UNIVERSITA' DEGLI STUDI DI ROMA
"TOR VERGATA"

FACOLTA' DI INGEGNERIA

*Tesi di Laurea in Ingegneria Informatica*

# "Navigazione e localizzazione del robot Nomad 150 mediante filtro di Kalman"

Relatore
Ing. F. Martinelli

Laureando
Marco Gerosi

Anno Accademico 2004/2005

# Indice

**1 Introduzione** **5**

1.1 Generalità . . . . . 5

**2 Il Nomad 150 e il software a corredo** **7**

2.1 Caratteristiche Hardware e Software . . . . . 7

2.1.1 Il robot e il computer utilizzati . . . . . 7

2.1.2 Nserver e librerie di gestione

9

**3 Il problema della navigazione e localizzazione** **11**

3.1 Il problema della localizzazione . . . . . 11

3.1.1 Filtro di Kalman esteso per il Nomad 150 . . . 11

3.1.2 Caso particolare del Nomad 150 . . . . . 20

3.2 Navigazione e aggiramento ostacoli . . . . . 29

**4 Applicazioni sperimentali** **37**

4.1 L' Ambiente . . . . . 37

4.2 Il contesto sperimentale . . . . . 37

4.3 Le prove condotte . . . . . 40

4.4 I risultati . . . . . 42

**5 Conclusioni** **47**

# Capitolo 1

# Introduzione

## 1.1 Generalità

Nella presente relazione viene illustrato un programma integrato di navigazione e localizzazione implementato e testato sul robot Nomad150. L' algoritmo include un filtro di Kalman esteso utilizzato per fondere i dati provenienti dai sonar con quelli odomentrici e rendere così più affidabile e robusta la localizzazione del robot. La fusione sensoriale cioè l'uso sinergico dell' informazione fornita da più dispositivi sensoriali comporta una serie di aspetti che sono stati affrontati in questo lavoro e sono: la ridondanza, la complementarità, la contemporaneità. Questi aspetti, se da un lato permettono di avere un' informazione più accurata, contenente caratteristiche dell'ambiente impossibili da rilevare con un singolo sensore, in minor tempo e ad un costo inferiore, dall'altro lato, invece, comportano notevoli difficoltà in fase progettuale. Tali difficoltà derivano dal fatto che non esistono metodi generali per la progettazione di un sistema di fusione ma, bensi', tutto dipende dall'obiettivo che si vuole perseguire e dai sensori usati. In questa sede sono stati analizzati i mezzi hardware a disposizione e le caratteristiche di movimento quali la velocità massima del robot e il raggio d' azione dei sonar, al fine di scegliere le soluzioni tecniche più appropriate. Inoltre, poichè il campo della robotica mobile richiede tempi di elaborazione contenuti (dovendo operare in tempo reale), è stato necessario ottimizzare il controllo dei flussi di dati, in quanto l'aumento dei sensori a disposizione implica un aumento dei dati da elaborare. Altro problema affrontato è stata la coordinazione dei dati, in quanto l'uso di sensori differenti, comporta una differente cadenza con cui essi operano e una diversa risoluzione. Le informazioni ottenu-

te sono state, quindi, necessariamente uniformate. L'altro obiettivo di questa tesi è stato la creazione e la successiva implementazione di un algoritmo di navigazione che permettesse al Nomad150 di raggiungere una destinazione desiderata, evitando uno o più ostacoli.

L'impiego del filtro di Kalman esteso (al caso non lineare) nella tesi qui sviluppata è giustificato se si considera la rumorosità dei sonsori acustici di bassa qualità e i risultati grossolani derivanti dall'integrazione diretta dei dati tachimetrici da cui deriva la necessità d'impiego di tecniche di fusione sensoriale.

La relazione è strutturata in una sezione in cui si descrive il problema della navigazione in ambienti parzialmente noti, in una in cui si descrive il robot e il software utilizzato per la tesi e un'ultima parte in cui si mostrano i risultati sperimentali ottenuti in un ambiente con pareti di legno disposto nell'atrio al quarto piano dell'edificio di Ingegneria dell' Informazione dell' Università di Roma Tor Vergata.

# Capitolo 2

# Il Nomad 150 e il software a corredo

## 2.1 Caratteristiche Hardware e Software

### 2.1.1 Il robot e il computer utilizzati

Il sistema utilizzato per questo lavoro è composto dal robot *Nomad 150* e un computer portatile ASUS.

Il robot *Nomad150[3]* è un sistema dotato di tre ruote simmetriche azionate da due motori uno per gli spostamenti traslazionali e uno per quelli di rotazione. Quindi le tre ruote traslano insieme e ruotano tutte nello stesso verso. Un terzo motore controlla la rotazione della torretta. Il raggio di rotazione è pari a zero, cioè il robot può ruotare su se stesso senza traslare. Il sistema e' *anolonomo*[1], in quanto presenta dei vincoli non integrabili nelle velocità.

Il controllo dei motori e' affidato ad un microprocessore MOTOROLA MC68008/ASIC, che permette una velocità massima di 20pollici/sec[2] per la traslazione e fino a 45 gradi/sec.

Il Nomad 150 è dotato di tre tipi di sensori: tattili, odometrici e acustici. Le informazioni tattili sono date dal *Sensus 100$^{TM}$ Tactile System*, un complesso di 20 sensori di pressione (*bumper*), organizzati in due anelli di 10 sensori che coprono tutta la circonferenza del robot

[1]Si può chiarire meglio il concetto di anolonomo prendendo in considerazione un' automobile. Se si vuole parcheggiare non e' possibile affiancarsi al posto libero e spostarsi ortogonalmente ad esso, ma bensì si devono eseguire una serie di rotazioni insieme a delle traslazioni.
Con un sistema olonomo è possibile evitare tutto questo in quanto è in grado di portarsi in qualunque punto dello spazio di lavoro con uno spostamento unico; si pensi ad un robot con due motori traslazionali oppure delle sfere.

[2]1polllice=2,54cm.

Figura 2.1: Sistema Nomad150 e computer portatile ASUS

con una risoluzione di 18 gradi. I dati acustici provengono dal *Sensus 200$^{TM}$ Sonar Ranging System,* un sistema di 16 sonar alimentati a 5 Volts che restituiscono valori da 6 a 255 pollici con una precisione dell'1% sull'intero range di misura e con una risoluzione di 22,5 gradi.

Il sistema di sonar e' basato sul tempo che intercorre tra invio del segnale acustico e ricezione dello stesso; tali segnali hanno una frequenza di 49,4 KHz; per evitare che il sonar confonda il ronzio di emissione del segnale che esso stesso emette con il segnale di ritorno, nel momento di emissione del segnale, viene disattivato il circuito di *echo,* ed è per questo che la distanza miniva rilevata è di 6 pollici.

Il NOMAD 150 viene alimentato con tre batterie al piombo a secco da 12V e con una capacità di 12 Ah ciascuna. Tali batterie hanno una durata di piu' di sette ore. I metodi con cui si può comandare il robot sono tre: tramite joystick, tramite porta seriale, tramite modem radio Ethernet. Il programma realizzato per questa tesi prevede l'uso diretto della porta seriale.

Il computer utilizzato è un portatile ASUS L3400Tp con processore Intel Pentium 4 con una frequenza di 2GHz e 256MB di memoria Ram. Le dimensioni relativamente contenute del computer hanno permesso il suo posizionamento sopra la torretta del Nomad evitando l'uso modem radio Ethernet.

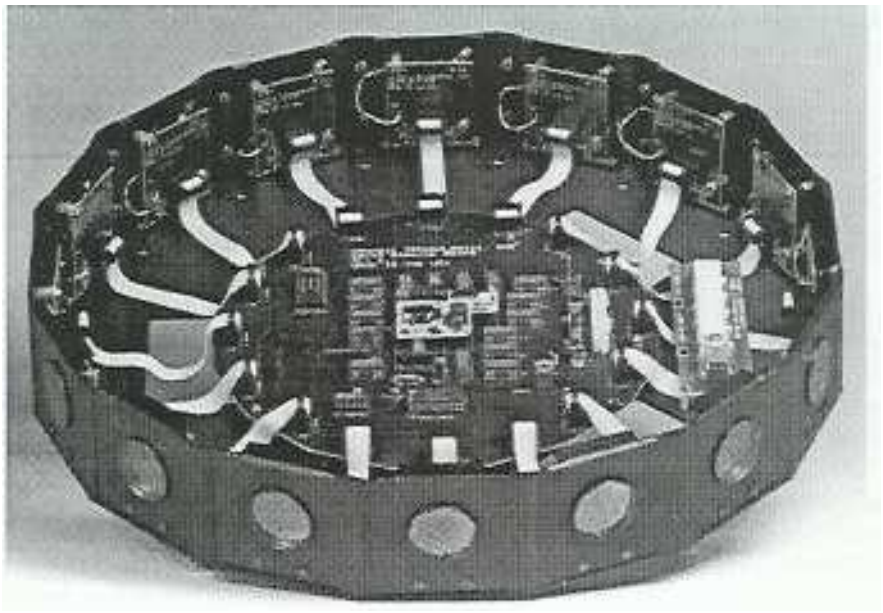

Figura 2.2: Il sistema Sensus 200 Sonar System

Il sistema robot-portatile però, è vincolato dalla scarsa autonomia del computer, per cui nei vari test è stato necessario l'utilizzo di una prolunga collegata alla rete elettrica per alimentarlo, mentre invece le batterie del robot si sono dimostrate molto efficienti e in grado di farlo funzionare anche una giornata intera.

### 2.1.2 Nserver e librerie di gestione

Il robot nomad 150 è corredato di un software di gestione chiamato Nserver[4] che comprende un server - simulatore che se installato su un sistema operativo Linux, permette di impartire al robot tutti i comandi disponibili da libreria, e di testare in simulazione i programmi realizzati per il robot.

Quando si scrive un programma per il Nomad, si hanno a disposizione anche delle librerie di gestione C/C++ : Nclient.h e Ndirect.h; tali librerie sono proprietarie della Nomadic Inc. e permettono di impartire al robot tutte le principali funzioni di movimento di gestione dei sensori e dei valori nel vettore di stato[3] .

L' Nserver genera una interfaccia grafica (Figura 2.3), che permette di vedere i dati provenienti dai sensori, la posizione del robot e la

[3]Il vettore di stato è un grande array dove vengono aggiornati ad ogni movimento o cambiamento di stato del robot, i valori degli encoder, dei sonar, dei bumper; per informazioni piu' dettagliate fare riferimento al manuale utente.

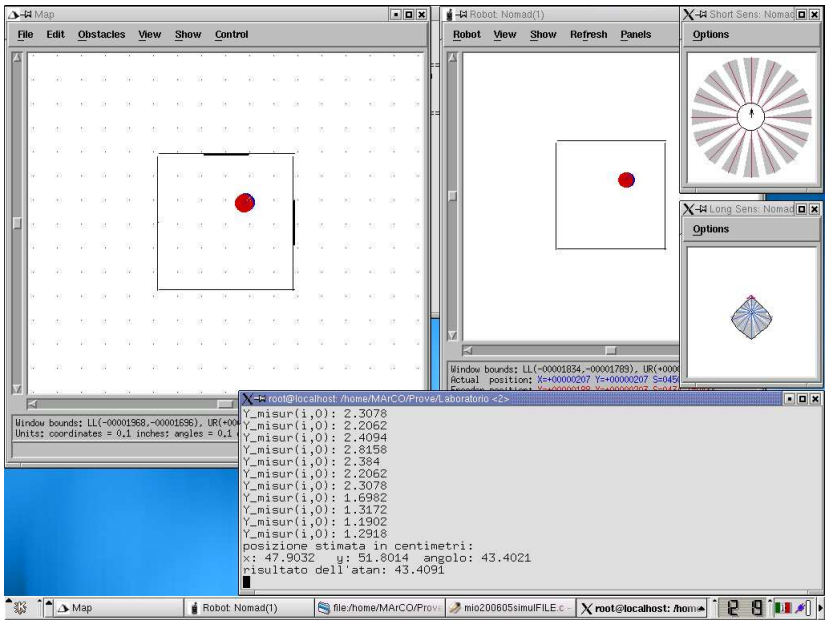


Figura 2.3: Schermata principale del programma Nserver

mappa dello spazio in cui esso si muove. Il programma di localizzazione e navigazione realizzato utilizza proprio l'Nserver (sotto Linux, ), ma volendo è possibile utilizzare un programma di emulazione terminale (*hyperterminal* sotto Windows e *minicom* sotto Linux) per far eseguire al Robot comandi elementari; entrambe le modalita' sono accessibili con connessione diretta seriale oppure radiomodem.

Come già detto prima esistono due librerie di gestione: Nclient.h e Ndirect.h. La prima se compilata insieme al programma come file oggetto renderà il programma (scritto dall'utente) *client* e l'Nserver farà da server, appunto. La seconda libreria invece serve nel caso sia già testato il programma e si è certi del corretto funzionamento; infatti permette un collegamento diretto tra programma e robot senza alcun server che fa da tramite.

Per le prove finali si è utilizzato il simulatore per mostrare la convergenza dei valori stimati verso quelli "reali", o meglio simulati.

# Capitolo 3

# Il problema della navigazione e localizzazione

## 3.1 Il problema della localizzazione

Per localizzazione robotica si intende la stima della posizione di un robot grazie all' acquisizione di dati dall'ambiente esterno tramite sensori. Poichè la natura dei dati dei sensori è sempre eterogenea e soggetta ad errori, si utilizzano tecniche di fusione sensoriale in grado di integrare fra loro le informazioni provenienti da differenti dispositivi sensoriali. In questo modo si possono ottenere stime di posizione accurate.

