Tecniche di auto-localizzazione robotica mediante l'uso di webcam

Valerio Mantini

Indice

Capitolo 1

Introduzione

L'evoluzione della scienza robotica ha portato i robot moderni a raggiungere autonomia e mobilità. L'autonomia è la capacità del robot di eseguire i propri compiti senza l'intervento umano. La mobilità è la capacità di un robot di potersi muovere liberamente nel mondo.

Il grado di autonomia è strettamente legato all'utilizzo che viene fatto del robot. La mancanza di essa, se ci si trova in ambito industriale, non rappresenta un problema nel caso in cui l'azione da svolgere sia sempre la stessa e non ci sia alcun bisogno di prendere decisioni.

I robot autonomi invece sono impiegati quando ci si trova in situazioni in cui non è possibile stabilire a priori le sequenze di azioni da compiere per raggiungere un determinato obiettivo.

Per fornire al robot queste due capacità, autonomia e mobilità, dobbiamo equipaggiarlo con strumenti che gli permettano di analizzare l'ambiente circostante.

Il costante sviluppo della tecnologia ha portato ad ottenere sorprendenti ri-

sultati nell'ambito della visione artificiale. Oltre a sensori ed encoder, infatti, la visione artificiale riveste un ruolo di notevole importanza nella robotica in quanto permette di automatizzare ed integrare nel robot una vasta gamma di processi tipici della percezione visiva.

La visione del robot può essere definita come il processo di estrazione, caratterizzazione e interpretazione delle informazioni provenienti dalle immagini di un mondo tridimensionale. Un sistema di visione non deve essere indipendente dal resto del sistema, ma deve interagire con gli altri moduli del robot, scambiando e analizzando solo le informazioni strettamente necessarie, dato il notevole carico computazionale richiesto dal problema in questione.

In ogni caso, nonostante siamo ben lungi dall'avvicinarci alla complessità e alle prestazioni dell'occhio umano, ma dato il costante sviluppo delle tecniche di elaborazione immagine e il costante aumento delle velocità di elaborazione dei calcolatori elettronici, assisteremo ad un costante sviluppo della visione artificiale che presto raggiungerà risultati sempre più soddisfacenti.

La navigazione attraverso la visione artificiale, quindi, ha il compito di rendere un robot autonomo ed in grado di muoversi da una posizione ad un'altra nell'ambiente che lo circonda.

Possiamo riassumere il problema della navigazione di un robot in tre semplici domande:

- Dove sono?
- Dove devo andare?
- Come ci vado?

Nei capitoli a seguire cercheremo di dare una risposta soddisfacente alla prima domanda, la quale, implicitamente, ci aiuterà a rispondere alle altre due. L'obiettivo principale di questo lavoro è stato quello di permettere al robot di capire dove si trova, analizzando delle immagini acquisite tramite webcam, riconoscendo all'interno di esse forme note.

Come tecnica di riconoscimento è stata utilizzata la **trasformata di Hough** la quale si è rivelata uno strumento matematico di grande affidabilità e robustezza. L'estrazione delle rette infatti è avvenuta con una soddisfacente precisione, nonostante la inevitabile presenza di rumori e disturbi.

Le maggiori difficoltà si sono verificate a causa della forte sensibilità dell'obiettivo della webcam alla variazione della luminosità nell'ambiente.

Capitolo 2

Descrizione hardware e software del Nomad 150

2.1 Il Nomad 150

Il Nomad 150 prodotto dalla Nomadic Technologies è un robot mobile, dotato di tre ruote disposte su due assi che permettono un movimento traslatorio ed uno rotatorio. Le ruote effettuano contemporaneamente movimenti di rotazione e traslazione, permettendo una notevole agilità di movimento al robot stesso.

Il robot è costituito da una base ed una torretta mobile. Nella base si trovano i tre motori, dei quali due hanno il compito di far muovere il robot, mentre il terzo di far ruotare la torretta.

Nei motori sono presenti degli encoder che permettono di determinare la posizione relativa del robot. Il controllo dei motori è effettuato tramite un microprocessore Motorola MC68008/Asic, che permette una velocità massi-

Figura 2.1: Robot Nomad 150

ma di traslazione di 0.51 m/s e una di rotazione di 45 gradi/s.
Il Nomad possiede due sistemi di sensori, costituiti da sensori ultrasonici il primo, e tattili (bumper) il secondo. I sensori ad ultrasuoni sono sedici e possono rilevare ostacoli ad una distanza di circa 45 cm con una risoluzione di 22,5 gradi. I sensori tattili sono disposti su due circonferenze a due diversi livelli di altezza, ogni livello ha dieci bumpers, come si può notare nella fig.1.2: Ł'alimentazione è fornita da tre batterie a piombo secco da 12 V/12 Ah.

Il robot necessita del supporto di un portatile per poter svolgere i suoi compiti, ed il collegamento può essere effettuato con un cavo seriale in due modi, tramite una porta Host o tramite porta Console. La porta Host si usa quando si comunica con il robot tramite il programma Nserver, che funziona anche da simulatore, per far eseguire programmi scritti in linguaggio C.
La porta Console invece si utilizza per programmare il robot tramite un programma di emulazione terminale, ma in questo caso i comandi che possono

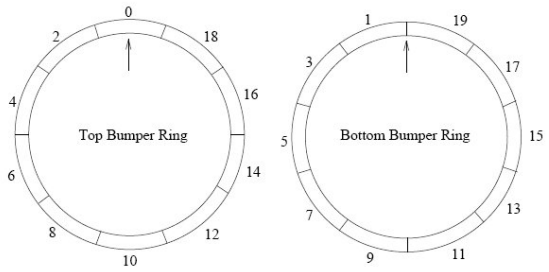

Figura 2.2: I sensori tattili

essere trasmessi sono molto limitati.

Un ulteriore possibilità per il controllo del movimento è data dall'uso del Joystick a cui è dedicata una porta sul robot. Il Joystick in dotazione è a due assi e permette il controllo dei motori di rotazione e di traslazione nonché della rotazione della torretta.

2.2 Il software

Il software fornito con il robot Nomad 150 è Nserver. Tramite questo programma è possibile monitorare i dati forniti dai sensori e dagli encoder.

Il programma Nserver ha due modalità di funzionamento. La prima è quella di semplice visualizzazione della posizione del robot nell'ambiente, tramite l'utilizzo di un'interfaccia grafica. La seconda è quella di simulazione dell'ambiente, con l'eventuale aggiunta di ostacoli di qualunque forma, allo scopo di testare i programmi implementati prima del loro utilizzo.

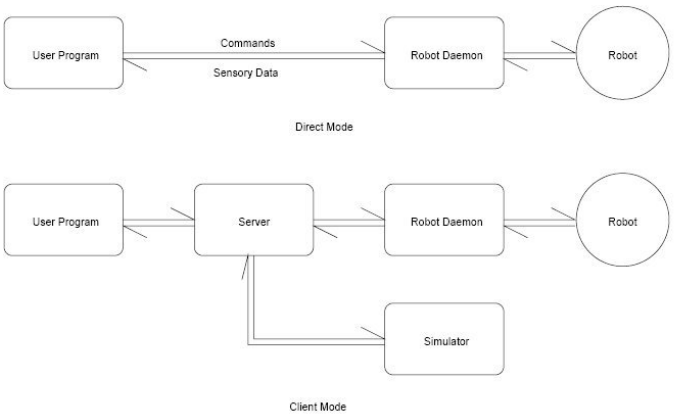

Figura 2.3: Metodologie di comunicazione

L'utente ha a disposizione una interfaccia grafica che permette di vedere il robot nel mondo circostante e ha la possibilità di aggiungere ostacoli di qualunque forma per simulare al meglio i programmi scritti.
Il programma Nserver è stato sviluppato per funzionare con qualsiasi versione del sistema operativo Linux, i cui programmi, scritti in linguaggio C, possono essere compilati con il compilatore *gcc* nel seguente modo:

```
gcc -o esempio esempio.c Nclient.o
```

Questo comando si usa quando si vuole utilizzare il programma Nserver come Server, dove Nclient.o è il file oggetto contenente le librerie del Nomad 150. Nel caso invece si voglia eseguire il programma direttamente sul robot il file oggetto da includere è Ndirect.o.
Ogni programma da compilare per poter utilizzare le funzioni della libreria del robot deve avere nell'intestazione

#include Nclient.h

. Le possibilità di comunicazione sono mostrate in figura 2.3.

2.3 I componenti utilizzati

Per gli esperimenti in laboratorio è stato usato un notebook ASUS con processore Pentium 4 e velocità 2 GHz, 256 MB di RAM. Il software è stato implementato in ambiente Linux, con la distribuzione Mandrake, compatibile con il programma di gestione Nserver.
La webcam utilizzata per la visione artificiale era una Philips ToUcam Pro 740 dotata di un sensore CCD (Charged Coupled Device) con risoluzione massima di 640x480 pixel (VGA) e una profondità a di colore di 24 bit. La lente interna è una 6mm e può raggiungere un framerate di 30 fps, alla risoluzione di 320x240.

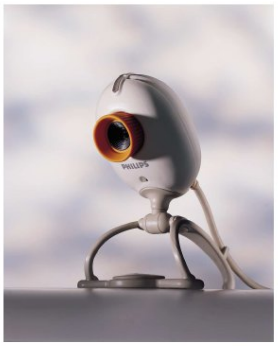

Figura 2.4: Philips ToUcam pro 740

2.4 Implementazione di un programma per l'Obstacle Avoidance

Uno degli obbiettivi della robotica mobile è quello dell'*obstacle avoidance*, cioè quello di navigare in un ambiente ignoto individuando e quindi evitando gli ostacoli che si presentano lungo il cammino.

Il programma che viene mostrato di seguito sfrutta i sonar del Nomad 150, che vengono utilizzati per rilevare la distanza a cui si trovano gli ostacoli e per mantenere sempre una minima distanza prefissata da essi, evitando in questo modo l'impatto.

Nel momento in cui il sonar frontale, ovvero quello orientato nella direzione del movimento, rileva un ostacolo lungo il cammino, il robot inizia a ruotare verso destra finché il sonar non trova più, nella sua direzione, l'ostacolo.

