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

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Creazione nuove mappe del laboratorio con Tobor

Elaborato di esame di:

Bruno Cerutti, Fabio Ducoli, Luca Lombardi, Enrico Rizzardi

Consegnato il: **29 febbraio 2012**

Sommario

Il lavoro svolto consiste nello sviluppo di un software, per il robot Tobor, che permetta l'esplorazione di un ambiente qualsiasi con l'obiettivo di ricavarne una mappa perimetrale tramite l'utilizzo del sensore laser URG-04LX della Hokuyo. Tale software deve consentire al robot di circolare nell'ambiente evitando gli ostacoli. I dati rilevati dal laser vengono elaborati dal programma Mapper3 al fine di creare e modificare opportunamente la mappa.

1. Introduzione

Tobor è un robot della serie Pioneer 1 della MobileRobots, che possiede tre ruote di cui due motrici fisse ed una folle pivottante. Nella parte frontale presenta una pinza, dotata di paraurti, ed è attrezzato di sonar anteriori e laterali per il rilevamento di ostacoli.

Il sensore laser URG-04LX, montato anch'esso nella parte anteriore del robot, possiede una capacità di rilevamento da 20mm a 5600mm con un margine di errore di 10mm, nei rilevamenti dai 20mm ai 1000mm, mentre commette un errore dell'1% per rilevazioni oltre i 1000mm. L'angolo di scansione è di 240 gradi.

Tobor è dotato di un calcolatore accessibile anche da remoto e sul quale è possibile eseguire opportuni programmi per comandare il robot.

L'ambiente individuato per il progetto è il laboratorio di Robotica Avanzata della Facoltà di Ingegneria ed il relativo piazzale antistante.

2. Il problema affrontato

L'obiettivo di scansionare un ambiente al fine di crearne la relativa mappa, prevede la risoluzione di alcune problematiche. Innanzitutto, si deve interagire con il laser e capire come elaborare i dati rilevati dal sensore. Inoltre, esplorando un ambiente sconosciuto, il robot deve poter muoversi evitando gli ostacoli. Altra problematica è la modifica del file di configurazione pion1m.p in modo da testare il comportamento del robot, con il relativo laser, attraverso l'utilizzo del simulatore MobileSim.

3. La soluzione adottata

La soluzione elaborata per il problema esplicato al punto precedente può essere suddivisa in due fasi.

La prima fase è stata la comprensione del lavoro svolto dall'Ing. Mattei al fine di capire come si poteva utilizzare il laser in dotazione al robot, pertanto come e quali parametri passare al programma al fine di far riconoscere il componente che, come detto in precedenza, è un componente aggiuntivo, e normalmente la tipologia del robot usata ne è sprovvista.

La seconda fase ha riguardato l'elaborazione di un programma che permettesse al robot di navigare all'interno dell'ambiente, evitando gli ostacoli e contemporaneamente eseguisse le scansioni ed elaborasse la mappa.

In questa fase si è definito un sottoproblema, ossia quello di sfruttare MobileSim per simulare il robot pion1m, dotandolo di un laser, in quanto ne è sprovvisto.

Per questa ragione come vedremo nel paragrafo successivo, sono state definite delle procedure per la definizione di una nuova tipologia di robot che fondesse le caratteristiche del pion1m (categoria alla quale appartiene il robot Tobor, con il quale bisognava effettuare la mappatura) con quelle del laser.

3.1. Creazione di un nuovo modello di robot per il simulatore

Per definire un nuovo modello di robot utilizzabile all'interno dell'ambiente simulato è necessario modificare i due seguenti file chiave che vengono caricati dal simulatore:

- file di configurazione del robot, presente all'interno della cartella d'installazione della libreria Aria (`/usr/local/Aria/params/`). Solitamente questi file hanno la nomenclatura: *nome_robot.p*;
- file di configurazione dei modelli virtuali che vengono usati dal simulatore, presente nella cartella d'installazione del simulatore (`/usr/local/MobileSim/`). Il file in questione è denominato *PioneerRobotModels.world.inc*.

Per modificare il file di configurazione del robot è opportuno aprirlo con un qualsiasi editor di testo.

A questo punto è necessario aggiungere i parametri relativi ai nuovi componenti, per far questo si è confrontata la struttura del file con quella del file di un altro robot provvisto del laser. In questo modo si riesce ad aggiungere il laser al modello che ne è sprovvisto.

```
LaserType urg                   ; type of laser
LaserPortType serial            ; type of port the laser is on
LaserPort COM3                  ; port the laser is on
LaserAutoConnect false          ; if the laser connector should autoconnect this laser or not
LaserFlipped false              ; if the laser is upside-down or not
LaserPowerControlled true       ; if the power to the laser is controlled by serial
LaserMaxRange 0                 ; Max range to use for the laser, 0 to use default
                                ; (only use if you want to shorten it from the default), mm
LaserCumulativeBufferSize 0     ; Cumulative buffer size to use for the laser, 0 to use default
LaserX 18                       ; x location of laser, mm
LaserY 0                        ; y location of laser, mm
LaserTh 0                       ; rotation of laser, deg
LaserZ 0                        ; height of the laser off the ground, mm (0 means unknown)
```

Si è ritenuto opportuno creare un nuovo file relativo al robot col laser, in modo di non perdere la configurazione originale del robot, ossia quello senza laser. Una volta terminata questa procedura si è passati alla modifica del file di configurazione del modello virtuale di MobileSim.

