UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica
(Prof. Riccardo Cassinis)

MorgulDoc3
Software di entrata nella stazione di ricarica

Elaborato di esame di: **Francesca Periti, Enrico Villani**

Consegnato il: **3 agosto 2004**

Sommario

Il progetto affrontato prevede lo sviluppo di un sistema software per guidare il robot "Morgul" alla propria stazione di ricarica.

In particolare, al momento del transito in prossimità della stazione di ricarica, il software deve governare il robot in modo da rendere possibile l'entrata del robot nella stazione.

L'avvicinamento alla stazione stessa deve avvenire in modo che il robot possa completare l'operazione con successo, mantenendo quindi sempre in vista i marcatori che identificano l'obiettivo ed evitando di urtare le pareti della stazione di ricarica.

Al termine della procedura, il robot, Saphira ed il software di acquisizione video si spengono.

1. Introduzione

Il compito affrontato si inserisce all'interno del progetto, sviluppato presso il Laboratorio di Robotica Avanzata della Facoltà di Ingegneria dell'Università di Brescia, che ha come obiettivo la possibilità di dotare la Facoltà stessa di un robot con funzioni di sorveglianza.

In particolare, il robot predisposto a tale compito deve potersi muovere liberamente nel giardino della Facoltà e monitorare, tramite una telecamera ed i propri sensori, eventuali attività irregolari.

A tale scopo, il robot deve essere in grado di tornare nel Laboratorio e di interfacciarsi con la propria stazione di ricarica nel momento in cui viene meno la sua autonomia dal punto di vista dell'alimentazione: il progetto ha come obiettivo quest'ultima attività.

Dal punto di vista della realizzazione, il progetto è stato suddiviso in quattro fasi distinte:

1. progettazione e costruzione della stazione di ricarica e delle componenti fisiche necessarie al robot per l'interfaccia;
2. implementazione di un sistema di visione per il robot basato su marcatori luminosi;
3. implementazione di un sistema software per guidare il robot alla stazione di ricarica;
4. implementazione di una mappa elettronica del giardino della Facoltà.

Il compito da noi affrontato consiste nell'esecuzione della terza fase.

2. Il problema affrontato

L'obiettivo del progetto in esame consiste nell'implementazione di un sistema di guida software che permetta al robot "Morgul", poste determinate condizioni descritte in seguito, di avvicinarsi alla stazione di ricarica, di entrarvi e di interfacciarsi fisicamente con il dispositivo per eseguire la carica.

Al termine dell'operazione, il robot ed i software di gestione si spengono.

2.1. Ipotesi iniziali

Per l'implementazione del software, sono state formulate delle ipotesi a livello operativo in linea con l'obiettivo da raggiungere: non è garantito il funzionamento del sistema nel caso in cui la situazione reale non rispetti quanto modellizzato.

2.1.1. La stazione

Si suppone che la stazione sia pronta a ricevere il robot nel momento della ricarica. In particolare:

- il pannello di accesso alla stazione deve essere abbassato, per consentire al robot l'ingresso nella stazione stessa;
- i marcatori che permettono al robot di individuare la stazione di ricarica devono essere in funzione ed avere una collocazione corretta: in questo caso, due lampadine poste in verticale l'una sopra l'altra in posizione rialzata rispetto alla postazione di ricarica; devono inoltre essere visibili da parte della telecamera posta sul robot;
- la telecamera deve essere il più possibile in asse con il corpo del robot;
- in prossimità della stazione di ricarica non vi devono essere ostacoli che la nascondano alla telecamera o che ne rendano impossibile l'accesso da parte del robot;
- le pinze a cui il robot si connette fisicamente devono essere dotate di corrente per permettere al robot di eseguire la ricarica.

2.1.2. Il robot

Si suppone che il robot:

- si trovi già in prossimità della stazione di ricarica (non sono state considerate le modalità con cui l'avvicinamento alla stazione viene effettuato);
- nel momento iniziale della simulazione possa avere qualunque orientamento spaziale, a patto che i marcatori siano nel campo visivo della telecamera.