A questo punto viene effettuata una rotazione nel verso opposto rispetto alla precedente, per tornare lungo la direzione che il robot stava percorrendo.

La funzione principale del programma, cioè quella che permette l'aggiramento dell'ostacolo è riportata di seguito:

```
void aggira_dx (void) {
int i=0,a=1;
while(a){
pr(30,200,200); // rotazione verso destra per evitare l'ostacolo
wait_for_stop(STATE_VEL_STEER);
gs();
i=i+20;
```

```
while((State[STATE_SONAR_0]>200)
   &&(State[STATE_SONAR_12]>20)   // controllo continuo sui sonar
&& State[STATE_SONAR_14]>15) {
pr(50,0,0);
wait_for_stop(STATE_VEL_TRANS);
gs();
if(State[STATE_SONAR_12]>20){
pr(30,-200,-200); // rotazione verso sinistra per riprendere il cammino
wait_for_stop(STATE_VEL_STEER);
i=i-20; }
}
if (i==0 && (State[STATE_SONAR_0]>200) ) a=0; }
}
```

Nella figura 2.5 possiamo notare, attraverso la grafica del simulatore Nserver come il robot abbia aggirato un ostacolo presente nel suo cammino.

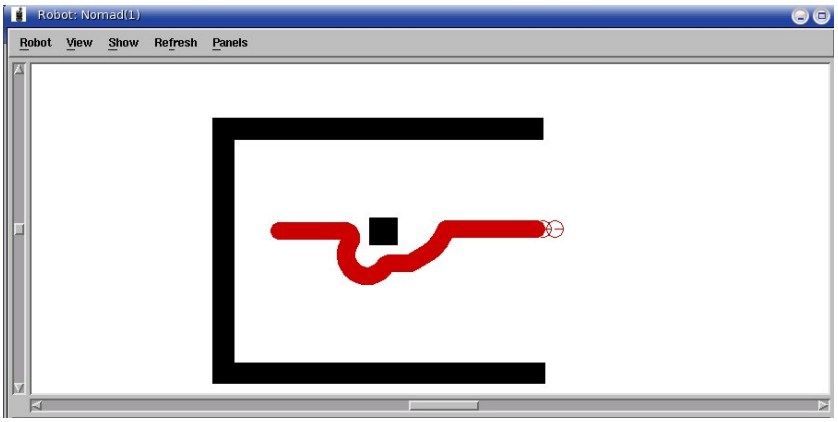

Figura 2.5: Obstacle avoidance

Capitolo 3

La visione artificiale

3.1 L'elaborazione delle immagini

Il compito principale della visione robotica consiste nell'analizzare le immagini acquisite dalla webcam. Questo compito viene effettuato in sei passi:

- Percezione
- Pre-elaborazione
- Segmentazione
- Descrizione
- Riconoscimento
- Interpretazione

Nella prima fase, la *percezione*, viene fornita una immagine dell'ambiente di lavoro. Nella fase di *pre-elaborazione* avviene il filtraggio dell'immagine, necessario per ridurre il rumore e per migliorarne i particolari.

Nella *segmentazione* l'immagine viene suddivisa in oggetti di interesse. Nella fase di *descrizione* avviene il calcolo delle caratteristiche utilizzabili per differenziare un oggetto da un altro. Nella fase di *riconoscimento* viene identificato l'oggetto (in questo caso il Landmark). Infine nell'*interpretazione* viene interpretata l'informazione per conferire un significato all'oggetto riconosciuto.

Le informazioni visive vengono convertite in segnali elettrici da sensori ottici. Se campionati spazialmente e quantizzati in ampiezza, questi segnali forniscono un'immagine digitale.
Possiamo pensare di rappresentare l'immagine bidimensionale in uscita dalla webcam con una funzione f(x,y), dove x e y rappresentano le coordinate spaziali (cioè sul piano dell'immagine) e il valore di f in ogni punto è proporzionale alla luminosità (intensità) dell'immagine nel punto.
Nel nostro programma avremo quindi un array di dimensione 320x240, con all'interno il valore di intensità di ogni pixel dell'immagine. L'immagine viene acquisita dalla webcam attraverso i comandi:

```
ioctl(webcam,VIDIOCMCAPTURE,grab buf);
ioctl(webcam,VIDIOCSYNC,grab buf);
```

Nella prima linea di codice si effettua l'acquisizione dell'immagine corrente (parametro VIDIOCMCAPTURE passato alla funzione ioctl) da parte del dispositivo richiamato come primo parametro da ioctl (webcam). L'immagine viene trasferita in memoria all'interno di un array di elementi di tipo "unsigned char". La seconda linea permette la sincronizzazione tra la Webcam e la memoria del PC per il processo di acquisizione dati.

3.2 Filtraggio

Le immagini dalla webcam vengono acquisite generalmente nel formato YUV420P che è un formato che permette una buona compressione delle immagini a scapito della qualità. Il formato YUV420P ha tre canali, uno per la luminosità e due per i colori.
Questo formato però si presta poco all'elaborazione, quindi è necessario un opportuno filtraggio per rendere l'immagine sia più snella, in termini di memoria, sia più adatta all'interpretazione ed al riconoscimento, convertendola in un formato in scala di grigi.
A questo punto è possibile applicare il filtro di Sobel per delimitare i contorni delle immagini e migliorare il contrasto degli elementi contenuti in essa.

Il filtro di Sobel è un operatore gradiente che analizza, per ogni pixel, il suo intorno 3x3, cioè gli otto pixel contigui. Se il valore della luminosità del pixel, e quindi il valore del gradiente, è maggiore di una prefissata soglia, il valore della luminosità viene posto uguale a 255, cioè bianco. In caso sia minore viene lasciato invariato.
Questo processo permette di eliminare gran parte delle informazioni poco utili dell'immagine che aumenterebbero soltanto i tempi di calcolo.
Questo filtro, comunque, non è ancora sufficiente a delineare completamente i contorni degli oggetti che si trovano in una immagine. Dato quindi un valore di soglia del grigio opportuno, se il valore di un pixel supera questa soglia viene posto uguale a 255 (bianco), altrimenti a 0 (nero).
A questo punto abbiamo un'immagine pronta per essere analizzata nella ricerca del landmark in quanto abbiamo eliminato una buona componente

rumorosa e le informazioni poco utili al fine del riconoscimento.

3.3 Riconoscimento

Come già accennato, in una immagine catturata con la telecamera sono di particolare interesse le tecniche di riconoscimento degli oggetti.
La parte più essenziale e significativa del comportamento degli esseri umani è l'abilità nel riconoscere ciò che li circonda, infatti siamo in grado di riconoscere altre persone, lettere dell'alfabeto, oggetti familiari e tante altre cose. Quello che fa il nostro cervello è semplicemente analizzare un'immagine 2-D fornita dai recettori sulla retina, e sorprendentemente riesce a riconoscere lo stesso oggetto in infinite forme e angolazioni.
Molto più difficile è far riconoscere ad un robot anche l'oggetto più semplice in quanto le tecniche per farlo sono ancora in via di sviluppo e presentano molte difficoltà. Le reti neurali prendono spunto dal funzionamento dei neuroni del cervello umano per riprodurre via software gli stessi meccanismi.
Il riconoscimento è un processo di identificazione, quindi la funzione degli algoritmi per il riconoscimento è quella di identificare ogni oggetto segmentato in una scena e di assegnare un identificatore (es. chiave, pinza...) a quell'oggetto.
Nella maggior parte dei casi gli stadi di riconoscimento operano nell'ipotesi che le immagini devono essere acquisite in una geometria visiva nota, che solitamente è perpendicolare allo spazio di lavoro.
Tra le numerose tecniche in uso per il riconoscimento nella visione artificiale ne verranno approfondite due: *la trasformata di Hough* e il *template*

matching.

3.3.1 La trasformata di Hough

La trasformata di Hough è una tecnica che permette di riconoscere particolari configurazioni di punti presenti nell'immagine, come segmenti, curve o altre forme prefissate. Questa trasformata è un tipico operatore globale che permette quindi di individuare forme descritte in forma analitica ed è parzialmente insensibile alle occlusioni oltre che al rumore.
La sua utilità consiste nell'isolare determinate forme dei contorni all'interno dell'immagine. Il maggior vantaggio di questa trasformata consiste nella sua tolleranza alla mancanza di completezza dei contorni e alla presenza di disturbi.
Il principio fondamentale è che la forma cercata può essere espressa tramite una funzione nota che fa uso di un insieme di parametri. Una particolare istanza della forma cercata è quindi completamente precisata dal valore assunto dall'insieme di parametri.
La trasformata di Hough è applicabile a qualsiasi funzione della forma $g(x, c) = 0$, con 'x' vettore delle coordinate e 'c' vettore dei coefficienti. Supponiamo che, data un'immagine di n punti su un piano XY vogliamo determinare sottoinsiemi che giacciono su linee rette. Prendendo in considerazione un punto (x_i, y_i) , attraverso esso passano infinite rette nella forma $y_i = a \cdot x_i + b$ per diversi valori di a e b. Riscrivendo però l'equazione come $b = -x_i \cdot a + y_i$ e analizzando il piano ab abbiamo trovato l'equazione di una singola retta passante per il punto di coordinate (x_i, y_i) .

