Departamento de Ciencias de la Computación e Inteligencia Artificial
Konputazio Zientziak eta Adimen Artifiziala Saila




# A Society of Agents and Music for Machines

Dissertation submitted to the Department of Computer Sciences and Artificial Intelligence of the University of the Basque Country in partial fulfillment of the requirements for the PhD degree in Computer Sciences by:

Urko Esnaola Campos

Directed by:

Tim Smithers

Donostia - San Sebastián, September 2005

# Acknowledgements

I acknowledge the support (financial and technical) of the San Sebastián Technology Park, and also the Iñaki Goenaga Foundation. I thank all the people at the third floor; particularly Alberto Oraá for all the electronic supplies and technical attention, Juan Cendoya for his disposition to help when trying to repair the motion control of the robot MiReLa, Josetxo Pena for his invaluable efforts contacting people to help in various aspects of the project, and Ibane Rodriguez, who we brought mad preparing the transport of MiReLa to the Moncloa Palace. I gratefully acknowledge Manuel Cendoya and Amaia Bernaras for the great effort they did to start up and maintain the project MiReLa. To Amaia, also for her attention and effort to make the life of the investigation group better.

I acknowledge the collaboration of Ulahi S.A, particularly of Borja Inillos for their help in the modifications done to one of the lifts of the Main Building of the San Sebastián Technology Park, for part of the project reported here.

I acknowledge the conversations with all the guys in the investigation group of the San Sebastián Technology Park. In particular the active help of Mirari Olaizola, Ruben León, Pablo Aramendia, and Iñigo Martínez, and specially the great disposition to share his knowledge in computer science and to help in various aspects of Jon Azpiazu.

I acknowledge the kind attention received in the EHU from my tutor Basilio Sierra, and the secretary of CCIA Elena Bidondo.

I can't forget to mention the cleaners and the "txispas" of the Main Building of the San Sebastián Technology Park who have always been ready to help testing the robot MiReLa. And I acknowledge the security guards of the Main Building of the San Sebastián Technology Park for their great company in the long hours we spent together (the robot MiReLa's home is next to the security place in the building) and for their disposition to help. Specially Bernardino Perez, who filmed many tests done with the robot MiReLa. Thank you Bernardino.

My most grateful thanks are for Tim Smithers, for all the conversations, suggestions, ideas, and knowledge he shared with me, his invaluable corrections (sometimes discouraging) over the work presented in this thesis, and in general for his behaviour not only as a thesis director but also as a friend.

Thank you to my parents (Eskerrik asko Aita eta Ama!) and to my girlfriend, Marta. I chose to do a PhD thesis, they suffered the consequences. All my love for them, specially for Marta to whom sometimes I gave grounds to be jealous of MiReLa, but who even so, revised this thesis document several times.

# Abstract

A robot is a machine including a number of sensors and a number of actuators. A living creature has also a number of sensors and a number of actuators. They help to survive in different kinds of environments. There are many kinds of creatures with many kinds of sensors and actuators which can survive in many different kinds of environments. Many of the living creatures developed a neuron based structure: a brain. Some of them (as humans) developed a very complex structure which allows them not only to react to the things sensed from their sensory system, but also to learn new abilities, remember things, have emotions, reason, etc. The term mind is used to refer to such higher functions of the human brain. Although no body totally understands how the mind really works, many theories have been proposed about how it may work. One of them is presented in the book *The Society Of Mind* of Marvin Minsky, a mathematician who is considered one of the fathers of Artificial Intelligence. Antonio Damasio, Professor of Neurology, in his book *Descartes' Error* coincides in many details with the view of Minsky about how mind may work.

Minsky uses the following words to describe the essence of how mind may work:

> "I'll call "Society of Mind" this scheme in which each mind is made of many smaller processes. These we'll call agents. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these agents in societies—in certain very special ways—this leads to true intelligence. "
>
> —Marvin Minsky, "The Society of Mind", pp 17.

*The Society Of Mind* has served as source of inspiration to develop *Societies of Agents* which have been implemented in the robot MiReLa (a RWI, B21 robot) and in the lamp MiFaRe, (a lamp operated by a musical language). This societies are formed from agents which can do very simple things by themselves, the communication between them is also simple and limited, but all together make the machines able to achieve progressively more complex tasks.

For the interaction from people to the machines and from machines to the people, sound generation/recognition has been chosen as the main interaction way. It is a way we humans share with many living creatures to exchange information remotely. Humans developed spoken languages, which, coding/decoding sound according to complex phonetic and grammatical rules, makes possible a very rich and effective interaction. The problem

is that nowadays there is nothing available to help decode reliably any of these spoken languages artificially in no controlled environments. Nor is clear that such a complex coding/decoding system is necessary to support simple interaction with machines such as, for example, to switch the lights on and off. An artificial music language named *MiReLa Music Language* has been created which uses music to support people-machine interaction.

The *MiReLa Music Language* can be played by many musical instruments, mobile phones and PDAs, and it can be whistled. It has been used to interact with the robot MiReLa and to operate a lamp and it has probed to be robust and reliable in noisy environments.

This thesis proposes *Societies of Agents* to be implemented in machines or sensor networks (such as those used in Ambient Intelligence) to make it possible to incrementally add new capabilities and suggests one way to build them, keeping the agents and the communication between them simple. For people-machine, machine-machine interaction, it proposes a Musical Language, which is easy to whistle.

# Laburpena

Robot batek zenbait sentsore eta zenbait ekintzaile biltzen ditu bere baitan. Izaki bizidun batek ere zenbait sentsore eta zenbait ekintzaile biltzen ditu bere baitan. Berauek ingurune desberdinetan bizirauten laguntzen diote. Izaki bizidun askok neuronazko egitura bat garatu dute: garun bat. Berauetako zenbaitek (gizakiek adibidez) oso egitura complexua garatu dute zeinak beraien sentsoreetatik jasotako estimuluen aurrean erreakzionatzeko ezezik, gaitasun berriak ikasteko, gauzak gogoratzeko, emozioak bizitzeko, arrazoitzeko, eta abarrerako gai egiten baititu. Gizakiaren garunaren gorengo mailako funtzio hauei erreferentzia egiteko, adimena hitza erabiltzen da. Naiz eta inork ez duen bere osotasunean ulertzen adimenak benetan nola funtzionatzen duen, teoria asko proposatu izan dira nola funtzionatu lezakeen argitu nahian. Marvin Minsky-ren *Adimenaren Elkartea* (*The Society Of Mind*) liburuak teoria hauetako bat azaltzen du. Minsky matematikari bat da eta Inteligentzia Artifizialaren sortzailetariko bat. Neurologian katedradun den Antonio Damasiok, bere *Descartesen Errorea* liburuan, Minskyrekin bat egiten du alderdi askotan, adimenak nola funtzionatu lezakeen adierazterakoan.

Minskyk honako hitzok erabailtzen ditu adimenak nola funtzionatu lezakeenaren esentzia esplikatzeko:

> " "Adimenaren Elkartea" deituko diot adimen bakoitza prozezu txikiagoko askoz osatua dagoen eskema honi. Adimeneko agente bakoitzak berekasa, batere adimenik edo pentsamendurik behar ez duten zenbait gauza sinple egin ditzake soilik. Baina agente hauek—oso era berezitan—elkartetan batzen baditugu, honek benetako inteligentziara garamatza."
>
> —Marvin Minsky, *The Society Of Mind* (*Adimenaren Elkartea*), pp 17.

*Adimenaren Elkartea* inspirazio iturri izan da *Agente Elkarte* batzuk garatu eta implementatzeko MiReLa robotean (RWI, B21 robota) eta baita MiFaRe lanparan ere (zeina musikazko lengoaia baten bidez piztu eta itzali baitaitekeen). Elkarte hauek, beren kabuz oso gauza sinpleak egin ditzaketen agenteek osatzen dituzte, berain arteko komunikazioa ere sinplea eta mugatua da, baina denen artean posible egiten dute makinek gero eta egiteko konplexuagoak burutu ditzaten.

Jendea eta makinen arteko elkarreraginerako soinua sortuz/errekonozituz egitearen bidea aukeratu da. Gizakiok eta beste izaki bizidun askok, soinua erabiltzen dugu informazioa urrutitik transmititzeko. Gizakiaren kasuan, hitzegindako hizkuntzak, garatu ditugu. Hauek, soinua, zenbait lege fonetiko eta gramatikalekin, kodifikatu eta deskodifikatuz,

oso elkarreragin aberatsa eta eranginkorra posible egiten dute. Gauza da, ez dagoela gaur egun ezer hitzegindako hizkuntzarik errekonozitu dezakeenik egoera ez kontrolatuetan. Ez dago argi, ezta ere, era honetako kodifikazio/deskodifikazio sistema konplexua beharrezkoa denik, argiak piztu eta itzali, erako elkarreragin sinpleak gauzatzeko makinekin. Jendea eta makinen arteko elkarreraginerako, musika bidezko lengoaia artifizial bat sortu da. MiReLa Musika Lengoaia deitu zaio.

MiReLa Musika Lengoaia, musika instrumentu, telefono mugikorr eta PDA askorekin, sortu daiteke, baita txistu eginda ere. MiReLa robotarekin eta MiFaRe lanpararekin elkarreragiteko erabili da eta toki zaratatsuetan erabiltzeko baliogarria dela frogatu da.

Tesi honek, makinetan edo sentsore sareetan (Inteligentzia Anbientalean erabiltzen direnetan adibidez) *Agente Elkarte*ak implementatzeko proposamena luzatzen du, eta agenteak eta beraien arteko komunikazioa sinple mantenduz, elkarte hauek eraikitzeko era bat iradokitzen du. Jendea-makinak, makinak-makinak elkarreragina egiteko berriz, txistuz jotzeko erraza den Musika bidezko Lengoaia bat erabiltzeko proposamena luzatzen du.

# Table of Contents

**I Introduction** **1**

**1 Introduction** . . . . . **3**

1.1 Motivation . . . . . 3

1.2 Research Goals and Approaches . . . . . 4

1.2.1 A Way of Putting Things Together . . . . . 4

1.2.2 An Effective Way for People-Machine and Machine-Machine Interaction . . . . . 6

1.3 Main Outcomes . . . . . 8

1.3.1 The *Society of Agents* . . . . . 8

1.3.2 *MiReLa Music Language* . . . . . 8

1.3.3 Application Experiments . . . . . 8

1.3.4 Main Research Conclusions . . . . . 10

**2 Robots and People Today** . . . . . **11**

2.1 Service Robots of Type II. Tendencies and Directions . . . . . 11

2.1.1 Towards Human Shape . . . . . 12

2.1.2 People-Machine Interaction . . . . . 17

2.1.3 Do We Really Need Human Like Service Robots? . . . . . 18

2.2 Other MiReLa Type Robots . . . . . 20

2.2.1 Sage . . . . . 20

2.2.2 Rhino and Minerva . . . . . 25

2.2.3 Hermes . . . . . 31

2.2.4 RoboX . . . . . 35

2.3 Critical Evaluation . . . . . 40

**II A Society of Agents and A Musical Language 41**

**3 A Society of Agents . . . . . 43**

3.1 The Society of Mind . . . . . 44

3.1.1 Mind Separated . . . . . 45

3.1.2 *A-brain* and *B-brain* . . . . . 47

3.1.3 Conflict Handling . . . . . 47

3.2 A *Society of Agents* . . . . . 48

3.2.1 *Request/Serve Data* . . . . . 49

3.2.2 *Send/Receive Instructions* . . . . . 49

3.2.3 An Example . . . . . 49

3.2.4 Conflict Handling: Priority Levels . . . . . 51

3.3 Implementation . . . . . 52

3.3.1 Agent Interaction . . . . . 52

3.3.2 Watchdogs . . . . . 55

3.3.3 Programing the Agents—C++ and Perl . . . . . 56

3.4 Concluding Comments . . . . . 58

**4 MiReLa Music Language (MML) . . . . . 59**

4.1 Looking for a Frequency Range for MML . . . . . 60

4.2 The MML Alphabet . . . . . 61

4.3 MML Phrases . . . . . 62

4.3.1 Word Separation: A Silence and the Tonic Note $(c, C)$ . . . . . 62

4.3.2 End of a Phrase. . . . . 62

4.3.3 Note Duration. . . . . 63

4.4 From Musical Sounds to MML Phrases . . . 63
4.5 Directed Listening. . . . 64
4.6 Example of an Interaction in MML. . . . 65

**III Application Experiments 67**

**5 The Lamp MiFaRe . . . 69**
5.1 Hardware Configuration . . . 70
5.2 A Society of Agents to Control Lights using the MML . . . 71
5.2.1 The FFTServer Agent . . . 74
5.2.2 The NoteChecker Agent . . . 75
5.2.3 The WordChecker Agent . . . 77
5.2.4 The WhistleHelper Agent . . . 78
5.2.5 The SoundServer Agent . . . 79
5.2.6 The AmbientNoise Agent . . . 81
5.2.7 The SoundPainter Agent . . . 83
5.2.8 The RelayController Agent . . . 84
5.3 Concluding Comments . . . 85

**6 The Lift . . . 87**
6.1 Operating a Lift with DTMF Tones . . . 88
6.1.1 The NORCOMM NC400 Board . . . 88
6.2 Installation . . . 90
6.2.1 Installation of a System in the Ground Floor and in the First Floor to Call the Lift Using DTMF Tones . . . 91
6.2.2 Installation of a System in the Cabin of the Lift to Operate It Using DTMF Tones . . . 94
6.3 Concluding Comments . . . 96

**7 The Robot MiReLa . . . 99**

7.1 The robot . . . 100
7.1.1 Electrical Power . . . 101
7.1.2 Computer System of MiReLa . . . 101
7.1.3 Ultra-sound Sensors . . . 101
7.1.4 Infra-red Sensors . . . 104
7.1.5 Laser Range Finder . . . 104
7.1.6 Bumper Sensors . . . 104
7.1.7 Electronic Compass . . . 105
7.1.8 Sound Processing Devices . . . 105
7.1.9 Cameras and Pan Tilt Units . . . 105
7.1.10 Motion Control . . . 105
7.1.11 Multi Sensor Processors MSPs . . . 106
7.2 Navigation System . . . 107
7.2.1 Motion Control . . . 107
7.2.2 Navigation . . . 109
7.3 A Society of Agents for MiReLa . . . 115
7.3.1 The Navigator Agent . . . 117
7.3.2 The SensorServer Agent . . . 119
7.3.3 The BumperMove Agent . . . 120
7.3.4 The DoorClosedAlert Agent . . . 121
7.3.5 The PhraseChecker Agent . . . 123
7.3.6 The MusicFollower Agent . . . 124
7.3.7 The GestureServer Agent . . . 126
7.3.8 The MovieServer Agent . . . 128
7.3.9 The NavPainter Agent . . . 129
7.3.10 The LiftHandler Agent . . . 131
7.3.11 The WebPage Agent . . . 133
7.3.12 The InformationPoint Agent . . . 135

7.3.13 The Hunger Agent . . . 137
7.3.14 The LampSwitcher Agent . . . 139
7.3.15 The VisionPeopleDetector Agent . . . 140
7.3.16 The VisionPeopleTracker Agent . . . 141
7.4 Concluding Comments . . . 142

**8 Trials, Results, and Evolution . . . 143**
8.1 MiReLa at the Open Day Trials . . . 144
8.1.1 The 2002 Open Day . . . 145
8.1.2 The 2003 Open Day . . . 148
8.1.3 The 2004 Open Day . . . 156
8.2 MiReLa and MiFaRe at the Moncloa Palace . . . 158
8.3 Whistle From Far Test . . . 160
8.4 Whistling the Name Test . . . 163
8.5 Concluding Comments . . . 163

**IV Evaluation, Conclusions and Future Work 165**

**9 Evaluation . . . 167**
9.1 A *Society of Agents* . . . 167
9.1.1 Agent Data Analysis . . . 168
9.1.2 Comparison with The Society of Mind . . . 174
9.1.3 Emotions . . . 178
9.1.4 Some Properties . . . 179
9.1.5 Concluding Comments . . . 180
9.2 A Music Language for Machines . . . 181
9.2.1 Goal . . . 181
9.2.2 The MiReLa Music Language . . . 181
9.2.3 Evaluation of the DTMF Sound Interaction With the Lift . . . 183

9.2.4 Concluding Comments . . . . . 184

**10 Conclusions and Future Work . . . . . 187**

10.1 Conclusions . . . . . 187

10.1.1 A *Society of Agents* for Developing Complex Behaviours in Machines . . . . . 187

10.1.2 A Music Language for Machines . . . . . 188

10.2 Future Directions . . . . . 189

10.2.1 Future Development Ideas . . . . . 190

10.2.2 The Navigator agent as a *Society of Agents* . . . . . 192

10.2.3 Remembering and Learning . . . . . 194

10.2.4 Emotions and reasoning . . . . . 199

**References . . . . . 205**

# List of Figures


2.1 Pictures of Honda's bipedal robot Asimo and android robot Repliee Q1Expo from Osaka University. . . . 12

2.2 Robotic faces. . . . 14

2.3 Animatronic Faces. . . . 15

2.4 Bipedal Robots using precise joint-angle control. . . . 16

2.5 Bipedal Robots using passive-dynamic control. . . . 17

2.6 The robot Sage. . . . 20

2.7 Visual landmark situated on the end of a route. . . . 22

2.8 Robots Rhino on the left and Minerva on the right. The picture on the top left corner indicates the location of the different sensors of the robot Rhino. The picture on the top right corner, shows the robotic face installed for the robot Minerva. . . . 25

2.9 Minerva's exclusive Web control interface. Users give target positions by clicking inside the map. The robot moves to targets on a first-come-first-serve manner. . . . 26

2.10 Control page of Rhino Web interface. Users can choose exhibits they want the robot to go to as well as viewpoints. . . . 27

2.11 Software Architecture of Minerva and Rhino [33]. . . . 29

2.12 The robot Hermes. . . . 31

2.13 Hardware architecture of Hermes. . . . 32

2.14 The software architecture of Hermes. A situation-oriented, behaviour-based robot architecture. . . . 33

2.15 a) Main components of RoboX b) A picture of RoboX. . . . 35

2.16 a) Hardware Architecture of RoboX. . . . 36

2.17 RoboX. Software Architecture. . . . . 38

3.1 An agent adds one skill, it may accept some kinds of instructions about how to perform the skill, and it may generate some kinds of data. It may also need to send instructions and get data from other agents. . . . . 48

3.2 The names written in the middle of the circles, are names of the agents. A lower case name with an '*' above the agent name, indicates the type of instructions the agent accepts. A lower case name without an '*' below the agent name, indicates the output data generated by the agent. Continuous arrows indicate that the agent at the source instructs the agent at the target; the label of the arrows indicate the type of the instruction it is sending. Dashed arrows indicate that the agent at the source requests data from the agent at the target; the label of the arrow indicates which output data are requested. . . . . 50

3.3 An *Instruction Message* data structure. . . . . 54

3.4 *Instruction Message* that the Telephone agent may send to the TV agent to lock the `Volume Level to Set` instruction type in priority level 2. . . . . 55

3.5 *Instruction Message* that the Telephone agent may send to the TV agent to lock the `Volume Level to Set` instruction type in priority level 2. . . . . 55

4.1 Figure of a piano keyboard indicating the note names following the octave designation system of the American Acoustic Society, [3]. It shows that the MML alphabet notes are defined in the octave C6-C7. . . . . 61

4.2 An example of a vocabulary item in MML. . . . . 63

5.1 The lamp MiFaRe. . . . . 69

5.2 The lamp MiFaRe is composed of a conventional desk lamp connected to one of the eight relays of a relay board MICROPIK PS6E8S [73]. This relay board is connected to the serial port of a PC, which has a microphone and a loudspeaker connected to its sound card. . . . . 70

5.3 Representation of the agents of the *Society of Agents* implemented in the lamp MiFaRe. Section 3.2 explains the meaning of the different types of arrows (dashed or continuous), their labels and the nature of the names written at each agent (agent name, accepted instructions, and generated data). . . . . 72

5.4 The SoundPainter agent's screen-shot. In the left window, the `Detected Note`s are painted on the upper part, and the `FFT Spectrum` is painted on the lower part. In the right window the `Detected Note`s are painted in a musical staff. . . . . 83

6.1 Picture of the lift door in the ground floor. It shows a detailed view of where the loudspeaker and the microphone have been installed. . . . . 91

6.2 Block diagram of the components of an installation done for one floor, to operate, the button to call the lift, using DTMF tones. This diagram shows that two relays of the NC400 are connected in parallel to the button switch. 92

6.3 Picture of the lift cabin. It shows a detailed view of where the microphone and the loudspeaker have been installed in the lift cabin. . . . . 94

6.4 Block diagram of the components of the installation to operate the lift cabin to go to the ground floor, and the first floor, and to maintain the doors open, using DTMF tones. This diagram shows that two relays of one NC400 are connected in parallel to the button switch of the lift to go to the ground floor and the first floor and one relay of the other NC400 is connected in serial to the button switch to maintain the doors of the lift open. . . . . 95

7.1 Front view and back view of the robot MiReLa. . . . . 100

7.2 Picture indicating the location of the sensors in the robot MiReLa. . . . . 102

7.3 Hardware architecture of the robot MiReLa. . . . . 103

7.4 A filter gets data from the sensors and processes it to get two values, an $\alpha_f$ (forward speed) and a $\beta_f$ (rotational speed). . . . . 107

7.5 The $\alpha_f$ values $(\alpha_{f1}, \alpha_{f2}, ..., \alpha_{fn})$ are combined in a harmonic mean to get a unique $\alpha_m$, and the $\beta_f$ values $(\beta_{f1}, \beta_{f2}, ..., \beta_{fn})$ are combined in a harmonic mean to get a unique $\beta_m$. The resulting $\alpha_m$ is the forward speed sent to the translation motors and $\beta_m$ is the rotational speed sent to the rotation motors. . . . . 108

7.6 A *Transition* specifies the active *Filters*: the *Filters* whose $\alpha_f$ and $\beta_f$ values need to be combined; and the termination conditions: the conditions which, if one is satisfied, terminate the *Transition*. . . . . 109

7.7 Each *Transition Group* specifies a set of *Transitions* and the *Transition Groups* the robot is allowed to start doing next, ANTG (Allowed Next *Transition Groups*). . . . . 109

7.8 A general graph made from the ANTGs of the *Transition Groups*. . . . . 110

7.9 Plan of the ground floor of the Main Building of the San Sebastián Technology Park, with the representation of the *Transition Group* graph implemented in MiReLa for the 2003 Open Day (see section 8.1.2). . . . . 113

7.10 Representation of the agents of the *Society of Agents* implemented in the robot MiReLa. Section 3.2 explains the meaning of the different types of arrows (dashed or continuous), their labels and the nature of the names written at each agent (agent name, accepted instructions, and generated data). . . . . 116

7.11 OpenDoor movie. Picture designed and produced by Tim Smithers. . . . . 128

7.12 The NavPainter agent's screen-shot. On the top part of the figure, titled GLEnclosure, a representation of the `Laser`, `Sonar` and `Infrared` (enclosure) data is shown. The cylinder represents the robot MiReLa and the walls are painted from the `Laser` data values. The `Sonar` data and the `Infrared` data are painted in the form of beams. On the window on the lower left corner titled GLCompass, the `Compass` data is represented and on the window on the lower right corner titled GLBaseWin the `Infrared` data got from the base is represented. . . . . 130

7.13 Screen-shot of the Web page to operate the robot MiReLa. It offers the possibility to tell the robot to go to five different places in the Main Building of the San Sebastián Technology Park. They are Reception, Video-Conference room, MiReLa's home, Bar, and the Fun Technology Room. The place is selected between the five different options and when the button Joan/Ir (Go) is clicked the petition to navigate to the selected place is sent to the robot MiReLa. The Web page also offers the possibility to send a request to the robot to stop and do a pan motion by clicking on the button StartPan and to stop doing the pan clicking on the button StopPan. . . . . 134

7.14 Screen-shot of the Information Point. The top right part of the window shows a plan of the Main Building of the San Sebastián Technology Park. Clicking on different points of it, shows information of the selected part of the building. It offers the possibility to be guided to the selected point by the robot MiReLa. The plan also shows where is the robot located at each moment. . . . . 136

8.1 Plan of the ground floor of the Main Building of the San Sebastián Technology Park. The trajectory followed by the robot MiReLa in this floor at the 2002 Open Day is marked. MiReLa navigated from "MiReLa Etxea" (MiReLa's home: the room where MiReLa is usually kept) to the lift on the ground floor, on its way to the Auditorium on the first floor, and it navigated from the lift to "MiReLa Etxea" in the way back from the Auditorium. 145

8.2 Plan of the first floor of the Main Building of the San Sebastián Technology Park. The trajectory followed by the robot MiReLa on this floor at the 2002 Open Day is marked. MiReLa navigated from the lift to the Auditorium and returned back to the lift on this floor. . . . . 146

8.3 Plan of the ground floor of the Main Building of the San Sebastián Technology Park, with a representation of the *Transition Group* graph implemented in MiReLa to navigate in the ground floor at the 2003 Open Day. . . . . 149

8.4 Picture showing a person video-conferencing with the person at the "Reception Desk" through MiReLa. . . . . 150

8.5 Picture showing the computer with the Web page control interface at the "Reception Desk". . . . . 151

8.6 Pictures showing the occupancy level of the Hall of the Main Building of the San Sebastián Technology Park between 12:00 and 13:30 during the 2003 Open Day. . . . . 154

8.7 Picture taken at the entrance of the "Consejo de Ministros" building of the Moncloa Palace, of MiReLa and the three components of the team which travelled for the event INGENIO 2010. The names of the persons from left to right are Urko Esnaola, Jon Azpiazu, and Tim Smithers. . . . . 159

8.8 Picture of a person whistling to MiReLa from the balcony of the Hall of the Main Building of the San Sebastián Technology Park to tell it to go to "MiReLa Etxea". (Figure built from two shots of a video-clip.) . . . . . 161

8.9 Picture of a person whistling to the lamp MiFaRe from outside of the office. (Figure built from three shots of a video-clip.) . . . . . 162

10.1 The Navigator agent split in many simpler agents arranged in a *Society of Agents*. . . . . 193

# List of Tables


3.1 Signals to determine the connection type. . . . . 52

3.2 Table with the names and description of the Perl scripts implemented for the *Society of Agents*. . . . . 57

4.1 Human singing voice ranges, in Hz. . . . . 60

4.2 The MML alphabet: note names and frequencies. For the note names, the first column shows the names given to the notes of the MML alphabet based on the English notation (notes A, B, C, are written in capital letters to help making more clear them to represent higher notes than the notes c, d, e, f, g), the second row shows the names of the alphabet notes using the notation created by Guido D'Arezzo [67] used in Latin countries, which have been used to define the name of the robot MiReLa and the lamp MiFaRe, and the third column shows their names following the octave designation system of the American Acoustic Society, [3]. . . . . 61

5.1 Skill Table for the FFTServer agent. . . . . 74

5.2 Skill Table for the NoteChecker agent. . . . . 75

5.3 Skill Table for the WordChecker agent. . . . . 77

5.4 Skill Table for the WhistleHelper agent. . . . . 78

5.5 Skill Table for the SoundServer agent. . . . . 79

5.6 Skill Table for the AmbientNoise agent. . . . . 81

5.7 Skill Table for the SoundPainter agent. . . . . 83

5.8 Skill Table for the RelayController agent. . . . . 84

6.1 The DTMF signals used to call the lift in the ground floor and the first floor. 93

6.2 The DTMF signals used to operate the lift, inside the lift cabin. . . . . 96

7.1 How the ANTGs of the *Transition Groups* are completed. . . . . 112

7.2 *Transition Groups* used for the 2003 Open Day. . . . . 113

7.3 Skill Table for the Navigator agent. . . . . 117

7.4 Skill Table for the SensorServer agent. . . . . 119

7.5 The data types generated by the different kinds of sensors of the robot MiReLa. . . . . 119

7.6 Skill Table for the BumperMove agent. . . . . 120

7.7 Skill Table for the DoorClosedAlert agent. . . . . 121

7.8 The DoorClosedAlert agent alerts. Instructions sent by the DoorClosedAlert agent to other agents depending on the time elapsed since a door in the path of the robot was detected to be closed. . . . . 122

7.9 Skill Table for the PhraseChecker agent. . . . . 123

7.10 Skill Table for the MusicFollower agent. . . . . 124

7.11 Skill Table for the GestureServer agent. . . . . 126

7.12 Shows the gestures the robot MiReLa does when a specific instruction is received. . . . . 127

7.13 Skill Table for the MovieServer agent. . . . . 128

7.14 Skill Table for the NavPainter agent. . . . . 129

7.15 Skill Table for the LiftHandler agent. . . . . 131

7.16 Skill Table for the WebPage agent. . . . . 133

7.17 Skill Table for the InformationPoint agent. . . . . 135

7.18 Skill Table for the Hunger agent. . . . . 137

7.19 Skill Table for the LampSwitcher agent. . . . . 139

7.20 Skill Table for the VisionPeopleDetector agent. . . . . 140

7.21 Skill Table for the VisionPeopleTracker agent. . . . . 141

9.1 Data generated by the Agents. . . . . 169

9.2 Accepted instructions data. . . . . 170

9.3 Table showing which agents request the data generated by each of the nine agents which generate data. . . . . 171

9.4 Instructions received by the agents. . . . . 172

9.5 Classification of all the agents in *A-brain* and *B-brain* terms. . . . . 176

# Part I

# Introduction

# Chapter 1

# Introduction

## 1.1 Motivation

Service robots are being coming a reality, more robot applications and demonstrations are seen every day. Between service robots, two types can be distinguished:

**Type I:** Those designed to do tasks for people.

**Type II:** Those designed to help people.

Between Type I service robots, a big effort is being done in developing cleaning robots [90], [89], [47]. One of the most successful products in this field is the vacuum cleaner robot Roomba from iRobot Corp. [69], which achieves the task of vacuum cleaning in a rather successful way and it can be acquired for a significantly lower price (€ 206) than other robots doing the same task, as for example the robot Trilobite of Electrolux [107] (€ 1,480).