Aprendo tale file, tramite un qualsiasi editor di testo, si nota che esso contiene un link di riferimento al file contenuto nella cartella `/usr/local/Aria/params/`, ed una serie di parametri per l'utilizzo all'interno del simulatore. Anche in questo caso si è provveduto ad un confronto con i parametri degli altri modelli per poter definire correttamente il nuovo robot. Di seguito troviamo il frammento di codice per la definizione di un nuovo modello:

```
define pion1mlaser pioneer (
      pioneer_robot_subtype "pion1mlaser"
```

```
color "blue"
```

Il resto del codice della definizione è la medesima del modello *pion1m.p* senza laser, tuttavia le righe contenenti il laser verranno richiamate dal file *pion1mlaser.p* definito precedentemente.

Come si può notare, all'inizio si va a definire un nuovo modello, come un sottotipo della classe pioneer ed è da sottolineare che la nomenclatura del robot deve essere la stessa usata per nominare il file all'interno della cartella `/usr/local/Aria/params/`. In caso contrario il simulatore comunicherà l'impossibilità di caricare il modello del robot poiché non è stato definito all'interno di Aria.

Una volta eseguite queste due modifiche il nuovo modello del robot è a disposizione dell'utente. Bisogna tuttavia avere l'accortezza di richiamarlo nella fase di avvio di MobileSim, poiché non compare nel menù a tendina della schermata d'avvio. Per fare ciò bisogna utilizzare il seguente comando:

```
$ MobileSim –r <path del modello>
```

Esempio:

```
$ MobileSim –r /usr/local/Aria/params/pion1mlaser
```

Il file del modello deve essere necessariamente fornito senza estensione, in caso contrario il programma non funzionerà correttamente, ed il simulatore userà il modello di default.

3.2. Creazione programma per l'esplorazione di ambienti

Per l'esplorazione dell'ambiente in cui il robot è collocato si è deciso di riutilizzare il programma "Caramelle" precedentemente sviluppato per altri progetti. Il suo compito era quello di comandare il robot affinché circolasse in un ambiente qualsiasi evitando gli ostacoli e trasportando un contenitore di caramelle da distribuire.

Il programma "Caramelle" utilizza il concetto di Action, oggetti che si occupano di regolare i movimenti del robot in una determinata situazione. Sfruttando il principio di più Action che agiscono contemporaneamente è possibile permettere comportamenti avanzati del robot, come ad esempio muoversi all'interno di una stanza evitando gli ostacoli.

Di seguito sono riportate le Action presenti nel programma "Caramelle":

- ArActionStallRecover recoverAct;
- ArActionBumpers bumpAct;
- ArActionAvoidFront avoidFront;
- ArActionConstantVelocity constantVelocityAct("Constant Velocity", 1000).

La Action "recoverAct" interviene per reagire a situazioni di stallo delle ruote, "bumpAct" per reagire ad input provenienti dai paraurti, "avoidFront" per evitare gli ostacoli di fronte mentre "constantVelocityAct" serve per stabilire la velocità di movimento del robot.

A tali Action è assegnato un peso (rappresentato da un numero intero) che identifica la priorità delle azioni durante l'esecuzione del programma.

- tobor.addAction(&recoverAct, 90);
- tobor.addAction(&bumpAct, 100);
- tobor.addAction(&avoidFront,79);
- tobor.addAction(&constantVelocityAct, 50).

Per cercare di migliorare il comportamento del robot all'interno di un ambiente non molto ampio e ricco di ostacoli come il laboratorio di robotica, si è deciso di aggiungere al progetto alcune Action non presenti in "Caramelle" e di cambiare alcuni parametri ad esse legati.

Le Action definitive sono le seguenti:

- ArActionStallRecover recoverAct;
- ArActionBumpers bumpAct;
- ArActionConstantVelocity constantVelocityAct("Constant Velocity", 600);
- ArActionAvoidFront slowFrontFarAct("Slow down on far", 1000, 400, 0);
- ArActionAvoidFront avoidFrontFarAct("Avoid Front Far", 500, 100, 60);
- ArActionAvoidFront avoidFrontNearAct("Avoid Front Near", 90, 0, 90);

Per prima cosa si può notare che si è deciso di diminuire la velocità di movimento del robot da 1000 mm/sec a 600 mm/sec, per essere più prudenti all'interno di un locale così stretto.

In seconda battuta sono state aggiunte due nuove Action di tipo ArActionAvoidFront a quella già presente. Il programma prevede quindi un comportamento denominato "slowFrontFarAct" che si occupa solamente di rallentare ulteriormente a 400 mm/sec la velocità del robot quando si trova ad una distanza di 1000 mm da un ostacolo frontale. Quando l'ostacolo è invece a 500 mm, interviene l'action "avoidFrontFarAct" che riduce la velocità a 100 mm/sec e fa sterzare il robot di 60° rispetto alla sua direzione. Infine se l'ostacolo è circa a 90 mm dal robot, "avoidFrontNearAct" ferma il robot e lo fa girare di 90° per trovare una via libera in cui proseguire il suo percorso.

