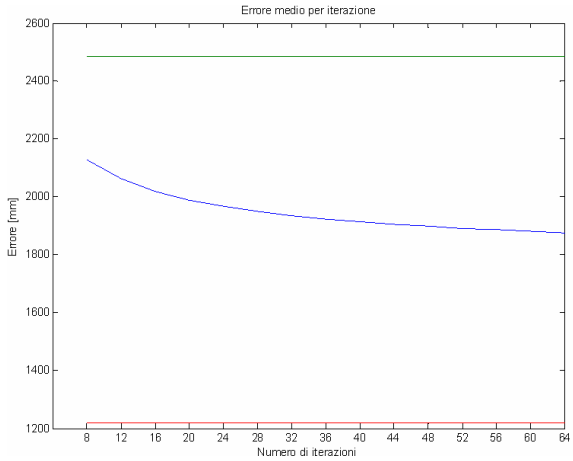

Figura 9.7: Simulazione nella sala macchine. Errore medio in funzione del numero di iterazioni nel punto. Le curve rappresentano: **rossa** algoritmo EKF-SLAM, **verde** algoritmo odometrico, **blu** algoritmo RLS-SLAM. Simulazione a 64 bit.

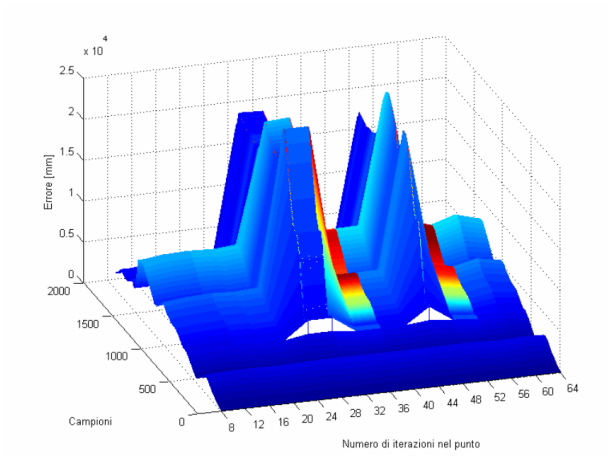

Figura 9.8: Errore di posizionamento in funzione del numero di iterazioni nel punto e del numero di campione, relativo all'algoritmo SQR-RLS SLAM. Simulazione a 16 bit.

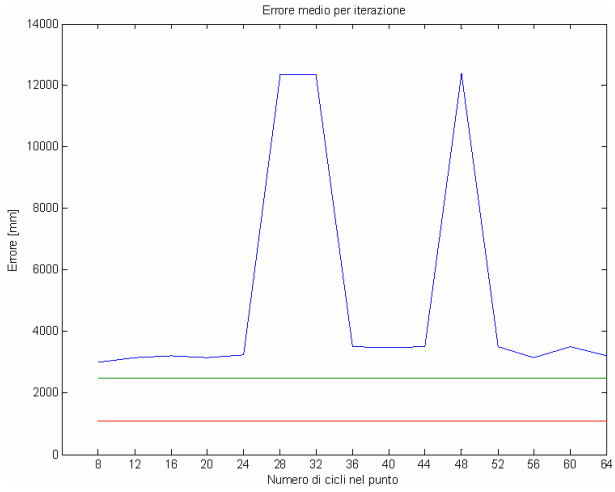

Figura 9.9: Errore di posizionamento medio in funzione del numero di iterazioni nel punto e del numero di campione, relativo all'algoritmo SQR-RLS SLAM. Simulazione a 16 bit.

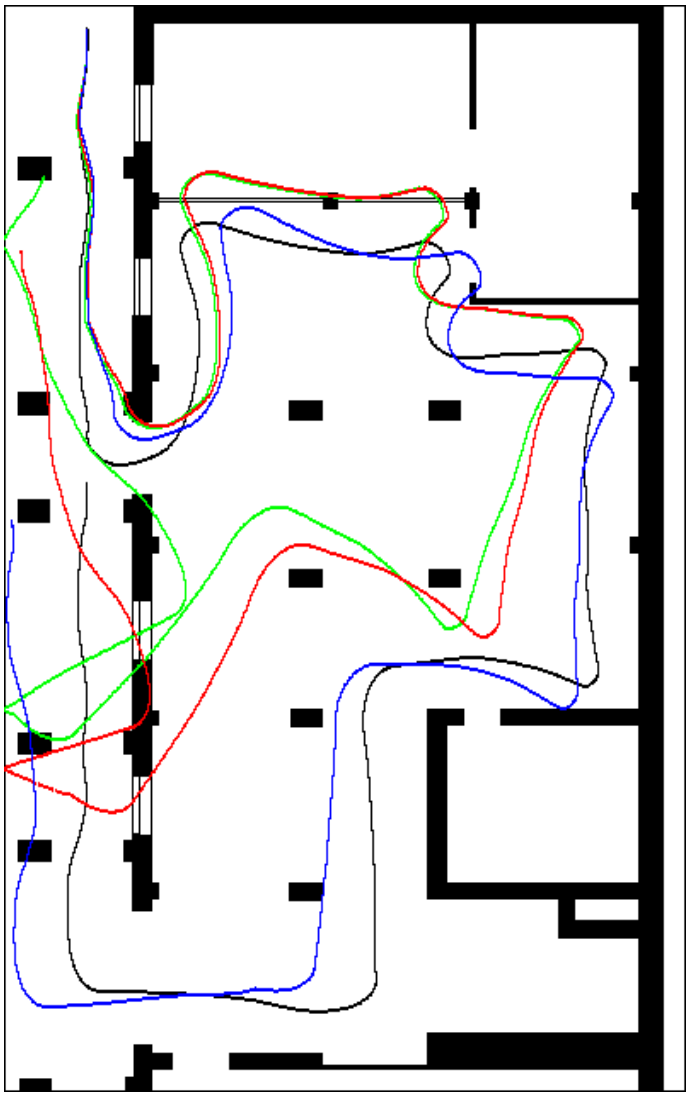

Figura 9.10: Simulazione con errore gaussiano a media non nulla. Simulazione nella sala macchine. Precisione a 64 bit. Le traccie rappresentano: **nera** percorso reale del robot, **rossa** traccia ricavata dalla fusione con filtro di Kalman, **verde** traccia ricavata dall'odometria, **blu** traccia ricavata dall'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

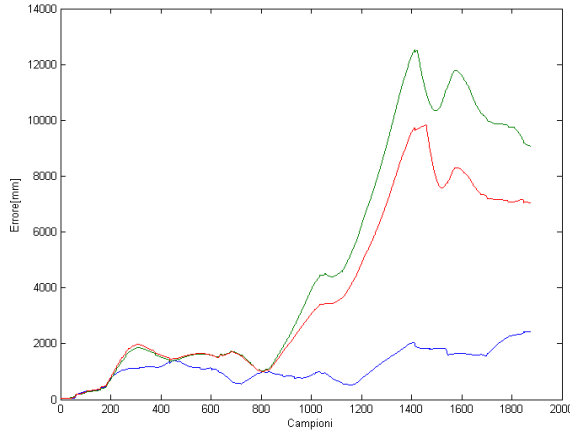

Figura 9.11: Simulazione con errore gaussiano a media non nulla. Precisione a 64 bit. Le curve rappresentano: **rossa** errore nella fusione con filtro di Kalman, **verde** errore dell'odometria, **blu** errore nell'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

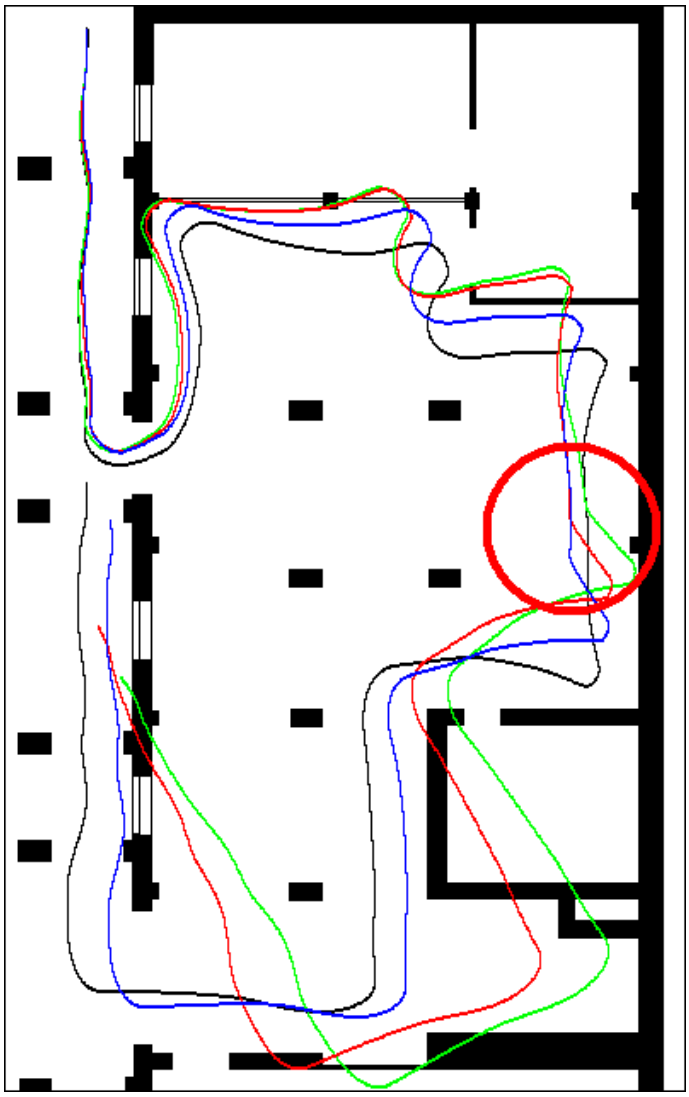

Figura 9.12: Simulazione con errore non gaussiano casuale di forte entità. Precisione a 64 bit. Le traccie rappresentano: **nera** percorso reale del robot, **rossa** traccia ricavata dalla fusione con filtro di Kalman, **verde** traccia ricavata dall'odometria, **blu** traccia ricavata dall'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