Nella tesi sviluppata è stato utilizzato il filtro di Kalman esteso (EKF) così come descritto in[1] per un contesto simulativo. Tale filtro è stato adattato per un utilizzo reale sul Nomad150. Rispetto al lavoro teorico che era stato condotto in origine, e' stato necessario adattare il filtro in quanto sono sorti problemi che sul simulatore non compaiono e che quindi, non erano stati considerati (come ad esempio i prolungati tempi di acquisizione dei dati dei sonar e l'errore a cui sono soggetti).

Nelle prossime pagine si spiega il funzionamento del filtro e il modello del robot. Per ulteriori approfondimenti si fa riferimento a[1] e [2].

### 3.1.1 Filtro di Kalman esteso per il Nomad 150

#### Filtro di Kalman esteso

Nel 1960, Kalman pubblica un articolo in cui descrive un metodo per risolvere i problemi di filtri lineari a tempo discreto. Da allora è stato

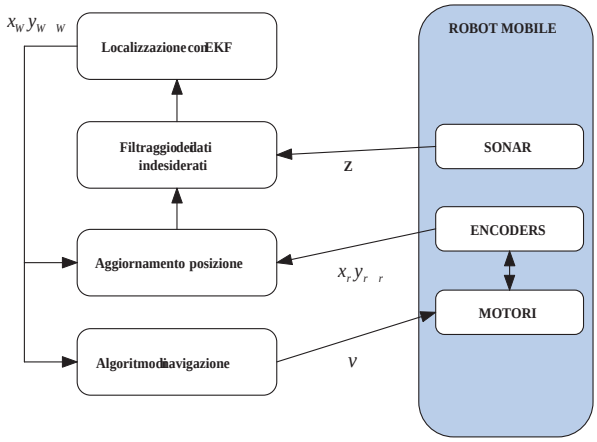


Figura 3.1: Struttura dell'algoritmo di localizzazione in ambiente noto.

usato in larga scala nel calcolo numerico e in particolare nel campo della navigazione autonoma o assistita. Il filtro di Kalman è composto da alcune equazioni matematiche che permettono di stimare lo stato di un sistema efficientemente minimizzando l'errore quadratico medio. Il filtro in questione è molto potente e permette di stimare stati passati, presenti e futuri anche se il modello del sistema non è perfettamente conosciuto.

Nel campo della robotica il filtro di Kalman viene molto utilizzato nella sua versione estesa, in quanto si ha generalmente a che fare con sistemi non lineari, proprio come nel caso del robot Nomad.

Si consideri di voler stimare lo stato x $\in \Re^n$ di un processo a tempo discreto governato dall' equazione alle differenze lineare stocastica

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{3.1}$$

con misure $z \in \Re^m$:

$$z_k = Hx_k + v_k. \tag{3.2}$$

Le variabili aleatorie $w_k$ e $v_k$ rappresentano i rumori sul processo e sulle misurazioni rispettivamente. Tali rumori sono assunti come indipendenti[1], gaussiani, a media nulla e con matrice di covarianza simmetrica e semidefinita positiva:

$$E[w_i] = 0 \qquad\qquad E[v_i] = 0$$

[1] Vedi [9].

$$\begin{aligned} E[w_i w_i'] &= Q \geq 0 \quad , \quad E[v_i v_i'] = R \geq 0 \\ E[w_i w_j'] &= 0,\, i \neq j \quad E[v_i v_j'] = 0,\, i \neq j \end{aligned} \tag{3.3}$$

Le matrici Q ed R, cioè le covarianze sul rumore del sistema e sul rumore di misura (rispettivamente), dovrebbero cambiare ad ogni ciclo di misura, ma qui si assume che restino costanti.

La matrice $A$ $n \times n$ nell'equazione alle differenze (3.1) mette in relazione lo stato al tempo *k-1* con lo stato al tempo attuale $k$ (in assenza di ingressi e rumori sul processo). La matrice B $n \times l$ relaziona l'ingresso di controllo (opzionale) $u \in \Re^l$ con lo stato $x$. La matrice $m \times n$ $H$ invece, relaziona lo stato con le misure dirette $y_k$. Entrambe le matrici $A$ e $C$ cambiano ad ogni ciclo di misura, ma vengono assunte costanti.

Definiamo ora $\hat{x}_k^- \in \Re^n$(notare l'apice con il segno meno) come la stima dello stato al tempo $k$ *a priori,* conoscendo cioè soltanto gli stati antecedenti a $k$, e $\hat{x}_k \in \Re^n$ come la stima dello stato al tempo $k$ *a posteriori*, cioè conoscendo anche la misura $y_k$.

Possiamo così definire degli errori di stima *a priori* e *a posteriori:*

$$\begin{aligned} e_k^- &\equiv x_k - \hat{x}_k^- \\ e_k &\equiv x_k - \hat{x}_k \end{aligned}$$

La matrice di covarianza dell'errore di stima *a priori* è uguale a:

$$P_k^- = E[e_k^- e_k^{-T}] \tag{3.4}$$

mentre la covarianza per quello a posteriori è:

$$P_k = E[e_k e_k^T]. \tag{3.5}$$

Nel derivare le equazioni del filtro di Kalman, si può cominciare col ricavare una equazione che calcoli una stima dello stato *a posteriori* $\hat{x}_k$ come combinazione lineare di una stima *a priori* $\hat{x}_k^-$ ed una differenza pesata tra una misura attuale $z_k$ed una predizione di misura $H\hat{x}_k^-$come mostrato sotto:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{3.6}$$

Il termine $(z_k - H\hat{x}_k^-)$ è chiamato *termine d'innovazione o residua-*

*le* e riflette la discrepanza tra la misura attuale al tempo $k$ e misura predetta *a priori*. Un *residuale* pari a zero indica una stima perfetta.

La matrice K $n \times m$ nella (3.6) è chiamata guadagno di Kalman e serve a minimizzare la covarianza[2] di errore *a posteriori*. Tale guadagno può essere ricavato sostituendo la (3.6) in $e_k$ nella (3.5); dopodichè si deriva $P_k$in $K$ e si eguaglia tale derivata a zero. In questo modo trovo il valore di K che minimizza la covarianza dell'errore (proposto qui di seguito nella sua forma più comune):

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ &= \frac{P_K^- H^T}{H P_k^- H^T + R} \end{aligned} \tag{3.7}$$

dove $K_k$ è uguale alla seconda espressione soltanto se $n = 1$. Si può vedere dalla (3.7) che se la covarianza sull'errore di misura R tende a zero, il guadagno K diventa più grande, quindi tende a far pesare di più il *termine d'innovazione* nell'(3.6)

$$\lim_{R_k \longrightarrow 0} K_k = H^{-1}$$

Nel caso in cui, invece, sia la covarianza sull'errore di stima $P_K^-$ a tendere a zero, il guadagno K diminuisce e fa pesare meno il residuale:

$$\lim_{P_K^- \longrightarrow 0} K_k = 0.$$

Il filtro di Kalman stima un processo usando una forma di controllo in retroazione: il filtro stima lo stato del sistema ad un certo tempo e poi ottiene in retroazione una misura rumorosa. Quindi le equazioni del filtro di Kalman si dividono in due gruppi: equazioni di *predizione* ed equazioni di *stima* (Figura 3.2). Le equazioni di *predizione* fanno una stima *a priori* (cioè al tempo k) dello stato del sistema al tempo k+1, mentre le equazioni di *stima* fanno una correzione *a posteriori* (cioè al tempo k+1) del valore che era stato stimato al tempo k sfruttando le misure tornano in retroazione.

Sono riportate di seguito le equazioni della fase di *predizione*

[2] Vedi [9]

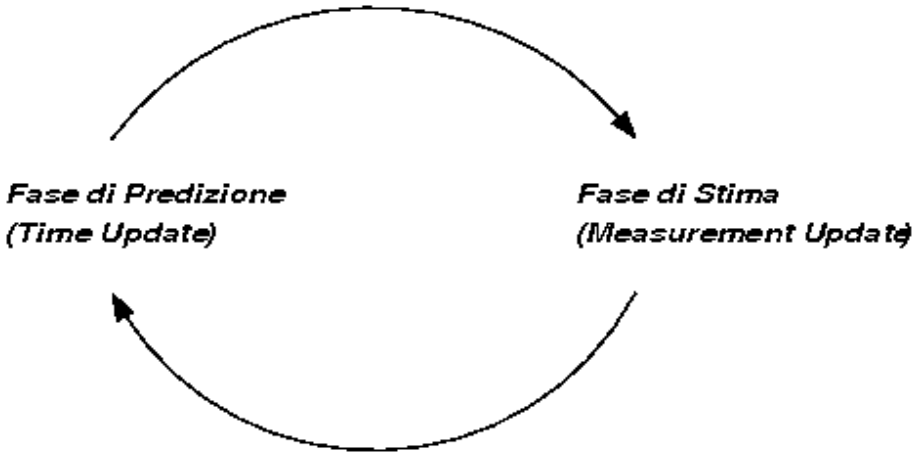


Figura 3.2: Schema ciclo Predizione-Stima

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{3.8}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{3.9}$$

e quelle della fase di *stima*

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{3.10}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3.11}$$
$$P_k = (I - K_k H)P_k^- \tag{3.12}$$

Le matrici $A$ e $B$ sono le stesse della (3.1), mentre Q è la matrice di covarianza del rumore del sistema (3.3) .

Nella fase di stima, prima di tutto si calcola il guadagno di Kalman $K_k$. Da notare che la (3.10) è uguale alla (3.7). Nella (3.11) si calcola la stima *a posteriori* dello stato, cioè si fa la correzione della stima *a priori* usando i dati delle misure $z_k$. Infine si calcola la matrice di covarianza dell'errore *a posteriori* (3.12).

Questo metodo ricorsivo ad ogni ciclo calcola prima le equazioni della fase di stima e poi fa la predizione sul tempo successivo. Nella Figura 3.3 si può vedere schematicamente il funzionamento.

La covarianza sul rumore di misura R può essere calcolata prima di far partire l'algoritmo di Kalman, in quanto è plausibile poter rilevare qualche misura OFF-LINE per calcolare la varianza sul rumore di misura. La covarianza sul rumore del processo Q è invece, più difficile

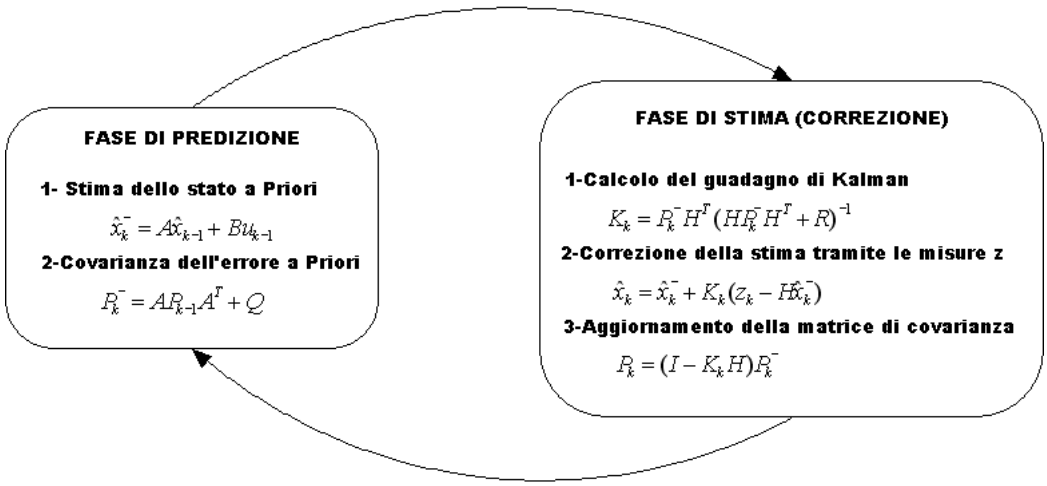


Figura 3.3: Schema Ciclo Predizione-Stima per sistemi lineari

da determinare in quanto in genere non si può osservare direttamente il processo che vogliamo stimare. Di fatto molte volte il filtro può essere più performante se si regolano le matrici Q e R.

Nella condizione in cui Q e R restano costanti, si può notare che la covarianza sull'errore di stima $P_k$ e il guadagno di Kalman $K_k$ si stabilizzano rapidamente e restano costanti.

Come già detto il filtro di Kalman si usa di solito per trovare una stima dello stato $x \in \Re^n$ di un sistema controllato tempo-discreto governato da una equazione alle differenze lineare stocastica. Ma cosa succede se le relazioni del processo e o delle misure sono non lineari? In questi casi il filtro di Kalman si rivela molto utile e viene chiamato Filtro di Kalman Esteso (EKF).

Facendo uno sviluppo in serie di Taylor, possiamo linearizzare la stima dello stato $\hat{x}_k$ intorno alla stima corrente $\hat{x}_{k-1}$ usando le derivate parziali. Si comincia quindi modificando le relazioni (3.1) e (3.2). Il processo è ora governato da una equazione alle differenze stocastica non lineare

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{3.13}$$

e le misure $z \in \Re^m$ sono date dalla seguente equazione

$$z_k = h(x_k, v_k) \tag{3.14}$$

dove $w_k$ e $v_k$ rappresentano ancora i rumori del processo e di misura di cui non si conoscono i valori ad ogni ciclo di lavoro. Comunque si

può approssimare il vettore di stato e di misura senza tali rumori (in quanto tutti uguali e a media nulla )

$$\tilde{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0) \tag{3.15}$$
$$\tilde{z}_k = h(\tilde{x}_k, 0) \tag{3.16}$$

E' importante notare che il difetto fondamentale dell' EKF è che la distribuzione delle varie variabili aleatorie non è più normale (gaussiana) quando si passa alle rispettive trasformazioni non lineari. L' EKF è quindi soltanto un'osservatore ottimo dello stato che approssima l'ottimalità della regola di Bayes per linearizzazione.