Inoltre, anche un secondo punto (x_j, y_j) avrà una linea nello spazio dei parametri ad esso associata e tale linea intersecherà quella associata a (x_i, y_i) in (a', b') dove a' è la pendenza e b' l'intercetta della linea contenente sia (x_i, y_i) che (x_j, y_j) sul piano XY. I punti situati su tale linea avranno 'linee', nello spazio dei parametri, che si intersecano in (a', b') . In seguito si suddivide lo

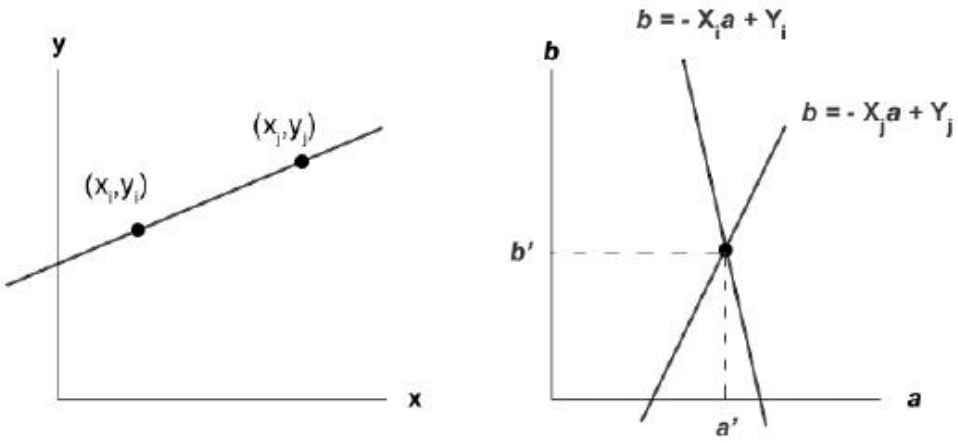

Figura 3.1: Spazio dei parametri

spazio dei parametri in celle accumulatrici scegliendo dei valori massimi per i due parametri che possano contenere i valori previsti. Inizialmente tutte le celle sono poste uguali a zero. Per ciascun punto (x_k, y_k) sul piano dell'immagine, troviamo i vari valori di b risolvendo l'equazione con tutti i valori della suddivisione consentiti di a. Per ogni coppia (a, b) trovata aumentiamo di 1 il valore della cella accumulatrice.

Al termine del procedimento un valore M nella cella $A(i, j)$ corrisponde a M punti sul piano XY appartenenti alla retta $y = a_i \cdot x + b_j$. Questo procedimento, su un'immagine di n punti implica $n \cdot K$ calcoli, dove K è il numero

di suddivisioni dell'asse a.

Quando però la retta da rappresentare è in posizione verticale la pendenza si avvicina all'infinito causando dei problemi all'algoritmo. Una soluzione efficace per ovviare a questo inconveniente consiste nel passare alla rappresentazione polare di una retta, mostrata in figura 3.2, ovvero $x \cdot cos\theta + y \cdot sin\theta = \rho$.

Quindi M punti giacenti sulla linea $x \cdot cos\theta + y \cdot sin\theta = \rho$ daranno luogo a M

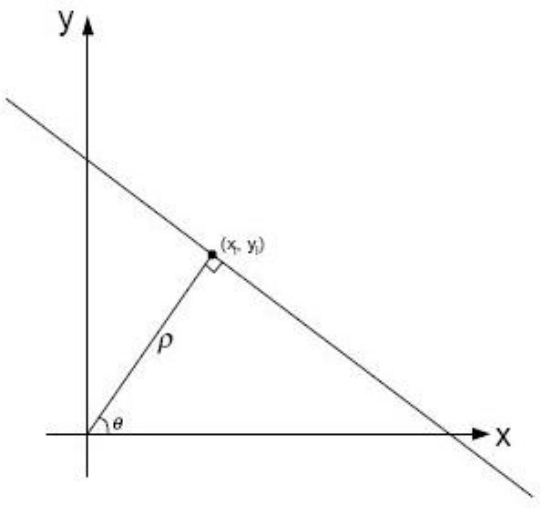

Figura 3.2: Rappresentazione polare di una retta

curve sinusoidali che si intersecano in (θ_j, ρ_j) nello spazio dei parametri. A questo punto è possibile riempire le celle accumulatrici risolvendo l'equazione in ρ incrementando θ .

Nel caso del cerchio, la cui funzione è: $(x - c_1)^2 + (y - c_2)^2 = c_3^2$, si può procedere nello stesso modo visto per le rette. La differenza principale è che ora abbiamo 3 parametri c1, c2 e c3 che danno origine ad uno spazio tridi-

mensionale (celle cubiche A(i,j,k)).
Il procedimento richiede di incrementare c1 e c2 e ricavare il valore di c3 che soddisfa l'equazione del cerchio aggiornando l'accumulatore corrispondente alla cella associata alla terna (c_1, c_2, c_3). Ovviamente la complessità computazionale cresce notevolmente con il numero dei parametri e dei coefficienti usati.

Nella figura 3.3 viene messo in risalto il grafico della trasformata di Hough di una retta nel piano $\rho\theta$. Come si può notare una retta corrisponde ad un fascio di curve sinusoidali che passano tutte per lo stesso punto di coordinate (ρ, θ) i quali parametri corrispondono alle coordinate della retta nella rappresentazione polare.

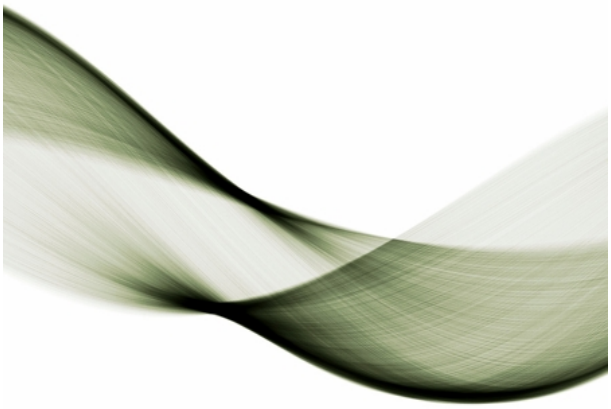

Figura 3.3: Retta nel piano $\rho\theta$

E' stata inoltre proposta [11] una generalizzazione della trasformata di Hough che permette di individuare oggetti di forma qualunque.

3.3.2 Template matching

Nel contesto dell'interpretazione delle immagini, l'obiettivo centrale è quello di riconoscere gli oggetti presenti all'interno di esse. 'Riconoscere' un oggetto significa verificare che nell'immagine è presente un'istanza dell'oggetto di cui è memorizzata una rappresentazione.

Il primo e più semplice approccio al problema del riconoscimento nel caso delle immagini è quello di confrontare direttamente l'immagine dell'oggetto cercato con l'immagine in esame. Questo tipo di approccio si definisce *template matching* e si basa sulla misura della similarità esistente tra il prototipo dell'oggetto da riconoscere (template) e l'immagine (o una parte di essa).

Siccome non si conoscono a priori le regioni in cui l'istanza può presentarsi, è necessario confrontare il template con tutte le sottoparti dell'immagine che hanno le stesse dimensioni di esso.

A questo scopo, il template viene fatto scorrere sequenzialmente sull'intera immagine, valutando per ogni possibile posizione la similarità con la regione dell'immagine.

Denominando $T(x, y)$ il template, con $(x, y) \in DT$ dominio di definizione del template, e con $I(i, j)$ l'immagine di dimensioni NRxNC, T viene confrontato con tutte le regioni $I(u + x, v + y)$, con $(x, y) \in DT$, tali che:$1 \leq u + x \leq NR$ e $1 \leq v + y \leq NC$.

Quando si considerano immagini binarie invece, una misura efficiente si può ottenere considerando una versione modificata del template T' per effettuare

una corelazione.

$$T'(x,y) := \begin{cases} 1 & \text{se } T(x,y) = 1 \ , \\ -1 & \text{se } T(x,y) = 0. \end{cases} \tag{3.1}$$

In questo caso la correlazione raggiunge il massimo valore quando $T(x,y)$ e $I(u+x, v+y)$ combaciano perfettamente.

Il valore massimo (template e porzione dell'immagine perfettamente coincidenti) è pari al numero N_1 di pixel uguali a 1 nel template, mentre il valore minimo (template e porzione dell'immagine perfettamente speculari) è pari a $-N_0$, dove N_0 è il numero di pixel uguali a 0 nel template.

Se quindi si sommasse N_0 al risultato della correlazione, questa varierebbe tra 0 ed A, dove $A = N0 + N1$ è il numero di pixel contenuti nel template.

Capitolo 4

Auto-localizzazione robotica

4.1 Tecniche di auto-localizzazione

L'auto-localizzazione è uno dei compiti più critici per un robot mobile autonomo. Il problema dell'auto-localizzazione consiste nel rendere il robot capace di riconoscere autonomamente la propria posizione, in ambienti non strutturati, e rappresenta uno dei filoni di ricerca tra i più attivi ed importanti. Si può solitamente rispondere alla tipica domanda "Where am I?" attraverso una combinazione di tecniche di misura delle posizioni assolute e relative.

A seconda di come viene codificata la posizione del robot e di quale metodo di misura della posizione viene usato, possiamo identificare diverse categorie di localizzazione. Una prima suddivisione, quindi, può essere fatta nel modo seguente:

- Localizzazione dipendente dalla codifica della posizione

  - Localizzazione in coordinate cartesiane;
  - Localizzazione su mappa topologica;

- Localizzazione dipendente dal metodo di misura
  - Localizzazione relativa (Ricostruzione Odometrica)
  - Localizzazione assoluta (Landmark naturali/artificiali)

Uno dei metodi di localizzazione maggiormente in uso è quello della ricostruzione odometrica, che consiste nel calcolare il numero di giri compiuti dalle ruote del robot in un lasso di tempo unitario, mediante encoder calettati sui motori di trazione e stimare quindi lo spazio percorso.
L'odometria è tuttavia affetta da errori che possono facilmente causare nel tempo una deriva della posizione stimata del robot rispetto a quella reale, con conseguente perdita totale dell'orientamento.

4.2 Navigazione tramite landmark

Per equipaggiare un robot mobile di capacità di apprendimento, i recenti sviluppi della robotica mobile hanno portato all'utilizzo di tecniche di visione robotica per analizzare immagini dell'ambiente circostante e pianificare un percorso realizzabile verso la destinazione.

I landmark (o *marker*) sono tratti distintivi dell'ambiente che il robot può riconoscere con il proprio sistema sensoriale. Generalmente hanno una posizione fissa e nota, relativamente alla quale il robot può localizzare se stesso.

