POLITECNICO DI MILANO

Dipartimento di Ingegneria Informatica

E-2?: SVILUPPO DI COMPORTAMENTI PER ROBOT AUTONOMO DI ATTRAZIONE PER AMBIENTI PUBBLICI

AI & R LAB
LABORATORIO DI INTELLIGENZA ARTIFICIALE
E ROBOTICA DEL POLITECNICO DI MILANO

Relatore: Prof. Andrea Bonarini

Elaborato finale di:

Francesco Balzani, matricola 669515

Anno accademico 2008/2009

Sommario

Lo scopo di questa tesi è quello di sviluppare dei comportamenti di un robot da intrattenimento che possa operare in maniera autonoma per poter sostituire (anche se solo in parte) le funzioni svolte da esseri umani. Tale robot è chiamato “E-2?” che in inglese suona come “Hei tu?” e richiama “ET” l'extra-terrestre il cui aspetto estetico è vicino a quello del robot utilizzato. Il presente progetto si colloca nell’ambito della *robotica*, una scienza che studia i comportamenti degli esseri intelligenti, cercando di sviluppare delle metodologie che permettano ad una macchina chiamata *“robot”*, dotata di opportuni dispositivi atti a percepire l'ambiente circostante ed interagire con esso quali sensori e attuatori, di eseguire dei compiti specifici, tipicamente umani. I robot da intrattenimento sono progettati in modo da essere accattivanti, simpatici e divertenti, ma solitamente sono poco autonomi e le possibilità di modificarne la struttura o il comportamento è in genere assai scarsa o nulla. Il robot “E-2?” si presenta come un robot in grado di muoversi autonomamente all’interno di un area prestabilita delimitata sul pavimento. All'interno di tale area è in grado di avvicinarsi a persone, evitare collisioni con ostacoli e fornire informazioni su un determinato evento interagendo con i visitatori mediante frasi sintetizzate, espressioni del viso e movimenti del collo. Il lavoro svolto include lo sviluppo di tutti i comportamenti del robot e l'interfacciamento delle parti meccaniche con il software di controllo che gestisce e ne governa i movimenti permettendo al robot di evitare collisioni con ostacoli, identificare le persone nel proprio campo visivo, raggiungerle e intrattenerle. Tali funzionalità sono state testate in simulazione ed in ambienti semplici e controllati: il prototipo del robot risolta corrispondente agli obiettivi proposti.

Ringraziamenti

Ringrazio la mia famiglia che mi ha sempre sostenuto e motivato in questa lunga e difficile scelta di vita. Un grazie speciale ad Alessandra per essere, in ogni momento, un punto di riferimento nel caos costante della mia esistenza.

Grazie a Paola, che nelle situazioni più difficili, mi ha aiutato a dare il giusto valore a ciò che mi stava accadendo.

Ringrazio il prof. Andrea Bonarini per avermi dato la possibilità di partecipare ad un progetto così concreto e stimolante; grazie ai ragazzi dell'AIRLab: Simone Ceriani per avermi supportato pazientemente nel critico approccio a MRT, Davide Rizzi per le fatiche condivise e per le giornate trascorse insieme.

Infine, non meno importanti, ringrazio Luca ed Andrea, compagni di questo tanto amato, quanto faticoso cammino, per aver condiviso questi anni che rimarranno per sempre scolpiti nella mia memoria e che hanno contribuito a creare ciò che sono diventato.

Grazie a tutti gli amici.

Cocce

INDICE

INDICE DELLE ILLUSTRAZIONI

CAPITOLO 1
INTRODUZIONE

Lo scopo di questa tesi è quello di sviluppare dei comportamenti di un robot da intrattenimento che possa operare in maniera autonoma in ambienti pubblici per poter sostituire (anche se solo in parte) le funzioni svolte da esseri umani. Tale robot è chiamato “E-2?” che in inglese suona come “Hei tu?” e richiama “ET” l'extra-terrestre il cui aspetto estetico è vicino a quello del robot utilizzato. Il presente progetto si colloca nell’ambito della *robotica*, una scienza che studia i comportamenti degli esseri intelligenti, cercando di sviluppare delle metodologie che permettano ad una macchina chiamata *“robot”*, dotata di opportuni dispositivi atti a percepire l'ambiente circostante ed interagire con esso quali sensori e attuatori, di eseguire dei compiti specifici, tipicamente umani. I robot da intrattenimento sono progettati in modo da essere accattivanti, simpatici e divertenti, ma solitamente sono poco autonomi e le possibilità di modificarne la struttura o il comportamento è in genere assai scarsa o quasi nulla. Nella realtà alcuni di essi sono considerati poco più che bambole animate molto sofisticate. Uno degli impulsi maggiori nello sviluppo di

robot sempre più sofisticati nasce dall'esigenza di aggredire l'industria dell'intrattenimento. Le motivazioni alla base del progetto includono la necessità di poter fornire servizi di intrattenimento interattivo da parte di robot più o meno umanoidi, rendendo tali servizi diversi e innovativi.

Il lavoro svolto include lo sviluppo di tutti i comportamenti (espressi mediante regole basate su logica fuzzy) del robot e l'interfacciamento delle parti meccaniche con il software di controllo che gestisce e governa i movimenti. I comportamenti sviluppati permettono al robot di evitare collisioni con ostacoli, identificare le persone nel proprio campo visivo, raggiungerle e intrattenerle. L'intrattenimento viene supportato da una moltitudine di espressioni del viso; il movimento della bocca, delle sopracciglia e degli oggi rendono il robot molto espressivo. Una volta avvicinata la persona destinata a ricevere informazioni il robot saluta ed interagisce riproducendo frasi create da un software di sintesi vocale. E' stato inoltre introdotta la possibilità di prendere il controllo completo anche a distanza mediante un joypad senza fili.

Le funzionalità richieste quindi per rendere il servizio di intrattenimento interattivo speciale e soprattutto diverso dal solito che il robot deve essere in grado di effettuare sono:

- Movimento autonomo
- Evitamento degli ostacoli
- Limitare il movimento in un'area ben definita
- Identificazione di una persona come obiettivo
- Raggiungimento dell'obiettivo
- Interazione con l'obiettivo una volta raggiunto

- Movimenti del collo
- Movimenti della testa
- Espressioni del viso
- Sintesi vocale per riprodurre frasi
- Controllo remoto
- In caso di collisioni, abilitare un blocco di sicurezza che possa rendere sicuro il servizio

Il presente lavoro si è svolto basandosi su un architettura multiagente resa disponibile dal framework DCDT (Device Communities Development Toolkit) [1]. Grazie alle caratteristiche modulari del framework anche il software realizzato risulta di facile integrazione ed estensione. Tutte le funzionalità sono state inoltre testate in ambienti semplici e controllati e mediante un software di simulazione chiamato MRTSim.

Il prototipo del robot con le funzionalità appena descritte ha partecipato alla fiera della Robotica 2009 aprendo la manifestazione.

Di seguito viene fornita una breve descrizione delle parti che compongono il presente documento:

- Nel capitolo 2 viene presentato un elenco dei programmi che rendono possibile il funzionamento del robot, dal sistema di base ai moduli di controllo che permettono la comunicazione tra i componenti hardware ed il software. Sono inoltre accennati alcuni aspetti teorico/matematici degli algoritmi utilizzati per il riconoscimento del viso e per il tracciamento degli oggetti in movimento.

- Nel capitolo 3 vengono illustrate tutte le fasi di progetto, cominciando dalla fase di analisi dei requisiti e degli obiettivi da raggiungere. Vengono quindi analizzate e discusse le scelte effettuate: l'elenco dei componenti necessari per ottenere dei risultati soddisfacenti, le caratteristiche richieste per il software di controllo, il numero di comportamenti necessari e le regole per il loro controllo.

- Nel capitolo 4 viene presentata in dettaglio la scelta implementativa adottata, successivamente vengono considerati i possibili domini applicativi e nella parte finale vengono presentate le problematiche riscontrate durante la realizzazione del progetto e le soluzioni adottate.

- Il capitolo 5 termina il documento dimostrando la validità delle scelte fatte e del lavoro svolto per raggiungere gli obiettivi prefissati. Si affrontano inoltre alcuni limiti e si descrivono ulteriori sviluppi futuri dell'applicazione legati all'evoluzione del robot.

- Nell'appendice si riporta il codice utilizzato per la configurazione ed esecuzione del sistema “Mr.Brian” di controllo del robot.

CAPITOLO 2
SOFTWARE E TECNOLOGIE UTILIZZATE

Il progetto è stato sviluppato in ambiente Linux utilizzando il linguaggio C++ per la realizzazione del codice di interfacciamento con l'hardware e mediante regole espresse in logica fuzzy. Sono stati inoltre utilizzati software e hardware precedentemente creati dal "Politecnico di Milano" nell'ambito del progetto MRT (Modular Robot Toolkit) [2]:

- DCDT, un middleware per lo scambio di messaggi all'interno dei componenti di MRT
- Mr.Brian, per la programmazione e gestione dei comportamenti del robot basati sulla logica fuzzy
- MRTSim, per la simulazione virtuale dei comportamenti

Sono state inoltre riutilizzate parti di codice sorgente C++ già sviluppati dall'AIRLab per il progetto LURCH [3]. In questa sezione vengono descritti tutti gli strumenti hardware e software su cui il progetto si basa. Saranno elencati i software ritenuti fondamentali per la realizzazione del progetto "E-2?".

Figura 2.1: Il robot E-2?

Nello specifico verrà presentato un elenco dei programmi che rendono possibile il funzionamento del robot, dal sistema di base ai moduli di controllo che permettono la comunicazione tra i componenti hardware ed il software. Sarà inoltre citato il programma di simulazione MRTSim che è stato utilizzato per stesura dei comportamenti e la validazione delle regole fuzzy prima che il robot fosse fisicamente completato. Verranno inoltre accennati alcuni aspetti teorico/matematici

degli algoritmi utilizzati per il riconoscimento del viso e per il tracciamento degli oggetti in movimento.

2.1 Hardware

La configurazione hardware di partenza è basata su un precedente progetto sviluppato dal Politecnico di Milano chiamato Milan Robocup Team [4]. In particolare la struttura hardware di base del progetto è stata realizzata partendo dal robot calciatore chiamato "*IANUS*". Esso incorpora due ruote nella base, ognuna alimentata da un motore che funziona a 12V. Le ruote sono allineate al centro del corpo.

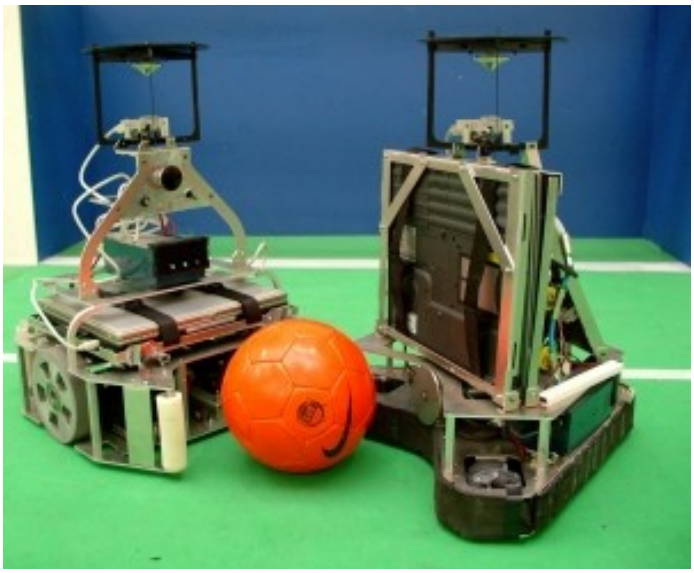

Figura 2.2: Robot giocatori di calcio. Da sinistra IANUS e TRISKAR

2.1.1 Personal Computer

Il personal computer *PCBrick* è costituito da una scheda madre Mini-ITX, equipaggiata con un processore Core2 Duo T7200 @2GHz, 2 GB di RAM, un HardDisk da 80 GByte di dimensioni standard 2.5 pollici. La piattaforma garantisce elevate prestazioni computazionali, senza sacrificare troppo spazio o consumi.

Figura 2.3: Il computer "*PCBrick*"

2.1.2 Sensori ad infrarossi

Il sistema per il rilevamento delle linee bianche, realizzato in AIRLab, è

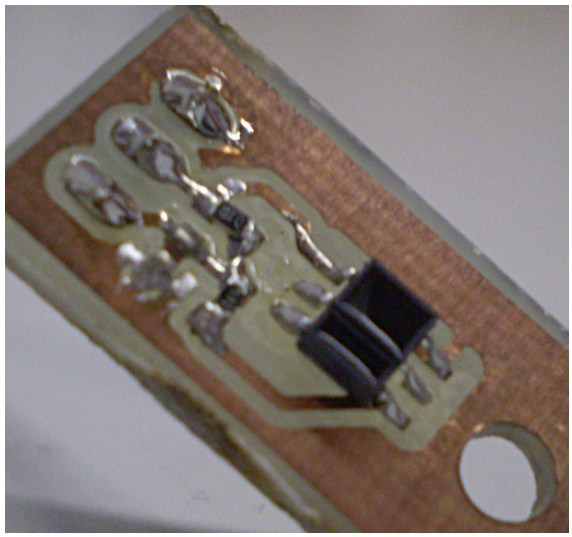

Figura 2.4: Sensore IR ad infrarossi

costituito da 8 sensori ad infrarossi collegati ad un micro controllore. Tale sistema viene utilizzato dal robot per identificare sul pavimento la zona in cui operare. I sensori sono composti da una coppia led-fototransistor ad infrarossi, incapsulati in un unico package. Il sensore produce un valore di tensione proporzionale alla quantità di luce riflessa, che viene poi convertito in digitale per essere inviato al PC. Con una operazione di taratura della soglia è possibile discriminare se il sensore si trova sul pavimento o su una superficie diversa che presenti un contrasto rispetto al pavimento.

2.1.3 Sonar

I sonar offrono diversi tipi di output, nel caso si è optato per l'uscita PWM, in cui l'ampiezza di emissione è proporzionale alla distanza misurata. La cintura sonar, sviluppata ed assemblata in AIRLab, è realizzata con 7 sensori Maxbotix Ez-2, collegati ad un unico micro controllore. Il micro controllore si occupa di sequenziare i sonar e di convertire le distanze misurate dai singoli sensori (max 4 contemporaneamente) e di inviarle al PC. Tale apparato viene utilizzato per misurare la distanza del robot dagli ostacoli. Viene anche utilizzato in cooperazione con il modulo di visione per identificare la distanza del target da raggiungere.

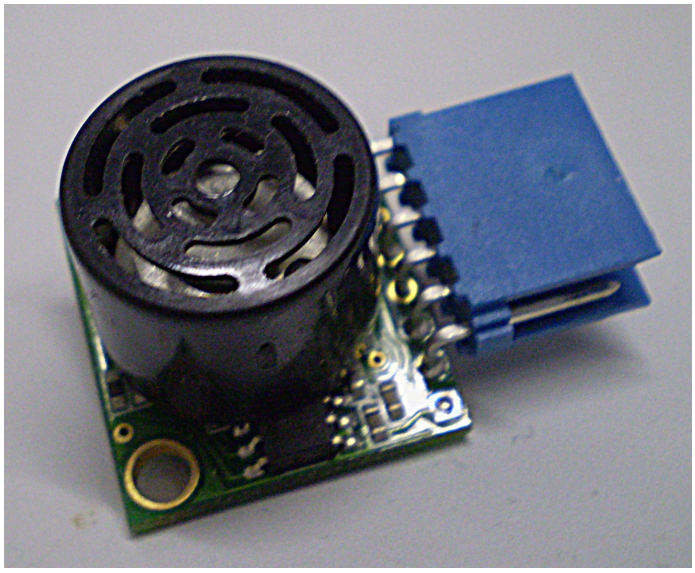

Figura 2.5: Dispositivo sonar

2.1.4 Motori

Figura 2.6: Motore

Figura 2.7: Scheda di controllo dei motori

2.1.5 Servomotori

Il sistema dei servomotori che muovono testa e robot si articola attorno ad una scheda realizzata ad hoc in AIRLab, in grado di comandare fino a 12 servomotori da modellismo. Il controllo avviene ovviamente in posizione, ma la scheda è in grado di limitare sia velocità che accelerazione, interpolando i setpoint da dare ai motori. Inoltre offre la possibilità di salvare nella flash del microcontrollore anche sequenze predefinite, in modo da rendere più semplice l'utilizzo della stessa scheda. Tale apparato è utilizzato per il controllo delle espressioni facciali ed il movimento del collo e della testa.

2.1.6 Camera

La camera è una micro camera UVC (USB Video Class), del tipo impiegabile come webcam da incorporare in notebook o smartphone. Le dimensioni ridotte, dell'ottica in particolare, ne facilitano l'occultamento. Con una risoluzione di 640x480 pixel offre il giusto compromesso tra complessità computazionale e performance della procedura di identificazione dell'obiettivo, nello specifico degli algoritmi di "*HaarCascade*" e "*CAMShift*".

Figura 2.8: Micro camera UVC

2.1.7 Speakers

Alla scheda audio incorporata sulla motherboard del PC Brick è collegato un amplificatore recuperato ed adattato da un sistema stereo da PC. Su questo canale vengono sintetizzate le frasi e le espressioni che il robot propone al suo interlocutore durante l'interazione.

2.1.8 Alimentazione

L'alimentazione è assicurata da 8 batterie SLA (Sealed Lead Acid), batterie sigillate al piombo, con tensione e capacità nominali di 12V e 7.2Ah. Le batterie sono collegate come un parallelo di 4 serie da 2, in modo da offrire 24V, essendo questa la tensione di funzionamento dei motori. Il PC è alimentato tramite una propria PSU di tipo automotive. I sensori, i servomotori e il sistema di amplificazione audio sono alimentati da una tripla PSU in grado di fornire le tensioni richieste (5V, 6V e 14V). Con le batterie impiegate l'autonomia misurata è maggiore di 5h di funzionamento continuo.

2.1.9 Dispositivo di comando manuale a distanza

Questo dispositivo di controllo si interfaccia al computer mediante un'interfaccia di comunicazione di tipo USB (Universal Serial Bus). E' composto da un controllo digitale con quattro frecce direzionali e due analogici composti da una leva mobile con range angolare di 360°. Sono presenti inoltre 12 pulsanti digitali programmabili di cui due attivabili mediante le due leve analogiche. Ha una portata di trasmissione/ricezione di circa 10 metri. La carica completa delle batterie permette un'autonomia di circa 100 ore.

Figura 2.9: Il joypad Logitech Rumble Pad 2 senza fili

2.2 DCDT: The Middleware

Un aspetto essenziale per lo sviluppo e l'implementazione di un progetto di robotica, consiste nell'integrazione dei componenti che comandano il robot.

A tale scopo è stata necessaria la creazione di un sistema base, un middleware, che riesca a incorporare i vari strumenti disponibili e che ne garantisca la loro interazione.

Il software DCDT (Device Communities Developement Toolkit), sviluppato all'interno dell'AIRLab e parte integrante del progetto MRT, è un sistema orientato alla gestione degli agenti. Un agente è un'entità astratta, è rappresentata nel codice da classi di tipo Expert.

Per svolgere l'attività a cui sono destinati, gli agenti, hanno bisogno di eseguire i loro compiti in totale autonomia e di poter scambiare messaggi con gli altri agenti operanti nel medesimo ambiente. DCDT rende disponibile un ambiente multiagente senza gravare l'utente degli aspetti legati alla programmazione in multithreading e della comunicazione tra i diversi thread.

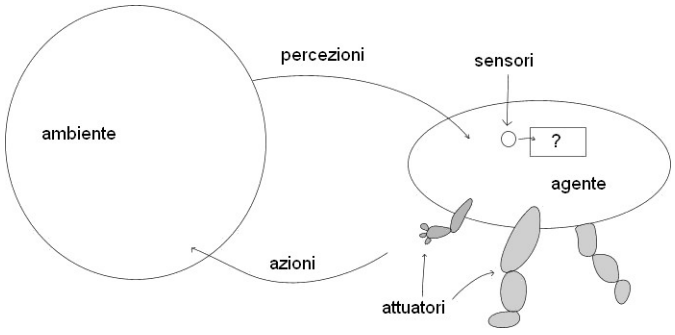

Figura 2.10: Interazione di un agente intelligente con l'ambiente esterno

Tale strumento permette allo sviluppatore di gestire con facilità la creazione e la comunicazione di più agenti, anche se tali agenti operano su computer differenti.

2.2.1 Dispatcher e agenti

La processo di scambio dei messaggi è rappresentabile con un modello di un ipotetico ufficio postale che ha il compito di ricevere, smistare e consegnare i messaggi mediante una classe chiamata *Dispatcher* che svolge il lavoro di postino. DCDT ha una politica pubblica/sottoscrivi che nasconde la distribuzione fisica dei messaggi. In questo modello, tutti gli agenti hanno la possibilità di pubblicare messaggi inviandoli all'ufficio postale. Ad ogni messaggio è assegnato un tipo; un agente ha la possibilità di sottoscriversi ad una lista di interesse, specificando il tipo di messaggi che intende ricevere. In questo modo solo i diretti interessati possono ottenere le informazioni precedentemente richieste e non vengono, quindi, prodotte copie inutili dello stesso messaggio, aumentando l'efficienza e l'affidabilità del sistema. Nel caso in cui dei messaggi arrivino nell'ufficio postale e non esista nessun agente che ha sottoscritto quella tipologia di messaggio, i messaggi vengono scartati.