Per stimare un processo governato da equazioni non lineari, si può cominciare con lo scrivere le nuove equazioni del sistema che linearizzano le (3.15) e (3.16):

$$x_k \approx \tilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1}, \tag{3.17}$$
$$z_k \approx \tilde{z}_k + H(x_k - \tilde{x}_k) + Vv_k. \tag{3.18}$$

dove

- $x_k$ e $z_k$ sono i vettori dello stato e della misura attuale,
- $\tilde{x}_k$ e $\tilde{z}_k$ sono i vettori dello stato e della misura approssimato come in (3.15) e (3.16),
- $\hat{x}_k$ è una stima *a posteriori* dello stato al tempo $k$
- $w_k$ e $v_k$ sono ancora le variabili aleatorie che rappresentano i rumori di processo e di misura come nella (3.3),
- $A$ è la matrice Jacobiana delle derivate parziali di $f$ rispetto a $x$

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0),$$

- $W$ è la matrice Jacobiana delle derivate parziali di $f$ rispetto a $w$

$$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{x}_{k-1}, u_{k-1}, 0),$$

- $H$ è la matrice Jacobiana delle derivate parziali di $h$ rispetto a $x$

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\tilde{x}_k, 0),$$

- $V$ è la matrice Jacobiana delle derivate parziali di $h$ rispetto a $v$

$$V_{[i,j]} = \frac{\partial f_{[i]}}{\partial v_{[j]}}(\tilde{x}_k, 0)$$

Queste matrici A, W, H, e V pur variando ad ogni step temporale, verranno in seguito indicate senza pedice k per semplicità.

Adesso si può definire un nuovo errore di predizione

$$\tilde{e}_{x_k} \equiv x_k - \tilde{x}_k, \tag{3.19}$$

e un nuovo errore di misura

$$\tilde{e}_{z_k} \equiv z_k - \tilde{z}_k. \tag{3.20}$$

Poichè nella realtà non si conosce il valore reale di $x_k$ e $z_k$, allora tali espressioni dell'errore vanno modificate come segue:

$$\tilde{e}_{x_k} \approx A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k, \tag{3.21}$$

$$\tilde{e}_{z_k} \approx H\tilde{e}_{x_k} + \eta_k. \tag{3.22}$$

dove $\varepsilon_k$ e $\eta_k$ sono due nuove variabili aleatorie aventi media nulla e matrice di covarianza $WQW^T$ e $VRV^T$ con $Q$ e $R$ come da (3.3).

La (3.21) e la (3.22) sono equazioni lineari molto simili alla (3.1) e alla (3.2). Per questo si può sostituire l'errore di misura attuale $\tilde{e}_{z_k}$ nella (3.20) e si può introdurre un altro (ipotetico) filtro di Kalman per stimare l'errore di predizione $\tilde{e}_{x_k}$ dato dalla (3.21). Tale stima viene chiamata $\hat{e}_k$, può essere usata nella (3.19) per ottenere una stima dello stato *a posteriori* per il sistema originale non lineare:

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k. \tag{3.23}$$

Le variabili aleatorie della (3.21) e (3.22) hanno distribuzioni di probabilità

$$p(\tilde{e}_{x_k}) \sim N(0, E[\tilde{e}_{x_k}\tilde{e}_{x_k}^T])$$

$$p(\varepsilon_k) \sim N(0, WQ_kW^T)$$

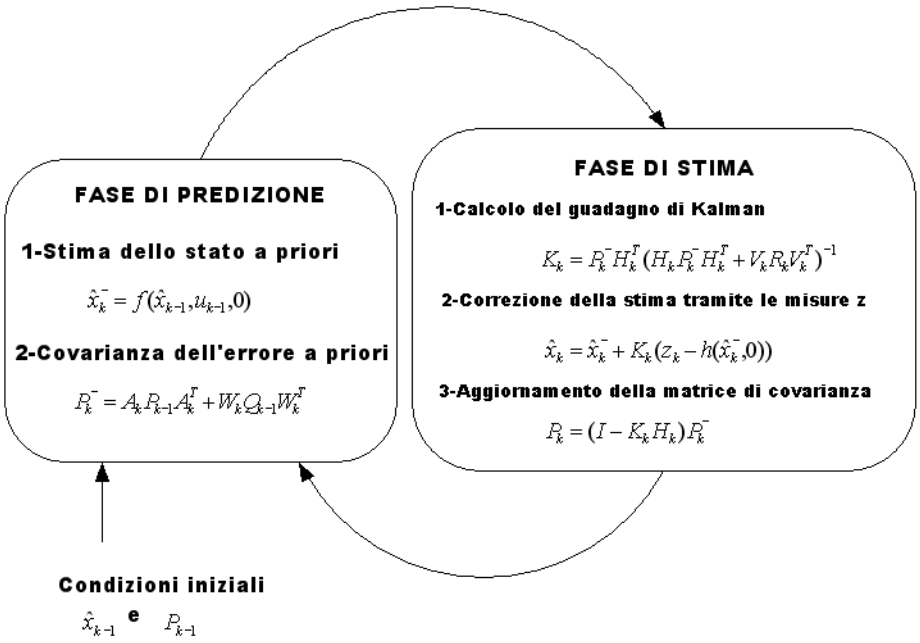


Figura 3.4: Schema riassuntivo dell'EKF per sistemi non lineari

$$p(\eta_k) \sim N(0, VR_kV^T)$$

Il filtro di Kalman usato per stimare $\hat{e}_k$ è

$$\hat{e}_k = K_k \tilde{e}_{z_k}. \tag{3.24}$$

Sostituendo la (3.24) nella (3.23) e, usando la (3.20) si può constatare che il secondo ipotetico filtro di Kalman non serve:

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k \tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k(z_k - \tilde{z}_k) \end{aligned} \tag{3.25}$$

A questo punto, l'equazione (3.25) può essere usata insieme al $K_k$ della (3.10) (opportunamente modificata per la covarianza dell'errore di misura), per le equazioni di predizione del filtro di Kalman esteso. I termini $\tilde{x}_k$ e $\tilde{z}_k$ derivano dalle (3.15) e (3.16).

In Figura 3.4 si riporta uno schema riassuntivo del filtro di Kalman esteso suddiviso in *fase di predizione* e *fase di stima.*

Dallo schema si può notare che $\tilde{x}_k$ è stata sostituita con $\hat{x}_k^-$ per rimanere coerenti con la notazione usata per il filtro di Kalman discreto (per sistemi lineari) e che $A$, $W$, $H$, e $V$ hanno il pedice $k$ per sottolineare il fatto che queste matrici si modificano ad ogni step di

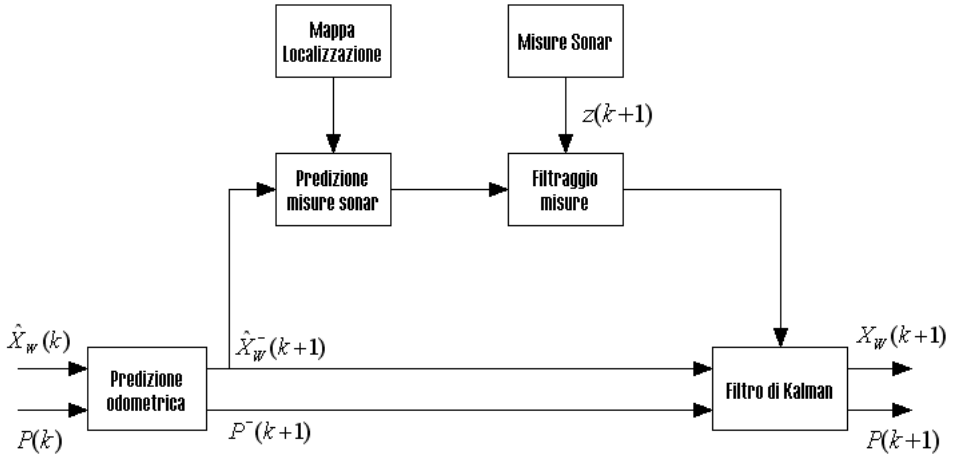


Figura 3.5: Schema dell'EKF

tempo.

Sempre dallo schema si può capire come funziona il filtro: nella fase di predizione si stima il valore che può avere lo stato al tempo corrente $k$ utilizzando i dati del tempo $k-1$ e inoltre si calcola una matrice di covarianza dell'errore *a priori* $P_k^-$ avendo a disposizione soltanto $P_{k-1}$ e $Q_{k-1}$ (covarianza sul rumore di processo). Nella fase di stima si calcola il guadagno di Kalman e si opera una correzione sulla stima fatta in precedenza. Si calcola il valore che dovrebbe avere $z_k$ cioè $h(\hat{x}_k^-, 0)$ e poi lo si confronta con il $z_k$ reale e si somma, opportunamente pesato grazie al guadagno $K_k$, alla stima *a priori* $\hat{x}_k^-$. Infine si aggiorna $P_k$.

### 3.1.2 Caso particolare del Nomad 150

La stima della posizione di un robot può essere ottenuta raccogliendo informazioni dall'ambiente circostante. A seconda del tipo di movimentazione che si vuole ottenere, dell'ambiente in cui esso si deve muovere, e del tipo e precisione dei sensori a disposizione, le informazioni necessarie si differenziano. Per l'applicazione sviluppata in questo ambito, il filtro ha una forma molto specifica adattata per le esigenze che si volevano soddisfare. Tali esigenze prevedevano la stima della posizione e dell'orientamento del robot in tempi relativamente contenuti, utilizzando sensori odometrici (encoder sulle ruote e sulla torretta), e sensori acustici (16 sonar posti sulla circonferenza del robot). Schematizzazione riportata in Figura 3.5.

Se si suppone un ambiente bidimensionale (Fig. 3.6), la configura-

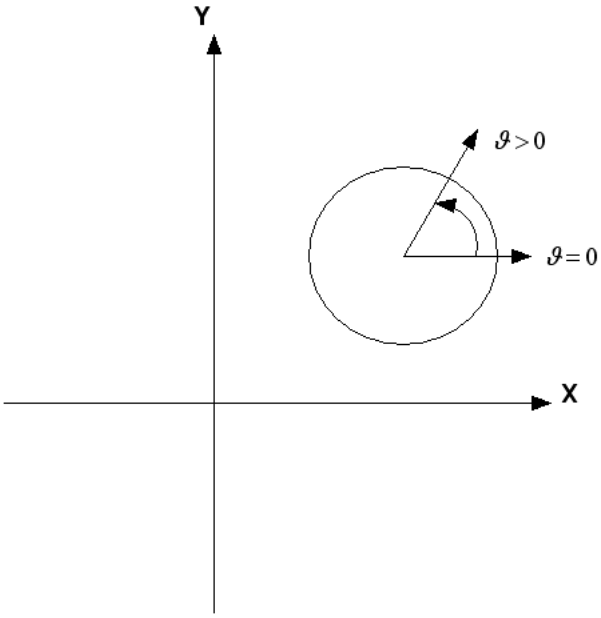


Figura 3.6: Sistema di riferimento adottato

zione del robot rispetto ad un sistema di riferimento fisso solidale con l'ambiente può essere rappresentata con un vettore di tre parametri:

$$X_r = \begin{bmatrix} x & y & \vartheta \end{bmatrix}' \tag{3.26}$$

Se si ipotizza un caso ideale in cui il robot si muova in condizioni di perfetta aderenza con il suolo, con sensori calibrati e in assenza di rumori di misura, possono essere utilizzate le equazioni (come considerato in [5]):

$$\begin{aligned} \dot{x} &= v(t)\cos\vartheta(t) \\ \dot{y} &= v(t)\sin\vartheta(t) \\ \dot{\vartheta} &= \omega(t) \end{aligned} \tag{3.27}$$

con v(t) e $\omega$(t) rispettivamente le velocità di traslazione e rotazione del robot.

Se si definisce T un tempo di campionamento opportuno, le stesse equazioni (3.27) possono essere discretizzate.

Per la sola x

$$\frac{x((k+1)T) - x(kT)}{T} = v(kT)\cos\left(\vartheta(kT) + \frac{\omega(kT)T}{2}\right)$$

$$
\begin{aligned}
&\Longrightarrow \\
x((k+1)T) &= x(kT) + Tv(kT)\cos\left(\vartheta(kT) + \frac{\omega(kT)T}{2}\right) \\
&\Longrightarrow \\
x(k+1) &= x(k) + Tv(k)\cos\left(\vartheta(k) + \frac{\omega(k)T}{2}\right)
\end{aligned}
$$

Quindi il risultato ottenuto può essere esteso a tutte e tre le equazioni

$$
\begin{aligned}
x(k+1) &= x(k) + Tv(k)\cos\left(\vartheta(k) + \frac{\omega(k)T}{2}\right) \\
y(k+1) &= y(k) + Tv(k)\sin\left(\vartheta(k) + \frac{\omega(k)T}{2}\right) \\
\vartheta(k+1) &= \vartheta(k) + T\omega(k)
\end{aligned} \tag{3.28}
$$

Quanto scritto vale però come gia detto solo per casi ideali, in quanto il robot durante il suo moto è soggetto a slittamenti, urti e si imbatte in discontinuità del suolo. Tutto ciò provoca degli errori molto grandi nei dati riportati dagli encoder che si accumulano e portano ad un rapido ed inevitabile disallineamento della posizione effettiva con quella misurata. Per questa ragione, alle equazioni (3.28) vanno aggiunti dei disturbi aleatori:

$$
\begin{aligned}
x(k+1) &= x(k) + Tv(k)\cos\left(\vartheta(k) + \frac{\omega(k)T}{2}\right) + r_x(k+1) \\
y(k+1) &= y(k) + Tv(k)\sin\left(\vartheta(k) + \frac{\omega(k)T}{2}\right) + r_y(k+1) \\
\vartheta(k+1) &= \vartheta(k) + T\omega(k) + r_\vartheta(k+1)
\end{aligned} \tag{3.29}
$$

dove $r(k) = [r_x\, r_y\, r_\vartheta]$ sono disturbi gaussiani a media nulla, con matrice di covarianza Q simmetrica[3] e semidefinita positiva. Per aumentare la precisione e ovviare alla instabilità del metodo di integrazione diretta, sono necessarie altre misure. Nel caso del Nomad 150 queste misure supplementari derivano dai sonar che, se inserite in un vettore $y(k)$, permettono di eprimere l'obiettivo di questo lavoro: un algoritmo di stima che minimizzi la differenza tra posizione reale e stimata.