Service robots of Type I are also being developed for agriculture. Robots are being used to control automate milking [72] or for weed control purposes [71], [99],[8], for example. They are becoming more and more important in medicine [45], [59]. And they are also being developed for people search and rescue tasks and surveillance.

Between Type II robots, they are those designed to guide people through buildings such as museums [81],[98],[97] or hospitals [75], [86]. And also some attempts have been done to install robots in shopping centres [61], [58]. The robot MiReLa [51] is one of such service robots of Type II. It is a RWI (Real World Interface Inc.) B21 [85] robot which has been developed to guide people through the Main Building of the San Sebastián Technology Park.

The main difference between Type I and Type II service robots is that the Type II robots need to interact more actively and effectively with people to get to help them.

Service robots (Type I and Type II) are machines that offer greater and more complex functionality than many of the machines we are used to using, except perhaps for the

computer, but robots are supposed to work in the kinds of places and spaces we live and work in. Many of them (and all of the guide robots), have to move around the real world, thus they need to sense the environment and deal with its variations to achieve robust navigation. In addition, service robots of Type II must engage in an effective human-robot interaction. They have to process the data obtained from their sensors in different ways for interaction purposes or for navigation and they also have to control the actuators in different ways depending on if they need to navigate or interact with people. For example, a robot equipped with some cameras and a pan-and-tilt actuator to move them, may need to move the cameras to track different objects using artificial vision, but it may also need to move them in such ways to make gestures in order to better interact with people. It is also important that the capabilities a robot has, to be useful for it to help acquiring new capabilities, and not to have to start from scratch when ever a new capability is intended to be implemented in the robot.

There are many different aspects which need to come together in a service robot, specially in those of the Type II. To be practical, robust, reliable, and maintainable, a way is needed of putting together all these things, in a simple and extensible way: an open approach is needed to the integration of the robot's behaviours and capacities.

For service robots of Type II, to be useful, usable, and acceptable, also a way is needed to interact with them, a way that works well in the uncontrolled conditions the robots need to operate in.

The robot MiReLa, and the task assigned to it of guiding people through the Main Building of the San Sebastián Technology Park, have served as an experimental platform and task for this thesis to investigate solutions to these needs.

## 1.2 Research Goals and Approaches

From the needs identified in the previous section, two goals have been defined to be investigated in this thesis project:

1. A simple and extensible way, to put together all the different aspects involved in the operation of a service robot of Type II, and which facilitates progressive implementation of new capabilities, taking the maximum advantage from the ones already implemented.
2. An effective, efficient and acceptable way for people-machine and machine-machine interaction.

### 1.2.1 A Way of Putting Things Together

One common practice to achieve a complex goal (such as the resolution of a problem, to make a robot able to perform a task, etc), is to identify different parts, split the goal into simpler subgoals, and try to achieve each of the subgoals. But achieving each of the

subgoals doesn't necessarily mean to achieve the original goal. It is necessary to find a way so that each of the achievements complement the others and all together help achieving the original goal.

In the concrete case of a robot, these subgoals may, for example, be software applications which give to the robot different capabilities and which might be developed by different people. They, all together, should make the robot able to achieve one task.

A robot offers many sensors and actuators. As said in the previous section, they may need to be used for different tasks. Also the capabilities implemented into the robot, may need to be used in different ways to help the robot acquire new capabilities.

It has been investigated a way to support the development of systems composed of elements doing simple things. These elements may be developed in any programing language by different people, thus a simple way must be offered to combine these elements in the system to achieve progressively more complex goals. Each element should be able to use the capabilities implemented by other elements so as not to have to re-implement them by itself. Also, an easy and effective way must be offered to administer resources, physical resources (sensors and actuators) and software resources (other software element's capabilities).

#### *The Society Of Mind* as an Approach

A robot is a machine including a number of sensors and a number of actuators. A living creature has also a number of sensors and a number of actuators. They help the creature to survive in different kinds of environments. There are many kinds of creatures with many kinds of sensors and actuators which can survive in many different kinds of environments. Many of the living creatures developed a neuron based structure: a brain. Some of them (as humans) developed a very complex structure which allows them not only to react to the things sensed from their sensory system, but also to learn new abilities, remember things, have emotions, reason, etc. The term mind is used to describe such higher capabilities of the human brain. Although no body completely understands how the mind really works, many theories have been proposed about how it may work. One of them is presented in the book *The Society Of Mind* [74] of Marvin Minsky, a mathematician who is considered one of the fathers of the Artificial Intelligence. He presents some ideas and directions about how a mind could be implemented artificially. Antonio Damasio, Professor of Neurology, in his book *Descartes' Error* [44] coincides in many details with the view of Minsky about how mind may work.

Minsky uses the following words to describe the essence of his idea about how the mind may work:

> "I'll call "Society of Mind" this scheme in which each mind is made of many smaller processes. These we'll call agents. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these agents in societies—in certain very special ways—this leads to true intelligence."
>
> —*Marvin Minsky, "The Society of Mind", pp 17.*

Minsky and Damasio, in their respective books, *The Society Of Mind* and *Descartes' Error*, understand the mind as a distributed system. The two books are studies of the human mind, this complex entity resulted from many thousands of years of evolution, governing the human body full of sensors and actuators. Marvin Minsky suggests it is built from very simple elements, doing very simple things. For this reason, it is worth to analyse their suggestions in detail to get directions to develop a simple way to help put together all the different aspects involved in the operation of a service robot of Type II, and which facilitates the incremental implementation of new capabilities taking the maximum advantage from the ones already implemented.

### 1.2.2 An Effective Way for People-Machine and Machine-Machine Interaction

The Classical approach to improving people-machine interaction is to make machines seem more like us, and, in particular, by trying to give them the ability to use Human Natural Languages. The trouble is, such attempts do not yet work well in uncontrolled and noisy environments, such as where we live and work. Nor do they mean that the machines really use our languages in the way we do: they often do not speak much like we do, and we typically have to speak to them in unnatural ways.

As time goes on, research goes on and it should be expected speech recognition to be better and better until it works well in uncontrolled and noisy environments. We shouldn't expect this to happen in the near future however. But even when reliable speech recognition systems become possible, we need to consider if we really want to talk to every machine we use. We talk to people. If the machines start understanding what we say, a big effort will be needed to avoid misunderstandings, as they may understand orders from our conversation with other people, from the radio or from the TV.

But, sound generation/reception is a very interesting (may be the most interesting) way of interaction for human beings. It is an effective natural way, for many living creatures (including humans) to transmit information remotely. This communication requires a sound generator and a sound receptor. Gesture making is another way of transmitting information remotely, it requires a gesture generator (which requires some kind of movement; hands, head, legs, eyes, eyebrows, mouth, nose ...) and a visual receptor [60]. A spoken language is a way to code information in sound according to some phonetic and grammatical rules. This also requires a way to decode the sound received by the receptor. The result is a very effective communication, making us able to exchange large and varied amounts of information in a very reliable way. Do we need such a complex system to perform simple tasks as switching the lights on and off? Why not look for a simpler code?

Why not use music?. Why not direct the detection to specific single frequencies which form a music melody, instead of the complex combinations of frequencies which form the words (vowels and consonants) of any Human Natural Language? We have a natural way to generate almost single frequency sounds, that is whistling. Using music to interact with machines would avoid them to understand orders from people talking to each other, from the TV or from the radio.

Other means could be also used apart from sound, such as wireless communications, to interact with the machines, they make an effective interaction but secret at the same time, and they cannot be generated naturally by humans. Using sound makes the interaction not to be secret to anybody. A person may not understand the meaning of a sound communication between two machines, but he or she understands there is a communication being held between them, which, in the case of wireless communication it is not detectable by human beings, thus it remains secret.

An interaction mode based upon sound has been investigated, simpler to recognise or generate than a Human Natural Language and which makes possible for recognition to be reliable in noisy environments.

### A Musical Language as an Approach

The most extended means used by humans to communicate between each other are Human Natural Languages. But also whistled languages are still used in some parts of the world. They make a whistled version of one spoken language. One example is the Silbo [7] on the island of La Gomera in the Canary Islands: it is a whistled version of Spanish. Others exist or existed in all parts of the world: the language Mazateco is whistled by the Zapotecs of Oaxaca (Mexico) [94]; the Pirahã people, living in Brazil, along the Maici river, a tributary of the Amazonas, have a whistled version of their language Pirahã [54]; the inhabitants of Kuskoy, which literally mean the Bird Village use a whistled version of Turkish [76]; The people from Aas in the Pyrenees speak a Spanish-derived dialect which used to be adapted into a whistle language; the Chepang language is whistled in Nepal [35]; whistled languages are especially common and robust today in parts of West Africa, used widely in such populous languages as Yoruba and Ngwe (see [57]). They are reliable for long distance communication.

An artificial musical language is worth to be investigated as a mean to achieve effective people-machine and machine-machine interaction. Using an artificial musical language, can be expected to achieve reliable interaction even in noisy environments and it would avoid misunderstandings from people talking. It can also be easy to avoid misunderstandings from any background music being played on the TV, the radio or whistled by any person, since the probability of these sequences of notes to correspond to a phrase of the music language is very low.

## 1.3 Main Outcomes

### 1.3.1 The *Society of Agents*

To achieve the goal 1, presented at the beginning of the previous section 1.2, a specific kind of agent network is proposed. It has been named *Society of Agents*, in reference to *The Society Of Mind*. A *Society of Agents* is formed by agents. The goal of each agent is to add one skill to the system. The communication between the agents is simple and limited. They do not negotiate with one another and they do not find effective ways to adjust to each other's interference. They ask for data and they send instructions between each other.

A proposal about how to build *Societies of Agents* is presented in chapter 3. Some of the ideas explained by Marvin Minsky in *The Society Of Mind* and by Antonio Damasio in *Descartes' Error* served as an important source of inspiration to develop the *Society of Agents*. They are presented in section 3.1. Section 9.1.2 of the Evaluation chapter, compares these ideas with the *Society of Agents* system presented in this thesis. Some other ideas taken from *The Society Of Mind* and *Descartes' Error* have been considered very interesting for future development of the *Society of Agents* and they have been collected in the sections 10.2.3 and 10.2.4.

### 1.3.2 *MiReLa Music Language*

To achieve effective people-machine, machine-machine interaction, a musical language has been developed and tested, which has been named the *MiReLa Music Language*. Chapter 4 explains this language.

### 1.3.3 Application Experiments

MiReLa, as said before, is a RWI B21 [85] robot which has been developed to guide people through the Main Building of the San Sebastián Technology Park. The robot and the task assigned to it have served to test and develop the *Society of Agents* and the *MiReLa Music Language*. Recognition and generation of the *MiReLa Music Language* has been implemented in the form of a *Society of Agents*. The main way to interact with robot MiReLa is through the *MiReLa Music Language*, although it can also be interacted from a Web page or from an Information Point (a computer to be installed in the building offering information about the building and also the possibility to be guided by the robot). The *Society of Agents* and the *MiReLa Music Language* have also been implemented to operate (switch on and off) a lamp using the *MiReLa Music Language*. The lamp has been named MiFaRe and it also served as a platform to test the *Society of Agents* and the *MiReLa Music Language*.

Chapter 7 presents the characteristics of the robot MiReLa and the implementation of the *Society of Agents*. It also explains how navigation is achieved. It has been developed the possibility to be able to tell to the robot, at any time, to navigate to different parts of the

building. This is essential for the robot to offer guidance assistance and it is important to know how it is done, in order to understand the implementation of the *Society of Agents* in the robot MiReLa.

The *MiReLa Music Language* recognition and generation is done by a number of agents implemented in robot MiReLa. These same agents have been implemented in the lamp MiFaRe together with one more agent. They make possible for the lamp to be operated using the *MiReLa Music Language*. Chapter 5 presents the implementation.

The way chosen to explain the *Society of Agents* implementation in robot MiReLa and the lamp MiFaRe is to explain each of the agents which form the respective societies. A diagram is presented at the beginning with the connections of the agents. The agents are explained in one or two pages each. The fact that all the agents of MiFaRe but one are set for recognition and generation of the *MiReLa Music Language*, makes MiFaRe a nice example to show how this is done. This is the reason why MiFaRe is explained in chapter 5, before MiReLa. These agents are not explained again in chapter 7 although they are part of the *Society of Agents* implemented in MiReLa.

In order to reliably guide people through the Main Building of the San Sebastián Technology Park, MiReLa has to use the lifts. It doesn't have any hand or manipulator, thus a system must be installed to allow it to operate them. As reasoned before, the choice made in this thesis for people-machine and machine-machine interaction, has been to use sound. Recognition and generation of the *MiReLa Music Language* is done through a PC computer. This makes quite difficult and expensive to install the *MiReLa Music Language* to operate the lifts. Computers are becoming smaller and cheaper every day, thus this is expected not to be a problem in a very near future. One practical alternative way has been investigated to make MiReLa able to operate a lift. There are many inexpensive circuit boards that decode DTMF sounds into relay activation/deactivation, thought for domotic applications. They can be used in the lift so that they make possible the lifts to be operated from DTMF sounds. Although DTMF tones can't be whistled, they offer a sound based interaction. The experience and the knowledge gained from the installation and testing of a system to make possible to operate the lifts from DTMF sounds, is very useful for a future implementation of the *MiReLa Music Language* to operate them. Chapter 6 explains the implementation of a system done in one of the lifts of the Main Building of the San Sebastián Technology Park to allow to operate it from DTMF sound signals. It must be noted that for the simplicity of the interaction required and for reliability reasons, the interaction has been designed to be DTMF sound signal based. No DTMF sound based language was created.

Chapter 8 explains the trials whose results helped to develop both the *Society of Agents* system and the *MiReLa Music Language*. As said before, they have been developed and tested in robot MiReLa. This chapter explains the evolution the *Society of Agents* system and the *MiReLa Music Language* went through. The Open Day trials, together with demos done to people visiting the Main Building of the San Sebastián Technology Park helped for it. The Open Day trials are explained in detail in the section 8.1 of this chapter. They are events set to present the work of the San Sebastián Technology Park, and the various research centres, companies, and other organisations situated in the Technology Park, to a general public. They are held in November each year and almost two thousand

people visit the Main Building of the San Sebastián Technology Park during the morning reserved for an Open Day. These are tough condition for MiReLa to operate in, but the evaluation of the results of the performance of the robot in them is very valuable to set future research directions.

Chapter 9 evaluates the *Society of Agents* implementations done for the robot MiReLa and the lamp MiFaRe, and the *MiReLa Music Language*.

Chapter 10 presents the main conclusions of the research presented in this thesis, together with some directions for future work.

### 1.3.4 Main Research Conclusions

Creating *Societies of Agents* in the way proposed in this thesis (see chapter 3), to put together all the aspects involved in a service robot of Type II, showed to be effective, usable, and extensible. It is also shown how it supports the integration and incremental development and evolution of systems as a whole. This is evaluated in section 9.1.4.

The use of the *MiReLa Music Language* is shown to be reliable and effective in noisy environments to communicate with machines. It is also shown that it is possible to make a machine recognise the *MiReLa Music Language* phrases whistled by a person located far (about fifteen metres) from it.

# Chapter 2

# Robots and People Today

The aim of this chapter is to present a review of service robots today, particularly service robots of Type II. As explained in the introduction, they are robots designed to help people. It aims to show the tendencies in this type of robotics research. It also presents some robots prepared for similar tasks to MiReLa, the robot being developed to guide people through the Main Building of the San Sebastián Technology Park. It focusses on the interaction methods and the software architecture used by these robots.

## 2.1 Service Robots of Type II. Tendencies and Directions

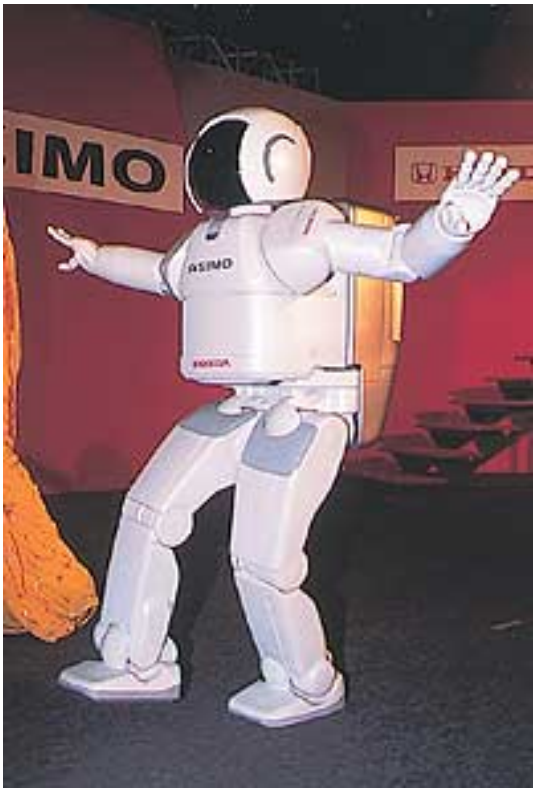
The tendency in service robots of Type II seems to be that of making the robots as much human like as possible. One example is the robot Asimo [36], developed by Honda which can walk, dance, and climb up and down the stairs imitating human movements. Another example is the robot Repliee Q1Expo [112], developed at the Osaka University. It is an android robot which has flexible silicone for skin rather than hard plastic, and a number of sensors and motors to allow the robot to turn its face and react in a human-like manner (as far as it could be found in the literature, this robot moves its face, not the whole body). Figure 2.1 shows a picture of both these robots.

The most often repeated reason to build human-like robots is that this makes easier for human beings to interact with them. In the words of Professor Ishiguro, designer of Repliee:

> "I have developed many robots before, but I soon realised the importance of its appearance. A human-like appearance gives a robot a strong feeling of presence."
>
> Japanese develop 'female' android—BBC News Web site [112]

They are made human like in shape (they are given a face and try to make them bipedal) and they are made human like in interaction (using Human Natural Languages). In both of the areas (shape and interaction) very interesting results have already been achieved.But,

it is important not to forget that making a machine more human like, doesn't necessarily make it more useful.




The Honda Asimo




The Osaka University Repliee Q1Expo

Figure 2.1: Pictures of Honda's bipedal robot Asimo and android robot Repliee Q1Expo from Osaka University.

The following section, Towards Human Shape, shows many researchers from all around the world share the point of view of Professor Ishiguro. But it is not clear,that this is the best option to achieve reliable people-machine interaction. This is discussed in section 2.1.3.

### 2.1.1 Towards Human Shape

There are one hundred and forty-five android projects all around the world. Most of them are being developed in Japan (fifty-four) and in the US (thirty) [114]. Almost all of the most of the well-known research centres in the world are working in at least one android project. Android projects are in fashion these days. Important effort is being done in building faces for robots and also to make them able to walk as humans.

**Robot Faces**

Between those investigations concerned with faces, many approaches try to make expressive faces. They are robotic faces and they aim to give expression to the robot. They include human face parts, such as eyes, eyebrows and lips in a robotic imitation. Kismet is well known among them [23]. Kismet engages people in natural and expressive face-to-face interaction. The museum tour guide robot, Minerva also has a face with its mouth, eyes and eyebrows [103]. Dokia robotic head built by Intelligent Earth [48], incorporates two cameras, two motion sensors, one chemical sensor, one loud speaker and is awaiting the addition of two microphone 'ears'. The C-Lab at the University of Paderborn (Germany) has built a head for simulating emotions. Its called Mexi (Machine with Emotionally eXtended Intelligence) [34]. Hossein Mobahi from the Azad University of Tehran (Iran) has developed an animatronic face, called Aryan. The face is comprised of eight degrees of freedom. Its controllable features are eyes, eyebrows, jaw and neck. MAVERic is a versatile robotic vision head developed for oculo-motor research at the RIKEN Brain Science Institute's Laboratory for Mathematical Neuroscience, in collaboration with the University of Southern California's Computational Learning and Motor Control Lab. Maxwell from Medonis Engineering, is a robot incorporating an animatronic head [49]. Robot Maximilian is a face built from a mannequin. It is equipped with two cameras and it can close and open its eye-lids [84]. Inkha is a reactive robotic head built by Walker and Longyear from the UK. It tracks moving objects using cameras inside its eyes, it issues verbal statements based on visual, sensory, environmental and touch-screen input, and it demonstrates various behaviours including fright, nonchalance and boredom [110]. They are shown in figure 2.2.

Other approaches try to build faces identical to humans. They are called animatronic faces. Some examples are Saya and Roberta [115] from the Science University of Tokyo, K-Bot [20] of Hanson , who also sculpted the head for the EAP (Electro-Active Polymer) [9] head project from NASA. Robotic faces build by David Ng [78], the anthropomorphic Robot Head [106], the talking head Doc Beardsley from Carnegie Mellon University [100]. Figure 2.3 show them.

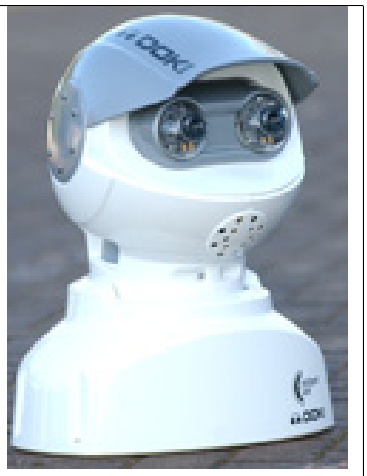
Other approaches, like Vikia [32] install a face, on a screen as an avatar [65]. Others [81] simply paint a face on to the robot. The main goal is to make humans not be afraid of interacting with the robot.

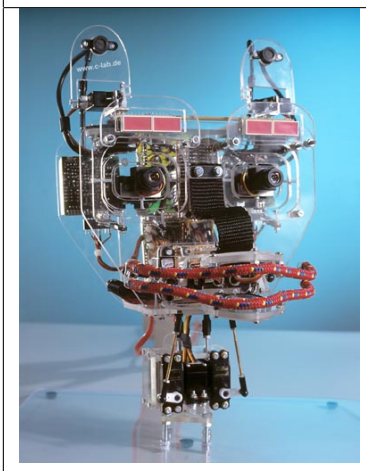
Although some approaches are very interesting, it is not always necessary for a robot to have a face to engage in an effective communication. A face on a robot could be useful but not necessary.

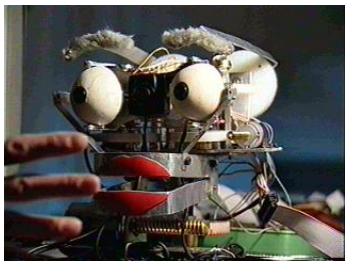
What we see from all these work is much engineering sophistication, but little communality or clear evaluation of the real effectiveness of robotic heads and faces in the robot-people interaction. In particular, there is still no clear signs of what is needed, how those artificial heads and faces need to work in order to support good acceptable robot-people interaction.


Kismet


Minerva


Doki


Mexi

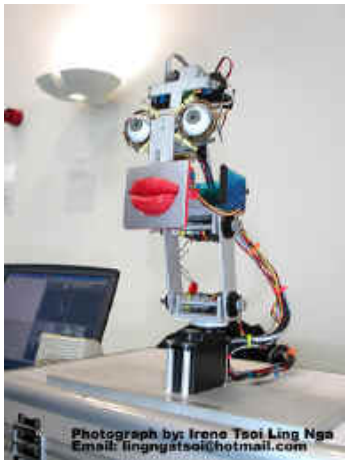

Aryan



MAVERic


Maxwell


Maximilian



Inkha

Figure 2.2: Robotic faces.



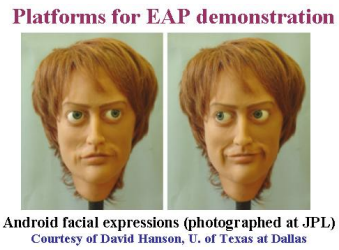

Saya



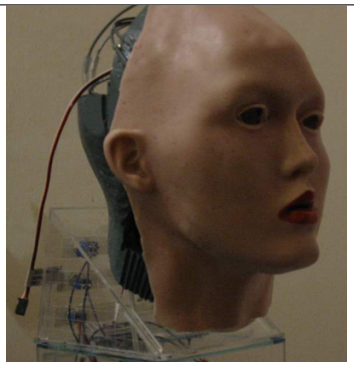
Roberta



K-Bot



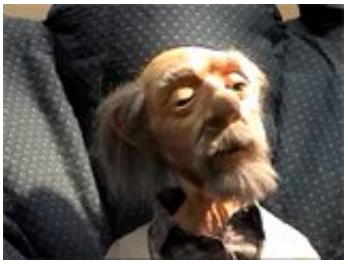

NASA'S Head Project



NG-face



YFX head



Beardsley



Valerie

Figure 2.3: Animatronic Faces.

### Bipedal Robots

In the way towards human like robots, another very challenging research field is that of building reliable bipedal machines. Most of the investigation is directed to use the mainstream control paradigm, namely, *precise joint-angle control*. They actively control every joint angle. This work started to show some good and impressive results during the last few years. Robot Asimo [62] is one of them. It can walk, dance, and climb up and down stairs. It stands about 120 cm tall and it weighs about 43 kg. Honda spent 10 years developing this android which was introduced in Tokyo, December 20, 1996. Kawada Industries HRP-2P [64] robot does some very interesting things, such as assist in assembling a panel and stand up from the ground. It is 154 cm tall, weighs 58 kg and has 30 degrees of freedom. HUBO is another example of humanoid robot from Korea Advanced Institute of Science and Technology (KAIST) [95]. Toyota has another bipedal robot, as yet unnamed [96].