2.1.3. Il software ausiliario

L'aspetto fondamentale per risolvere il problema in esame è la possibilità, da parte dei comportamenti che gestiscono il robot, di conoscere in ogni istante la posizione della stazione di ricarica, grazie all'utilizzo di marcatori (i marcatori sono oggetti fissi, facilmente individuabili e riconoscibili senza incertezza da parte del robot). Al robot vengono quindi fornite dal software ausiliario "*trovaBlob*" (sviluppato nella fase 2 del progetto) le coordinate x ed y dei due marcatori utilizzati.

3. La soluzione adottata

L'obiettivo del progetto è stato raggiunto in due fasi distinte:

- uno studio iniziale sulle modalità di avvicinamento per avere un orientamento il più possibile favorevole per le operazioni di entrata nella stazione di ricarica;
- lo sviluppo del software.

3.1. L'algoritmo di avvicinamento

Date le ipotesi iniziali descritte nel capitolo 2.1 (il robot si trova in prossimità della stazione con un orientamento casuale ma con i marcatori nel campo visivo della telecamera), l'avvicinamento alla stazione avviene nel seguente modo:

- il robot determina la direzione di provenienza (da destra o da sinistra) rispetto all'asse della stazione di ricarica (figura 1);

Figura 1 – Ipotesi iniziale: il robot si trova in una posizione arbitraria con i marcatori nel campo visivo della telecamera. Il comportamento determina la direzione di provenienza del robot (in questo caso, da destra).

- il robot ruota in senso antiorario od orario, a seconda della posizione dei marcatori, fino a che l'ascissa di uno dei due marcatori raggiunge il bordo destro o sinistro dell'immagine della telecamera (con una certa soglia) su cui vengono eseguite le estrapolazioni per ottenere le coordinate dei marcatori (le ascisse delle coordinate dei marcatori sono sfasate l'una rispetto all'altra in modo proporzionale alla direzione di provenienza del robot). Questo comportamento garantisce che il robot mantenga sempre i marcatori nel campo visivo della telecamera e che possa raggiungere l'asse della stazione di ricarica seguendo il percorso più breve (figura 2);

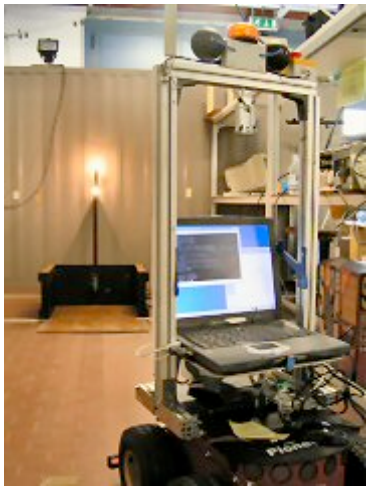

Figura 2 – Il robot ruota verso sinistra per allineare l'ascissa del marcatore più alto al bordo destro dell'immagine.

- il robot avanza in linea retta fino a che le ascisse dei marcatori sono sovrapposte: il robot si trova sull'asse della stazione di ricarica, ma con un orientamento errato (figura 3);

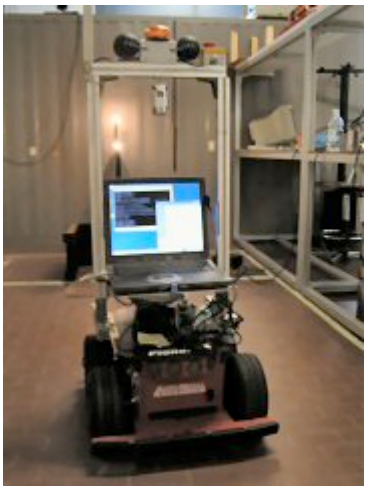

Figura 3 – il robot si trova sull'asse della stazione e deve girarsi verso destra per allinearsi correttamente con l'asse stesso.