[3]Si considera questa matrice diagonale e, i primi due elementi della diagonale rappresentano l'incertezza sulla stima di posizione, mentre il terzo elemento rappresenta l'incertezza sull'orientamento.

$$\hat{X}_r(k+1) = \Lambda(\hat{X}_r(k), y(k)) \tag{3.30}$$

Il vettore contenente i dati dei sonar $Y(k) = [y_1, ..., y_{16}]'$ è dipendente dalla posizione del robot e dall' ambiente di lavoro:

$$Y(k) = h(k, X_r) + \omega(k) \tag{3.31}$$

con $h(k, X_r)$ che rappresenta la relazione esistente tra la posizione del robot e il rilevamento dati sonar; $\omega(k)$ è il rumore gaussiano a media nulla, con matrice di covarianza diagonale $R(k)$, utilizzata per modellare gli errori di misura.

Volendo relazionare quindi, la posizione del robot con le misure dei sonar, si deve ricercare una funzione $\Lambda$ che ricorsivamente nel tempo fornisca una stima della posizione del robot fondendo le (3.29) con le (3.31):

$$X_r(k+1) = \Lambda(X_r(k), Y(k+1)) \tag{3.32}$$

e per realizzare tale algoritmo si utilizza il filtro di Kalman esteso illustrato nel precedente paragrafo.

Quindi, sapendo che il robot si muoverà in un ambiente descritto da una mappa dell'ambiente quadrilatera con vertici delle pareti espressi rispetto al sistema di riferimento fisso, e sapendo che esso è in grado di rilevare la velocità delle ruote e la variazione dell'orientamento, è stata sviluppata la seguente idea: nota la posizione e l'orientamento del robot al tempo k ( $X_r(k)$ )e date le misure di velocità lineare e rotazionali, si può fare un'ipotesi sulla posizione che il robot assumerà al tempo *k+1* ($X_{r,ipotesi}(k+1)$). In questa nuova posizione ipotizzata dallo stimatore, si possono predire i valori che assumeranno i sonar. Al tempo $k+1$ poi, si rilevano i nuovi valori sonar e li si confrontano con quelli predetti al tempo $k+1$. In questo modo si confronta il risultato dell'integrazione diretta delle velocità con le misure dei sensori, riducendo continuamente potenziali errori di stima.

Esplicitando la (3.31):

$$h(k, X_r) \quad = \quad \begin{bmatrix} h_1(k, X_r) \\ \vdots \\ h_{16}(k, X_r) \end{bmatrix} \tag{3.33}$$

$$h_i = \sqrt{(S_{i,x} - x_p)^2 + (S_{i,y} - y_p)^2} \tag{3.34}$$

La (3.34) spiega l'origine della misura sonar e definisce il rapporto tra questa e la posizione del robot. Il termine $S_i$ indica la posizione del sensore i-esimo rispetto al sistema di riferimento assoluto che dipende da posizione e orientamento del robot. Le coordinate $(x_p, y_p)$ appartengono al punto generico frutto dell'intersezione tra una parete dell'ambiente e la retta ortogonale alla tangente alla circonferenza (del robot) nel punto di collocazione del sonar. Nella (3.34) si è omessa la dipendenza da k per semplicità di notazione.

Introducendo una nuova notazione, si puo' definire:

$$\hat{X}_r(k|k) \triangleq \left[\hat{x}(k|k)\ \hat{y}(k|k)\ \hat{\theta}(k|k)\right]'$$

Quindi nell'ottica della nuova notazione si può riscrivere la (3.32) come

$$\hat{X}_r(k+1|k+1) = \Lambda(\hat{X}_r(k|k), Y(k+1)) \tag{3.35}$$

Ad ogni passo l'algoritmo stima lo stato del robot e la relativa matrice di covarianza (indice dell'accuratezza della stima).

**Fase di predizione**

Conoscendo $X_r(\hat{k}|k)$ (stima al tempo k, dopo k misure sonar) e la matrice di covarianza

$$P(k|k) = E\left\{\left[\hat{x}_r(k|k) - \bar{\hat{x}}_r(k|k)\right]\left[\hat{x}_r(k|k) - \bar{\hat{x}}_r(k|k)\right]'\right\} \tag{3.36}$$

si può fare una predizione al tempo $k+1$

$$\begin{aligned} x(k+1|k) &= x(k|k) + Tv(k)\cos\left(\vartheta(k|k) + \frac{\omega(k)T}{2}\right) \\ y(k+1|k) &= y(k|k) + Tv(k)\sin\left(\vartheta(k|k) + \frac{\omega(k)T}{2}\right) \\ \vartheta(k+1|k) &= \vartheta(k|k) + T\omega(k) \end{aligned} \tag{3.37}$$

$$P(k+1|k) = E\left\{\left[\hat{x}_r(k+1|k) - \bar{\hat{x}}_r(k+1|k)\right]\left[\hat{x}_r(k+1|k) - \bar{\hat{x}}_r(k+1|k)\right]'\right\} \tag{3.38}$$

Mentre le (3.37) sono facilmente calcolabili, la (3.38) deve essere semplificata.

Si linearizzano[4] le (3.28) calcolando la matrice Jacobiana F(k):

$$F(k) \triangleq \left[\begin{array}{ccc} 1 & 0 & -Tv(k)\sin\left(\vartheta(k) + \frac{\omega(k)T}{2}\right) \\ 0 & 1 & Tv(k)\cos\left(\vartheta(k) + \frac{\omega(k)T}{2}\right) \\ 0 & 0 & 1 \end{array}\right]_{X_r = \hat{X}_r(k|k)} \tag{3.39}$$

e inserendola nella covarianza P(k+1|k) in quanto

$$x_r(k+1|k) = F(k)x_r(k|k) + \upsilon_{x_r}(k) \tag{3.40}$$

$$\rightrightarrows P(k+1|k) = F(k)P(k|k)F'(k) + Q(k,l) \tag{3.41}$$

dove $Q(k,l) \triangleq E\left\{\upsilon_{x_r}(k)\upsilon_{x_r}(k)'\right\}$ matrice simmetrica e semidefinita positiva dipendente dalla distanza percorsa $l$.

Le equazioni (3.37) e (3.41) possono essere utilizzate per predire la posizione futura al tempo k+1 del robot e la covarianza conoscendo le informazioni al tempo k. Con queste informazioni si può fare una predizione sulle misure dei sonar

$$\hat{Y}(k+1) = h(k, \hat{x}_r(k+1|k)).$$

Il valore di tale predizione sarà poi confrontato con i valori reali al tempo *k+1*.

Nella (3.34) erano state introdotte le coordinate dei sensori sonar $S_{i,x}$ e $S_{i,y}$. Nel dettaglio si calcolano come

$$\begin{aligned} S_{i,x} &= x + r\cos(\vartheta_0 + \vartheta) \\ S_{i,y} &= y + r\sin(\vartheta_0 + \vartheta) \\ S_{i,\vartheta} &= \vartheta_0 + \vartheta \end{aligned} \tag{3.42}$$

[4]Per tutti i passaggi intermedi fare riferimento a [1]

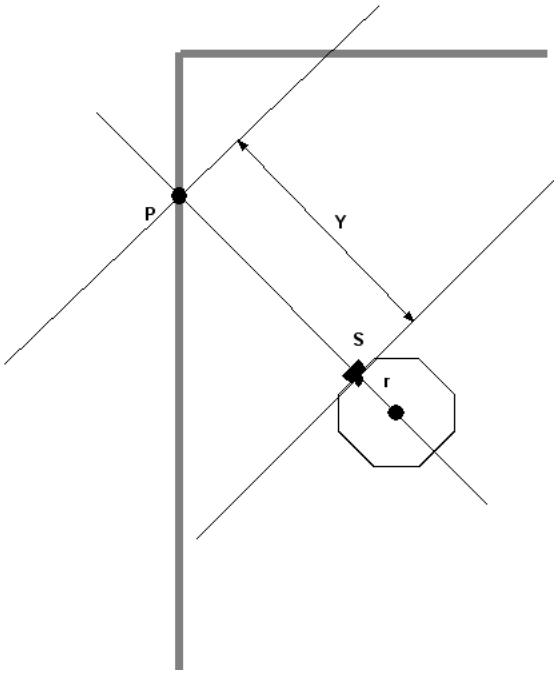


Figura 3.7: Punto d'intersezione tra la traiettoria del sonar e la parete

$\vartheta_0$ è l'angolo del sensore rispetto all'orientamento $\vartheta$=0 del robot.

Per capire quale sarà il punto della parete colpito dal segnale del sensore, si osservi bene la Figura 3.7.

Se una parete dell'ambiente ha vertici $P_1$ e $P_2$ si può trovare il coefficiente angolare $m_p$ e il termine noto $q_p$ della retta passante per il segmento $P_1P_2$ risolvendo il sistema seguente:

$$\begin{aligned} P_{1,y} &= m_p P_{1,x} + q_p \\ P_{2,y} &= m_p P_{2,x} + q_p \end{aligned} \tag{3.43}$$

Si considera che il sonar misuri la distanza lungo una retta ortogonale al suo orientamento che ha come incognita il termine noto $q_s$:

$$S_{i,y} = \tan(S_i, \vartheta) S_{i,x} + q_s$$

Le due rette trovate si intersecano in un punto P (che apparterrà alla parete) le cui coordinate si ottengono dalla risoluzione del sistema:

$$y_p = m_p x_p + q_p$$

$$y_p = \tan(S_{i,\vartheta})x_p + q_s \tag{3.44}$$

Quindi il valore del sonar i-esimo predetto sarà

$$h_i(k, x_r) = \sqrt{(S_{i,x} - x_p)^2 + (S_{i,y} - y_p)^2}.$$

**Fase di Stima**

In questa fase viene fatta la correzione sui valori predetti nella fase precedente. Al tempo $k+1$ si suppongono note le misure dei sonar, per cui si definisce la differenza tra valore reale e valore predetto

$$\tilde{Y}(k+1) \triangleq Y(k+1) - \hat{Y}(k+1) \tag{3.45}$$

Tale differenza verrà utilizzata per la correzione sulla stima. La matrice si covarianza di $\tilde{Y}$, che è un indice sull'errore di stima dei sonar, è pari a:

$$\begin{aligned} S(k+1) &\triangleq E\left\{\tilde{y}(k+1)\tilde{y}'(k+1)\right\} \\ &= E\left\{[y(k+1) - \hat{y}(k+1)][y(k+1) - \hat{y}(k+1)]'\right\} \end{aligned}$$

Questa matrice S, come la covarianza P per l'errore di stima della posizione, va semplificata e quindi anche qui si effettua una linearizzazione, ma questa volta sui valori $h(k, x_r)$

$$h(k, x_r) = \begin{bmatrix} h_1(k, x_r) \\ h_2(k, x_r) \\ \vdots \\ h_N(k, x_r) \end{bmatrix} = \begin{bmatrix} ((S_{1,x} - x_1)^2 + (S_{1,y} - y_1)^2)^{\frac{1}{2}} \\ ((S_{2,x} - x_1)^2 + (S_{2,y} - y_1)^2)^{\frac{1}{2}} \\ (\cdots) \\ ((S_{N,x} - x_1)^2 + (S_{N,y} - y_1)^2)^{\frac{1}{2}} \end{bmatrix} \tag{3.46}$$

$$= \begin{bmatrix} ((x_{r,x} + r\cos(\vartheta_1 + x_{r,\vartheta}) - x_1)^2 + (x_{r,y} + r\sin(\vartheta_1 + x_{r,\vartheta}) - y_1)^2)^{\frac{1}{2}} \\ ((x_{r,x} + r\cos(\vartheta_2 + x_{r,\vartheta}) - x_2)^2 + (x_{r,y} + r\sin(\vartheta_2 + x_{r,\vartheta}) - y_2)^2)^{\frac{1}{2}} \\ (\cdots) \\ ((x_{r,x} + r\cos(\vartheta_N + x_{r,\vartheta}) - x_N)^2 + (x_{r,y} + r\sin(\vartheta_N + x_{r,\vartheta}) - y_N)^2)^{\frac{1}{2}} \end{bmatrix}$$