I pesi attribuiti ai vari comportamenti sono qui riportati:

- tobor.addAction(&bumpAct, 100);
- tobor.addAction(&recoverAct, 90);
- tobor.addAction(&avoidFrontNearAct, 50);
- tobor.addAction(&avoidFrontFarAct, 49);
- tobor.addAction(&slowFrontFarAct,48);
- tobor.addAction(&constantVelocityAct, 25).

3.3. Creazione Mappa

Dopo aver verificato che il laser è collegato e funzionante si può procedere alla mappatura vera e propria del laboratorio (o di un altro ambiente).

Per poter far ciò si è utilizzata la classe ArLaserLogger fornita dalla libreria Aria, che si occupa di scrivere su un file i log delle rilevazioni del laser. Si è quindi aggiunta al programma la seguente istruzione:

```
ArLaserLogger logger(&tobor, sick, 300, 25, filename.c_str(),false);
```

I parametri passati al costruttore corrispondono, nell'ordine all'istanza del robot (Tobor) su cui viene eseguito il programma, all'istanza del laser da cui ricevere le rilevazioni (sick), la distanza percorsa dalla quale prendere una nuova lettura (300) ogni quanti gradi prendere una nuova lettura (25) e il nome del file su cui scrivere il log.

A questo punto si è pronti per la mappatura vera e propria dell'ambiente. Per iniziare le scansioni si devono accendere i motori di Tobor, bisogna in seguito collegarsi mediante un client ssh a Tobor e lanciare lo script di shell contenente i parametri per l'esecuzione del programma.

Assumendo di essere nella cartella in cui è presente il programma sviluppato, si deve eseguire il seguente comando:

```
$ ./run.sh
```

Il contenuto dello script *run.sh* è il seguente:

```
#!/bin/bash
./caramelle -rp /dev/ttyUSB0 -laserType urg -lp /dev/ttyACM0 -lpt serial
```

In dettaglio i comandi inviati sono:

- -rp, significa Robot port: seguito dal nome di una porta indica al programma la porta a cui connettersi per comunicare col robot. In caso non gli venisse fornita, il programma proverebbe a connettersi al simulatore;
- -laserType: serve ad fornire al programma il tipo di laser collegato. Nel nostro caso il laser è di tipo urg;
- -lp, acronimo di laser port: seguito dal path di una porta serve ad indicare il percorso della porta a cui il laser è collegato;
- -lpt, laser port type: fornisce al programma il tipo di porta utilizzato dal laser, nel nostro caso è di tipo seriale simulata.

A questo punto il robot comincerà a vagare con spostamenti casuali per il laboratorio raccogliendo i dati delle scansioni.

Quando si riterrà che i dati acquisiti siano sufficienti (ossia quando il robot è passato grosso modo in ogni parte della stanza) è possibile fermare il processo con lo shortcut da tastiera "Ctrl+c".

Ora nella cartella da cui si è eseguito lo script *run.sh* verrà creato un nuovo file chiamato *1scans.2d* contenente le rilevazioni.

- **Il file *1scans.2d* viene sovrascritto ad ogni esecuzione del programma.**

Copiare il file *1scans.2d* su Dumbbot, o su un calcolatore opportuno, tramite il seguente comando, eseguendolo dalla cartella di Tobor in cui è presente il file *1scans.2d* appena creato:

```
$ scp ./1scans.2d eserc1@dumbbot:/home/eserc1/Desktop/
```

È ora necessario convertire il file copiato su Dumbbot in formato *.map* (per poter poi essere utilizzato per l'autolocalizzazione) mediante il software Mapper3. Dalla cartella in cui è stato copiato il file *1scans.2d* eseguire:

```
$ Mapper3 1scans.2d
```

Quindi il programma richiede di specificare dove si vuole salvare il file convertito, suggerendo automaticamente come estensione *map*. Una volta scelti nome del file e la cartella di destinazione il programma Mapper3 provvederà a convertire automaticamente il file nel formato desiderato.

È importante notare che in un qualunque sistema di mappatura il laser vede tutti gli ostacoli sia fissi che mobili (ad esempio: operatori, cestini, sedie, etc…). Nell'eventualità venissero rilevati ostacoli mobili indesiderati, il software Mapper3 permetterà di rimuoverli manualmente. È inoltre possibile marcare aree come non accessibili, e aggiungere linee che il robot non deve varcare.

3.4. Mappatura laboratorio di robotica avanzata

Al fine di creare una mappa per il laboratorio di robotica avanzata è stato necessario provvedere a togliere dal laboratorio il maggior numero possibile di ostacoli mobili, in modo da favorire i rilevamenti utili da parte del robot. Tali ostacoli comprendono cestini, zaini, sedie e via dicendo, oltre a ciò è stata chiusa la porta che da sull'esterno, per scongiurare l'eventuale uscita del robot.

Al momento dell'avvio del programma per eseguire la mappatura, il robot era posizionato nella stazione di ricarica, definita convenzionalmente home.

Mentre il robot esegue le scansioni all'interno del laboratorio è possibile osservare dal terminale le rilevazioni. Ultimata questa fase, si provvede a copiare il file *1scans.2d* sul calcolatore dove è installato Mapper3, in modo da poterlo convertire in formato utile per l'autolocalizzazione.