Il file principale necessario alla creazione di un robot è *kernel.cpp*, del quale vengono commentati i punti principali. Per una guida completa a DCDT si rimanda al suo manuale utente. Nell'esempio sotto si illustra la possibile creazione del *dispatcher* con inserimento di un esperto.

```
// SonarExpert
// dichiarazione dell'esperto
#define SONAR

#ifdef SONAR
```

```
	#include "SonarExpert.h"
	int sonar_period=20000; // in microsecondi
#endif

// definizione del file di configurazione di dcdt.
// permette di impostare se DCDT modalità di esecuzione(locale/rete)

string dcdt_config_file("./config/dcdt.conf");

int main(int argc, char** argv){
	// creazione del dispatcher
	ModuleDispatch *Dispatch = new ModuleDispatch
			((char*)dcdt_config_file.c_str(), basename(argv[0]));
	// creazione della lista degli agenti
	std::vector < StringModuleMember * >agents;
	#ifdef SONAR
		// creazione dell'esperto SonarExpert
		SonarExpert *sonarExprt = new SonarExpert(Dispatch,
								sonar_period);
		// aggiunta dell'esperto alla lista di agenti
		agents.push_back(sonarExpert);
	#endif
	// assegnamento della lista degli agenti al dispatcher
	for (std::vector<StringModuleMember*>::iterator i = agents.begin();
			i != agents.end (); i++)
		Dispatch->AddMember (*i);
	// attivazione di tutti gli agenti creati
	for (std::vector<StringModuleMember*>::iterator i = agents.begin();
			i != agents.end (); i++)
		Dispatch->ActivateMember(*i);
	// attivazione del dispatcher
	Dispatch->LetsWork();
}
```

Ogni singolo agente dovrà implementare i metodi RunInit() e RunDuty() e seguirà una struttura come quella dell'esempio seguente:

```
// costruttore
SonarExpert::SonarExpert(ModuleDispatch* kernel,int period) :
			StringModuleMember(kernel,period,"SonarExpert"){
	readSonar=0;
	device=string("/dev/ttyUSB0");
	to_meter = 0.1;
	ifstream OpenFile("./config/sonar/sonar_config.txt");
	char buffer[255];
	while(!OpenFile.eof()) {
		OpenFile >> buffer;
	}
	device=string(buffer);
#ifdef EXPERTS_DEBUG
```

```
	cout << "SonarExpert::Open device:'" << buffer << "'" << endl;
#endif
	OpenFile.close();
	cout <<"\n SonarExpert created!"<<endl;
};
// distruttore
SonarExpert::~SonarExpert(){
	 cout<<"\n SonarExpert destroyed!"<<endl;
};
// il metodo RunInit() viene eseguito al momento della creazione dell'agente
void SonarExpert::RunInit(){
	// Abilitala ricezione dei messaggi di tipo MSG_FROM_BRIAN
	AddMessageRequest( MSG_FROM_BRIAN, LOCAL );
	cout << endl " SonarExpert initialized!"<<endl;
};

// il metodo RunDuty() viene eseguito in modo ciclico
void SonarExpert::RunDuty(){
	/*
	il metodo ReceiveLastMessage riceve l'ultimo messaggio del
	tipo specificato ed elimina l'esperto this dalla lista degli
	esperti in attesa di ricezione
	*/
	if((ReceiveLastMessage(MSG_FROM_BRIAN, buffer,FALSE) > 0)){
		// Do something...
	}
	// End...
}
```

E' importante precisare che il metodo *RunDuty()* viene eseguito ciclicamente ad intervalli regolari, definiti dalla costante *period*, durante la fase di creazione dell'agente.

Assegnando un valore zero al parametro *period*, l'agente verrebbe eseguito in modo continuo. Il tempo di esecuzione di un agente viene influenzato sia dal codice che lo stesso implementa che dall'istruzione di ricezione dei messaggi *ReceiveLastMessage*, infatti, impostando il secondo parametro a *true* l'esecuzione dell'agente verrebbe interrotta fino a alla ricezione di un messaggio del tipo richiesto. Diversamente specificando il valore *false* l'esecuzione del flusso del codice procederebbe anche in mancanza dell'arrivo di un nuovo messaggio.

2.3 Mr.Brian

I comportamenti del robot sono stati implementati mediante regole espresse in logica fuzzy. Mr.Brian (Multilevel Ruling BRIAN) [5], è un estensione di BRIAN (BRIAN Reacts by Inferring ActioNs) software sviluppato dal Politecnico di Milano all'interno del progetto MRT. Esso permette di utilizzare dei predicati espressi in logica fuzzy per fornire le regole di comportamento che il robot impiegherà per svolgere i propri compiti in modo autonomo.

Questo strumento permette di sviluppare rapidamente ed in modo modulare i comportamenti che si vogliono fare assumere al robot e, nella sua estensione Mr.Brian, consente di specificare le modalità con cui essi interagiscono, definendo anche delle gerarchie fra gli stessi.

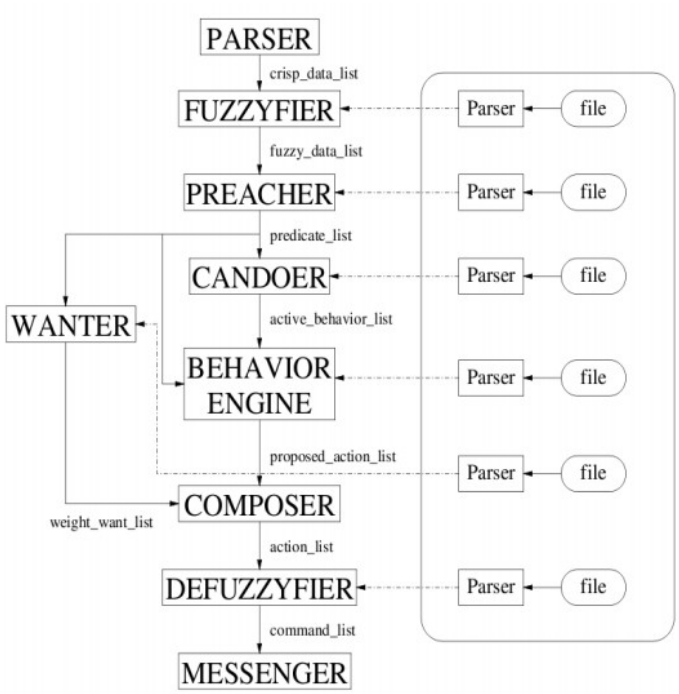

Figura 2.11: Schema di funzionamento di Brian

Tralasciando, inizialmente, gli aspetti legati alle gerarchie fra i vari comportamenti e riferendoci in particolare al software Brian, il processo che, a partire dai dati di input, produce in output l'azione di controllo è il seguente:

1. Fuzzyficazione dei dati di ingresso.
2. Valutazione del valore di verità dei predicati.
3. Scelta dei comportamenti da attivare.
4. Valutazione delle regole dei singoli comportamenti.
5. Fusione dei risultati.
6. Defuzzyficazione dei risultati.

2.3.1 Logica fuzzy

La logica fuzzy, detta anche logica sfumata o logica sfocata, fu introdotta per la prima volta da Zadeh nel 1965 [6] ed da quel momento è stata largamente utilizzata nella realizzazione di controllori robotici. E' una logica mediante la quale si può attribuire a ciascuna proposizione un grado di verità compreso tra 0 e 1. È una logica polivalente, e pertanto un'estensione della logica booleana. È fortemente legata alla teoria degli insiemi sfocati e, già intuita da Cartesio, Bertrand Russell, Albert Einstein, Werner Karl Heisenberg, Jan Łukasiewicz e Max Black, venne concretizzata da Lotfi Zadeh.

Con grado di verità o valore di appartenenza si intende quanto è vera una proprietà: questa può essere, oltre che vera (= a valore 1) o falsa (= a valore 0) come nella logica classica, anche pari a valori intermedi.

Come mostrato in figura 2.12, le tre funzioni, *freddo* (in blu), *tiepido* (in arancio) e *caldo* (in rosso) sono rappresentate nel diagramma riferite alla comune variabile, la temperatura. Ogni rilevamento della temperatura quindi ha tre valori logici, uno per ciascuna delle tre funzioni. Una misura della temperatura utilizzata per prevenire dei blocchi del sistema dovuti a temperature dell'acqua troppo fredde è mostrata in figura come una linea grigia verticale. Finché la freccia rossa punta a zero, la funzione *Caldo* non è vera (temperatura non calda, con operatori matematici: "*NOT Caldo*"). La freccia arancione (che punta a 0,2) indica che la funzione *Tiepido* è vera solo in piccola parte (si può descrivere a parole come "*un po tiepido*"); al contrario la freccia blu (che punta a 0,8) indica che la funzione *Freddo* è abbastanza vera ("abbastanza freddo").

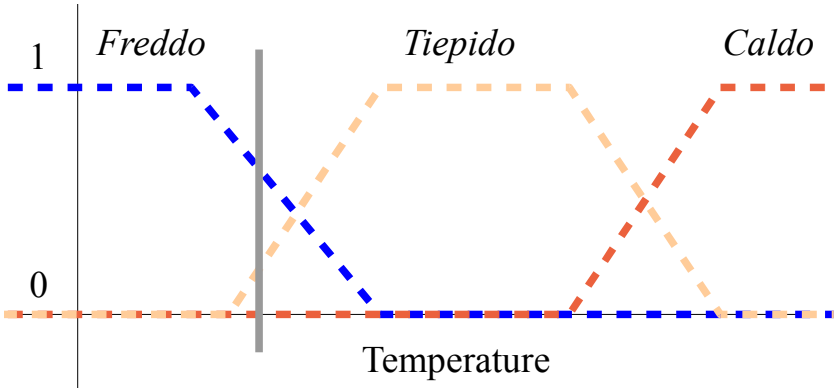

Figura 2.12: Esempio di logica fuzzy

La teoria degli insiemi fuzzy costituisce un'estensione della teoria classica degli insiemi poiché per essa non valgono i principi aristotelici di non-contraddizione e del terzo escluso (detto anche "*tertium non datur*"). Si ricorda che, dati due insiemi "*A*" e "*!A*" (non-A), il principio di non-contraddizione stabilisce che ogni elemento appartenente all'insieme A non può contemporaneamente appartenere anche a non-A; secondo il principio del terzo escluso, d'altro canto, l'unione di un insieme A e del suo complemento non-A costituisce l'universo del discorso. In altri termini, se un qualunque elemento non appartiene all'insieme A, esso necessariamente deve appartenere al suo complemento non-A. Il più antico e forse celebre di tali paradossi è quello attribuito ad Eubulide di Mileto (IV secolo a.C.), noto anche come paradosso del mentitore, il quale, nella sua forma più semplice, recita:

"Il cretese Epimenide afferma che il cretese è bugiardo"

Essendo una proposizione autonegante, riassumibile nel significato:

"Questa frase è falsa"

non è possibile dimostrare se tale affermazione sia vera o falsa con un approccio di logica tradizionale. Grazie alle logica fuzzy,essendo basata su un insieme di valori di verità più ampio rispetto al "*vero* o *falso"* della logica aristotelica, associando un valore di verità che può variare tra 0 e 1, il paradosso ammetterebbe una soluzione impostando un valore di verità pari a 0,5 ad entrambi i predicati.

Grazie a questo tipo di approccio la logica fuzzy si presenta come potente e semplice strumento nello sviluppo di applicazioni software (sistemi basati sulla conoscenza) con l'obiettivo di rimpiazzare, per quanto possibile l'esperto umano cui spettano le decisioni, e per la gestione di sistemi di controllo di apparecchiature hardware.

I punti di forza della logica fuzzy sono i seguenti:

- E' concettualmente facile da capire
- E' flessibile
- Tollera dati imprecisi rendendo il controllo molto robusto
- Può modellare funzioni non lineari di complessità arbitraria
- Può essere costruita sulla base dell'esperienza degli esperti
- Si può unire alle tecniche di controllo convenzionali
- Si basa sul linguaggio naturale.

2.3.2 Fuzzyficazione

E' il procedimento attraverso il quale le variabili di ingresso (ad esempio misure, temperature, etc...) vengono convertite in misure fuzzy della loro appartenenza a determinate classi come ad esempio Nulla, Bassa, Media, Alta, Molto Alta. Tale conversione da grandezze deterministiche a fuzzy viene effettuata attraverso le funzioni di appartenenza predefinite per quelle classi. Per utilizzare tale meccanismo è necessario specificare i valori di input e le tipologie di variabili fuzzy ad essi associate. Ad esempio:

```
(PersonDistance DISTANCE)
```

dichiara che il dato in ingresso *PersonDistance* è di tipo *DISTANCE*. E' inoltre necessario specificare il tipo di dato fuzzy *DISTANCE*.

Ad esempio:

```
(DISTANCE
(SNG (OUT_OF_RANGE 0))
(TRA (IN_CONTACT 1 1 40 40))
(TRA (VERY_CLOSE 35 40 70 110))
(TRA (CLOSE 70 110 130 190))
(TRI (NEAR 130 190 250))
(TOR (FAR 190 250))
(TOR (VERY_FAR 250 650))
)
```

definisce il tipo di variabile *DISTANCE* come mostrato in figura 2.13

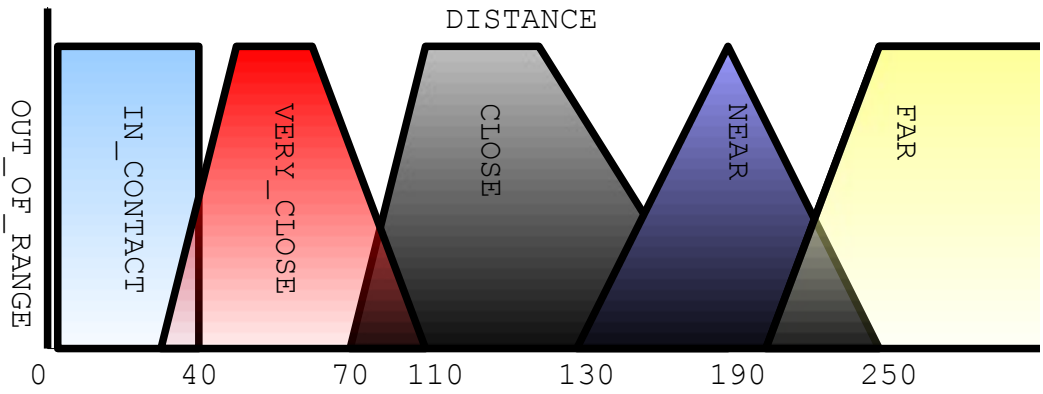

Figura 2.13: Esempio di un insieme fuzzy usato in E-2?

Nella figura 2.14 vengono mostrati tutti i tipi di insiemi fuzzy supportati da Mr.Brian.

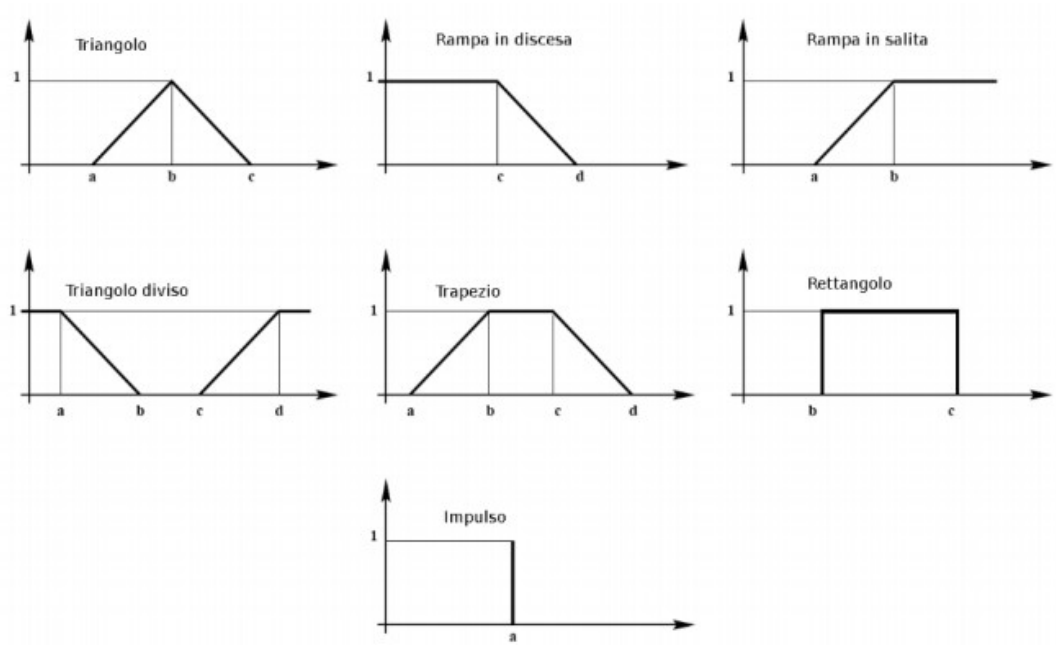

Figura 2.14: Tipi di insieme fuzzy supportati da Mr.Brian

2.3.3 Definizione dei predicati

Una volta definite le variabili fuzzy e la loro tipologia è necessario dichiarare i predicati ad esse associati. In logica fuzzy un predicato è un letterale a cui si associa un valore di verità con range da 0 a 1.

Ad esempio:

```
PersonVeryClose = (D PersonDistance VERY_CLOSE);
```

definisce il predicato *PersonVeryClose* che avrà un valore di verità uguale al grado di appartenenza del dato *PersonDistance* all'interno dell'insieme *VERY_CLOSE*, definito precedentemente.

E' inoltre possibile definire dei predicati composti a partire da altri, tramite le operazioni booleane, che in logica fuzzy vengono così rappresentate:

```
NOT P = 1-<P>
S OR P = max(<S>,<P>)
R AND P = min(<R>,<P>)
```

dove *R* e *S* sono predicati generici, mentre <*R*> e <*S*> sono i rispettivi valori.

2.3.4 Scelta dei comportamenti da attivare

In Brian, tramite regole fuzzy, è possibile stabilire quali comportamenti **possono** essere eseguiti (condizioni *CANDO*). Ad ogni comportamento viene associato un predicato il cui valore corrisponderà al valore con cui esso dovrà essere attivato.

Ad esempio:

```
InteractWithPerson =
              (AND
                     (AND (P CanMove)(P OpModeAuto))
                     (P PersonVeryClose)
              );
```

indica che il comportamento *InteractWithPerson* può essere attivato se la persona è molto vicina, il robot funziona in modalità automatica ed il robot ha la possibilità di potersi muovere (cioè non è in stato di blocco dovuto a malfunzionamenti di dispositivi fisici).

2.3.5 Valutazione delle regole

Dopo aver identificato i comportamenti attivi, cioè quelli il cui valore di attivazione è superiore ad una determinata soglia, Brian valuta le singole regole relative ad ognuno di essi.

Ogni regola è espressa nel seguente modo:

```
(antecedente) => (variabile VALORE) (variabile VALORE) …
```

Dove l'antecedente è un predicato (o una composizione di essi), variabile è il nome di una variabile fuzzy corrispondente ad un uscita e VALORE è l'insieme fuzzy di valori assegnati a tale variabile. La variabile fuzzy assume il valore indicato se il valore dell'antecedente è maggiore ad una certa soglia; alla coppia variabile-valore proposta dalla regola è associato anche il valore dell'antecedente della regola stessa che rappresenta l'importanza della regola e quindi del valore da essa generato. Più regole possono specificare valori diversi per la stessa variabile, in questo caso, il comportamento proporrà più coppie variabile-valore per la stessa variabile e ad ognuna di esse sarà associata un'importanza diversa, in base al valore dell'antecedente della regola che l'ha generata.

2.3.6 Fusione dei risultati

Su tutte le coppie variabile-valore proposte da tutte le regole di tutti i comportamenti attivi, viene calcolata la media dell'importanza, tale media sarà pesata sul valore di WANT associato ad ogni comportamento. E' infatti possibile associare ad ogni comportamento, con la stessa sintassi delle regole CANDO, delle regole WANT che rappresentano quando e con che intensità un certo comportamento deve essere eseguito.

2.3.7 Defuzzyficazione

Così come i dati in ingresso (variabili reali) sono trasformati in variabili fuzzy, i valori fuzzy proposti dai vari comportamenti sono trasformati in valori reali. Brian implementa, attualmente, solo insiemi di tipo impulsivo per definire le variabili fuzzy usate come uscita. In questo caso il processo di defuzzyficazione consisterà semplicemente nel calcolare la media dei vari valori associati ad una variabile, pesata sull'importanza di ogni coppia variabile-valore.

2.3.8 Mr.Brian, l'evoluzione multilivello

L'approccio di Brian permette di sviluppare i comportamenti in modo modulare e riutilizzabile. Inoltre, dividendo le condizioni CANDO da quelle WANT, permette, quando si riutilizza un comportamento in situazioni diverse, di riutilizzare le condizioni CANDO, modificando solo quelle WANT. Questo tipo di approccio non

permette di gestire eventuali conflitti che possono verificarsi tra i diversi comportamenti. Ad esempio il comportamento *GoToPerson* potrebbe proporre di girare a sinistra, mentre allo stesso momento, il comportamento *AvoidObstacles* potrebbe proporre di ruotare a destra, compromettendo il risultato voluto dal comportamento *GoToPerson*. Situazioni del genere potrebbero essere evitate modificando opportunamente le condizioni *CANDO* e *WANT* associate ai singoli comportamenti, ma questo vanificherebbe il principio di sviluppo modulare e completamente autonomo degli stessi.

L'estensione di Brian, chiamata Mr.Brian risolve tale ostacolo introducendo la possibilità di specificare una gerarchia per i vari comportamenti. Il flusso di controllo utilizzato da Mr.Brian risulta essere il seguente:

1. Fuzzyficazione dei dati di ingresso.
2. Valutazione del valore di verità dei predicati.
3. Scelta dei comportamenti da attivare.
4. Valutazione delle regole dei singoli comportamenti del livello i.
5. Fusione dei risultati.
6. Defuzzyficazione dei risultati.
7. se i è l'ultimo livello
    1. allora fine
    2. altrimenti incrementa i, ritorna al punto 1

I dati in ingresso analizzati al punto 1 saranno, oltre ai dati sensoriali, anche i valori in uscito al punto 6 del livello precedente e i predicati dipendenti da essi saranno rivalutati ad ogni livello. Le regole dei vari comportamenti potranno quindi basarsi su predicati che rappresentano i valori proposti dai livelli precedenti e,

inoltre, possono annullare del tutto i valori proposti dai livelli precedenti. Ad esempio il comportamento *AvoidObstacles*, può annullare ogni spostamento proposto dagli altri comportamenti, se posto ad un livello superiore, con la seguente regola:

```
(AND (OstacoloVicinoFrontale) (DirAvanti)) =>(&DEL.cmd_v ANY)(cmd_v STEADY);
```

2.4 MRTSim

Il software MRTSim, sviluppato dal Politecnico di Milano, è un sistema di simulazione basato su Linux e le librerie grafiche Qt [7]. Integra un motore per la simulazione della fisica di corpi rigidi chiamato ODE (Open Dynamics Engine).

2.4.1 ODE – Open Dynamics Engine

ODE (Open Dynamics Engine) [8] è una libreria software opensource per la modellazione basata sulla fisica, particolarmente adatta per simulazioni della dinamica di veicoli terrestri, robot antropomorfi e oggetti generici in ambienti virtuali. Tra le caratteristiche di interesse vi sono:

- implementazione multipiattaforma
- processo di integrazione stabile e veloce
- supporto per vari tipi divincoli meccanici
- sistema built-in per il rilevamento delle collisioni
- gestione dell'attrito
- ottima integrazione con le librerie grafiche.