Si realizza quindi una matrice H calcolata sul punto $X_r$

$$H \triangleq \begin{bmatrix} \frac{\partial h_1}{\partial x_{r,x}} & \frac{\partial h_1}{\partial x_{r,y}} & \frac{\partial h_1}{\partial x_{r,\vartheta}} \\ \frac{\partial h_2}{\partial x_{r,x}} & \frac{\partial h_2}{\partial x_{r,y}} & \frac{\partial h_2}{\partial x_{r,\vartheta}} \\ \dots & \dots & \dots \\ \frac{\partial h_N}{\partial x_{r,x}} & \frac{\partial h_N}{\partial x_{r,y}} & \frac{\partial h_N}{\partial x_{r,\vartheta}} \end{bmatrix} \tag{3.47}$$

dove i singoli termini sono dati da

$$\frac{\partial h}{\partial x_{r,x}} = \frac{m_p}{m_p \cos(\vartheta_i + x_{r,\vartheta}) - \sin(\vartheta_i + x_{r,\vartheta})} \tag{3.48}$$

$$\frac{\partial h}{\partial x_{r,y}} = \frac{1}{m_p \cos(\vartheta_i + x_{r,\vartheta}) - \sin(\vartheta_i + x_{r,\vartheta})} \tag{3.49}$$

e per $x_{r,\vartheta}$ ponendo

$$\begin{aligned} g &= m_p^2 \cos(\vartheta_i + x_{r,\vartheta})^2 - 2\sin(\vartheta_i + x_{r,\vartheta}) m_p \cos(\vartheta_i + x_{r,\vartheta}) + \\ &+ 1 - \cos(\vartheta_i + x_{r,\vartheta})^2 \end{aligned}$$

$$\begin{aligned} \frac{\partial h}{\partial x_{r,\vartheta}} &= \frac{q_p \cos(\vartheta_i + x_{r,\vartheta}) - x_{r,y} \cos(\vartheta_i + x_{r,\vartheta}) + m_p \cos(\vartheta_i + x_{r,\vartheta}) x_{r,x}}{g} + \\ &+ \frac{\sin(\vartheta_i + x_{r,\vartheta}) m_p q_p - m_p x_{r,y} \sin(\vartheta_i + x_{r,\vartheta})}{g} \end{aligned} \tag{3.50}$$

Con $m_p$ e $q_p$ ricavati nel (3.43) e (3.44). Dopo aver ricavato H, il calcolo[5] della matrice di covarianza *S(k+1)* si semplifica considerevolmente

$$\begin{aligned} S(k+1) &\triangleq E\left\{[y(k+1) - \hat{y}(k+1)][y(k+1) - \hat{y}(k+1)]'\right\} \\ &= H(k+1) P(k+1|k) H'(k+1) + R(k+1) \end{aligned}$$

La matrice di covarianza trovata è un indice sulla qualità del confronto effettuato tra le misure dei sonar. Adesso si devono calcolare soltanto i guadagni K(k+1) del filtro di Kalman. Note le matrici S(k+1) e P(k+1) le equazioni per la fase di stima sono:

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K_{k+1}^o \tilde{y}(k+1) \tag{3.51}$$

[5]Per tutti i passaggi intermedi si rimanda a [1]

$$
\begin{aligned}
P(k+1|k+1) &= [I - K^o_{k+1}H(k+1)]P(k+1|k) && (3.52)\\
K^o_{k+1} &= P(k+1|k)H'(k+1)\cdot \\
&\cdot \quad (R(k+1) + H(k+1)P(k+1|k)H'(k+1))^{-1} \\
&= P(k+1|k)H'(k+1)S^{-1}(k+1) && (3.53)
\end{aligned}
$$

Quindi, riassumendo, nella prima fase sono noti i dati al tempo k e si generano ipotesi sul valore che avrà lo stato del robot al tempo k+1. Nella seconda fase si eseguono le misure e le si confrontano con le predizioni fatte al tempo k. Ad ogni tempo k si ripercorre tutto l'algoritmo esposto.

## 3.2 Navigazione e aggiramento ostacoli

La realizzazione di robot autonomi implica la risoluzione in primis del problema della pianificazione del moto (robot motion planning). L'obiettivo principale della pianificazione del moto è quello di trovare tecniche grazie alle quali un robot possa trovare un percorso per muoversi da una posizione iniziale ad una finale assegnata.

Nel caso in questione, si e' realizzata una tecnica in grado di soddisfare delle particolari specifiche:

- evitare urti con ostacoli e pareti presenti nello spazio di lavoro
- mantenere un margine di sicurezza nei movimenti
- rendere l'algoritmo efficiente nel caso di ostacoli imprevisti

Per quanto riguarda i problemi riscontrati si possono citare:

- bassa qualità dei sensori(e dei conseguenti errori odometrici che si accumulano );
- tempo di computazione e acquisizione dati dai sonar abbastanza elevato(1,4 secondi);
- velocità di traslazione e rotazione limitate, in quanto oltre una certa soglia, l'EKF sbaglia;
- navigazione in prossimità delle pareti difficoltosa a causa dei valori minimi riportati dai sonar.

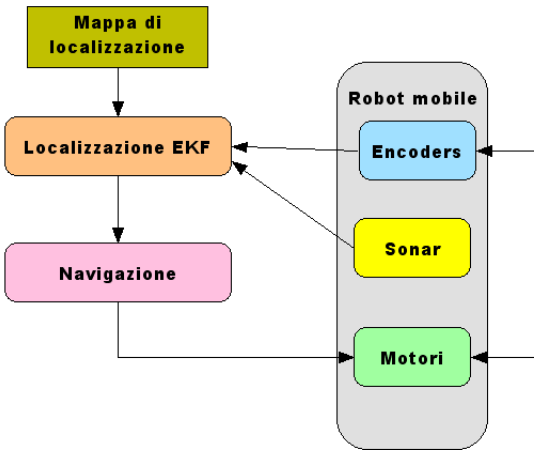


Figura 3.8: Struttuta dell'algoritmo di navigazione

L'algoritmo sviluppato in questa tesi è frutto di un'intuizione tanto semplice quanto efficace. Il funzionamento si basa vagamente sul metodo dei potenziali magnetici anche se poi, in pratica è molto diverso. E' adatto alla navigazione in ambienti parzialmente noti in quanto permette di aggirare ostacoli imprevisti in una mappa assegnata a priori al robot. Infatti tutto si basa sulla localizzazione. Se il robot conosce la sua posizione nella mappa data e se conosce la coordinata in cui si deve andare a posizionare, anche nel caso dell'introduzione di un ostacolo, riesce comunque a raggiungere il goal.

Ovviamente il limite forte dell'algoritmo è che in caso di troppi ostacoli la mappa di localizzazione diventa inutile in quanto il robot non ottiene più i dati predetti, o meglio ottiene errori sulla predizione troppo grandi che, di conseguenza vengono filtrati.

Si mostra nel dettaglio l'algoritmo di navigazione e una schematizzazione (Fig. 3.8).

Questa versione dell'algoritmo è stata sviluppata per una mappa dell'ambiente quadrilatera e se si volesse aumentare il numero di lati della stessa, bisognerebbe mettere in preventivo un tempo di computazione molto maggiore in quanto il robot dovrebbe calcolare l'incrocio dei raggi dei sonar con molti più lati (come già illustrato nel paragrafo sull'EKF per il Nomad).

L' universalità di tale sistema di navigazione è limitato soltanto dalla tipo di sensori usati (encoder e sonar), ma nulla vieta di modificarlo nel caso si voglia usare un altro modello di robot. La descrizione viene comunque data considerando il Nomad 150.

Si comincia impostando dal codice C++ una mappa quadrangolare

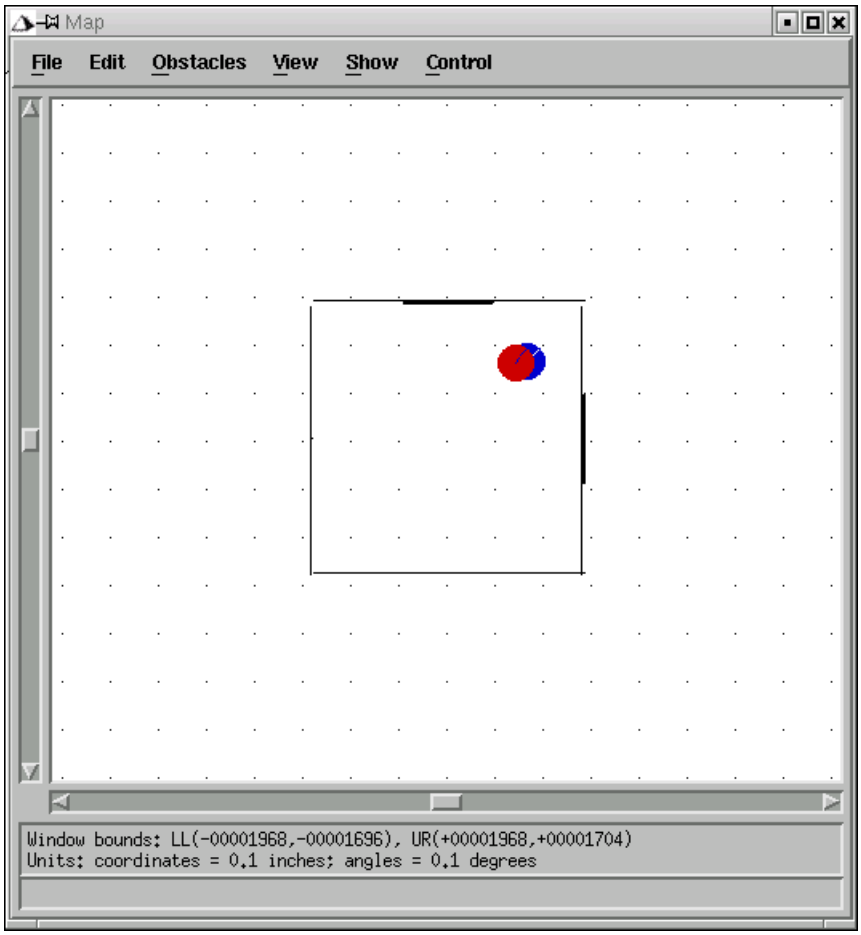


Figura 3.9: Screenshot dell'Nserver.

assegnando le coordinate dei vertici della stessa. Il sistema di riferimento è preso in modo arbitrario purchè l'origine si trovi all' interno della mappa, per una questione di praticità nella successiva misurazione della posizione raggiunta dal robot. In questa tesi l'origine è stata posizionata al centro della mappa, o meglio i vertici della stessa sono stati presi equidistanti dall'origine .in Figura 3.9 il pallino rosso indica indica la posizione stimata del robot nell'ambiente assegnato(quadrato, in questo caso).Il pallino blu sottostante è la posizione "reale" del robot quando si è nella modalità Simulatore.

Ora, appena si avvia il programma di navigazione - localizzazione, viene richiesta una coordinata finale (quella iniziale è l'origine, ma volendo si può far richiedere dal programma modificando debitamente il codice sorgente) e il robot, dopo aver eseguito l'operazione di HOME (che azzera gli encoder e porta le ruote e la torretta nella posizione base), si orienta verso la posizione finale, come mostrato in Figura 3.10.

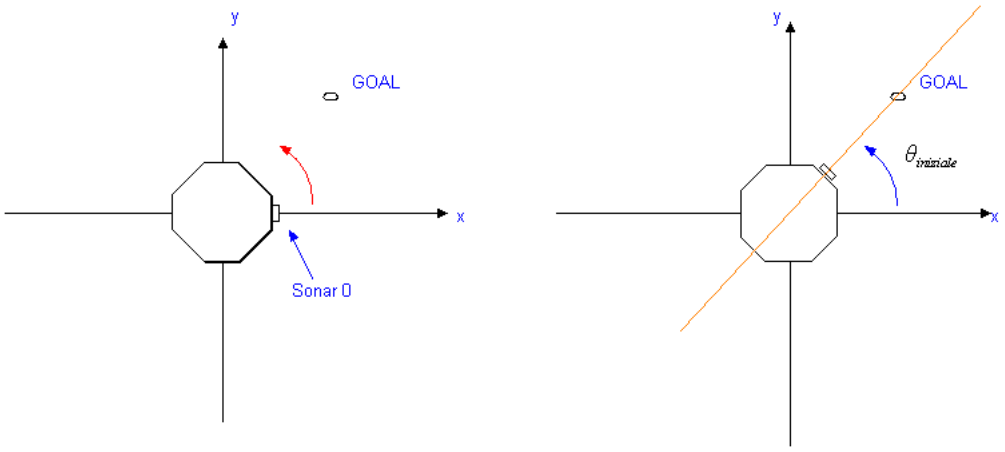


Figura 3.10: Fase di allineamento verso il goal che il robot effettua all' inizio della missione.

D'ora in avanti la posizione finale verrà chiamata GOAL, per intendere l'obiettivo prefissato.

Dopo essersi orientato, il robot verifica subito mediante i sonar se tra sè e il goal c'è un ostacolo e se sì, abilita una parte del codice che, in condizioni normali resta esclusa risparmiando sul tempo di computazione. Nel caso non vi siano ostacoli il robot comincia a muoversi in avanti verso il goal e, appena ci arriva vicino, inizia la fase di allineamento "fine" che verrà chiarita in seguito. Nel caso invece, che tra il robot e il goal vi sia un ostacolo che non permette il raggiungimento diretto, il robot comincia a muoversi comunque, ma appena si trova a circa dieci centimetri dall'ostacolo si ferma e comincia ad aggirarlo.

Il criterio che permette l'aggiramento dell'ostacolo è stato concepito in modo che questo possa avere qualsiasi profilo. Infatti si basa su una direzione di aggiramento sempre uguale finchè non si raggiunge il goal. Nello specifico, si ponga il caso in cui si cominci ad aggirare l'ostacolo verso sinistra. Il robot ruota torretta e ruote finchè davanti a se non ha campo libero (almeno per una ventina di centimetri), dopodichè ricomincia a traslare in avanti; adesso però, ad ogni ciclo di localizzazione, controlla se alla propria destra c'è campo libero. Appena ciò risulta vero, il robot comincia a ruotare a destra e se trova la direzione sgombra verso il goal, si dirige verso di esso, altrimenti se l'ostacolo aggirato ha una forma che presenti delle insenature, l'algoritmo porta a seguirne il profilo finchè esso non viene aggirato completamente. Nel caso l'ostacolo arrivasse a toccare la parete, il robot comincerebbe a seguire tutto il profilo della parete fino a raggiungere l'ostacolo dalla parte opposta. La descrizione può essere

meglio compresa se si osservano i disegni di Figura 3.11.

E' molto importante far notare che, ad ogni ciclo in cui il programma effettua la localizzazione e decide in che direzione far muovere il robot, viene effettuato un controllo sulla raggiungibilità del goal, cioè se la lunghezza della retta che congiunge la posizione attuale del robot con quella del goal è pari o minore a quella riportata dal sonar direzionato verso la posizione finale, allora il robot ferma tutto e va verso il goal.