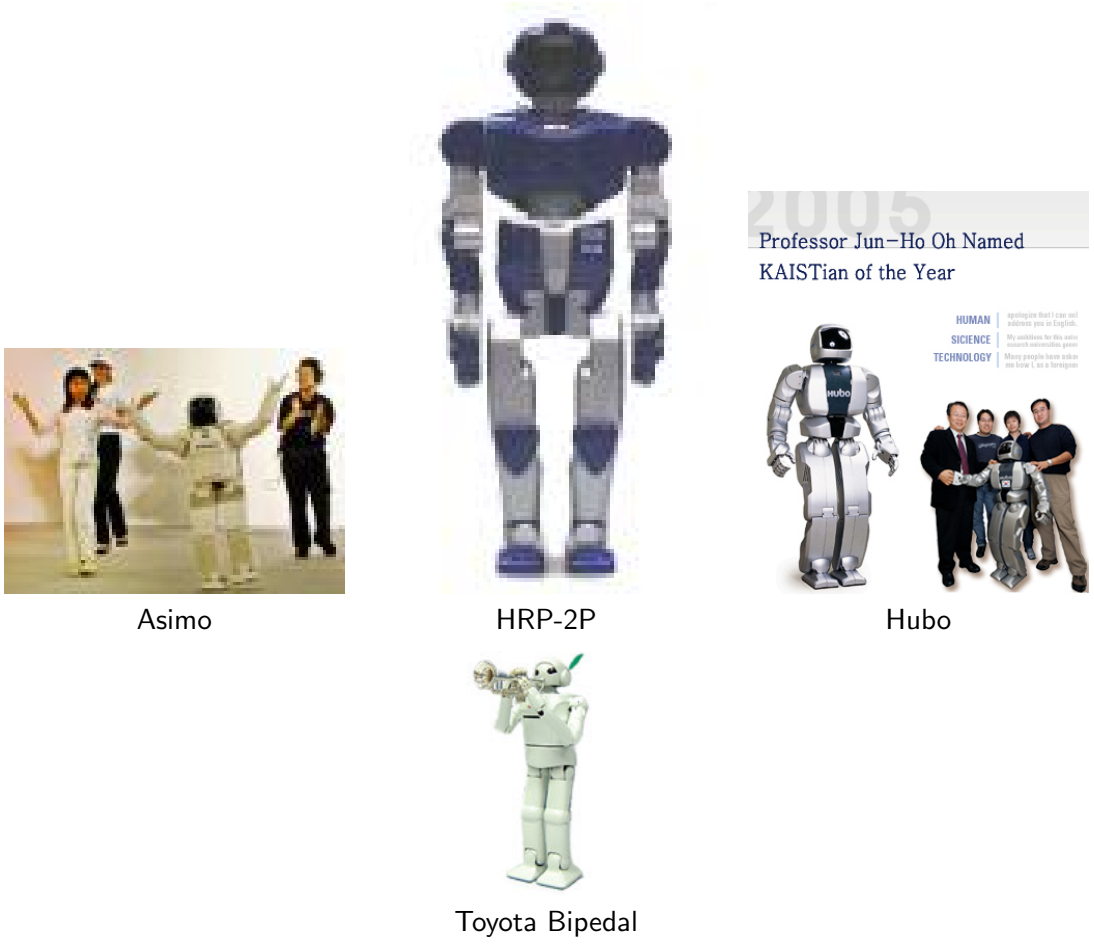
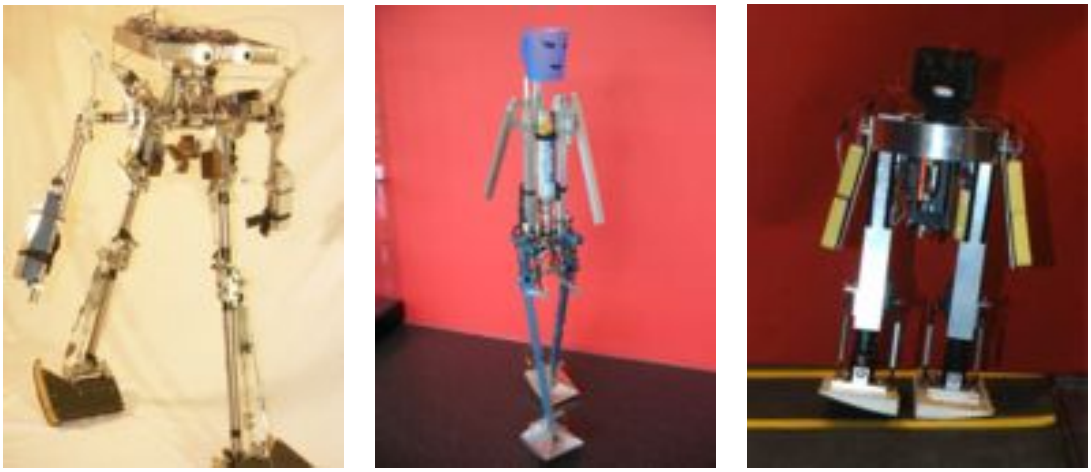



Figure 2.4: Bipedal Robots using precise joint-angle control.

The fact of doing precise joint-angle control requires large amounts of energy and quite complex control. In contrast to mainstream robots, passive-dynamic walkers do not control

any joint angle at any time. Although the work in this kind of research has not yet produced results as spectacular as the precise joint-angle control based robots (such as going up/down stairs, standing up from the ground, etc) it is a very promising research field. This approach makes the most of dynamics of structures and material properties to make a much more simpler movement control [39].



Cornell Uni. Delf Uni. MIT

Figure 2.5: Bipedal Robots using passive-dynamic control.

Passive dynamics are not applied only for bipedal robots, other researchers (R. Pfeifer et al. [56]) are investigating this approach building quadruped robots.

Once again, much impressive and impact progress is seen on solving the difficult engineering problems of bipedal locomotion in all this work, but what it is not seen is much convergence on types of solutions for types of applications. Nor is there any real attempt in this body of work to establish when bipedal robots are needed over other mobile options—such as plain old wheels. In particular, there is no evaluated results that clearly show when bipedal mobility is needed to support effective Type II service robots.

### 2.1.2 People-Machine Interaction

Sound has always been an important medium for communication between people, but somehow, it seems to be rather neglected as a medium for people-machine communications. In the case of human-computer interaction, it's more a case of having been forgotten. As [109] comments, many early computer users took advantage of an auditory output to know better what the computer was doing. One quite common trick was to tune an AM (Amplitude Modulated) radio to the RF (Radio Frequency) "noise" produced by the circuits of the computer CPU, and thereby listen to what it was doing. The sound patterns produced were quite easily learned and recognised, and helped programmers to diagnose and even debug their programs.

Vickers and Alty, [109], present a modern extension of this early computer "music," in a

program debugging support system that uses music, in its normal full sense, to provide well structured, rich, but easily learned and recognised information about a program's execution flow. They argue that this kind of *auralisation* of information offers benefits over and above the visualisation offered by conventional graphical user interface techniques.

Brewster, [25, 26], has also tried to promote the use of musical sounds in computer interfaces, with his work on *earcons* (ear-cons, as opposed to eye-cons = i-cons = icons). Earcons are abstract synthetic sounds that can be used in structured combinations to create sound messages that represent different parts of an application interface. Brewster et al, have used earcons to sonify several interfaces and have shown them to be effective at communicating complex information in sound. Brewster's pioneering work has, however, seen little further application, despite proving to be an effective way of improving and extending human-computer interfaces, and not just for visually impaired users.

An interesting alternative approach, using whistling, has been promoted and demonstrated by Marc Bohlen, an artist, and his Universal Whistling Machine (UWM), [16]. The Universal Whistling Machine (UWM), is a machine which senses the presence of moving creatures with its built-in camera and attracts people with a signature whistle. Given a response whistle, UWM counters with its own composition, based on a time-frequency analysis of the original whistled response [17].

Apart from these approaches using sound, the research on people-machine interaction has been directed towards Human Natural Language detection/generation. Many approaches, and companies can be found researching and developing speech recognition systems (refer to [40]).

### 2.1.3 Do We Really Need Human Like Service Robots?

Very interesting research approaches have been presented in the previous subsections. For example, researchers working in the development of the robot Repliee Q1Expo (see 2.1) say that people new to the robot, don't notice it is an android in the first minute or so. People may find this, attractive, impressive and amazing, because it is novel and it is clear it requires a deep study about the human face (muscles and movements) and about the materials to create artificial skins to copy human skin. Is this what we need to interact more comfortably with robots? Do we really want, not to be able to distinguish if we are in front of a robot or in front of a person?

The robotic faces, such as Kismet (see figure 2.2) are not less impressive. They have been built with the main purpose to help machines transmit emotions to humans, and thus help in the interaction [22], [21]. Many of these approaches succeed in transmitting the impression of the robot being happy, sad, or bored. A discussion about if they really feel these emotions would be interesting, but is not presented here. Humans and most of the mammals transmit many of their emotions via their faces, but is it the only way to transmit emotions? Does a robot, really need a face to give the impression that it is happy or sad? Robotic faces, are built from a complex machinery (servo-motors, and processors) to make their parts move imitating human faces. This adds more complexity to the generally tough task of maintaining a service robot. As the number of hardware components increases,

the probability for one component to fail increases. It is important for the robots to be built from the components strictly necessary, to increase their dependability.

Other approaches use avatars in computer screens mounted on the robots. The reason they claim for doing it, is that interaction is more natural for a person if he or she can direct to a face (even if it is a representation on a screen) than if they direct to something with no face. There have not been found any discussion which compares both the approaches (with face and with no face) to determine which of them makes a more effective interaction to happen. Even if experiments would be done for some concrete approaches, it would be very difficult to get conclusions objectively. We interact with many machines that don't remind us of any human aspect, and we feel comfortable with them. Why should we think that robots are more than a kind of machine capable of doing complex tasks?

Advances achieved in the attempt to make robots to walk on two feet, as humans do, have been large and impressive. But do we need service robots to be able to walk? The environments they have been thought to operate in, and were we usually live and work in, are places accessible by devices on wheels. Most of the buildings and the streets are prepared for people in wheel chairs. In terms of energy, and safety for a robot, it is more efficient to move on wheels than on two feet. Thus it is not clear service robots need to be bipedal. Nevertheless, the research done in this field may be interesting to build artificial legs to be offered to disabled people.

As said in the introduction, there is not any speech recognition system available which recognises speech in environments were the sound is not controlled. Even if they become a reality one day, it is not clear we want to talk to all the machines. Other kinds of interactions are worth to be investigated. Whistling is something we can do naturally, there are whistled languages as el Silbo at Gomera, shepherds send commands to their sheep dogs by whistling, this is not a language, but it is effective sound communication. A language based on music notes, which can be whistled is an attractive alternative, worth to be investigated as an alternative to Human Natural Languages to communicate with machines.

Service robotics is closely related to Science Fiction. Much research has been done inspired by the many robots of Science Fiction. One of the most well known robots from Science Fiction is the robot R2D2, from the film "Star Wars", directed by George Luckas. This robot speaks a language based on musical sounds, it doesn't speak any Human Natural Language. Also in the film "Close Encounters of the Third Kind", directed by Stephen Spielberg, humans use a musical language to communicate with the extraterrestrial. So it can be seen that the *MiReLa Music Language* type communication is not new to Science Fiction.

## 2.2 Other MiReLa Type Robots

### 2.2.1 Sage



Figure 2.6: The robot Sage.

Sage [81] is a robot that was installed at the Carnegie Museum of Natural History. Its goal was to provide educational content to museum visitors in order to augment their museum experience. It was installed for eight months and a half from May 22, 1998 to February 2, 1999. During this period some renovations were done at the museum covering September to November. The robot didn't operate during these three months. Thus the robot was operational for five months and a half.

Sage's original job description called for it to provide an informative 30 minute tour of Dinosaur Hall, taking visitors to a number of exhibits throughout the hall and providing a narrative at each stop. The tour given by Sage consisted of six tour stops, each with a five minute presentation.

### Hardware Architecture

The lower robot housing, extending from the floor to the top of the second ultra-sound sensor ring, is a modified Nomad XR4000 robot [79] (Nomadic Technologies, Inc. Mountain View, CA), shown in figure 2.6. The drive system consists of four wheels, each on independent, motorised wheels and driven rotationally by independent motors as well. Eight motors drive the four wheels, rotationally and translationally, enabling fully holonomic motion (see for example [5] page 90). The XR4000 sensor system consists of rotation and translation encoders on all wheels, touch-sensitive panelling covering the complete cylindrical surface and two rings of ultra-sound sensors, with twenty-four ultra-sound sensors in each ring. Computing power consists of a single on-board Pentium P166. On top of the XR4000 are mounted the multimedia components and housing, consisting of a Pioneer laser-disc player, an Altec Lansing actively amplified sound system, and a NCR fifteen inch colour active matrix display. Finally, protected by the top housing at a height of 1.80m is a Toshiba KPD50 colour CCD camera. A Matrox Meteor frame-grabber interfaces this camera to the Pentium processor in the XR4000.

The overall power consumption of the entire Sage robot is 300 watts during active operation. On-board battery capacity is designed to comfortably allow for eight hours of activity with 25% charge remaining.

### Software Architecture

The software architecture consists of a low-level reactive system complemented with a higher-level planner. At the low level it has the ability to avoid obstacles and navigate. At the higher-level it has the ability to serve as an interactive "educator".

Obstacle avoidance is done using the ultra-sound sensors. It creates no internal representation of the obstacle-ridden environment. Every cycle, Sage computes the most free direction of travel, within a pre-specified window of the desired direction of travel, purely as a function of the current sensor readings. Translational speed is computed independently from the direction of travel.

Two concessions were made to surmount Sage's navigation challenge. First, modifications to the environment were allowed, in the form of visual landmarks. Three high-contrast pink rectangles, approximately 60 cm on a side, were installed on walls terminating three of the hallways travelled by Sage (see figure 2.7). Using a video camera able to track these landmarks, the robot was able to navigate these halls reliably. Second, the robot was limited to a predefined set of unidirectional, safe routes, so chosen because these routes are navigable using straight-line paths. Straight paths are followed during the motion control process by means of straightforward, functional algorithms, eliminating the need

for more complex motion planning and route following software.




Figure 2.7: Visual landmark situated on the end of a route.

**Robot-People Interaction**

Sage offered to the visitors the option to ask it to start a tour in the museum pressing a button located on the robot. During the tour, in each of the six stops programed in it, the robot Sage, generated a five minute long narrative. The script was narrated by a well-known public television personality. A variety of source material was used, including computer animation, stock footage of paleontologists, re-creations, sound effects and still images.

Thus, the way for the people to interact with the robot was using the buttons installed in it. They couldn't choose the place they wanted the robot to go, they could only choose to start a tour. The way to interact from the robot to the people was from voice.

In one rather common mode of interaction, museum visitors were often observed standing in the robot's path, attempting to either disorient the robot or frustrate its planned direction of travel. In some cases, individuals would even jump up to cover the CCD camera with their hands.

In order to counteract this tendency from the people, Sage was made an affective robot.

> "The idea was that a robot that expresses its goals in an emotive manner may garner greater respect from museum visitors. For instance, a robot that expresses frustration when its educational goals are blocked by pranks may be able to convince those pranksters to treat it as an educator and allow it to

> fulfil its mission. The goal was not to humanise the robot in order to confuse the distinction between machine and man; rather, the goal was to give Sage the affective tools to communicate successfully with humans, so that it would operate in an atmosphere more conducive to its educational mission."
>
> –Illah R. Nourbakhsh et al., An Affective Mobile Robot Educator with a Full-Time Job [81]

A mood transition scheme was implemented for Sage. Depending on the event happening and the task being performed by the robot, it would be in one concrete mood. The moods chosen were: "happy", "lonely", "tired", "frustrated", and "confused".

**Comments**

The Sage project is an interesting work for two main reasons: Sage operated in a museum for a long period, five months and a half, and all the processing was done using a single PC Pentium P166. Sage uses vision and ultra-sound sensors to navigate. Image processing can become very expensive in terms of computation, thus, having this limited computation power made the researches look for simple and effective vision processing algorithms. The use of landmarks gives the opportunity to simplify computation creating a topological representation of the environment, were the nodes correspond to the landmarks. This way, the use of metric maps and statistical techniques to localise the robot in the environment, which are computationally expensive, were avoided. But the use of artificial landmarks means that they must be installed and maintained in any environment the robot must work in. A more interesting approach would be to use marks intrinsic to the environment. There are many characteristic marks in a building, for example, doors, emergency exit panels [70], or changes in the floor (different colour or material).

The control code is one process, single threaded which it is the way to waste less computational power when this is limited. But, a multi-processed, parallel development, where the processes run independently but interact with each other, would be a preferred choice, to develop systems to perform progressively more complex tasks. This would help to localise failures more easily and when a process would fail, it would not be necessary to stop the whole system as happens in single process, single threaded applications.

In total Sage operated for one hundred and seventy-four days, it was operational for four thousand and eight hours, and moving, giving tours, for about one thousand hours. It can be deduced from these data that Sage was active each day for five and a half hours giving tours. From the reported data, It was not possible to find the average distance the robot travelled in each tour. Nor was it possible to find the average speed the robot moved at. These data would be interesting to have in order to evaluate the navigation efficiency of Sage. It was found that the average distance travelled by the robot was of 1.3 km, but not knowing the average number of tours the robot gave a day, it is useless for deducing the length of each tour. The distance travelled in each tour is a more significant value than the whole distance travelled in one day. It is more challenging for a robot to do some few long tours than many short tours, even if the total distance travelled is the same in both of the cases.

In one hundred and thirty-five of the days it operated error free. This gives a mean time to failure of four days. If the failure can generally be fixed relatively easily (restarting the system for example) this is an acceptable rate for an investigation project (but not for a product). It is not acceptable if the robot has to be stopped for the whole day. The failures reported were generally of the first type (they had to stop the robot for the whole day on seven occasions during the exhibition period). The nature of the failures was sometimes related to the hardware and sometimes related to the software. Some hardware components had to be replaced during the operation in the museum. Many problems came from the sonar sensor, which was reseting or stopping unpredictably. This meant everything had to be stopped and the robot had to be restarted as the application was designed to be one single process. Some important software bugs in the code were found, which caused failures, sometimes difficult to localise. The common experience was to add a new bug when solving the localised one. A distributed system would allow to localise failures easier. To repair them it would not be necessary to stop the whole system.

The robot did a five minute presentation at each of the six programmed stops. Only 4% of the visitors remained with the robot for the whole tour. Most of the visitors (74%) remained with the robot between five and fifteen minutes. Five minute long presentations are very long for each stop. Even for a person, it is difficult to engage people to follow five minute long explanations at each stop, and it it harder for a robot. Shorter explanations would help maintain the interest of the people for the whole tour.

The way for the visitors to interact with the robot was very limited. They could only choose to start a tour. Some common behaviour from the people towards the robot was observed to try to confuse the robot. This behaviour was reduced when the mood transition schema was installed. It is important to note that the goal when this affective behaviour was implemented was to make the robot able to perform its task more reliably and not to humanise the robot. But other techniques would be worth investigating, rather than a mood transition schema, which would prevent this kind of behaviour from the people to the robot, instead of reduce them.

### 2.2.2 Rhino and Minerva

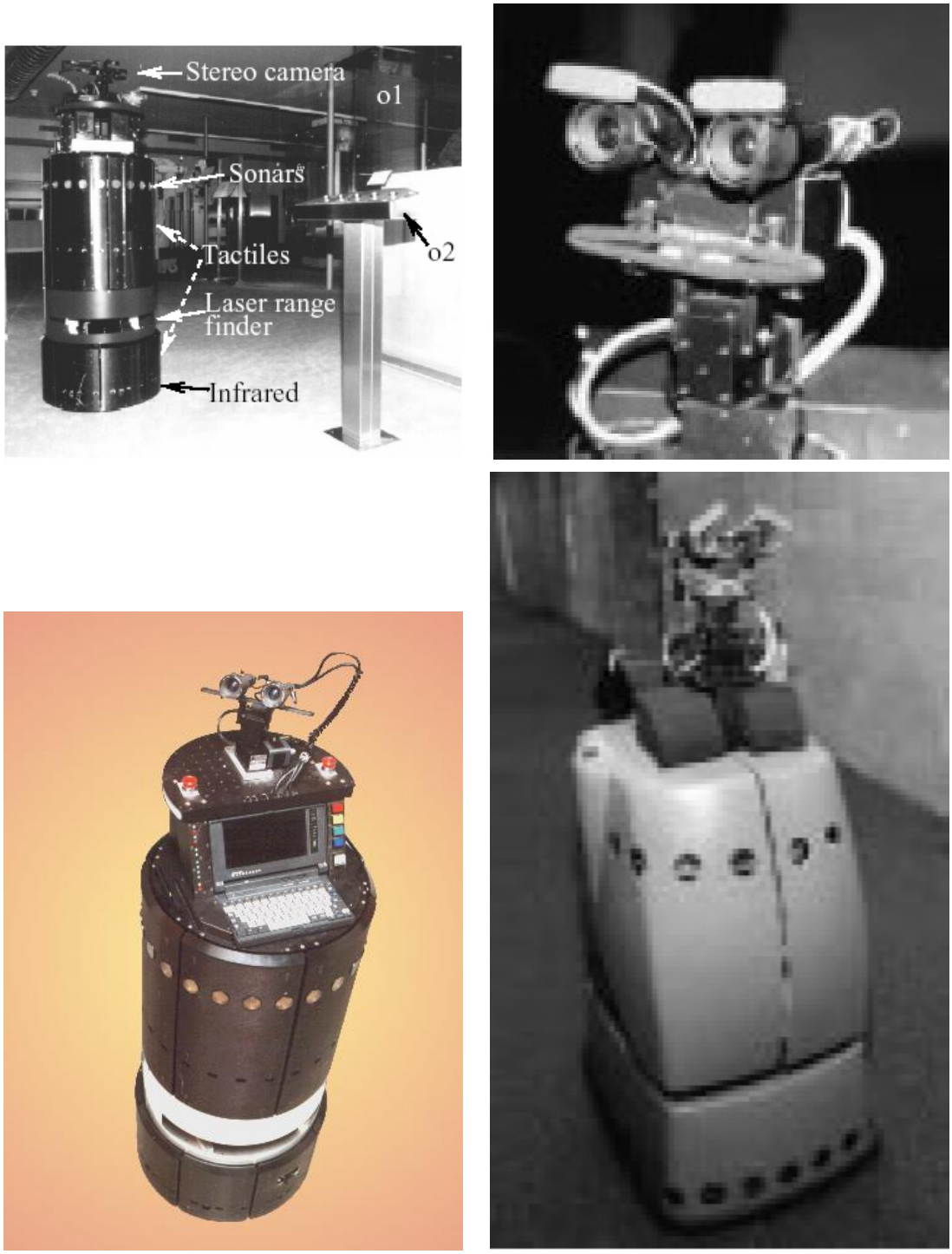



Figure 2.8: Robots Rhino on the left and Minerva on the right. The picture on the top left corner indicates the location of the different sensors of the robot Rhino. The picture on the top right corner, shows the robotic face installed for the robot Minerva.

Rhino and Minerva were built to assist and entertain people in public places, such as museums. Their primary task was to give interactive tours through an exhibition, providing explanations to the various exhibits along the way, using pre-prepared multi-media presentations. In May 1997, Rhino was deployed in the Deutsches Museum Bonn. During a six-day installation period the robot gave tours to the visitors. Through an interactive Web-Interface, people could also watch the robot's operation and even control its operation [92], [33]. Users could choose different targets for the robots from the Web page, by clicking in the maps shown in the Web pages. The robots made tours by going to the places chosen by the Web users. Figure 2.9 shows the Web-Interface used by the robot Minerva and figure 2.10 shows the Web-Interface used by the robot Rhino. Minerva was installed in the Smithsonian's National Museum of American History, for a period of two weeks, in the summer of 1998, in an exhibition hosted by Lemelson Center for Invention and Innovation.




Figure 2.9: Minerva's exclusive Web control interface. Users give target positions by clicking inside the map. The robot moves to targets on a first-come-first-serve manner.

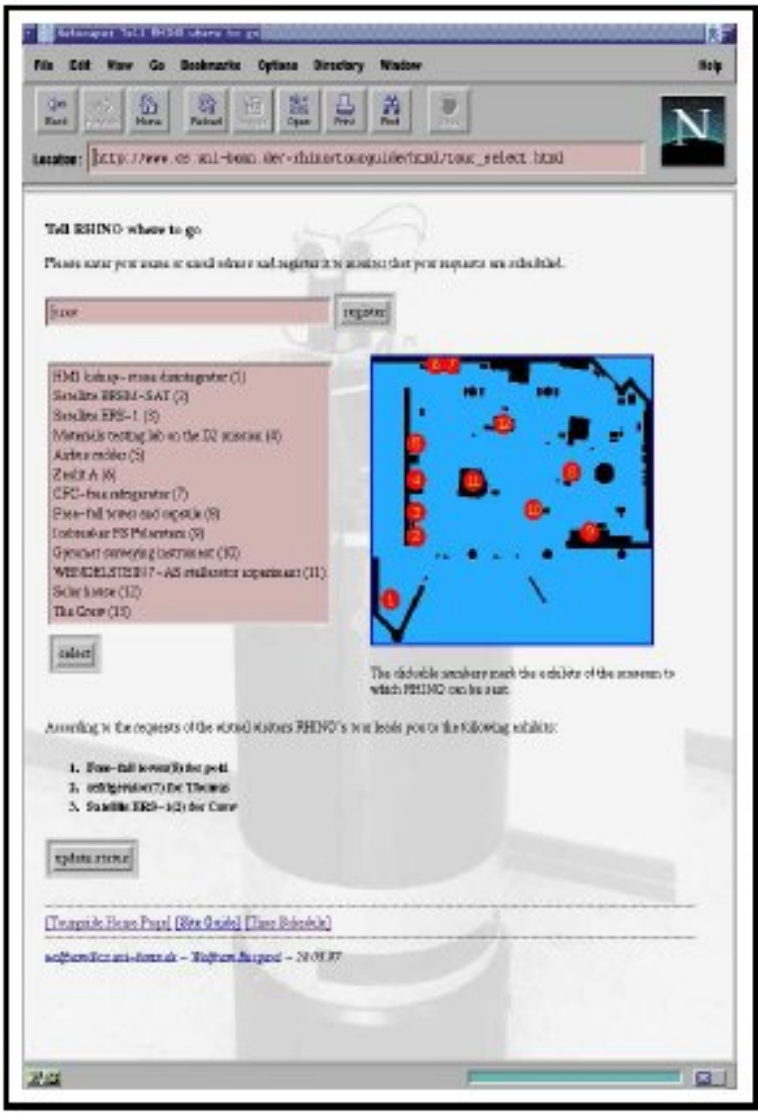



Figure 2.10: Control page of Rhino Web interface. Users can choose exhibits they want the robot to go to as well as viewpoints.

A face was used to provide a focal point for interaction (see figure 2.8). The robot is supplied with an "emotional" state, expressed outwardly by facial expressions and sounds. Adaptation occurs in one of the interaction tasks using a memory based learner [91].

### Hardware Architecture

Rhino was based upon a RWI B21 robot. It was a synchro-drive robot equipped with a stereo camera system and twenty-four ultra-sound transducers, two laser range finders, fifty-six tactile sensors and 32 IR sensors (see figure 2.8). Its maximum velocity is 90 cm/s. It is equipped with two on-board 486 PCs and a laptop that are connected via a

radio Ethernet link to several off-board SUN Sparc stations. Some of the processing was done off-board (it was not possible to find in the literature which part of the processing was done off-board). A second radio link communicated video images to a Datacube, a special purpose machine for processing images in real-time [104]. Thus, vision processing and one part of the processing related to navigation was done off-board.

Minerva was based on a RWI B-18 robot. It used two laser range finders which provided 360° field of view around the robot at 45 cm height, with an angular resolution of 1°[87].

The robot face (see figure 2.8) had four degrees of freedom which were implemented via servo motors controlled by a serial port interface. One degree of freedom was used to separately control each eyebrows and two degrees of freedom were used to control the mouth. The "eyes" of the robot were a pair of Sony XC-999 colour cameras. These cameras were not used for navigation or obstacle avoidance in the system, but were present for the sole purpose of transmitting a robot's eye view of the museum to Web visitors. The "lips" didn't move in synchronisation to the speech generated by the robot. Instead, a bar graph style LED display was mounted behind the mouth. The bars of the display illuminated in response to the speech generated by the robot. The head assembly was mounted on a Directed Perception PTU-46-17.5 pan-and-tilthead. This allowed the head to rotate approximately $90^o$ from the centre line of the robot and to tilt slightly from the horizontal [91].