I landmark, generalmente, sono scelti in maniera tale da essere facilmente riconoscibili (ad es. contrastano con lo sfondo) e possono essere di due tipi:

- Landmark naturali
- Landmark artificiali

I *landmark naturali* sono oggetti già pre-esistenti nell'ambiente (porte, lampade, tavoli...) ed attraverso questi si può lavorare in ambienti strutturati come corridoi, ospedali, uffici... Con questo tipo di oggetti è importante effettuare una fase preliminare di addestramento (training) in cui il robot viene messo in posizioni prescelte che gli consentono di memorizzare i landmark dove acquisisce informazioni sia visive (via telecamera) che di prossimità (via sonar).
Quando il robot si trova nelle vicinanze di una posizione di addestramento, cerca in memoria i landmark che dovrebbero essere visibili dalla posizione attuale, dopodiché calcola l'aspetto che questi dovrebbero avere da quel punto di vista. A questo punto, facendo un matching delle informazioni, confronta l'immagine acquisita con l'immagine da lui elaborata e calcola la distanza tramite il sensore di prossimità.
Il problema principale di questo tipo di landmark consiste nel riuscire a riconoscere gli oggetti nell'ambiente in cui si trovano. E' necessario quindi un buon algoritmo di "pattern recognition" che riesce a confrontare le immagini acquisite dalla telecamera con altre già immagazzinate in memoria.
Le difficoltà maggiori nel riconoscimento delle immagini si riscontrano quando le condizioni di luminosità sono diverse dal normale nell'ambiente circo-

stante, o quando vengono acquisite da un'angolazione poco favorevole che determina una distorsione dell'oggetto.

Il riconoscimento tramite *landmark artificiali* è notevolmente più semplice in quanto sono costruiti per avere un contrasto ottimo con il resto dell'ambiente.

Per quanto riguarda la grandezza e la forma, esse sono conosciute a priori dal robot. L'utilizzo di questi landmark permette anche l'inserimento di informazioni aggiuntive quali possono essere: la successiva istruzione che il robot deve seguire, dati sulla posizione attuale, oppure informazioni che possono essere interpretate da altri sensori, ad esempio un codice a barre per un sensore laser.

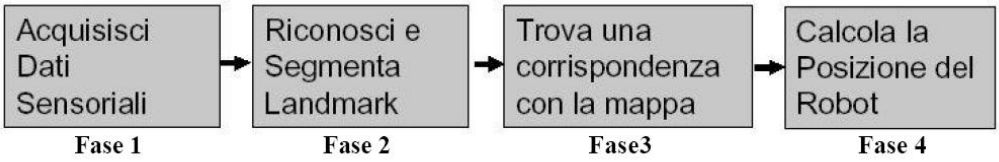

Figura 4.1: Fasi di localizzazione mediante landmark

L'uso dell'odometria permette di semplificare in maniera significativa il problema dell'acquisizione dei landmark da parte del robot in quanto, fornendogli già una stima approssimativa della posizione e orientamento, fa sì che la ricerca dei tratti distintivi sia ristretta ad un'area più limitata.

Allo stesso tempo però, anche la localizzazione tramite visione permette di correggere gli errori dell'odometria; infatti tramite soli due landmark il robot è in grado di ricavare informazioni sulla direzione e sulla distanza di questi,

dando così un riferimento assoluto agli encoder posti sulle ruote. Questo sistema così organizzato riesce ad ottenere una precisione del centimetro.

4.3 Strategia di ricerca landmark

La struttura generale del metodo di navigazione proposto consiste principalmente in tre componenti:

- Ricerca landmark
- Localizzazione landmark
- Interpretazione landmark

La *ricerca landmark* determina ogni possibile landmark nello spazio di lavoro del robot mobile. Nello specifico di cui trattiamo, acquisita una immagine monoculare, l'elaborazione sull'immagine con la tecnica dell'edge thinning, permette la rilevazione dei contorni del landmark e quindi anche delle informazioni contenute in esso.
La *localizzazione landmark* determina la posizione relativa del landmark rispetto al robot e provvede a fornire le informazioni geometriche necessarie per la trasformazione di inversione prospettica perché avvenga il riconoscimento.
L'*interpretazione landmark* estrae ogni informazione contenuta nell'immagine, cioè nel landmark, ed eventualmente immagazzina le informazioni aggiuntive contenute nello stesso.

4.3.1 Ricerca landmark

Come già accennato, lo scopo di questa fase è quello di trovare i landmark all'interno di un ambiente. Per questo è necessario che debbano avere determinate caratteristiche:

1. Sebbene nello spazio di lavoro possano esserci differenti tipi di landmark, ognuno di essi deve avere una forma predefinita con un colore che abbia un forte contrasto con lo sfondo.

2. Al fine di evitare errori di processing dell'immagine ogni landmark deve avere un tratto distintivo, che permetta al robot di distinguerlo da altri oggetti dell'ambiente che possano essere simili ad esso.

3. La forma del landmark deve essere di facile interpretazione.

I landmark considerati nell'esperienza effettuata, come possiamo vedere dalla figura 5.2, prima dell'elaborazione sull'immagine, ed in figura 4.2 dopo la trasformazione monocromatica, hanno la forma di rettangolo. I quattro angoli della figura forniscono le informazioni necessarie per ricavare la posizione del robot.

In questa fase quindi il robot determina quando un landmark entra nella visuale della webcam, attraverso il riconoscimento dell'immagine processata con l'uso della trasformata di Hough e trova i vertici della figura nell'immagine attraverso le rette ricavate. Effettuata la trasformazione, assumendo che (ρ_i, θ_i) siano i parametri di linea del contorno i del landmark:

$$L_i : \rho_i = x_i \cdot cos\theta_i + y_i \cdot sin\theta_i. \tag{4.1}$$

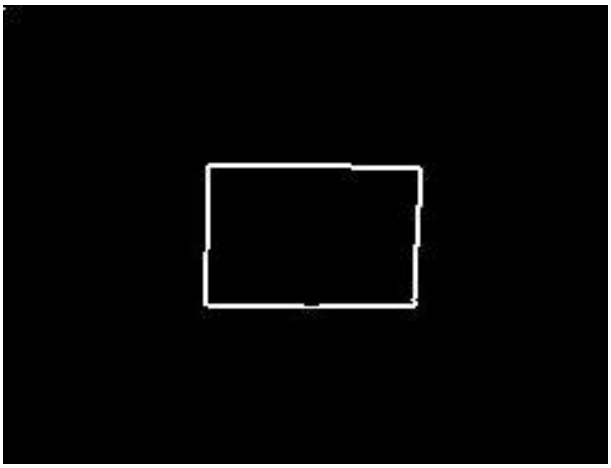

Figura 4.2: Landmark dopo una trasformazione monocromatica

Dal momento che il numero di regioni discrete dello spazio dei parametri è finito e L_i può solo essere stimato approssimativamente, applichiamo il minimo errore quadratico medio per migliorare l'accuratezza di L_i. Il procedimento adatto è quello di determinare ρ e θ tali che l'errore quadratico

$$E^2 = \sum_{i=1}^{n} (\rho_i - x_i \cdot cos\theta_i + y_i \cdot sin\theta_i)^2$$

sia minimo. Con n è indicato il numero di punti (x_i, y_i) dell'immagine.

4.3.2 Localizzazione landmark

Una volta ottenute le equazioni delle rette, possiamo ricavare i vertici del landmark, che verranno utilizzati nella *localizzazione landmark* calcolando le intersezioni tra esse. Lo scopo di questa procedura, come descritto da [2], è di individuare la posizione del robot (x_c, y_c, z_c) rispetto al landmark appena osservato. Andiamo a definire due sistemi di coordinate per descrivere le

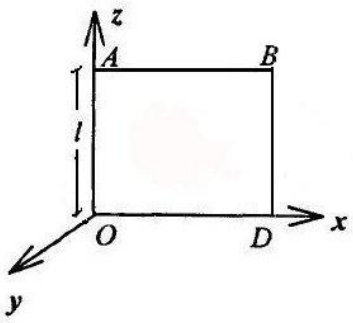

Figura 4.3: Sistema coordinate robot

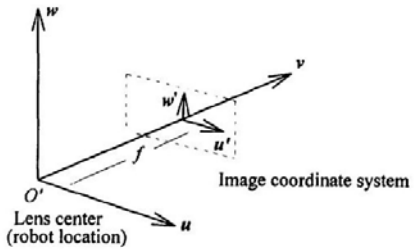

Figura 4.4: Sistema coordinate landmark

relazioni tra il robot e il landmark: un sistema $O - x - y - z$ di coordinate del landmark e un sistema $O' - u - v - w$ di coordinate del centro della lente della webcam. Il piano $u' - w'$ corrisponde al piano dell'immagine collocato a distanza $v = f$ dal sistema di coordinate della webcam.