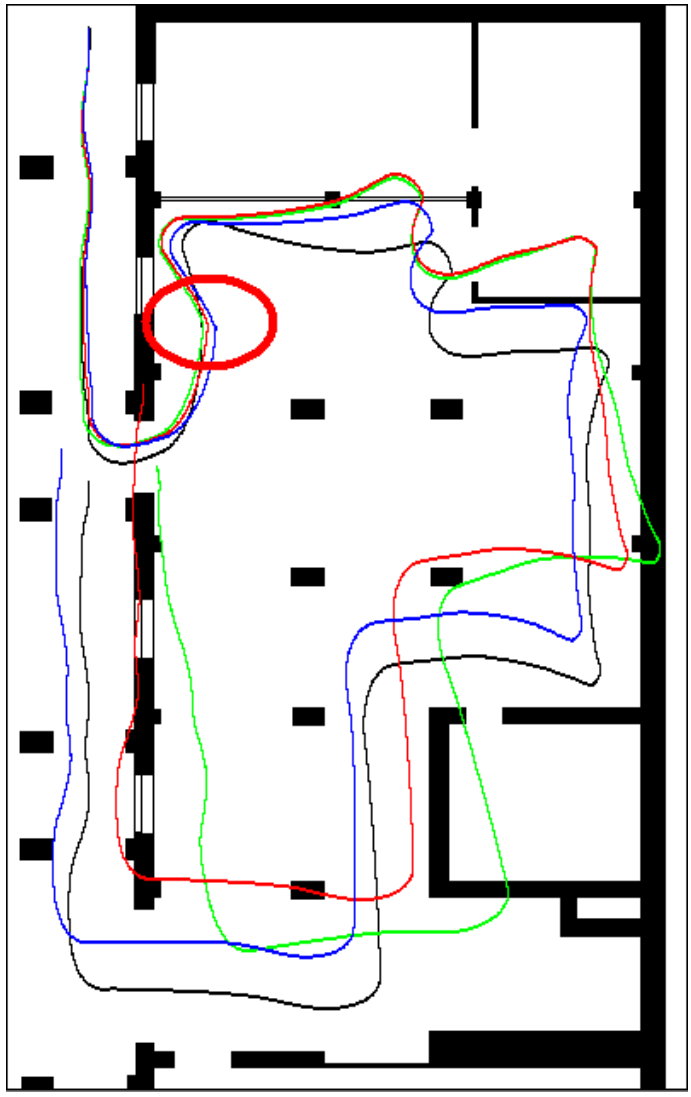

Figura 9.13: Simulazione con errore non gaussiano casuale di forte entità. Precisione a 64 bit. Le traccie rappresentano: **nera** percorso reale del robot, **rossa** traccia ricavata dalla fusione con filtro di Kalman, **verde** traccia ricavata dall'odometria, **blu** traccia ricavata dall'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

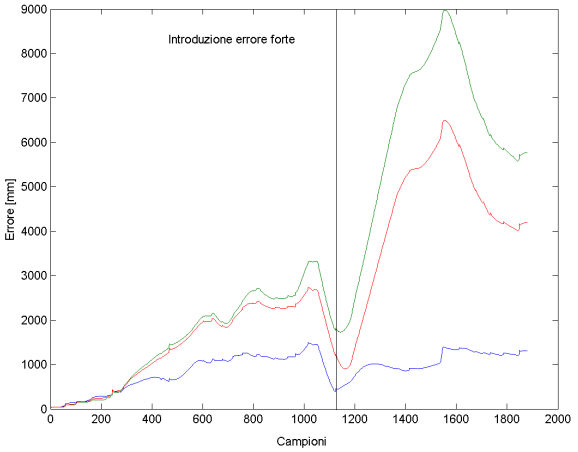

Figura 9.14: Simulazione con errore non gaussiano casuale di forte entità. Precisione a 64 bit. Le curve rappresentano: **rossa** errore nella fusione con filtro di Kalman, **verde** errore dell'odometria, **blu** errore nell'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

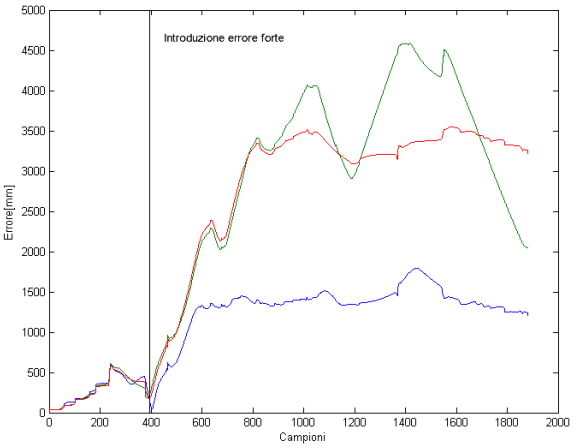

Figura 9.15: Simulazione con errore non gaussiano casuale di forte entità. Precisione a 64 bit. Le curve rappresentano: **rossa** errore nella fusione con filtro di Kalman, **verde** errore dell'odometria, **blu** errore nell'algoritmo di fusione basato su fast RLS-SQR, 20 cicli nel punto.

9.2 Simulazioni in line

Le simulazioni in line rispecchiano lo schema di figura 9.16.

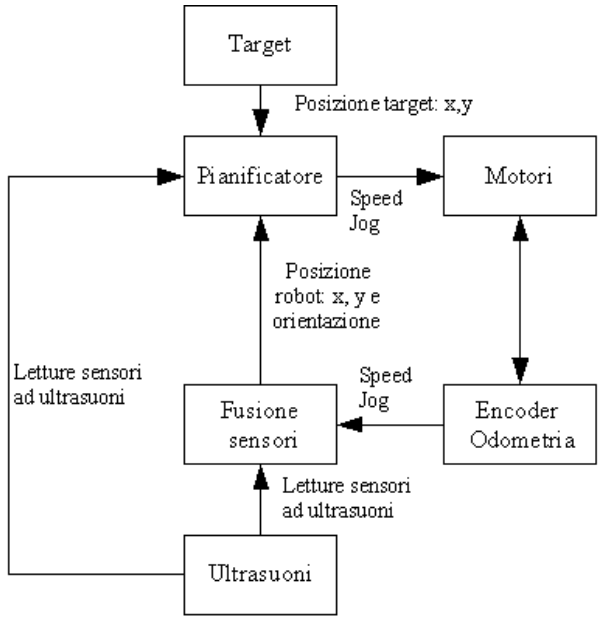

Figura 9.16: Schema delle simulazioni in line

Lo scopo è quello di ricreare la reale condizione di lavoro, ossia quando la pianificazione della traiettoria è tracciata in base alla posizione presunta fornita dall'algoritmo di fusione sensoriale.

Si è manipolato a questo scopo il simulatore a disposizione, inserendo nei moduli `pathplan` e `DR` l'opzione fusione nella navigazione. Il caso, anche se analogo al precedente, subito ha rilevato alcune divesità sostanziali:

- un errore nella valutazione della posizione induce una catena di situazioni che concorrono all'aumento dell'errore. Quando la posizione presunta è lontana da quella reale, navigando in base all'informazione sbagliata il robot si conduce fuori strada. Nel nuovo ambiente, la presenza di ostacoli non previsti lo costringe ad evoluzioni complesse per evitare quest'ultimi.

Inevitabilmente, durante le operazioni di aggiramento subentrano molti altri errori, rendendo difficile il ripristino della normalità;

- per permettere che la fusione abbia corso, si deve costringere il robot a seguire durante la navigazione le pareti. Questo garantisce la presenza di *landmark*. Se il robot possiede un'informazione sbagliata della propria posizione è possibile che abbandoni la tecnica di navigazione *wall following*, portandosi, per esempio, in centro della stanza. In questa condizione vengono a mancare le *feature* necessarie per la fusione sensoriale.

9.2.1 Simulazioni nella mappa di prova standard: il quadrato

In letteratura [14] i problemi di localizzazione di robot mobili vengono studiati in mappe non proprio reali. In genere vengono adottati percorsi chiusi quadrati. Data la complessità del problema si è deciso di adottare per le prove "in-line" una mappa che descrive un corridoio quadrato di 100m di lunghezza. In questa fase d'analisi non è possibile confrontare direttamente, prova per prova, i vari algoritmi fra loro. Infatti per la navigazione viene utilizzata la posizione presunta in uscita da un algoritmo. Per confrontare i vari algoritmi direttamente fra loro si ricorre al metodo di analisi del comportamento medio.

Si sono condotte due serie di simulazioni sulla stessa mappa, una in senso orario ed una in senso antiorario. In questa fase di test non sono considerati gli algoritmi RLS e SQR-RLS. Sono stati considerati finora perché sviluppati, nel corso della ricerca, col fine di comprendere caratteristiche del filtro RLS utilizzato per fusione sensoriale. Nel capitolo precedente sono stati analizzati e paragonati fra loro tutti gli algoritmi, il risultato favorevole pende senz'altro dalla parte degli algoritmi fast SQR-RLS-SLAM e EKF-SLAM. Quindi è ragionevole continuare lo studio concentrandosi solo su questi ultimi.

L'errore utilizzato durante queste prove è di natura identica a quello utilizzato nelle prove "off-line". Le costanti sono state ridotte, i valori utilizzati sono i seguenti:

$k_\rho =$ 0.0002;

$k_\theta^{\Delta\rho} =$ 0.00001;

$k_\theta^{\Delta\theta} =$ 0.01.

Simulazione in senso orario

Sfruttando la modalità `Activate from file` si possono eseguire prove ripetute nelle stesse condizioni. Nel file `target.txt` vengono inserite,in sequenza, le coordinate dei *target* da raggiungere nella mappa. Quando il robot è in prossimità del `target` questi viene aggiornato. In questa simulazione quindi la sequenza di target è impostata in modo da far percorre al robot la mappa in senso orario.

Le coordinate sono espresse in pixel, corrispondenti a punti della mappa. Il contenuto del file `target.txt` per questa prova è il seguente:

```
60 60
530 54
533 542
60 540
60 60
```

Sono state condotte dieci prove per ogni algoritmo. I risultati ottenuti sono stati riportati nella tabella 9.7. Le simulazioni sono state condotte anche con precisione limitata, utilizzando solo 8 bit di mantissa 9.8.

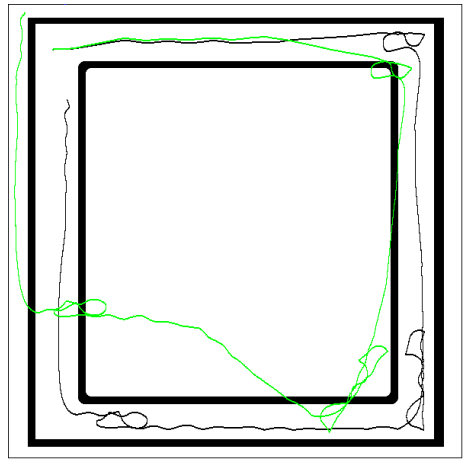

Figura 9.17: Traccia percorso reale (nera) e traccia percorso presunto (verde). Algoritmo odometrico.