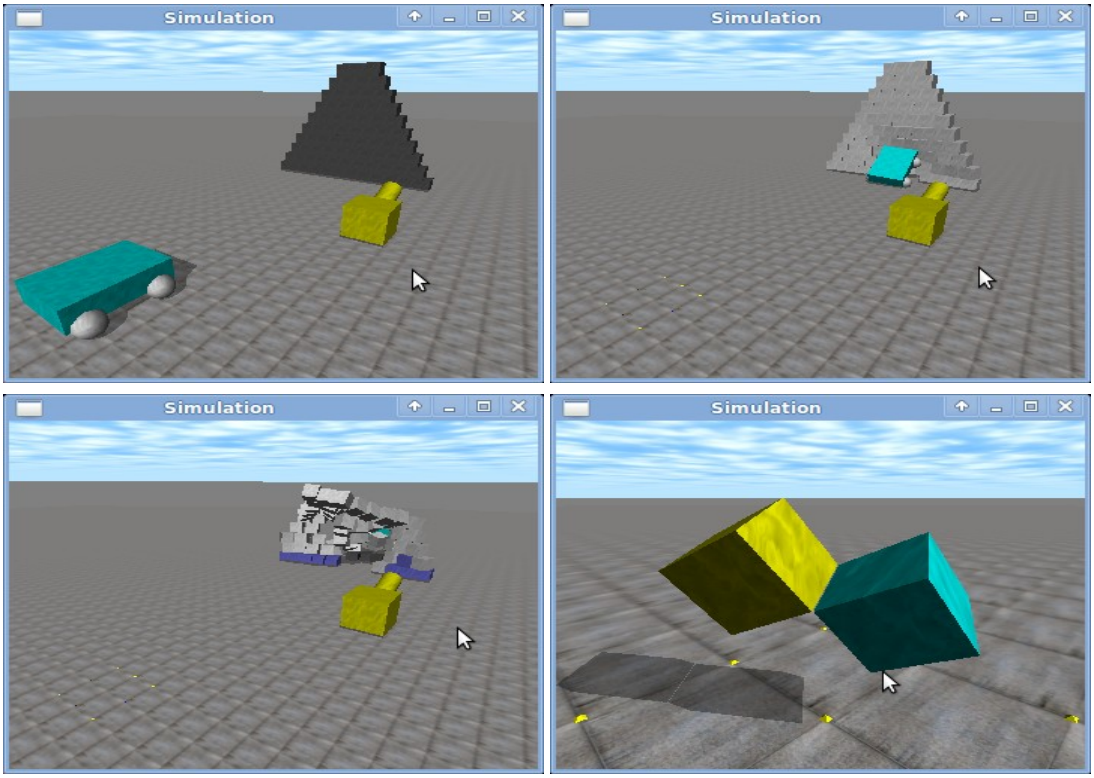

Figura 2.15: Esempi di implementazione con la libreria ODE: .

Nella figura 2.15 vengono mostrate alcuni esempi di implementazione (gestione delle collisioni e dei collegamenti) realizzabili con la libreria ODE.

2.5 OpenCV

Uno dei campi più affascinanti della ricerca scientifica è volto alla riproduzione artificiale delle capacità umane. Tra queste capacità la visione, intesa come pura acquisizione di immagini, è attualmente considerabile un problema già risolto, o

almeno un punto già segnato, visto che le capacità visive di sistemi ottici e relativi sensori hanno ampiamente superato le possibilità dell'occhio umano in quanto a sensibilità, velocità e risoluzione. Il passo successivo, cioè la capacità di interpretare ed utilizzare correttamente le informazioni acquisite, presenta invece ancora molti problemi insoluti. Convertire un'immagine in informazioni "oggettive" astraendone il contenuto dalla pura rappresentazione luminosa, sebbene sia un'operazione banale per un cervello umano adulto è, a tutt'oggi, un problema di elevata complessità per un sistema automatico.

Le librerie open source OpenCV (Open Source Computer Vision) [9] sono state create proprio per questo scopo. La versione iniziale della libreria è stata sviluppata da un gruppo di ricerca sponsorizzato da Intel. E' infatti parzialmente basata sulla Intel Image Processing Library (IPL). Attualmente è stato inserito nel prodotto commerciale chiamato IIPP (Intel Integrated Performance Primitives) [10].

Il problema principale è quello di estrarre dalle immagini dati significativi e trattabili in modo automatico. Tale campo di studio trova le sue applicazioni più comuni nella robotica, nei sistemi di videosorveglianza evoluti e nei sistemi di monitoraggio e sicurezza, oltre che in ogni sistema di archiviazione automatica di informazioni visive.

Tale libreria include attualmente più di 300 funzioni, che coprono le più svariate esigenze di trattamento di immagini, comprese funzioni matematiche ottimizzate (elevamento a potenza, logaritmi, conversioni cartesiane-polari, ecc.) ed un completo pacchetto di algebra matriciale, sviluppato funzionalmente al resto del sistema.

Durante la fase di acquisizione immagine il software cattura un'immagine mediante una camera. La camera frontale è localizzata sulla testa del robot. Essa ha una risoluzione di 640x480 pixel nel formato YUV411. Una volta catturata l'immagine, viene processata dal modulo di riconoscimento del viso. Tale processo analizza l'immagine per identificare il bersaglio e seguirlo durante tutti gli spostamenti all'interno del campo visivo della robot.

Per il riconoscimento del target (persona da intrattenere) è stata utilizzata una combinazione di 2 specifici algoritmi: *Haar matching* e *CAMShift tracking*. Mediante il primo algoritmo viene identificata la faccia all'interno del campo visivo del robot, successivamente mediante il secondo algoritmo, si mantiene “agganciato” l'oggetto che viene rilasciato solamente dopo un numero prestabilito di secondi se non viene identificato una nuovo viso. Al contrario per identificare un nuovo viso e liberare il precedente oggetto, il viso deve rimanere valido per un numero prefissato di secondi.

2.5.1 Haar matching

I metodi più semplici di template matching prevedono una semplice correlazione a livello di pixel tra un’immagine campione e la finestra di ricerca; in genere i risultati possono essere accettabili in casi molto specifici (condizioni di illuminazione costanti, stessa prospettiva sotto cui l’oggetto è inquadrato.). Diversamente in casi più generali è opportuno usare dei classificatori addestrati con un buon numero di esempi.

Con il termine classificazione si intende una procedura statistica che permette di associare ciascun oggetto (immagine, pattern, dato numerico...), appartenente a un generico spazio multidimensionale, a una o più etichette, corrispondenti alle possibili classi di cui può far parte un oggetto; si parla di classificazione esclusiva, quando ciascun oggetto può appartenere a una sola classe, o di classificazione continua o fuzzy se un oggetto può appartenere, con un certo grado di probabilità, a più classi.

Per ottenere una classificazione si usano le informazioni riguardanti alcuni tratti salienti (feature, caratteristiche) degli oggetti in esame, e li si confronta, in un apposito spazio multidimensionale, con quelli di un *training-set*: se questo *training-set è* etichettato, e quindi le possibili classi sono note e ogni campione nel set è già associato alla classe di appartenenza, si parla di apprendimento supervisionato; in caso contrario, cioè quando le possibili classi vanno direttamente ricavate dai dati stessi, si fa riferimento ad apprendimento non supervisionato.

Una formalizzazione del problema di classificazione può essere la seguente:

partendo da una serie di dati di addestramento $\{(x1, y), (x2, y), \ldots, (xn, y)\}$ si deve produrre un classificatore $h: X \to Y$ che realizzi la mappatura di nuovi elementi $x \in X$ sulle etichette $y \in Y$.

Nella realizzazione dell'algoritmo di riconoscimento di volti proposto da “Viola & Jones” [11],[12] sono tre sono le componenti che rendono questo riconoscitore uno dei più efficienti:

1. L’utilizzo di Immagini Integrali
2. La selezione delle caratteristiche importanti attraverso l’algoritmo “*Adaboost*“

3. Cascata di classificatori - Multiclassificatori

Esso è utilizza l'insieme di feature "*HaarLike*"(un insieme derivato da trasformate di Haar), per analizzare le caratteristiche delle immagini.

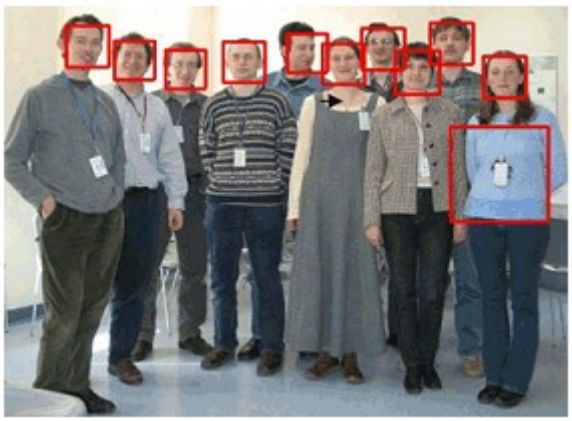

Figura 2.16: Esempio di riconoscimento di volti

L'utilizzo di tali feature permette di definire un classificatore come un albero decisionale con almeno due foglie, per determinare se una zona di un'immagine di input corrisponda a qualche immagine da lui conosciuta. La caratteristica usata in un classificatore è definita dalla sua forma e dalla posizione all'interno della regione di interesse e dalla scala.

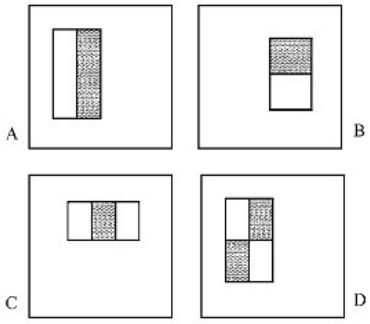

Figura 2.17: Esempio di feature a 2,3,4 rettangoli

Immagini Integrali

In un'immagine, calcolare ripetutamente somme di pixel compresi fra aree rettangolari potrebbe risultare molto costoso. Per far si che sia possibile calcolarle molto velocemente sono state introdotte le immagini integrali. Queste immagini di supporto sono calcolate molto semplicemente dalle immagini di input attraverso somme cumulative dei pixel: ogni pixel dell'immagine integrale *ii(x',y')* corrisponde alla somma dei pixel *i(x,y)* con *x* minore di *x'* e *y* minore di *y'*:

$$ii(x', y') = \sum_{x' < x,\, y' < y} i(x, y)$$

Attraverso le seguenti formule è possibile calcolare l'immagine integrale in un solo passo computazionale :

$$s(x, y) = s(x, y-1) + i(x, y) \qquad ii(x, y) = ii(x-1, y) + s(x, y)$$

Dove s(x, y) è la soma cumulative della riga.

Alla luce di queste semplici formule consideriamo l'esempio rappresentato in figura 2.18.
La somma dei pixel del punto D può essere calcolata con semplici operazioni sui valori dei punti (1, 2, 3, 4).

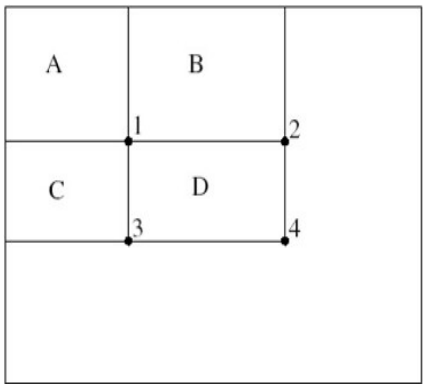

Figura 2.18: Esempio di immagine integrale

Il valore del punto 1 corrisponde alla somma cumulativa dei pixel dell'area A,; B corrisponde al valore del punto 2 sottratto a quello del punto 1; C è il valore del punto 3 dopo aver sottratto quello del punto 1.

Nello specifico i valori sono i seguenti:

- `A = 1.`
- `B = 2 - 1 .`
- `C = 3 - 1 .`
- `D = 4 - A - B - C .` → `D = 4 - 1 - (2 - 1 ) - ( 3 - 1 ) .`

L’algoritmo Adaboost

Per costruire un classificatore si possono utilizzare diversi approcci. Nel riconoscitore che si sta analizzando, l’algoritmo *Adaboost* viene utilizzato sia per analizzare l’immagine in input, sia per addestrare i classificatori base. Nella sua forma originale, l’algoritmo in questione viene usato per accelerare le performance di classificazione di un algoritmo di apprendimento. In un’immagine 24x24 vi sono più di 160.000 caratteristiche *haar-like* da poter prendere in considerazione. Ipotizzando di considerare un’immagine più' grande, il numero di caratteristiche andrebbe moltiplicato per tutte le sotto-finestre di 24x24 pixel che sono presenti nell’immagine. E’ evidente il costo proibitivo che questa procedura richiederebbe per il calcolo; la soluzione quindi è l'utilizzo delle immagini integrali. L’algoritmo A*daboost* si occupa di scegliere la caratteristica più importante fra tutte. Ad ogni ciclo analizza tutte le caratteristiche tra le più di 160.000 (presenti nell’immagine 24x24 positive e negative in tonalità di grigio) e seleziona quella che compare più volte, ovvero la caratteristica più frequente, con il miglior “epsilon”e che minimizza la differenza, e quindi l'errore, tra il vettore delle “*labels*”(apprendimento supervisionato) ed il vettore “*e*”, il cui i-esimo elemento rappresenta la classificazione dell'immagine di input (*e=1* se l'immagine *xi* è una faccia, 0 altrimenti).

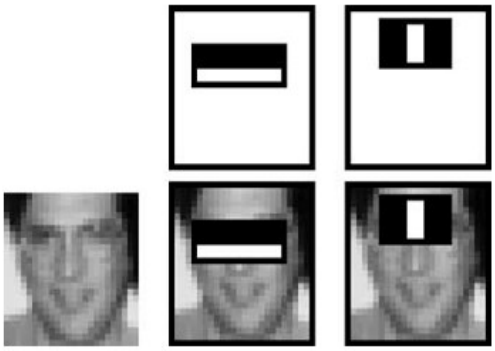

Figura 2.19: Esempio di feature utilizzate da *Adaboost*

Nella figura 2.19 si mostra l'utilizzo delle feature dell'algoritmo di *Adaboost*; la prima misura la differenza di intensità tra la regione degli occhi e la regione sottostante delle guance verificando che la zona degli occhi è spesso più scura della zona delle guance. La seconda feature confronta l'intensità degli occhi con quella della parte superiore del naso.

La Cascata di Classificatori

Ad ogni passo di "*boosting*" viene creato un classificatore base (weak). Per riconoscere la presenza di un oggetto conosciuto in una immagine arbitraria sarebbe necessario raccogliere tutti questi classificatori base e integrarli per crearne uno che sia valido per il riconoscimento effettivo dell'oggetto, utilizzando questo classificatore generale ("*strongClass*") per verificarne la presenza in tutte le possibili sotto-finestre 24x24 dell'immagine da analizzare.

Procedendo in questo modo, il costo computazionale diviene visibilmente proibitivo. L'idea di Viola e Jones e quella di utilizzare una struttura a cascata invece del classificatore generale.

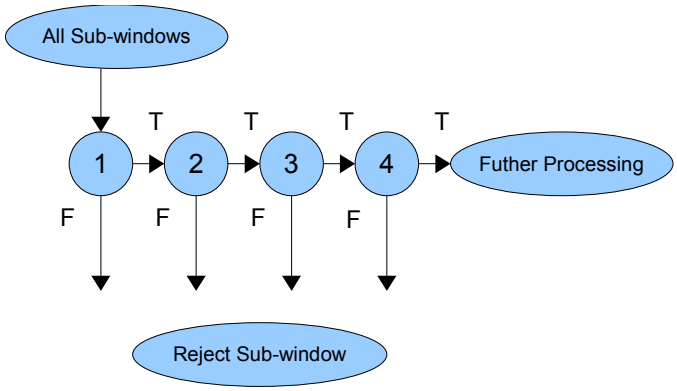

Figura 2.20: Schema di classificazione a cascata

Come schematizzato in figura 2.20 una serie di classificatori vengono applicati a tutte le sotto-finestre presenti nell'immagine. Per ognuna di queste, un risultato positivo dal primo classificatore conduce all'analisi di un secondo classificatore, che in caso di risultato positivo conduce all'analisi di un terzo classificatore e cosi' via, eliminando la sotto-finestra corrente alla prima occorrenza di un risultato negativo. In questo modo le risorse verranno consumate solo dalle aree che potrebbero condurre ad una individuazione dell'oggetto cercato, riducendo drasticamente il numero di ricerche nell'immagine.

2.5.2 CAMShift tracking

L'algoritmo chiamato *CAMShift* [13],[14] (acronimo di "Continuously Adaptive Mean Shift") è un evoluzione del suo predecessore di nome: "*MeanShift*".

MeanShift [15], è un algoritmo di *clustering* non parametrico; si basa sul calcolo di funzioni matematiche chiamate momenti, di ordine zero e uno, all'interno di una finestra di ricerca. Detta finestra viene continuamente riposizionata, e la sua dimensione è ricalcolata ogni volta in funzione dei momenti.

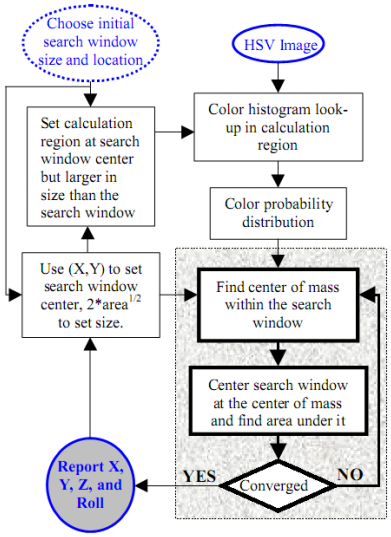

Figura 2.21: Schema a blocchi dell'algoritmo CAMShift

L'algoritmo procede nella computazione fino a che non converge, ossia finché certi valori cambiano meno di una soglia prefissata.

Nella figura 2.21 viene mostrato uno schema a blocchi del funzionamento dell'algoritmo *CAMShift* .

La differenza tra i due, è che *CAMShift* individua una opportuna finestra iniziale di ricerca, e ricalcola continuamente la distribuzione su ogni frame (da cui il suo nome), mentre il *MeanShift* ne utilizza una statica, a meno che l'oggetto dell'analisi non abbia cambiamenti significativi in termini di forma, dimensione, e ovviamente di colore.

Nell'esempio mostrato in figura 2.22, selezionando una finestra iniziale di ricerca sulla zona contenente il colore che si desidera tracciare, l'istogramma delle tinte viene aggiornato a quelle presenti nell'area di interesse per la conversione in densità di probabilità. Viene riportata un'immagine con la rispettiva versione prodotta da *CAMShift* che evidenzia le probabilità. E' presente inoltre l'istogramma di partenza delle tinte da tracciare.

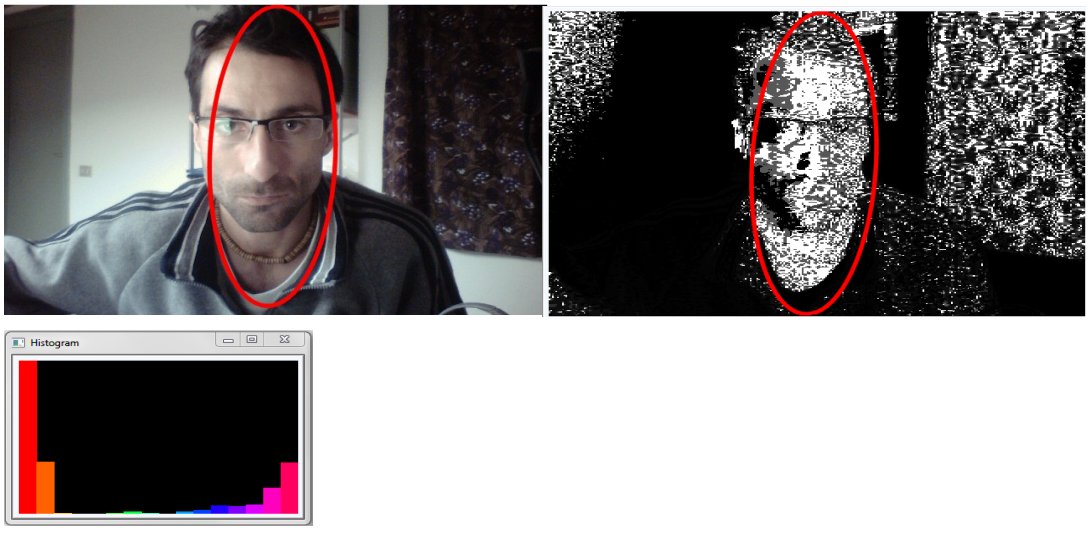

Figura 2.22Algoritmo CAMShift, probabilità e istogramma delle tinte

Su un immagine in formato Hue a 8 bit, l'intervallo dei valori della probabilità varia tra 0(nero) e 255(bianco). Apposite funzioni si occupano di computare e disegnare l'ellissoide che evidenzia l'area il cui istogramma è quello selezionato.

2.6 eSpeak text to speech

ESpeak [16] è un programma di sintesi vocale utilizzato in un ampio spettro di applicazioni, soprattutto in ambiente Linux. In ambiente Windows si interfaccia al sistema tramite le chiamate di sistema Sapi5. È disponibile presso il sito di eSpeak anche un editor che consente di modificare le regole di pronuncia del programma. Il sintetizzatore vocale supporta molte lingue tra cui l'italiano.

Una volta scelta la lingua, ci sono diverse opzioni che possono servire per migliorare la comprensibilità del testo letto.

```
espeak -v it "Ciao sono E-2? Benvenuto nel mondo dei robot."
```

Il parametro forse più importante, per rendere la voce il più simile possibile ad una voce italiana, è la velocità di lettura controllabile mediante l'opzione -s (speed):

```
espeak -s 120 -v it "Leggo questo testo lentamente"
```

L'argomento dell'opzione rappresenta la quantità di parole al minuto. Essendo il valore predefinito pari a 160, in questo esempio si ottiene una lettura leggermente più lenta.