- il robot ruota finché le ascisse dei marcatori sono al centro dell'immagine ottenuta dalla telecamera: il robot si trova quindi sull'asse della stazione di ricarica e ha l'orientamento corretto per effettuare le operazioni di inserimento nella stazione stessa (figura 4);

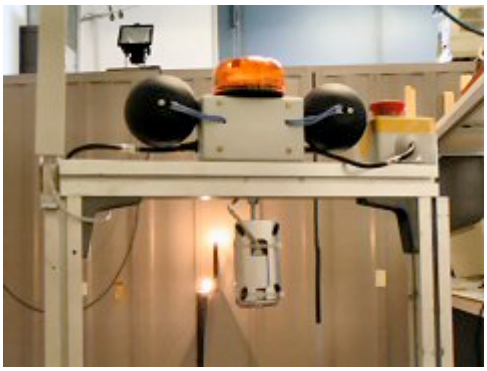

Figura 4 – il robot è orientato parallelamente all'asse della stazione.

- il robot avanza in linea retta effettuando, se necessario, delle correzioni sul proprio orientamento, cioè mantenendo le ascisse dei marcatori al centro dell'immagine (figura 5);

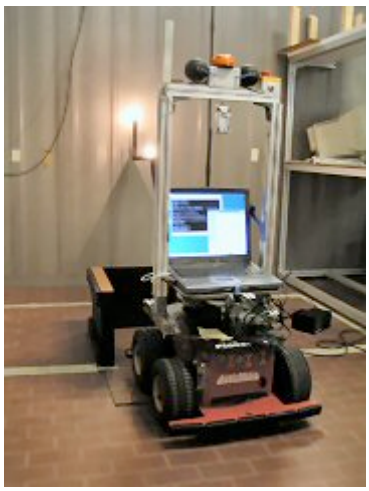

Figura 5 – il robot prosegue in linea retta, mantenendo l'allineamento con l'asse della stazione.

- il robot entra nella stazione ed avanza fino a che oltrepassa una fotocellula che determina lo spegnimento dei marcatori (hardware sviluppato nella fase 1): nel momento in cui i marcatori scompaiono dall'immagine ricevuta dalla telecamera, il robot ed i software di gestione si spengono (figura 6).

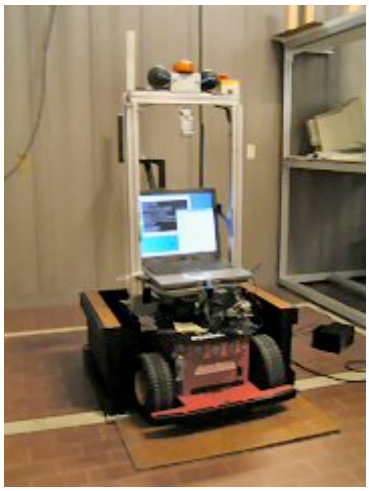

Figura 6 – il robot entra nella stazione e si connette fisicamente alla pinze per la ricarica.

Il robot compie le rotazioni richieste senza fermare i motori, mantenendo la velocità costante. Uniche eccezioni sono:

- quando il robot rallenta se è almeno per metà dentro la stazione;
- quando è appena partito, per allinearsi più velocemente con l'asse della stazione.

3.2. Il software

Il sistema di guida software è stato sviluppato nell'ambiente *Saphira*, un'applicazione per il controllo di robot basato su un'architettura client/server, che a sua volta utilizza *Aria*, un ambiente analogo che opera a più basso livello.

Il software sviluppato si basa sulle librerie messe a disposizione da questi due ambienti per quanto riguarda l'interazione diretta con il robot, vale a dire per istruzioni relative al movimento (come la regolazione della velocità e dell'angolo di rotazione).

L'implementazione dell'algoritmo è stata invece realizzata ricorrendo ai *behaviors* (comportamenti). Un behavior è un insieme di istruzioni che hanno come scopo l'esecuzione di un singolo obiettivo in un contesto ristretto, tenendo conto delle informazioni sullo stato interno del robot.

