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)

Miglioramento della qualità delle mappe generate dal programma "Caramelle"

Elaborato di esame di:

Gianfranco Toninelli 86867

Consegnato il: **29 Luglio 2013**

Sommario

Il progetto consiste nel migliorare il dettaglio delle mappe di un programma (Caramelle) realizzato con lo scopo di generare nuove mappe tramite le rilevazioni del laser URG-04LX montato sul robot Tobor della Pioneer.

1. Introduzione

Tobor è un robot della MobileRobots costituito da tre ruote (due motrici fisse ed una folle pivottante, quindi è in grado di compiere rotazioni "sul posto"), da sonar anteriori e laterali per il rilevamento di ostacoli e da una pinza frontale. Sul robot del laboratorio ARL è montato un sensore laser URG-04LX, posizionato frontalmente, 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. (caratteristiche tecniche prese dal sito del costruttore).
L'angolo di scansione è di 180 gradi.

Il programma "Caramelle" è stato realizzato per generare mappe attraverso l'esplorazione dell'ambiente da parte del robot e le rilevazioni col laser, si basa sul perseguimento di un percorso che segue le pareti e di azioni atte ad evitare gli ostacoli.

2. Il problema affrontato

Il problema principale del programma Caramelle è il fatto che per alcune rilevazioni, certi dettagli della mappa non vengono mai riconosciuti; questo è dovuto dalla limitata capacità angolare del laser montato sul robot in laboratorio, che essendo di 180 gradi ha una visione solo frontale dell'ambiente.

A causa di questa visione alcuni dettagli della mappa vengono persi, generando così una mappa che per certi ambienti presenta "buchi" in zone che invece dovrebbero essere raggiungibili dal robot.

Un altro difetto del programma è che per certi ambienti più complessi, con molte stanze e di grandi dimensioni la mappa risulta spesso incompleta.
Per il laboratorio ARL il programma va più che bene, ma per altri ambienti questi difetti non sono trascurabili.
Per ovviare a questo problema sono necessarie azioni per un'esplorazione più sistematica e per una visuale a 360°.

3. La soluzione adottata

Il primo passo per un corretto svolgimento dell'elaborato è la comprensione del funzionamento delle librerie Aria e quindi in particolare del programma "Caramelle".

Il passo successivo consiste nel simulare l'ambiente di esecuzione (robot, laser e mappe) attraverso il programma MobileSim.

L'ultimo passo è la progettazione dei cambiamenti da applicare al software esistente.

Per la comprensione delle librerie Aria è stata consultata la documentazione in rete [1], mentre per la comprensione del funzionamento del software "Caramelle" è stato consultato il codice sorgente, al quale verranno apportate le modifiche necessarie.

3.1. Funzionameto del programma “Caramelle”

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;
- 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);

La Action “recoverAct” interviene per reagire a situazioni di stallo delle ruote, “bumpAct” per reagire ad input provenienti dai paraurti, “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(&recoverAct, 90);
- tobor.addAction(&avoidFrontNearAct, 50);
- tobor.addAction(&avoidFrontFarAct, 49);
- tobor.addAction(&slowFrontFarAct,48);
- tobor.addAction(&constantVelocityAct, 25).

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

Di seguito vengono riportate le operazioni da svolgere per una corretta simulazione dell’ambiente di esecuzione, tali istruzioni sono state prese dalla relazione degli studenti che hanno realizzato il software “caramelle” [2]:

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.

3.3. Soluzioni per un’esplorazione più strutturata

Per l’esplorazione di ambienti con caratteristiche diverse differenti da quelle del laboratorio di robotica o dell’ambiente esterno (ambiente ampio ma poco complesso, le complessità maggiori sono date da ostacoli ad un altezza troppo bassa per essere rilevata dal laser), è necessario un movimento meno casuale e più sistematico, è stata quindi aggiunta un’azione creata in un precedente elaborato [3] che consente al robot si seguire il perimetro degli ostacoli che incontra:

- ArActionTrack segui("segui", 130, 30, 10, 70);

L'azione "segui" permette di seguire il perimetro delle pareti (o ostacoli), i parametri sono la distanza dagli ostacoli da mantenere, l'isteresi della distanza da mantenere, l'angolo di cui ruotare in caso di avvicinamento o allontanamento eccessivo, e la velocità da mantenere.

Tale azione è stata quindi aggiunta al set di azioni del robot con la relativa priorità.