Oltre alla velocità di lettura, può essere regolato anche il tono di voce (l'intonazione); con l'opzione -p (pitch), che attende un argomento composto da un numero che va da 0 a 99: più è grande, più acuto è il tono di voce. Il tono predefinito

corrisponde al valore 50, pertanto, l'esempio seguente sintetizza il testo con il tono più acuto possibile:

```
espeak -p 99 -v it "Voce più acuta!"
```

CAPITOLO 3
PROGETTAZIONE

In questo capitolo vengono illustrate tutte le fasi di progetto, cominciando dalla fase di analisi dei requisiti e degli obiettivi da raggiungere. Verranno quindi analizzate e discusse le scelte effettuate: il numero di componenti necessari per ottenere dei risultati soddisfacenti, le caratteristiche richieste per il software di controllo, il numero di regole e comportamenti. Per poter realizzare le funzionalità richieste è necessario tradurre i requisiti funzionali in obiettivi più semplici che supportino la fase di progettazione e sviluppo. In particolare è necessario:

- Scegliere nel dettaglio la componentistica hardware da utilizzare (come ad esempio il numero di sensori infrarossi e sonar da impiegare) per perseguire gli obiettivi proposti
- Progettare e implementare un'architettura di controllo modulare che permetta di realizzare funzionalità richieste
- Analizzare i possibili dati di ingresso e di uscita prodotti dal robot

- Testare i comportamenti del robot in simulazione ed in ambienti controllati al fine di analizzare e valutare l'efficacia delle funzionalità realizzate, identificandone pregi, difetti e i possibili miglioramenti

3.1 Rilevazione di contatto e di distanza

Per la rilevazione degli ostacoli e dei contatti con oggetti nella zona operativa del robot sono stati utilizzati dispositivi sonar, bumper e sensori ad infrarossi.

Nello specifico gli 8 sonar sono così disposti:

- 3 nella zona frontale che coprono 120° (2,3,4)
- 4 che coprono i 240° rimanenti (0,1,5,6)

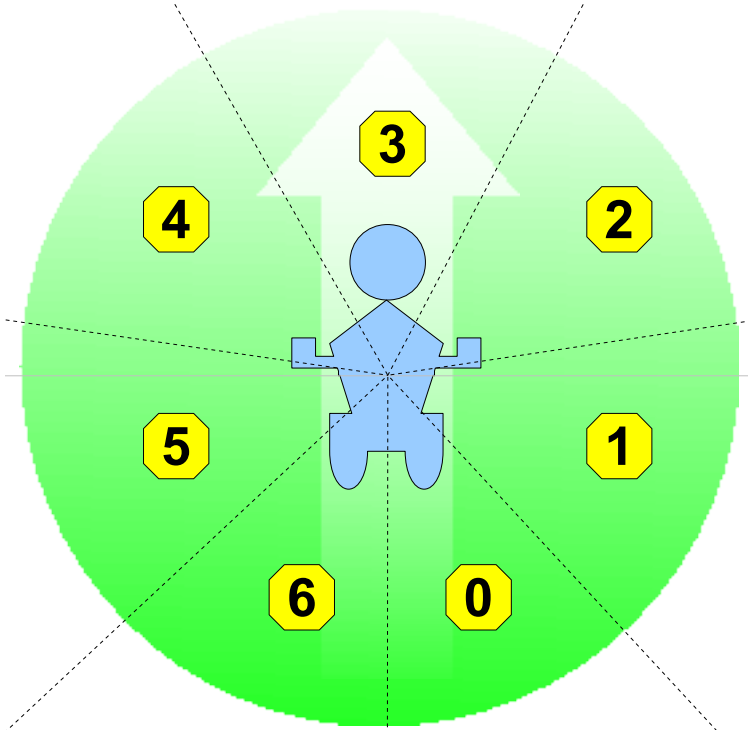

Figura 3.1: Numerazione e zone di rilevamento dei sonar

In figura 3.2 viene mostrata la numerazione e la posizione dei dispositivi ad infrarossi e dei bumper utilizzati per il rilevamento delle linee bianche sul terreno.

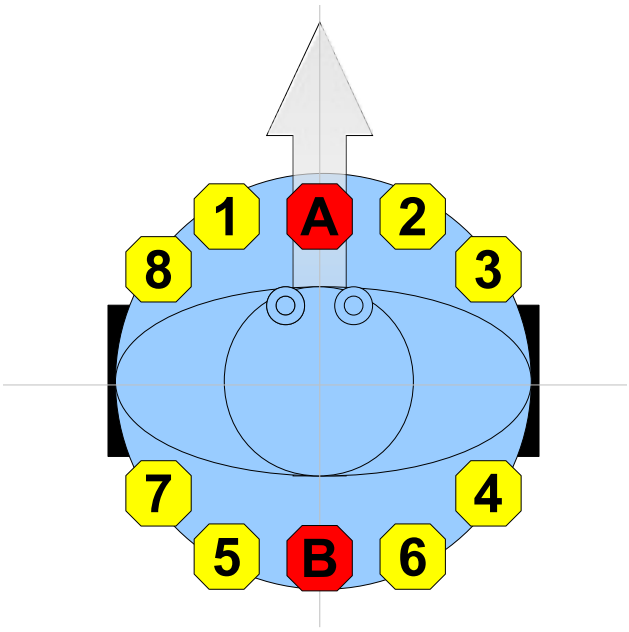

Figura 3.2: Numerazione sensori a infrarossi in giallo e bumper in rosso

3.2 Messaggi di ingresso

Di seguito vengono elencati i messaggi che Mr.Brian deve ricevere come ingresso per poter svolgere i compiti prefissati. Sono suddivisi in base al modulo di competenza che li produce:

- Visione

  Posizione della persona sul piano immagine 2D

- 2 variabili con range di valore da 0 a risoluzione massima (pixel) dell'immagine (x: 0-639, y: 0-479)

- Sonar

  Valori di prossimità identificati dai sonar

  - 8 variabili di range da 0 a 600 centimetri

- Infrarossi

  Valori delle soglie rilevati dai sensori a infrarossi per le linee bianche

  - 8 variabili con valore booleano (vero/falso)

- Bumper

  Valori segnalati dai due bumper utilizzati per le collisioni frontali e posteriori

  - 2 bumper con valore booleano (vero/falso)

- Joypad

  - 12 variabili per i bottoni con valore booleano (vero/falso).
  - 4 variabili per gli assi analogici con valori di range 0 a velocità massima.
  - 2 variabili per gli assi digitali con valore booleano (vero/falso)

3.3 Messaggi di uscita

Di seguito vengono elencati i messaggi di uscita, cioè i comandi, che invece Mr.Brian produce dopo l'elaborazione dei messaggi ricevuti. Sono suddivisi in base al modulo di competenza a cui sono diretti:

- Motor

  Movimento della base

  - Velocità di rotazione (Vr), variabile di range da -100 a +100
  - Velocità tangenziale (Vt), variabile di range da -20 a +40

- Servo

  Movimenti della testa e del collo

  - Postura preconfigurata in scheda servo, variabile di range da 1 a n

- Audio

  Frasi da sintetizzare

  - Frase predefinita, variabile di range da 1 a n

CAPITOLO 4
IMPLEMENTAZIONE

In questa sezione vengono descritti nel dettaglio i moduli (agenti) che sono stati sviluppati. Si ricorda che il meccanismo di pubblicazione/sottoscrizione dei messaggi è completamente gestito dal framework DCDT. Di seguito sono analizzati e descritti gli agenti sviluppati in codice C++ con particolare attenzione alle funzioni che ogni singolo modulo deve svolgere per garantire il corretto funzionamento del sistema. Per motivi di spazio vengono tralasciate le parti standard che compongono gli agenti e solamente le parti sviluppate nel seguente progetto.

4.1 Scambio dei messaggi tra agenti

Nella figura 4.1 viene rappresentato uno schema riassuntivo che mette in evidenza l'interazione tra agenti ed i messaggi scambiati reciprocamente dagli stessi. Come si può notare il centro del flusso è rappresentato dal modulo chiamato *"BrianExpert"* che si occupa di ricevere i messaggi da tutti gli agenti con una specifica funzione e successivamente, come descritto nel paragrafo 4.2.1, una volta elaborati i dati di ingresso di produrre i dati di uscita e di inviare i messaggi ai relativi agenti.

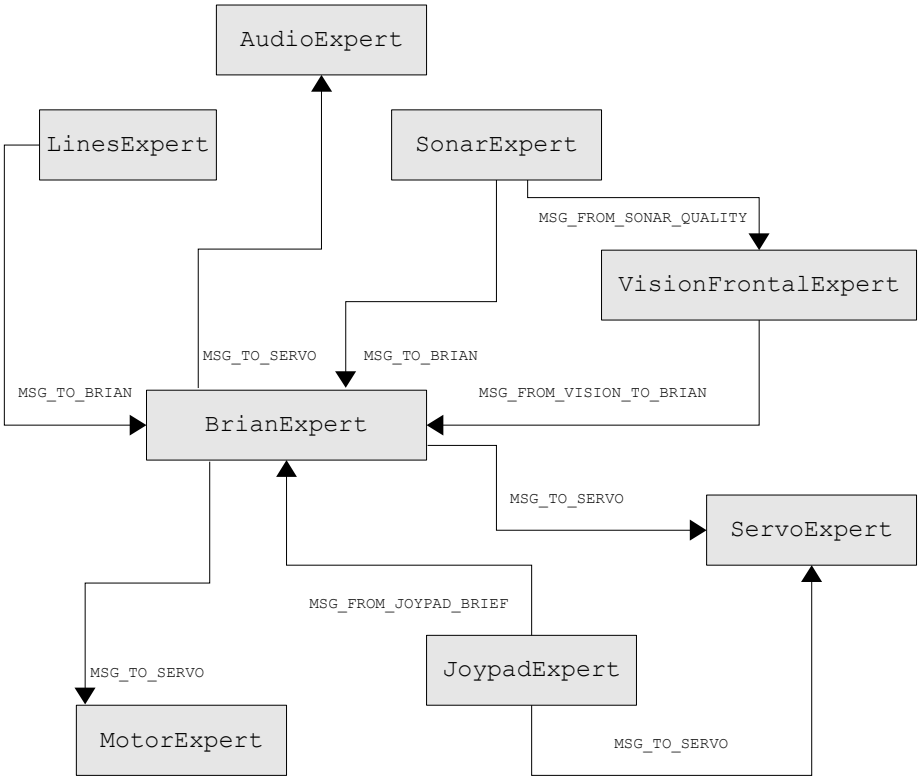

Figura 4.1: Schema a blocchi dello scambio di messaggi tra gli agenti

4.2 Esperti

Di seguito viene descritto un elenco degli agenti creati per l'interfacciamento e il controllo dell'hardware.

4.2.1 Mr.Brian – BrianExpert

Per motivi di spazio vengono tralasciate le parti standard che compongo gli agenti. Vengono quindi specificate solamente le parti sviluppate. Le parti principali di questo modulo sono l'abilitazione e la ricezioni dei messaggi da altri agenti. Il modulo BrianExpert è il componente principale di scambio dati tra le regole fuzzy ed il resto dei componenti. E' tale modulo che immagazzina tutti i dati di ingresso, effettua il ciclo descritto nel capitolo 2.3 e ne restituisce i risultati.

Vengono inizializzate le richieste di ricezione messaggi per DCDT.

```
void BrianExpert::RunInit()
  // ...
  AddMessageRequest(MSG_FROM_VISION_FRONTAL,LOCAL);
  AddMessageRequest(MSG_TO_BRIAN,LOCAL);
  AddMessageRequest(MSG_FROM_JOYPAD_BRIEF,LOCAL);
  AddMessageRequest(MSG_FROM_SONAR_QUALITY,LOCAL);
  AddMessageRequest(MSG_FROM_VISION_TO_BRIAN,LOCAL);
  // ...
}
```

Vengono elaborati i messaggi di uscita da Brian e successivamente vengono creati i messaggi per la richiesta di recapito a DCDT ad ogni esecuzione ciclica della funzione RunDuty.

```
void BrianExpert::RunDuty()
{
// ...
```

```
        ParseStringMessages();
        // Invio di messaggi al modulo motori
        SendStringMessage(MSG_TO_MOTION);
        // Creazione del messaggio al modulo servo
        NewMessage(MSG_TO_SERVO,COMMAND_DATA);
        mCurrentMessage << "<D>"
                << "lookforperson" << " " << nLookforperson << "</D>\n";
        mCurrentMessage << "<D>"
                << "interact" << " " << nInteract << "</D>\n";
        // Invio di messaggi al modulo servo
        SendStringMessage(MSG_TO_SERVO);
// ...
}
```

Mediante la funzione *ParseStringMessages* viene effettuato un parsing dei messaggi recapitati da DCDT utilizzando la funzione *brianlex_memory*.

```
void BrianExpert::ParseStringMessages()
{
  // ...
  unsigned char* buffer;

 if (ReceiveLastMessage(MSG_TO_BRIAN,buffer,false)>0){
       printf("\nBrianExpert::Ricevuto messaggio per brian\n");
    brianlex_memory((const char*)buffer,this);
 }

 if (ReceiveLastMessage(MSG_FROM_JOYPAD_BRIEF,buffer,false)>0)
    brianlex_memory((const char*)buffer,this);

 if (ReceiveLastMessage(MSG_FROM_SONAR_QUALITY,buffer,false)>0){
    brianlex_memory((const char*)buffer,this);
       printf("\nBrianExpert::Ricevuto messaggio from SONAR\n");
 }

 if (ReceiveLastMessage(MSG_FROM_VISION_TO_BRIAN,buffer,false)>0){
    brianlex_memory((const char*)buffer,this);
       printf("\nBrianExpert::MSG_FROM_VISION_TO_BRIAN\n");
 }
  // ...
  //printf("\nBrianExpert::Fine ParseStringMessages\n");
}

void BrianExpert::FillMessages(command_list * cl,
             action_list * al, predicate_list* linkcadl,
             weight_want_list* linkwwl, predicate_list* linkpdl){
  // Creazione del messaggio per il MotorExpert
  NewMessage(MSG_TO_MOTION,COMMAND_DATA);
```

```
  command_multimap::iterator c_it;
  command* c = 0;
  for (c_it = cl->begin(); c_it != cl->end(); c_it++){
      c = c_it->second;
      AddCommandData(c_it->first,c->get_set_point());
    }
  rules_line_type* pRulesLines = TheBrian->get_rules_lines();
  rules_line_type::iterator rl_it = pRulesLines->begin();

   // Creazione del messaggio per gli altri agenti
   NewMessage(MSG_FROM_BRIAN,BRIAN_DATA);
   predicate_list::iterator i;
   weight_want_list::iterator w;
   for (i = linkcadl->begin(); i != linkcadl->end(); i++){
       w = linkwwl->find(i->first);
       AddBehaviorData(i->first,i->second->get_value(),
                    w->second->get_value());
     }
    if (linkpdl != 0){
       predicate_list::iterator p;
       for (p = linkpdl->begin(); p != linkpdl->end(); p++){
         AddFuzzyData(p->first,p->second->get_value(),1.0);
       }
    }
}


  // Metodo per creare la struttura dati
void BrianExpert::BuildData(command_list* cl)
{
  // ...
  //Messaggi per i sonar
  char sName[100];
  float sonarValue;
  for (int y=0; y <8 ; y++){
         sonarValue = 0.0;
         sprintf(sName,"sonar_%d",y);
         SonarSgn = cdl->find(sName);
         if (SonarSgn != cdl->end()){
               sonarValue = SonarSgn->second->get_value();
         }
       SetValue(sName,sonarValue,1.0);
        }

  // .Variabile di ritorno da Brian per il LookForPerson
  Lookforperson = cdl->find("lookforperson");

}

  // Funzioni di supporto per la formattazione dei messaggi
void BrianExpert::AddBehaviorData(string name,
```

```
                                    float cando_value,
                                    float want_value){
  mCurrentMessage << "<D>"
                  << name << " " << cando_value << " " << want_value
                  << "</D>\n";
}
void BrianExpert::AddFuzzyData(string name,
                                   float value,
                                   float reliability){
  mCurrentMessage << "<F>"
                  << name << " " << value << " " << reliability
                  << "</F>\n";
}
void BrianExpert::AddCommandData(string name, float value){
       mCurrentMessage << "<C>"
                  << name << " " << value
                  << "</C>\n";
       printf("\n===%s--%f",name.c_str(),value);
}
```

4.2.2 Controllo manuale a distanza – JoypadExpert

E' stato sviluppato un sistema di controllo manuale per poter controllare del robot a distanza. Grazie a questo meccanismo è possibile controllare il robot per effettuare degli spostamenti, test hardware ed disabilitare la modalità automatica per prendere il controllo del robot in situazioni critiche. Ad esempio è possibile posizionare il robot all'interno dell'area delimitata dalle righe bianche senza che questo vada in conflitto con i comportamenti a cui fa riferimento (evitamento ostacoli e rimbalza indietro). Mediante la pressione di tasti del joypad precedentemente programmati è possibile inviare all'agente *AudioExpert* i comandi per fare parlare il robot.

Gli esperti che partecipano a questo contesto sono: *MotorExpert*, *BrianExpert* e *AudioExpert*.

```
/* Costruttore dell'agente */
JoypadExpert::JoypadExpert(ModuleDispatch * kernel,long period,int aID) :
  StringModuleMember(kernel,period,"joypad")
```

```
{
	int mm_lp,mm_rp,mm_dp;
	// Caricamento del file di calibrazione
	calib_file=string("./config/joypad/joypad_calibration.cal");

	// Apertura del dispositivo sulla porta js0
	device=string("/dev/input/js0");
	// ..
}

// Metodo RunDuty che viene eseguito ciclicamente
void JoypadExpert::RunDuty()
{
#ifdef EXPERTS_DEBUG
  cout << "\n Joypad::RunDuty";fflush(stdout);
#endif
	if(gestJoy->readJoypad()>=0){//>=0 è ok (evento o no), <0 è errore
		//printf("joy expert read true");
		scaleFactor=gestJoy->getScaleFactor();


	// Cambio di modalità automatica/manuale
	if(gestJoy->getToggleObstacleAvoidance()){
		op_mode = !op_mode;
#ifdef EXPERTS_DEBUG
  cout << "\n Joypad::getToggleObstacleAvoidance :op_mode=" << op_mode ;
  fflush(stdout);
#endif

	}
	// Shutdown del sistema, in caso di situazioni critiche
	if(gestJoy->getRequestStop()){
#ifdef EXPERTS_DEBUG
		printf("\nJoypad::Request stop\n");
#endif
		this->Shutdown(2);
	}
	// Invio dei messaggi a Brian
	SendMessage();
}

void JoypadExpert::SendMessage(){
	vel_joy v_joy;
	vel_cart v_cart;
	/* Creazione messaggio brief per brian, solo i dati di fwrw e rxlx*/
	NewMessage(MSG_FROM_JOYPAD_BRIEF,JOYPAD_DATA_BRIEF);
	mCurrentMessage.precision(3);
	mCurrentMessage<<fixed;
	switch(gestJoy->getCanaleInUso()){
		case GestioneJoypad::joy_analogic_left:
			m_mean_lp_x->addData(gestJoy->getRxLx());
```

```
                m_mean_lp_y->addData(gestJoy->getFwRw());
                m_mean_dp_x->addData(0);
                m_mean_dp_y->addData(0);
                m_mean_rp_x->addData(0);
                m_mean_rp_y->addData(0);
                v_joy.y = m_mean_lp_y->calculateMean();
                v_joy.x = m_mean_lp_x->calculateMean();
                break;
        case GestioneJoypad::joy_analogic_right:
                m_mean_lp_x->addData(0);
                m_mean_lp_y->addData(0);
                m_mean_dp_x->addData(0);
                m_mean_dp_y->addData(0);
                m_mean_rp_x->addData(gestJoy->getRxLx());
                m_mean_rp_y->addData(gestJoy->getFwRw());
                v_joy.y = m_mean_rp_y->calculateMean();
                v_joy.x = m_mean_rp_x->calculateMean();
                break;
        case GestioneJoypad::joy_digital:
                m_mean_lp_x->addData(0);
                m_mean_lp_y->addData(0);
                m_mean_dp_x->addData(gestJoy->getRxLx());
                m_mean_dp_y->addData(gestJoy->getFwRw());
                m_mean_rp_x->addData(0);
                m_mean_rp_y->addData(0);
                v_joy.y = m_mean_dp_y->calculateMean();
                v_joy.x = m_mean_dp_x->calculateMean();
                break;
        default:
                v_joy.y = 0;
                v_joy.x = 0;
                fprintf(stderr,"ERROR: strange,
                                joypad channel unknow...\n");
                printf("ERROR: strange, joypad channel unknow...");
                break;
}
v_joy=deadZone->apply(v_joy);
if(joyMap==NULL)    printf("ERROR joymap=NULL mumble mumble");
v_cart = joyMap->Map(v_joy);
cout << endl << "ScaleFactor" << scaleFactor << endl;
v_cart.v=v_cart.v*scaleFactor;
v_cart.w=v_cart.w/**scaleFactor*/*0.5;
#ifdef DEBUG
        printf("### - vjoy (%f,%f), vcart (%f,%f)\n",
                        v_joy.y,v_joy.x,v_cart.v,v_cart.w);
#endif

mCurrentMessage << D_OPEN_DATUM << PAD_V
                << " " << v_cart.v  <<" 1.0 "<<D_CLOSE_DATUM ;
mCurrentMessage << D_OPEN_DATUM << PAD_W
                << " " << v_cart.w  <<" 1.0 "<< D_CLOSE_DATUM << endl;
```

```
	mCurrentMessage << D_OPEN_DATUM << OP_MODE
			<< " " << op_mode <<" 1.0 "<< D_CLOSE_DATUM << endl;

	/*
		Invio dei messaggi per i comandi
		AVANTI-INDIETRO, DESTRA-SINISTRA,AUTO-MANUAL
	*/
	SendAllStringMessages();

	/* Creazione del messaggio di comando per AudioExpert*/
	if(gestJoy->getRequestAudio() >= 0){
		NewMessage(MSG_TO_SERVO_AUDIO,COMMAND_DATA);
		mCurrentMessage << "<D>"
				<< "face"  << " " << gestJoy->getRequestAudio()
				<< "</D>\n";
		cout << endl << "AUDIO: " <<
				gestJoy->getRequestAudio() << endl;
		SendStringMessage(MSG_TO_SERVO);
	}
	// Invio del messaggio
	SendAllStringMessages();
}
```