Prima di scrivere i comportamenti è stato necessario scomporre l'algoritmo di avvicinamento in sotto-problemi più semplici, in base all'obiettivo di ciascuno ed alla strategia di risoluzione:

- "raggiungere l'asse della stazione di ricarica";
- "mantenersi in asse";
- "andare avanti diritto".

I punti precedenti si sono quindi tradotti nei comportamenti descritti successivamente nei capitoli 3.2.1, 3.2.2, 3.2.3. I comportamenti sono stati scritti in linguaggio *C++* e compilati in uno *shared object* (percorso */usr/local/Saphira/lib/dock.so*) in modo da essere accessibili dall'ambiente Saphira. Per maggiori chiarimenti, si rimanda alla documentazione di commento del codice.

3.2.1. SfAxAction

SfAxAction:

- individuare la direzione di provenienza del robot.

  Per identificare la direzione di provenienza, il software confronta la distanza tra le ascisse dei marcatori (il parametro *distanza*) con un valore di riferimento *DIST* (è il parametro che determina l'attivazione del comportamento: fornisce un valore di soglia per stabilire se il robot si trova o meno in asse con la stazione di ricarica):

  - se *distanza* è maggiore di *DIST* significa che il robot sta arrivando da destra (figura 7);
  - se *distanza* è minore di *-DIST* significa che il robot sta arrivando da sinistra (figura 8).

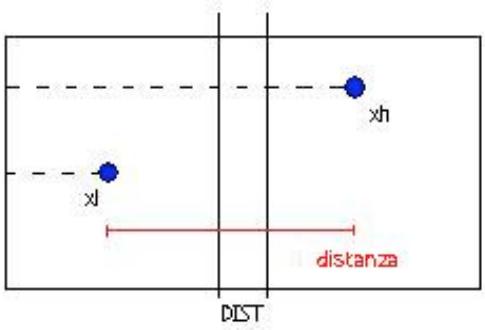

Figura 7 – Avvicinamento da destra: *distanza* è una valore positivo (in quanto definito come *xh-xl)* e viene confrontato con *DIST*: se *distanza* è minore di *DIST*, significa che il robot si trova in asse con la stazione, altrimenti è fuori asse.

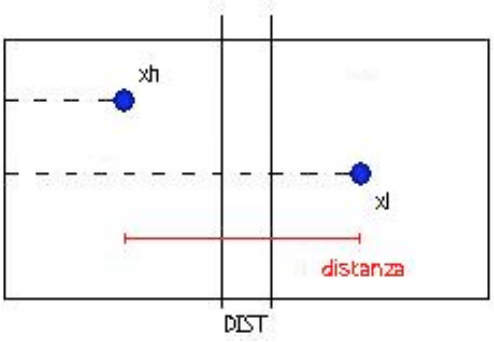

Figura 8 – Avvicinamento da sinistra: *distanza* è una valore negativo (in quanto definito come *xh-xl)*, per cui viene confrontato con *–DIST*.

Per effettuare i calcoli necessari, il software riceve le coordinate dei marcatori tramite una *FIFO*, che riceve i dati dal software ausiliario "*trovaBlob*", di cui al capitolo 2.1.3. Per accedere ai parametri, è necessario effettuare un parsing, in quanto la *FIFO* li trasmette non come singoli interi ma in formato stringa.

- ruotare il robot fino a che l'ascissa del punto medio dei due marcatori è vicina al bordo dell'immagine.

  La rotazione del robot viene eseguita calcolando la distanza tra un valore predefinito (il parametro *offset*) e l'ascissa del punto medio dei marcatori, ottenendo il parametro *lato*. Si è scelto di non utilizzare come margine il bordo dell'immagine per garantire una maggiore accuratezza nelle operazioni di avvicinamento alla stazione, tenendo conto di eventuali imprecisioni nella raccolta dei dati dovute ad oscillazioni della telecamera. In questo modo infatti, il sistema risulta robusto per quanto riguarda la perdita dei marcatori da parte del robot. La direzione della rotazione viene identificata dal parametro *segno*, ottenuto come rapporto tra il parametro *lato* ed il suo valore assoluto.