Dalle prove effettuate si vede chiaramente che applicare al sistema di navigazione la capacità di fusione delle informazioni rilevate durante la marcia

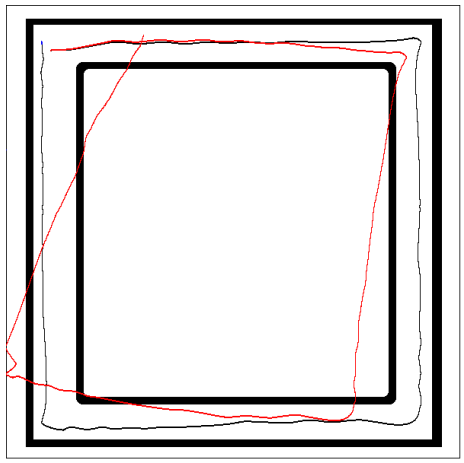

Figura 9.18: Traccia percorso reale (nera) e traccia percorso presunto (rossa). Fusione con filtro di Kalman.

diminuisce mediamente l'errore sulla determinazione della posizione del robot. Scendendo di precisione gli algoritmi, preposti alla fusione presentati, accumulano mediamente un errore di entità maggiore. L'algoritmo basato su RLS tuttavia mantiene valori medi al di sotto dell'odometria. Quindi in queste condizioni risulta ancora vantaggioso utilizzarlo per compiere fusione sensoriale. L'algoritmo basato su Kalman invece è estremamente instabile numericamente e non è più utilizzabile.

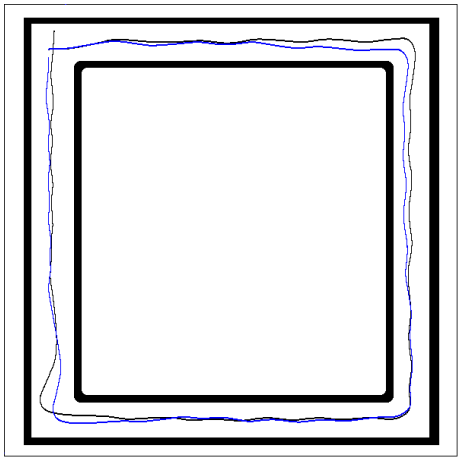

Figura 9.19: Traccia percorso reale (nera) e traccia percorso presunto (blu). Fusione con filtro RLS.

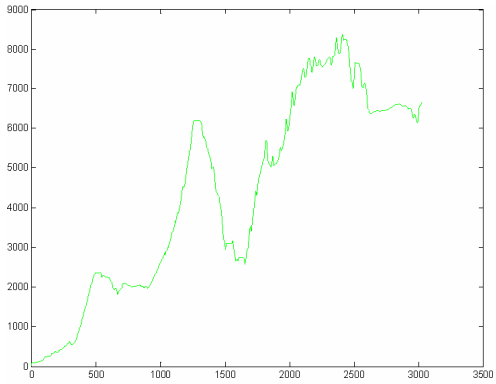

Figura 9.20: Andamento della distanza euclidea tra posizione reale e posizione presunta. Algoritmo odometrico.

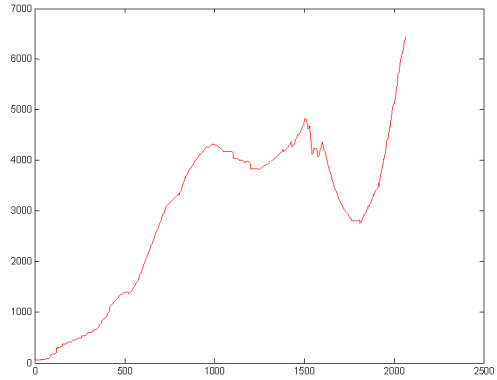

Figura 9.21: Andamento della distanza euclidea tra posizione reale e posizione presunta. Fusione con filtro di Kalman.

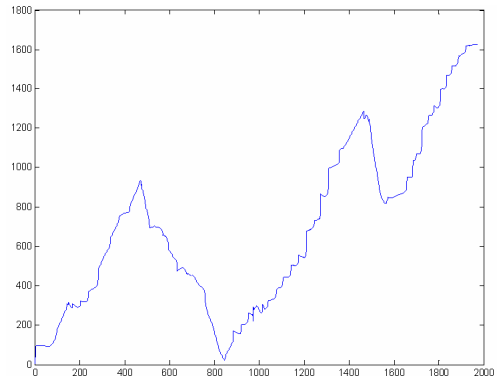

Figura 9.22: Andamento della distanza euclidea tra posizione reale e posizione presunta. Fusione con filtro RLS.

Numero simulazione	Odometria		Kalman		RLS	
	$\mu[m]$	$\sigma^2[m^2]$	$\mu[m]$	$\sigma^2[m^2]$	$\mu[m]$	$\sigma^2[m^2]$
1	7.787	22.832	2.340	1.505	1.755	0.228
2	3.665	4.010	6.182	21.711	3.544	2.364
3	3.999	4.301	3.585	3.906	0.688	0.184
4	2.305	1.366	2.917	2.468	1.520	0.685
5	2.928	4.270	4.206	6.812	0.716	0.110
6	4.319	6.230	5.946	6.562	1.044	0.254
7	7.024	15.888	5.316	16.846	1.275	0.571
8	3.818	4.172	1.208	1.032	1.600	0.399
9	2.282	2.378	4.080	7.019	1.058	0.421
10	3.912	3.138	13.933	52.750	0.913	0.201
Errore medio	4204	6.858	4971	12.061	1411	0.542

Tabella 9.7: Errore di posizionamento medio, relativo alle simulazioni in-line. Mappa del corridoio quadrato, senso orario.

Numero simulazione	Kalman		RLS	
	$\mu[m]$	$\sigma^2[m^2]$	$\mu[m]$	$\sigma^2[m^2]$
1	15.375	55.401	2.807	2.839
2	13.048	38.042	3.189	4.634
3	14.251	58.614	3.652	4.038
4	12.190	50.275	5.527	11.861
5	15.351	68.162	3.831	8.180
6	20.902	112.033	3.370	4.374
7	20.183	82.178	3.163	6.316
8	19.481	74.201	4.820	15.585
9	15.086	44.503	4.932	17.420
10	20.137	80.276	5.522	19.479
Errore medio	16.600	66.365	4.081	9.472

Tabella 9.8: Errore di posizionamento medio, relativo alle simulazioni in-line. Mappa del corridoio quadrato, senso orario, 8 bit di mantissa

Simulazione in senso antiorario

Studiando gli algoritmi in un percorso chiuso è "obbligatorio" [14] verificare il loro comportamento in entrambi i sensi di marcia. Si sono condotte quindi delle prove anche percorrendo il corridoio in senso antiorario. Come nel caso precedente si analizza il comportamento medio del sistema, tabella 9.9 e tabella 9.10.

Numero simulazione	Odometria		Kalman		RLS	
	$\mu[m]$	$\sigma^2[m^2]$	$\mu[m]$	$\sigma^2[m^2]$	$\mu[m]$	$\sigma^2[m^2]$
1	3.594	2.907	2.885	2.569	1.886	0.899
2	2.970	1.651	1.576	0.509	1.218	0.296
3	3.643	5.515	3.342	2.428	1.656	0.343
4	4.257	5.648	2.096	1.767	1.217	0.201
5	7.524	32.198	4.397	7.256	1.360	0.271
6	1.081	0.474	3.777	1.570	1.572	0.369
7	5.071	5.529	3.132	5.656	2.725	1.575
8	5.178	7.186	9.239	62.367	1.452	0.216
9	9.353	27.933	9.304	21.325	0.917	0.093
10	6.960	15.321	10.098	29.747	1.454	0.261
Valori medi	4.963	10.436	4.984	13.519	1.546	0.452

Tabella 9.9: Errore di posizionamento medio, relativo alle simulazioni in-line. Mappa del corridoio quadrato, senso antiorario.

Le prove in senso antiorario ripropongono sostanzialmente i risultati ottenuti in senso orario.

Numero simulazione	Kalman		RLS	
	$\mu[m]$	$\sigma^2[m^2]$	$\mu[m]$	$\sigma^2[m^2]$
1	21.903	76.923	3.020	1.556
2	26.898	86.845	2.897	3.198
3	14.780	99.632	2.984	1.641
4	29.317	105.331	4.434	7.942
5	20.945	110.497	2.825	1.680
6	22.123	111.010	3.505	3.191
7	23.201	56.642	4.297	7.345
8	25.943	104.728	3.300	1.512
9	11.158	38.966	3.522	2.271
10	18.577	99.721	4.851	6.386
Valori medi	21.484	89.025	3.564	3.672

Tabella 9.10: Errore di posizionamento medio, relativo alle simulazioni in-line. Mappa del corridoio quadrato, senso antiorario, 8 bit di mantissa

Capitolo 10

Il calcolo delle operazioni

10.1 Operazioni dell'algoritmo EKF-SLAM

Le fasi essenziali dell'algoritmo EKF-SLAM sono:

- Linearizzazione nel punto
- Predizione
- Inizializzazione delle feature
- Controllo *feature*
- Osservazione
- Aggiornamento

N rappresenta il numero di stati del sistema, M il numero di stati osservabili. Nel caso in questione $N = 6$ e $M = 3$.

10.1.1 Linearizzazione nel punto

Il calcolo di A_k comporta al massimo 4 moltiplicazioni, 2 divisioni e 4 coseni/seni:

$$A(k) = \begin{pmatrix} 1 & 0 & \frac{s}{\omega}[\cos(\omega\Delta t + \theta) - \cos\theta] & 0 & 0 & 0 \\ 0 & 1 & \frac{s}{\omega}[\sin(\omega\Delta t + \theta) - \sin\theta] & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{10.1}$$

Per il calcolo di H_k si implementano al massimo 2 moltiplicazioni e 2 coseni/seni per sensore.