#### Software Architecture

The overall software architecture for Rhino and Minerva consisted of approximately 25 modules (processes), which for the case of Rhino were executed in parallel on 3 on-board PCs and 2 off-board SUN workstations, connected via Ethernet (no information was found about how Minerva distributed these modules). The software modules communicated using TCX [55], a decentralised communication protocol for point-to-point socket communication. Figure 2.11 shows the overall software architecture along with the major software modules and the flow of information between them. The Rhino and Minerva systems were also organised in a hierarchical manner, with the device drivers at the lowest level and the user interfaces at the highest. The hierarchy, however, was not strict in the sense that modules would pass information only within the same or across adjacent layers. Modules often communicated across multiple layer boundaries.

#### Robot-People Interaction

Rhino and Minerva used a Human Natural Language to give explanations about the museum exhibits to the visitors. Visitors could interact with the robots, to choose the tour they wanted the robot to do for them, by pushing buttons on the robot. People could also interact with the robots via a Web page.

Minerva, had a robotic face installed on top of it. Similarly to the robot Sage (see previous section 2.2.1), affective states were defined, they were named "emotional states" by the developers of Minerva. But the use of the name "emotional states" doesn't suggest the

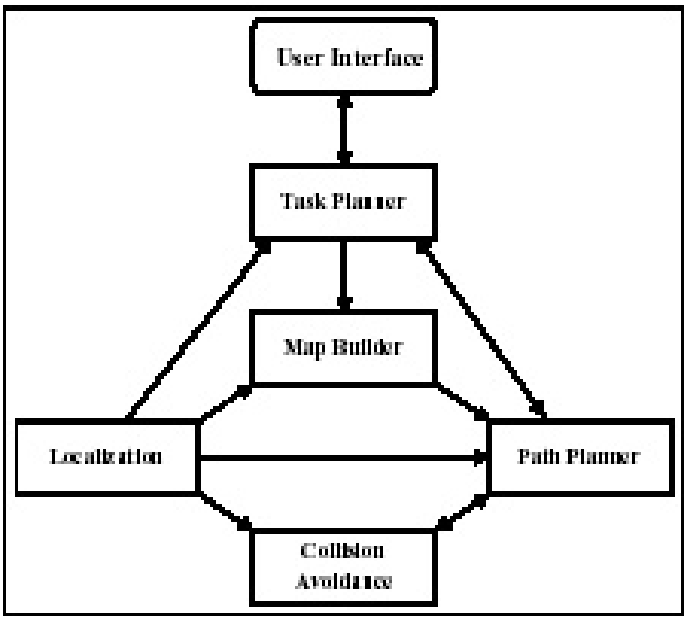



Figure 2.11: Software Architecture of Minerva and Rhino [33].

system has the property of "emotion", they use the term to indicate an emotional state that the person observing Minerva would impart to it. Minerva used its face, head direction, and voice to interact with people. A stochastic finite state automaton (with 4 states) was employed to model simple "emotional states" (moods), allowing the robot to communicate its intent to visitors in a social context with which they were already familiar (see [91]). Selected moods were "happy", "neutral", "sad" and "angry". The robot would change from one state to another depending on the persistence of the people who blocked the robot's path. Depending on its "motional state" the robot changed its facial expression and the timing and volume of speech. When "happy", Minerva smiled and politely asked for people to step out of the way; when "angry", its face frowned and its voice sounded angry.

### Comments

It is interesting that Rhino and Minerva have both operated in real environments such as the Deutsches Museum and Smithsonian's National Museum of American History. They did it in a less controlled environment than Sage, as they didn't use artificial landmarks to simplify the representation of the environment to a topological representation. Instead they had a metric map and use statistical methods to globally localise the robot with respect to this map. A big disadvantage of this approach is that these methods are computationally very expensive, and do not scale well with even moderate increases in the area that needs to be mapped: for a linear increase of the size of the map, the computational cost grows exponentially.

Regarding to the software architecture, it is made from different modules or processes

running in parallel. Using the decentralised communication protocol for point-to-point socket communication, TCX [55], makes it possible to share the modules between the different computers they use for doing the computation. This also makes failure detection easier.

The researchers which developed the architecture, claim modules are hierarchically organised, but their hierarchies are not strict as often modules communicate across multiple layer boundaries. Thus it is not clear this to be a real hierarchical organisation rather than a layered organisation.

Rhino and Minerva use a Human Natural Language to communicate from the robot to the visitors. They don't try to do speech recognition, as it is not reliable for uncontrolled environments such as museums.

An "emotional state" approach was used in the robot Minerva, to make the robot more capable to show its intentions so that people could get them faster. The intention was to exploit traditional social interactions between humans, to communicate intent during spontaneous, short-term interaction. These were thought to help the robot achieve its goal (such as to navigate from one place to another in places with people around) more effectively. In [91] a comparison is done with the robot Rhino which didn't use any approach of this type. One of the conclusions is that the average navigation speed of Minerva (38.8 cm/s) was faster than Rhino's (33.8 cm/s), even if they essentially used the same control and navigation software, and even if Minerva operated in a considerably more populated environment. This improvement was attributed to Minerva using the "emotional state" approach to communicate its intentions during spontaneous short-term interaction. It is not easy to know from these data (without seeing the performance of the two robots) if this improvement (5 cm/s) was really due to the mentioned "emotional state" approach, and not due to the difference in the geometry configuration of the environments the two robots operated in, or to the tours they were programed to give, or to the different hardware of the two robots. 5 cm/s in average is not a big difference and it may also depend on how the average speed was calculated for them. However, trying to imitate the human social spontaneous, short-term interaction is interesting and worth to be investigated towards a more reliable people-robot interaction. And it is not essential to use a Human Natural Language for it.

### 2.2.3 Hermes




Figure 2.12: The robot Hermes.

The robot HERMES was built by the Institute of Measurement Science / Intelligent Robots Laboratory at the Bundeswehr University of Munich.

Hermes was field-tested for 6 months (October 2001- April 2002) in the Heinz Nixdorf MuseumsForum (HNF) in Paderborn, Germany, the world's largest computer museum. It was exhibited in the special exhibition "Computer Brain" In which the HNF presented the current status of robotics and artificial intelligence and displayed some robots from international laboratories including Hermes [11],[10].

#### Hardware Architecture

Hermes had an omni-directional mobile base with four wheels, arranged on the centres of the sides of its base. The front and rear wheels were driven and actively steered; the lateral wheels were passive. The manipulator system consisted of two articulated arms with six degrees of freedom each on a body that could bend forward (130°) and backward(-90°). The work space extended up to 120 cm in front of the robot. Each arm was equipped with a two-finger gripper that was sufficient for basic grasp operations.

The main sensors were two video cameras mounted on independent pan-and-tilt drive units in addition to the pan-and-tilt unit that controlled the common "head" platform.

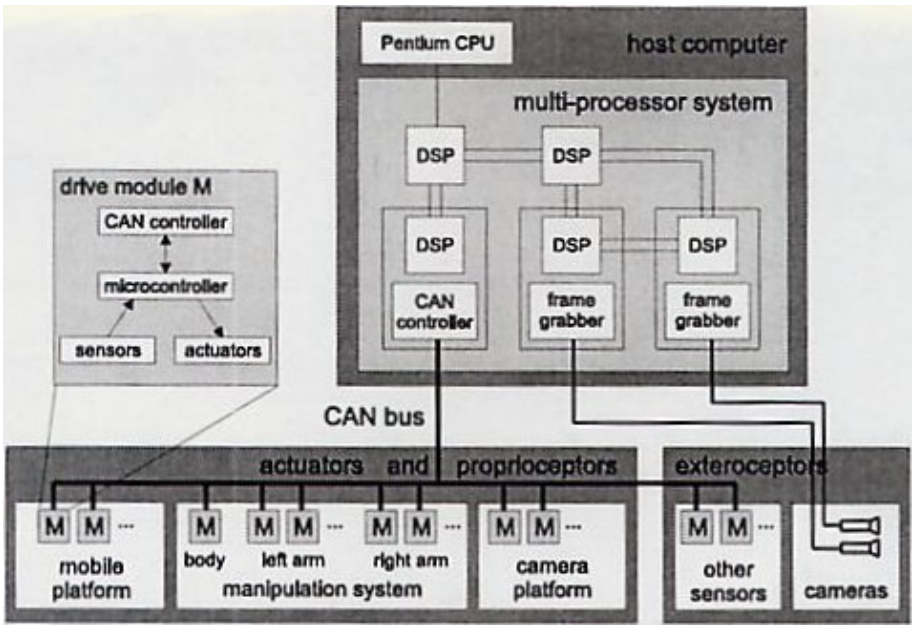



Figure 2.13: Hardware architecture of Hermes.

Hermes was built from twenty-five drive modules with identical electrical and similar mechanical interfaces yielding twenty-two degrees of freedom. Each module contained a motor, a Harmonic Drive gear, a micro-controller, power electronics, a communication interface and some sensors. The modules were connected to each other and to the main computer by a single bus.

A hierarchical multi-processor system was used for information processing and robot control. The control and monitoring of the individual drive models was performed by the sensors and controllers embedded in each model. The main computer was a network of digital signal processors (DSP, TMS 320C40) embedded in a ruggedized, but otherwise standard industrial PC. Sensor data processing (including vision), situation recognition, behaviour selection, and high-level motion control were performed by the DSPs, while the PC provided data storage, Internet connection and a user interface [13].

### Software Architecture

In its core the system was behaviour-based, however, to be able to select behaviours and to pursue long-term goals in addition to purely reactive behaviours, a situation-oriented deliberative component was introduced which was responsible for situation assessment and behaviour selection.

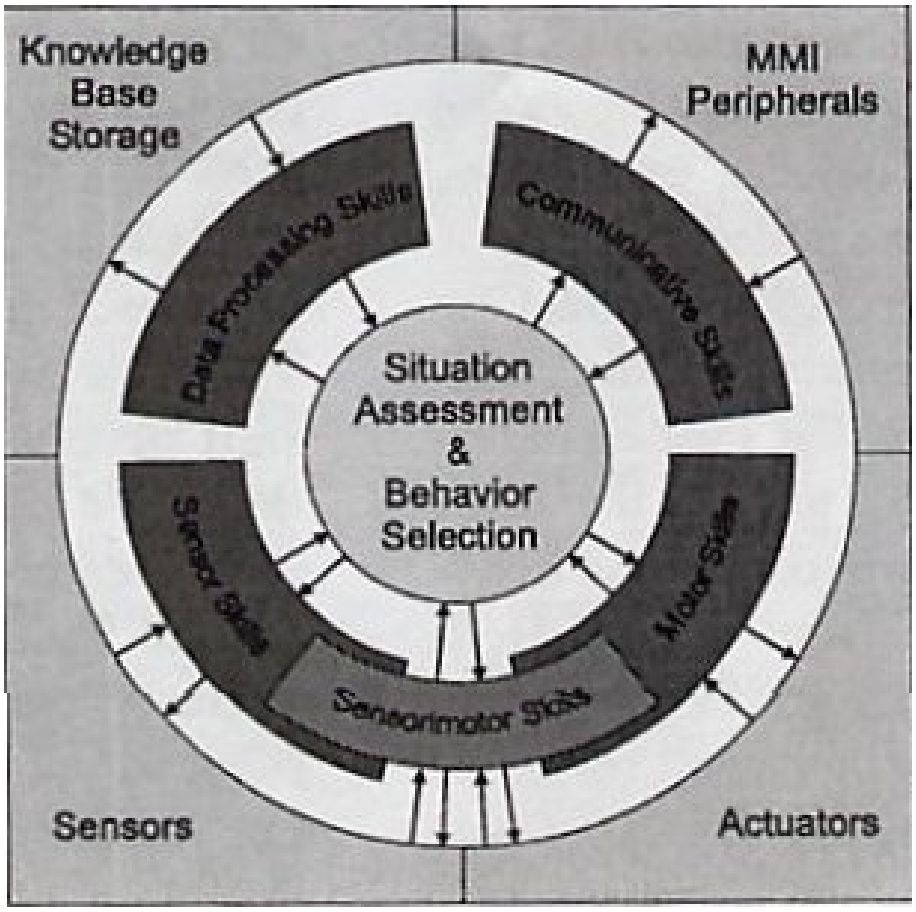



Figure 2.14: The software architecture of Hermes. A situation-oriented, behaviour-based robot architecture.

Figure 2.14 shows the basic elements of the situation-oriented, behaviour-based robot architecture. The situation module (situation assessment & behaviour selection) acted as the core of the whole system and was interfaced via "skills" in a bidirectional way with all other hardware components—sensors, actuators, knowledge base storage and people-machine, machine-machine interface peripherals.

These skills had direct access to the hardware components and, thus, actually realized behaviour primitives. They obtained certain information, for example, sensor readings, generate specific outputs, for example, arm movements or speech, or plan a route based on a topological map knowledge. Skills reported to the situation module via events and messages on a cyclic or interruptive basis to enable a continuous and timely situation update and error handling.

The situation module fused skills data and information from all system components to make situation assessment and behaviour selection possible. It provided general system management. Therefore, it was responsible for planning an appropriate behaviour sequence to reach a given goal, that is, it had to coordinate and initialise the in-built skills. By activating and deactivating skills, a management process within the situation module

realised the situation-dependent concatenation of elementary skills that leads to complex and elaborate robot behaviour [12].

### Robot-People Interaction

A speaker independent speech recognition for several languages (German, English and French) formed the basis for various kinds of robot-human interaction. Hermes used a commercially available speech recognition engine (Lernout & Hauspie). A special grammar was designed to enable natural language understanding with limited computational resources. The fewer the number of words in its vocabulary and the stricter the syntax the better the results of the speech recognition as fewer hypotheses have to be verified. Thus Hermes varies the words to be listening to depending on the dialog, for example, if it did a question which answer should be "yes" or "no", it restrict the number of words to be able to recognise to these two words. It provides a wireless microphone to be used by the humans to send their utterances if the ambient noise is high [14].

### Comments

The fact that Hermes was built from drive modules with identical electrical and similar mechanical interfaces, each with its microprocessor, motor, sensors and actuators, simplifies the task of replacing them when they are broken. Thus the maintenance of the robot is easier as if a failure occurs it is enough to locate the module not working and replace it.

Regarding to the software architecture, The control of the robot is behaviour based, thus, it doesn't use the classic model-based approach. There are not any numerical models of the kinematics and dynamics of the robot and of the external objects that the robot should interact with, nor are any quantitative sensor models used as the basis for controlling the robot's motions. This allows the robot not have to depend on having accurate quantitative agreement between reality and the models.

The way how the system can be developed is by adding "skills". This offers the possibility to add in a quite straight forward way new "skills" to the robot to make it capable to perform different tasks. It is not mentioned in the literature if these skills can get advantage from or help each other. Nothing was found that says that they do. This property would be desirable to not to have to build each "skill" from scratch. The "skills" are controlled by a Situation Assessment and Behaviour Selection object, which selects which of them to use at each moment. The "skills" are centralised in this object. It would be nice to be able to get rid of this central object, and, for example, not have to reconfigure it when a new "skill" is added.

Hermes uses a Human Natural Language recognition system. A wireless microphone had to be used by the speaker if the ambient noise is high. It is claimed in [14] that under normal conditions the microphone could be placed on the robot. It would be interesting to know how are these normal conditions. A special grammar was created for the robot, thus it is not really true it can recognise speech in three languages. It recognises some kinds of constructs in each of the three languages.

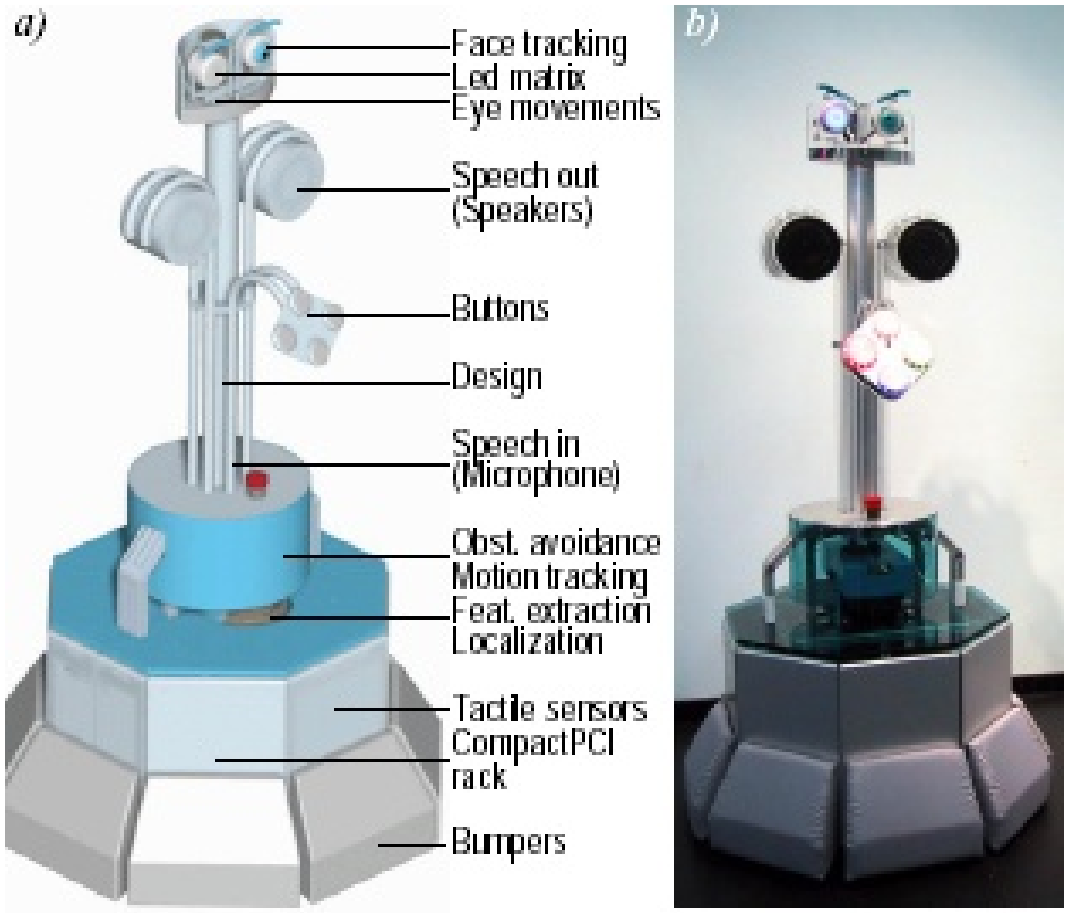



Figure 2.15: a) Main components of RoboX b) A picture of RoboX.

### 2.2.4 RoboX

RoboX is a mobile robot designed to operate in large exhibitions. It was built based on the experience of the Autonomous Systems Laboratory at Swiss Federal Institute of Technology Lausanne (EPFL). The production of 11 of these machines was done by a spin-off of the laboratory: BlueBotics SA. They were built with the goal to maximise the autonomy and interactivity of the mobile platform while ensuring high robustness, security and performance [105].

Eleven RoboX robots operated for one hundred and fifty nine days (about five months) from May 15 to October 20, 2002 at the Robotics Pavilion of the Swiss National Exhibition Expo 02. They were developed to guide people to the different exhibits presented at the Expo 02.

During its operation, the 11 RoboX operated guiding visitors through the Robotics Pavilion at the Swiss National Exhibition Expo 02, up to 12 hours per day, seven days a week. Throughout this period, 686,000 people had the opportunity to be in contact with one of the RoboX robots.

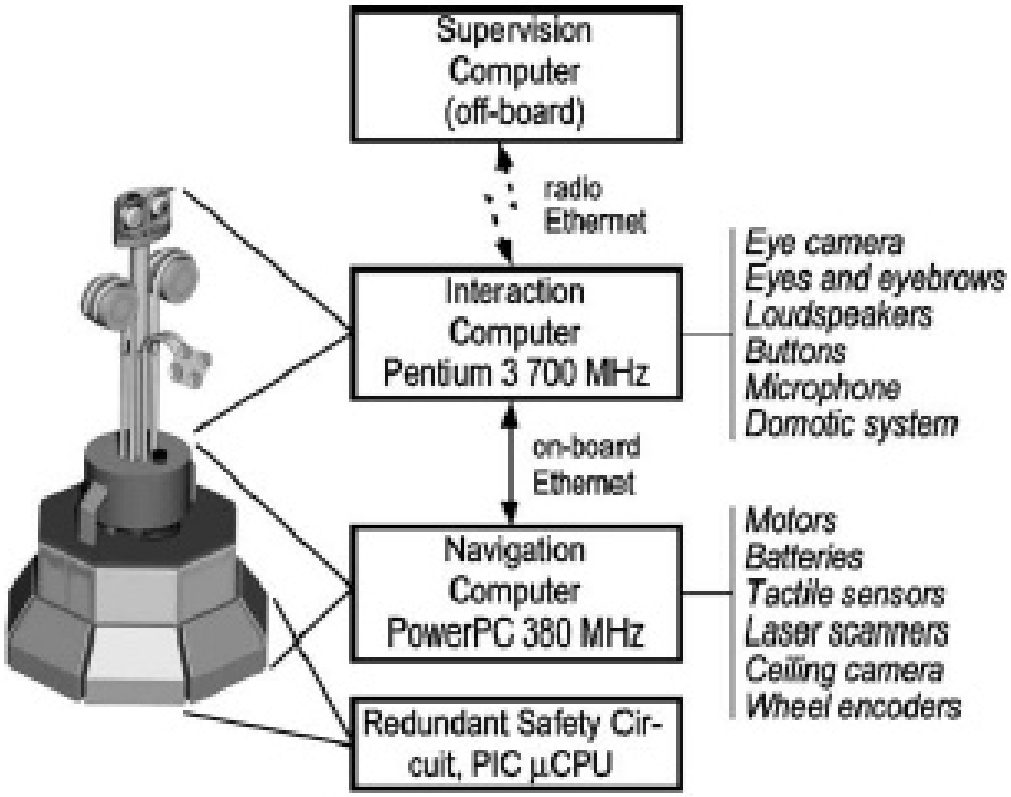



Figure 2.16: a) Hardware Architecture of RoboX.

The basic functionalities of the exhibition robot RoboX are shown in figure 2.15.

### Hardware Architecture

The RoboX hardware consists of three main components: mobility, interactivity and safety (figure 2.16). The base of the robot contains a Compact-PCI controller, two laser range finders (Sick LMS 200), the drive motors, a redundant safety circuit (see figure 2.16), the batteries, and bumpers that contain tactile sensors. Each RoboX was motorised by two differentially actuated wheels on the middle axis using 1:50 harmonic drives and a castor on the front and rear, one of them on a spring suspension. This gave each RoboX a good manoeuvrability and stability in spite of its height of 1.65 m. The battery pack made the robot able to operate for around 12 hours and required 12 hours for re-charging.

Each RoboX face included two eyes with independently actuated pan-and-tilt units and mechanically coupled eyebrows. The right eye was equipped with a Firewire colour camera and left eye contained a LED matrix.

Each RoboX was equipped with a PowerPC 750 (G3) at 380 MHz. The PowerPC interfaced the two laser range finders, a camera looking at the ceiling (via a Bt848-based frame grabber), the tactile sensors, the safety circuit and the drive motors/encoders (figure 2.16).

It communicated with an on-board PC (Pentium III Compact-PCI board at 700MHz) through the on-board Ethernet. The PC had access to the Firewire eye camera, the controller for eye pan-and-tilts and eyebrows, the input buttons, the LED matrix, the microphone, and the two loudspeakers. Via a domotic system, the PC could furthermore interact with elements in the environment: switch on and off light spots, trigger flashes, start demonstrations or control other exhibits via infra-red (no information was found about how is this done exactly). The robot was linked via radio Ethernet with an off-board PC that served as a supervision and logging tool only. It displayed important state variables of the robot at any time and stored relevant log information on a daily basis. However, each RoboX did not rely on this connection because it was always running from its on-board computers.

#### Software Architecture

Safety and time-critical processes were assigned to the PowerPC running the hard real-time operating system XO/2 [24]. Objects requiring commercial, off-the-shelf components were implemented on the on-board PC running Windows operating system. The resulting software architecture is shown in figure 2.16. Watchdogs were implemented for the speed controller, the obstacle avoidance, the bumpers, and the laser scanner driver. Failure of one of these tasks was detected by the security controller, which attempted to restart the crashed task or, if the restart failed, stopped the robot and sent an e-mail to the operator.

#### Robot-People Interaction

The Firewire colour camera was used for people tracking and the LED matrix to display symbols and icons. A speech synthesiser enabled the robot to speak in one of four required languages. The central input device enabling a bi-directional communication with the visitors were four buttons which allowed people to select the language and respond to questions the robot asked. Two of the 11 RoboX were additionally equipped with a directional microphone matrix for speech recognition. As said before, speaker-independent speech recognition in four languages in a museum exhibition-like context is not possible today. On-site experiments demonstrated the feasibility of recognising simple words like yes and no in each of four languages.

#### Comments

Each RoboX was able to operate other systems in the environment, such as switch on and off spot lights, trigger flashes, start demonstrations or control other exhibits via infra-red. It could be understood as part of a domotic system. Using a specific communication protocol, and communication means (infra-red for example) it can communicate with them and control them. Communication from the robot to the people is done via synthesised speech, communication from persons to the robot is done via a number of buttons installed at the robot. Speech recognition is hard in un-controlled environments, as an exhibition room might be. An interesting approach would be to use the same communication way in

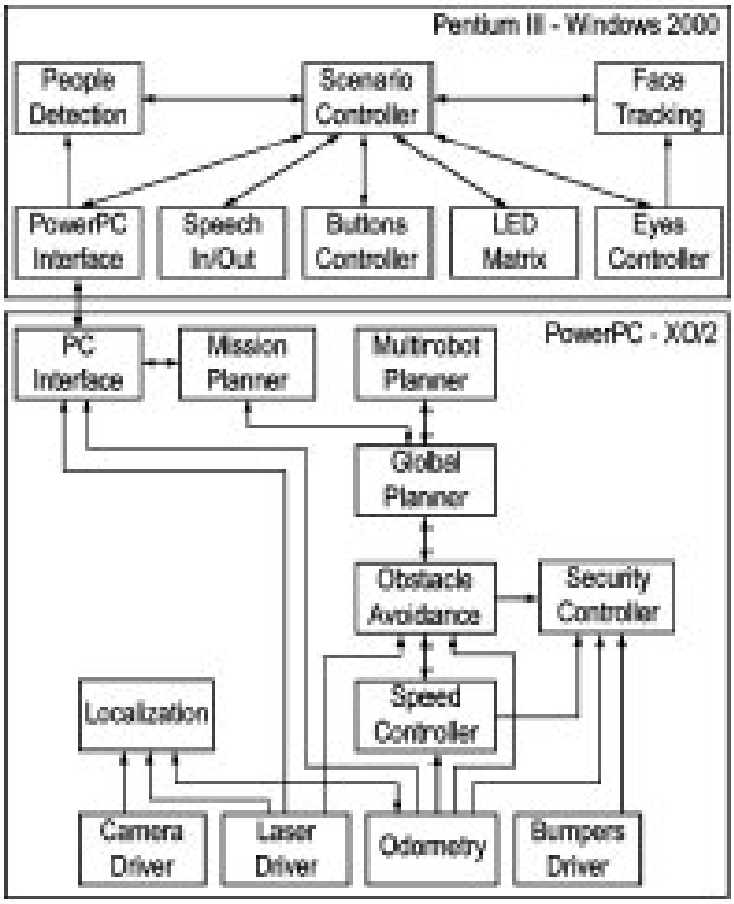



Figure 2.17: RoboX. Software Architecture.

every case.

With respect to the software architecture, it is separated in different modules which make it possible to develop different aspects separately. They are separated in two computers running two different operating systems. The modules in one computer need to communicate with the modules in the other computer. This is done through one module. It would be nicer if any component could communicate with any other component regardless of which operating system it is running in, without the need of an interface object. This would make the system more robust as all the communication would not have to depend on this object.