4.2.3 Audio – AudioExpert

Mediante l'esperto di controllo *AudioExpert* vengono inviati a programma di sintesi vocale *eSpeak* i comandi con le frasi da sintetizzare rendendo così possibile la riproduzione di frasi prestabilite.

```
#include "AudioExpert.h"
// ...
extern void InterfaceSetDatalex_memory(const char* src,
		InterfaceSetData* p);
extern int MessageLineNum;
/* Costruttore dell'esperto */
AudioExpert::AudioExpert(ModuleDispatch * kernel, long period, INT32 aID) :
	StringModuleMember(kernel, period, "audio") {
	// Variabili per la ricezione dello stato da Brian
	nInteractWithPerson=0;
	nLookupForPerson=0;
}

void AudioExpert::RunInit() {
	// Abilita la ricezione dei messaggi da SERVO
	AddMessageRequest(MSG_TO_SERVO, LOCAL);
}
```

```
void AudioExpert::RunDuty() {

#ifdef EXPERTS_DEBUG
	cout << "\n Audio starts\n";cout.flush();
#endif
	printf("###Audio::RunDuty\n");
	try {

			// Ricezione dei messaggi da Brian
		int message_length = 0;
		nInteractWithPerson=0;
		nLookupForPerson=0;
		message_length =
			ReceiveLastMessage(MSG_TO_SERVO,buffer_rec,false);
		printf("\n###message_length:%d\n",message_length);
		if (message_length > 0){
			InterfaceSetDatalex_memory((const char *)
					buffer_rec,this);
			printf("###AudioExpert---%s\n",buffer_rec);
		}
		// Se è abilitato lo stato di ricerca di una persona
		if (nLookupForPerson){
				system("./config/audio/playLookup.sh" );
				printf("###AudioExpert::Looking\n");
		}else if (nInteractWithPerson){
				system("./config/audio/playInteract.sh" );
				printf("###AudioExpert::Speaking\n");
		}

	} catch (...) {
		cout << "Audio.cacca! Exception caugth: " << endl;
		cout.flush();
		exit(0);
	}
#ifdef EXPERTS_DEBUG
	cout << "\n Audio ends\n";cout.flush();
#endif
}


/* Funzione che viene chiamata automaticamente da InterfaceSetDatalex
	Vengono valorizzate le due variabili:
		nLookupForPerson e  nInteractWithPerson
*/
void AudioExpert::SetData(std::string name,float value){
	if (!name.compare(LOOK_FOR_PERSON)){
		nLookupForPerson=1;
	}else if (!name.compare(INTERACT_WITH_PERSON)){
		nInteractWithPerson=1;
```

```
	}
	printf("###AudioExpert: %s %f\n",name.c_str(),value);
}
// Funzione di supporto per aggiungere il dato formattato
void AudioExpert::AddDatum(string name, float value, float reliability) {
	mCurrentMessage << "<D>" << name << " " << value << " " <<
reliability << "</D>\n";
}
```

Di seguito viene mostrato il contenuto di esempio del file *"playInteract.sh"* chiamato utilizzando la shell di Linux mediante il comando *system(....)*.

```
#!/bin/sh
espeak -v it 'Ciao.  Sono Etuuuuuuu?'
```

4.2.4 Verifica delle linee bianche e urti – LinesExpert

Questo modulo gestisce sia il rilevamento della soglia delle righe bianche mediante i sensori ad infrarossi che i possibili urti ad oggetti mediante due bumper. Vengono letti i dati dalla scheda sonar e successivamente viene creato il messaggio da inviare a Mr.Brian. Le soglie per l'abilitazione dei sensori ad infrarossi sono salvate staticamente in un file di configurazione caricato a run-time nel costruttore dell'esperto.

```
#include "LinesExpert.h"
// ...
/* Costruttore dell'esperto */
LinesExpert::LinesExpert(ModuleDispatch * kernel, long period, INT32 aID) :
	StringModuleMember(kernel, period, "lines") {
	INT32 j;
	CHAR deviceName[1024];
	ifstream ifs;
	// Apertura della porta seriale
	ifs.open(__SERIAL_DEVICE_FILE, ifstream::in);
	if (!ifs.is_open()) {
```

```
                cout << "LinesExpert::I can't open " <<
                        __SERIAL_DEVICE_FILE << "!!!\n";
                exit(0);
        }
        ifs >> deviceName;
        cout << "LinesExpert::Lines using device: " << deviceName << endl;
        ifs.close();
        // Inzializzazione della porta seriale
        this->serialPort.InitCommunication(deviceName, 57600);
        ifs.open(__THRESHOLD_FILE, ifstream::in);

        // Caricamento del file di soglia
        if (!ifs.is_open()) {
                cout << "LinesExpert::I can't open " <<
                        __THRESHOLD_FILE << "!!!\n";
                exit(0);
        }

        for (j = 0; j < LinesExpert::__howMany && ifs.good(); j++) {
                ifs >> this->thresholds[j];
                ifs.ignore(INT32_MAX, ',');
        }

        if (j != LinesExpert::__howMany) {
                cout << "LinesExpert::There're no at least "
                                inesExpert::__howMany << " thresholds!\n";
                cout.flush();
                exit(0);
        }
        ifs.close();
}
// Funzione chiamata ciclicamente
void LinesExpert::RunDuty() {
        cout << "\n LinesExpert::RunDuty";cout.flush();
        stringstream ss;
        INT32 ret = 0;
        INT32 j = 0;
        INT32 fields[10];
        CHAR c;
        // Creazione del messaggio da inviare a Mr.Brian
        NewMessage(MSG_TO_BRIAN, COMMAND_DATA);
#ifdef EXPERTS_DEBUG
        cout << "\n LinesExpert::Lines starts";cout.flush();
#endif
        c = 0x00;
        // Valorizzazione delle variabili
        while (c != '+') { // Synch with a response
                ret = this->serialPort.read(&c, 1, -1);
                if (ret <= 0) {
                        c = 0x00;
                }
```

```
}
cout << "\n LinesExpert::Lines fine riga dopo + \n";cout.flush();
while (c != '\n') {
       ret = this->serialPort.read(&c, 1, -1);
       if (ret > 0) {
              ss << c;
       } else {
              c = 0x00;
              }
}
cout << "LinesExpert::fields:";
for (j = 0; j < LinesExpert::__howMany && ss.good(); j++) {
       ss >> fields[j];
       cout << " " << fields[j];
       ss.ignore(INT32_MAX, ',');
}
cout << endl;

if (j != LinesExpert::__howMany) {
       cout << "LinesExpert::There're no " <<
              LinesExpert::__howMany << " fields!\n";
       cout << "LinesExpert::Flushing serial stream  <<
                     "and trying to recover\n";
       cout.flush();
this->serialPort.flush(true, true);
       return;
       exit(0);
}

// Costruzione della struttura del messaggio per tutti i canali
j = 0;
AddDatum(LINE_NLX, fields[j] < this->thresholds[j] ?
                     fuzzy_TRUE : fuzzy_FALSE, 1.0f);

j++;
AddDatum(LINE_NRX, fields[j] < this->thresholds[j] ?
                     fuzzy_TRUE : fuzzy_FALSE, 1.0f);

j++;
AddDatum(LINE_ELX, fields[j] < this->thresholds[j] ?
                     fuzzy_TRUE : fuzzy_FALSE, 1.0f);

j++;
AddDatum(LINE_ERX, fields[j] < this->thresholds[j] ?
                     fuzzy_TRUE : fuzzy_FALSE, 1.0f);

j++;
AddDatum(LINE_SLX, fields[j] < this->thresholds[j] ?
                     fuzzy_TRUE : fuzzy_FALSE, 1.0f);

j++;
```

```
	AddDatum(LINE_SRX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);

	j++;
	AddDatum(LINE_WLX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);

	j++;
	AddDatum(LINE_WRX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);

	j++;
	AddDatum(BUMP_NLX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);

	AddDatum(BUMP_NRX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);

	j++;
	AddDatum(BUMP_SLX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);

	AddDatum(BUMP_SRX, fields[j] < this->thresholds[j] ?
	                    fuzzy_TRUE : fuzzy_FALSE, 1.0f);
#ifdef EXPERTS_DEBUG
	cout << "\nLinesExpert::Lines ends";cout.flush();
#endif

	// Invio del messaggio completo
	SendStringMessage(MSG_TO_BRIAN);
	cout << "\n LinesExpert::SendStringToBrian";cout.flush();
	cout << endl << "Lines=-=" ;
	for (int k=0; k<10; k++){
		cout << "[" << k << "]:" <<
	                    (fields[k] < this->thresholds[k]) << " ,";
	}
	cout << endl;
}
// Chiusura della porta seriale
void LinesExpert::RunClose() {
	this->serialPort.EndCommunication();
}
// Funzione di supporto per la formattazioen del messaggio
void LinesExpert::AddDatum(string name, float value, float reliability) {
	mCurrentMessage << "<D>" << name << " " << value << " " <<
reliability << "</D>\n";
	//cout << "\nLinesExpert::AddDatum:" << mCurrentMessage << endl;
}
```

4.2.5 Controllo dei sonar – SonarExpert

Il modulo *SonarExpert* gioca un ruolo fondamentale nel presente progetto. Esso è utilizzato non soltanto dalla funzione di evitamento ostacoli, ma anche nell'identificazione dell'obiettivo. Infatti i tre sonar montati frontalmente vengono impiegati per fornire una stima della distanza tra il robot e la persona che deve essere raggiunta per poi poter interagire con essa.

```
#include "SonarExpert.h"
// ...
/* Costruttore dell'esperto */
SonarExpert::SonarExpert(ModuleDispatch * kernel,long period,int aID) :
  StringModuleMember(kernel,period,"sonar")
{
       readSonar=0;
       device=string("/dev/ttyUSB0");
       to_meter = 0.1;
       /*
              Apertura del dispositivo caricando dinamicamente
                    da file di configurazione
       */
       ifstream OpenFile("./config/sonar/sonar_config.txt");
       char buffer[255];
       while(!OpenFile.eof()) {
              OpenFile >> buffer;
       }
       device=string(buffer);
#ifdef EXPERTS_DEBUG
  cout << "SonarExpert::Open devide:'" << buffer << "'" << endl;
#endif
       OpenFile.close();
}

void SonarExpert::RunInit()
{
       if(readSonar){
              delete readSonar;
              readSonar=NULL;
       }
       try{
              // Apertura del dispositivo
              printf("SonarExpert::Create communication object\n");
              readSonar = new ReadSonar(device,to_meter);
              readSonar->sendRun();
       }
```

```
	catch (ReadSonarDeviceException &e){
		sendMessageQualityBad();
		printf("SonarExpert:: error open device\n");
		exit(-1);
	}
}

// Funzione eseguita ciclicamente
void SonarExpert::RunDuty()
{
#ifdef EXPERTS_DEBUG
  cout << "\n Sonar starts";fflush(stdout);
#endif
	if(!readSonar){
		#ifdef EXPERTS_DEBUG
			printf("readSonar NULL\n");
		#endif
			/* In caso di malfunzionamento
				viene inviato un messaggio di ERRORE
			*/
		sendMessageQualityBad();
		exit(-1);
	}
	int meas_progress=0;
	int line_readed=0;
	do{
		if(readSonar->readData()==0){
			unsigned int n_line;
			n_line=readSonar->getLineToParseNum();
			line_readed+=n_line;
			for(unsigned int i=0;i<n_line;i++){
#ifdef EXPERTS_DEBUG
				printf("parse %d/%d lines",i,n_line);
#endif
				switch(readSonar->parseLine()){
					case ReadSonar::parse_err:
						meas_progress=0;
						break;
					case ReadSonar::parse_meas_b1:
						if(meas_progress==0)meas_progress++;
						break;
					case ReadSonar::parse_meas_b2:
						if(meas_progress==1)meas_progress++;
						break;
					case ReadSonar::parse_dbg:
						meas_progress=0;
						break;
					case ReadSonar::parse_ok:
						meas_progress=0;
						break;
					case ReadSonar::parse_response_err:
```

```
                                        meas_progress=0;
                                        break;
                                default:
                                        meas_progress=0;
                                        break;
                                }
                                if(meas_progress==2){
                                        meas_progress=0;
                                        mCurrentMessage.precision(3);
                                        mCurrentMessage<<fixed;

        // Creazione del messaggio per Mr.Brian
        NewMessage(MSG_FROM_SONAR_QUALITY,SONAR_QUALITY);
                                for (unsigned int i=0;i<8;i++){
                                        mCurrentMessage << D_OPEN_DATUM <<
                                                "sonar_" << i << " "
                                                << readSonar->getMeasure(i)
                                                 <<" 1.0 "<<D_CLOSE_DATUM ;
                                                cout << i << ":" <<
                                                readSonar->getMeasure(i) << "," ;
                                        }
                                        cout << endl ;
        // Invio del messaggio per Mr.Brian
                                        SendAllStringMessages();
        // Invio del messaggio di funzionamento corretto
                                        sendMessageQualityOk();
                                }
                        }
                }
                else{
                                /* In caso di malfunzionamento
                                        viene inviato un messaggio di ERRORE
                                */
                        sendMessageQualityBad();
#ifdef EXPERTS_DEBUG
                        printf("\nSonar Read Error\n");
#endif
                        exit(-1);
                }
        }while(line_readed<=max_line_err);
        /*
                In caso di malfunzionamento
                viene inviato un messaggio di ERRORE
        */
        sendMessageQualityBad();
}

// Chiusura del dispositivo
void SonarExpert::RunClose()
{
        if(readSonar){
```

```
            readSonar->sendStop();
            delete readSonar;
            readSonar=NULL;
      }
}
// Funzione di invio messaggio ERRORE
void SonarExpert::sendMessageQualityBad(){
      NewMessage(MSG_TO_BRIAN,SONAR_QUALITY);
      AddDatum(SONAR_QUALITY_DATA,bad_quality,1.0);
      SendAllStringMessages();
}
// Funzione di invio messaggio OK
void SonarExpert::sendMessageQualityOk(){
      NewMessage(MSG_TO_BRIAN,SONAR_QUALITY);
      AddDatum(SONAR_QUALITY_DATA,0,1.0);
      SendAllStringMessages();
}
// Funzione di supporto per la formattazione del messaggio
void SonarExpert::AddDatum(string name,
                     float value,
                     float reliability)
{
            mCurrentMessage << "<B>"
              << name << " " << value << " " << reliability
              << "</B>\n";
}
```

4.2.6 Visione – VisionFrontalExpert

Il modulo di visione utilizza inizialmente l'algoritmo "*Haar Cascade*" per identificare il viso basato sul modello chiamato "*haarcascade_frontalface_alt*", successivamente una volta localizzato, sfrutta l'algoritmo di "*CAMShif*" per tenerlo "agganciato" rendendo stabile il soggetto a cui il robot propone l'interazione. Viene caricato a run-time un istogramma con la statistica delle tinte memorizzato su file precedentemente calibrato. E' necessario ricalibrare l'istogramma ogni volta che il robot viene spostato in un edificio differente o in caso di una notevole variazione della luce. Si è preferito partire da un agente già presente chiamato *VisionFrontalExpert* in modo da poter meglio integrare il lavoro svolto all'interno del progetto *MRT*.

```
#include "VisionFrontalExpert.h"
// ...
int cycles = 0;
#ifdef FACE_DETECTOR
	extern void InterfaceSetDatalex_memory(const char* src,
InterfaceSetData* p);
	extern int MessageLineNum;
#endif
VisionFrontalExpert::VisionFrontalExpert(ModuleDispatch* kernel,
                                int period,
                                VideoAcquisition* fg,
                                const char* kcc_file,
                                const char* intrinsic_parameters,
                                const char* extrinsic_parameters,
                                unsigned char feature_blob_selection)
: StringModuleMember(kernel,period,"vision_frontal"){
// ...

  cout << "Start CaptureFrameFrontal" << endl;
// Cattura del fotogramma da camera frontale
  pImage = fgHandler->CaptureFrameFrontal();
// Rilascio della camera
  fgHandler->UnLockFrameFrontal();
  cout << "End CaptureFrameFrontal" << endl;
```

```
// Se è abilitato il face detector allora inizializzo una nuova classe
#ifdef FACE_DETECTOR
  string cascadeName("config/recvision/haarcascade_frontalface_alt.xml");
  string nestedCascadeName("");
  faceDetect = new FaceDetector(cascadeName,"");
#endif
// ...
}
// Funzione eseguita una volta in fase di inizializzazione dell'agente
void VisionFrontalExpert::RunInit(){
// ...
#ifdef FACE_DETECTOR
       // Abilito la ricezione dei messaggi dal SonarExpert
       AddMessageRequest(MSG_FROM_SONAR_QUALITY,LOCAL);
#endif
// ...
}

// Funzione eseguita ciclicamente
void VisionFrontalExpert::RunDuty(){
// ...
#ifdef FACE_DETECTOR
       // Eseguo la funzione di RICONOSCIMENTO ruotando l'immagine di 180°
       BOOL bFaceFound = faceDetect->faceDetect(pImage,1,180);
              cout << "???" << faceDetect->getRadius() << " " <<
                     faceDetect->getPoint().x<< " " <<
                      faceDetect->getPoint().y << endl;
#endif
  // Creazione ed invio dei messaggi
  NewMessage(MSG_FROM_VISION_FRONTAL,VISION_DATA_FRONTAL);
  SendVisionMessage();
  SendStringMessage(MSG_FROM_VISION_FRONTAL);

#ifdef FACE_DETECTOR
/*
       Ricezione dei messaggi da SonarExpert per la stima
       della posizione della persona nel campo visivo del robot
*/
       float fPersonAngle,fPersonDistance;
       unsigned char *buffer_rec;
       int message_length =
              ReceiveLastMessage(MSG_FROM_SONAR_QUALITY,buffer_rec,false);
       // Se è stato ricevuto un messaggio dal SONAR
       if (message_length > 0){
              //motorlex_memory((const char *)buffer,this);
              // Parsing del messaggio
              InterfaceSetDatalex_memory((const char *)buffer_rec,this);
              printf("\n###VisionFrontalExpert::---%s\n",buffer_rec);
       }

       /*
```

```
            Creazione nuovo messaggio per Mr.Brian
            contenente la posizione della faccia
      */
      NewMessage(MSG_FROM_VISION_TO_BRIAN,COMMAND_DATA);
      AddAttribute(VISION_FACE_POS_X, faceDetect->getPoint().x  ,1.0);
      AddAttribute(VISION_FACE_POS_Y, faceDetect->getPoint().y  ,1.0);
      float fMin = fSonar3 < fSonar4 ?  fSonar3 : fSonar4;
      fMin = fMin < fSonar2 ? fMin : fSonar2;
      if ( !bFaceFound ){
            fMin = 0;
      }
      cout << endl <<"=-=FaceFound: " << bFaceFound << endl;
      /*
            Aggiunta attributo con distanza tra robot e obiettivo
            a messaggio corrente
      */
      AddAttribute("PersonDistance", fMin  ,1.0);
      fMin = (((float)faceDetect->getPoint().x - 320.0f) / 320.0f) *20.0f ;
      /*
            Aggiunta attributo con angolazione tra robot e obiettivo
            a messaggio corrente
      */
      AddAttribute("PersonAngle", fMin,  1.0);
      // Invio del messaggio completo
      SendAllStringMessages();
#endif
// ...
}
```

4.3 Comportamenti

Come già descritto i comportamenti del robot vengono programmati mediante delle regole fuzzy che il modulo di Brian utilizzerà per effettuare le proprie scelte. La struttura gerarchica di tali regole viene mostrata in figura 4.2. Per chiarezza non vengono mostrati tutti i comportamenti che compongono il flusso di elaborazione.

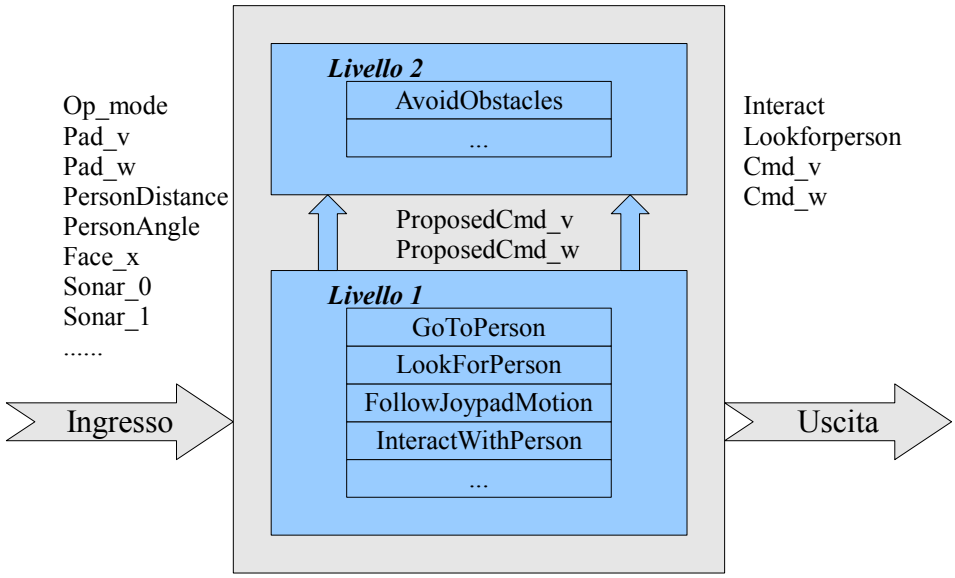

Figura 4.2: Controllore fuzzy a più livelli

Seguendo il meccanismo esposto nella sezione 2.3, il modulo di Mr.Brian verifica i comportamenti che potrà utilizzare nello specifico ciclo di elaborazione per prendere le proprie decisioni. Vengono ora descritte le regole principali sviluppate:

4.3.1 InteractWithPerson

Cando.ini

```
InteractWithPerson        =      (AND
                                       (AND   (P CanMove)(P OpModeAuto))
                                       (P PersonVeryClose)
                                 );
```

Want.txt

```
InteractWithPerson        =      (P Always);
```

InteractWithPerson.rul

```
(InteractWithPerson)      =>     (interact TRUE);
(FaceLeft)                =>     (cmd_w SLOWLX);
(FaceRight)               =>     (cmd_w SLOWRX);
```

Secondo quanto scritto nel file *Cando.ini*, la possibilità di poter eseguire l'azione è subordinata ai predicati: *CanMove*, *OpModeAuto* e *PersonVeryClose*. Quindi Mr.Brian deciderà di poter interagire con il soggetto solamente se il robot può muoversi (cioè non è in stato di blocco dovuto a un malfunzionamento del modulo dei motori), è disabilitato il controllo manuale e la persona ricercata è molto vicina. Secondo quanto presente nel file *want.txt*, la volontà di eseguire veramente l'azione è, nello stesso modo, subordinata al predicato *Always* che nello specifico risulta essere sempre valido. Questo significa che una volta abilitata la possibilità di intraprendere l'azione *InteractWithPerson* essa viene sempre eseguita. L'elenco delle variabili di uscita (azioni) che verranno intraprese sono inserite nel rispettivo file chiamato *InteractWithPerson.rul*.

La variabile di uscita interact abilità la funzione di sintesi vocale intercettata dagli agenti *AudioExpert* e *ServoExpert*, mentre i comandi *cmd_w* vengono intercettati dall'agente *MotorExpert* e hanno la funzione di posizionare il robot centrato frontalmente al suo interlocutore.