10.1.2 Predizione

Equazione	×	÷	∨	cos/sin
$\begin{pmatrix} X_\nu^-(k) \\ \hat{X}_m^-(k) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu^+(k-1), U(k)) \\ \hat{X}_m^+(k-1) \end{pmatrix}$	4	2	-	4
$P^-(k) = \nabla_x f(k) P^+(k-1) \nabla_x f^T(k) + Q(k)$	$2N^3$	-	-	-
$\hat{Z}^-(k) = h(\hat{X}^-(k))$	6	-	-	6

10.1.3 Osservazione

Equazione	×	÷	∨	cos/sin
$\alpha(k) = Z(k) - \hat{Z}^-(k)$	-	-	-	-
$S(k) = \nabla_x h(k) P^-(k) \nabla_x h^T(k) + R(k)$	$MN^2 + M^2N$	-	-	-

10.1.4 Inizializzazione delle feature

L'inizializzazione delle feature comporta le seguenti operazioni:

```
state[i]=measurement+state[0]+50*cos(state[2])+270*sin(state[2]);
```

quindi 2 moltiplicazioni e 2 seni/coseni.
Il calcolo della matrice g per l'inizializzazione del *landmark* necessita di 2 moltiplicazioni, 2 coseni/seni per sensore.

Equazione	×	÷	∨	cos/sin
$x_i^+(k) = g_i(\hat{x}_\nu^-(k), y(k))$	6	-	-	6
$P^-(k) = \nabla_x g(k) P^{*-}(k) \nabla_x g^T(k)$	$2N^3$	-	-	-

10.1.5 Controllo *feature*

Il controllo della feature comporta il calcolo della norma di Mahalanobis. Si utilizza la seguente espressione:

Equazione	×	÷	∨	cos/sin
$d_{fi} = \alpha^T(k) S_\alpha^{-1}(k) \alpha(k)$	$M^3 + M^2 + M$	-	-	-

10.1.6 Aggiornamento

Equazione	×	÷	∨	cos/sin
$G(k) = P^-(k) \nabla_x h^T(k) S_\alpha^{-1}(k)$	$N^2M + M^2N$	-	-	-
$\hat{X}^+(k) = \hat{X}^-(k) + G(k)\nu(k)$	NM	-	-	-
$P^+(k) = P^-(K) - G(k) S_\alpha(k) G^T(k)$	$N^2M + M^2N$	-	-	-

10.2 Operazioni dell'algoritmo RLS-SLAM

Nelle relazioni dell'algoritmo n rappresenta il numero di punti, k il numero di iterazioni, N rappresenta il numero di stati del sistema[1]. In questo caso $N = 4$. Spostamento da punto a punto:

Equazione	×	÷	∨	cos/sin
$\begin{pmatrix} X_\nu(n) \\ \hat{X}_m(n) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu(n-1), U(n)) \\ \hat{X}_m(n-1) \end{pmatrix}$	4	2	-	4

Predizione:

Equazione	×	÷	∨	cos/sin
$X^-(k) = X^+(k-1)$	-	-	-	-

Filtraggio nel punto n:

Equazione	×	÷	∨	cos/sin
$K(k=0) = K(n-1)$	-	-	-	-
$X(k=0) = X(n-1)$	-	-	-	-
$\hat{Z}^-(k) = h(\hat{X}^-(k))$	2	-	-	2
$\alpha(k) = Z(k) - \hat{Z}^-(k)$	-	-	-	-
$C(k) = \frac{\lambda^{-1} K(k-1) H_k}{1 + \lambda^{-1} H_k^T K(k-1) H_k}$	$2N^2 + N$	N	-	-
$X(k) = \lambda^{-1/2} X(k-1) + \lambda^{-1/2} C(k) \alpha(k)$	$2N + 1$	-	-	-
$K(k) = \lambda^{-1} K(k-1) - \lambda^{-1} C(k) H_k K(k-1)$	$N^2 + 4$	-	-	-

[1] N è ovviamente anche l'ordine del filtro.

10.3 Operazioni dell'algoritmo RLS-SQR-SLAM

Nelle relazioni dell'algoritmo n rappresenta il numero di punti, k il numero di iterazioni, N è l'ordine del filtro. In questo caso al posto della matrice K si utilizza la matrice S.
Spostamento da punto a punto:

Equazione	×	÷	∨	cos/sin
$\begin{pmatrix} X_\nu(n) \\ \hat{X}_m(n) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu(n-1), U(n)) \\ \hat{X}_m(n-1) \end{pmatrix}$	4	2	-	4

Predizione:

Equazione	×	÷	∨	cos/sin
$X^-(k) = X^+(k-1)$	-	-	-	-

Filtraggio nel punto n:

Equazione	×	÷	∨	cos/sin
$S(k=0) = S(n-1)$	-	-	-	-
$X(k=0) = X(n-1)$	-	-	-	-
$\hat{Z}^-(k) = h(\hat{X}^-(k))$	2	-	-	2
$\alpha(k) = Z(k) - \hat{Z}^-(k)$	-	-	-	-
$F(k) = S^T(k-1)H_k$	N^3	-	-	-
$L(k) = S(k-1)F(k)$	N^2	-	-	-
$\beta(k) = \lambda + F^T(k)F(k)$	N	-	-	-
$\alpha(k) = \frac{1}{\beta(k)+\sqrt{\lambda\beta(k)}}$	-	1	1	-
$S(k) = \frac{1}{\sqrt{\lambda}}\left[S(k-1) - \alpha(k)L(k)F^T(k)\right]$	$N+1$	-	-	-
$C(k) = \frac{L(k)}{\beta(k)}$	-	N	-	-
$X(k) = \lambda^{-1/2}X(k-1) + \lambda^{-1/2}C(k)\alpha(k)$	$2N+1$	-	-	-

10.4 Operazioni dell'algoritmo fast RLS-SQR-SLAM

Nelle relazioni dell'algoritmo n rappresenta il numero di punti, k il numero di iterazioni, N è l'ordine del filtro.
Spostamento da punto a punto:

Equazione	×	÷	∨	cos/sin
$\begin{pmatrix} X_\nu(n) \\ \hat{X}_m(n) \end{pmatrix} = \begin{pmatrix} f(\hat{X}_\nu(n-1), U(n)) \\ \hat{X}_m(n-1) \end{pmatrix}$	4	2	-	4

Predizione:

Equazione	×	÷	∨	cos/sin
$X^-(k) = X^+(k-1)$	-	-	-	-

Filtraggio nel punto n:

Equazione	×	÷	∨	cos/sin
$S(k=0) = S(n-1)$	-	-	-	-
$X(k=0) = X(n-1)$	-	-	-	-
$v_n = H_k(k\%N)$	1	-	-	-

Equazione	×	÷	∨	cos/sin
$f_{k-1}(k) = v_n + D_{k-1}^T(\sqrt{\lambda}T_{k-1}^{-1}Z_{k-1})$	$6N$	$2N+1$	N	-
$f_k(k) = \gamma_{k-1}f_{k-1}(k)$	1	-	-	-
$Z_k = \sqrt{\lambda}T_{k-1}^{-1}Z_{k-1} - D_{k-1}f_{k-1}(k)$	N	-	-	-
$\alpha_k = \lambda\alpha_{k-1} + f_k(k)f_{k-1}(k)$	2	-	-	-
$\sigma_k = \lambda\sigma_{k-1} + v_k^2$	2	-	-	-
$\overline{D}_k = \begin{bmatrix} D_k \\ \beta_k^{-1/2}b_k(k) \end{bmatrix}$	$5N+1$	$2N+2$	$N+1$	-
$\gamma_k = \gamma_{k-1} - \alpha_k^{-1}f_k^2(k) + \beta_k^{-1}b_k^2(k)$	2	2	-	-
$T(i,j) = d_i d_j c_j e_j^{-1/2}$	$\frac{3}{2}N^2 + \frac{3}{2}N$	-	-	-
$S_k = \frac{1}{\sqrt{\lambda}}S_{k-1}T_k$	N^3+1	-	-	-
$\hat{Z}^-(k) = h(\hat{X}^-(k))$	2	-	-	2
$\alpha(k) = Z(k) - \hat{Z}^-(k)$	-	-	-	-
$X(k) = \lambda^{-1/2}X(k-1)$ $+\lambda^{-1/2}C(k)\alpha(k)$	$2N+1$	-	-	-

10.5 Confronto diretto fra i vari algoritmi

Si può calcolare per ogni punto quante operazioni effettuano i vari algoritmi in totale. In questo modo è possibile studiare la complessità dei vari algoritmi comparandoli fra loro:

I risultati, presentati nella tabella 10.1, si riferiscono al caso medio, quando si hanno rilevazioni da parte di un solo sensore ultrasonico, con 10 cicli nel punto per gli algoritmi RLS.

Algoritmo	$\times$	$\div$	$\vee$	cos/sin
Filtro di Kalman	1468	2	-	10
RLS	854	42	-	24
SQR-RLS	1004	52	11	24
fast SQR-RLS	1634	167	90	24

Tabella 10.1: Operazioni compiute, dai singoli algoritmi, nel punto. La situazione rappresenta il caso medio: rilevazione da un solo sensore ultrasonico, 10 cicli nel punto per gli algortimi RLS.

Capitolo 11

Conclusione

Con questo lavoro si è cercato di sviluppare ed analizzare un nuovo algoritmo di fusione sensoriale. Umilmente si può affermare che una piccola goccia in questo mare è stata aggiunta.

Lo scopo era di creare un algoritmo di fusione secondo il filtro di Kalman, attraverso il filtro RLS. Sono stati creati tre algoritmi di filtraggio per fusione sensoriale. Due di questi hanno evidenziato caratteristiche modeste, non impiegabili poi nella realtà. L'algoritmo fast SQR-RLS-SLAM ha dimostrato di possedere caratteristiche interessanti, in molti casi superiori allo stesso algoritmo di Kalman:

- capacità di riduzione del rumore maggiore,
- capacità di gestire forti perturbazioni,
- capacità di gestire errori con media non nulla,
- stabilità maggiore.

Tuttavia esso mantiene due problematiche già riscontrata con l'algoritmo di Kalman:

- l'errore nell'associazione dati; ma a questo problema sono disponibili più soluzioni [7];
- l'incapacità di gestire slittamenti della mappa;

La sua semplice struttura, principalmente iterativa e non vettoriale e la sua capacità di evolvere in aritmetica finita, suggerisce un impiego esterno. Ad esempio su di un DSP oppure un FPGA opportunamente programmata. Questa possibilità permetterebbe di installare sistemi di fusione sensoriale nei robot mobili con capacità di calcolo ridotte.

Sono state condotte centinaia di simulazioni e nessuna ha fatto emergere caratteristiche negative. Senz'altro i risultati ottenuti incoraggiano a procedere gli studi su questo fronte. Di importanza fondamentale sarà d'ora in avanti testare l'algoritmo proposto su di un sistema robotico reale.