The large majority of failures were caused by software problems. In the first three weeks of operation the failure average was of more than one failure per hour, per robot, which was reduced after the first month of operation to become of one failure each eight hours, that's to say, about one failure a day per robot. Most of the software problems happened in the interaction system (PC running Window 2000), which were not considered safety critical. Navigation system (PowerPC running real-time system OX/2) had at the beginning a failure average of 1 failure between 20-40 hour and ended up with an average of one failure every 60 hours. This average is very good, again, for a investigation project, not for a commercial product. No serious hardware problems are reported, only some problems caused by the amplifiers of the motors. Not having hardware problems is very important for a robot. The simpler the hardware the lower the probability for something to fail.

The RoboX robots didn't use speech recognition, but they could generate speech in four languages. It is explained in [97] that due to the large number of visitors and robots sharing the exhibition hall, it was sometimes difficult to understand the artificial voice of the robot. To recognise human voice would not be possible in such conditions for any robot.

Regarding to the acceptance from the visitors, it is reported that some of them were disappointed about the performance and intelligence of the RoboX robots. This is a common reaction from people whose knowledge about robots come from the Science Fiction. Most of the visitors described the robot as amusing, and pleasant and the functionality of the four buttons was easily learned by them.

## 2.3 Critical Evaluation

The first part of this chapter, shows the approaches being taken towards robots that are like humans. As said before, it is not clear our goal should be to build robots that can be confused with humans, to build robots identical to humans so that when we are talking to one we don't really know if we are talking to a person or to a robot. Is that how we really want the robots to be like?

Each of the parts and the shapes of the human body have a practical function. The shape of the nose, the eyebrows, the hair, etc, all have a practical function. The parts and the shapes of the body of any living creature have a practical function. The robots should also be built with this philosophy. The discussion about what is practical for a robot arises here. As said before in this chapter, it is not clear that making them more human like, makes the interaction with them better, but it makes the design and maintenance of them a lot more complicated. Thus when designing and building two legs or a face for a robot, a question should be placed: does the robot really need two legs and a face for the task it is required to do?

In the second part some concrete approaches have been analysed. It was possible in the case of the robot Sage to get information about the main problems the robot had during the operation time. It can be seen that hardware problems are an important handicap for service robot developments. Competitors in the Mobile Robot Competition and Exhibition, also suffer from these problems, as it is reported in [101]. Some information was also found for the RoboX robots, which claim they didn't have many hardware problems. This kind of information is missed in the literature and it would be very useful to help to get an idea about which are the general problems a robot may have.

The software architectures used for the four robots presented in the second part, make them able to do many tasks, but some limitations have been identified in each of the cases. It is not easy to find detailed information about the software architecture of the robot, in the literature. The software architecture found most interesting is the one used for the robots Rhino and Minerva which is formed by a network of independent modules.. But it was not possible to find information about the cost of extending it. For example, information about which is the cost of adding a new module, and how does this affect the rest of the modules. Nor it was clear how hierarchies were defined between them. It is worth to investigate an approach formed by independent processes, modules, or agents which communicate between each other to help each other, and were adding a new element doesn't require to apply any changes to the already implemented ones.

The four approaches synthesise human voice. But only Hermes uses speech recognition. It is claimed that it works reliably in normal noise conditions. As said before, it would be necessary to define what normal conditions exactly are. In the case of Hermes, and also in most of the other approaches which do speech recognition, they define a certain number and kind of language constructions for the robot or the machine to be able to recognise them. In other words, the robot can't recognise any kind of phrase said in a Human Natural Language but a sub-group of phrases. Other communication methods are worth to be investigated, which use sound, but not speech.

# Part II

# A Society of Agents and A Musical Language

# Chapter 3

# A Society of Agents

This chapter presents the solution developed to help to achieve the goal one defined in section 1.2 of the Introduction chapter: A simple and extensible way to help to put together all the different aspects involved in the operation of a service robot of type II, which facilitates the incremental development of new capabilities taking the maximum advantage of the capabilities already implemented.

The solution has been investigated as an agent network. Ideas explained by Marvin Minsky in his book *The Society Of Mind* [74] have been used to define a way to build and grow agent networks to achieve the goal mentioned above. In *The Society Of Mind*, Minsky tries to explain how human mind may work. Such an agent network has been named a *Society of Agents* in reference to *The Society Of Mind* book.

In a *Society of Agents*, each agent must contribute some particular elemental skills to the collection of agents as a whole. And any agent can take advantage of any skills of any of the other agents. All the agents are of the same level of importance and there isn't any centralised entity to control them. This makes the *Societies of Agents* to be fully extensible, where, adding a new agent does not require any changes for the other agents already in the society. It allows different people to add different skills to the system in different programing languages. They just have to agree on the communication protocol.

This chapter proposes a way to create agents and to put them in societies so that, by exploiting each other, they make the whole more capable.

Section 3.1 presents the most relevant ideas drawn from *The Society Of Mind*, which contributed to design a proposal about a way to build *Societies of Agents*. It also makes some references to *Descartes' Error* [44] by the neurologist Antonio Damasio, which also tries to explain how human mind may work. These ideas are on the same path of Minsky's and also contributed in the design of a proposal about a way to build *Societies of Agents*.

Section 3.2 presents the details of this proposal. It explains how agents should be like and how they interact between each other to form societies.

Section 3.3 explains the software tools built to test and implement *Societies of Agents* for this thesis work.

## 3.1 The Society of Mind

Marvin Minsky received the BA and PhD (Doctor of Philosophy) in mathematics at Harvard and Princeton. He is considered to be one of the fathers of Artificial Intelligence and he is Co-founder of the Artificial Intelligence Laboratory at MIT (Massachusetts Institute of Technology, now the Computer Science and Artificial Intelligence Laboratory (CSAIL)). In his book *The Society Of Mind*, Minsky offers a collage of some ideas about how the human mind may work and some notions and algorithms about how to implement them artificially. It is composed by thirty chapters with sections of one page each. In some cases there is not a clear connection between chapters, even between sections. There are no concrete solutions to all the questions it presents, neither is there a concrete way explained to implement the theories and ideas presented. This makes the reader feel free to use his or her imagination.

Minsky, while introducing the idea of *frames*[1], explains what happened when he described, not in high detail, this idea in 1974:

> "I first conceived the idea of frames in the early 1970s, while working on making a robot that could see, and I described the theory in a 1974 essay entitled "A Framework for Representing Knowledge." The essay influenced the next decade of research on Artificial Intelligence, despite the fact that most readers complained that its explanations were too vague. In retrospect, it seems those explanations were at just the right level-bands of detail to meet the needs of that time, which is why the essay had the effect it did. If the theory had been any vaguer, it would have been ignored, but if it had been described in more detail, other scientists might have "tested" it, instead of contributing their own ideas. Then they might have found my proposals were inadequate. Instead, many versions were suggested by other people, and "frame-based" programming became popular."
>
> —*The Society Of Mind*– 25.6 THE FRAME IDEA pp. 259

The way the theories are presented in the book looks for a similar effect in its readers. This has been taken as a license, and some ideas presented in *The Society Of Mind* has been used as a source of inspiration to create a particular version of a *Society of Agents*, which it is presented and evaluated through the investigation presented here.

Antonio Damasio is the M. W. Allen Professor of Neurology and head of the Department of Neurology at the University of Iowa College of Medicine. He is also adjunct professor at the Salk Institute for Biological Studies, La Jolla, California. In his book *Descartes' Error* he explains, from his neurologist point of view, and based on the results of many experiments done with people with damage in certain parts of their brain, how the human mind may work.

It is very interesting to contrast the ideas from both of these authors (a mathematician and a neurologist) about how the human mind may work, to see they share a similar point of view. This section has been designed to collect these ideas that served as inspiration

[1] Frames are not explained in this thesis. See chapters 21-26 of *The Society Of Mind*.

to develop a proposal (presented in the next section 3.2) about how to build *Societies of Agents*, in order for them to serve to achieve the goal one, referred to at the beginning of this chapter: find a way to put together all the different aspects involved in the operation of a service robot of type II.

### 3.1.1 Mind Separated

*The Society Of Mind* in Minsky's words,

> "... tries to explain how minds work. How can intelligence emerge from non-intelligence? To answer that, we'll show that you can build a mind from many little parts, each mindless by itself."
>
> — 1 Prologue pp. 17

... and he continues by defining these little parts as follows:

> " I'll call "Society of Mind" this scheme in which each mind is made of many smaller processes. These we'll call agents. Each mental agent by itself can only do some simple thing that needs no mind or thought at all. Yet when we join these agents in societies—in certain very special ways—this leads to true intelligence. "
>
> — 1 Prologue pp. 17

The word agent in this thesis, is used with the meaning defined by Minsky. The agents that will be presented in the next chapters are of the level of those considered the simplest in Minsky's *The Society Of Mind*.

> "But we should not try to find a close analogy between the low-level agents of a single mind and the members of a human community. Those tiny mental agents simply cannot know enough to be able to negotiate with one another or to find effective ways to adjust to each other's interference."
>
> —3.2 Noncompromise pp. 33

The agents presented in this thesis, are small processes which can communicate with each other, but the communication between them is very limited, they only send and receive data, they don't negotiate with one another nor do they look for effective ways to adjust to each other's interference.

Minsky also makes clear the sense in which he uses the word "society":

> " Society(1.1): In this book, an organisation of parts of a mind. I reserved the term "community" for referring to organizations of people because I did not want to suggest that a human mind resembles a human community in any particular way."
>
> —Glossary and Bibliography pp. 331

In this thesis the word society is used to refer to an organisation of agents. It doesn't want to suggest that the proposed agent organisations resemble any human community.

Even with limited interaction capabilities they are organised in societies. A big (may be the biggest) strength of any society is that their components can be, and usually are, specialised agents, which share resources. They can exploit other's capabilities to get the task they are specialised in done, without knowing how the others work. It is possible to find an example in our, more complex, human society: if our car needs to be repaired, we usually take it to the mechanic for him or her to repair. We don't usually learn how to do this ourselves. Minsky says:

> "... the best each can do is learn to exploit what the others can do, without understanding how they do it."
>
> —16.6 Motivation pp. 168

> "Achieving a goal by exploiting the abilities of other agencies might seem a shabby substitute for knowing how to do the work oneself. Yet this is the very source of the power of societies."
>
> —16.7 Exploitation pp. 169

Antonio Damasio in his book *Descartes' Error* also gives a view of the mind as formed from different parts, not controlled or supervised by a single process:

> "Although we have the illusion that everything comes together in a single anatomical theater, recent evidence suggests that it does not. Probably the relative simultaneity of activity at different sites binds the separate parts of the mind together."
>
> —A mysterious alliance pp. 84

> "If activity occurs in anatomically separate brain regions, but if it does so within approximately the same window of time, it is still possible to link the parts behind the scenes, as it were, and create the impression that it all happens in the same place."
>
> —An integrated mind from parcellated activity pp. 95

> "... time binding requires powerful and effective mechanisms of attention and working memory, and nature seems to have agreed to provide them."
>
> —An integrated mind from parcellated activity pp. 96

Damasio suggests it is time that gives us the impression of unity, this impression of having a single entity which controls the whole activity of our minds. He suggests this single entity doesn't exist, that it is an illusion built by the interaction of the activity in different parts of the mind.

From these ideas, a suggestion can be formed for a system formed from agents, each doing a specific task. They are organised in a society, but it can't be compared to a human society. The agents in this society have very limited means of interaction. But they must be able to exploit other's abilities in order to achieve their own goal. As what each agent does, influences the system, the whole system's behaviour depends upon what each agent is doing in each moment.

### 3.1.2 *A-brain* and *B-brain*

In *The Society Of Mind*, the agents are classified into two main types: the agents interacting with the real world through sensors and actuators, and the agents interacting with no direct connections with the real world but with other agents. The agents with direct connections to the real world are presented as part of an *A-brain* while the rest are presented as part of a *B-brain*:

> "Connect the *A-brain*'s inputs and outputs to the real world—so it can sense what happens there. But don't connect the B-brain to the outer world at all; instead, connect it so that the A-brain is the B-brain's world!"
>
> —6.4 *B-brain*s pp. 59

Thus, the *B-brain* is formed by:

> "Any part of the brain connected not to the outside world, but only to another part of the same brain. Like a manager, a *B-brain* can supervise an *A-brain* without understanding either how the *A-brain* works or the problems with which the *A-brain* is involved—for example, by recognizing patterns of activity that indicate the *A-brain* is confused, wasting time in repetitive activity, or focused on an unproductive level of detail."
>
> —Glossary And Bibliography pp. 326

An *A-brain* consists of the agents in direct contact with the real world and the rest of the agents, which are in contact with other agents, form a *B-brain*.

### 3.1.3 Conflict Handling

In a society where their components exploit the abilities of the other components, conflicts may arise. A conflict may happen when two agents try to exploit a third one to do two different things at the same time. For example, if there is an agent which adds to the system (a mobile robot for example) the ability to navigate to different places, and two different agents tell it to navigate to two different places at the same time, or if one agent tells it to navigate to one place and before the robot reaches the requested place another agent tells it to navigate to a different place, if no mechanism has been selected to handle the situations a conflict occurs.

Minsky presents the *Principle of Noncompromise* and cross-exclusion arrangements to handle these situations.

> "The *Principle of Noncompromise*: The longer an internal conflict persists among an agent's subordinates, the weaker becomes that agent's status among its own competitors. If such internal problems aren't settled soon, other agents will take control and the agents formerly involved will be "dismissed.""
>
> —3.2 NONCOMPROMISE pp. 33

Cross-exclusion arrangements could provide a basis for the *Principle of Noncompromise* in regions of the brain where competitive mental agents lie close together. Cross exclusion is explained as:

> "An arrangement in which each of several agents is connected so as to inhibit all the others—so that only one of them can remain active at a time."
>
> —GLOSSARY pp. 327

These ideas are interesting when implementing a society of competitive agents. Even if the proposal about how to build *Societies of Agents* explained in the next section has been inspired by *The Society Of Mind*, none of these ideas have been used to handle conflicts between the agents in the society. Instead the goal has been these agents not to be competitive but fair, and thus, build societies of fair agents.

## 3.2 A *Society of Agents*

The contribution of each agent to the society is to add one skill.

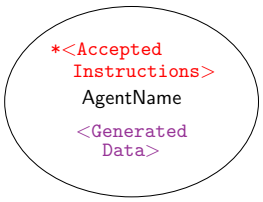



Figure 3.1: An agent adds one skill, it may accept some kinds of instructions about how to perform the skill, and it may generate some kinds of data. It may also need to send instructions and get data from other agents.

To achieve its goal, an agent may need to interact with other agents. It may need data from others and it may also need to instruct others. Two interaction mechanisms have been defined:

- *Request/Serve Data*.
- *Send/Receive Instructions*.

### 3.2.1 Request/Serve Data

An agent may generate data as output and may require data as input, which may be data generated as output by other agents. Thus each agent must know where to find its own required data. An agent serves data to any agent requesting it and acts as a client for any data it gets from other agents. When one agent connects as a client to another agent and requests data, the server agent sends them to it. While the connection is open, when ever the data change, the server agent send the new data to all the client agents. Each agent admits any number of client connections.

### 3.2.2 Send/Receive Instructions

An agent may accept some kinds of instructions from other agents, which may influence the way it performs its skill. It may also need to send some instructions to others.

### 3.2.3 An Example

An example to illustrate these agent society mechanisms has been found in the home environment. The TV could be considered as an agent. It performs the skill of processing the TV signal to show an image with sound. We use the remote control to send the TV some instructions about how to perform its skill. It accepts many types of instructions, such as the channel to tune to, or the volume with which to generate the sound.

In many homes, the telephone is placed in the sitting room, were also the TV is placed. A telephone could be thought which have an infra-red emitter and receptor. It might be programed to automatically turn the volume of the TV down when it receives an incoming call, and after the conversation finishes, or after the ring ceases in case nobody picks it up, restore the volume of the TV. Thus, the TV, should generate at least one data type: the level of the volume of the sound it is generating. It would be requested by the telephone, thus when ever the sound volume of the TV would change, the TV would send the new value to the telephone via infra-red (*Request/Serve Data* interaction type). When the phone would start to ring, it would store the value of the volume and send an instruction to the TV to generate sound at a lower volume (*Send/Receive Instructions* interaction type). When the call would be finished or the ringing ceased (in case nobody picked the telephone up), it would send again an instruction to restore the volume.

This example is useful to illustrate the idea of the *Society of Agents* and the kind of communication the agents require from each other. The amount of transmitted data is low and usually not done very frequently.

The *Societies of Agents* presented in this thesis are societies formed from software agents which communicate using the TCP/IP (Transmission Control Protocol/ Internet Protocol). To find an exact analogy with the example previously explained and the *Societies of Agents* presented here, three software agents will be presented: TV, Telephone, and RemoteControl, which control the TV, the telephone and the remote control respectively. They will be used in this chapter to illustrate the characteristics of a *Society of Agents*.

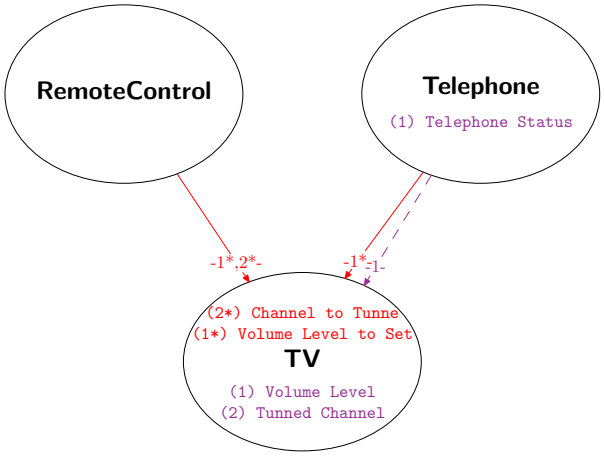



Figure 3.2: The names written in the middle of the circles, are names of the agents. A lower case name with an '*' above the agent name, indicates the type of instructions the agent accepts. A lower case name without an '*' below the agent name, indicates the output data generated by the agent. Continuous arrows indicate that the agent at the source instructs the agent at the target; the label of the arrows indicate the type of the instruction it is sending. Dashed arrows indicate that the agent at the source requests data from the agent at the target; the label of the arrow indicates which output data are requested.

The TV agent accepts instructions of the type `Channel to Tune` and `Volume Level to Set`. When it gets an instruction of type `Channel to Tune` it tunes the TV to the channel specified by the instruction, and when it gets an instruction of the type `Volume Level to Set` it sets the volume of the TV to the value specified by the instruction. It generates data of the type `Tuned Channel` and `Volume Level`, making them available for any agent requiring them. The `Tuned Channel` data indicates what channel the TV is tuned to and the `Volume Level` indicates the volume the TV is generating sound in. The Telephone agent generates the `Telephone Status` data which indicates if the telephone is in use or not. This data is not requested by the agents involved in this example. The Telephone agent requests the `Volume Level` data from the TV agent (this is represented in figure 3.2 by a dashed arrow with label 1), thus, the TV agent sends its `Volume Level` data to the Telephone agent, when ever this changes. The Telephone agent sends instructions of the type `Volume Level to Set` to the TV agent when it requires the volume of the TV to be turned down or up (this is represented in figure 3.2 by a continuous arrow with label 1*). The RemoteControl agent doesn't accept any instructions and doesn't generate any data type. It doesn't request any data from the TV agent and it sends instructions of the type `Volume Level to Set` and `Channel to Tune`, to the TV agent, when a button of the remote control is pushed (this is represented in figure 3.2 with a continuous arrow with labels 1*,2*).

### 3.2.4 Conflict Handling: Priority Levels

From the time an agent can be instructed by other agents, there is a possibility of conflicts to arise, since two agents may instruct it to perform the skill in two different ways at the same time. In many cases an agent needs the effect of its instruction to be maintained for a certain amount of time.

Continuing with the previous example, the telephone may receive an incoming call and someone may take the call, so the TV agent gets an instruction from the Telephone agent to generate sound in a lower volume. While one person is talking from the phone, another person may try to turn the TV up using the remote control. A conflict thus occurs, as two agents need the TV agent to do two different things at the same time. The Telephone agent needs it to maintain its volume low until the call finishes and the RemoteControl agent (the person using the remote control) needs the TV to be turned up instantly.

To solve this, five priority levels were created to be assigned to the instructions. Level one is the level with the lowest priority and level five is the level with the highest priority. An agent can lock in a specific priority level the instruction type accepted by another agent, so that this agent will only accept another instruction from another agent if it comes with a higher priority level, until it is unlocked again. If a higher level gets locked, the instructions in the locked lower level are ignored until the higher level is unlocked. Every time an agent sends an instruction, the instruction receiver agent responds indicating if the instruction has been accepted or not. If it is a lock request, the instruction receiver agent responds indicating if the lock succeed or not.

The Telephone agent in the previous example needs the TV agent not to accept other instructions of the type `Volume Level to Set` (if they aren't more important than the telephone's instructions) from when the telephone starts to ring until it is hung up, or ceases to ring in case nobody picks it up. Thus, when it starts to ring, the Telephone agent should send a request to the TV agent to lock the instruction type `Volume Level to Set` in, for example, level two. If it is not locked by any other agent in priority level two, the TV agent accepts the lock request. The Telephone agent should instruct then the TV agent to turn the volume down. If RemoteControl tries to send an instruction in the levels one or two, while the call is on, the instruction is ignored. When the telephone is hung up, the Telephone agent should send an instruction to the TV agent to restore the volume and then it should unlock the instruction type.

Locking certain instruction types in certain priority levels, makes the effect of the sent instructions to last for a certain amount of time, until the instruction type is unlocked or another agent sends an instruction of the same type in a higher level. This can be seen as the instruction type to be *being used* by an agent and no other agents can use it in the same or lower priority levels until it is unlocked.

But not all the agents require the effect of their instructions to last a specific amount of time. This is the case, for example, of the RemoteControl agent: when it sends an instruction of the type `Channel to Tune` to select a channel, it doesn't require the TV to be in that specific channel for a specific amount of time. Thus an agent can send an instruction of a specific type in a specific priority level without previously sending any

lock request. If the instruction type is not locked by any other agent in that priority level, it is accepted and processed. If it is locked, the instruction is not accepted.

As will be explained in part III, the priority based instruction system has been enough to control all the possible conflicts arising in the societies implemented in the robot MiReLa (chapter 7) and in the lamp MiFaRe (chapter 5). But the agents must be fair to assign an importance to their instructions, thats to say, to choose the priority level of the instructions they send. Competitive agents would make the society not to work, as they would try to get their instructions to be processed above those of other agents, so they would all send them using the highest priority level.

## 3.3 Implementation

### 3.3.1 Agent Interaction

When one agent connects to another, it must specify the type of the connection. This is done by sending two integer values: One to indicate it is an agent connection and one to indicate the type (*Send/Receive Instructions* or *Request/Serve Data*) of the connection.

| **Name** | **Number** | **Meaning** |
|---|---|---|
| AGENT_SIGNAL | 6 | Agent connection |
| INSTRUCTION | 8 | Connection of type *Send/Receive Instructions* |
| SHOW_DATA | 4 | Connection of type *Request/Serve Data* |

Table 3.1: Signals to determine the connection type.

Table 3.1 shows the names assigned to the numbers used to determine the connection type. The numbers have been chosen arbitrarily. For example, if an agent needs to connect to another agent to send instructions, it must send integer numbers 6 and 8 after the connection is established. This prepares the receptor agent to receive instructions from the connection maker agent.

The two integer numbers to indicate the type of the connection must be sent when ever a connection between two agents starts. It is not necessary, though, to send them when ever an instruction or data is transmitted. The type of the connection is established when the connection starts and it is maintained until the connection closes. It is not possible to change it without closing it first. A connection can be closed deliberately by the connection receiver agent or by the connection maker agent, and it is also closed if one of them dies.

Each connection is treated as a separate thread.

**Request/Serve Data Communication.** When an agent connects to another agent and defines it is a *Request/Serve Data* connection (sending numbers 6 and 4), the connection receiver agent waits to receive a message indicating what data type to serve. This message

is sent by the connection making agent (following the two numbers which determine the connection type), indicating what data type it requests. If the requested data type is available to be served by the connection receiver agent, it sends the data of this type to the client (the connection making agent) whenever it has new data of the specified type. All data are served labelled with an integer number. In other words, the data are sent preceded with an integer number. Each data type has a unique identifier number. This makes the client agent aware of what data type is going to receive so that it can be ready to receive it.

For example, the Telephone agent in the previous example, needs the `Volume Level` data from the TV agent. It should establish a connection with the TV agent and send numbers 6 and 4 to determine it is a *Request/Serve Data* connection and then send a message ("volume;" for example) indicating it requests the `Volume Level` data from the TV agent. Then when ever the volume of the TV changes, its value is sent by the TV agent to the Telephone agent preceded by an integer number indicating the data type.

The frequency with which an agent receives data, if it receives it from an agent running in a different computer, depends on the network capacity and load. If the data being served is big and changes at a frequency quite high it is important, first not to saturate the network and second to make the data requester agent receive actualised data. Each connection is handled as a separate thread. The thread which actualises the data, also actualises one general counter and sends a broadcast signal for all the threads waiting for this counter to be actualised. They are the threads which handle the connections with other agents to serve them data. They have their own counter, if the data to be sent is the data they are set to serve, they send it and actualise their counter. In this way, if the network doesn't permit to transmit all the data in the frequency it is generated, not all the data is sent, but the newest data. For example, it may happen that a thread detects data have been actualised, thus starts to serve it. Meanwhile data may be actualised a couple of times, thus the general counter also actualises a couple of times. When the thread finishes sending the data and compares its own counter with the general counter, it finds they are different and it starts to serve the last actualised data. Thus it may happen that even if an agent actualises data at a certain frequency, another agent at another computer receives them at a lower frequency, but, it always receives the latest available data.

**Send/Receive Instructions Communication.** When an agent connects to another agent and establishes a *Send/Receive Instructions* connection (sending numbers 6 and 8), it sends instructions in the form of *Instruction Message*s. An *Instruction Message* is a data structure including an array of 100 characters to store the message to be sent, an integer indicating the network port (see [113] for a detailed explanation of network ports) where the instructor agent bound to, an integer indicating the priority level the instructor agent uses to instruct the instructed agent, and an integer indicating the instruction type.

```
struct instruction_msg
{
  char msg[100];
  int from_port;
  int priority;
  int instruction_type;
};
```

Figure 3.3: An *Instruction Message* data structure.

If the instructor agent requires to lock an instruction type in a certain priority level, it sends an *Instruction Message* where the message is: <lock>. An instruction written between <> symbols indicates that it is a command and not a common instruction message. Only two commands have been defined, which are: <lock> and <unlock>. If the instructed agent can lock the requested instruction type at the requested priority level, it responds with a LOCK_SUCCEED signal (which is one integer number). Other wise it responds with a LOCK_FAILED signal.

If a LOCK_SUCCEED signal is received, the instructor agent sends the instructions to the instructed agent.

A priority level of an instruction type can be unlocked by the instructor agent when it finishes sending the instructions, sending an *Instruction Message* where the message is: <unlock>. It can also be unlocked by the instructed agent when a condition, such as a timeout, is satisfied or when a connection with the agent which locked the instruction is closed.

For example, returning to the example, when the telephone starts to ring, the Telephone agent, may request a connection with the TV agent and send the numbers 6 and 8 to define the communication is of type *Send/Receive Instructions*. Then it may send an *Instruction Message* including:

- The message <lock>.
- The port the Telephone agent is bound to.
- The number 2 for the priority level, meaning it intends to lock level two.
- The number which corresponds to the instruction type `Volume Level to Set` of the TV agent.

Figure 3.4 illustrates the *Instruction Message* the Telephone agent may send to the TV agent to lock the instruction type `Volume Level to Set` accepted by the TV agent in priority level two. If it is not already locked in priority level two, the agent TV responds with a LOCK_SUCCEED message.

```
struct instruction_msg
{
char msg[100] = <lock>
int from_port = port number agent Telephone is bound to
int priority  = 2
int instruction_type = identifier number of instruction
                       Volume Level to Set
};
```

Figure 3.4: *Instruction Message* that the Telephone agent may send to the TV agent to lock the `Volume Level to Set` instruction type in priority level 2.