Dopo aver aperto Mapper 3 e cliccato sul bottone open, si seleziona il file *.2d* desiderato, nell'esempio seguente il file è chiamato *mappaInterno.2d*.

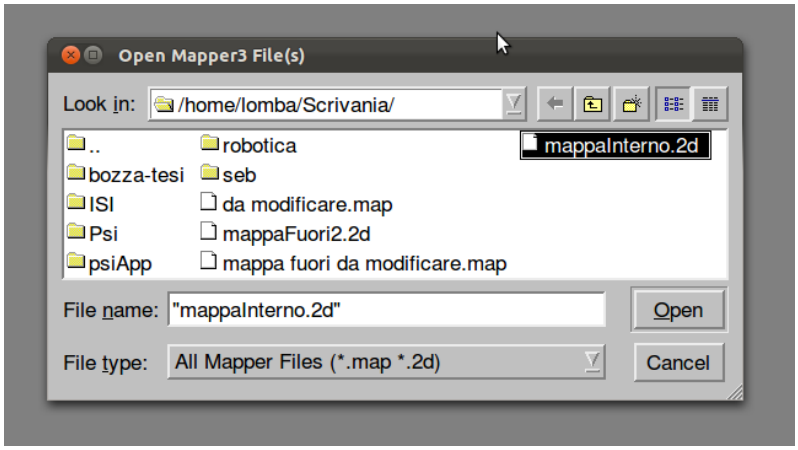

Figura 1 - Apertura file .2d

La figura 2 mostra lo screenshot di ciò che Mapper3 visualizza una volta aperto il file *.2d* non ancora elaborato. É interessante osservare che in seguito all'apertura del file, il programma ricostruisce passo a passo la mappa, mostrando a video il tragitto seguito dal robot, marcando i vari oggetti rilevati fino a definire l'ambiente esplorato.

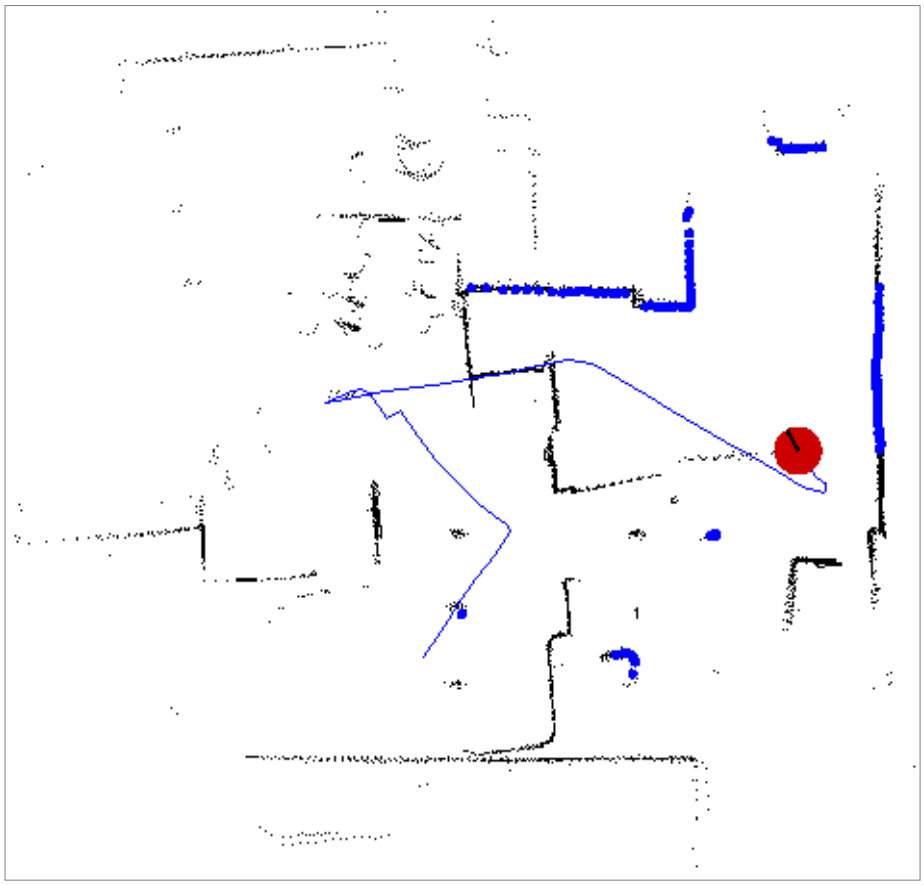

Figura 2 - Ricostruzione mappa ottenuta a seguito delle scansioni

Una volta finita la ricostruzione il programma chiede all'utente di salvare il file con estensione *map*, prima di eseguire una qualsiasi operazione di modifica. A questo punto l'utente può pulire la mappa da rilevazioni ritenute non necessarie oppure può definire aree in cui non si ritiene opportuno che il robot possa accedere.

In figura 3 è mostrata la mappa durante l'elaborazione da parte dell'utente. Quest'ultimo tramite il cursore ed un'apposita opzione, chiamata "Forbidden Area", posta fra le scelte rapide in alto, definisce le aree interdette. Le aree così definite avranno un colore differente dal resto della mappa, la quale solitamente è solo in bianco e nero, in modo che siano immediatamente riconoscibili.