Dopo aver raggiunto una coordinata nei pressi del goal (nello specifico sono 20 cm sull'asse dalla x finale e 20 cm dalla y finale) il robot comincia la fase di allineamento fine (funzione PRGOAL). In pratica il programma non fa altro che far allineare ruote e torretta a zero gradi rispetto al sistema di riferimento stabilito all'inizio (cioè il sonar 0 sarà parallelo all'asse delle ascisse e in direzione di questo crescente); poi calcola la distanza $x_{finale} - x_{attuale}$e si muove di quella distanza impartendo una velocità per un tempo opportuno. Dopodichè ruota di nuovo torretta e ruote in direzione parallela all'asse delle ordinate e verso di questo crescente (Fig. 3.12).

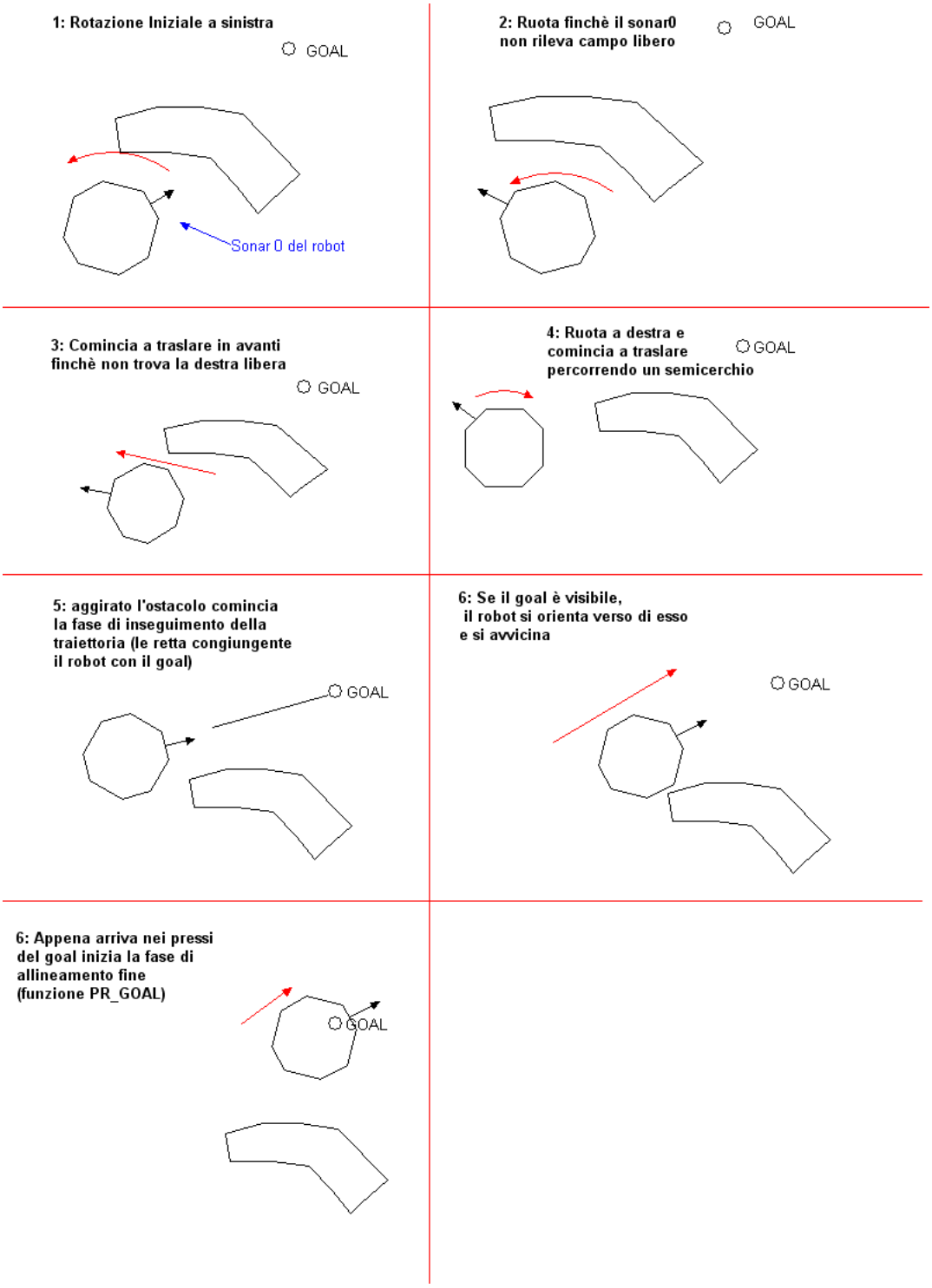


Figura 3.11: Fasi di aggiramento di un ostacolo tipico

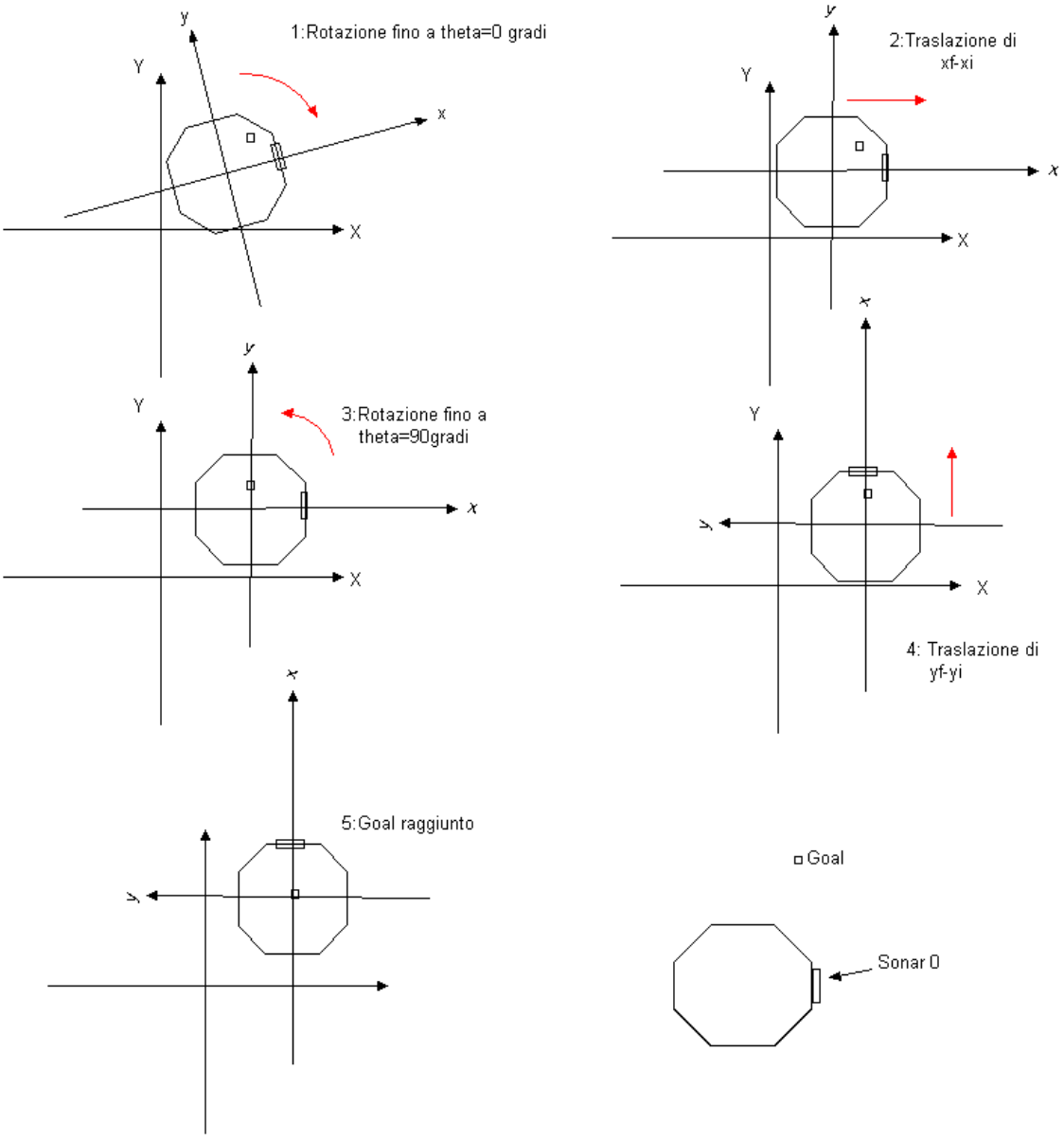


Figura 3.12: Fase di allineamento fine

# Capitolo 4

# Applicazioni sperimentali

## 4.1 L' Ambiente

Tutti gli esperimenti sono stati eseguiti al quarto piano dell'edificio di Ingegneria dell'Informazione dell'Università di Roma Tor Vergata.

In un primo tempo l'ambiente in cui il robot veniva fatto muovere era rettangolare, con pareti di cartone e dalle dimensioni abbastanza contenute di 140x400 centimetri. Lo spazio angusto però non permetteva di sfruttare le potenzialità del software in quanto non era possibile testare l'algoritmo di aggiramento degli ostacoli per raggiungere la coordinata finale. In un secondo tempo si è deciso di ampliare l'ambiente e si è creato un ring più grande nell'atrio sempre del quarto piano dell'edificio suddetto. Il nuovo ambiente era di dimensioni 348x348 cm con pareti in legno multistrato (composte da pannelli modulari) che hanno permesso al robot di ricevere segnali sonar più precisi (non avendo rumori come porte o termosifoni) e hanno permesso inoltre l'introduzione di ostacoli e varie discontinuità sul pavimento (di marmo lucido) che dovevano introdurre grandi errori sui sensori encoder. Tali discontinuità del suolo hanno permesso di saggiare la reale utilità e funzionalità dell'algoritmo implementato.

## 4.2 Il contesto sperimentale

Come spiegato in precedenza il robot è comandato da un computer portatile posizionato sulla torretta. Il software a corredo (Nserver), usato come server per il programma qui sviluppato (client), ha permesso di avere a schermo una interfaccia grafica che permetteva di vedere la posizione del robot all'interno del perimetro della mappa

(vedi Figura 3.9). Per quanto riguarda la localizzazione, il compito era di adattare al contesto reale un filtro di Kalman del tipo sviluppato in [1] . Infatti i risultati matematici visti nel paragrafo 3.1 si basano su un modello ipotetico. Si è dovuto, pertanto, prendere atto delle non idealità, studiare le fonti di errore, compensarle e minimizzarle. Ciò che interessa modellare per la localizzazione è la stima della posizione del robot, ottenuta con l'algoritmo, e la sua posizione reale, a prescindere dalla posizione desiderata.

I sensori sonar sono posizionati sulla torretta mobile e nel lavoro qui svolto, venivano ruotati insieme alle ruote. Il metodo di rilevamento utilizzato è molto simile a quello utilizzato dal corpo umano: se il corpo gira, gira anche la testa. In questo modo il sensore di riferimento (gli occhi nel caso del corpo umano e il SONAR 0 nel caso del Nomad 150) resta sempre frontale rispetto al movimento che si sta effettuando e quindi il codice del programma diventa più semplice, in quanto non si devono cambiare ogni volta i sonar n-esimi che si devono controllare per fare una manovra, per evitare un ostacolo. Inoltre facendo in questo modo non c'è mai il rischio che il robot, mentre si sta muovendo, non abbia un sonar frontale che permetta di recepire un ostacolo a cui sta andando incontro.

La singola distanza riportata da un sonar, non fornisce un'informazione affidabile, perchè ad essa è associata un' incertezza angolare di 22.5 gradi, poichè i sonar sono comunque soltanto 16 distribuiti sui 360 gradi del perimetro del robot. Per questa ragione si è preferito far ruotare il robot soltanto di quantità multiple di 22.5 gradi, per evitare ulteriori errori sulla navigazione.

Altro problema affrontato è stato quello del filtraggio dei dati non attendibili. E' stato detto che durante la navigazione del robot, vari ostacoli non previsti possono interferire con il processo di localizzazione. Riferendosi alla Figura 4.1 si supponga che in un certo istante nell'ambiente venga a trovarsi un ostacolo imprevisto e che questo venga rilevato dal sensore i-esimo; inoltre, si supponga che per il sensore j-esimo fosse prevista l'intercettazione dello spigolo A e che, per un qualche errore esso sia mancato.