L'obiettivo è di ridurre entro un limite di soglia (il parametro *SOGLIA*) il valore di *lato*: se questo accade infatti, significa che i marcatori sono in una posizione tale per cui il robot, proseguendo in linea retta, interseca l'asse della stazione.

A seconda della direzione di provenienza, il parametro *lato* viene calcolato in modo differente:

- con provenienza da destra, *lato* è la distanza tra *xm* ed il bordo destro dell'immagine (in realtà dall'offset), come illustrato in figura 9

- con provenienza da sinistra, *lato* è la distanza tra *xm* ed il bordo sinistro dell'immagine (in realtà dall'offset), come illustrato in figura 10

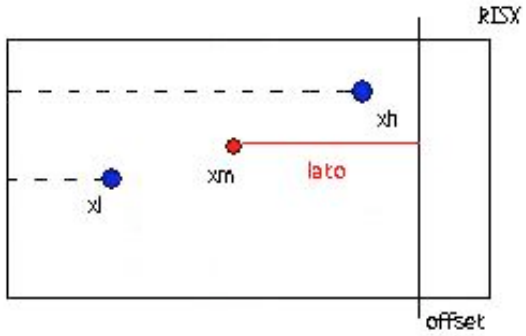

Figura 9 – Provenienza da destra: *lato* è calcolato come differenza tra i parametri *RISX* (la risoluzione dell'immagine in questo caso 640 pixel), *xm* (punto medio tra le ascisse dei marcatori) ed *OFFSET* (valore inizializzato in base a prove eseguite sul campo).

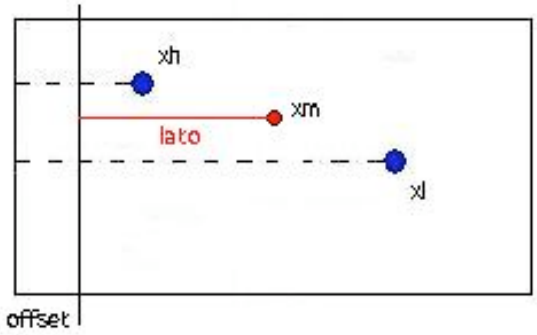

Figura 10 – Provenienza da sinistra: *lato* è calcolato come differenza tra i parametri *xm* (punto medio tra le ascisse dei marcatori) ed *OFFSET* (valore inizializzato in base a prove eseguite sul campo).

Il comportamento *SfAxAction* esegue inoltre anche altre operazioni:

- chiama la funzione *parsInt* per eseguire il parsing delle coordinate provenienti dalla *FIFO*.
- chiama la funzione *termina* quando il robot è entrato nella stazione di ricarica. Tale funzione lancia lo script *stop* dalla cartella */home/robot/DOCK*, che determina lo spegnimento del robot, di Saphira e del software di acquisizione delle immagini. Per identificare il momento in cui il robot si trova all'interno della stazione si controlla se le coordinate dei marcatori sono nulle (quando il robot si connette fisicamente alla stazione, i marcatori vengono disattivati) e se il parametro *distanzaYOld* - il valore di *distanzaY* al passo precedente - è minore di un valore di soglia predefinito (il parametro *DISTYDOCK*).
- chiama la funzione *tornaIndietro* quando il robot è in una posizione tale per cui non è possibile concludere con successo l'operazione di inserimento nella stazione.

Questo si verifica quando il parametro *distanza* è maggiore del parametro *CRITICALDIST* ed il parametro *distanzaY* è maggiore del parametro *CRITICALDISTY*, oppure quando il parametro *segnoOld* ha valore –100 (cioè il robot non ha mai visto i marcatori): se questo accade, il robot si ferma per non danneggiare la stazione.