4.3.2 LookForPerson

Cando.ini

```
LookForPerson       =       (AND
                                    (AND   (P Always)
                                           (P OpModeAuto))
                                    (P PersonOutOfRange )
                            );
```

want.txt

```
LookForPerson       =       (P Always);
```

LookForPerson.rul

```
(LookForPerson)     => (lookforperson TRUE) (cmd_v STEADY) (cmd_w SLOWRX);
```

Il seguente comportamento viene abilitato solamente in caso di controllo automatico e se la persona è fuori dal campo visivo. Viene sempre scelto di eseguire il comportamento se esso è abilitato. L'esecuzione abilita una variabile di uscita di tipo booleano *lookforperson* che viene intercettata dal modulo *ServoExpert* e *AudioExpert* per produrre movimenti e frasi che attirino l'attenzione.

4.3.3 GoToPerson

Cando.ini

```
GoToPerson          =       (AND
                                    (AND
                                           (AND   (P CanMove)(P OpModeAuto))
                                           (NOT (P PersonVeryClose)))
                                    (NOT (P PersonOutOfRange))
                            );
```

want.txt

```
GoToPerson          =       (P Always);
```

GoToPerson.rul

```
(PersonVeryClose)   => (cmd_w STEADY)    (cmd_v STEADY);
(PersonOutOfRange)  => (cmd_w STEADY)    (cmd_v STEADY);
. . . .
(AND (PersonNear)(PersonN))       =>     (cmd_w STEADY)(cmd_v FASTFW);
```

Il seguente comportamento viene abilitato solamente in caso di controllo automatico, se la persona è molto vicina e non fuori dal campo visivo e se il robot ha

la possibilità di muoversi. Viene sempre scelto di eseguire il comportamento se esso è abilitato. L'esecuzione abilita due variabili di uscita *cmd_w* e *cmd_v* che vengono intercettate dal modulo *MotorExpert* per produrre i movimenti delle ruote in modo da poter raggiungere la persona identificata.

4.3.4 FollowJoypadMotion

Cando.ini

```
FollowJoypadMotion          =       (AND
                                            (AND    (P CanMove)
                                                    (NOT(P OpModeAuto)))
                                            (P JoypadMove)
                                    );
```

want.txt

```
FollowJoypadMotion          =       (P Always);
```

FollowJoypadMotion.rul

```
(FwSpeedJoyVerySlow)        => (cmd_v VSLOWFW);
(RxSpeedJoyMedium)          => (cmd_w MEDIUMRX);
```

Il seguente comportamento viene abilitato solamente in caso di controllo manuale, di ricezione di un comando di movimento dal joypad e se il robot può muoversi. Viene sempre scelto di eseguire il comportamento se esso è abilitato. L'esecuzione abilita due variabili di uscita *cmd_w* e *cmd_v* che vengono intercettate dal modulo *MotorExpert* per produrre i movimenti delle ruote in modo da poter raggiungere la persona identificata.

4.3.5 AvoidObstacles

Cando.ini

```
AvoidObstacles              =       (AND
                                            (AND (P Always) (P OpModeAuto))
                                            (NOT (P PersonVeryClose))
                                    );
```

want.txt

```
AvoidObstacles              =       (P Always);
```

AvoidObstacles.rul

```
(AND    (DirAvanti)
        (AND    (OstacoloMedioFrontale)
                (AND    (NOT    (OstacoloVicinoFrontale) )
                        (OR     (PropFwFast)
                                (PropFwVeryFast)
                        )
                )

        )
)       =>(&DEL.cmd_v ANY)  (cmd_v MEDIUMFW);
```

Il seguente comportamento viene abilitato solamente in caso di controllo automatico e se la persona non è molto vicina. Viene sempre scelto di eseguire il comportamento se esso è abilitato. Essendo un comportamento di "livello 2", esso ha la possibilità di cancellare variabili di uscita valorizzate dalle regole di "livello 1". Nello specifico in caso di ostacolo in posizione frontale e se la direzione del robot è in avanti allora cancella il valore della variabile di uscita *cmd_v* e la sostituisce con un valore più appropriato. L'esecuzione abilita due variabili di uscita *cmd_w* e *cmd_v* che vengono intercettate dal modulo *MotorExpert* per produrre i movimenti delle ruote in modo da poter evitare la collisione con eventuali ostacoli rilevati.

4.3.6 BumpStop

Cando.ini

```
BumpStop                    =       (P Always);
```

want.txt

```
BumpStop                    =       (P Always);
```

BumpStop.rul

```
(AND    (BumpN)
                (DirAvanti)
)       =>(&DEL.cmd_v ANY)(&DEL.cmd_w ANY);


(AND  (BumpN)(BumpS))
        =>(&DEL.cmd_v ANY)(&DEL.cmd_w ANY)(cmd_v STEADY)(cmd_w STEADY);
```

Il seguente comportamento viene sempre abilitato e sempre viene scelto di eseguire il comportamento se esso è abilitato. Essendo un comportamento di "livello 2", esso ha la possibilità di cancellare variabili di uscita valorizzate dalle regole di "livello 1". Nello specifico in caso di collisione sulla parte frontale del robot e se la direzione del robot è in avanti allora cancella il valore della variabile di uscita *cmd_v* e la sostituisce con un valore che significa "*stai fermo*". L'esecuzione abilita due variabili di uscita *cmd_w* e *cmd_v* che vengono intercettate dal modulo *MotorExpert* per produrre i movimenti delle ruote.

4.3.7 BounceBack

```
Cando.ini
BounceBack                  =       (AND (P Always) (P OpModeAuto));

want.txt
BounceBack                  =       (P Always);

BounceBack.rul
(LineActive)          => (&DEL.cmd_v ANY) (&DEL.cmd_w ANY);
...
(LineN)               => (cmd_v FASTRW)(cmd_w STEADY);
```

Il seguente comportamento viene abilitato solamente in caso di controllo automatico. Viene sempre scelto di eseguire il comportamento se esso è abilitato. Essendo un comportamento di "livello 2", esso ha la possibilità di cancellare variabili di uscita valorizzate dalle regole di "livello 1". Nello specifico in caso di rilevazione di una riga bianca in posizione frontale allora viene cancellato il valore della variabile di uscita *cmd_v* e viene imposto di spostarsi all'indietro. L'esecuzione abilita due variabili di uscita *cmd_w* e *cmd_v* che vengono intercettate dal modulo *MotorExpert* per produrre i movimenti delle ruote..

4.4 Simulazione e test finale del robot

MRTSim per il seguente progetto è stato uno strumento fondamentale, almeno nelle prime fasi in quando ha reso possibile l'implementazione ed i test dei comportamenti di Mr.Brian prima che il robot fosse completamente assemblato. Per rendere possibile la simulazione in MRTSim è stato necessario come prima cosa modellare tutti i componenti che rappresentano gli attori nella scena in cui il robot opera.

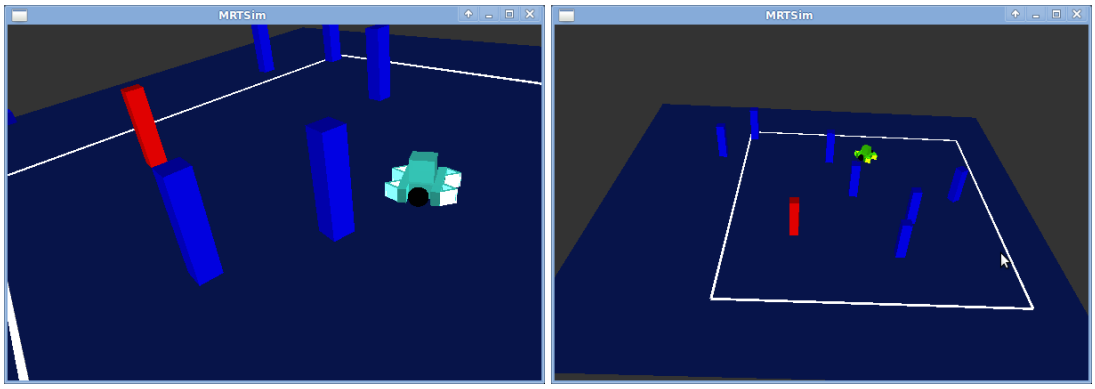

Figura 4.3: MRTSim in esecuzione su piattaforma Linux.

Mediante la gestione delle collisioni della libreria *ODE* vengono simulati i messaggi che gli agenti invierebbero a Mr.Brian in un ambiente reale.

Nelle figure 4.3 vengono mostrate due anteprime del software *MRTSim* in esecuzione. Gli oggetti di colore blu sono classificati come ostacoli, mentre quello di colore rosso viene identificato come *target*. Viene calcolata la distanza tra il robot e gli ostacoli e vengono inviati a Brian i messaggi che simulano le funzionalità di rilevamento delle distanze dei sonar.

Una volta terminato l'assemblaggio del robot è stato possibile effettuare dei test dei comportamenti sviluppati per verificare che le specifiche fossero soddisfatte. In figura 4.4 viene mostrato ciò che il modulo di visione del robot acquisisce mediante la camera e produce una volta effettuata la procedura di riconoscimento del viso. L'ellissi verde è disegnata quando una faccia viene identificata come valida dall'algoritmo *HaarCascade*; l'ellissi viola invece è disegnata quando l'algoritmo *CAMShift* restituisce una probabilità di identificazione delle tinte maggiore della soglia.

Figura 4.4: Prove di riconoscimento del viso e tracciamento della faccia.

CAPITOLO 5
CONCLUSIONI E SVILUPPI FUTURI

In questo capitolo si riassumono i risultati ottenuti, vengono descritte le problematiche riscontrate nell'attuale implementazione. Si evidenziano i punti di forza e ed i possibili sviluppi futuri per poter migliorare alcuni aspetti del progetto e successivamente viene discussa la possibilità di poter utilizzare il robot "E-2?" in domini applicativi differenti.

5.1 Valutazioni conclusive

Lo scopo del progetto è quello di sviluppare i comportamenti di un robot autonomo di attrazione per ambienti pubblici capace di evitare gli ostacoli, identificare un possibile interlocutore, interagire con esso per poter fornire un servizio di intrattenimento interattivo e accattivante.

Le regole fuzzy che controllano il funzionamento autonomo rispondono coerentemente alle specifiche iniziali. Grazie ad un elaborato sistema di

riconoscimento del viso, la funzione di ricerca e raggiungimento del target risulta stabile e robusta.

Il robot "*E-2?*" soddisfa i requisti richiesti e raggiunge l'obbiettivo prefissato. Lo sviluppo del progetto non è comunque da considerasi concluso in quanto ci sono alcuni punti critici e nuove funzionalità che sono emerse durante tutta la fase del progetto che mostrano una prospettiva di sviluppo a lungo termine. Di seguito vengono elencate alcune delle possibili implementazioni, includendo sia nuove funzionalità che dei miglioramenti ad funzioni esistenti

5.2 Problematiche riscontrate

Un aspetto critico emerso durante la fase di test finale del robot è il sistema di identificazione del target. Durante la simulazione mediante il programma *MRTSim*, in una prima versione, veniva identificato il target all'interno del gruppo di ostacoli mediante un metodo casuale. Successivamente è stato sostituito con la verifica di "delta" minimo espresso in gradi tra l'angolo della posizione del robot e quello dell'oggetto identificato come target. In questo modo si è cercato di simulare il limite del campo visivo della camera. Nella realtà, non potendo identificare con certezza la distanza dell'obbiettivo nel campo visivo, dato che le facce possono avere dimensioni diverse posizionate a distanze diverse, è stato necessario utilizzare i dati ricevuti dai sonar frontali per stimarne la posizione. Una volta acquisita tale distanza, i dati vengono combinati con le informazioni acquisite dal modulo di visione e se è stata rilevato un viso ed un ostacolo frontale a distanza di rilevamento sonar, essi vengono combinati ed inviati a *Mr.Brian*. Questo meccanismo purtroppo preclude la possibilità, una volta identificato il viso dell'obiettivo, di localizzare un ostacolo

posizionato di fronte al robot. Una possibile soluzione a tale problema verrà affrontata nella sezione successiva.

Un'ulteriore difficoltà emersa durante il progetto è stata la realizzazione di un algoritmo di identificazione dell'obiettivo che fornisse robustezza e precisione. Infatti utilizzando solamente l'algoritmo "*HaarCascade*" veniva a mancare la stabilità nell'identificazione dell'obiettivo e di conseguenza il robot continuava costantemente a saltare da una persona all'altra. Le cause principali di questo comportamento sono da attribuire ad un obbiettivo non completamente immobile, ad un ulteriore viso ritenuto "migliore" comparso nel campo visivo della camera o ad un falso positivo legato all'algoritmo. Spesso si manifesta un comportamento anomalo venendo identificato un viso anche in presenza di poster appesi al muro o oggetti che vengono classificati come facce.

Si è cercato quindi di non permettere, una volta localizzato l'obbiettivo, che possa andare perduto per una delle cause sopracitate. Si è scelto quindi di utilizzare due algoritmi differenti: uno per il rilevamento del viso ed uno per il tracciamento dell'oggetto una volta identificato come valido.

5.3 Scenario di utilizzo

L'utilizzo di robot capaci di assumere caratteristiche sempre più umane è vincolato solamente alla tecnologia necessaria per la realizzazione. Più la tecnologia evolverà, permettendo sempre una maggiore somiglianza tra le due specie, più gli scenari di utilizzo possibili aumenteranno. In questo senso possiamo affermare che se un robot "primordiale" potrà essere usato principalmente come attrazione a causa delle caratteristiche "rozze" e di novità, un androide evoluto potrà invece sostituire compiti che attualmente vengono affidati solamente agli esseri umani perché, proprio per il peculiarità di essere poco distinguibili, non sarà possibile notare facilmente la differenza e quindi verranno a mancare pregiudizi sull'utilità di queste macchine.

5.3.1 Intrattenimento

Il futuro dell'intrattenimento permetterà l'interazione con i macchine e robot simulando l'interazione tra umani. Sarà possibile giocare senza aver bisogno di alcun controller, ma usando il proprio corpo per impartire i comandi. Andranno sempre più scomparendo i controller, joypad, telecomandi a pulsanti, dispositivi di input per lasciare spazio a comandi vocali, gesti, espressioni facciali e persino alle sfumature emotive della voce. Sarà sufficiente porsi di fronte al sensore e impartire istruzioni semplicemente parlando. L' impiego più indicato è quello di utilizzare il robot "E-2?" in un qualsiasi luogo privato o pubblico per attirare l'attenzione della gente a scopo propagandistico ed informativo. Lo scopo principale quindi è quello di catalizzare l'attenzione proprio in virtù del fatto che essendo una novità riesce a scaturire interesse curiosità nei visitatori. In futuro quando la robotica diventerà di uso più comune robot simili potranno essere usati in altri luoghi come ad esempio: in locali di ristorazione, in punti di informazione nella funzione di "*receptionist*" magari coadiuvati da un controllo audio remoto.

Al momento il robot ha partecipato alla fiera della robotica 2009 - “ExhiBot - A robot for exhibitions” - aprendo la manifestazione e dando il benvenuto al sindaco di Milano, Letizia Moratti.

Figura 5.1: ExhiBot - Fiera della Robotica 2009

5.3.2 Domotica

La domotica è la scienza interdisciplinare che si occupa dello studio delle tecnologie atte a migliorare la qualità della vita nella casa e più in generale negli ambienti antropizzati. Il termine domotica deriva dal greco "*domos"*che significa "*casa"*. La domotica svolge un ruolo importante nel rendere intelligenti apparecchiature, impianti e sistemi. Nonostante alcuni compiti specializzati siano già ora affidati ad apparecchi automatici come "*Roomba"*, per la pulizia dei pavimenti, o "*Robomow"* per la manutenzione del prato, possiamo ipotizzare che "E-2?" verrà in futuro utilizzato nell'ambiente domestico per compiti per i quali è preferibile avere una maggiore sembianza umana. Come Asimov amava immaginare nei suoi racconti, l'utilizzo di robot umanoidi potrebbe essere applicato ai compiti propri di un maggiordomo, e quindi la manutenzione della casa, il ricevimento di ospiti ma anche il controllo della struttura.

5.4 Sviluppi futuri

Una possibile modifica al modulo di identificazione dell'obiettivo, sarebbe riuscire ad rendere indipendente tale funzione dal modulo di rilevamento dei sonar. Per fare ciò si potrebbe progettare un modulo di visione basato su una camera stereoscopica, oppure con più camere distinte e successivamente combinando tutte le immagini. In questo modo si potrebbe avere una più precisa localizzazione del viso. Un'ulteriore miglioria sarebbe la possibilità di registrare uno storico di persone già approcciate tenendo un riferimento a chi è già stato a contatto con il robot dando la possibilità di differenziare le frasi di benvenuto. Si potrebbe inoltre migliorare la procedura di evitamento ostacoli implementando una funzione di “Person Detector”. Con questa nuova caratteristica si validerebbe l'obiettivo solamente in caso di identificazione di un' intera persona, ad esempio cercando altre parti del corpo come busto, braccia, mani, dando quindi la possibilità di ignorare tutti gli altri oggetti che verrebbero classificati come ostacoli. Un ulteriore funzionalità semplice che renderebbe molto più accattivante il momento di scambio di informazioni tra robot e utente sarebbe la capacità del robot di poter modificare l'inclinazione della propria testa seguendo l'asse della faccia dell'obbiettivo.

Per un robot interattivo è tipico possedere sensori acustici uniti a sistemi per il riconoscimento della voce che rendano possibile il riconoscimento e l'esecuzione di comandi vocali. Con questa nuova caratteristica l'interazione tra il robot ed il suo interlocutore verrebbe notevolmente migliorata, dando la possibilità di poter rispondere a frasi più o meno definite con comportamenti prestabiliti.

Una possibile proposta sarebbe la realizzazione di un meccanismo di localizzazione che renda disponibile una mappa in tempo reale dell'ambiente e della porzione in cui il robot opera. Con questo scenario sarebbe possibile migliorare il movimento autonomo del robot permettendogli di memorizzare obiettivi già raggiunti, percorsi favorevoli e zone da evitare; aggiungendo poi una funzione di "ritorno alla base" attivabile, sia automaticamente che mediante il comando a distanza, si potrebbe riportare il robot in una zona determinata, come ad esempio dietro un banco di benvenuto. Tale funzione si potrebbe realizzare applicando un marker (come per esempio il logo del Politecnico di Milano) sul soffitto, su un muro o in una specifica posizione.

Bibliografia

1. Paolo Meriggi
   Processing and Communication Systems for Device Communities
   Università degli studi di Brescia, 2005

2. Modular Robot Toolkit
   http://airwiki.elet.polimi.it/mediawiki/index.php/MRT

3. LURCH - The autonomous wheelchair
   http://airwiki.elet.polimi.it/mediawiki/index.php/LURCH_-_The_autonomous_wheelchair

4. Milan Robocup Team
   http://robocup.ws.dei.polimi.it/MRT/

5. A. Bonarini and M. Matteucci and M. Restelli. A novel model to rule behavior interaction. In *Proceedings of the 8th Conference on Intelligent Autonomous Systems* (IAS-8), page 199 206, Amsterdam, 2004. IOS Press

6. L. A. Zadeh. Fuzzy sets. *Information and Control* ,8:338–353,1965.

7. Qt – Cross-platform application and UI framework
   http://qt.nokia.com/

8. Open Dynamics Engine Library
   http://ode.org

9. Open Computer Vision Library
http://sourceforge.net/projects/opencvlibrary/

10. Intel Integrated Performance Primitives Cryptography Library
http://software.intel.com/en-us/intel-ipp/

11. Viola, Jones: *Robust Real-time Object Detection*, IJCV 2001
http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_IJCV.pdf

12. Viola, Jones: *Rapid object detection using a boosted cascade of simple features*
http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_CVPR2001.pdf

13. G.R. Bradski, *Computer video face tracking for use in a perceptual user interface*, Intel Technology Journal, Q2 1998.

14. 1.M.J. Swain and D.H. Ballard, *Color Indexing, International Journal of Computer Vision*, vol. 7:1, 1991.

15. D. Comaniciu, V. Ramesh, and P. Meer, *Real-Time Tracking of Non-Rigid Objects using Mean Shift*, 2000

16. eSpeak text to speech
http://espeak.sourceforge.net/

Appendice

In questa sezione viene mostrato il codice completo dei comportamenti utilizzati la configurazione ed esecuzione del modulo Mr.Brian.