- tobor.addAction(&segui, 30);

Per poter fare ciò è stata fatta una copia esatta del file del programma precedente e sono state aggiunte le azioni appena descritte.

Per quanto riguarda la visuale a 360° è stato scelto di simularla via software facendo compiere al robot delle rotazioni di 360° gradi.
Tale azione è studiata in modo da poter decidere ogni quanto tempo effettuare una rotazione completa, di quanti gradi alla volta far ruotare il robot (max 180°), e quanto tempo far passare tra una rotazione parziale e l'altra, tale tempo di attesa è necessario per far sì che il laser faccia in tempo a "fotografare" l'ambiente nella posizione scelta.

La nuova azione è composta nei due file ArActionLookAround.h (file delle definizioni) e ArActionLookAround.cpp.

Tale azione, al contrario delle altre utilizzate per il progetto, non è soggetta alla priorità, nel senso che viene svolta comunque il qualsiasi situazione allo scadere del tempo di attesa prefissato (per questo può essere utilizzata soltanto con robot in grado di ruotare completamente senza effettuare altri spostamenti).

L'azione viene creata in questo modo:

- ArActionLookAround look("look",2000,180,15000);

I tre parametri da passare sono: il tempo che intercorre tra una rotazione parziale e l'altra (in msec), l'entità di ogni rotazione parziale, e il tempo di attesa tra una rotazione completa e l'altra.
I dati inseriti indicano che il robot effettuerà una rotazione completa ogni 15 secondi, ogni rotazione sarà composta da 2 rotazioni da 180° intervallate da 2 secondi di pausa.
Questa azione garantisce che, indipendentemente dalle azioni effettuate durante l'attesa tra una rotazione parziale e l'altra, alla fine dell'azione completa il robot si ritrovi nella stessa posizione (di rotazione) in cui si trovava all'inizio della rotazione.

L'azione è stata quindi inserita nel set delle azioni del robot, e la priorità è stata posta a scopo puramente indicativo, in quanto l'azione verrà compiuta indipendentemente dalla priorità assegnata.

- Tobor.addAction(&look,29);

3.4. Istruzioni per l'esecuzione del programma

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

```
$ ./launch.sh
```

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

```
#!/bin/bash
```

```
./TrovaMappa -rp /dev/ttyUSB0 -laserType urg -lp /dev/ttyACM0 -lpt serial –lasertype urg –lds -90 –lde 90
```

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.
- -lasertype, dipo di laser (parametro non necessario perché viene letto automaticamente dal robot).
- -lds, ampiezza in gradi iniziale del laser (parametro non necessario perché viene letto automaticamente dal robot).
- Lde, ampiezza finale in gradi del laser, (parametro non necessario perché viene letto automaticamente dal robot).

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.

3.5. Risultati Ottenuti

Le prove riguardanti il nuovo programma sono state effettuate tramite simulazioni con diverse mappe preesistenti, tra cui quella del laboratorio ARL.

Sono state effettuate delle simulazioni con MobileSim su due mappe, una è quella del laboratorio ARL, una mappa piccola che veniva ricostruita bene anche col vecchio "Caramelle"; l'altra mappa è un tipo di mappa più grande e con più stanze.

Questa è la mappa del laboratorio utilizzata per la simulazione:

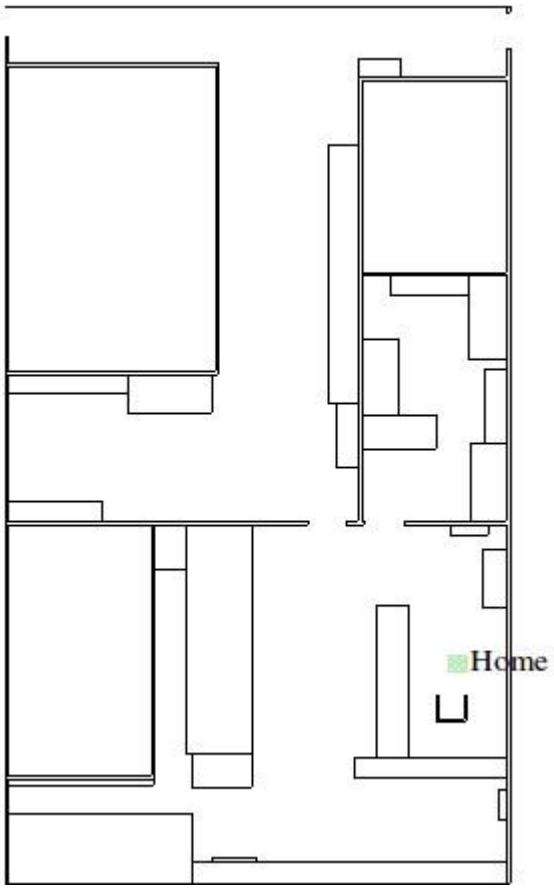

Fig 1: mappa del laboratorio di robotica ARL utilizzata nel simulatore

Questa è la mappa ricostruita con il programma "Caramelle":

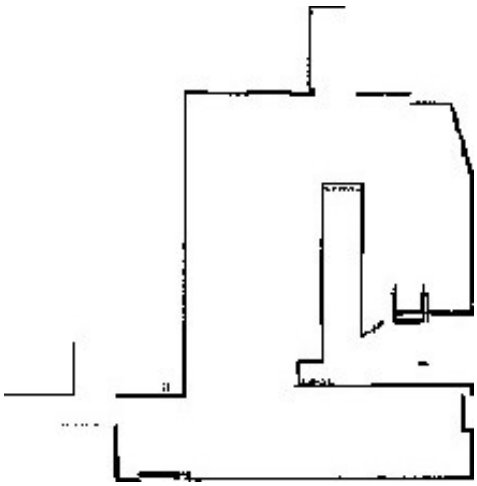

Fig 2: mappa del laboratorio di robotica ARL ricostruita col programma “Caramelle”

Di seguito invece quella col programma migliorato:

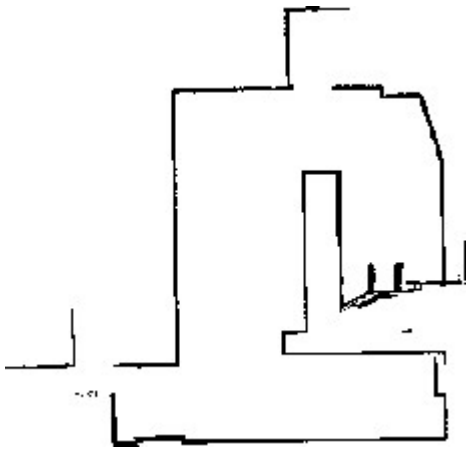

Fig 3: mappa del laboratorio di robotica ARL ricostruita col programma “TrovaMappa”

Come si può vedere in entrambi i casi la mappa è ricostruita correttamente, tranne che nei punti in cui il robot non poteva fisicamente passare.

Questa è invece la mappa più complessa (si trova nella cartella d’installazione di Aria, nella sottocartella maps, office.map)

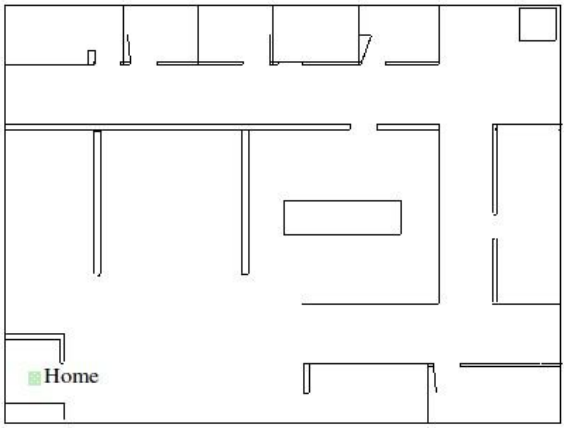

Fig 4: mappa di un ambiente "complesso"

Questa è la ricostruzione fatta con "Caramelle"

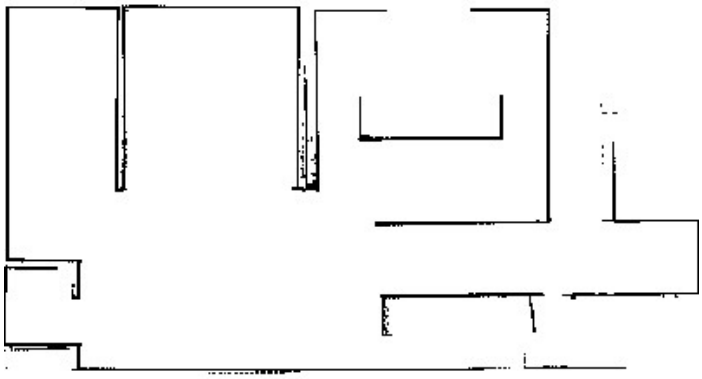

Fig 5: mappa ricostruita col programma "Caramelle"

La ricostruzione non è completa, e il robot continuava a percorrere lo stesso ciclo all'infinito senza esplorare nuove aree.