- aggiorna in tempo reale i parametri *DIST*, *OFFSET*, *SPEED* e *CRITICALDIST*, che governano il comportamento, in base alla distanza del robot dalla stazione di ricarica (quindi in base al parametro *distanzaY*). L'aggiornamento viene eseguito per rendere più robusto il sistema: l'accuratezza delle coordinate infatti diminuisce più il robot si avvicina alla stazione ed è quindi necessario rendere più "tollerante" il comportamento.
- nel caso in cui il robot perda i marcatori (ad esempio, la visione della telecamera è stata bloccata temporaneamente da un ostacolo oppure il robot si è girato troppo), inverte il senso di rotazione del robot fino a che i marcatori non tornano nel campo visivo della telecamera.
- si disattiva quando il robot arriva in prossimità della stazione di ricarica, per evitare conflitti con il comportamento *SfCenterAction*. Si suppone infatti che quando il robot arriva ad una certa distanza dalla stazione (quando il parametro *distanzaY* è maggiore del parametro *DISTY*), sia in grado di mantenersi in asse e di proseguire diritto, entrando correttamente nella stazione.
- controlla se il comportamento è stato attivato per la prima volta (tramite il parametro *startup* con valore 1) oppure è già stato attivato in precedenza (*startup* ha valore 0). Nel primo caso infatti, il robot deve girarsi nella direzione corretta prima di muoversi, per evitare possibili errori di traiettoria; altrimenti si trova già lungo la direzione corretta e può proseguire. Questo controllo è particolarmente utile nei casi limite in cui il robot parte molto vicino alla stazione: se si avviasse senza essere già allineato nella giusta direzione, potrebbe non riuscire ad allinearsi in tempo, perdendo i marcatori.

3.2.2. SfCenterAction

SfCenterAction: ha come obiettivo:

- mantenere il robot sull'asse della stazione di carica, in particolare durante la fase finale dell'avvicinamento;

  Per mantenere il robot sull'asse della stazione, il software calcola il parametro *fromCenter*, vale a dire la distanza dal centro dell'immagine (in questo caso, 320 pixel) del punto medio delle ascisse dei due marcatori. Se *fromCenter* è minore di un certo valore di soglia (il parametro *SOGLIA*) oppure se il parametro *distanza* (la differenza tra le ascisse dei marcatori) è minore del parametro *DIST,* il robot si trova in asse (figura 11), altrimenti deve girarsi di conseguenza, nella direzione indicata dal parametro *segno*, calcolato come rapporto tra il parametro *fromCenter* ed il suo valore assoluto.

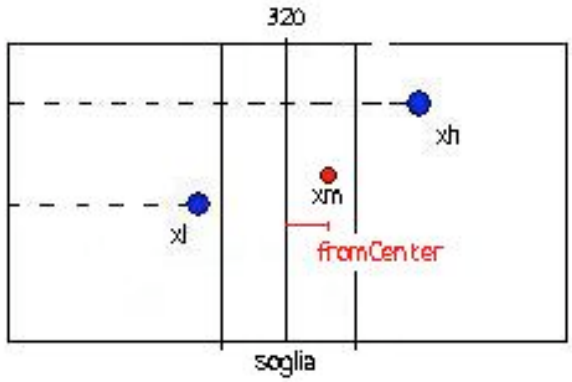

Figura 11 – *fromCenter* è minore di *SOGLIA*, per cui il robot si trova in asse con la stazione e viene attivato il comportamento *SfForwardAction*.

Il comportamento riceve le coordinate dei marcatori dal comportamento *SfAxAction* tramite una FIFO. Per accedere ai parametri, è necessario effettuare un parsing, in quanto la *FIFO* li trasmette non come singoli interi ma in formato stringa.

Il comportamento *SfAxAction* esegue inoltre anche altre operazioni:

- chiama la funzione *parsInt* per eseguire il parsing delle coordinate provenienti dalla *FIFO*.
- aggiorna in tempo reale i parametri *DIST*, *SOGLIA* e *SPEED*, analogamente a quanto detto per il comportamento *SfAxAction*.