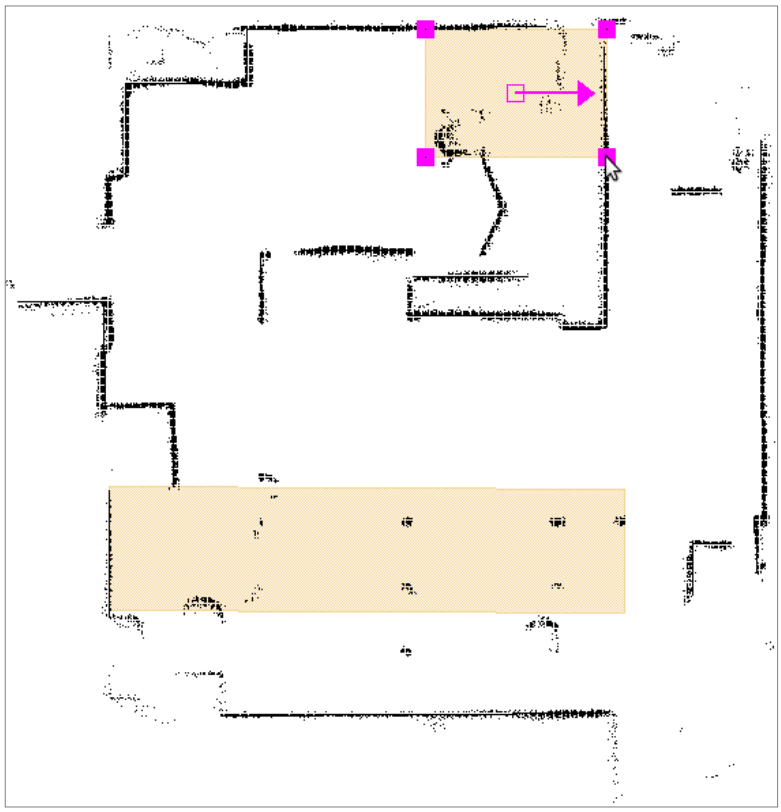

Figura 3 – Mappa in fase di modifica con Mapper3

In figura 4 abbiamo un esempio di come la mappa si presenta una volta completa. Come si può notare è evidenziata la stazione di ricarica, “Home”, raffigurata mediante un quadratino giallo, in cui è rappresentato il verso di uscita del robot da essa, tramite un’apposita linea.

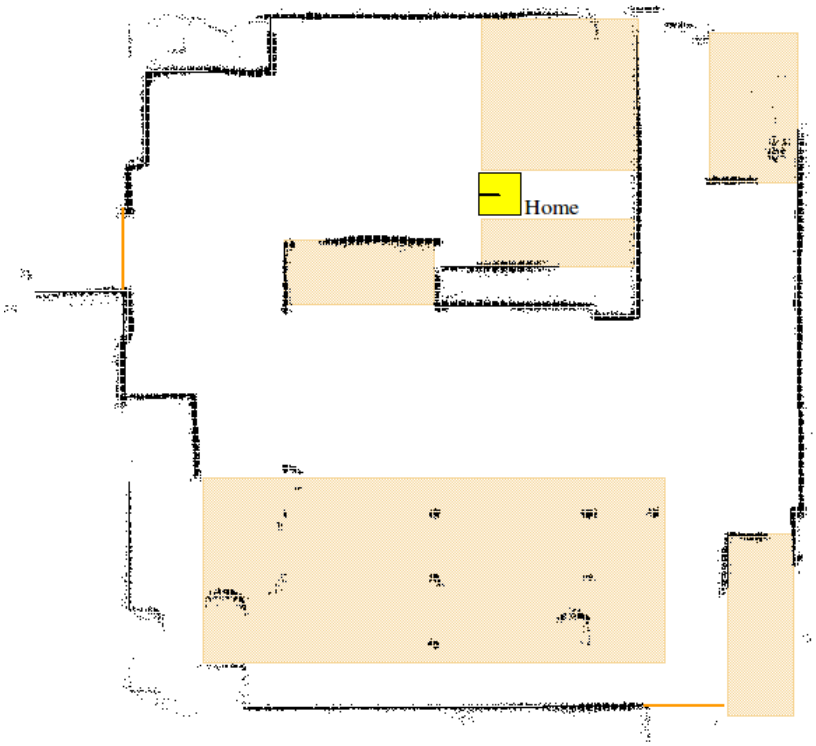

Figura 4 - Mappa completa dopo la rielaborazione

All’interno dell’immagine sono riconoscibili:

- le “Forbidden Area”: aree in cui il robot non può accedere, pur non avendo rilevato ostacoli attraverso i propri sensori;
- una “Forbidden Line”: del tutto analoga alla “Forbidden Area” ma prende in considerazione solamente una linea. Essa è utile per completare muri o porte non segnalati/e dal laser o rilevati/e in modo insufficiente;
- la stazione di ricarica, “Home”. È utile qualora si voglia definire in modo completo l’ambiente di lavoro, nella figura è rappresentato il verso di uscita da essa, tramite un apposita linea.