Then, the Telephone agent may send a new *Instruction Message* indicating the volume level it requires the TV to generate sound, such as the message shown in figure 3.5.

```
struct instruction_msg
{
char msg[100] = The required volume level
int from_port = port number agent Telephone is bound to
int priority  = 2
int instruction_type = identifier number of instruction Volume Level to Set
};
```

Figure 3.5: *Instruction Message* that the Telephone agent may send to the TV agent to lock the `Volume Level to Set` instruction type in priority level 2.

And after the call is finished it may send a new instruction to restore the volume level to the initial level and a final instruction to unlock the instruction type `Volume Level to Set` in the priority level two.

### 3.3.2 Watchdogs

Watchdogs are utilities that monitor for specific system events and failures. These utilities can be either software or hardware, where available. The watchdog waits until a specific event occurs and executes a predefined action [111].

For the *Society of Agents* implementation being presented in this section, three types of software watchdogs have been designed and implemented. They all monitor the agents running in the society to restart them in case they die.

#### Agent Watchdog—One watchdog for each agent

Each agent is a two process program. The child process is the agent itself while the only thing the main process does is to wait for the child to finish. The child process finishes

only when it crashes. When this happens, and after a specified elapse time, the main process restarts the child process. The elapse time has been set to five seconds. It may happen that due to some configuration errors, the child process crashes at start-up. In this case it is not convenient for the main process to try to restart it as it will crash again. The elapse time was designed to avoid this situation to happen, thus if the child process crashes at start-up, the main process doesn't restart it and the program finishes.

#### General Watchdog—A watchdog for a list of agents on a computer

A higher level watchdog was implemented to watch a list of agents (the two process program, explained above) running in a computer. A list of agents running in a computer is given to this watchdog, and after a period of time (the period was set to five seconds) it checks to see if any of them have crashed. If one or more have stopped, it restarts them.

This watchdog has been designed to ensure that all the agents in the list are running. Other wise, if an agent stops, that it to say, if the two processes which are part of the agent stop, there would not be any mechanisms to restart them again. Agents added to the list must be those which have been tested before and don't crash at start-up, as the general watchdog will try to restart them each period of time.

For the implementations of the *Society of Agents* developed in this work (explained in chapters 5 and 7), the watching period of five seconds has been chosen. This period has been considered fast enough for the system to react when an agent crashes, to restart it without this causing any big harm in the performance of the whole system.

#### Watchdogs of the General Watchdog

Two additional watchdogs were implemented to watch the general watchdog. Each of them, watches for the other watchdog and also for the general watchdog. One has been set to watch every seven seconds and the other to do it every nine seconds. Setting the checking times to different amounts of times makes the checks to be asynchronous. Sometimes they may do their checking almost at the same time and the provability exists for both of them to crash close in time so that neither of them can restart the other. Practice showed that due to the simplicity of the code of the watchdogs, it is very unusual for them to crash. It never happened in the implementations of the *Society of Agents* done for the robot MiReLa (chapter chapters 5) and the lamp MiFaRe (chapter 7).

There are many ways to implement watchdogs and many of them are good enough for most of the applications. The most important thing is to have in mind an effective watchdog system is essential to be implemented for any multi-process application.

### 3.3.3 Programing the Agents—C++ and Perl

Software tools were created to help to add the interaction capabilities to the agents and to build, start, kill, and monitor their activity.

### Interaction capabilities in C++

The C++ programming language was used to implement the interaction capabilities explained in the previous sections for the agents. This language was chosen because it offers object oriented constructs that make it easier to create the tools as objects and reuse them for every agent. Being a compiled language and not an interpreted language (see [41] for a detailed comparison) makes it more suitable to use in machines with low computing power.

Each agent is a two process program: the agent watchdog and the agent itself, and it is bound to one port. The connections done from other agents to send instructions or to get data and the connections done to other agents to get data from them are handled as separated threads. Some of the agents have some activities apart from those derived from the interaction with other agents. They are also done as separated threads. Each agent is a two-process, multi-threaded program.

### A makefile.

A makefile was created which automatically generates the objects (.o files) necessary to build an executable file, from its dependencies, without the need for explicitly indicate all the objects needed to be created.

### Perl scripts.

Some Perl scripts were created to handle the creation, the start-up, and the shutting-down of the agents. The general watchdog and the watchdogs of the general watchdog explained in section 3.3.2 also have been programmed in Perl.

| Script | Description |
|---|---|
| createAgent.perl | Creates the files with the C++ code necessary to compile an agent with interaction capabilities with other agents. |
| makeAgents.perl | Generates the executable files of the agents specified as arguments. It executes the makefiles for them. |
| startAgents.perl | Starts (if they are not already running) the agents specified as arguments. It accepts as argument the way to start them: with no watchdogs, with general watchdog only, or with general watchdog plus watchdogs of the general watchdog. |
| killAgents.perl | Stops the agents specified as arguments. |

Table 3.2: Table with the names and description of the Perl scripts implemented for the *Society of Agents*.

## 3.4 Concluding Comments

Based on the basic ideas of *The Society Of Mind* of Minsky (section 3.1) a proposal about how to build *Societies of Agents*, to put together all the aspects involved in a service robot of Type II, has been defined (section 3.2). The proposal is to build them from agents doing simple things (each agent adds one simple skill to the whole), which use a simple way to communicate between them (no negotiation with one another, nor any attempt for effective ways to adjust to each other's interference). The communication between agents is limited to serve and request data, and to send and receive instructions. The relation between them is a Sever-Client relation. This is the only architecture principle applied to build the *Societies of Agents*.

# Chapter 4

# MiReLa Music Language (MML)

The *MiReLa Music Language* is the result of the research done to find an effective people-machine, machine-machine form of interaction. As said in chapters 1 and 2, the biggest effort, nowadays from researchers from all over the world, is being directed towards Human Natural Language synthesis and recognition in order for people to better interact with machines. One big problem is that these attempts haven't yet achieved reliable speech recognition in environments where the ambient noise is not controlled. And even if reliable speech recognition could be achieved for any environment, it is not clear this is the best solution to interact with any machine. Many people don't like the idea of talking to the fridge, the window blinds, the lights, etc. When machines use a Human Natural Language, it makes them look more intelligent than they really are. This may scare some people instead of bringing them closer to them.

But sound interaction is a very effective form of interaction for many animals (not only humans) as it permits to transmit and receive information remotely. As said in section 2.1.2, an interesting alternative approach, using whistling, has been promoted and demonstrated by Marc Bohlen, an artist, and his Universal Whistling Machine (UWM), [16]. Here is how he describes his work:

> "UWM is an investigation into the vexing problem of human-machine interface design. Whistling is much closer to the phoneme-less signal primitives compatible with digital machinery than the messy domain of spoken language. As opposed to pushing machines into engaging humans in spoken language, UWM suggests we meet on a middle ground. Whistling occurs across all languages and cultures. All people have the capacity to whistle, though many do not whistle well. Lacking phonemes, whistling is a pre-language language, a candidate for a limited Esperanto of human-machine communication."
>
> —*Marc Bohlen, The Universal Whistling Machine, 2004*

Music is sound formed from musical notes. It is much more simple for generation/-recognition than any Human Natural Language and can be generated naturally by humans, whistling. We whistle to our dog or to the horses, which we own to serve us. We know

they are not that intelligent to control us. If we whistle to the machines we may associate for them the kind of relationship we have with our dog for example. If we talk to them, we may associate a higher intelligence to them, and this may scare some people. Thus music for interaction may be better at bringing the machines closer to the humans.

It would be very useful to have a way to interact remotely with many machines by whistling. For example, when we are carrying some bags or a box in our hands, into a building, it would be useful to be able to operate the lift whistling, open some doors whistling and switch the lights on whistling, without having to ask for help, or leave the box or the bags on the floor. This could be done using very short melodies that are easy to whistle.

The *MiReLa Music Language* can be played by many musical instruments or modern mobile phones and PDAs, but it has been specially designed to be easy to whistle. In the implementations of the *MiReLa Music Language* done for robot MiReLa and the lamp MiFaRe, the system also helps whistling the language (see WhistleHelper 5.2.4).

This chapter explains the *MiReLa Music Language* (MML).

## 4.1 Looking for a Frequency Range for MML

Experiments have shown that a healthy young person hears all sound frequencies from approximately 20 to 20,000 hertz [42]. A typical person, however, can sing in only a rather narrow range of frequencies. Table 4.1 [63] shows the ranges demanded by the different voices of classical opera.

| **Voice** | **Range Hz** |
|---|---|
| Bass | 87.31 – 349.23 |
| Baritone | 98.00 – 392.00 |
| Tenor | 130.00 – 493.88 |
| Contralto | 130.81– 698.46 |
| Soprano | 246.94 – 1,174.70 |

Table 4.1: Human singing voice ranges, in Hz.

*MiReLa Music Language* (MML) has been designed to use notes from the C major scale: 1,046.5 Hz (C6, note *c*, octave 6, following the octave designation system of the American Acoustic Society, [3]) to 2,093 Hz (C7, note *c*, octave 7). Figure 4.1 shows the MML note rage in a piano keyboard. This is higher than most humans can reach by singing so as to avoid interference from people (particularly children) speaking (and singing), yet, it is the frequency range most people whistle more comfortably. Based upon some trials conducted with different people, we concluded that people can whistle up to a frequency of about 2,093 Hz (C7) without great effort, but higher notes are not very easy to whistle. And they can whistle down to a frequency of 783 Hz without great effort (G5), but lower notes are not very easy to whistle with high power. The notes in the frequency range C6-C7

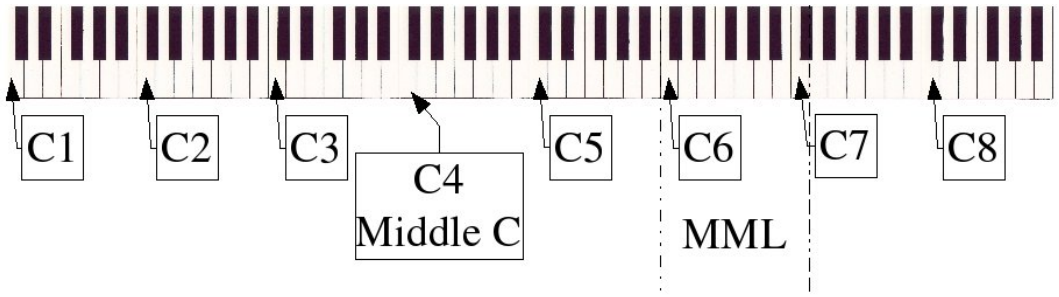



Figure 4.1: Figure of a piano keyboard indicating the note names following the octave designation system of the American Acoustic Society, [3]. It shows that the MML alphabet notes are defined in the octave C6-C7.

can also be played on many simple musical instruments, such as Xylophones, Harmonicas, Treble Recorders, and modern mobile telephones.

## 4.2 The MML Alphabet

MML is a proper language, albeit, an artificial one, because it has a well defined grammar, lexicon, dictionary, as well as some simple rules for constructing words. It is also a proper language because both the people and the machines that use it, must both hear and recognise it, as well as produce it correctly. The alphabet used to form the words of MML is thus formed by the eight notes shown in table 4.2.

| NOTE | | | FREQUENCY (Hz) |
|---|---|---|---|
| c | Do | C6 | 1,046.52 |
| d | Re | D6 | 1,174.64 |
| e | Mi | E6 | 1,318.52 |
| f | Fa | F6 | 1,396.92 |
| g | Sol | G6 | 1,567.98 |
| A | La | A6 | 1,760.00 |
| B | Si | B6 | 1,975.52 |
| C | Do | C7 | 2,093.04 |

Table 4.2: The MML alphabet: note names and frequencies. For the note names, the first column shows the names given to the notes of the MML alphabet based on the English notation (notes A, B, C, are written in capital letters to help making more clear them to represent higher notes than the notes c, d, e, f, g), the second row shows the names of the alphabet notes using the notation created by Guido D'Arezzo [67] used in Latin countries, which have been used to define the name of the robot MiReLa and the lamp MiFaRe, and the third column shows their names following the octave designation system of the American Acoustic Society, [3].

## 4.3 MML Phrases

Each statement, or sentence, starts with the name of the machine to which it is directed, unless this is clear from the context, or not needed. The structure of a MML sentence can take one of the following forms:

```
[Name]+Verb+Arguments
[Name]+Adverb
[Name]+Interjection
```

where

| | | |
|---|---|---|
| `Name` | = | the name of the machine to which the sentence is directed. |
| `Verb` | = | the action to be performed. |
| `Arguments` | = | needed parameters values for the action, which can be: Nouns, or Adjectives. |
| `Noun` | = | predefined names of things, such as robot transitions, and numbers. |
| `Adjective` | = | indicates a quality of a noun. |
| `Adverb` | = | "yes", or, "no", or "OK". |
| `Interjection` | = | "hello", or "bye". |

Communication from a machine to people follows the same structure, except that machines don't add the name of the person (which typically they do not know, nor have a way of knowing). Instead they may add their own name at the beginning or not, depending on the context.

### 4.3.1 Word Separation: A Silence and the Tonic Note ($c$,$C$)

The tonic note $c$ (C6 or C7) is special. It serves to separate words of MML phrases. The words in MML include any note in the alphabet except $c$ or $C$ which, if heard or detected, indicate the word has finished.

A silence after a detected note, indicates also the end of a word. Thus note $c$ in MML is the sound of silence. (In the implementation of the MML described here, a silence is detected by the NoteChecker agent, (see 5.2.2 for more details.)

### 4.3.2 End of a Phrase.

A phrase is finished with the tonic note $c$ or with a silence. When the note $c$ or a silence is detected, depending on the words detected before, it is possible to determine if it is the end of the phrase or not.

If a long silence is detected, even if the phrase is not a complete phrase, it is considered to be terminated. If it is not a complete, legal MML phrase, the statement is ignored. For example, if *a name + a verb* are detected and a specified amount of time passes with no more note detections (thus, with no arguments detected for the verb), the phrase is considered to be finished and it is ignored as it is incomplete. (In the implementation of the MML described here, the WordChecker agent determines when a long silence is detected, see section 5.2.3 for more details.)

### 4.3.3 Note Duration.

The duration of the notes doesn't affect the meaning of the words they form. That's to say, there is no restriction on the rhythm of the MML melodies. This makes it easier to play them on any instrument or to whistle them and get them to be recognised.

Even if the duration of the notes doesn't affect the detection of the *MiReLa Music Language*, a uniform way was chosen to generate the language by machines. It was decided to use all notes of the same duration to generate words in MML, and make the silences to last double the duration of a note. Thus, in the examples presented in the next sections, semi-crochets are used to represent the notes which form the words in the *MiReLa Music Language* and crochets for the silence notes $c, C$.

## 4.4 From Musical Sounds to MML Phrases

The vocabulary for MML, is defined and stored in a text file, as a set of triples: word; category; musical note sequence. It is used to translate sequences of notes into words. Figure 4.2 shows an example of a vocabulary item.

```
{
 MiReLa // Name of a machine
 NOUN   // Category
 edA    // Musical note sequence
}
```

Figure 4.2: An example of a vocabulary item in MML.

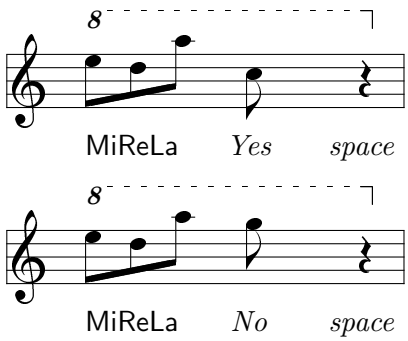
To improve the reliability of note detection and recognition, all the consecutive repetitions of a note are considered as one note. So, the sequence $c\ c\ c$ is detected and recognised, as one note $c$, and not three notes. This helps to make the recognition more reliable in places with significant acoustic echo, as is the case where the lifts are located in the Main Building of the San Sebastián Technology Park. MML words thus cannot be defined using repeated notes.

In MML, Names are composed of any three non-repeating notes, except the top or bottom tonic $c$. Verbs, Adverbs, Nouns and Interjections may be composed of any non-repeating number of notes, except top or bottom tonic $c$. The adverbs "Yes" and "No" are commonly

used in reply to questions. To simplify the use of these words, two special case rules have been defined, which are applied to the sound heard in response to a question needing an answer of type "Yes" or "No":

| | | |
|---|---|---|
| Name + $c$ | means | "Yes" , and |
| Name + $g$ | means | "No". |

For example to answer "Yes" or "No" to a question from the robot MiReLa it is possible to play:




## 4.5 Directed Listening.

Knowing the structure of the phrases and the musical composition of the words, it is possible to focus the sound signal processing to try to detect certain specific notes and so not have to try to look for all possible notes in the sound signal.

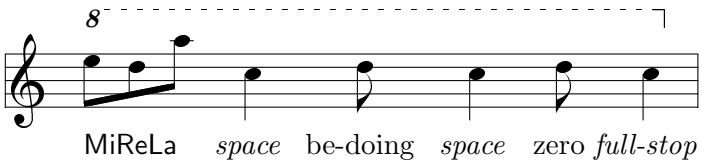
For example, in the case of the robot MiReLa, as all the phrases directed to it start with its name which is composed of the notes *Mi, Re, La* or *e, d, A*, the robot waits to detect the note *e*. Any other note played is ignored until an *e* is detected. When it detects an *e* it tries to detect the *d* that follows, and then the *A* that should come next in the subsequent notes.

Then, the robot waits to listen for a *c* or to detect a silence. It then looks in the vocabulary for the defined Verbs, Adverbs and Interjections and focusses the sound signal processing on only the possible note that can fit in each position. When a verb is detected it focusses the attention on the notes that could form possible arguments.

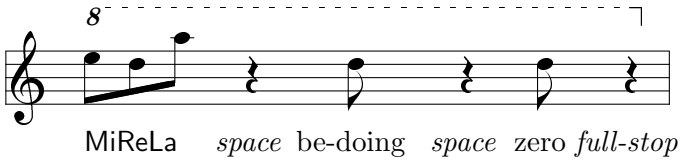
This technique filters the detection at a low level. It is similar to what we humans do when listening to a spoken language. In human communication, there is a very close relation between what we hear and what we think we are listening to and so what we expect to hear, based upon our inherent knowledge of the language or languages we use [1].

## 4.6 Example of an Interaction in MML.

For example, to tell the robot MiReLa, presented in section 7.1, to go to a particular place, it is asked to "be doing" a particular *Transition Group* (see section 7.2). This makes MiReLa execute the series of *Transition Group*s it needs to complete, before it can be doing the nominated one, which if it is associated with a place (by us), rather than a translation, is some kind of robot motion that has a zero translation component: slowly scanning from side to side, for example. To tell MiReLa to be doing Transition Group zero in MML, we play or whistle:

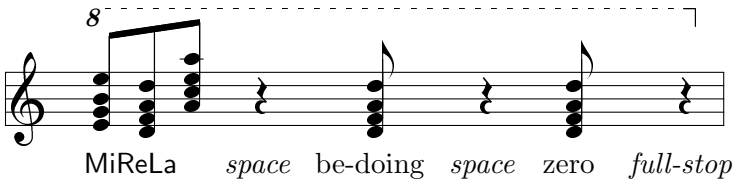



or, instead of using a tonic note, $c$ or $C$ (the octave above), to mark the word spaces, it is possible to use a rest (silence) as follows:




or any combination of rests and tonics, whatever makes for the easier production. Similarly, the *full-stop* at the end can be marked by a tonic $c$ or $C$. It can be seen from this example that the Transition Group zero is only denoted by the number zero, the note $d$, since the verb be-doing can only take, as an argument, a *Transition Group* number. Furthermore, it can also be noted that in this example the note $d$ is used both to denote the verb be-doing and the number zero. Once again, the correct interpretation of these two one note words depends upon its place in the full phrase: only a verb, an adverb or an interjection can follow a name, so, in this case the first $d$ is a verb (be-doing) and not a number, and the second $d$ must be a noun argument for the preceding verb, not a verb, an adverb or an interjection. But, to avoid confusion, a verb, an adverb, and an interjection must not be equal (same notes), as any of them can go after a Name in MML. The same happens for nouns and adjectives, which can be arguments of a verb in MML.

To the request to be doing *Transition Group* zero, MiReLa replies saying:




from which it is seen that MiReLa has a different voice; one that speaks in chords, rather than in single notes (see the SoundServer agent 5.2.5). MiReLa generates the phrases in *MiReLa Music Language* using silences for spaces and full-stops, not the tonic notes $c$ or $C$.

# Part III

# Application Experiments

# Chapter 5

# The Lamp MiFaRe



Figure 5.1: The lamp MiFaRe.

The lamp MiFaRe is a desk lamp that can be operated (switch on or off) using the *MiReLa Music Language* (MML). It consists of a conventional desk lamp, one circuit board (MICROPIK PS6E8S [73]) containing eight relays which can be controlled from a PC over a standard serial port, a PC, a microphone, and a loudspeaker connected to the sound card of the PC. When the name of the lamp MiFaRe in MML is played or whistled and detected, the lamp changes its state: it switches on if it was off; or it switches off if it was on; and it says "OK" in MML.

This system has proved to be both reliable and robust in a normal office environment, with typical kinds and amounts of environmental noise from computer ventilation systems, people, music, telephone conversations, etc. The lamp MiFaRe has been operative since March 2005.

Section 5.1 presents the hardware components used for the lamp MiFaRe and how they have been connected.

Section 5.2 explains how a *Society of Agents* makes it possible to operate the lamp MiFaRe using MML. A diagram is presented at the beginning of the section which includes all the agents of the society, indicating the data generated by each, the instructions admitted by each, and the connection to request data or send instructions between each other. A brief explanation follows about how this agent network works. The subsections of this section have been designed to explain in detail each of the agents. One subsection for each of the agents. Each subsection is about one page long, and they all have the same structure (following the style of Minsky's book *The Society Of Mind*).

## 5.1 Hardware Configuration

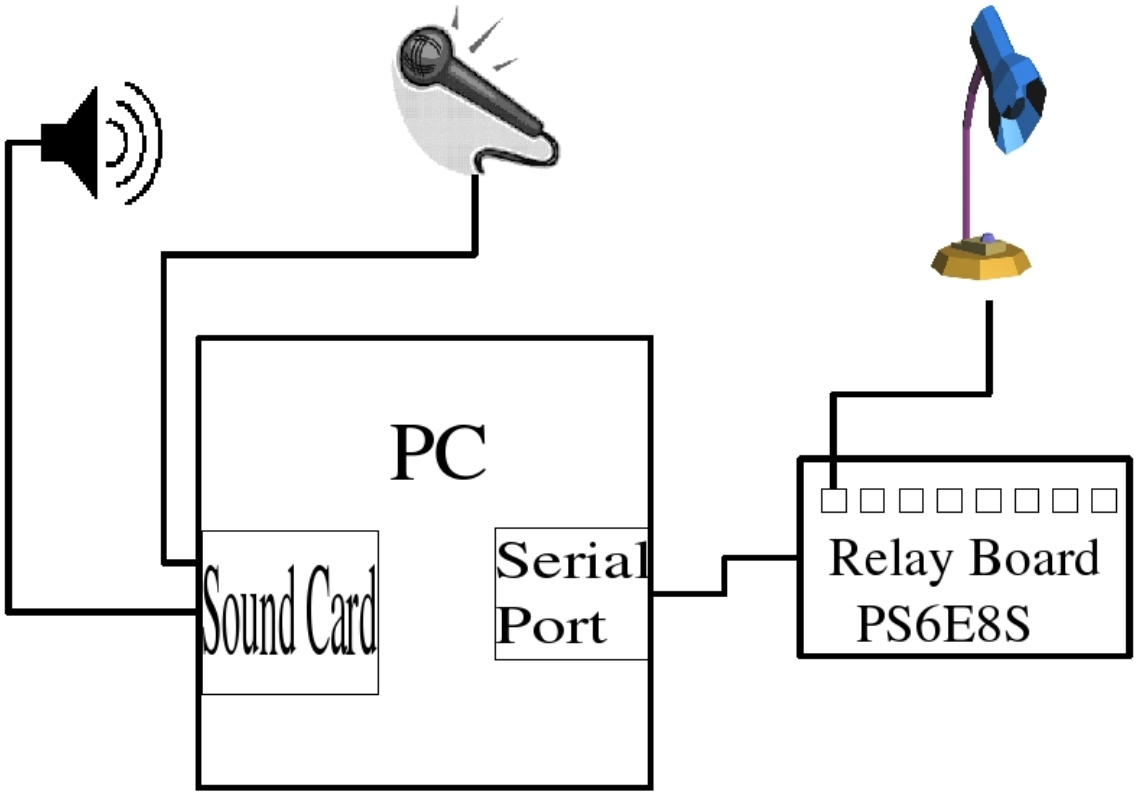



Figure 5.2: The lamp MiFaRe is composed of a conventional desk lamp connected to one of the eight relays of a relay board MICROPIK PS6E8S [73]. This relay board is connected to the serial port of a PC, which has a microphone and a loudspeaker connected to its sound card.

The lamp MiFaRe can be controlled using the *MiReLa Music Language*. Figure 5.2 shows the configuration of the hardware components which make this possible. It is composed by a PC (equipped with two Pentium III processors at 800 MHz), where the software computation (explained in the next section) is done. It needs to be able to sense, capture sound from the ambient and act to control a desk lamp and generate sound. To get sound input, a common inexpensive desktop directional microphone has been connected to the sound card of the PC. Two loudspeakers connected to the sound card of the PC enable the PC to generate sound. A relay board which can be controlled through a serial connection has been connected to the serial port of the PC, and a desktop lamp has been connected to one of the relays of the relay board. Activating and deactivating the relay, makes it possible to switch the lamp on and off.

Sound input is obtained from the microphone, and it is processed in the PC. Musical notes *e,f,d* or *Mi,Fa,Re* (refer to the MML alphabet table presented in section 4.2) correspond to the name of the lamp MiFaRe in MML. When ever this sequence of notes is detected, the PC changes the state of the relay where the lamp has been connected, that's to say, it activates it if it was deactivated and it deactivates it if it was activated. The result is that the lamp is switched on if it was off, it switches off if it was on, and says "OK" in MML through the loudspeakers.

## 5.2 A Society of Agents to Control Lights using the MML

This section presents all the agents which form the *Society of Agents* of MiFaRe. Figure 5.3 presents all the agents which run in the PC. It shows the data generated by each agent and the instructions accepted by them, and represents the connections to get data from and to send instruction to each other. The agents are presented in subsections of one or two pages each. One subsection for each agent. Each subsection starts with a table indicating the skill the agent adds to the system, the type of instructions it accepts, the data it generates, the type of instructions (indicating to which agents) it sends, and the data types it generates. Then the operation of the agent is explained.