3.2.3. SfForwardAction

SfForwardAction: ha come obiettivo:

- fare avanzare il robot in linea retta.

  Il comportamento pone a 0 la velocità di rotazione del robot e al valore *speedVal*, che viene passato al comportamento dalla action *init*.

3.2.4. Init.act

È una *action* scritta in linguaggio *Colbert* (percorso */home/robot/DOCK*) che viene caricata nell'ambiente Saphira. La sua funzione è quella di caricare lo *shared object dock.so* contenente i comportamenti e chiamare i comportamenti stessi, passandovi gli opportuni parametri e stabilendo la loro priorità. L'ordine di esecuzione è stato fissato in base all'importanza dell'obiettivo da eseguire ed è il seguente:

- SfAxAction: il robot deve innanzitutto portarsi sull'asse della stazione. Questo comportamento gestisce anche i casi limite e l'arrivo nella stazione, quindi deve necessariamente avere priorità massima.
- SfCenterAction: il robot deve quindi proseguire mantenendosi sull'asse senza deviazioni.
- SfFrowardAction: una volta stabilita correttamente la traiettoria da seguire, il robot prosegue in linea retta.

La *action* inoltre, si occupa di rimuovere eventuali copie preesistenti dei comportamenti che attiva.

3.2.5. Stop

È uno script situato in */home/robot/DOCK* che viene invocato quando il robot completa l'ingresso nella stazione di ricarica e vi si interfaccia fisicamente. Ha il compito si spegnere il robot, Saphira ed il software di acquisizione delle immagini.

4. Modalità operative

4.1. Componenti necessari

Per potere eseguire il software di guida sono necessari i seguenti componenti software:

- *Saphira*, per caricare il programma di guida (la versione utilizzata è per i test è la 8.3.2);
- *trovaBlob,* lo script per ottenere le coordinate dei marcatori (fase 2) e quanto contenuto nella cartella */home/robot/RV*;

- *init.act,* (percorso */home/robot/DOCK*)è il programma in linguaggio Colbert da caricare nell'ambiente Saphira;
- *dock.so*, (percorso */usr/local/Saphira/lib/dock.so*) è lo *shared object* contenente i comportamenti e viene caricato da *init.act*;
- *stop,* (percorso */home/robot/DOCK*) è lo script che ferma il robot ed i software di gestione.

A livello hardware è necessario che la telecamera posta sul robot sia accesa ed orientata correttamente.

4.2. Modalità di installazione

Non è necessario installare alcun componente per eseguire il software di guida (supponendo che *Saphira* e la cartella */home/robot/RV* siano già presenti).

4.3. Modifiche al Software

Per effettuare delle modifiche ai comportamenti - quindi al file *dock.cpp* - è necessario avere installati:

- il *makefile* (percorso */home/robot/DOCK*) per compilare lo *shared object* dal file C++.
- *Aria*, le librerie (la versione utilizzata per i test è stata la 1.3.2).

> ➢ **Invece di installare le librerie Aria come pacchetto aggiuntivo, è meglio utilizzare quelle presenti nell'ambiente Saphira nella directory /usr/local/Saphira/ohandler/include/Aria, altrimenti il file viene compilato correttamente, ma non funziona nell'ambiente Saphira (resituisce un errore nelle librerie Aria). Bisogna quindi modificare il *makefile* come indicato:**
>
> **ARIAD = $(SAPHIRA)/ohandler/include/Aria**
>
> **INCLUDE = -I$(INC) –I$(ARIAD)**

- *gcc-2.95.3,* per compilare il file C++.

> ➢ **È obbligatorio utilizzare la versione 2.95.3, perché altrimenti Saphira genera un errore e non carica i comportamenti.**

4.4. Esecuzione

Per lanciare il programma di guida, è necessario:

- avviare Saphira;
- lanciare lo script *trovaBlob* dalla cartella */home/robot/RV* (oppure lo script ausiliario *start*, che si trova nella medesima cartella e si occupa anche della ricompilazione e della rimozione di processi residui di esecuzioni precedenti);
- connettere il robot;
- lanciare dal terminal di Saphira la *action* Colbert *init.act*:
  - '*load init*' per caricare la *action*;
  - '*start dock*' per avviare i comportamenti.