Appendice A

Dimostrazioni omesse durante la trattazione

A.1 Capitolo 5

A.1.1 Dimostrazione preposizione 1

Usando la 5.4 si può fornire la seguente formulazione del vettore di predizione a priori:

$$A_k = -\left(\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} X_{n-1}^T\right)^{-1} \left(\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} v_n\right) \tag{A.1}$$

Si consideri ora il termine:

$$A_k^T \sum_{n=0}^{k} \lambda^{k-n} X_{n-1} f_k(n) \tag{A.2}$$

Usando l'equazione A.1:

$$-\left(\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} X_{n-1}^T\right)^{-1} \left(\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} v_n\right) \left(\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} f_k(n)\right) = 0 \tag{A.3}$$

Considerando che $R_{XX}^{-1}(k-1)$ è definita positiva, la prima equazione ricavata nella proposizione può essere subito ricavata. La seconda è derivabile nello stesso modo.

A.1.2 Dimostrazione preposizione 2

Si dimostrerà la prima espressione: usando l'equazione 5.29 si ricava:

$$\sum_{n=0}^{k} \lambda^{k-n} v_n^2 = \sum_{n=0}^{k} \lambda^{k-n} \left(f_k(k) - A_k^T X(k-1) \right)^2 \tag{A.4}$$

$$= \sum_{n=0}^{k} \lambda^{k-n} f_k^2 + A_k^T \sum_{n=0}^{k} \lambda^{k-n} X_{n-1} X_{n-1}^T A_k - 2 A_k^T \sum_{n=0}^{k} \lambda^{(k-n)} f_k(n) X_{n-1}$$

Utilizzando la proposizione 1 e

$$\begin{aligned} \alpha_k &= \sum_{n=0}^{k} \lambda^{k-n} f_k^2(n) \\ \beta_k &= \sum_{n=0}^{k} \lambda^{k-n} b_k^2(n) \end{aligned}$$

si ricava la prima espressione. La seconda è dimostrabile in modo analogo.

A.1.3 Dimostrazione preposizione 3

Usando l'equazione 5.29 e sostituendo v_n nel primo termine della prima espressione, si ottiene la seguente relazione:

$$\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} v_n \tag{A.5}$$

$$\sum_{n=0}^{k} \lambda^{k-n} f_k(n) X_{n-1} - \left(\sum_{n=0}^{k} \lambda^{k-n} X_{n-1} X_{n-1}^T \right) A_k$$

Quindi, usando la preposizione 1, si dimostra la prima relazione. La seconda è dimostrabile in modo analogo.

A.1.4 Dimostrazione lemma 1

Usando la

$$\overline{R}_{XX}(k) = \sum_{n=1}^{k} \lambda^{k-n} \overline{X}(n) \overline{X}^T(n)$$

e la

$$\overline{X}(k) = \begin{bmatrix} v_k \\ X(k-1) \end{bmatrix} = \begin{bmatrix} X(k) \\ r_{k-1} \end{bmatrix}$$

si ricava che:

$$\overline{R}_{XX}(k) = \sum_{n=1}^{k} \lambda^{k-n} \begin{bmatrix} X(k) \\ r_{k-1} \end{bmatrix} \begin{bmatrix} X^T(k) & r_{k-1} \end{bmatrix} \tag{A.6}$$

$$= \begin{bmatrix} \sum_{n=1}^{k} \lambda^{k-n} X(k) X^T(k) & \sum_{n=1}^{k} \lambda^{k-n} X(k) r_{k-1} \\ \sum_{n=1}^{k} \lambda^{k-n} X^T(k) r_{k-1} & \sum_{n=1}^{k} \lambda^{k-n} r_{k-1}^2 \end{bmatrix}$$

Usando l'equazione 5.21 e la preposizione 2 e 3, la precedente relazione diventa:

$$\overline{R}_{XX}(k) = \begin{bmatrix} S_k^{T^{-1}} S_k^{-1} & -S_k^{T^{-1}} S_k^{-1} B_k \\ -B_k^T S_k^{T^{-1}} S_k^{-1} & \beta_k - B_k^T S_k^{T^{-1}} S_k^{-1} B_k \end{bmatrix} \tag{A.7}$$

Si consideri il seguente partizionamento della matrice $\overline{S}_k^{-1}$:

$$\overline{S}_k^{-1} = \begin{bmatrix} \Omega & \Theta \\ \Psi^T & \xi \end{bmatrix} \tag{A.8}$$

dove Ω è una matrice $N \times N$, Θ e Ψ sono vettori colonna di N elementi e ξ è uno scalare. Quindi dalla

$$\overline{R}_{XX}^{-1}(k) = \overline{S}_k \overline{S}_k^T$$

e dalla A.8:

$$\overline{R}_{XX}(k) = \overline{S}_k^{T^{-1}} \overline{S}_k^{-1} = \begin{bmatrix} \Omega^T\Omega + \Psi\Psi^T & \Omega^T\Theta + \Psi\xi \\ \Theta^T\Omega + \xi\Psi^T & \Theta^T\Theta + \xi^2 \end{bmatrix} \tag{A.9}$$

Dalla equazione A.7 e A.9, si ottiene:

$$\Omega^T\Omega + \Psi\Psi^T = S_k^{T^{-1}} S_k^{-1}$$

$$\Omega^T\Theta + \Psi\xi = -S_k^{T^{-1}} S_k^{-1} B_k$$

$$\Theta^T\Theta + \xi^2 = B_k^T\beta_k + S_k^{T^{-1}}S_k^{-1}B_k$$

Se si pone $\Psi = 0$, la A.8 diventa triangolare superiore. I suoi elementi sono ottenuti come:

$$\Omega = S_k^{-1} \quad \Theta = -S_k^{-1}B_k \quad \xi = \beta_k^{\frac{1}{2}}$$

La matrice partizionata A.8 può essere semplicemente invertita, da cui una delle sue sotto matrici è nulla.

A.1.5 Dimostrazione lemma 2

La dimostrazione del lemma 2 è analoga a quella del lemma 1:

$$\overline{R}_{XX}(k) = \sum_{n=1}^{k} \lambda^{k-n} \begin{bmatrix} v_k \\ X(k) \end{bmatrix} \begin{bmatrix} v_k & X^T(k) \end{bmatrix} \tag{A.10}$$

$$= \begin{bmatrix} \alpha_k A_k^T S_k^{T^{-1}} S_k^{-1} A_k & -A_k^T S_k^{T^{-1}} S_k^{-1} \\ -S_k^{T^{-1}} S_k^{-1} A_k & S_k^{T^{-1}} S_k^{-1} \end{bmatrix}$$

dove sono state usate le preposizioni 2 e 3. Quindi si può partizionare la matrice nel seguente modo:

$$\widehat{S}_k^{-1} = \begin{bmatrix} \xi & \Theta^T \\ \Psi & \Omega \end{bmatrix} \tag{A.11}$$

dove Ω è una matrice $N \times N$, Θ e Ψ sono vettori colonna di N elementi e ξ è uno scalare.

Quindi:

$$\widehat{R}_{XX}(k) = \widehat{S}_k^{T^{-1}}\widehat{S}_k^{-1} = \begin{bmatrix} \xi^2 + \Psi^T\Psi & \xi\Omega^T\Theta + \Psi^T\Omega \\ \xi\Theta + \Omega^T\Psi & \Theta\Theta^T + \Omega^T\Omega \end{bmatrix} \tag{A.12}$$

Ponendo $\Theta = 0$ e ponendo A.10 uguale alla A.12 si ottiene:

$$\widehat{S}_k^{-1} = \begin{bmatrix} \alpha_k^{\frac{1}{2}} & 0^T \\ -S_{k-1}^{-1}A_k & S_{k-1}^{-1} \end{bmatrix} \tag{A.13}$$

Invertendo la A.13 si conclude la dimostrazione.

A.1.6 Dimostrazione lemma 3

Usando la fattorizzazione tratta nel lemma 2 si ottiene la seguente espressione dell'inversa della matrice di autocorrelazione:

$$\overline{R}_{XX}^{-1}(k) = \begin{bmatrix} \alpha_k^{-1} & \alpha_k^{-1} A_k^T \\ \alpha_k^{-1} A_k & S_{k-1} S_{k-1}^T + \alpha_k^{-1} A_k A_k \end{bmatrix} \tag{A.14}$$

Considerando l'equazione

$$\overline{R}_{XX}(k) = \overline{S}_k^{T^{-1}} \overline{S}_k^{-1}$$

e assumendo che la matrice $\overline{S}_k$ è triangolare superiore:

$$\overline{S}_k = \begin{bmatrix} \xi & \Theta^T \\ 0 & \Omega \end{bmatrix} \tag{A.15}$$

Quindi combinando le ultime due:

$$\overline{R}_{XX}^{-1}(k) = \begin{bmatrix} \xi + \Theta^T \Theta & \Theta^T \Omega^T \\ \Omega\Theta & \Omega\Omega^T \end{bmatrix} \tag{A.16}$$

Eguagliando le due equazioni:

$$\begin{aligned} \xi + \Theta^T \Theta &= \alpha_k^{-1} \\ \Omega\Theta &= \alpha_k^{-1} A_k \\ \Omega\Omega^T &= S_{k-1} S_{k-1}^T + \alpha_k^{-1} A_k A_k \end{aligned}$$

quindi è facile dimostrare (eq.5.45) che l'ultima equazione può venir riscritta come:

$$\Omega\Omega^T = S_{k-1}(I + \alpha_k^{-1} Z_k Z_k^T) S_{k-1}^T \tag{A.17}$$

quindi dalla precedente e dalla 5.46 si ottiene per Ω:

$$\Omega = S_{k-1} L_k \tag{A.18}$$

da cui poi:

$$\Theta = \alpha_k^{-1} L_k^{-1} Z_k \tag{A.19}$$

Premoltiplicando il termine di destra per $L_k^T L_k^{T^{-1}}$ e usando la 5.46, la matrice Θ può essere rappresentata come:

$$\Omega = \alpha_k^{-1} L_k^T (I + \alpha_k^{-1} Z_k Z_k^T)^{-1} Z_k \tag{A.20}$$

allo stesso modo usando il lemma di inversione delle matrici:

$$(I + \alpha_k^{-1} Z_k Z_k^T)^{-1} = I - \frac{\alpha_k^{-1}}{1 + \alpha_k^{-1} Z_k^T Z_k} Z_k Z_k^T \tag{A.21}$$

La forma finale di Θ è:

$$\Theta = L_k^T Z_k \frac{\alpha_k^{-1}}{1 + \alpha_k^{-1} Z_k^T Z_k} \tag{A.22}$$

Si calcola ora $\Theta^T \Theta$ col fine di calcolare il termine ξ.

$$\Theta^T \Theta = \alpha_k^{-2} \frac{Z_k^T Z_k}{1 + \alpha_k^{-1} Z_k^T Z_k} \tag{A.23}$$

Quindi:

$$\xi^2 = \alpha_k^{-1} - \Theta^T \Theta = \frac{\alpha_k^{-1}}{1 + \alpha_k^{-1} Z_k^T Z_k} \tag{A.24}$$

quindi:

$$\xi = \sqrt{\frac{\alpha_k^{-1}}{1 + \alpha_k^{-1} Z_k^T Z_k}} \tag{A.25}$$

Appendice B

Propagazione degli errori

In questa appendice introdurremo un formalismo per il trattamento dell'incertezza. Tale strumento ci consentirà di affrontare il problema della localizzazione di un veicolo mobile in presenza di errori.