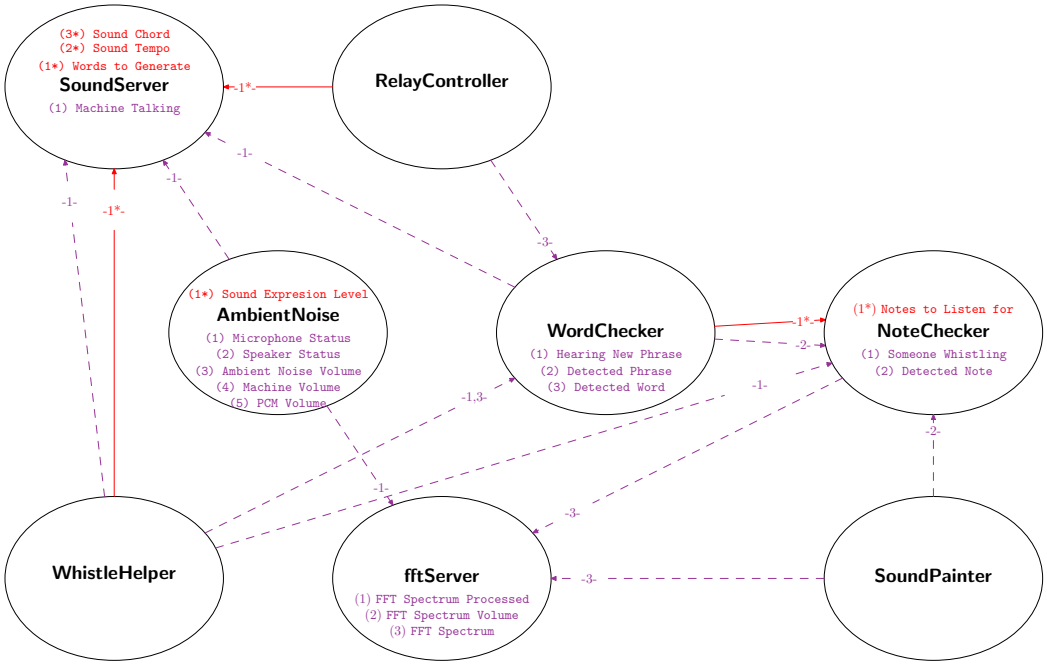



Figure 5.3: Representation of the agents of the *Society of Agents* implemented in the lamp MiFaRe. Section 3.2 explains the meaning of the different types of arrows (dashed or continuous), their labels and the nature of the names written at each agent (agent name, accepted instructions, and generated data).

Figure 5.3 presents the *Society of Agents* implemented in the lamp MiFaRe to control one relay of the PSE8S relay board using MML. This *Society of Agents* is formed by eight agents. As explained in the introduction, seven of them are designed to help recognise and generate the *MiReLa Music Language*. These seven agents, form part of the *Society of Agents* implemented in the lamp MiFaRe, and of the *Society of Agents* implemented in the robot MiReLa (formed by twenty-three agents), which is presented in chapter 7. The interaction needed with the lamp MiFaRe is very simple. It only uses the name of the lamp (one word) to switch on and off the lamp. But the interaction with the robot MiReLa is reacher (using phrases with a name, a verb and its arguments). These seven agents put in a society allow any phrase in *MiReLa Music Language* to be recognised and generated.

The FFTServer agent generates the FFT (Fast Fourier Transform) [68] and makes it available for any agent requiring it. The NoteChecker agent gets the FFT Spectrum from the FFTServer agent and looks for any MML notes being played at each moment. The WordChecker agent, gets the recognised notes from the NoteChecker agent and builds the words into a phrase. As the recognition is directed to some particular notes, each time the WordChecker agent gets a note, it instructs the NoteChecker agent about what notes to direct the recognition to. When the WordChecker agent detects a whole phrase, it makes it available for any agent that needs it.

The `Machine Talking` data generated by the SoundServer agent, indicates when is the machine talking. This is used by many agents. One of them is the WordChecker agent,

which uses it to recognise if a note obtained from the NoteChecker agent was generated by the machine or some other source—a person.

The WhistleHelper agent was designed to help people whistle the MML phrases. It gets data from other agents, such as the NoteChecker and WordChecker agent, and if it detects someone is trying to whistle, it helps whistling by playing the name of the machine. The AmbientNoise agent was designed to control the sound generation power (volume) depending on the ambient noise level.

These are the seven agents which also are part of the *Society of Agents* of the robot MiReLa and make the robot able to recognise/generate the *MiReLa Music Language*.

The RelayController agent (which has only been implemented in the lamp MiFaRe) controls the relay board where the lamp MiFaRe has been connected. It gets data from the WordChecker agent, when the name of the lamp is recognised, it changes the state of the lamp (switch on if it was off and switch off if it was on), and instructs the SoundServer agent to generate the MML notes for the word 'OK'.

### 5.2.1 The FFTServer Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Generate FFT Spectrum from sound | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| | `FFT Spectrum`<br>`FFT Spectrum Volume`<br>`FFT Spectrum Processed` | |

Table 5.1: Skill Table for the FFTServer agent.

#### Description:

This agent generates the FFT (Fast Fourier Transform) of the digitised sound signal received from the microphone. It generates new `FFT Spectrum` data at a frequency of 21.5 Hz, which, if the connection permits, are served to all the other agents that have requested this type of data at this same frequency (see section 3.3 for more details about how is the data served). It calculates the power (area under the FFT curve) of the whole FFT Spectrum and the power of the Spectrum for the frequencies smaller than 1046.50 Hz. This is the frequency of the note *c* (see table 4.2) which is the lowest note in the musical alphabet of MML (see section 4.2). This provides a way of measuring the ambient noise. It also identifies the frequency with the largest amplitude in the full Spectrum.

To make all this information available to other agents that need it, three data types are defined:

- `FFT Spectrum`: Data type containing the whole FFT Spectrum.
- `FFT Spectrum Volume`: Data type containing the value of the power calculated from the whole FFT Spectrum.
- `FFT Spectrum Processed`: Data type containing three numbers:
  - The power calculated from the whole FFT Spectrum.
  - The power calculated from the values in the Spectrum for the frequencies smaller than 1046.50 Hz.
  - The frequency of largest amplitude in the Spectrum.

### 5.2.2 The NoteChecker Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Detect notes from a FFT Spectrum | `Notes to Listen for` |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| `FFT Spectrum`→5.2.1 FFTServer | `Detected Note` `Someone Whistling` | |

Table 5.2: Skill Table for the NoteChecker agent.

#### Description:

This agent obtains the `FFT Spectrum` data from the FFTServer agent and processes it to detect particular notes.

The NoteChecker agent maintains a list with the notes to direct the listening to. An agent may send to the NoteChecker agent an instruction of the type `Notes to Listen for`, containing the list it needs the NoteChecker agent to direct the listening too. When ever it receives an instruction of the type `Notes to Listen for` it replaces the old list with the new list. It doesn't add the notes in the instruction of the type `Notes to Listen for` to the old list, as in this way, after receiving several instructions it would end up directing the detection to all the notes which form the MML alphabet.

The algorithm for the note detection is as follows:

1. Check if the largest amplitude of the whole FFT is of a frequency lower than 1046.50 Hz (note *c*, the first note on the MML alphabet, see table 4.2 in section 4.2). In other words, check to see if the frequency with the largest amplitude is in the ambient (non-MML) range of the `FFT Spectrum` data.
2. If this is not the case, calculate the sound power for a range ($\pm\frac{1}{4}$ of a tone) of frequencies from the exact frequencies of the notes defined in the MML alphabet. It gets eight values for each FFT, one for each note of the alphabet.
3. To consider a valid note has been detected two conditions must be fulfilled:
   (a) One of these eight values is three times larger than the third largest value. It is compared with the third largest value and not with the second because as the power is calculated from a range of frequencies (step 2), the second largest value is the previous or the next note in the alphabet to the note with the largest power, and the difference in the power is not larger than twice its value.

   (b) The note must be in the list for the notes to direct the detection to (given by the instruction type `Notes to Listen for`.

4. It considers a silence has been detected if more than one second has elapsed with no MML note detection. This is served in the `Detected Note` data with "-" character.

The NoteChecker agent generates as output the `Someone Whistling` and `Detected Note` data. `Someone Whistling` is a flag turned on when a sound of any frequency defined in the MML alphabet is detected, regardless of whether the agent has been instructed to listen for it or not. In other words, when the condition one of the step three of the note detection algorithm is fulfilled, regardless if the condition two is fulfilled or not, the `Someone Whistling` flag is turned on. The `Detected Note` data, is a data type designed to serve the notes which have been detected by the NoteChecker agent to any agent requesting them. This data type is formed by the name of the note and its duration. As soon as the start of a note is detected, The `Detected Note` data is served (including the note's name and its duration which in this case is zero). When the detected note terminates (because a silence or another note has been detected) it is served again (including the note's name and its duration which in this case is the time the note lasted).

### 5.2.3 The WordChecker Agent

| Skill | Accepted Instructions |
|---|---|
| Detect words from detected notes | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `Detected Note`→5.2.2 NoteChecker<br>`Machine Talking`→5.2.5 SoundServer | `Hearing New Phrase`<br>`Detected Phrase`<br>`Detected Word` | `Notes to Listen for`→ 5.2.2 NoteChecker |

Table 5.3: Skill Table for the WordChecker agent.

#### Description:

The WordChecker agent checks the `Machine Talking` data served by the SoundServer agent to see if the machine is generating MML notes or not. In case it isn't, it checks the notes served by the NoteChecker agent to decide if something meaningful in MML is being detected. To make the detection more reliable, it instructs the NoteChecker agent to focus the detection to some specific notes, each time. These notes depend on the notes and words detected so far (see section 4.4).

The WordChecker agent does the word detection and makes them available to other agents in the form of the `Detected Word` data, and it also does the detection of the phrases in MML, making them available in the form of `Detected Phrase` data. The WordChecker agent instructs the NoteChecker agent to focus the detection to some specific notes. They depend on the words the WordChecker agent is expecting to hear and these words depend on the previously detected words. Thus the decision was taken to make the formation of detected notes into phrases in a single agent, which is the WordChecker agent.

`Hearing New Phrase` is a flag which is turned on when the WordChecker agent is ready to hear a new phrase, and is turned off when it starts to hear a new phrase. When the WordChecker agent gets the note *c* or a silence, it can determine if the sentence has finished or not, depending on the words detected before. But it doesn't always wait until it gets a meaningful sentence to determine it has finished. If more than four seconds pass after a silence, or note *c* is detected, the MML phrase is considered to be terminated, and the WordChecker agent prepares itself and the NoteChecker agent to listen for a new phrase. If a complete phrase has not been recognised, it is ignored.

### 5.2.4 The WhistleHelper Agent

| Skill | Accepted Instructions |
|---|---|
| Help whistling the *MiReLa Music Language* | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `Detected Word`→5.2.3 WordChecker<br>`Hearing New Phrase`→5.2.3 WordChecker<br>`Machine Talking`→5.2.5 SoundServer<br>`Someone Whistling`→5.2.2 NoteChecker | | `Words to Generate`→ 5.2.5 SoundServer |

Table 5.4: Skill Table for the WhistleHelper agent.

#### Description:

*MiReLa Music Language* is designed to be easily whistled. People are typically quite good at whistling a song using the correct intervals between notes. But to whistle the exact frequencies we often need a reference tone to help us get started in the right key.

As the WordChecker and PhraseChecker agents do, the WhistleHelper agent also checks the `Machine Talking` data served by the SoundServer agent to know if the machine is generating MML notes or not. If it isn't, when `Someone Whistling` flag (from the NoteChecker agent) indicates that a sound of any frequency defined in the MML alphabet is being heard, and if `Hearing New Phrase` (from the WordChecker agent) indicates the machine is listening for the start of a new phrase, and if no note detections are obtained from the `Detected Note` data (from the NoteChecker agent), the WhistleHelper agent concludes there is someone trying to whistle to the machine and instructs the SoundServer agent to play the name of the machine to try to help the person get the right notes to start a MML phrase. This also helps people learn the MML name of the machine.

### 5.2.5 The SoundServer Agent

| **Skill** | | **Accepted Instructions** |
|---|---|---|
| Generate sound | | `Words to Generate`<br>`Sound Tempo`<br>`Sound Chord` |
| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
| | `Machine Talking` | |

Table 5.5: Skill Table for the SoundServer agent.

#### Description:

The SoundServer agent makes the machine generate MML. The output data generated is `Machine Talking`. `Machine Talking` is a flag turned on when the machine is generating MML notes (talking). This is very useful for other agents (for example the NoteChecker, WhistleHelper, and AmbientNoise agents) to determine if the detected sound was generated by the machine or came from some other sound source.

This agent accepts three kinds of instructions.

`Words to Generate` : The words to generate.

`Sound Tempo` : The speed of generation of the musical language words. As explained in section 4.3.3, it was decided all the notes forming any word in MML to have the same duration. The SoundServer agent accepts the following instructions of the type `Sound Tempo`:
Very Slow (0.6 s/note), Slow (0.45 s/note), Medium (0.3 s/note), Fast (0.2 s/note), Very Fast (0.15 s/note), Last State, and Default.

`Sound Chord` : The chord in which to generate the words.
Monophonic (single), Mayor, Minor, Last State, and Default.

At the reception of some kinds of `Words to Generate`, the response of the SoundServer agent is to play a song. It has the ability of playing songs in its own "voice" using the `Sound Chord`, and the `Sound Tempo` selected at the moment. Or it can play songs in MP3 or WAV format.

For the `Sound Tempo` and `Sound Chord`data types, when instruction "Last State" is received they return to the state they were before. For example, if `Sound Tempo` was "Medium" and changed to "Fast" on command, if it receives "Last State" it changes to "Medium" again.

These two data types have also a default state, which is the state they start up. They return to this state if they receive the instruction “Default”.

### 5.2.6 The AmbientNoise Agent

| Skill | Accepted Instructions |
|---|---|
| Adjust sound output power (volume) depending on the ambient noise | `Sound Expresion Level` |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `FFT Spectrum Processed`→5.2.1 FFTServer<br>`Machine Talking`→5.2.5 SoundServer | `Microphone Status`<br>`Speaker Status`<br>`Ambient Noise Volume`<br>`Machine Volume`<br>`PCM Volume` | |

Table 5.6: Skill Table for the AmbientNoise agent.

#### Description:

The AmbientNoise agent controls the output power (volume) of the generated sounds. It obtains sound data from the FFTServer agent and decides on the output power to use for the sound generation. So the AmbientNoise agent makes the sound output power depend on the level of the ambient noise: when the ambient noise level is high, the power of the generated sound is high and when the ambient noise level is low, the power of the generated sound is low. It checks `Machine Talking` to distinguish the ambient noise from the sound generated by the machine.

The ambient power is calculated using the following formula:

$$AmbientPower_n = (1 - \alpha) * AmbientPower_{n-1} + \alpha * InstantPower \qquad (5.1)$$

where $\alpha$ can take values between 0 and 1. This is the equation for an exponentially moving average filter [102], also known as a low pass filter. $InstantPower$ is the power calculated from the whole FFT Spectrum, if `Machine Talking` is set to true, and is the power calculated from the values in the Spectrum for the frequencies smaller than 1046.50 Hz, if `Machine Talking` is set to false. Both the values are obtained from the `FFT Spectrum Processed` data. The value 0.01 has been assigned to $\alpha$: `InstantPower` value is received from the FFTServer agent at a frequency of 21.5 Hz commonly (refer to sections 3.3.1 and 9.1.1 for more details about the communication). This means that in five seconds 107.5 values are received. Choosing the value 0.01 for $\alpha$ in this equation means the `InstantPower` data is being weighted so as to calculate an average with the values received in about the last five seconds. This is a value to allow the AmbientNoise agent to react fast enough to the changes in acoustics conditions but avoiding power peaks to be too significant.

The ambient noise power is served in the form of the `Machine Volume` data if `Machine Talking` is set to true and in the form of `Ambient Noise Volume` if it is set to false.

The machine may need to "whisper" or may need to "shout". Thus, the AmbientNoise agent accepts instructions of the type `Sound Expresion Level`. There are five levels defined for this expression value, they are presented as fractions of the full power an amplifier can generate: Very Low (20%), Low (40%), Medium (60%), High (80%), and Very High (100%).

As it happens for the `Sound Tempo` and `Sound Chord` data of the SoundServer agent, the `Sound Expresion Level` data of the AmbientNoise agent also accepts the instructions "Last State" and "Default". The effect is the same as for the `Sound Tempo` and `Sound Chord` data explained at the SoundServer agent (section 5.2.5).

The power of the generated sound depends on the selected `Sound Expresion Level` and the `Ambient Noise Volume`. It is served in the form of the `PCM Volume` data.

Depending on the value of the *AmbientPower* and if the machine is generating sound or not, it is possible to detect if the microphone and the loudspeakers are working properly. For example if the *AmbientPower* is 0, it means that the microphone is probably unplugged, or has been switched off or that it has failed and if `Machine Talking` is true and `Machine Volume` is 0, it means that the loudspeakers are probably unplugged, or that the amplifier has been turned off or has failed. This information is served in the form of the `Microphone Status` and `Speaker Status` data.

### 5.2.7 The SoundPainter Agent

| Skill | Accepted Instructions |
|---|---|
| Paint Sound Data | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `FFT Spectrum`→5.2.1 FFTServer<br>`Detected Note`→5.2.2 NoteChecker | | |

Table 5.7: Skill Table for the SoundPainter agent.

**Description:**

The SoundPainter agent requests the `FFT Spectrum` data from the FFTServer agent and the `Detected Note` data from the NoteChecker agent and paints them to a window on the computer screen. The graphical part of this agent was programed by Iñigo Martínez in C++ using the OpenGL libraries.

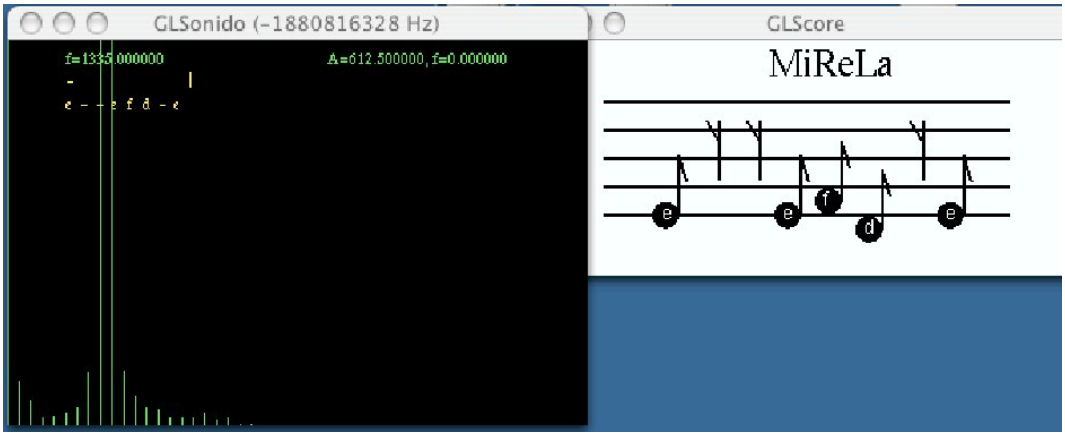



Figure 5.4: The SoundPainter agent's screen-shot. In the left window, the `Detected Notes` are painted on the upper part, and the `FFT Spectrum` is painted on the lower part. In the right window the `Detected Notes` are painted in a musical staff.

### 5.2.8 The RelayController Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Control relays from serial port | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| `Detected Word`→5.2.3 WordChecker | `Relay Status` | `Words to Generate`→ 5.2.5 SoundServer |

Table 5.8: Skill Table for the RelayController agent.

#### Description:

The RelayController agent controls the PS6E8S relay board [73]. It requires the `Detected Word` data from the WordChecker agent.

When the name of the lamp is recognised, it changes the state of the relay 1 (to which the desk-lamp is connected), that's to say, it switches the light on, if it was off, or it switches the light off, if it was on; and it also sends an instruction of the type `Words to Generate` to the SoundServer agent with the word 'OK'.

## 5.3 Concluding Comments

The installation developed to operate the desk-lamp MiFaRe using the *MiReLa Music Language* has been explained in this chapter. It shows that the recognition and generation of the *MiReLa Music Language* is done by seven of the eight agents which form the *Society of Agents* of MiFaRe. This makes MiFaRe a good example to explain how the recognition and generation of the *MiReLa Music Language* is done.

The installation of such a system to operate MiFaRe using the *MiReLa Music Language* is not complex nor expensive. It requires a computer, a common microphone, a loudspeaker, and a relay board controlled from the PC serial port. The PC used by the lamp MiFaRe is also used for common working tasks in the office, such as text processing, programing, or Internet navigation. The processing power required for the lamp operation is low and doesn't affect to the performance of the rest of the task being done in the same PC.

This example serves to show that the *MiReLa Music Language* can be used to interact with many machines or devices, not only with service robots. Many devices in people's houses could be controlled from a similar installation, connecting the device(s) to the relay board (the concrete relay board used has eight relays and only one was used) to operate them using the *MiReLa Music Language*.

# Chapter 6

# The Lift

A system has been designed and installed that uses Dual Tone Multi Frequency (DTMF) [46] sounds—like telephones—that can be used to control one of the lifts in the Main Building of the San Sebastián Technology Park. This installation uses DTMF boards (NORCOMM NC400 [80]) which offer three relay controls, together with pairs of microphones and loudspeakers in the control panels of the lift on each floor, and one inside the lift cabin. This set up allows a robot, or any other machine that can generate DTMF tones, to call this lift, to hold the lift doors open, and to state which floor the lift should go to, using different DTMF tones.

The lift is also able to generate sound (not DTMF tones) to indicate that it has detected and recognised a request. This simple system has proved to be relatively easy to design, install, and make work reliably, despite the large variation in acoustics between the area in front of the lifts at each floor, and the lift cabin.

The main motivation to install an extra system to operate the lift came from the need for the robot MiReLa to be able to operate it. As said in the introduction, MiReLa is being developed to guide people through the Main Building of the San Sebastián Technology Park, thus it needs to be able to call the lift, enter into it, go to a different floor and get out of it. But it doesn't have hands or manipulators to push the buttons of the lift. Many ways could be found to interact with the lift. They could be based on an infra-red or a wireless Bluetooth connection for example. The biggest disadvantage of using them, is that the interaction robot-lift remains secret to the people following the robot. The system selected for doing this interaction, uses sound (DTMF tones from the robot to the lift and sounds of some specific frequencies from the lift to the robot). This means that even if the people following the robot can't understand what the sounds mean, they see an interaction is occurring between the robot and the lift.

This system was developed with the help of Pablo Aramendia, who built and tested one of the devices for the control panels on the floors and the device for the lift cabin, as part of an Enterprise Practical Program of SEIM (Centro de Estudios Oficiales). This work is reported in [4].

This is an example of machine-machine sound interaction. Interaction is done using DTMF

sounds, they can be generated by any telephone, but, they can't be generated by whistling, nor with any (conventional) musical instrument.

DTMF based sound communication was chosen because it allows interaction using sound and there are several inexpensive boards available (mainly designed for domotics) which translate DTMF sound into relay activation/deactivation. This is a first step towards installing a full *MiReLa Music Language* system. It has been done to test the viability of using sound in such variable and uncontrolled environments.

Figures 8.1 and 8.2 in section 8.1.1 show the location of the lift in a general plan of the Main Building of the San Sebastián Technology Park.

## 6.1 Operating a Lift with DTMF Tones

The lifts are usually operated using the buttons by the side of the door on each floor and inside the lift cabin (for example, pushing the button 1 inside the lift cabin, instructs it to go to the floor number one). These buttons are basically circuit switches: pushing a button closes or opens a certain circuit, which generates a specific signal for the lift controller to do a specific thing, such as make the lift go to a particular floor, or open the lift's doors.

In the lifts of the Main Building of the San Sebastián Technology Park almost all the buttons (inside the cabin and at each floor) close a circuit when pushed. Only, the button inside the cabin to keep the doors of the lift open, opens a circuit when it is pushed. This is important when inserting an extra control device to operate the lift. If a button closes a circuit when pushed, the new switch should be installed in parallel with the button switch and replicate its behaviour; that's to say, maintain the switch opened and close it for a moment in response to an activation code or a special event. If the lift button opens a circuit when it is pushed, the new switch should be installed in serial with the button switch, and it should be maintained closed until an activation code is detected or a special event happens.

A commercially available and inexpensive circuit board, which can decode DTMF tones into activation/deactivation of different relays was selected.

### 6.1.1 The NORCOMM NC400 Board

The board chosen for the task of decoding the DTMF sounds into relay activation/deactivation signals is the NC400 board of NORCOMM [80]. It was selected because it supports reliable DTMF sound recognition in signals from microphones placed in uncontrolled environments, rather than in low noise telephone lines and because it can generate sounds when it detects a valid DTMF tone sequence. This second aspect is very important as it allows bidirectional communication: from a machine to the device (the lift in the case) and from the device to a machine.

The NC400 board is equipped with three relays, which have been named, the primary,

the secondary and the momentary relay and two decoders, which have been named, the primary and the secondary decoder.

Two operation modes have been defined for the relays of the NC400.

**Timed mode:** The relay is activated by a DTMF code and automatically deactivates after a programed time delay.

**Latched mode:** The relay is activated by a DTMF code and it is deactivated from a different DTMF code.

All the relays of the NC400 boards used in the lift installation are used in timed mode.

#### The primary decoder for the primary and the momentary relays

The primary decoder controls the primary and the momentary relays. This means that the primary and the momentary relays are always activated by the same code, and thus, they can't be used to control two different buttons of the lift. The primary relay can operate in timed mode, where time for the relay to remain active after activation DTMF code detection, must be specified in minutes (from one to nine minutes, user programmable) or latched mode (where it is deactivated by another DTMF code). The momentary relay can only operate in timed mode where time before deactivation must be specified in seconds (one to nine seconds user programmable ). That's why it is called the momentary relay, because it can only remain active after activation DTMF code detection for not more than nine seconds.

The primary decoder has three activation addresses, which means, it can be controlled using three different codes to activate the primary and the momentary relay (but each code activates both the relays at the same time). It is also possible to deactivate the primary relay (if it is programed to operate in latched mode) using other three (same or different) DTMF codes.

#### The secondary decoder for the secondary relay

The secondary decoder controls the secondary relay. It can operate in timed mode where time period before deactivation must be specified in seconds (from one to nine seconds).

The secondary decoder has one activation address, thus, the secondary relay can only be activated/deactivated by one code.

#### Sound response

The NC400 board can generate sound when it detects a valid DTMF sound sequence. The sound it can generate, is not DTMF sound, however. The primary decoder generates short

beeps of a sound of frequency 1200 Hz/600 Hz alternately for one to twelve seconds (user programmable). The secondary decoder generates one beep at 600 Hz for two seconds.

## 6.2 Installation

As previously said, the lifts are usually operated using the buttons by the side of the door at each floor and inside the lift cabin. This is the case of one of the lifts of the Main Building of the San Sebastián Technology Park. This section explains the details of the installation done in this lift to make it possible to operate it by DTMF tones. The installation can be split in two parts:

1. Installation of a system in each floor to make it possible to call the lift using DTMF tones.
2. Installation of a system inside the lift cabin to make possible to operate it using DTMF tones.

The actual installation done to the lift makes it possible to operate it by DTMF tones at the ground floor and the first floor. In other words, two systems have been installed in the ground floor and the first floor, which make it possible to call the lift using DTMF tones from these two floors, and another system has been installed in the cabin, which allows the lift to be operated to go to the ground floor or the first floor using DTMF tones. The system installed in the lift cabin also allows to open and maintain the doors open for a specified amount of time using DTMF tones. This is useful for the robot MiReLa so that the doors don't close while it is getting out of the lift.

### 6.2.1 Installation of a System in the Ground Floor and in the First Floor to Call the Lift Using DTMF Tones




Figure 6.1: Picture of the lift door in the ground floor. It shows a detailed view of where the loudspeaker and the microphone have been installed.

For the installations done in the ground floor and in the first floor, the speaker has been placed in the upper part (at 180 cm from the ground) while the microphone has been placed in the lower part of the door (at 35 cm from the ground), see figure 6.1. The reason why this configuration has been chosen is because the robot MiReLa has its loudspeakers in the lower part of the robot at 31 cm from the ground, and its microphone is located at the top of the robot, at 141 cm from the ground (see section 7.1). The loudspeaker and the microphone at the first floor have been installed at the same height as in the ground floor.