Di conseguenza un generico punto P di coordinate landmark (x, y, z) e coordinate webcam (u, v, w) è in relazione al corrispondente punto $P'(u', w')$ nel piano immagine dalle:

$$(x, y, z)^t = R \cdot (u, v, w)^t + (x_c, y_c, z_c)^t \tag{4.2}$$

$$(u, v, w) = \lambda(u', f, w') \tag{4.3}$$

dove si è considerato $\lambda = \overline{O'P}/\overline{O'P'}$ come fattore di scala e R una matrice di rotazione $3x3$. Dalle proprietà geometriche della figura 4.3 possiamo ottenere:

$$\overline{AB} = \overline{OD} \tag{4.4}$$

$$\overline{AO} = l \tag{4.5}$$

$$\overline{AB} \perp \overline{AO}. \tag{4.6}$$

Sostituendo le coordinate immagine dei punti O,A,B e C nelle suddette proprietà, possiamo ricavare dalle seguenti equazioni, la distanza focale f e i fattori di scala λ_O, λ_A, λ_B, λ_C .

$$f = \left[\frac{(K_B u'_B - K_A u'_A) \cdot (K_O u'_O - K_A u'_A) + (K_B v'_B - K_A v'_A) \dot{(} K_O v'_O - K_A v'_A)}{(A_B - K_A) \cdot (K_O - K_A)} \right]^{1/2} \tag{4.7}$$

$$\lambda_O = \frac{1}{\left[(K_A u'_A - u'_O)^2 + (K_A f - f)^2 + (K_A v'_A - v'_O)^2 \right]^{1/2}} \tag{4.8}$$

$$\lambda_A = K_A \lambda_O \quad , \quad \lambda_B = K_B \lambda_O \quad , \quad \lambda_D = K_D \lambda_O \tag{4.9}$$

dove K_A, K_B e K_D sono costanti. A questo punto possiamo ricavare dall'equazione (3.3) le coordinate dei punti A,B,D,O nel sistema di riferimento dell'immagine e la lunghezza del lato $\overline{OD}$. Indicando rispettivamente con L_A, L_D, L_O le distanze tra i punti A, D ed O e il centro della lente della webcam O', avremo:

$$L_O^2 = \lambda_O^2((u'_O)^2 + f^2 + (v'_O)^2) = x_c^2 + y_c^2 + z_c^2 \tag{4.10}$$

$$L_A^2 = \lambda_A^2((u'_A)^2 + f^2 + (v'_A)^2) = x_c^2 + y_c^2 + (z_c - l)^2 \tag{4.11}$$

$$L_D^2 = \lambda_D^2((u'_D)^2 + f^2 + (v'_D)^2) = (x_c^2 - (\overline{OD})^2) + y_c^2 + z_c^2 \tag{4.12}$$

Risolvendo le equazioni possiamo ricavare le coordinate del robot (x_c, y_c, z_c):

$$z_c = \frac{1}{2l} \cdot (L_O^2 - L_A^2 + l^2) \tag{4.13}$$

$$x_c = \frac{1}{2\overline{OD}} \cdot (L_O^2 - L_D^2 + (\overline{OD})^2) \tag{4.14}$$

$$y_c = \sqrt{L_O^2 - x_c^2 + y_c^2} \tag{4.15}$$

4.4 Metodi di posizionamento

Come già accennato, al robot mobile deve essere fornita la mappa dell'ambiente in cui si trova, con le informazioni sulla posizione dei landmark. Scopo della localizzazione è, quindi, quello di trovare la posizione assoluta del robot nel sistema di coordinate della mappa in relazione alla disposizione dei marker trovati.

Idealmente, il robot deve determinare la propria posizione in base a quella dei landmark riconosciuti, ma per raggiungere questo obiettivo non basta calcolare la distanza tra il robot e il marker. Si deve, infatti, anche tenere conto della distorsione di prospettiva di quest'ultimo rispetto alla visuale frontale, e quindi ideale, della forma.

Per ottenere un algoritmo robusto è quindi utile utilizzare due landmark in modo da fornire sufficienti informazioni al robot e permettergli di stimare con più precisione la posizione.

Come si può notare in figura 4.6, il posizionamento angolare del landmark α corrisponde alla somma di due contributi: il primo è l'angolo di rotazione α_{pan}, il secondo è α_{offset}, cioè l'angolo tra il centro dell marker ed il centro

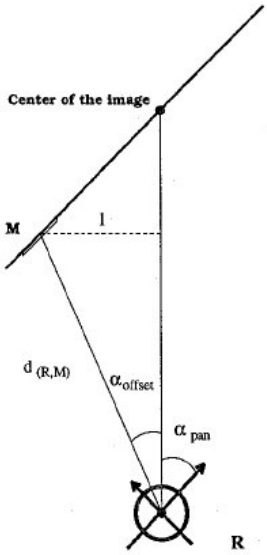

Figura 4.5: Orientamento geometrico del landmark

dell'immagine, che può essere calcolato dalla distanza l tra i due punti nello spazio dell'immagine. La distanza stimata tra il robot e il centro del landmark è indicata con $d_{R,M}$.

La posizione del robot può essere individuata anche tramite l'utilizzo di due landmark. Stimando le due distanze $d_{R,M1}$ e $d_{R,M2}$ e calcolando l'intersezione tra le due circonferenze centrate nel marker e di raggio pari alle distanze appena stimate otteniamo due punti che potrebbero corrispondere alla posizione del robot. Una di esse verrà scartata in base al valore dell'angolo α.

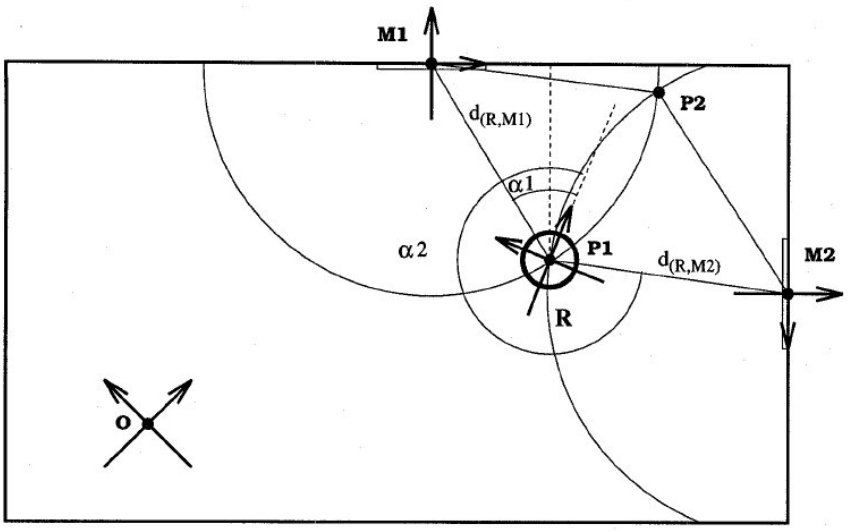

Figura 4.6: Localizzazione tramite l'uso di due landmark

Capitolo 5

Risultati sperimentali

5.1 Introduzione

Il lavoro sperimentale è stato svolto nel laboratorio di robotica pesante dell'università di Roma Tor Vergata con il robot Nomad 150. Dapprima sono stati implementati algoritmi di *obstacle avoidance* e di navigazione casuale in ambienti ignoti, al fine di poter permettere al robot di muoversi in un ambiente chiuso evitando di collidere sia con ostacoli posti lungo il cammino sia con le pareti dell'ambiente.

All'algoritmo di aggiramento ostacolo, trattato nel primo capitolo, sono state aggiunte funzioni che permettono di vagare nel mondo circostante muovendosi verso la direzione più libera. A questo punto il sistema robot-computer è stato ampliato tramite l'utilizzo della webcam, in modo tale che, in contemporanea al movimento, il robot può effettuare un continuo controllo sulle immagini alla ricerca dei landmark al fine di localizzarsi.

Il riconoscimento del landmark è stato effettuato tramite l'uso della trasfor-

mata di Hough, descritta nel capitolo 2.

5.2 Implementazione trasformata di Hough

Per l'implementazione in linguaggio C della trasformata di Hough sono state usate due funzioni: la prima (*trasformata_hough*) effettua la ricerca delle rette nell'immagine e riempie le celle accumulatrici, la seconda (*ricerca_rette*) elabora le informazioni appena ricavate per calcolare la lunghezza dei lati del landmark.

La funzione *trasformata_hough* inizializza un array di dimensione 180×400, dove 180 rappresenta la massima ampiezza di un angolo θ nella notazione polare di una retta, mentre 400 la lunghezza massima del modulo ρ ottenuta da:

$$\rho_{max} = \sqrt{width^2 + height^2} \tag{5.1}$$

dove si è indicato con height e width la lunghezza e l'altezza dell'immagine, cioè 320×240. In seguito esamina l'immagine pixel a pixel, quando il valore di uno di essi è uguale a 255 (bianco), calcola i valori del modulo ρ per ogni θ compreso nell'intervallo [0;180] aumentando di uno il valore della cella accumulatrice associata alla coppia (ρ, θ).

```
void trasformata_hough(){
int i,j,r,Om,max,pix;
double Th,sinTh,cosTh ;
int rho,the,count=0;
for (i=0;i<180;i++)
```

```
for(j=0;j<396;j++)
Tr2_Acc_Ary[i][j]=0;
for(i=0;i<grab_height;i++){
for(j=0;j<grab_width;j++){
if (BW[j][i]==255) {
for(Om=0;Om<180;Om++ ) {
Th=CONV*Om;
sinTh=sin(Th);
cosTh=cos(Th);
r=(int)(j* sinTh + i*cosTh);
Tr2_Acc_Ary[Om][r] += 1 ; } }
}
}
```

Al termine dell'operazione viene chiamata la funzione *ricerca_rette* che analizza, una per una, tutte le celle accumulatrici. Quando una di esse contiene un valore maggiore di una determinata soglia scelta a priori (nel caso in questione 40), memorizza i corrispondenti valori di ρ e θ. A questo punto vengono ricavate le rette verticali ed orizzontali, cioè quelle il cui valore di θ risulta essere 0 o 180 per le verticali e 90 o 270 per le orizzontali.
Come si può notare dalla figura 5.1, in base al valore di soglia scelto, nell'analisi delle celle, si ottengono più o meno rette nell'immagine.

L'immagine acquisita e filtrata presenta molti errori e disturbi, quindi individuare rette perfettamente verticali o orizzontali presenta molti problemi. L'algoritmo di ricerca è quindi stato migliorato per approssimare a vertica-

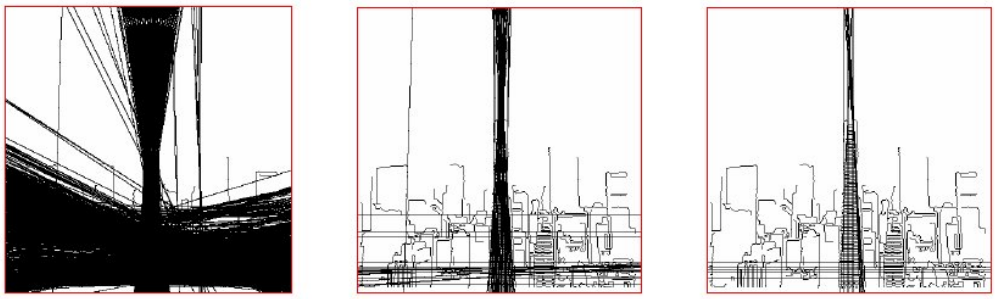

Figura 5.1: Rette in un'immagine reale per diversi valori di soglia

li le rette che hanno pendenze molto elevate, ed a orizzontali le rette con pendenza minima. A partire quindi dalla retta scritta in forma polare

$$x \cdot cos\theta + y \cdot sin\theta = \rho \tag{5.2}$$

possiamo ricavare il coefficente angolare e il termine noto dalle seguenti formule:

$$m = -\frac{1}{tan\theta} \tag{5.3}$$

$$q = \frac{\rho}{sin\theta} \tag{5.4}$$

Esaminando i vari valori assunti dal coefficiente angolare, quando uno di essi supera il valore massimo, imposto a 10, la retta viene considerata verticale; al contrario, quando scende oltre il valore minimo di 0.1, la retta viene con-

siderata orizzontale.