La ricostruzione appare corretta invece col programma "Trovamappa":

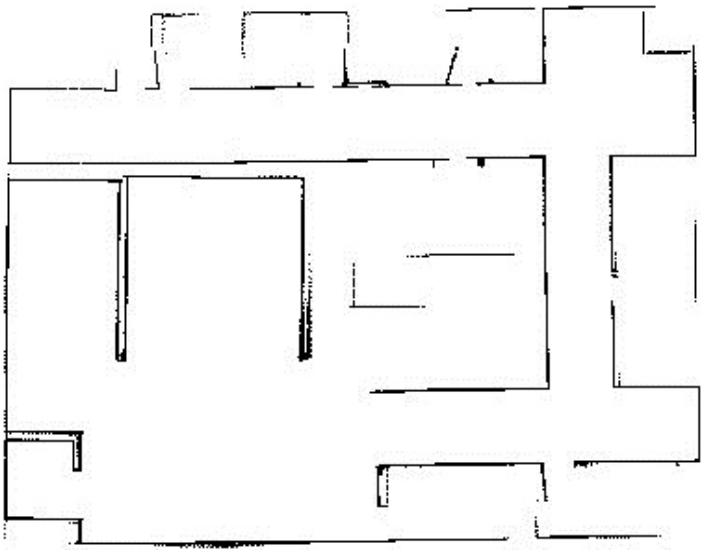

Fig 6: mappa ricostruita col programma “TrovaMappa”

Questo perché seguendo il perimetro delle pareti il robot tende a percorrere il perimetro esterno con la stessa probabilità con cui esegue il percorso interno (quello percorso col programma “Caramelle”), in questo modo viene meno la deterministicità e il percorso appare casuale, e questo permette di coprire tutte le zone della mappa.

Nella fase di lettura delle scansioni si è notato che il percorso del robot eseguito col programma “Caramelle” risulta essere più “pulito” e a seguire un percorso preciso; invece nella scansione delle mappe generate con “TrovaMappa” il percorso è risultato casuale, andando a ricoprire quasi tutti gli spazi della mappa.

4. Modalità operative

Le modalità di installazioni e le caratteristiche tecniche sono le stesse utilizzate per il programma”Caramelle”, le informazioni descritte in seguito sono quindi prese dalla relazione di tale elaborato [2].

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 1 - 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
```

5. Conclusioni e sviluppi futuri

Lo sviluppo del programma non ha portato a grosse difficoltà, in quanto la struttura base era già costruita col programma “Caramelle”.
La difficoltà maggiore è stata capire il funzionamento del programma preesistente e delle librerie Aria, successivamente l’elaborazione e lo sviluppo delle nuove azioni non ha portato grandi difficoltà.

Il programma funziona egregiamente e le mappe appaiono complete, la cosa migliorabile più evidente riguarda le tempistiche con cui la mappa viene navigata.
In futuro si potrebbero apporre migliorie e ottimizzazioni sull’esplorazione dell’ambiente, in quanto spesso vengono esplorate zone già visitate più volte prima di passare in zone della mappa non ancora esplorate, questo non ha portato ad alcuna incompletezza nella ricostruzione della mappa, ma i tempi potrebbero essere ridotti ad esempio mantenendo in memoria i percorsi già effettuati e completati.

Bibliografia

[1] Documentaizone Librerie Aria

http://www.ing.unibs.it/~arl/docs/documentation/Aria%20documentation/Current/.

[2] B. Cerutti, F. Ducoli, L. Lombardi, E. Rizzardi *: Creazione nuove mappe del laboratorio con Tobor.*
http://www.ing.unibs.it/~arl/docs/projects/Nav_27.pdf

[3] D. Ferrari, M. Guion: *WallTracking: un programma per l’esplorazione perimetrale di un ambiente*.
http://www.ing.unibs.it/~arl/docs/projects/Nav_21.pdf

Indice