Una volta completato il lavoro, la mappa è pronta per essere usata da un qualsiasi programma che richieda la localizzazione del robot. In figura 5 abbiamo un esempio di come il robot riconosce la propria posizione attraverso i sensori, confronta ciò che rileva con la mappa, passata dall'utente, e ciò permette al robot di capire in che posizione si trova all'interno dell'ambiente.

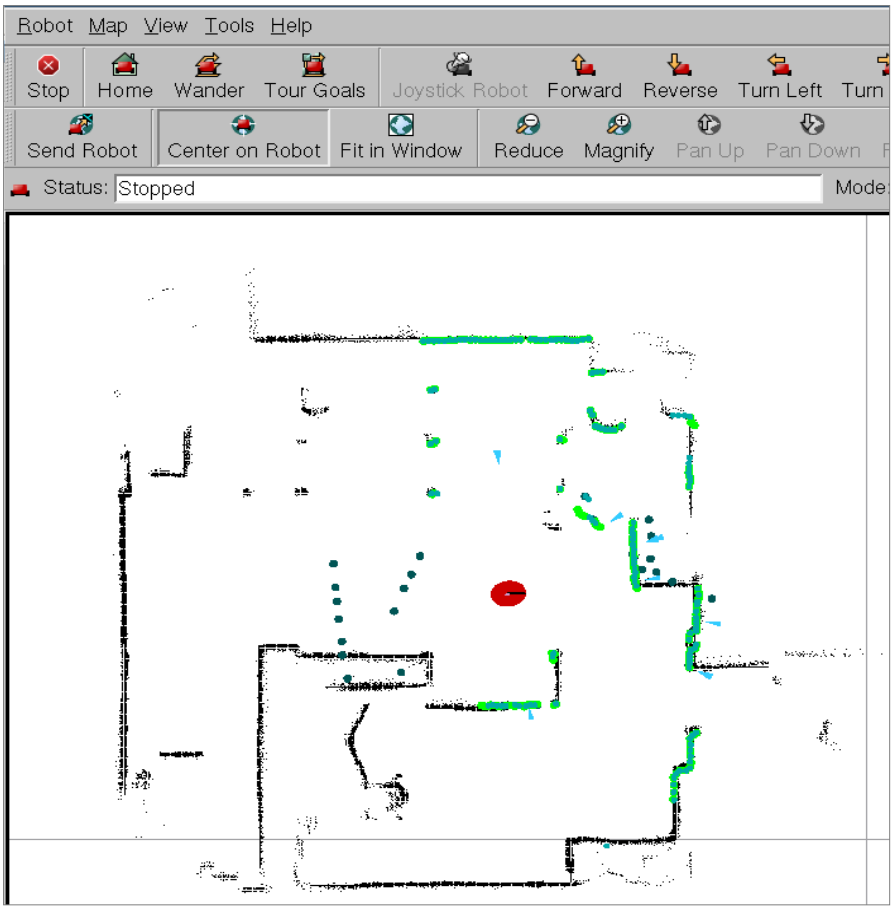

Figura 5 - Localizzazione del robot nella mappa creata

3.5. Mappatura zona antistante al laboratorio di robotica avanzata

In modo analogo a quanto presentato per la mappatura dell'ambiente interno del laboratorio di robotica presentiamo ora i risultati ottenuti per quanto concerne la zona antistante al laboratorio.

Figura 6 - Scansione dell'ambiente antistante al laboratorio di robotica

Le difficoltà maggiori per questa mappatura sono dovute a diversi fattori, tra cui: il transito e il parcheggio di veicoli, il suolo non esattamente piano e cordoli di delimitazione della aiuole troppo bassi per essere rilevati correttamente dai sensori del robot. A testimonianza di ciò si può notare come nelle rilevazioni siano presenti dei rilevamenti parziali, dovuti in buona parte agli ostacoli sopra menzionati.

Per ciò si è provveduto a introdurre delle “Forbidden Area”, allo scopo di delimitare la zona antistante il laboratorio ed impedire al robot di andare in stallo cercando di salire sul cordolo troppo basso, di altezza compresa tra 1cm e 1,5cm.

In figura 7 si mostra la mappa completa per la zona antistante al laboratorio di Robotica Avanzata. Il quadratino in verde rappresenta il punto di partenza di Tobor all'avvio dell'esecuzione del programma. In particolare, il robot è stato collocato in prossimità della porta d'accesso al laboratorio in direzione d' uscita da essa.