A questo punto, quindi, l'algoritmo ha memorizzato in due array X e Y i coefficienti delle rette rispettivamente verticali ed orizzontali.

Sempre a causa dei disturbi nell'immagine, dopo la trasformazione di Hough, vengono individuate molte rette parallele per ogni lato del landmark. Questa molteplicità potrebbe portare a errori di valutazione specialmente quando le rette corrispondenti ad un lato sono di numero maggiore di 10 dal momento che descrivono un'area di molti pixel.

Sotto le ipotesi che il landmark, per semplicità scelto rettangolare, sia di un colore che crei un forte contrasto con l'esterno e che non abbia altri oggetti o forme nelle vicinanze, è possibile separare le rette contigue di ogni lato con un algoritmo di scelta. Non resta quindi che fare una media matematica tra tutte le rette trovate al fine di ottenere un'unica equazione per ogni lato.

Il landmark può essere riconosciuto in base alle lunghezze dei lati, al loro rapporto e dalle pendenze delle rette.

5.3 Il landmark utilizzato

Nella figura 5.2 è illustrato il landmark utilizzato per gli esperimenti, un rettangolo blu che risalta molto rispetto allo sfondo, cioè il bianco delle pareti del laboratorio. Inizialmente è stata effettuata la localizzazione con un solo landmark posizionato su un muro ad una altezza da terra pari a quella della webcam posizionata sul robot, in seguito sono stati posizionati due landmark in due diversi punti della stanza, come verrà spiegato in seguito, per cercare

di migliorare i risultati ottenuti con un solo marker.

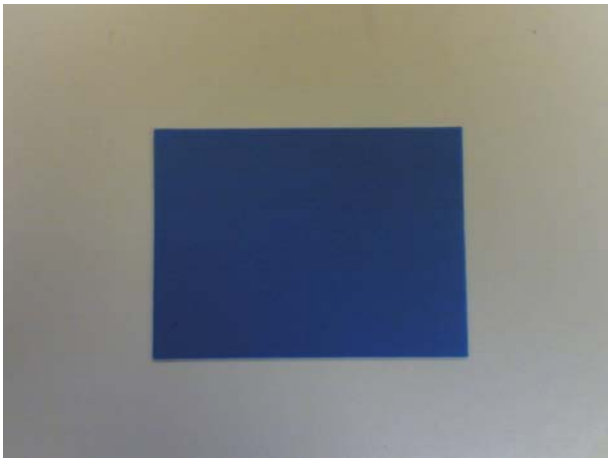

Figura 5.2: Landmark utilizzato per gli esperimenti

5.4 Localizzazione

Effettuato il collegamento tramite porta seriale tra il portatile ed il robot viene lanciato il programma di auto-localizzazione.
La prima operazione da attuare è quella di inizializzazione del robot, effettuata tramite i comandi della libreria "Nclient.h" seguendo i seguenti passaggi:

- Connessione al robot
- Impostazione periodo di timeout
- Azzeramento
- Assegnazione parametri base

La connessione viene effettuata tramite la chiamata alla funzione *connect_robot(robot_id)*, che inizia la comunicazione tra il robot ed il programma Nserver. Tramite la funzione *conf_tm(timeout)* impostiamo un intervallo di tempo oltre il quale, nel caso in cui il robot non riceva comandi, viene interrotto il movimento al fine di evitare che il robot rimanga incontrollato per troppo tempo.
In seguito avviene la chiamata alla funzione *zr()* che pone il robot in una posizione di "zero", cioè allinea il sonar zero e le ruote con la direzione del bumper 0. L'ultimo comando necessario per l'inizializzazione consiste nell'assegnare i parametri per la velocità e l'accelerazione massime.
Si rimanda all'appendice B per ulteriori informazioni sulle funzioni appena descritte.
Una volta inizializzato il Nomad, vengono chiamate tutte le funzioni di apertura della webcam, vengono impostati i seguenti valori: altezza e larghezza della visuale, framerate, luminosità, colore, tonalità, contrasto, bianco e profondità. In seguito viene attivata la funzione di snapshot per la cattura delle immagini e viene impostata la grandezza del buffer di memoria.

Lo scopo della localizzazione di un robot mobile è quella di ricavare, rispetto ai landmark, la sua posizione all'interno di una mappa immagazzinata in memoria. Nel programma, quindi, vengono fissate le coordinate dei due marker rispetto ad un'origine prestabilita all'interno della mappa.
Effettuati questi passaggi, il robot può iniziare la navigazione casuale nell'ambiente, per lui ignoto, verso la direzione che i 16 sonar indicano come la più "libera". Contemporaneamente al movimento vengono effettuati due controlli, il primo, tramite i sonar, per evitare collisioni con ostacoli presenti

sul cammino, il secondo, tramite la webcam alla ricerca del landmark, analizzando, istante per istante, le immagini catturate.
Su ogni fotogramma, quindi, la trasformata di Hough agisce alla ricerca del landmark, nel momento in cui questa ricerca dà esito positivo, il robot termina il suo movimento in quanto il landmark è stato individuato.
Il passo successivo consiste nella ricerca della distanza tra il robot ed il landmark che può essere effettuato in vari modi. Un primo approccio è quello descritto nel paragrafo 4.3.2, che comporta numerosi calcoli fornendo però una stima molto precisa.
Un secondo approccio, utilizzato nelle prove sperimentali, consiste nel ricavare la distanza letta dal sonar posto nella direzione in cui è orientata la webcam.
Un ultimo metodo è quello di ricavare la distanza webcam-landmark attraverso un'analisi del rapporto tra la lunghezza dei lati del landmark reale e di quello presente nell'immagine.

Tenendo traccia del movimento effettuato, possiamo facilmente conoscere, passo dopo passo, l'orientamento che la webcam ha rispetto alla posizione di partenza.
Immagazzinata in memoria la distanza tra il robot ed il landmark, l'orientamento della webcam e, note le coordinate della mappa, è facilmente ricavabile la posizione stimata del robot nella mappa attraverso le seguenti formule:

$$X_{robot} = X_{landmark} \pm D \cdot sin(\phi) \tag{5.5}$$

$$Y_{robot} = Y_{landmark} \pm D \cdot cos(\phi) \tag{5.6}$$

dove D è la distanza robot-landmark, mentre ϕ è l'orientamento della webcam, cioè della torretta del robot. I segni delle equazioni sono stabiliti in base al valore dell'orientamento.

5.5 Localizzazione mediante l'uso di due landmark

Nelle prove in laboratorio, per la localizzazione del robot, è stata effettuata anche la tecnica descritta nel paragrafo 4.4, ovvero l'uso di due landmark.
Sono quindi stati posizionati due marker, su due pareti adiacenti e non molto distanti tra loro, della stanza. Il robot si muove alla ricerca di uno di essi. Nel momento in cui la ricerca va a buon fine, viene calcolata la prima distanza robot-landmark, in seguito il robot ruota la torretta, e quindi la webcam, finché non viene individuato il secondo landmark e di conseguenza misurata la seconda distanza.
Calcolando le intersezioni tra le due circonferenze di centro il landmark e di raggio la rispettiva distanza, in base ai dati forniti dai sonar e alle coordinate della mappa, note al robot, viene individuata con buoni risultati la posizione del robot.

5.6 Risultati

Nelle immagini seguenti vengono illustrati i risultati degli esperimenti svolti in laboratorio. Come possiamo notare dalla figura 5.3, nella quale è mostrata

la mappa dell'ambiente utilizzato rispetto all'origine stabilita, le posizioni stimate dal programma sono molto vicine a quelle reali.

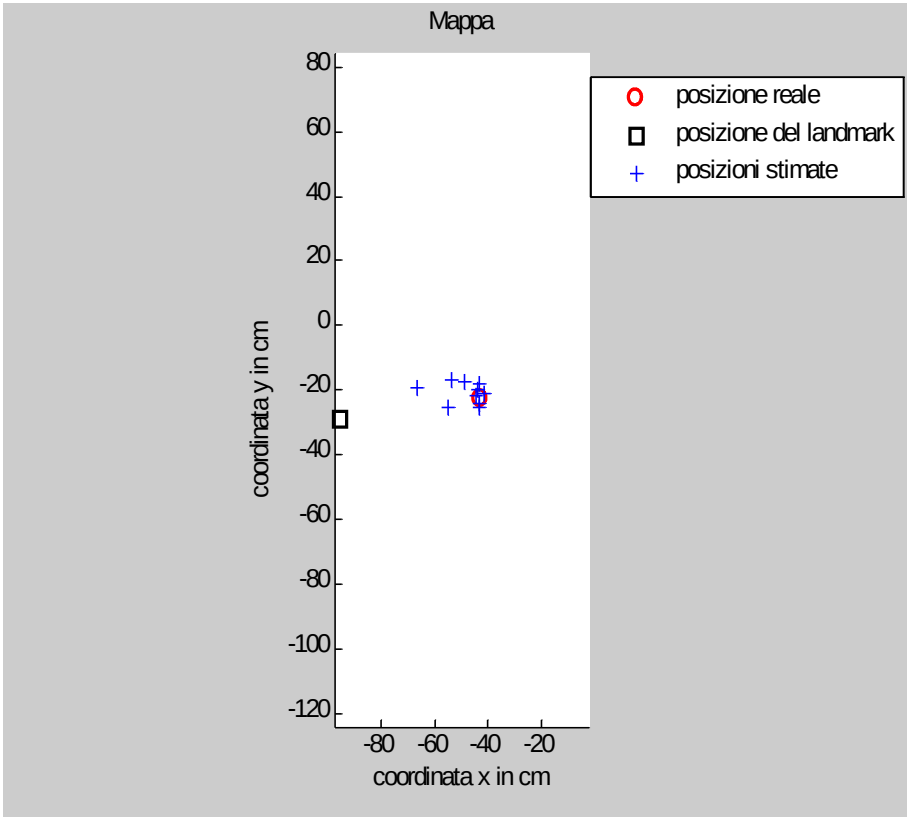

Figura 5.3: Posizioni del robot stimate dal programma

Dalle figure 5.4 e 5.5 possiamo notare che l'errore in percentuale sulle coordinate X e Y del robot è quasi sempre inferiore al 10%, con migliori risultati per la coordinata X.