B.1 Rappresentazione di parametri incerti

Un parametro è rappresentato da un vettore X di uno spazio di dimensione n. Nel nostro caso siamo interessati alla localizzazione di un robot mobile e quindi il parametro in questione è lo stato del veicolo, ovvero la sua posizione e orientazione in un sistema di riferimento cartesiano. Lo stato è descritto da un vettore

$$X = (x, y, \theta)^T \tag{B.1}$$

dove x e y sono le coordinate della posizione e θ l'orientazione data come una rotazione attorno al l'asse z. Gli errori che inevitabilmente intervengono nell'operazione di misura rendono incerta la stima del parametro. Un parametro incerto è rappresentato da una distribuzione di probabilità sulle sue componenti ovvero da una densita di probabilità $\zeta(x)$ che assegna ad ogni valore del parametro una probabilità:

$$P[X = a] = \zeta(a)dx \tag{B.2}$$

Una conoscenza dettagliata della distribuzione non è necessaria in gran parte delle applicazioni ed inoltre spesso non è accessibile a causa della mancanza di

un modello così accurato dei sensori. Per queste ragioni seguendo [52] scegliamo di individuarla stimandone i primi due momenti la media e la covarianza definiti come:

$$\overline{X} = E[X] \tag{B.3}$$

$$\Lambda_x = C[X] = E[(X-\overline{X})(X-\overline{X})^T] \tag{B.4}$$

La stima della media rappresenta il valore del parametro e la covarianza de-

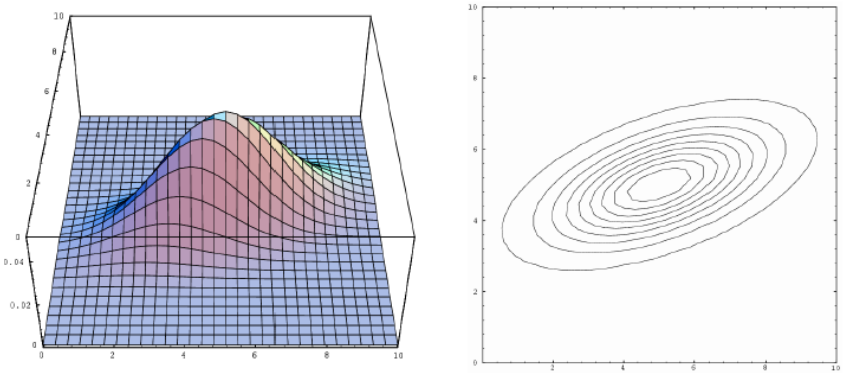

Figura B.1: Grafico e curve di livello di una distribuzione normale bivariata.

finisce una regione di confidenza per la stima. Nel seguito si scriverà $\overline{X}$ anche per indicare una stima della media della distribuzione, ovvero si confonderà la media della distribuzione con la media campionaria che ne è una stima. La misura di un parametro viene interpretata come una stima della media. In alcune circostanze può essere necessario determinare una particolare distribuzione che possieda i momenti stimati. Dati i momenti fino al secondo ordine, dal principio di massima entropia consegue che la distribuzione che assume meno informazione è la normale. Inoltre se la stima del parametro e ottenuta cumulando osservazioni diverse il teorema del limite centrale garantisce che la risultante distribuzione tende ad una normale (multivariata):

$$\zeta(X) = \frac{1}{\sqrt{(2\pi)^n |\Lambda_X|}} e^{-\frac{1}{2}(X-\overline{X})^T} \Lambda_X^T (X-\overline{X}) \tag{B.5}$$

Può essere utile rappresentare la regione di incertezza di un parametro tracciando un contorno che racchiuda una frazione assegnata (tipicamente prossima ad uno) della probabilità della distribuzione. Assumendo una distribuzione normale multivariata con i momenti assegnati, le curve di densità di probabilità costante sono iperellissoidi della famiglia:

$$(X - \overline{X})^T \Lambda_X^T (X - \overline{X} = c^2 \tag{B.6}$$

Gli assi di ciascun iperellissoide sono orientati come gli autovettori di Λ_X e la lunghezza è pari a $2c\sqrt{\widehat{\lambda}}$ (dove $\widehat{\lambda}$ è l'i-esimo autovalore di Λ_X). Si può dimostrare che:

$$P\left[(X - \overline{X})^T \Lambda_X^T (X - \overline{X} \leq u_\alpha\right] = \alpha \tag{B.7}$$

dove u_α è il quantile α-esimo di una distribuzione χ-quadrato (con n gradi di liberta, se n e la dimensione del parametro) Ad esempio, un parametro incerto in tre dimensioni $(\overline{X}, \Lambda_X)$, cade all'interno dell'ellissoide $c^2 = 7.81$ con probabilità 0.95. È importante sottolineare che le distribuzioni soggiacenti non si considerano normali. Stimiamo i momenti ed usiamo la distribuzione normale solo quando è richiesto di calcolare contorni di probabilità o, in generale, quando le circostanze impongono di assumere una particolare distribuzione. Il formalismo introdotto per la rappresentazione dell'incertezza fornisce un elegante ambito matematico per affrontare tre problemi che ogni modello dell'incertezza si trova a fronteggiare [53]:

confronto-¿ come valutare se due stime di un parametro sono consistenti;

combinazione-¿ come combinare due stime di un parametro in una sola che ne costituisca un raffinamento;

propagazione-¿ come ottenere stima di una funzione di un parametro incerto, nota una stima di quest'ultimo.

B.1.1 Confronto di misure incerte

Una operazione ricorrente nella percezione è il confronto o *matching*. I dati reali sono affetti da errori e quindi la prassi comune è di identificare due primitive quando la loro differenza abbia una norma inferiore ad una valore prefissato. La rappresentazione esplicita dell'incertezza e in particolare il formalismo introdotto nella sezione precedente, consente di usare l'incertezza come tolleranza nell'operazione di confronto. Questa consiste nel determinare se due stime si riferiscono allo stesso parametro, ovvero se due stime dello stesso parametro sono

consistenti. La similarità di due stime, ovvero la confidenza nell'identificazione, è misurata da una distanza normalizzata rispetto all'incertezza delle stime. Introduciamo la seguente forma definita positiva, detta distanza di Mahalanobis:

$$\left\|\overline{X}^m - \overline{X}^o\right\|_M = (\overline{X}^m - \overline{X}^o)(\Lambda_m^X + \Lambda_X^o)^{-1}(\overline{X}^m - \overline{X}^o) \tag{B.8}$$

Se $(\overline{X}^m - \overline{X}^o)$ una distribuzione normale a media nulla, la norma di Mahalanobis ha una distribuzione χ-quadrato con n gradi di libertà, come gia osservato precedentemente. Per esempio se identifichiamo due parametri incerti a 3 dimensioni quando la loro distanza di Mahalanobis e piu piccola di 7.81, si ha intervallo di confidenza con probabilita 0.95. Si osservi che l'utilizzo di questa metrica è naturale in presenza di quantita distribuite normalmente. Infatti, assegnata una distribuzione normale multivariata $(\overline{X}, \Lambda_X)$, un iperellissoide a probabilita costante è una sfera nella metrica di Mahalanobis:

$$B(c) = \left\{X \middle| \left\|X - \overline{X}\right\|_M \leq c^2\right\} \tag{B.9}$$

B.1.2 Propagazione di misure incerte

Quando due primitive sono legate da una trasformazione si pone il problema di propagare l'incertezza attraverso la trasformazione. Più precisamente se due parametri x e y sono legati da una relazione lineare $y = M\overline{x} + b$ ed è nota un stima (x, Λ_x) del primo, si vuole ottenere una stima del secondo. Usando le definizioni e la linearità dell'operatore di aspettazione, si verifica facilmente che:

$$y = M\overline{x} + b \tag{B.10}$$

$$\Lambda_y = M\Lambda_x M^T \tag{B.11}$$

Se la relazione $y = f(x)$ che lega x e y non è lineare è necessario approssimarla con lo sviluppo in serie di Taylor attorno al punto $\overline{x}$ ottenendo:

$$J_f(\overline{x}) = \frac{\delta f(x)}{\delta x}|_{x=\overline{x}} \tag{B.12}$$

è lo Jacobiano di f valutato in x. Troncando l'espansione di y al termine lineare e calcolando il valore di aspettazione si ottiene la stima lineare della media di y:

$$\overline{y} = f(\overline{x}) \tag{B.13}$$

Analogamente la stima del primo ordine della covarianza è:

$$\Lambda_y = J_f \Lambda_x J_f^T \tag{B.14}$$

B.1.3 Combinazione di misure incerte

Una componente fondamentale di un formalismo che tratti l'incertezza e la possibilità di raffinare incrementalmente le stime delle primitive man mano che nuovi dati si presentano. Più precisamente, se è nota una stima $(\overline{x}^m, \Lambda_x^m)$ di un parametro x e una nuova stima $(\overline{x}^o, \Lambda_x^o)$ dello stesso parametro viene ottenuta, si vorrebbe un metodo per combinarle che sortisca un raffinamento della stima del parametro. Immaginiamo che l'osservazione di x sia indiretta, ovvero che venga stimato un parametro y rappresentato da $(\overline{y}, \Lambda_y^o)$ e che sussista la relazione $y = Cx$. Il raffinamento della stima di x avviene mediante una tecnica derivata dal filtro di Kalman. La stima aggiornata $(\overline{x}^n, \Lambda_x^n)$, date la stima corrente (x^m, Λ_x^m) e la nuova osservazione $(\overline{y}^o, \Lambda_y^o)$, si calcola con:

$$K = \Lambda_x^m C^T \left[C\Lambda_x^m C^T + \Lambda_y^o\right]^{-1} \quad \text{(B.15)}$$

$$\overline{x}^n = \overline{x}^m + K(\overline{y}^o - C\overline{x}^m) \quad \text{(B.16)}$$