> ➢ **IMPORTANTE: l'ordine di avvio dei software NON è casuale ma deve essere quello indicato, altrimenti si generano dei problemi con le FIFO, il programma di guida non funziona e Saphira si blocca.**

5. Conclusioni e sviluppi futuri

Il programma di guida implementato consente al robot di avvicinarsi alla stazione di ricarica, allinearsi ad essa ed entrarvi per effettuare la ricarica, date le condizioni di cui al capitolo 2.1.

5.1. Prove sul campo

Sono stati eseguiti numerosi test in Laboratorio, che hanno permesso di:

- verificare la correttezza dell'algoritmo implementato: il robot, poste le condizioni di cui al capitolo 2.1, entra con successo nella stazione di ricarica;
- individuare i valori ottimali per i parametri che gestiscono i valori di soglia, offset, distanza e velocità;
- correggere delle imperfezioni presenti nel software, individuabili esclusivamente nella fase di testing. In particolare, in seguito alle prove pratiche, le modifiche apportate al software sono state:
    - alzare la velocità lineare del robot fino ad un valore di 150 mm/sec: l'inerzia del robot infatti ne ostacola il movimento a velocità ridotte;
    - alzare la velocità di rotazione del robot, portandola ad un valore di 3 rad/sec per il comportamento SfAxAction e 4 rad/sec per il comportamento SfCenterAction, sempre per motivi legati all'inerzia;
    - disattivare il comportamento SfAxAction quando il robot si trova in prossimità della stazione, per evitare conflitti tra i diversi comportamenti;
    - evitare di bloccare il robot ad ogni cambio di direzione, sia per evitare eventuali conflitti tra i comportamenti, sia per impedire un'eccessiva oscillazione dell'impalcatura su cui è posta la telecamera (con una conseguente imprecisione sulla cattura delle coordinate dei marcatori).
    - individuare la distanza critica oltre la quale il robot non riesce più ad entrare nella stazione ed inserire dei controlli per fermare il robot quando questo accade;
    - dotare il software di un controllo nel caso in cui i marcatori escano temporaneamente dal campo visivo della telecamera, per fare ruotare il robot in senso inverso;
    - utilizzare la media dei valori delle ascisse dei marcatori per ottenere una maggiore correttezza invece che un singolo valore;
    - utilizzare uno storico, cioè mantenere in memoria i valori di ascisse ed ordinate delle coordinate riferiti al passo precedente;
    - cercare di fare arrivare il prima possibile il robot in asse, per ridurre la possibilità di errore in fase di avvicinamento;

5.2. Sviluppi futuri

È possibile migliorare il software prodotto, secondo i seguenti spunti:

- sviluppare un sistema omnidirezionale di visione, in quanto il campo visivo della telecamera è piuttosto ridotto e la rotazione del robot può determinare la perdita dei marcatori;
- inserire un controllo tramite *bumper* durante la fase finale di avvicinamento, per rendere più precisa l'entrata nella stazione: nel caso in cui il robot urti la stazione, arretra, corregge la sua posizione e prova nuovamente ad entrare;
- gestire particolari casi limite: quando il robot si trova molto vicino alla stazione ed è ancora fuori asse, è possibile che il "corpo" del robot stia arrivando da una direzione, ma che la telecamera si trovi già oltre l'asse, quindi il robot sbaglia nel prevedere la direzione di avvicinamento.

Bibliografia

[1] Konolige, K. G.: “Saphira Software Manual”, Sri International, California, 2001.

[2] “Aria Reference Manual”, 2003.

[3] Konolige, K. G.: “Colbert: a language for reactive control in Saphira”, Sri International, California, 1997

[4] Konolige, K. G.: “Saphira Tutorial”, Sri International,California, 2002

Indice