In generale l'errore percentuale complessivo sul calcolo della posizione, risulta essere inferiore al 10% per otto delle dieci simulazioni effettuate, come si può notare dalla figura sottostante.

Negli esperimenti effettuati con due landmark, non si sono verificate dif-

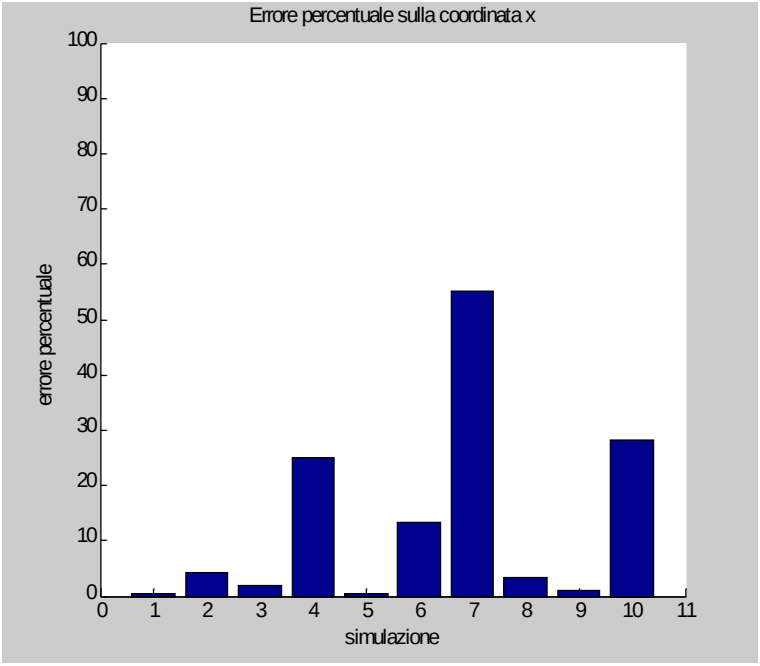

Figura 5.4: Errore percentuale sulla coordinata X

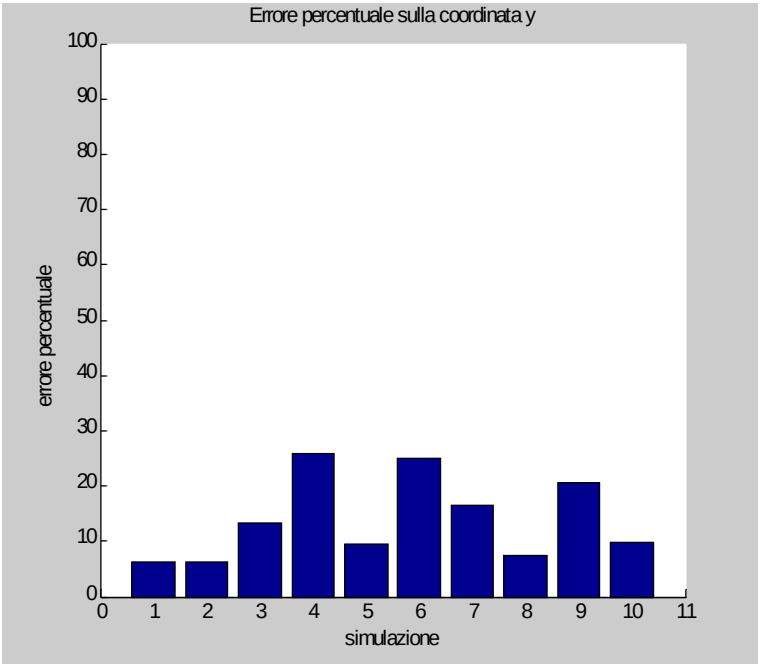

Figura 5.5: Errore percentuale sulla coordinata Y

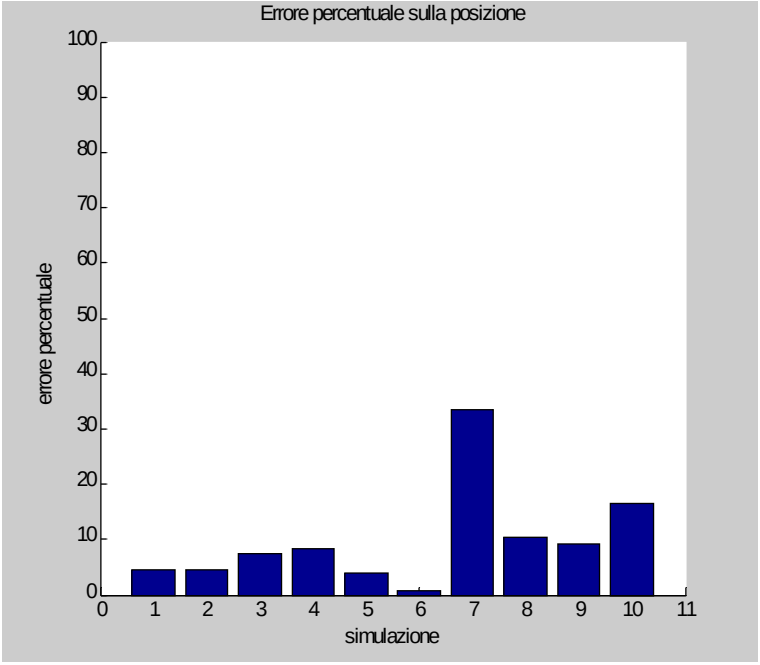

Figura 5.6: Errore percentuale sulla posizione

ferenze sostanziali nei risultati ottenuti.

Analizzando la figura 5.7 è chiaramente visibile che, anche in questo caso, la posizione stimata del robot si avvicina abbastanza a quella reale. Mentre in figura 5.8 possiamo vedere che l'errore è inferiore al 5% per cinque simulazioni su dieci, anche se per tre esperimenti si sono riscontrati evidenti errori di misura, specialmente del sonar. Gli errori riscontrati possono essere causati da numerose cause, come ad esempio una distanza mal rilevata da un sonar, o da una elaborazione poco precisa dell'immagine.

In conclusione, gli esperimenti hanno fornito i risultati attesi, il programma sviluppato costituisce, quindi, un soddisfacente mezzo per l'autolocalizzazione robotica mediante l'uso della visione artificiale.

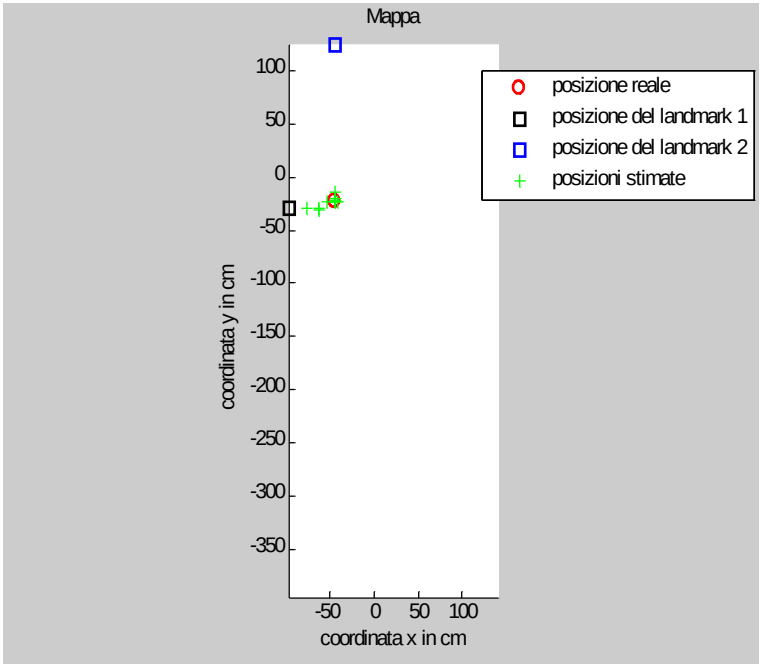

Figura 5.7: Posizioni del robot stimate dal programma con l'utilizzo di due landmark