$$\Lambda_x^n = (I - KC)\Lambda_x^m \quad \text{(B.17)}$$

Ove non si faccia alcuna ipotesi di normalità sulla distribuzione dell'errore di misura, il filtro di Kalman fornisce il migliore stimatore lineare non polarizzato di x. Questo significa che tra tutti gli stimatori che si esprimono come combinazione lineare della sequenza (temporale) di misure $\{\overline{y}\}$, otteniamo quello per cui il valore atteso dell'errore di stima e zero (non polarizzato) e la sua varianza e minima. Si puo vedere che la stima corrente del parametro $\overline{x}^m$ viene corretta di una quantità proporzionale all'errore $(\overline{y}^o - Cx^o)$ chiamata innovazione. Il fattore di proporzionalità K, è il così detto guadagno del filtro di Kalman. Se la relazione $y = f(x)$ che lega x e y non è lineare per applicare il filtro di Kalman è necessario linarizzarla, approssimandola con il suo sviluppo in serie di Taylor troncato al termine di primo grado. In pratica la matrice C viene sostituita dallo Jacobiano di f nelle formule precedenti. Se si dispone direttamente dell'osservazione di x, $(\overline{x}^o, \Lambda_x^o)$, si pone $C = I$ e si ottiene:

$$K = \Lambda_x^m \left[\Lambda_x^m + \Lambda_y^o\right]^{-1} \quad \text{(B.18)}$$

$$\overline{x}^n = \overline{x}^m + K(\overline{x}^o - C\overline{x}^m) \quad \text{(B.19)}$$

$$\Lambda_x^n = (I - K)\Lambda_x^m \quad \text{(B.20)}$$

Questo è il caso che si presenta nella localizzazione di un veicolo mobile, il parametro ovvero lo stato del veicolo è ottenibile direttamente dalla lettura di un sensore.

B.2 Applicazione alla localizzazione di un veicolo mobile

La capacità di autolocalizzazione è una delle caratteristiche desiderabili di ogni veicolo mobile autonomo. In particolare, in assenza di riferimenti ambientali noti, risulta importante conoscere la posizione del veicolo in coordinate cartesiane. Mentre il robot si muove i dati u provenienti da codificatori ottici solidali con le ruote vengono elaborati attraverso l'applicazione A che contiene l'informazione sulla geometria e la cinematica del veicolo per determinare la velocità del robot. Il risultato è integrato per calcolare lo spostamento:

$$p = \int_{t_0}^{t} A\left(\frac{\delta u}{\delta t}\right) dt \tag{B.21}$$

Il sensore logico che legge gli spostamenti delle ruote forniti dai codificatori e aggiorna la posizione del robot è l'odometro. Un problema sorge dal fatto che c'è sempre un errore associato con il movimento di un veicolo dunque la sua posizione è un parametro incerto.

B.2.1 Odometria

In questa sezione discuteremo dettagliatamente un modello per l'odometria di un veicolo mobile è seguendo Leonard e Durrant-Whyte [22] introdurremo un modello discreto per il calcolo della matrice di covarianza della posizione. Come già accennato in due dimensioni lo stato del veicolo mobile al passo k è rappresentata dal vettore

$$X(k) = (x(k), y(k), \theta(k))^T \tag{B.22}$$

dove $(x(k), y(k))$ sono le coordinate del punto medio dell'asse delle ruote, in un sistema di riferimento cartesiano globale, e θ è l'angolo formato dall'asse delle x con la retta perpendicolare all'asse delle ruote (direzione di movimento o orientazione). Si vuole determinare lo stato al tempo $k+1$ conoscendo l'incremento nella distanza percorsa $\Delta\rho(k)$ e la variazione nell'angolo di orientazione $\Delta\theta$. Semplici considerazioni geometriche ci permettono di calcolare $x(k+1)$ nel modo seguente. Assumiamo che il robot si muove lungo un arco circolare del raggio r. Questo è motivato dal fatto che ogni traiettoria puo essere divisa in successione degli archi. I casi particolari sono il movimento lineare ($r = \infty$) e rotazionale ($r = 0$).

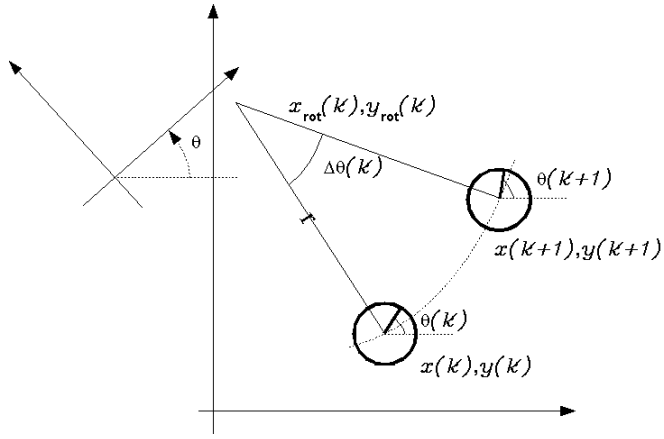

Figura B.2: Spostamento relativo

Introduciamo $\Delta\theta$ come la differenza dell'angolo di orientazione fra due passi successivi.

$$\theta(k+1) = \theta(k) + \Delta\theta(k) \tag{B.23}$$

Allora le due posizioni successive $(x(k), y(k))$ e $(x(k+1), y(k+1))$ sono messe in relazione

$$\begin{cases} x(k) = x_{rot}(k) + r\cos\left(\theta(k) - \frac{\pi}{2}\right) = x_{rot}(k) + r\sin\left(\theta(k)\right) \\ y(k) = y_{rot}(k) + r\sin\left(\theta(k) - \frac{\pi}{2}\right) = y_{rot}(k) - r\cos\left(\theta(k)\right) \end{cases} \tag{B.24}$$

e

$$\begin{cases} x(k+1) = x_{rot}(k) + r\cos\left(\theta(k+1) - \frac{\pi}{2}\right) = x_{rot}(k) + r\sin\left(\theta(k+1)\right) \\ y(k+1) = y_{rot}(k) + r\sin\left(\theta(k+1) - \frac{\pi}{2}\right) = y_{rot}(k) - r\cos\left(\theta(k+1)\right) \end{cases} \tag{B.25}$$

dove $(x_{rot}(k), y_{rot}(k))$ denotano le coordinate del centro di rotazione del movimento lungo l'arco. La posizione del robot al tempo $k+1$ può essere facilmente trovata conoscendo l'ingresso al tempo k:

$$\begin{cases} x(k+1) = x(k) + r\left(\sin\left(\theta(k+1)\right) - \sin\theta(k)\right) \\ y(k+1) = y(k) - r\left(\cos\left(\theta(k+1)\right) - \cos\theta(k)\right) \end{cases} \tag{B.26}$$

Supponiamo che $u(k) = (\Delta\rho(k), \Delta\theta(k))$ sia in ingresso dove $\Delta\rho(k)$ è la lunghezza dell'arco e differenza dell'angolo di orientazione. Allora il raggio del movimento

$r(k)$ è dato da:

$$r(k) = \frac{\Delta\rho(k)}{\Delta\theta(k)} \tag{B.27}$$

Notiamo che il $r(k)$ negativo corrisponde alla rotazione oraria mentre $r(k)$ positivo alla rotazione antioraria.

Il modello finora presentato può essere riscritto:

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) = \\ &= \begin{pmatrix} x(k) + r(k)(\sin(\theta(k) + \Delta\theta(k)) - \sin\theta(k)) \\ y(k) + r(k)(\cos(\theta(k) + \Delta\theta(k)) - \cos\theta(k)) \\ \theta(k) + \Delta\theta(k) \end{pmatrix} \\ &= \begin{pmatrix} x(k) + \frac{\Delta\rho(k)}{\Delta\theta(k)}(\sin(\theta(k) + \Delta\theta(k)) - \sin\theta(k)) \\ y(k) + \frac{\Delta\rho(k)}{\Delta\theta(k)}(\cos(\theta(k) + \Delta\theta(k)) - \cos\theta(k)) \\ \theta(k) + \Delta\theta(k) \end{pmatrix} \end{aligned} \tag{B.28}$$

Definita la dinamica di un sistema non LTI come segue:

$$x(k+1) = F(x(k), u(k)) \tag{B.29}$$

con ingresso:

$$u(k) = (\Delta\rho(k), \Delta\theta(k)) \tag{B.30}$$

B.2.2 Modello dell'errore nella posizione

A causa di una varietà di errori tra cui individuiamo:

- lo slittamento delle ruote;
- la loro deformazione dovuta al carico;
- l'errore di quantizzazione degli encoder;

la misura dello spazio percorso fornita dai codificatori è affetta da errore. Ciò si propaga rendendo imprecisa la posizione del robot calcolata dall'odometro. L'odometro fornisce una stima $\widehat{X}$ dello stato del robot. L'incertezza del valore può essere rappresentata mediante la matrice di covarianza $C[\widehat{X}]$. Si assume

che l'ingresso sia perturbato da un rumore addittivo $v(k)$ a media nulla, con matrice di covarianza nota:

$$\begin{array}{c} u^*(k) = u(k) + \nu(k) \\ E[u^*(k)] = u(k) \\ C[u^*(k)] = C[\nu(k)] \end{array} \tag{B.31}$$

Indicando con $\widehat{X}$ la stima dello stato si ottiene:

$$\widehat{X}(k+1) = F(\widehat{x}(k), u^*(k)) \tag{B.32}$$

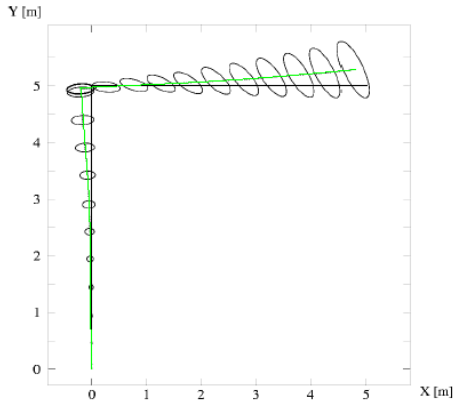

Figura B.3: Simulazione dell'errore di posizionamento. Le curve rappresentano: **nera**, percorso reale; **verde** un possibile percorso odometrico con errore relativo del 1%. Le due tracce se non ci fossero errori da parte dei sensori di movimento coinciderebbero. Le **elissi** rappresentano la regione di incertezza associata alla stima della posizione (con una confidenza del 95%).