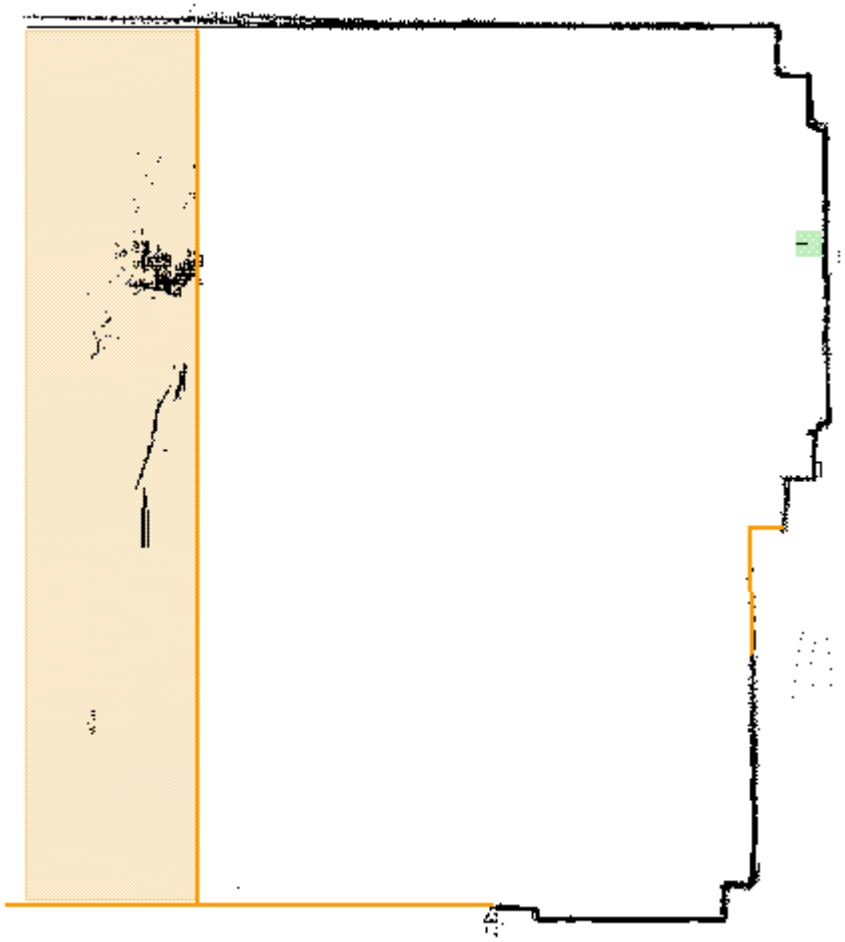

Figura 7 -: Mappa completa dopo la rielaborazione

4. Modalità operative

4.1. Componenti necessari

Hardware:

- Robot Pioneer1;
- Laser URG-04LX prodotto dalla Hokuyo;

Software:

- Sistema operativo Debian 32bit (o distribuzioni derivate);
- Aria;
- Arnl;
- Mapper3.

4.2. Caratteristiche del laser URG-04LX

Figura 8 - Laser URG-04LX prodotto dalla Hokuyo

Le caratteristiche del laser sono:

- Capacità di rilevamento: da 20 a 5600 mm (maggiore affidabilità fra 60 e 4095 mm);
- Accuratezza di rilevamento: dai 20 ai 1000 mm c'è un margine di errore di 10 mm mentre dai 1000 ai 4000 c'è un margine di errore dell' 1% della misurazione;
- Angolo di scansione: 240°;
- Risoluzione angolare: 0,36° (360/1024);
- Interfaccia: USB;
- Protocollo di comunicazione: SCIP 1.

4.3. Modalità di installazione

Per poter eseguire su Tobor il programma da noi sviluppato è necessario installare sul calcolatore del robot i pacchetti Aria e Arnl e su un'altro calcolatore Mapper3.

Innanzitutto è necessario scaricare il software. A tal fine è utile riferirsi ai seguenti collegamenti:

Aria, http://robots.mobilerobots.com/wiki/ARIA

Archivio contenente Arnl e Mapper3:

http://riffraff.ing.unibs.it/~cassinis/ARIA%20e%20annessi/Versioni%20correnti/Pacchetto%20ARNL.zip

Opzionalmente scaricare anche:

MobileEyes, http://robots.mobilerobots.com/wiki/MobileEyes

MobileSim, http://robots.mobilerobots.com/wiki/MobileSim

Tali software risultano utili qualora si volessero elaborare nuovi comportamenti del robot senza dover lavorare direttamente col robot fisico, ma con una sua versione simulata, come riportato nel paragrafo 3.1.

Per installare su Tobor il pacchetto ARIA usare il seguente comando:

```
$ sudo dpkg -i libaria_<versione_libaria>.deb
```

Estrarre il contenuto dell'archivio *Pacchetto ARNL.zip*. Nella cartella in cui è stato decompresso l'archivio verranno create delle sotto cartelle.

Posizionarsi nella sotto cartella `./Pacchetto ARNL/ARNL-SONARNL/` e copiare sul calcolatore di Tobor i pacchetti con estensione *deb* contenuti in essa. A questo punto installare tutti i pacchetti contenuti eseguendo il comando:

```
$ sudo dpkg -i <nomepacchetto1>.deb <nomepacchetto2>.deb ...
```

sostituendo a <nomepacchetto> i nomi dei vari pacchetti.

Il rimanente software, ossia Mapper3, MobileEyes e MobileSim, va installato sul calcolatore dal quale si intendono elaborare le scansioni acquisite (nel nostro caso Dumbbot), utilizzando i medesimi comandi relativi all'installazione dei pacchetti in relazione al sistema operativo utilizzato.

- **Mapper3 deve essere in versione completa e non "Basic" altrimenti non sarà possibile convertire in un formato utile i rilevamenti acquisiti con Tobor.**

Ora si può collegare tramite porta USB il laser URG a Tobor. Per verificare che il collegamento sia andato a buon fine può essere opportuno eseguire il seguente comando:

```
$ dmesg | grep URG
```

Il cui output in caso positivo è:

```
usb 1-1.2: Product: URG-Series USB Driver
```

se l'operazione non va a buon fine il comando non restituirà alcun output.