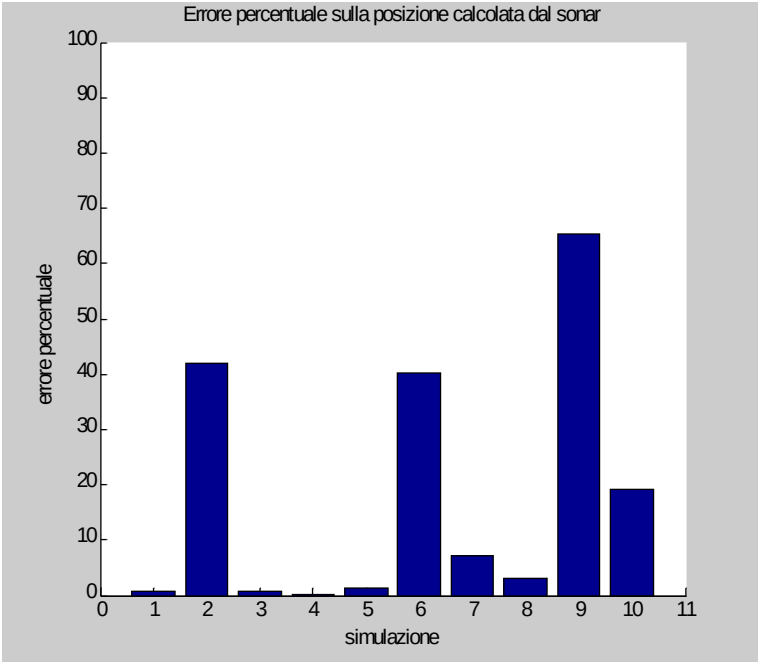

Figura 5.8: Errore percentuale sulla posizione con l'utilizzo di due landmark

Capitolo 6

Conclusioni

6.1 Problemi riscontrati e sviluppi futuri

I maggiori problemi che si sono presentati nelle prove sperimentali sono stati causati dalla scarsa risoluzione della webcam in uso. La scelta della webcam è stata motivata dalla semplicità di installazione e riconoscimento da parte del sistema operativo Linux Mandrake dal momento che sono facilmente reperibili su internet le librerie grafiche PWC per le webcam della Philips.
Essendo infatti il campo visivo della webcam alquanto limitato, sperimentalmente è accaduto che, a causa dei ritardi di elaborazione dell'immagine, il Landmark non fosse riconosciuto, a causa del fatto che era uscito dal campo visivo della webcam. Un utile miglioramento dell'argomento in esame potrebbe essere quindi l'utilizzo di una videocamera digitale con alta risoluzione che permetterebbe anche di riconoscere landmark dalle forme più articolate, o landmark naturali.
Un altro utile miglioramento potrebbe essere quello di utilizzare la *visione*

stereoscopica, ovvero sfruttare le immagini da due videocamere digitali per riconoscere oggetti fondendo le due imagini percepite sotto angolazioni diverse, facendo percepire l'oggetto come unico e solido.

Inoltre si potrebbe dotare la telecamera di una struttura *pan&tilt* in modo da garantire due gradi di mobilità alla vista del robot. In questo modo il landmark potrebbe essere cercato muovendo semplicemente la testa piuttosto che la torretta del robot.

I landmark inoltre potrebbero contenere al loro interno informazioni aggiuntive sulla destinazione del robot, come ad esempio la distanza dal luogo di arrivo o la direzione. In questo modo ogni landmark, all'interno dell'ambiente in cui il robot deve muoversi e localizzarsi, potrebbe essere differente dagli altri.

Le stime delle posizioni ricavate dai marker inoltre potrebbero essere migliorate sfruttando filtri di Markov o il filtro di Kalman al fine di eliminare gli errori di posizione.

Appendice A

Compilazione programmi

I programmi, scritti in linguaggio C, che sfruttano le librerie pwc della webcam philips e le libreie grafiche devono essere compilati con il seguente comando:

```
gcc -o prova esempio.c -I/usr/X11R6/include -L/usr/lib -lImlib2

-lttf -lm -ldl -lXext -lXext -lX11 -L/usr/X11R6/lib Nclient.o
```

dove "esempio.c" è il file da compilare, mentre "prova" è il nome dell'eseguibile.

I programmi per il movimento del robot, senza l'ausilio della webcam, possono essere compilati in due diverse modalità:

```
gcc -o prova esempio.c Nclient.o
```

nel caso in cui il collegamento voglia essere effettuato tramite il programma Nserver, avendo quindi anche la possibilità di simularlo oltre che di eseguirlo sul robot reale.

Invece scrivendo:

```
gcc -o prova esempio.c Ndirect.o
```

nel caso in cui il programma deve essere eseguito direttamente sul robot, senza avere in esecuzione Nserver.

Appendice B

Funzioni della libreria Nclient.h

Queste sono le principali funzioni della libreria Nclient.h:

- *connect_robot(long robot_id)* effettua la richiesta di connessione tra il server e il robot. Per ottenere la comunicazione prima di effettuare la chiamata bisogna impostare due parametri fondamentali: SERV_TCP_PORT=7019; SERVER_MACHINE_NAME="localhost" (usando la funzione per copiare le stringhe "strcpy") Robot_id è il numero di identificazione del robot.

- *disconnect_robot(long robot_id)* effettua la richiesta al server di interrompere la connessione con il robot.

- *conf_tm(int timeout)* imposta il periodo di timeout in secondi del robot. Se il robot non riceve comandi dall'host per un intervallo di tempo superiore al timeout, interrompe il movimento corrente.

- *real_robot(void)* permette di passare dalla modalità simulazione a quella reale sul robot.

- *simulated_robot(void)* effettua il passaggio dalla modalità real_robot alla modalità simulata.

- *sp(unsigned int tsp,unsigned int ssp,unsigned int rsp)* imposta le velocità di traslazione, rotazione e di rotazione della torretta rispettivamente. I parametri inseriti sono espressi in 1/10 di pollici/s per la traslazione o in 1/10 di grado/s per le rotazioni. I valori massimi sono sp(200,450,450).

- *ac(unsigned int t_ac,unsigned int s_ac,unsigned int r_ac)* imposta le accelerazioni di traslazione e delle due rotazioni rispetivamente. I valori massimi sono ac(300,500,500). I parametri inseriti sono espressi in 1/10 di pollici/s^2 per la traslazione o in 1/10 di grado/s^2 per le rotazioni.

- *pr(int trp,int spr,int rpr)* muove i motori del robot rispetto alle distanze fornite nei parametri usando le velocità definite in sp(). Gli argomenti sono espressi in 1/10 di pollice o 1/10 di grado e possono essere inseriti valori compresi nell'intervallo [-32000,32000].

- *vm(int tv,int sv,int rv)* muove il robot in base alle velocità espresse nei parametri.

- *mv(int t_mode,int t_mv,int s_mode,int s_mv,int r_mode,int r_mv)* muove i tre assi del robot indipendentemente. Negli argomenti ci sono due parametri per ognuno dei tre motori, una legge di controllo e un valore per il movimento. Le leggi di controllo sono contenute nel file Nclient.h e sono le seguenti:

  - MV_VM : imposta la modalità velocità

  - MV_PR : imposta la modalità posizione
  - MV_IGNORE : ignora le informazioni per l'asse specifico
  - MV_SP : imposta la velocità per l'asse
  - MV_AC : imposta l'accelerazione dell'asse

- *st(void)* ferma il movimento del robot.
- *ws(char wt,char ws,char wr,char timeout)* aspetta prima di interrompere il movimento del robot. Attende per un tempo pari al timeout i motori che vengono specificati negli argomenti.
- *zr(void)* allinea le ruote e la torretta con il bumper zero.
- *gs(void)* aggiorna lo stato del robot (sensori,encoder...) nel vettore di stato.
- *add_obstacle(long obs[21])* aggiunge nella mappa un ostacolo. Il primo valore dell'array corrisponde al numero di vertici dell'ostacolo (max 10) e i seguenti venti valori corrispondono alle coordinate x,y dei vertici.
- *delete_obstaclee(long obs[21])* elimina un ostacolo precedentemente inserito.
- *new_world(void)* elimina tutti gli ostacoli della mappa.

Nella libreria Nclient.h è presente un vettore globale, lo State Vector, che contiene i dati sullo stato corrente del robot, la sua configurazione e i dati provenienti dalla lettura dei sensori. I componenti principali di questo vettore sono:

- STATE_SIM_SPEED velocità della simulazione
- STATE_SONAR_0-15 contiene la distanza in pollici restituita dai sonar
- STATE_BUMPER lettura dei 20 sensori tattili
- STATE_CONF_X-Y coordinate x (oppure y) in 1/10 di pollice del robot rispetto alla posizione iniziale
- STATE_CONF_STEER orientamento ruote in 1/10 grado rispetto alla posizione iniziale
- STATE_CONF_TURRET orientamento della torretta in 1/10 grado rispetto alla posizione iniziale
- STATE_VEL_TRANS velocità traslatoria in 1/10 pollici/s
- STATE_VEL_STEER velocità rotazione in 1/10 pollici/s
- STATE_VEL_TURRET velocità rotazione torretta in 1/10 pollici/s

Bibliografia

[1] King-Sun Fu, R.C. Gonzalez, C.S.George Lee, *Robotica*, Mac-Graw Hill, 1989

[2] Jiann-Der Lee, Indoor Robot Navigation by Landmark Tracking, *Mathl. Comput. Modelling* 26 (4), pag. 79-89, 1997

[3] G.Adorni, G.Destri, M.Mordonini, F.Zanichelli, Robot self-localization by means of vision, *IEEE Proceedings of EUROBOT '96*, 1996

[4] G.Adorni, M.Gori, M.Mordonini, F.Zanichelli, Just-in-time landmarks recognition , *Academic Press*, 1999

[5] D.Scaramuzza, Progetto e realizzazione di un sistema di visione stereoscopica per la robotica, con applicazione all'inseguimento di corpi in moto e all'autolocalizzazione, Tesi di laurea in Ingegneria Elettronica vecchio ordinamento, *Università degli studi di Perugia*, 2004

[6] F.Romanelli, Sviluppo di sistemi integrati di visione per il controllo di robot mobili tramite retroazione basata su telecamere, Tesi di laurea in Ingegneria Informatica *Università degli studi di Roma Tor Vergata, 2003*

[7] Nomadic Technologies Inc., *Nomad 200 User's Manual*, 1997

[8] Nomadic Technologies Inc., *Language Reference Manual*, 1997

[9] Kyewook Lee, Application of the Hough transform, University of Massachusetts, 2006

[10] X.J.Li, A.T.P.So, S.K.Tso, CAD-Vision-Range-Based self-localization for mobile robot using one landmark, *Journal of intelligent and robotic system*, pag. 61-81, 2002

[11] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition*, pag. 111-122, 1981

[12] Nemosoft Unv, Video4Linux API additions for Philips USB Webcams, 2001

[13] C.Balkenius, L.Kopp, Elastic Template Matching as a Basis for Visual Landmark Recognition and Spatial Navigation, 1997

[14] http://webuser.unicas.it/tortorella/TTII/

Elenco delle figure