Se si considera il termine d'innovazione del filtro di Kalman (valore sonar misurato - valore sonar predetto) per i 16 sonar nel suddetto istante, si possono valutare le discrepanze corrispondenti ai sonar i-esimo e j-esimo. Si ottiene, infatti, per l'ostacolo una misura molto inferiore a quella attesa in base alla mappa assegnata e quindi un'inno-

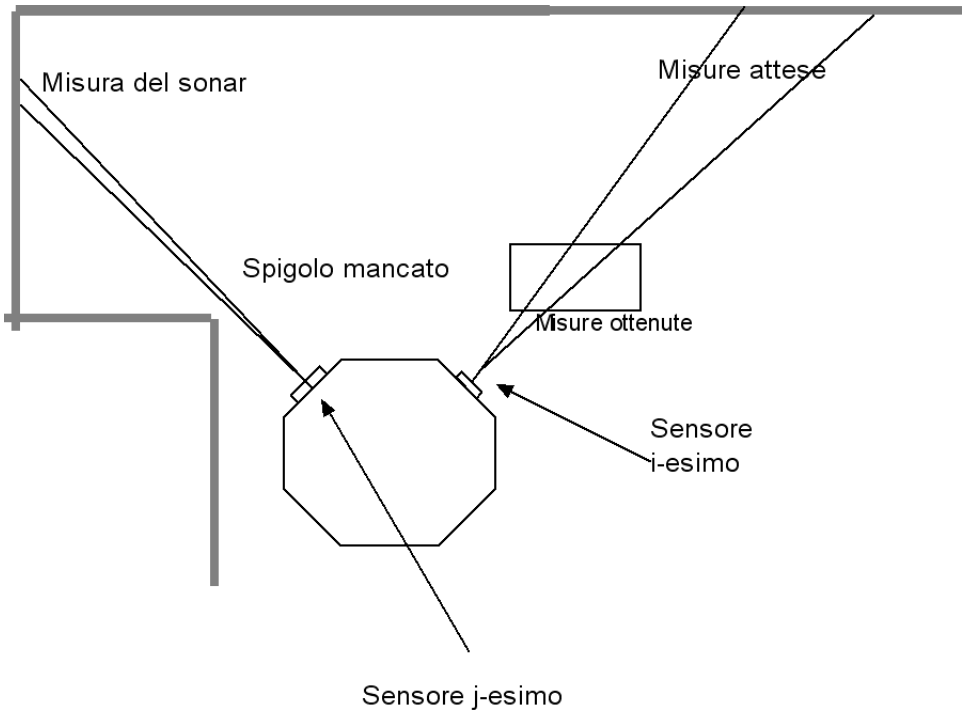


Figura 4.1: Esempio di errore sui sensori

vazione molto grande. Nel caso dello spigolo, si ha una misura molto maggiore a quella attesa e quindi un'innovazione negativa molto grande. Le misure i e j che vengono fornite al filtro di Kalman insieme alle altre misure per correggere la stima di posizione, possono compromettere le prestazioni del filtro, per cui vanno filtrate, in quanto portatrici di contributi non affidabili. Un modo semplice ed efficace per realizzare questo filtraggio è introdurre una soglia sull'innovazione che permetta di scartare le informazioni fallaci dei sonar in questione. La soglia è stata stabilita tramite metodo sperimentale, dopo aver condotto numerose prove e aver constatato l'errore medio riportato nel termine d'innovazione del filtro.

L'algoritmo sviluppato presuppone che il calcolo della posizione e dell'algoritmo di navigazione avvenga ad intervalli di tempo relativamente "piccoli", che dovrebbero essere determinati in gran parte dai limiti delle apparecchiature a disposizione. Il tempo di ciclo T del programma non è costante in quanto individuato dal tempo di computazione del filtro di Kalman e dalla navigazione. La minimizzazione di tale intervallo di tempo è stato uno dei problemi più difficili da risolvere in quanto ha determinato l'adattamento di tanti altri parametri come le velocità di traslazione e rotazione e la velocità di invio e ricezione segnali dei sonar.

Le operazioni più dispendiose in termini di tempo sono:

- la predizione delle letture sonar sulla mappa memorizzata
- i calcoli matriciali e, in particolare, le inversioni di matrici
- l'acquisizione dei dati sonar, che vengono effettuate sequenzialmente per evitare interferenze di un segnale inviato da un sonar, con la ricezione di quello successivo.

Dalle prove sperimentali, il miglior risultato ottenuto è stato di T=1,2 secondi, in quanto al di sotto la localizzazione tendeva a divergere in tempi brevi.

## 4.3 Le prove condotte

Le prove effettuate sono state molteplici e tutte volte a verificare l'efficienza del software presentato in questa tesi.

Data l'impossibilità di riportare grafici completi sul moto reale del robot e sull'effettiva convergenza al goal (coordinata desiderata), saranno riportati i risultati della stima di posizione in ogni ciclo di funzionamento con il filtro. Le prove senza il filtro di Kalman sono state condotte e hanno portato tutte ai risultati attesi: il robot infatti, in poco tempo perdeva la posizione e l'errore si accumulava sempre di più fino a farlo vagare nell'ambiente come se non avesse alcuna meta. Per l'impossibilità e la totale inutilità di graficare i movimenti del Nomad senza l'uso del filtro, si rimanda alle prove teoriche di [1] che contemplavano la sola integrazione odometrica. I risultati sono riportati in grafici matlab, in quanto ad ogni esecuzione del programma, venivano generati dei file di testo contenenti la posizione stimata ad ogni ciclo e, inoltre, a fine simulazione veniva sempre rilevata la coordinata stimata finale raggiunta e la posizione reale raggiunta. Le prove che sono state eseguite sono le seguenti:

- raggiungimento goal senza ostacoli;
- raggiungimento goal partendo da una posizione iniziale abbondantemente sbagliata rispetto a quella che il robot conosce all'inizio della prova;
- raggiungimento goal con interposizione di un ostacolo imprevisto;

- raggiungimento goal con ostacoli sul pavimento che introducono errori significativi sui sensori odometrici.

Tutte le prove precedenti sono state condotte anche senza filtro di Kalman, ma non se ne riportano i risultati in quanto l'errore di stima diventava subito grande e il programma di navigazione non ha mai portato il robot al goal.

Si descrive nel dettaglio ogni tipo di prova. La prima senza ostacoli è la più semplice ed è quella che ha fornito i risultati più fedeli , anche se di poco, mentre l'unico vantaggio riscontrato consiste soltanto nella velocità di allineamento che era più rapida rispetto alle altre prove .

La prova con partenza da posizione sbagliata è la più significativa al fine della verifica dell'efficienza del filtro di Kalman. Si è potuto constatare infatti, che, se il robot cominciava una nuova missione con partenza da una coordinata differente da quella iniziale attesa (ad esempio x=40cm e y=50cm anzichè x=0 e y=0) e l'orientamento era quello giusto, cioè non variato in base a quello atteso ($\vartheta$=0 gradi), bastavano 3 o 4 cicli di lavoro per far eguagliare il valore di stima a quello reale.

Il terzo tipo di prova è consistito nell'introdurre un bidoncino nell'ambiente di lavoro che si interponeva tra il robot e il goal. I risultati sono stati molto soddisfacenti e rispetto alla prova senza ostacoli l'errore di stima aumentava al massimo di 3-4 centimetri.

Il quarto ed ultimo tipo di prova è consistito nel mettere sul pavimento una serie di fogli di carta e cartoncino proprio sulla traiettoria che doveva compiere il robot e dai risultati ottenuti si è notato un grande errore di stima durante il passaggio del Nomad sulla carta, che poi è stato compensato, senza troppi problemi, dal filtro di Kalman appena il robot ricominciava a muoversi sul pavimento.

Le stesse prove, condotte senza filtro di Kalman non hanno condotto ad alcun risultato significativo, in quanto sia nell'aggiramento dell'ostacolo, sia nel passaggio sui fogli di carta, non vi era alcun tipo di correzione che permettesse di recuperare gli errori dell'integrazione odometrica.

Di ogni tipologia di esperimento sono stati condotte almeno 4 prove in cui il robot si doveva andare a posizionare almeno in ognuno dei 4 quadranti del sistema di riferimento. Nelle prossime pagine si riportano i grafici relativi a tutti i tipi di prova, per il raggiungimento del goal posizionato sulla coordinata (120cm,120cm). Il primo e il secondo grafico rappresenta, rispettivamente, il valore raggiunto dall'asse x e

dall'asse y rispetto al tempo. Il terzo grafico di ogni prova rappresenta proprio la posizione stimata rispetto al sistema di riferimento assoluto dell'ambiente.

## 4.4 I risultati

Si riportano i grafici della stima di posizione per ognuna delle tipologie di prova:

- Senza ostacolo
- Con ostacolo
- Con partenza da posizione sbagliata
- Con fogli di carta sul pavimento

L'andamento della traiettoria stimata dal robot è fortemente disturbata da rumore sui sensori sonar e, ovviamente da rumore sugli encoder. Rispetto alle simulazioni condotte nella [1], il movimento del robot, pur essendo nella realtà molto fluido, quando viene graficato in base ai dati registrati dell'algoritmo di stima, non solo è rappresentato da una spezzata (in quanto l'algoritmo è discreto), ma è anche molto incostante (se il robot sta fermo cioè, la posizione stimata oscilla continuamente intorno a quella reale di circa 15cm). La soglia scelta per considerare il goal raggiunto era di $\pm 2cm$ sia sull'asse delle x che su quello delle y.

Alla fine di ogni prova condotta è stata misurata la posizione raggiunta dal robot e confrontata con quella riportata a schermo.

I risultati ottenuti sono stati:

1. Prova senza ostacoli: posizione raggiunta (123,140); posizione stimata (122,138).
2. Prova con ostacolo: posizione raggiunta (115, 124); posizione stimata (119,120).
3. Prova con partenza sbagliata: posizione raggiunta (129,132); posizione stimata (124,124).
4. Prova con rumore sugli encoder: posizione raggiunta (128, 125); posizione stimata (123,117).

In media il punto che si è raggiunto alla fine di ogni prova era lontano dal goal circa 10-12cm.

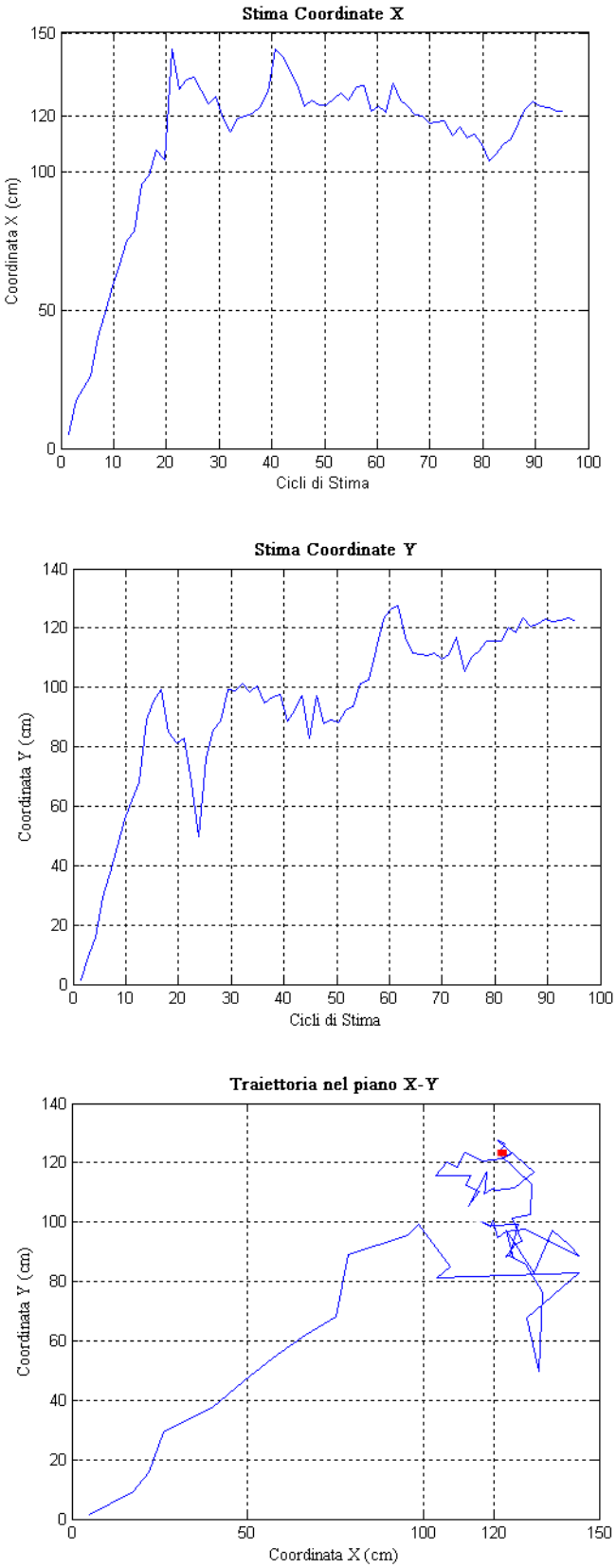


Figura 4.2: Andamento della stima con filtro dell' asse X, dell'asse Y e traiettoria stimata nel piano.Primo caso in assenza di ostacoli.

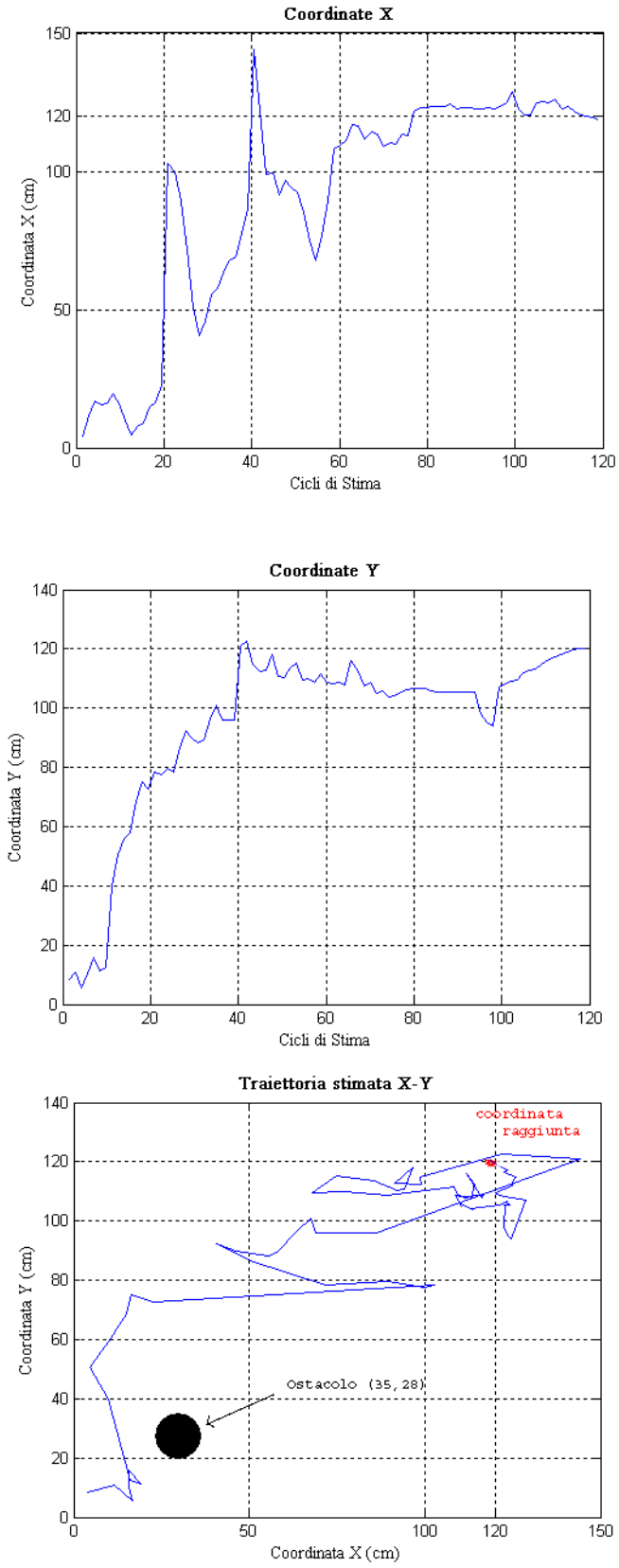


Figura 4.3: Stima con filtro dell' asse X, dell'asse Y e traiettoria stimata .Secondo caso: esperimento con ostacolo.