Al fine di eseguire il programma da noi sviluppato è necessario copiare su Tobor l'archivio contenente il codice sorgente del nostro elaborato. Posizionarsi nella cartella in cui è presente l'archivio e copiarlo su Tobor col seguente comando (nel nostro caso eseguito da Dumbbot):

```
$ scp ./elaborato.tar.gz user@tobor:/usr/local/Arnl/examples/
```

da Tobor, posizionarsi `/usr/local/Arnl/examples/` e decomprimere l'archivio col comando:

```
$ tar xfvz ./elaborato.tar.gz
```

Spostarsi nella sotto cartella *elaborato* appena decompressa (sempre su Tobor) e compilare il progetto con *make*:

```
$ cd elaborato
$ make
```

4.4. Avvertenze

Non propriamente legate allo sviluppo del progetto assegnato al gruppo, ma più strettamente collegato al lavoro con il robot Tobor, bisogna segnalare una serie di problematiche tecniche che hanno allungato significativamente i tempi di lavoro. Il gruppo ha rilevato una serie di problematiche legate a:

- ricarica della batteria del robot. In un caso si è dovuti procedere con la sostituzione della batteria poiché quella in uso non era più in grado di garantire una durata della carica sufficiente per una sessione di lavoro. Inoltre si è notato che spesso il robot aveva delle difficoltà a posizionarsi in modo corretto sui contatti della stazione di ricarica, pertanto pur rientrando correttamente, la batteria rimaneva scollegata dall'alimentazione scaricandosi progressivamente.
- Ricarica della batteria del calcolatore. Come per il caso del robot, citato al punto precedente, i contatti dell'alimentazione del calcolatore montato non venivano toccati con quelli posti sulla stazione, non effettuando pertanto l'operazione di ricarica.

Per tali cause il lavoro del gruppo è stato rallentato; le tempistiche ovviamente sono stati condizionate maggiormente dai tempi di ricarica in cui il gruppo rimaneva inoperativo in attesa che le batterie immagazzinassero sufficiente energia per la sessione di lavoro.

Un'altra problematica è legata alla costruzione della mappa, poiché sia il laser che il robot commettono errori di misurazione in relazione all'accuratezza del laser ed all'odometria ed al percorso compiuto dal robot.

5. Conclusioni e sviluppi futuri

Lo sviluppo di un programma che permetta di effettuare la mappatura del laboratorio in maniera automatizzata, ha consentito di analizzare e risolvere le problematiche di sviluppo legate all'utilizzo di nuove librerie, scritte in C. La difficoltà maggiore è sicuramente stata l'interfacciamento tra il programma e il sensore laser, ma comunque risolta tramite l'aiuto delle documentazione fornita dal produttore dell'hardware e l'analisi di altri elaborati.

Gli sviluppi futuri che si possono prevedere partendo da questo progetto sono legate in particolar modo al completamento del programma stesso, in quanto sarebbe interessante sviluppare un'applicazione di tipo client – server al fine di monitorare tramite altri programmi ciò che il sensore rileva con le proprie scansioni. Questo tipo di approccio era stato pensato inizialmente, tuttavia è stato abbandonato poiché si distaccava troppo da quello che era l'obiettivo principale dell'elaborato.

L'architettura pensata prevedeva:

- il server venisse eseguito direttamente sul robot;
- il programma client venisse eseguito da remoto, in modo da ottenere direttamente su un altro calcolatore la mappa da elaborare attraverso il programma Mapper3;
- da remoto venisse poi lanciato un programma di monitoraggio (es: MobileEyes) che collegandosi al robot, tramite il server, visualizzasse graficamente le rilevazioni del laser.

Riferendosi sempre al software, sarebbe interessante elaborare della action che consentano di dare un comportamento più intelligente al robot, per avere una mappatura più efficiente. Durante l'elaborato sono state provati vari gruppi di action che permettevano di avere un comportamento di tipo wander (gira a caso) e un comportamento invece legato al rilevamento e inseguimento di un muro.

Per la conformazione del laboratorio e per come è stata implementata la seconda funzione, il comportamento casuale di wander ha portato ad avere una mappatura completa in meno tempo. Sarebbe pertanto interessante elaborare un nuovo gruppo di action che migliori ulteriormente il programma implementato.

Terzo ed ultimo sviluppo, che risulta invece legato al simulatore, è la possibilità di sviluppare a piacimento nuovi modelli di robot, completando la documentazione dei file contenuti all'interno della cartella `/usr/local/Aria/params`, e della cartella `/MobileSim`.

Attualmente sono stati sviluppati dei modelli che rispecchiano il robot reale, ma la documentazione è insufficiente e presenta gravi lacune.

Bibliografia

[1] Cassinis, R.: “Titolo di un articolo a congresso mai scritto”, in *Proc. III International Symposium on Useless Computer Programs*, Honolulu, 1975.

[2] Cassinis, R., Pallino, P.: “Titolo di un articolo su rivista”, *Useless Computing* Vol. 27, marzo 2003.

[3] Cassinis, R. et al.: “Articolo senza alcun senso logico”, in*: A book of nonsense papers*, Pallino, P. ed., Mondadori Editore, Milano, 1854.

Indice