Comportamenti (behaviour.txt)

```
( level 1 GoToPerson      ./config/brian/rules/GoToPerson.rul )
( level 1 LookForPerson   ./config/brian/rules/LookForPerson.rul )
( level 1 FollowJoypadMotion
                          ./config/brian/rules/FollowJoypadMotion.rul )
( level 1 InteractWithPerson
                          ./config/brian/rules/InteractWithPerson.rul )
( level 1 HeadPosition    ./config/brian/rules/HeadPosition.rul )
( level 1 StayBlocked     ./config/brian/rules/StayBlocked.rul )

( level 2 AvoidObstacles  ./config/brian/rules/AvoidObstacles .rul )
( level 2 BounceBack      ./config/brian/rules/BounceBack.rul     )
( level 2 BumpStop        ./config/brian/rules/BumpStop .rul      )
( level 2 ForceBlocked    ./config/brian/rules/ForceBlocked.rul )
```

Fuzzyficazione (ctof.txt)

```
#######################################
## Generiche                         ##
#######################################
(always            BOOLEANFLAG)
```

```
(op_mode            OP_MODE)
(pad_v              V_SPEED)
(pad_w              W_SPEED)
(sonar_quality      SONAR_QUALITY)
(emergency_val      EMERGENCY_VALUE)
(batt_val           BATTERY_LEVEL)
(can_move           BOOLEANFLAG)
(prev_cmd_v         V_SPEED)
(prev_cmd_w         W_SPEED)
(enable_obstacle_avoidance BOOLEANFLAG)

########################################
## Variabili del livello precedente   ##
########################################
(Proposedcmd_v      V_SPEED)
(Proposedcmd_w      W_SPEED)


########################################
## Informazioni su target / robot     ##
########################################
(PersonDistance     DISTANCE )
(PersonAngle        ANGLE )

(RobotX             X_COORDINATES)
(RobotY             Y_COORDINATES)

(PersonX            X_COORDINATES)
(PersonY            Y_COORDINATES)

(HeadX              HEAD_X_COORDINATES)
(HeadY              HEAD_Y_COORDINATES)


########################################
## Sonar                              ##
########################################
(sonar_0            SONAR_DISTANCE)
(sonar_1            SONAR_DISTANCE)
(sonar_2            SONAR_DISTANCE)
(sonar_3            SONAR_DISTANCE)
(sonar_4            SONAR_DISTANCE)
(sonar_5            SONAR_DISTANCE)
(sonar_6            SONAR_DISTANCE)
(sonar_7            SONAR_DISTANCE)
(sonar_8            SONAR_DISTANCE)
(sonar_9            SONAR_DISTANCE)
```

```
########################################
## Linee Bianche                      ##
########################################
(lineNRX            BOOLEANFLAG)
(lineNLX            BOOLEANFLAG)
(lineSRX            BOOLEANFLAG)
(lineSLX            BOOLEANFLAG)
(lineWRX            BOOLEANFLAG)
(lineWLX            BOOLEANFLAG)
(lineERX            BOOLEANFLAG)
(lineELX            BOOLEANFLAG)


########################################
## Bump                               ##
########################################
(bumpNRX            BOOLEANFLAG)
(bumpNLX            BOOLEANFLAG)
(bumpSRX            BOOLEANFLAG)
(bumpSLX            BOOLEANFLAG)


########################################
## Posizione faccia su asse X         ##
########################################
(FacePosX           FACE_X_POSITION)
```

Insiemi di fuzzyficazione (shape_ctof.txt)

```
(BOOLEANFLAG
   (SNG (FALSE 0) )
   (SNG (TRUE  1 ) )
)

(SONAR_QUALITY
 (SNG (GOOD 0) )
 (SNG (BAD 1) )
)

(EMERGENCY_VALUE
   (SNG (NO_EMERGENCY 0) )
   (SNG (SW_EMERGENCY  1 ) )
   (SNG (RADIO_EMERGENCY 2) )
   (SNG (BUTTON_EMERGENCY  3 ) )
)

(BATTERY_LEVEL
   (TOL (BATT_TOO_LOW 18 20) )
```

```
   (TRA (BATT_LOW  18 20 21 23) )
   (TRA (BATT_OK 21 23 24 25) )
   (TRA (BATT_HIGH  24 25 27 28 ) )
   (TOR (BATT_TOO_HIGH  27 28 ) )
)

(OP_MODE
   (SNG (MANUAL 0) )
   (SNG (AUTOMATIC  1 ) )
   (SNG (SAFE  2 ) )
)

(V_SPEED
 (TOL (VFASTRW -1.2 -0.64))
 (TRI (FASTRW -1.2 -0.64 -0.32))
 (TRI (MEDIUMRW -0.64 -0.32 -0.16))
 (TRI (SLOWRW -0.32 -0.16 -0.08 ))
 (TRI (VSLOWRW -0.16 -0.08 0.0))
 (TRI (STEADY -0.08 -0.0 0.1))
 (TRI (VSLOWFW 0.0 0.1 0.2))
 (TRI (SLOWFW 0.1 0.2 0.4))
 (TRI (MEDIUMFW 0.2 0.4 0.8))
 (TRI (FASTFW  0.4 0.8 2))
 (TOR (VFASTFW 0.8 2))
)

(W_SPEED
 (TOL (VFASTRX -2 -0.8))
 (TRI (FASTRX -2 -0.8 -0.4 ))
 (TRI (MEDIUMRX -0.8 -0.4 -0.2))
 (TRI (SLOWRX -0.4 -0.2 -0.1))
 (TRI (VSLOWRX -0.2 -0.1 0))
 (TRI (STEADY -0.1 -0.0 0.1))
 (TRI (VSLOWLX 0 0.1  0.2))
 (TRI (SLOWLX 0.1 0.2 0.4))
 (TRI (MEDIUMLX 0.2 0.4 0.8))
 (TRI (FASTLX 0.4 0.8 2))
 (TOR (VFASTLX 0.8 2))
)

(DISTANCE
(SNG (OUT_OF_RANGE 0))
(TRA (IN_CONTACT 1 9 40 40))
(TRA (VERY_CLOSE 1 1 70 110))
(TRA (CLOSE 70 110 130 190))
(TRI (NEAR 130 190 250))
(TOR (FAR 190 250))
(TOR (VERY_FAR 250 650))
)
```

```
(ANGLE
(TRA (N1 0 0 11.25 33.75))
(TRA (NW 11.25 33.75 56.25 78.75))
(TRA (WNW 56.25 78.75 90 90))
(TRA (WSW 90.001 90.001 101.25 123.75))
(TRA (SW 101.25 123.75 146.25 168.75))
(TRA (S 146.25 168.75 191.25 213.75))
(TRA (SE 191.25 213.75 236.25 258.75))
(TRA (ESE 236.25 258.75 270 270))
(TRA (ENE 270.001 270.001 281.25 303.75))
(TRA (NE 281.25 303.75 326.25 348.75))
(TRA (N2 326.25 348.75 360 360))

(TRA (ALLN1 0 0 90 90))
(TRA (ALLN2 270.1 270.1 360 360))
(TRA (ALLS 90.1 90.1 270 270))

(TRA (K_N1 0 0 10 20))
(TRA (K_N2 340 350 360 360))
(TRA (K_S 160 170 190 200))
(TRA (OUT_W 10 20 160 170))
(TRA (OUT_E 190 200 340 350))
)

(ALIGN
(TRA (CENTERED1 0 0 10 20))
(TRA (VERY_SMALL_LEFT 5 10 20 25))
(TRA (SMALL_LEFT 10 20 40 90))
(TRA (BIG_LEFT 40 90 150 170))
(TRA (OPPOSITE 150 170 190 210))
(TRA (BIG_RIGHT 190 210 270 320))
(TRA (SMALL_RIGHT 270 320 340 350))
(TRA (VERY_SMALL_RIGHT 335 340 350 355))
(TRA (CENTERED2 340 350 360 360))

(TRA (CENTERED_FAR1 0 0 10 10))
(TRA (CENTERED_FAR2 350 350 360 360))
(TRA (CENTERED_NEAR1 0 0 25 25))
(TRA (CENTERED_NEAR2 335 335 360 360))
(TRA (CENTERED_CLOSE1 0 0 45 45))
(TRA (CENTERED_CLOSE2 315 315 360 360))
)


(PERSON_DELTA_ANGLE
(TOL (DELTA_NEG -4 0 ))
(TRA (DELTA_STEADY -4 0 0 4))
(TOR (DELTA_POS 0 4))
```

```
)

(PERSON_DELTA_DISTANCE
(TOL (APPROACHING -16 -4))
(TRA (KEEPING_DISTANCE -16 -4 4 16))
(TOR (MOVING_AWAY 4 16))
(TOL (APPROACHING_ANGLE -1 -1))
(TOR (MOVING_AWAY_ANGLE 1 1))
)

(AREA_TIME
(TRA (STAY 0 0 7.9 7.9))
(TRA (TRYKICK 8 8 8.9 8.9))
(TOR (LEAVE 9 9))
)

(X_COORDINATES
# 12m
(TRA (OPPONENT_GOAL_AREA_X 530 550 600 600))
(TRA (NEAR_OPPONENT_GOAL_AREA_X 530 530 550 550))
(TRA (OPPONENT_PENALTY_AREA_X 430 450 600 600))
(TRA (NEAR_OPPONENT_PENALTY_AREA_X 400 400 450 450))
(TRA (OWN_GOAL_AREA_X -600 -600 -550 -530))
(TRA (NEAR_OWN_GOAL_AREA_X -550 -550 -500 -500))
(TRA (OWN_PENALTY_AREA_X -600 -600 -450 -430))
(TRA (NEAR_OWN_PENALTY_AREA_X -450 -450 -400 -400))
(TOR (IN_OPPONENT_FIELD -25 25))
(TRA (IN_MIDDLE_FIELD -50 -25 25 50))
(TOL (IN_OWN_FIELD -25 -25))
(TOR (LINE_N_OUT 600 650))
(TOL (LINE_S_OUT -650 -600))
(TRA (LINE_N_CLOSE 500 550 600 650))
(TRA (LINE_S_CLOSE -650 -600 -550 -500))
)

(Y_COORDINATES
# 8m
(TRA (GOAL_AREA_Y -170 -150 150 170))
(TRA (NEAR_GOAL_AREA_Y -200 -200 200 200))
(TRA (PENALTY_AREA_Y -250 -250 250 250))
(TRA (NEAR_PENALTY_AREA_Y -300 -300 300 300))
(TOR (LINE_W_OUT 400 450))
(TOL (LINE_E_OUT -450 -400))
(TRA (LINE_W_CLOSE 300 350 400 450))
(TRA (LINE_E_CLOSE -450 -400 -350 -300))
)

(HEAD_X_COORDINATES
(TRA (HEAD_LEFT -1 -0.75 -0.25 0 ))
```

```
(TRA (HEAD_CENTER -0.5 0.25 0.25 0.5))
(TRA (HEAD_RIGHT 0 0.25 0.75 1 ))
)

(HEAD_Y_COORDINATES
(TRA (HEAD_TOP -1 -0.75 -0.25 0 ))
(TRA (HEAD_MIDDLE -0.5 -0.25 0.25 0.5))
(TRA (HEAD_DOWN 0 0.25 0.75 1 ))
)

(FACE_X_POSITION
(TOL (FACE_LEFT 95 190))
(TRA (FACE_CENTER 95 190 450 545))
(TOR (FACE_RIGHT 450 545 ))
)

(SONAR_DISTANCE
 (TOL (VICINO 0 80 ))
 (TRA (MEDIO 20 60 80 100))
 (TRA (LONTANO 80 100 450 500))
 (TOR (MOLTOLONTANO 450 600))
)
```

Attivazione dei componenti (Cando.INI)

```
############################
##  StayBlocked          ##
############################
StayBlocked         =      (OR    (OR    (NOT   (P CanMove))
                                             (NOT (P OpModeAuto))
                                             0)
                                   (AND   (P CanMove)
                                          (AND          (NOT (P OpModeAuto))
                                                        (P JoypadHalt)
                                                 )
                                          )
                              );


############################
##  GoToPerson           ##
############################
GoToPerson          =      (AND
                                   (AND
                                   (AND   (P CanMove)(P OpModeAuto))
                                          (NOT (P PersonVeryClose)))
                                          (NOT (P PersonOutOfRange))
                              );
```

```
##########################
##  InteractWithPerson   ##
##########################
InteractWithPerson  =      (AND
                           (AND   (P CanMove)(P OpModeAuto))
                           (P PersonVeryClose)
                           );


##########################
##  FollowJoypadMotion   ##
##########################
FollowJoypadMotion  =      (AND
                                  (AND (P CanMove)
                                        (NOT(P OpModeAuto)))
                                  (P JoypadMove)
                           );


##########################
##  HeadPosition         ##
##########################
HeadPosition        =      (AND
                                  (AND   (P CanMove)(P OpModeAuto))
                                  (P PersonVeryClose)
                           );


##########################
##  AvoidObstacles       ##
##########################
AvoidObstacles      =      (AND
                                  (AND (P Always) (P OpModeAuto))
                                  (NOT (P PersonVeryClose))
                           );


##########################
##  BounceBack           ##
##########################
BounceBack          =      (AND (P Always)
                                 (P OpModeAuto)
                           );
```

```
############################
##  BumpStop              ##
############################
BumpStop            =      (P Always);

#Versione valida solo per il controllo automatico
#BumpStop           =      (AND (P Always) (P OpModeAuto));


############################
##  LookForPerson         ##
############################
LookForPerson       =      (AND
                                (AND (P Always)
                                  (P OpModeAuto)
                                )
                                (P PersonOutOfRange )
                           );
```

Scelta dei componenti (want.ini)

```
AvoidObstacles            =      (P Always);
BounceBack                =      (P Always);
BumpStop                  =      (P Always);
FollowJoypadMotion        =      (P Always);
GoToPerson                =      (P Always);
HeadPosition              =      (P Always);
InteractWithPerson        =      (P Always);
LookForPerson             =      (P Always);
StayBlocked               =      (P Always);
```

Defuzzyficazione

```
( interact BOOLEANFLAG )

( lookforperson BOOLEANFLAG )

( blocked BOOLEANFLAG )

( cmd_v V_SPEED )
( cmd_w W_SPEED )

( cmd_head_delta_x HEAD_DELTA_X )
( cmd_head_delta_y HEAD_DELTA_Y )
```

```
( cmd_head_delta_z HEAD_DELTA_Z )
```

Insiemi di defuzzyficazione (s_shape.txt)

```
(V_SPEED
 (SNG  (VFASTRW     -20))
 (SNG  (FASTRW      -15))
 (SNG  (MEDIUMRW    -10))
 (SNG  (SLOWRW      -8))
 (SNG  (VSLOWRW     -5))
 (SNG  (STEADY      0))
 (SNG  (VSLOWFW     5))
 (SNG  (SLOWFW      10))
 (SNG  (MEDIUMFW    20))
 (SNG  (FASTFW      25))
 (SNG  (VFASTFW     40))
)

(W_SPEED
 (SNG  (VFASTRX           -100))
 (SNG  (FASTRX            -40))
 (SNG  (MEDIUMRX          -20))
 (SNG  (SLOWRX            -10))
 (SNG  (VSLOWRX           -5))
 (SNG  (STEADY            0))
 (SNG  (VSLOWLX           5))
 (SNG  (SLOWLX            10))
 (SNG  (MEDIUMLX          20))
 (SNG  (FASTLX            40))
 (SNG  (VFASTLX           100))
)

(BOOLEANFLAG
(SNG (TRUE 1))
(SNG (FALSE 0))
)


################################
##   Movimenti della testa    ##
################################
(HEAD_DELTA_X
(SNG (LEFT -1))
(SNG (CENTER -1))
(SNG (RIGHT 1))
)

(HEAD_DELTA_Y
(SNG (NORTH -1))
```

```
(SNG (MIDDLE 1))
(SNG (SUD 1))
)

(HEAD_DELTA_Z
(SNG (UP -1))
(SNG (MIDDLE 0))
(SNG (DOWN 1))
)
```

Predicati (Predicate.ini)

```
Always                      = (D always TRUE);
Never                       = (D always FALSE);

CanMove                     = (D can_move TRUE);
CantMove                    = (D can_move FALSE);

OpModeAuto                  = (D op_mode AUTOMATIC);
ObstacleAvoidanceRun        = (D enable_obstacle_avoidance TRUE);


##################################
##   predicati joypad FW/RW     ##
##################################
FwRwSpeedJoySteady          = (D pad_v STEADY);
FwSpeedJoyVerySlow          = (D pad_v VSLOWFW);
FwSpeedJoySlow              = (D pad_v SLOWFW);
FwSpeedJoyMedium            = (D pad_v MEDIUMFW);
FwSpeedJoyFast              = (D pad_v FASTFW);
FwSpeedJoyVeryFast          = (D pad_v VFASTFW);
RwSpeedJoyVerySlow          = (D pad_v VSLOWRW);
RwSpeedJoySlow              = (D pad_v SLOWRW);
RwSpeedJoyMedium            = (D pad_v MEDIUMRW);
RwSpeedJoyFast              = (D pad_v FASTRW);
RwSpeedJoyVeryFast          = (D pad_v VFASTRW);


##################################
##   predicati joypad LX/RX     ##
##################################
RxLxSpeedJoySteady          = (D pad_w STEADY);
RxSpeedJoyVerySlow          = (D pad_w VSLOWRX);
RxSpeedJoySlow              = (D pad_w SLOWRX);
RxSpeedJoyMedium            = (D pad_w MEDIUMRX);
RxSpeedJoyFast              = (D pad_w FASTRX);
RxSpeedJoyVeryFast          = (D pad_w VFASTRX);
```

```
LxSpeedJoyVerySlow        = (D pad_w VSLOWLX);
LxSpeedJoySlow            = (D pad_w SLOWLX);
LxSpeedJoyMedium          = (D pad_w MEDIUMLX);
LxSpeedJoyFast            = (D pad_w FASTLX);
LxSpeedJoyVeryFast        = (D pad_w VFASTLX);


################################
##   predicati joypad ON/OFF   ##
################################
JoypadHalt                = (AND (P FwRwSpeedJoySteady)
                                 (P RxLxSpeedJoySteady)
                           );
JoypadMove                = (NOT (P JoypadHalt));


################################
##   Sonar                    ##
################################
VicinoNord                = (D sonar_3 VICINO);
MedioNord                 = (D sonar_3 MEDIO);
LontanoNord               = (D sonar_3 LONTANO);
MoltoLontanoNord          = (D sonar_3 MOLTOLONTANO);

################## DISABILITATO IL SONAR 7 ############################
VicinoNordRight                 = (D sonar_7 VICINO);
MedioNordRight                  = (D sonar_7 MEDIO);
LontanoNordRight                = (D sonar_7 LONTANO);
MoltoLontanoNordRight           = (D sonar_7 MOLTOLONTANO);
VicinoNordLeft                  = (D sonar_7 VICINO);
MedioNordLeft                   = (D sonar_7 MEDIO);
LontanoNordLeft                 = (D sonar_7 LONTANO);
MoltoLontanoNordLeft            = (D sonar_7 MOLTOLONTANO);
################## DISABILITATO IL SONAR 7 ############################

VicinoNordEst                   = (D sonar_2 VICINO);
MedioNordEst                    = (D sonar_2 MEDIO);
LontanoNordEst                  = (D sonar_2 LONTANO);
MoltoLontanoNordEst             = (D sonar_2 MOLTOLONTANO);

VicinoEst                       = (D sonar_1 VICINO);
MedioEst                        = (D sonar_1 MEDIO);
LontanoEst                      = (D sonar_1 LONTANO);
MoltoLontanoEst                 = (D sonar_1 MOLTOLONTANO);


VicinoSudEst                    = (D sonar_0 VICINO);
MedioSudEst                     = (D sonar_0 MEDIO);
```

```
LontanoSudEst                    = (D sonar_0 LONTANO);
MoltoLontanoSudEst               = (D sonar_0 MOLTOLONTANO);

VicinoSudOvest                   = (D sonar_6 VICINO);
MedioSudOvest                    = (D sonar_6 MEDIO);
LontanoSudOvest                  = (D sonar_6 LONTANO);
MoltoLontanoSudOvest             = (D sonar_6 MOLTOLONTANO);

VicinoSud                        = (OR (P VicinoSudOvest)(P VicinoSudEst));
MedioSud                         = (OR (P MedioSudOvest)(P MedioSudEst));
LontanoSud                       = (OR (P LontanoSudOvest)
                                       (P LontanoSudEst));
MoltoLontanoSud                  = (OR (P MoltoLontanoSudOvest)
                                       (P MoltoLontanoSudEst));

VicinoOvest                      = (D sonar_5 VICINO);
MedioOvest                       = (D sonar_5 MEDIO);
LontanoOvest                     = (D sonar_5 LONTANO);
MoltoLontanoOvest                = (D sonar_5 MOLTOLONTANO);

VicinoNordOvest                  = (D sonar_4 VICINO);
MedioNordOvest                   = (D sonar_4 MEDIO);
LontanoNordOvest                 = (D sonar_4 LONTANO);
MoltoLontanoNordOvest            = (D sonar_4 MOLTOLONTANO);


################################
##   OSTACOLI                 ##
################################

OstacoloVicinoRotazioneAntioraria      =
                           (OR     (P VicinoNordOvest)
                                   (OR     (P VicinoOvest)
                                           (P VicinoSudOvest)
                                   )
                           );

OstacoloMedioRotazioneAntioraria       =
                           (OR     (P MedioNordOvest)
                                   (OR     (P MedioOvest)
                                           (OR     (P MedioSudOvest)
                                                   (P MedioNordLeft)
                                           )
                                   )
                           );

OstacoloLontanoRotazioneAntioraria     =
                           (OR     (P LontanoNordOvest)
                                   (OR     (P LontanoOvest)
```

```
                                        (P LontanoSudOvest)
                                )
                        );


OstacoloVicinoRotazioneOraria   =
                        (OR     (P VicinoNordEst)
                                (OR     (P VicinoEst)
                                        (P VicinoSudEst)
                                )
                        );

OstacoloMedioRotazioneOraria    =
                        (OR     (P MedioNordEst)
                                (OR     (P MedioEst)
                                        (P MedioSudEst)
                                )
                        );

OstacoloLontanoRotazioneOraria    =
                        (OR     (P LontanoNordEst)
                                (OR     (P LontanoEst)
                                        (P LontanoSudEst)
                                )
                        );


OstacoloVicinoFrontale    =
                        (OR    (P VicinoNordLeft)
                                        (OR    (P VicinoNord)
                                                        (P
                        VicinoNordRight)
                               )
                        );

OstacoloMedioFrontale     =
                        (OR    (P MedioNordLeft)
                               (OR    (P MedioNordEst)
                                              (OR    (P MedioNord)
                                                     (OR    (P
                                      MedioNordRight)
                                                            (P
                                      MedioNordOvest)
                                                     )
                                             )
                               )
                        );

OstacoloLontanoFrontale   =
```

```
                            (OR    (P LontanoNordLeft)
                                   (OR    (P LontanoNordEst)
                                                 (OR    (P LontanoNord)
                                                        (OR    (P
                            LontanoNordRight)
                                                                      (P
                            MedioNordOvest)
                                                        )
                                                 )
                                   )
                            );

OstacoloLontanoSchiena      =
                            (OR    (P LontanoSudOvest)
                                          (OR    (P LontanoSudEst)
                                                 (P LontanoSud)
                                                 )
                            );

OstacoloMedioSchiena        =
                            (OR    (P MedioSudOvest)
                                   (OR    (P MedioSudEst)
                                                 (P MedioSud)
                                                 )
                            );


OstacoloVicinoSchiena       =
                            (OR    (P VicinoSudOvest)
                                   (OR    (P VicinoSudEst)
                                                 (P VicinoSud)
                                                 )
                            );




################################
#  riconoscimento persona      ##
################################

PersonOutOfRange    = (D PersonDistance OUT_OF_RANGE);

PersonVeryClose     = (D PersonDistance VERY_CLOSE);
PersonInContact     = (D PersonDistance IN_CONTACT);
PersonClose         = (D PersonDistance CLOSE);
PersonNear          = (D PersonDistance NEAR);
PersonFar           = (D PersonDistance FAR);
```

```
PersonInContact     = (D PersonDistance IN_CONTACT);
PersonXVeryClose    = (AND (P PersonVeryClose)(NOT (P PersonInContact)));

PersonN             = (OR (D PersonAngle N1) (D PersonAngle N2));
PersonNE            = (D PersonAngle NE);
PersonENE           = (D PersonAngle ENE);
PersonESE           = (D PersonAngle ESE);
PersonSE            = (D PersonAngle SE);
PersonS             = (D PersonAngle S);
PersonSW            = (D PersonAngle SW);
PersonWNW           = (D PersonAngle WNW);
PersonWSW           = (D PersonAngle WSW);
PersonNW            = (D PersonAngle NW);

################################
#  InteractWithPerson          ##
################################
InteractWithPerson  = (P PersonVeryClose);

################################
#  LookForPerson               ##
################################
LookForPerson       = (P PersonOutOfRange);

##############################################
#  Movimenti testa in direzione della faccia ##
##############################################
HeadTop             = (D HeadY HEAD_TOP);
HeadMiddle          = (D HeadY HEAD_MIDDLE);
HeadDown            = (D HeadY HEAD_DOWN);

HeadRight           = (D HeadX HEAD_RIGHT);
HeadCenter          = (D HeadX HEAD_CENTER);
HeadLeft            = (D HeadX HEAD_LEFT);

################################
# Up and Down                 ##
################################
HeadXTop            = (D always TRUE);
HeadXDown           = (D always TRUE);

############################################
# BounceBack - 2 sensori in ogni direzione  ##
############################################
LineNRX             = (D lineNRX TRUE);
LineNLX             = (D lineNLX TRUE);
LineSRX             = (D lineSRX TRUE);
LineSLX             = (D lineSLX TRUE);
```

```
LineWRX             = (D lineWRX TRUE);
LineWLX             = (D lineWLX TRUE);
LineERX             = (D lineERX TRUE);
LineELX             = (D lineELX TRUE);

LineN               = (OR (D lineNRX TRUE) (D lineNLX TRUE));
LineS               = (OR (D lineSRX TRUE) (D lineSLX TRUE));
LineW               = (OR (D lineWRX TRUE) (D lineWLX TRUE));
LineE               = (OR (D lineERX TRUE) (D lineELX TRUE));

LineActive          = (OR
                                (OR
                                     (P LineW)
                                           (OR (P LineN)
                                                 (P LineS))
                                 )
                                 (P LineE)
                           );
LineDisactive       = (NOT (P LineActive));


########################################
# BumpStop - 2 sensori Nord e 2 Sud    ##
########################################
BumpNRX             = (D bumpNRX TRUE);
BumpNLX             = (D bumpNLX TRUE);
BumpSRX             = (D bumpSRX TRUE);
BumpSLX             = (D bumpSLX TRUE);
BumpN               = (OR (D bumpNRX TRUE) (D bumpNLX TRUE));
BumpS               = (OR (D bumpSRX TRUE) (D bumpSLX TRUE));

BumpActive          = (OR (P BumpN)(P BumpS));
BumpDisactive       = (NOT (P BumpActive));


FaceLeft            = (D FacePosX FACE_LEFT);
FaceRight           = (D FacePosX FACE_RIGHT);

#FaceTop            = (D FacePosY FACE_TOP);
#FaceBottom         = (D FacePosY FACE_BOTTOM);
```