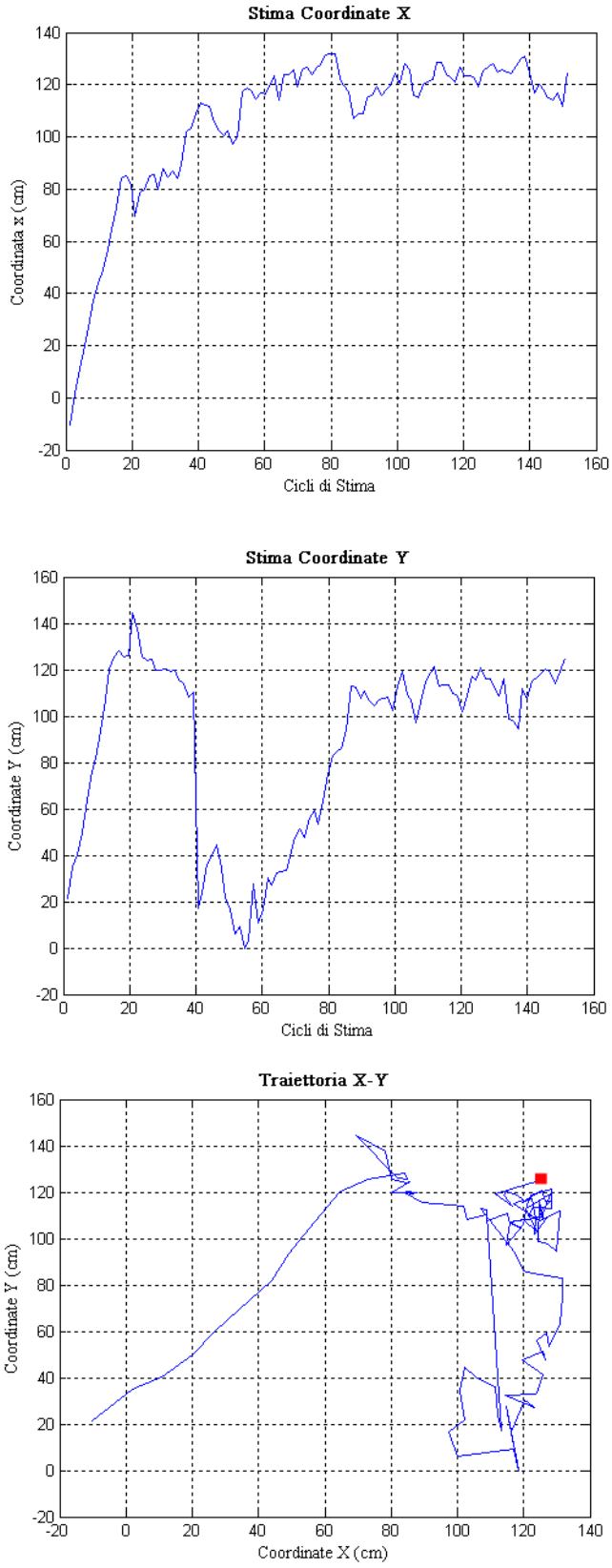


Figura 4.4: Andamento della stima con filtro dell' asse X, dell'asse Y e traiettoria stimata nel piano.Terzo caso: esperimento con partenza da posizione sbagliata.

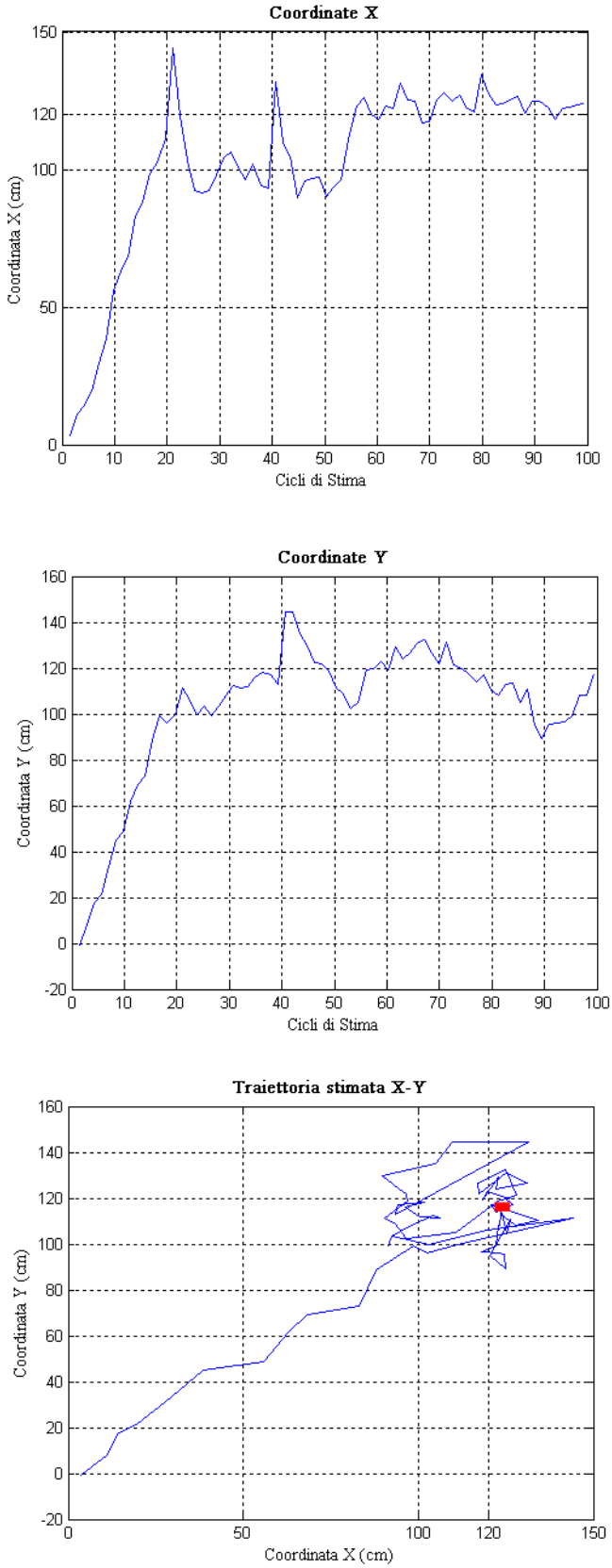


Figura 4.5: Stima con filtro dell' asse X, dell'asse Y e traiettoria stimata . Quarto caso: esperimento con rumore sui sensori odometrici.

# Capitolo 5

# Conclusioni

L'applicazione sviluppata si è verificata funzionante e, considerando la bassa qualità dei sensori a disposizione, anche efficiente.

Il filtro di Kalman utilizzato per la localizzazione è stato ottimizzato per un uso sul robot reale, e si è dimostrato essere una soluzione validaper il risolvere il problema della fusione sensoriale.

L'algoritmo di navigazione è stato sviluppato da zero senza l'utilizzo di algoritmi già esistenti. Pur non avendo alla base una teoria rigorosa, ma soltanto una logica semplice, esso permette al robot di giungere ad una posizione desiderata dell'ambiente di lavoro, con l'unico vincolo di conoscere semplicemente la propria posizione attuale (particolare richiesto anche dagli altri algoritmi di navigazione).

Gli errori di misura hanno provocato forti oscillazioni nella stima della posizione che però, grazie alla dinamica lenta del robot, sono state filtrate in modo naturale e quindi il risultato è stato un movimento abbastanza fluido del Nomad nell'andare a posizionarsi sulla coordinata finale. La stima dell'orientamento è la più delicata, in quanto dipende dai sensori sonar che sono caratterizzati da una bassissima risoluzione. Questa fragilità nella stima dell'angolazione delle ruote si è ripercossa nelle prova effettuate, in particolare in quelle con partenza da una posizione errata (posizione iniziale del robot diversa da quella stimata inizialmente). Si è notato, nel dettaglio, che anche se le coordinate iniziali (x e y) erano molto diverse da quelle stimate, se l'angolazione era esatta, lo stimatore recuperava la posizione in pochi cicli di lavoro. Nel caso in cui, invece, si partiva da posizione e orientamento errato, lo stimatore non riusciva a correggere lo stato e il sistema divergeva.

Un problema molto influente che non è stato posssibile risolvere ha riguardato la memorizzazione delle traiettorie percorse dal robot du-

rante le prove, che sarebbero risultate molto utili al fine di un confronto con le posizioni stimate e, quindi, con le traiettorie stimate. Queste ultime sono state memorizzate e graficate e permettono una visuale parziale del lavoro svolto. Uno sviluppo futuro potrebbe comprendere l'utilizzo di un mezzo di acquisizione video (una webcam ad esempio) che riprenda dall'alto il percorso del robot e lo digitalizzi in modo da trarne la traiettoria compiuta nel tempo. Altri sviluppi futuro potrebbero riguardare un upgrade della componentistica a disposizione come l'adozione di un maggior numero di sensori sonar con una maggiore risoluzione. Inoltre si dovrebbe implementare un algoritmo che permetta l'uso del filtro anche in ambienti non quadrilateri.

Infine per quanto riguarda l'algoritmo di navigazione si potrebbe considerare l'implementazione di un algoritmo che crei una mappa degli ostacoli in modo da avere una memoria dell'ambiente in cui il robot si sta muovendo per ottimizzare un successivo passaggio in un punto già visitato, proprio come avviene per gli algoritmi SLAM (simultaneous localization and mapping). A tal proposito potrebbe essere interessante un utilizzo del programma di navigazione qui sviluppato con l'algoritmo SLAM implementato in [12] e [13].

# Bibliografia


[1] F. Forni , [2004],"Filtro di Kalman Esteso per la Localizzazione di Robot Mobili",Tesi di Laurea in Ingegneria Informatica,Università di Roma Tor Vergata.

[2] G. Welck G. Bishop, [2004], "An Introduction to the Kalman Filter", University of North Carolina at Chapel Hill ,NC, USA.

[3] Nomadic Technologies, Inc. [1997] , *"Nomad 200* Hardware *Manual"* , Mountain View, CA, USA.

[4] Nomadic Technologies, Inc. [1997], *"Nomad 150- User's Manual" , Mountain View, CA, USA.*

[5] Progetto PRASSI, [2000], "Definizione di algoritmi per la gestione delle mappe", Realazione stato di avanzamento al 31/12/2000, Accordo ENEA-MURST L.95/95 "Calcolo parallelo con applicazioni alla robotica", Università di Tor Vergata.

[6] Progetto PRASSI, [2000], "Requisiti, Specifiche e Stato dell' Arte degli algoritmi di fusione sensoriale", Realazione stato di avanzamento al 31/12/2000, Accordo ENEA-MURST L.95/95 "Calcolo parallelo con applicazioni alla robotica", Università di Tor Vergata.

[7] Progetto PRASSI, [2000], "Sviluppo di algoritmi di fusione sensoriale e gestione di mappe", Relazione stato di avanzamento al 31/12/2000, Accordo ENEA-MURST L.95/95 "Calcolo parallelo con applicazioni alla robotica", Università di Tor Vergata.

[8] S. Pagnottelli,[2002], "Attività svolta presso gli impianti ENEA della Casaccia nell'ambito del progetto Prassi", Università di Roma Tor Vergata, Progetto PRASSI.

[9] G. Galati, M. Naldi, G. Pavan, [2002], "Teoria dei fenomeni aleatori: fondamenti e prime applicazioni ai modelli stocastici", Volume I, Texmat.

[10] G. Galati, M. Naldi, G. Pavan, [2002], "Teoria dei fenomeni aleatori: Modelli stocastici e Metodi Statistici", Volume I, Texmat.

[11] F. Romanelli, "Robot Nomad 150: nozioni di base", Laboratorio di Automatica (Ing. F. Martinelli ).

[12] F. Romanelli , [2005], "Filtri particellari per la localizzazione e la navigazione di robot mobili in ambienti non noti privi di landmark",Università di Tor Vergata, Roma.

[13] L.Spinello, [2005], "Risoluzione al problema dello SLAM tramite filtri particellari e filtro di Kalman esteso",Università di Tor Vergata, Roma.

[14] J. C. Latombe, [1991], "Robot motion planning", Kluwer Academic Publishers, Boston,Dordrecht,London.

[15] E. Fornasini, G. Marchesini, [1994], "Appunti di teoria dei sistemi", Edizioni Libreria Progetto.