Per l'aggiornamento della matrice di covarianza è necessario linearizzare il sistema ottenendo la formula (valida nell'ipotesi che il rumore e lo stato siano scorrelati):

$$C[\widehat{X}] = A(k)C[\widehat{X}(k)]A^T(k) + B(k)C[u^*(k)]B^T(k) \tag{B.33}$$

dove:

$$A(k) = \frac{\delta F}{\delta X}\Big|_{\widehat{X}(k)} \quad B(k) = \frac{\delta F}{\delta u}\Big|_{u(k)} \tag{B.34}$$

Assumendo che:

- gli errori relativi sulle misure dei due codificatori siano indipendenti ed identicamente distribuiti con media 0 e varianza σ^2[1];
- la distribuzione dell'errore nella distanza percorsa $\Delta\rho(k)$ abbia una varianza proporzionale alla distanza stessa, ovvero:

$$\sigma^2 = k_p|\Delta\rho(k)| \tag{B.35}$$

- la distribuzione dell'errore nella variazione dell'angolo di orientazione $\Delta\theta$ abbia una varianza che dipende sia dalla distanza percorsa, sia dalla rotazione compiuta $\Delta\theta$:

$$\sigma_\theta^2(k) = k_\theta^{\Delta\rho}|\Delta\rho(k)| + k_\theta^{\Delta\theta}|\Delta\theta(k)| \tag{B.36}$$

si ottiene la matrice di covarianza del rumore di ingresso:

$$C[\nu(k)] = \begin{pmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\theta^2 \end{pmatrix} \tag{B.37}$$

[1] Il risultato è ricavabile sperimentalmente

Bibliografia

[1] *Enciclopedia universale in quindici volumi*, Rizzoli Larousse, 1970.

[2] R. E. Kalman, *A new approach to linear filter and prediction problems,* 1960.

[3] Ali H. Sayed e Thomas Kailath, *A State-Space approach to adaptive RLS filtering*, 1994.

[4] S. Haykin, *Adaptive filter teory*, Prentice-Hall, 1991.

[5] E. Mumolo e A. Carini, *Fast square-root RLS adaptive filtering algorithms*, 1999.

[6] E. Mumolo, M. Nolich, G. Fenu e E. Ceperic, *Kalman data fusion in robot simulation using a neural network model of system dynamics*, 2002.

[7] E. Ceperic, *Localizzazione di un robot mobile in un ambiente chiuso,* tesi di laurea presso l'Università degli Studi di Trieste, 2001-2002.

[8] M. Policastro, *Dispense del corso di controlli automatici*, Università degli studi di Trieste, 2000.

[9] P. Ruckdeschel, *Ansätze zur Robustifizierung des Kalman-Filters*, 2001.

[10] M. Csorba, *Simultaneous Localisation and Map Building*, 1997.

[11] M.W.M.G. Dissanayake, P. Newman, H.F. Durrant-Whyte, S. Clark, and M. Csorba, *An experimental and theoretical investigation into simultaneous localisation and mapBuilding,* Experimental Robotics IV, 2000.

[12] E. Mumolo, A. Carini, *Volterra Adaptive Prediction of Speech with Application to Waveform Coding*, 1995.

[13] Martin Spengler, *Sensor Fusion using Kalman Filter*, info: spengler@inf.ethz.ch.

[14] J. Borenstein, *where am I? Sensor and methods for mobile robot positioning,* April 1996.

[15] Gerald J. Bierman, *Factorization Methods for Discrete Sequential Estimation,* Academi Press, 1977.

[16] M. Piaggio, *ERASMUS: A Distributed Architecture for Intelegent Robots,* PHD thesis, Università di Genova, 1998.

[17] ETHNOS homepage:
`http://www.genovarobot.com/english/ethnos/homepage_e.htm`, 2004.

[18] Bozma, R. Kuc, *Building a sonar map in a specular environment using a single mobile sensor,* IEEE Transaction on Pattern Analysis and Machine Intelligence, O. December 1991.

[19] R. Kuc, M. W. Siegel, *Physically based simulation model for acoustic sensor robot navigation,* july 1990.

[20] Kuc, V. B. Viard, *A phisycally based navigation strategy for sonar quided vehicles,* International Journal of Robotics Reserch, april 1991.

[21] J. Leonard, H. F. Durrant-Whyte, *Dynamic map building for an autonomous mobile robots,* International Jouranl of Robotics Research, I. J. Cox, august 1992.

[22] J. Leonard and H. F Durrant Whyte, *Direct Sonar Sensing for Mobile Robot Navigation*, Kluver Academic 1992.

[23] S. Koenig, R. Simmons, *Probabilistic navigation in partially observable environments*, volume 1 of Proceedings of the 1995 International Joint Conference on Artifccial Intelligence, pagine 1080-1087, 1995.

[24] W. Burgard, D. Fox, D. Hennig e T. Schmidt, *Estimating the absolute position of a mobile robot using position probability grids*, In proceedings of 14th National Conference on Artificial Intelligence (AAAI '96), 1996.

[25] B. Schiele e J. Crowley, *A comparison of position estimation techniques using occupancy grids*, Robotics and Automation Systems, 1994.

[26] R. Hinkel e T. Kinieriemen, *Environment perception with a laser radar in a fast moving robot*, In proceedings of Symposium on Robot Control, 1988.

[27] J. S. Gutmann e C. Schlegel, *AMOS: comparison of scan matching approaches for self-localization in indoor environments*, In 1st Euromicro Workshop on Advanced Mobile Robots, 1996.

[28] M. Beetz, W. Burgard, A. B. Cremers e D. Fox, *Active localization for service robot applications*, 1997.

[29] H. F. Durrant-Whyte, J. J. Leonard, *Mobile robot localization by tracking geometric beacons.* IEEE Transaction on Robotics and Auatomation, 7(3):376-382, Giugno 1991.

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

[31] Paul Michael Newman, *On the structure and solution of the simultaneous localization and map building problelem,* PhD thesis, The University of Sydney - Sydney Australia, 1999.

[32] D. Fox S. Thrun and W. Burgard, *A probabilistic approach for con current map acquisition and localization for mobile robots,* Technical report, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1997.

[33] W. Burgard F. Dellaert, D. Fox and S. Thrun, *Monte Carlo localization for mobile robots*, volume 2 of Proceedings of the 1999 IEEE In ternational Conference on Robotics and Automation., pages 1322-1328, 1999.

[34] F. Dellaert D. Fox, W. Burgard and S. Thrun, *Monte carlo localization: Effcient position estimation for mobile robots*, In Proceedings of the National Conference on Artificial Intelligence (AAAI), Orlando, FL, 1999.

[35] J. A. Castellanos, J. M. Martinez, J.Neira, E J. D. Tardos, *Simultaneous Map Building and Localization for Mobile Robots : A Multisensor Fusion Approach*, in IEEE International Conference on Robotics and Automation, Maggio 1998.

[36] S. Thrun, *Finding landmarks for mobile robot navigation,* volume 2 of Proceedings of the 1998 IEEE International Conference on Robotics and Auatomation, pages 958-963, 1998.

[37] C.F. Olson, *Subpixel localization and uncertainty estimation using occupancy grids*, volume 3 of Proceedings of the 1999 IEEE International Conference on Robotics and Auatomation, pages 1987-1992, 1999.

[38] A. Elves H. Moravec, *High resolution maps from wide angle sonar, Proceedings of the 1985 IEEE International Conference on Robotics and Automation,* pages 116-121, 1985.

[39] D. Henning T. Schmidt W. Burgard, D. Fox, *Estimating the absolute position of a mobile robot using position probability grids*, Proceedings of the 1996 National Conference on Artificial Intelligence, pages 896-901, 1996.

[40] J. S. Gutmann , W Burgard, D. Fox e K. Konolige, *An experimental comparison of localization methods, In International Conference on Intelligent Robots and Systems*, 1998.

[41] W. Burgard, D. Fox, D. Hennig e T. Schmidt, *Estimating the absolute position of a mobile robot using position probability grids*, In proceedings of 14th National Conference on Artificial Intelligence (AAAI '96), 1996.

[42] W. Burgard, D. Fox e S. Thrun, *Active mobile robot localization*, In proceedings of International Joint Conference on Artificial Intelligence, 1997.

[43] J. S. Gutmann, T. Weigel e B. Nebel, *Fast, accurate, and robust self - localization in polygonal environments*, In proceedings of International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 1999.

[44] K. Konolige e K. Chou, *Markov localization using correlation, In proceedings of International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

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

[46] J. A. Castellanos, J. M. Martinez, J.Neira, E J. D. Tardos, *Simultaneous Map Building and Localization for Mobile Robots : A Multisensor Fusion Approach,* in IEEE International Conference on Robotics and Automation, Maggio 1998.

[47] J. Leaonard e H. Durrant White, *Simultaneous Map Building and Localization for an Autonomous Mobile Robot*, in IEEE / RSJ I international Workshop on Intelligent Robots and Systems, 1991.

[48] Patric Jensfel, *Approaches to mobile robot localization in indoor environments*. PhD thesis, Royal Institute of Technology Stockholm Sweden, 2001.

[49] E.M. Guivant, J.E., Nebot, *Optimization of the simultaneous localization and map building algorithm for real-time implementation.* IEEE transaction on Robotics and, 17(3):242-257, Giugno 2001.

[50] H.F. Durrant-Whyte Nettleton EW, P.W. Gibbens, *Closed form solutions to the multiple platform simultaneous localisation and map building (slam) problem.* AeroSense 2000 24-28 , Orlando FL, USA, Aprile 2000.

[51] G.M.W.M., Durrant-Whyte-H.F. Gibbens, P.W., Dissanayake, *A closed form solution to the single degree of freedom simultaneous localisation and map building (slam) problem.* volume 1 of Proceedings of the 39th IEEE Conference on Decision and Control, pages 191 -196, 2000.

[52] R. Smith, M. Self, and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles, 1990.

[53] R. Ramparany, *An integrated support for fusion perceptual information, In Groen, Hirose, and Thorpe editors, Intelligent Autonomuos Systems.* IOS Press, 1982.

[54] *Manuale Cremonese di Meccanica Elettrotecnica Elettronica*, Parte generale, 1992.

[55] Greg Welch e Gary Bishop. *An introduction to the kalman filtering*, 2000. http://www.sc.unc.edu.

[56] P.S. Mayback, *Stochastic Models, Estimation and Control vol.1*, Academic Press, 1979.

[57] Sergio Bittanti, *Identificazione dei modelli e controllo adattativi*, Pitagora Editrice Bologna, 2000.

[58] Sergio Bittanti, *Teoria della predizione e del filtraggio*, Pitagora Editrice Bologna, 2000.