Predicati multilivello (PredicateAction.ini)

```
PropFwRwSteady      = (D Proposedcmd_v STEADY);
PropFwVerySlow      = (D Proposedcmd_v VSLOWFW);
PropFwSlow          = (D Proposedcmd_v SLOWFW);
PropFwMedium        = (D Proposedcmd_v MEDIUMFW);
PropFwFast          = (D Proposedcmd_v FASTFW);
```

```
PropFwVeryFast      = (D Proposedcmd_v VFASTFW);
PropRwVerySlow      = (D Proposedcmd_v VSLOWRW);
PropRwSlow          = (D Proposedcmd_v SLOWRW);
PropRwMedium        = (D Proposedcmd_v MEDIUMRW);
PropRwFast          = (D Proposedcmd_v FASTRW);
PropRwVeryFast      = (D Proposedcmd_v VFASTRW);

PropRxLxSteady      = (D Proposedcmd_w STEADY);

PropRxVerySlow      = (D Proposedcmd_w VSLOWRX);
PropRxSlow          = (D Proposedcmd_w SLOWRX);
PropRxMedium        = (D Proposedcmd_w MEDIUMRX);
PropRxFast          = (D Proposedcmd_w FASTRX);
PropRxVeryFast      = (D Proposedcmd_w VFASTRX);

PropLxVerySlow      = (D Proposedcmd_w VSLOWLX);
PropLxSlow          = (D Proposedcmd_w SLOWLX);
PropLxMedium        = (D Proposedcmd_w MEDIUMLX);
PropLxFast          = (D Proposedcmd_w FASTLX);
PropLxVeryFast      = (D Proposedcmd_w VFASTLX);


DirAvanti           = (OR  (P PropFwVerySlow)
                           (OR (P PropFwSlow)
                           (OR (P PropFwMedium)
                            (OR (P PropFwFast) (P PropFwVeryFast)))));
DirIndietro         = (OR  (P PropRwVerySlow)
                           (OR (P PropRwSlow)
                           (OR (P PropRwMedium)
                           (OR (P PropRwFast) (P PropRwVeryFast)))));
DirAntioraria       = (OR  (P PropLxVeryFast)
                           (OR (P PropLxFast)
                           (OR (P PropLxMedium)
                           (OR (P PropLxSlow) (P PropLxVerySlow)))));
DirOraria           = (OR  (P PropRxVeryFast)
                           (OR (P PropRxFast)
                           (OR (P PropRxMedium)
                           (OR (P PropRxSlow) (P PropRxVerySlow)))));

DirAvantiGTFast     = (P PropFwVeryFast);
DirAvantiGTMedium   = (OR  (P DirAvantiGTFast) (P PropFwFast));
DirAvantiGTSlow     = (OR  (P DirAvantiGTMedium) (P PropFwMedium));
DirAvantiGTVerySlow = (OR  (P DirAvantiGTSlow) (P PropFwSlow));

DirIndietroGTFast   = (P PropRwVeryFast);
DirIndietroGTMedium = (OR  (P DirIndietroGTFast) (P PropRwFast));
DirIndietroGTSlow   = (OR  (P DirIndietroGTMedium) (P PropRwMedium));
DirIndietroGTVerySlow= (OR (P DirIndietroGTSlow) (P PropRwSlow));
```

```
DirOrariaGTFast     = (P PropRxVeryFast);
DirOrariaGTMedium   = (OR  (P DirOrariaGTFast) (P PropRxFast));
DirOrariaGTSlow     = (OR  (P DirOrariaGTMedium) (P PropRxMedium));
DirOrariaGTVerySlow = (OR  (P DirOrariaGTSlow) (P PropRxSlow));

DirAntiorariaGTFast = (P PropLxVeryFast);
DirAntiorariaGTMedium= (OR(P DirAntiorariaGTFast) (P PropLxFast));
DirAntiorariaGTSlow = (OR  (P DirAntiorariaGTMedium) (P PropLxMedium));
DirAntiorariaGTVerySlow= (OR     (P DirAntiorariaGTSlow) (P PropLxSlow));

DirAntiorariaLTSlow = (P PropLxVerySlow);
DirAntiorariaLTMedium= (OR (P PropLxSlow) (P DirAntiorariaLTSlow));
DirAntiorariaLTFast = (OR (P PropLxMedium) (P DirAntiorariaLTMedium));
DirAntiorariaLTVeryFast= (OR (P PropLxFast) (P DirAntiorariaLTFast));

DirOrariaLTSlow     = (P PropRxVerySlow);
DirOrariaLTMedium   = (OR (P PropRxSlow) (P DirOrariaLTSlow));
DirOrariaLTFast     = (OR (P PropRxMedium) (P DirOrariaLTMedium));
DirOrariaLTVeryFast = (OR (P PropRxFast) (P DirOrariaLTFast));
```

Dettaglio dei comportamenti di livello 1

GoToPerson.rul

```
##################################
#      Person Out Of Range       #
##################################
(PersonVeryClose)                                   => (cmd_w STEADY)
       (cmd_v STEADY);
(PersonOutOfRange)                                  => (cmd_w STEADY)
       (cmd_v STEADY);


###################################
#            Person Far           #
###################################
(AND   (PersonFar) (PersonN))                 => (cmd_w STEADY)
(cmd_v VFASTFW);
(AND   (PersonFar) (PersonNE))          => (cmd_w VSLOWRX)         (cmd_v
FASTFW);
(AND   (PersonFar) (PersonENE))         => (cmd_w MEDIUMRX)        (cmd_v
FASTFW);
(AND   (PersonFar) (PersonS))                 => (cmd_w STEADY)
(cmd_v VFASTRW);
(AND   (PersonFar) (PersonNW))          => (cmd_w VSLOWLX)         (cmd_v
FASTFW);
(AND   (PersonFar) (PersonWNW))         => (cmd_w MEDIUMLX)        (cmd_v
FASTFW);
(AND   (PersonFar) (PersonSE))          => (cmd_w VSLOWLX)         (cmd_v
```

```
FASTRW);
(AND   (PersonFar) (PersonESE))         => (cmd_w MEDIUMLX)        (cmd_v
FASTRW);
(AND   (PersonFar) (PersonSW))          => (cmd_w VSLOWRX)         (cmd_v
FASTRW);
(AND   (PersonFar) (PersonWSW))         => (cmd_w MEDIUMRX)        (cmd_v
FASTRW);

#################################
#            Person Near        #
#################################
(AND   (PersonNear) (PersonN))          => (cmd_w STEADY)   (cmd_v FASTFW);
(AND   (PersonNear) (PersonNE))         => (cmd_w SLOWRX)(cmd_v FASTFW);
(AND   (PersonNear) (PersonENE))        => (cmd_w MEDIUMRX) (cmd_v FASTFW);
(AND   (PersonNear) (PersonS))          => (cmd_w STEADY)   (cmd_v FASTRW);
(AND   (PersonNear) (PersonNW))         => (cmd_w SLOWLX)(cmd_v FASTFW);
(AND   (PersonNear) (PersonWNW))        => (cmd_w MEDIUMLX) (cmd_v FASTFW);
(AND   (PersonNear) (PersonSE))         => (cmd_w SLOWLX)(cmd_v FASTRW);
(AND   (PersonNear) (PersonESE))        => (cmd_w MEDIUMLX) (cmd_v FASTRW);
(AND   (PersonNear) (PersonSW))         => (cmd_w SLOWRX)(cmd_v FASTRW);
(AND   (PersonNear) (PersonWSW))        => (cmd_w MEDIUMRX) (cmd_v FASTRW);

#################################
#            Person Close       #
#################################
(AND   (PersonClose) (PersonN))   => (cmd_w STEADY)(cmd_v FASTFW);
(AND   (PersonClose) (PersonNE))  => (cmd_w MEDIUMRX)(cmd_v MEDIUMFW);
(AND   (PersonClose) (PersonENE))=> (cmd_w FASTRX)(cmd_v MEDIUMFW);
(AND   (PersonClose) (PersonS))   => (cmd_w STEADY)(cmd_v FASTRW);
(AND   (PersonClose) (PersonNW))  => (cmd_w MEDIUMLX)(cmd_v MEDIUMFW);
(AND   (PersonClose) (PersonWNW))=> (cmd_w FASTLX)(cmd_v MEDIUMFW);
(AND   (PersonClose) (PersonSE))  => (cmd_w MEDIUMLX)(cmd_v MEDIUMRW);
(AND   (PersonClose) (PersonESE))=> (cmd_w FASTLX)(cmd_v MEDIUMRW);
(AND   (PersonClose) (PersonSW))  => (cmd_w MEDIUMRX)(cmd_v MEDIUMRW);
(AND   (PersonClose) (PersonWSW))=> (cmd_w FASTRX)(cmd_v MEDIUMRW);

#################################
#            Person Very Close  #
#################################
(AND   (PersonVeryClose) (PersonN))     => (cmd_w STEADY)(cmd_v MEDIUMFW);
(AND   (PersonVeryClose) (PersonNE))    => (cmd_w FASTRX)(cmd_v MEDIUMFW);
(AND   (PersonVeryClose) (PersonENE))   => (cmd_w FASTRX)(cmd_v SLOWFW);
(AND   (PersonVeryClose) (PersonS))     => (cmd_w STEADY)(cmd_v MEDIUMRW);
(AND   (PersonVeryClose) (PersonNW))    => (cmd_w FASTLX)(cmd_v MEDIUMFW);
(AND   (PersonVeryClose) (PersonWNW))   => (cmd_w FASTLX)(cmd_v SLOWFW);
(AND   (PersonVeryClose) (PersonSE))    => (cmd_w FASTLX)(cmd_v MEDIUMRW);
(AND   (PersonVeryClose) (PersonESE))   => (cmd_w FASTLX)(cmd_v SLOWRW);
(AND   (PersonVeryClose) (PersonSW))    => (cmd_w FASTRX)(cmd_v MEDIUMRW);
(AND   (PersonVeryClose) (PersonWSW))   => (cmd_w FASTRX)(cmd_v SLOWRW);
```

```
##################################
# Person in contact              #
##################################
(AND   (PersonInContact) (PersonN))     => (cmd_w STEADY)(cmd_v STEADY);
(AND   (PersonInContact) (PersonNE))    => (cmd_w SLOWRX)(cmd_v SLOWRW);
(AND   (PersonInContact) (PersonENE))   => (cmd_w SLOWRX)(cmd_v SLOWRW);
(AND   (PersonInContact) (PersonS))     => (cmd_w STEADY)(cmd_v STEADY);
(AND   (PersonInContact) (PersonNW))    => (cmd_w SLOWLX)(cmd_v SLOWRW);
(AND   (PersonInContact) (PersonWNW))   => (cmd_w SLOWLX)(cmd_v SLOWRW);
(AND   (PersonInContact) (PersonSE))    => (cmd_w SLOWLX)(cmd_v SLOWFW);
(AND   (PersonInContact) (PersonESE))   => (cmd_w SLOWLX)(cmd_v SLOWFW);
(AND   (PersonInContact) (PersonSW))    => (cmd_w SLOWRX)(cmd_v SLOWFW);
(AND   (PersonInContact) (PersonWSW))   => (cmd_w SLOWRX)(cmd_v SLOWFW);
```

LookForPerson.rul

```
(LookForPerson)     => (lookforperson TRUE) (cmd_v STEADY) (cmd_w SLOWRX) ;
```

FollowJoypadMotion.rul

```
##################################
#  Velocità avanti/indietro      #
##################################
(FwRwSpeedJoySteady)      => (cmd_v STEADY);
(FwSpeedJoyVerySlow)      => (cmd_v VSLOWFW);
(FwSpeedJoySlow)          => (cmd_v SLOWFW);
(FwSpeedJoyMedium)        => (cmd_v MEDIUMFW);
(FwSpeedJoyFast)          => (cmd_v FASTFW);
(FwSpeedJoyVeryFast)      => (cmd_v VFASTFW);
(RwSpeedJoyVerySlow)      => (cmd_v VSLOWRW);
(RwSpeedJoySlow)          => (cmd_v SLOWRW);
(RwSpeedJoyMedium)        => (cmd_v MEDIUMRW);
(RwSpeedJoyFast)          => (cmd_v FASTRW);
(RwSpeedJoyVeryFast)      => (cmd_v VFASTRW);

##################################
# velocità destra/sinistra       #
##################################

(RxLxSpeedJoySteady)      => (cmd_w STEADY);
(RxSpeedJoyVerySlow)      => (cmd_w VSLOWRX);
(RxSpeedJoySlow)          => (cmd_w SLOWRX);
(RxSpeedJoyMedium)        => (cmd_w MEDIUMRX);
(RxSpeedJoyFast)          => (cmd_w FASTRX);
(RxSpeedJoyVeryFast)      => (cmd_w VFASTRX);
(LxSpeedJoyVerySlow)      => (cmd_w VSLOWLX);
(LxSpeedJoySlow)          => (cmd_w SLOWLX);
```

```
(LxSpeedJoyMedium)          => (cmd_w MEDIUMLX);
(LxSpeedJoyFast)            => (cmd_w FASTLX);
(LxSpeedJoyVeryFast)        => (cmd_w VFASTLX);
```

InteractWithPerson.rul

```
(InteractWithPerson)        => (interact TRUE);
(FaceLeft)                  => (cmd_w SLOWLX);
(FaceRight)                 => (cmd_w SLOWRX);
```

HeadPosition.rul

```
(HeadRight)                 => (cmd_head_delta_x LEFT);
(HeadCenter)                => (cmd_head_delta_x CENTER);
(HeadLeft)                  => (cmd_head_delta_x RIGHT);

(HeadTop)             => (cmd_head_delta_y NORTH);
(HeadMiddle)          => (cmd_head_delta_y MIDDLE);
(HeadDown)            => (cmd_head_delta_y SUD);
```

StayBlocked.rul

```
(Always)              => (cmd_v STEADY) (cmd_w STEADY);
```

Dettaglio dei comportamenti di livello 2

AvoidObstacles.rul

```
#============================#
#  ostacoli frontali vicini  |
#============================#
(AND (OstacoloVicinoFrontale) (DirAvanti))
=>
(&DEL.cmd_v ANY)
(cmd_v STEADY);

(AND   (OstacoloVicinoFrontale)
       (AND   (DirAvanti)
              (AND   (PropRxLxSteady)
                     (NOT
                            (OR    (VicinoOvest)
                                   (OR    (VicinoEst)
                                          (OR    (VicinoNord)
                                                 (OR    (VicinoNordOvest)
                                                        (VicinoNordEst)

                                                 )
                                          )
                                   )
```

```
                              )
                        )
                  )
            )
)
=>
(cmd_v MEDIUMFW);

#============================#
#      Ostacoli              |
#============================#
(AND   (DirAvanti)
       (AND   (PropRxLxSteady)
              (AND   (OR    (NOT (OstacoloVicinoFrontale))
                            (AND   (OstacoloVicinoFrontale)
                                   (AND   (NOT (VicinoOvest))
                                          (AND   (NOT (VicinoEst))
                                                 (AND   (NOT (VicinoNord))
                                                        (AND   (NOT
(VicinoNordOvest))
                                                                      (NOT
(VicinoNordEst))
                                                        )
                                                 )
                                          )
                                   )
                            )
                     )
                     (AND   (NOT (OstacoloVicinoRotazioneOraria))
                            (AND   (NOT (OstacoloVicinoRotazioneAntioraria))
                                   (AND   (MedioNordRight)
                                          (MedioNordLeft)
                                   )
                            )
                     )
              )
       )
)
=>
(cmd_v SLOWFW);


#============================#
#   ostacoli frontali medi   |
#============================#

(AND   (DirAvanti)
       (AND   (OstacoloMedioFrontale)
              (AND   (NOT   (OstacoloVicinoFrontale) )
```

```
                    (OR    (PropFwFast)
                           (PropFwVeryFast)
                    )
             )

      )
)
=>
(&DEL.cmd_v ANY)
(cmd_v MEDIUMFW);

#============================#
# ostacoli frontali  lontani |
#============================#
(AND   (DirAvanti)
       (AND   (OstacoloLontanoFrontale)
              (AND   (NOT   (OstacoloMedioFrontale))
                     (PropFwVeryFast)
              )
       )
)
=>
(&DEL.cmd_v ANY)
(cmd_v FASTFW);



#=======================#
#|   ostacoli laterali  |
#=======================#

(AND (OstacoloVicinoRotazioneAntioraria) (DirAntioraria))
=>
(&DEL.cmd_w ANY)
(cmd_w STEADY);

(AND (OstacoloVicinoRotazioneOraria) (DirOraria))
=>
(&DEL.cmd_w ANY)
(cmd_w STEADY);

(AND   (OstacoloMedioRotazioneAntioraria)
       (AND   (AND   (OR    (PropLxMedium)
                            (OR    (PropLxFast)
                                   (PropLxVeryFast)
                            )
                     )
                     (DirAntioraria)
              )
```

```
              (NOT (OstacoloVicinoRotazioneAntioraria))
       )
)
=>
(&DEL.cmd_w ANY)
(cmd_w SLOWLX);

(AND   (OstacoloMedioRotazioneOraria)
       (AND   (AND   (OR    (PropRxMedium)
                            (OR    (PropRxFast)
                                   (PropRxVeryFast)
                            )
                     )
                     (DirAntioraria)
              )
              (NOT (OstacoloVicinoRotazioneOraria))
       )
)
=>
(&DEL.cmd_w ANY)
(cmd_w SLOWRX);

(AND
       (AND (DirAvanti)
              (OstacoloMedioFrontale)
       )
       (NOT (OstacoloMedioRotazioneAntioraria))
)
=>     (&DEL.cmd_v ANY) (cmd_v MEDIUMFW)(cmd_w FASTLX);


(AND
       (AND (DirAvanti)
              (OstacoloMedioFrontale)
       )
       (NOT (OstacoloMedioRotazioneOraria))
)
=>     (&DEL.cmd_v ANY) (cmd_v MEDIUMFW)(cmd_w FASTRX);
```

BounceBack.rul

```
(LineActive)
=> (&DEL.cmd_v ANY) (&DEL.cmd_w ANY);

(LineN)
=>  (cmd_v FASTRW) (cmd_w STEADY);

(LineS)
```

```
=>     (cmd_v FASTFW)      (cmd_w STEADY);

(LineW)
=>     (cmd_v STEADY) (cmd_w FASTLX);

(LineE)
=>     (cmd_v STEADY)      (cmd_w FASTRX);
```

BumpStop.rul

```
(AND (BumpN)
       (BumpS)
)
=>(&DEL.cmd_v ANY)(&DEL.cmd_w ANY)(cmd_v STEADY)(cmd_w STEADY);
############################
#  Bump Nord               #
############################
(AND   (BumpN)
              (DirAvanti)
)=>(&DEL.cmd_v ANY)(&DEL.cmd_w ANY);

(AND
       (BumpN)
              (AND (DirAvanti)
                     (NOT (OstacoloVicinoRotazioneAntioraria))
              )
)
=>(cmd_v VFASTRW) (cmd_w FASTRX);

(AND
       (BumpN)
              (AND (NOT (OstacoloVicinoRotazioneOraria))
                     (AND (OstacoloVicinoRotazioneOraria)
                            (DirAvanti)
                     )
       )
)
=>(cmd_v VFASTRW) (cmd_w FASTLX);


############################
#  Bump Sud                #
############################
(AND   (BumpS)
              (DirIndietro)
)=>(&DEL.cmd_v ANY)(&DEL.cmd_w ANY);



(AND
```

```
        (BumpS)
        (AND (NOT (BumpN))
                (AND (NOT (OstacoloVicinoFrontale))
                        (AND   (NOT (OstacoloVicinoRotazioneOraria))
                                (DirIndietro)
                        )
                )
        )
)
=>(cmd_v VFASTFW)(cmd_w VSLOWRX);
(AND
        (BumpS)
        (AND (NOT (BumpN))
                (AND (NOT (OstacoloVicinoFrontale))
                        (AND   (NOT (OstacoloVicinoRotazioneAntioraria))
                                (DirIndietro)
                        )
                )
        )
)
=>(cmd_v VFASTFW)(cmd_w VSLOWLX);
```