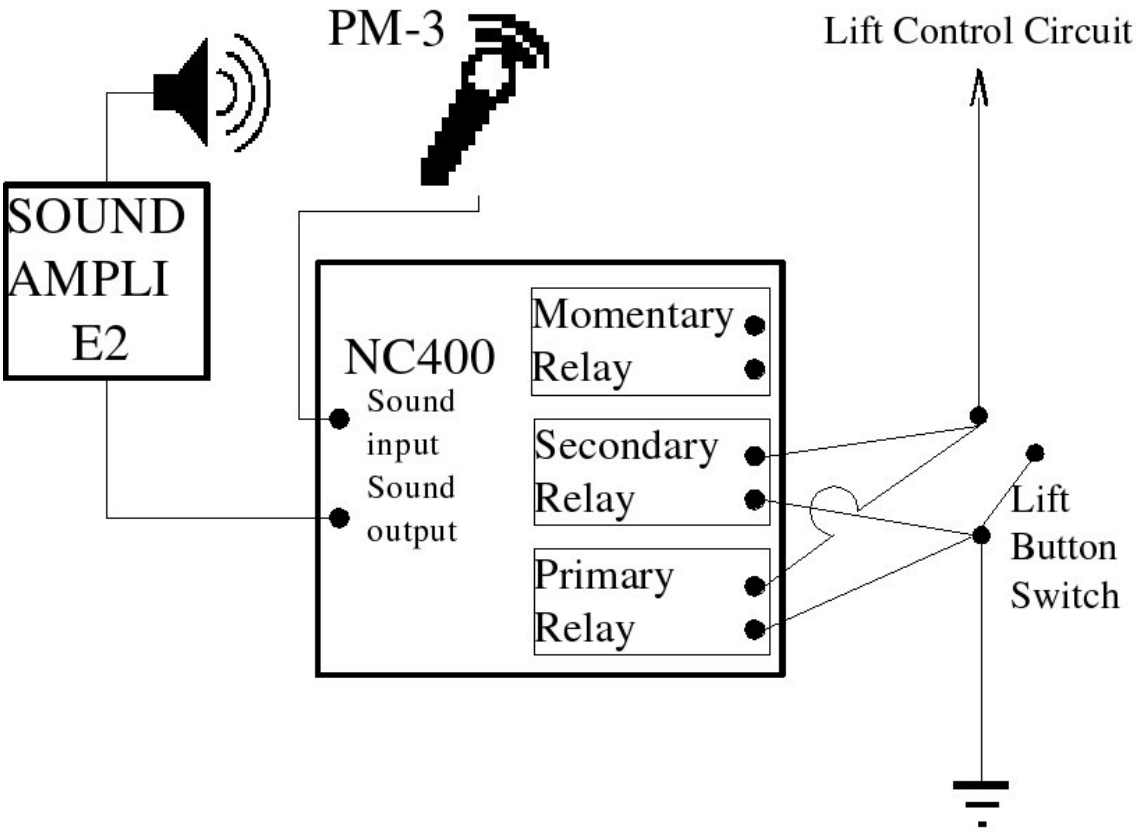



Figure 6.2: Block diagram of the components of an installation done for one floor, to operate, the button to call the lift, using DTMF tones. This diagram shows that two relays of the NC400 are connected in parallel to the button switch.

The installation done in each floor to operate the button to call the lift using DTMF tones is composed by one NC400 [80] board, a microphone with pre-amplification PM-3 [38] connected as input to the NC400 and a loudspeaker which receives the output of the NC400 amplified by an amplifier E-2 [37].

The installation of the device for each of the floors is identical. Two relays of the NC400 (the primary and the secondary) are connected in parallel with the lift call button. The primary relay has been programmed to work in timed mode, and to remain active for one minute after the activation DTMF sound code is detected. Although the primary relay, as it is controlled by the primary decoder, accepts three different activation/deactivation codes, only one has been used. The secondary relay has been programmed to work in timed mode, and to remain active for one second after the activation DTMF sound code is detected. This allows to operate the lift in two ways. One (using the secondary relay activation code), as if the lift call button was pushed for a second, which sends a petition to the lift to go to the floor at which the button is located. The second (using the primary relay activation code), as if the lift call button was pushed for a minute, which also sends a petition to the lift to go to the floor at which the button is located, but when the lift arrives, makes it maintain its doors open until the minute elapses. The second operation

mode is very useful to ensure the doors of the lift will be kept open until the robot MiReLa gets into the lift.

### Activation Codes

Table 6.1 shows the activation codes chosen for the devices installed at the ground floors and first floor. Only one digit has been used for each code. This is for reliability reasons. The geometry and materials of this part of the building mean that any sound produce a big echo, and detection reliability drops when sequences longer than one digit are used. Instead, using codes formed by one digit has proved to be reliable and robust.

| Floor | DTMF signal | Action |
|---|---|---|
| Ground | 5 | push button to call the lift for one second. |
| | 6 | push button to call the lift for one minute. |
| First | 3 | push button to call the lift for one second. |
| | 2 | push button to call the lift for one minute. |

Table 6.1: The DTMF signals used to call the lift in the ground floor and the first floor.

### 6.2.2 Installation of a System in the Cabin of the Lift to Operate It Using DTMF Tones




Figure 6.3: Picture of the lift cabin. It shows a detailed view of where the microphone and the loudspeaker have been installed in the lift cabin.

A system has been installed in the cabin of the lift to operate the buttons to go to the ground floor and first floor, and to keep the doors of the lift open, using DTMF tones. Figure 6.3 shows a picture of the lift cabin and a detail of the place where the microphone and the loudspeaker have been installed. They have been installed at a height of 150 cm from the ground of the cabin. This was the only place where they could be installed without making any modification to the lift cabin. In fact they were installed in a place reserved for a module for an additional alarm system in the cabin.

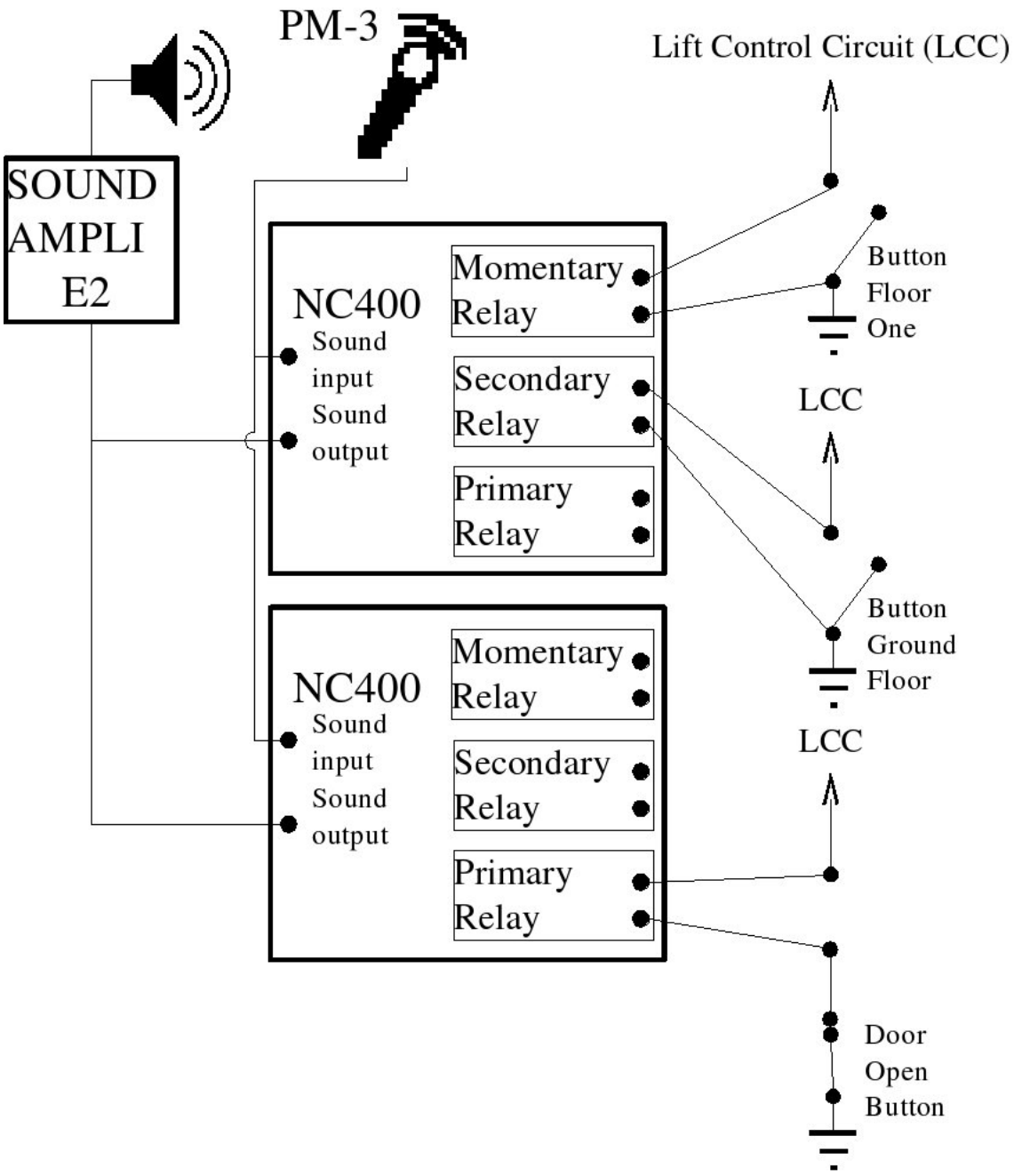



Figure 6.4: Block diagram of the components of the installation to operate the lift cabin to go to the ground floor, and the first floor, and to maintain the doors open, using DTMF tones. This diagram shows that two relays of one NC400 are connected in parallel to the button switch of the lift to go to the ground floor and the first floor and one relay of the other NC400 is connected in serial to the button switch to maintain the doors of the lift open.

The installation done in the lift cabin, to make possible to tell it to go to the ground floor, go to the first floor and maintain the doors of the lift open, using DTMF tones, consists of two NC400 [80] boards, a microphone with pre-amplification PM-3 [38] connected as

input to the NC400 boards and a loudspeaker which receives the output of the NC400 boards amplified by an amplifier E-2 [37].

The momentary and the secondary relays of one of the NC400 boards have been connected in parallel with the buttons to go to the ground floor and to go to the first floor respectively. They have been programed to close the circuit for one second when they detect their respective DTMF activation codes. The effect is as if the button to go to the first floor, or the button to go to the ground floor, were pushed for one second. From the other NC400 board, only the primary relay has been used. It has been connected in serial with the button to keep the doors of the lift open. As explained previously, this button opens a closed circuit when it is pushed, thus the relay has been connected in a way it usually maintains the circuit closed and it opens it when it is activated. It has been programed to operate in timed mode. When the activation code for this relay is detected it remains active (thus opens the circuit) for one minute. The effect is to maintain the door of the lift open for one minute. This is used by the robot MiReLa to be able to get out of the lift without the doors to be closed.

#### Activation Codes

Table 6.2 shows the activation codes chosen for the device. Inside the lift cabin, there is very little echo problem, compared to the waiting area outside. The use of sequences of three DTMF sounds for the activation of the relays has been tested and shown to be reliable. But, as only one digit codes were used outside the cabin, and as this makes the detection more reliable also inside the cabin, it was decided to use activation codes of one digit for the relays of the device installed inside the cabin.

| DTMF signal | Action |
|---|---|
| 1 | push button to open the door for one minute. |
| 9 | push button to go to the ground floor for one second. |
| 8 | push button to go to the first floor for one second. |

Table 6.2: The DTMF signals used to operate the lift, inside the lift cabin.

## 6.3 Concluding Comments

The fact that there are many inexpensive boards available on the market to decode DTMF sounds into the activation/deactivation of several relays makes possible to design and install (relatively easily) a system which makes it possible to operate the lift from DTMF tones. This has been considered as a first step towards operating the lift using the *MiReLa Music Language* (MML) presented in chapter 4. For recognition/generation of the MML a PC is needed. This makes it more difficult to install a system which can operate the

lift using the MML. But the installation presented in this chapter, allowed to test the viability of using sound in such variable and uncontrolled environments as are found in the waiting areas outside the lift and the area inside the lift cabin. It also helped to gain experience about how to install an extra system to operate the lift. See section 9.2.3 for an evaluation of the system.

For the installation done outside the lift cabin, it was possible to place the microphone at the same height of the loudspeakers of the robot MiReLa. Making it possible to be close to the sound source when MiReLa generate a DTMF tone to call the lift. This was not possible inside the lift cabin. Experience using the system showed that this fact is not important for reliability in the detection of the DTMF tones by the NC400. The sound echo is more determinant in this aspect and it is what makes the system detect DTMF codes more reliably inside the lift cabin than outside it. It has been noted that inside the cabin, the detection is more reliable when the doors are closed (echo is lower) and both outside and inside the lift cabin, the system works more reliably with people around as their bodies absorb part of the echo.

The common time the door of the lift remains open when it arrives to one specific floor (after someone calls it from the button at the floor or after someone operates it from inside the cabin) is eight seconds. This is not enough for the robot MiReLa to get into it or to get out of it. The robot needs about thirty seconds. The NC400 board doesn't offer the possibility of activating a relay for thirty seconds. As previously said, it allows some relays to be active between one and nine seconds, or between one and nine minutes. This is the reason why the relay to operate the button inside the cabin to open the doors and one of the relays to operate the button to call the lift in each floor have been programed to be active for one minute after detecting their respective activation code. The NC400 board proved to be reliable for the task it has been installed for. However, it would be interesting to have the possibility to control the time a relay remains active to be of thirty or forty seconds.

# Chapter 7

# The Robot MiReLa

MiReLa is a mobile robot (based upon a RWI (Real World Interface) B21 system) that is the centre piece of a research project of the San Sebastián Technology Park, see Figure 7.1.

The two basic aims of Project MiReLa, are robust navigation in semi-structured uncontrolled environments, and effective people-robot interaction in service robot scenarios. Two important aspects of this project are that it has adopted a purely Behaviour-based approach to developing a robust and reliable navigation competence for the robot, and that it has been conducted, from the start, in a real place, *not* in a laboratory: a real place where other people work, public events take place, large changes occur as part of its normal usage, and where there is "musak" (background piped music) everywhere.

MiReLa has been developed to act as a kind of artificial guide, that can take people from the main entrance of the building to other places, such as the bar (on the ground floor), auditorium or seminar rooms (on the first floor), or to people's offices (on the second and third floor), [51]. To aid and support this role, MiReLa uses an artificial language based upon words made up of musical notes (see chapter 4): and this is where its name *Mi-Re-La* comes from (the notes e, d, A).

- Section 7.1 explains the hardware of the robot.
- As previously mentioned, robust navigation in semi-structured uncontrolled environments, and effective people-robot interaction in service robot scenarios are being investigated with the robot MiReLa. Most of the times interaction with the robot is done to tell it to navigate from one place to another. A robust and effective way to instruct the robot at any time to navigate to different places has been investigated. Section 7.2 explains how this has been achieved. This was published in [50].
- Interaction with the robot is multi-modal. It can be accessed by the *MiReLa Music Language* (explained in chapter 4 ), from a Web page , and from the Information Point. To control and integrate all the different aspects that need to be taken into account in a service robot such as MiReLa, a *Society of Agents* has been created. Section 7.3 explains the agents developed for the robot MiReLa. The Web page (section 7.3.11) and the Information Point (section 7.3.12) are two of the agents in

the society. Currently twenty-three agents run and interact with each other in the robot MiReLa.

## 7.1 The robot

MiReLa is a RWI B21 [85] robot, manufactured by Real World Interfaces, Inc. It consists of a base and an enclosure. The enclosure is mounted on top of the base and rotates with respect to it. MiReLa is equipped by four wheels which make it able to move. They all rotate together with the enclosure. This gives the robot two degrees of motion: linear (forwards and backwards) and rotation (about its central axis).




Figure 7.1: Front view and back view of the robot MiReLa.

The robot base is the lower part of the B21 robot. It is 54 cm in diameter and 31 cm in height. It is covered by eight panels, each equipped with bumper and infra-red sensors. It houses the four 12 V batteries that supply all the electrical current used by the different components of the complete robot.

The enclosure, on top of the base, is 50 cm in diameter and 85 cm in height. On top of

the enclosure two cameras are mounted. The base, the enclosure and the cameras, make the robot MiReLa 1.44 m tall (see figure 7.1). It is rounded by six panels equipped with bumper, ultra-sound, and infra-red sensors.

The sensors, the robot MiReLa is equipped with, make the link of the robot with the environment it is in. They are the ultra-sound sensors, the infra-red sensors, the laser range finder sensor, the bumper sensors, the compass sensor, the odometry sensor and the microphone. The first two kinds of sensors (ultra-sound, laser range finder) are time of flight sensors. they generate a pulse wave, whose propagation speed is known (sound or light, for example) and they measure the elapse time until they receive an echo [18]. In the next subsections the components of the robot MiReLa are presented.

### 7.1.1 Electrical Power

MiReLa is equipped with four batteries of 12 V each. The robot base houses them. They are connected in serial, thus, they can give 48 V. They supply all the electrical current used by the different components of the complete robot. The 48 V are transmitted to the enclosure which houses the converters to deliver the 24 V, 12 V, 7.5 V and 5 V, needed by the different components of the robot. MiReLa can operate for four hours without re-charging its batteries.

### 7.1.2 Computer System of MiReLa

MiReLa is equipped with three computers: two PCs with dual base boards (SMP - "Symmetric Multi-Processing") equipped with *Pentium Pro* processors at 200 MHz and with 64 Mb of RAM memory, and one Apple Macintosh PowerBook G4 laptop. The two PCs are located inside the robot enclosure and have been named Guest2 and Guest3. The Laptop is placed on top of the enclosure and it has been named Guest4 (see figures 7.1 and 7.3 ). They are connected by a switch in an *Ethernet* net at 100 Mbits per second.

### 7.1.3 Ultra-sound Sensors

The ultra-sound sensors are active proximity sensors which measure the time of flight of a reflected sound wave of a frequency higher than humans can hear [43].

MiReLa has twenty-four ultra-sound sensors, placed at the height of 70 cm from the ground and separated 15° (see figure 7.2). Not every sonar can fire at once, as they might be confused by the sound waves simultaneously fired off by their immediate neighbours. They fire in a pre-set pattern, usually four or six sonars at a time (at most one sonar for each panel). MiReLa gets reading from its full set of ultra-sound sensors (24 sensors) about three time per second (3 Hz).

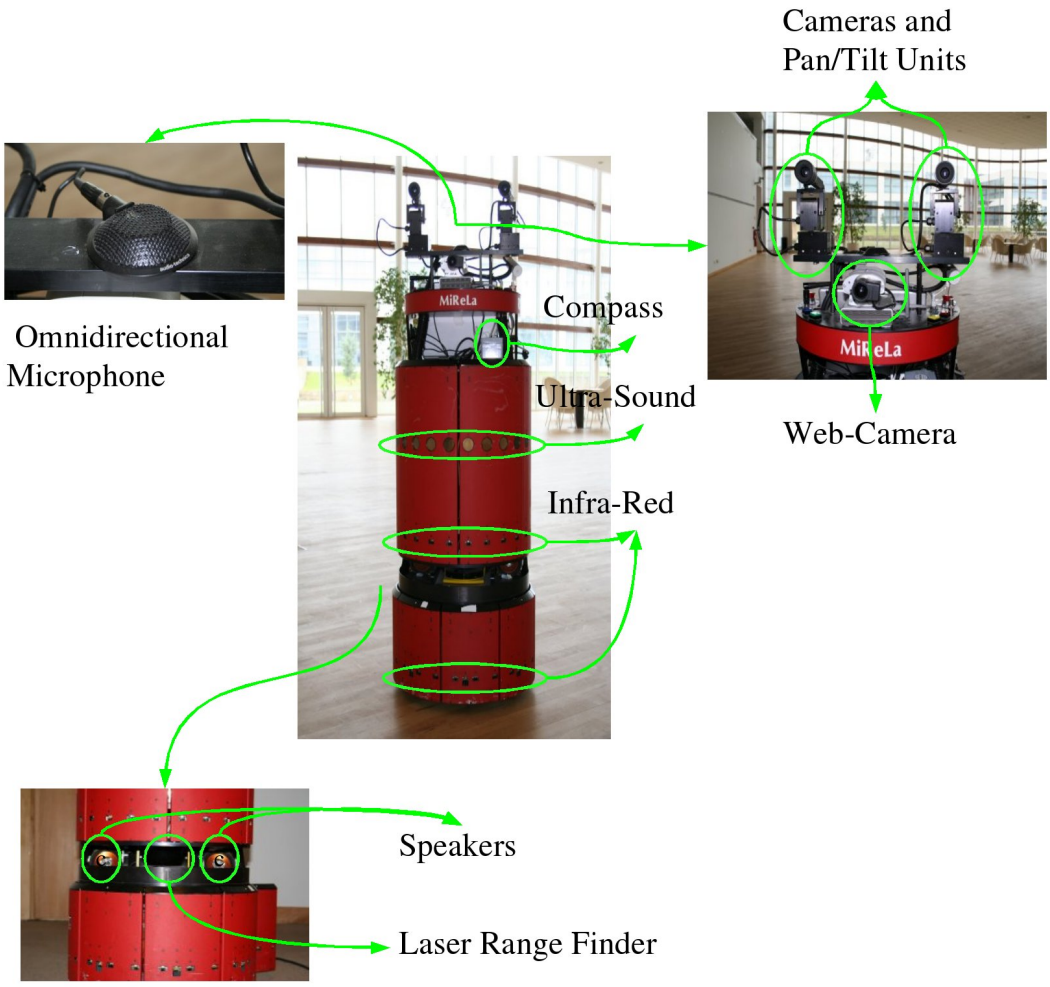



Figure 7.2: Picture indicating the location of the sensors in the robot MiReLa.

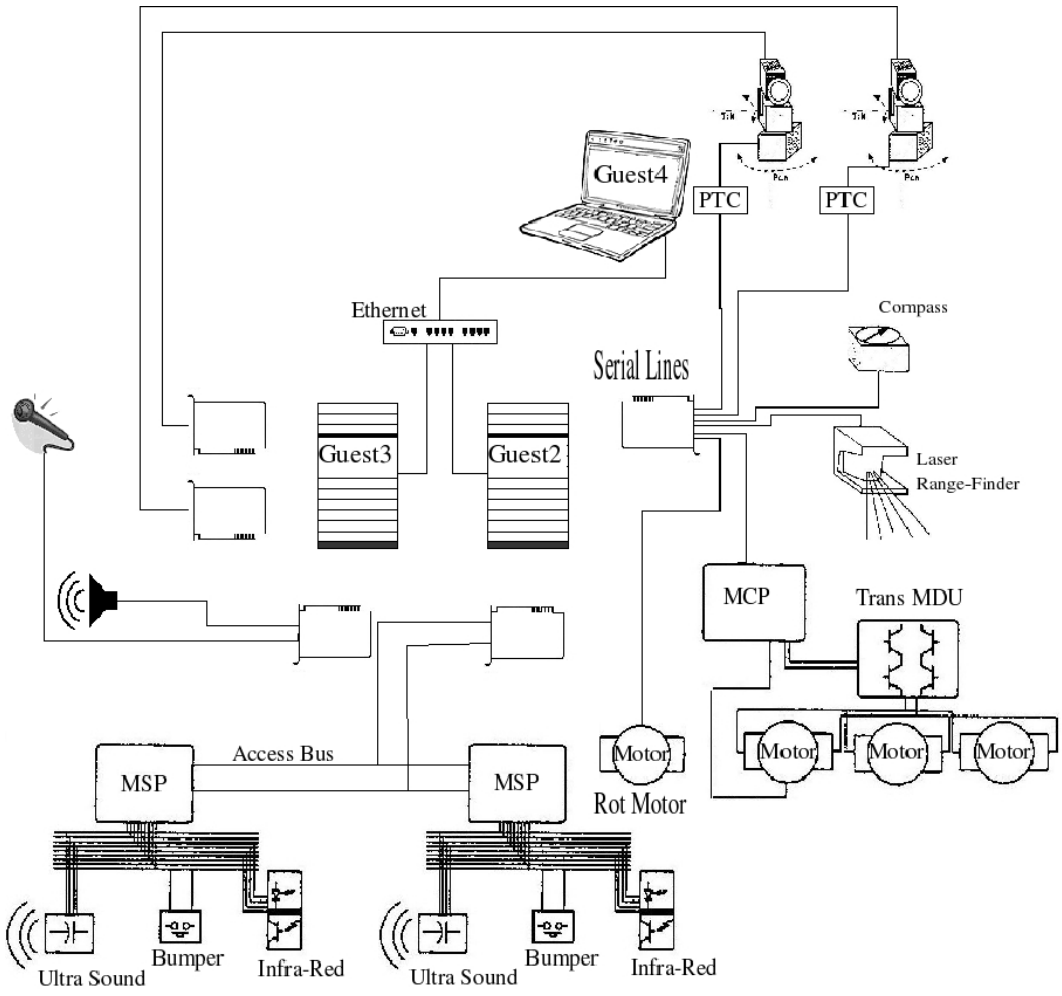



Figure 7.3: Hardware architecture of the robot MiReLa.

### 7.1.4 Infra-red Sensors

The infra-red sensors of the robots B21 are active proximity sensors. They operate by emitting infrared light, and detecting any reflections from distant surfaces. To differentiate the emitted IR from ambient IR from artificial lighting or the sun, the emitted signal is usually modulated with a low frequency (for example 100 Hz) [77]. The intensity of the emitted light is compared with the intensity of the received light in return. The intensity of the light in return becomes bigger as a reflecting surface gets closer to the sensor [66]. The infra-red sensors of MiReLa detect reflecting surfaces acceptably when they are at distances closer than about 30 cm from the sensor. But even then the colour of the reflecting surface is important. If it is of black colour it absorbs the light more than if it is of white colour, thus the intensity in return is lower and consequently it is more probable the sensor to fail in detecting it .

There are infra-red sensors located in the robot base and in the robot enclosure (see figure 7.2).

**Robot base:** These are all located at the same height, they are arranged to form two rings: one consisting on twenty-four infra-red sensors directed in parallel to the floor; the other, formed by eight infra-red sensors directed towards the floor at an angle of 45°.

**Robot enclosure:** Twenty-four infra-red sensors are located between the ultra-sound sensors and the laser range finder.

### 7.1.5 Laser Range Finder

MiReLa is equipped with a SICK-PLS (Proximity Laser Scanner) laser range finder situated on top of the base, just on the bottom of the enclosure. It is attached to the enclosure and it scans a 180° wide area. The resolution can be configured to scan every 0.5° or 1°. It has been configured to scan every 1° for the robot MiReLa. It is a time of flight sensor, composed by a laser photodiode emitting infra-red light with a wave length of $900 \pm 10$ nm, which is directed by a set of mobile mirrors. It can obtain 180 reads in 40 ms but it needs 80 ms before doing a new scan. The Laser range finder is connected to the computational hardware via a serial connection. The maximum range of scan is 50 m, but it has been limited to 5 m (by software) in MiReLa as the robot motion is not wanted to be influenced by laser inflections from surfaces that are not very close to it.

### 7.1.6 Bumper Sensors

The robot MiReLa has four bumper sensors in each panel of the robot enclosure and of the robot base: one at each corner. The enclosure is covered by six panels and the base is covered by eight panels, thus, the robot has 56 $(4 \times (6 + 8))$ bumper sensors.

The bump sensors of MiReLa are simple switches. A flexible piece of metal is placed in each of the corners of the panels covering MiReLa. When a bump or contact happens this

piece of metal bends to close a circuit. Depending on the circuit closed the robot can detect in which panel the bump happened.

### 7.1.7 Electronic Compass

MiReLa is equipped with an electronic compass KVH-C100. It is located on the top of the enclosure. It detects the orientation of the local magnetic field relative to the robot. In the absence of any strong artificial field, this is the earths magnetic field. In the presence of artificial magnetic fields, that result from electrical equipment typically installed in buildings, the detected field is the combination of the earth's field and the artificial field. As some electronic devices (such as loudspeakers) produce a magnetic field which can disturb the measurement of the compass, it is important to place the compass in such a position on the robot that this disturbance is minimised. The KVH-C100 is a fluxgate compass (see [77] for a detailed explanation about fluxgait compasses).

The KVH-C100 compass has a resolution of 0.1°, and it can produce readings at 10 Hz maximum. It is connected to the computer from a serial connection.

### 7.1.8 Sound Processing Devices

Sound is captured by an omni-directional microphone (Audio Tecnica AT8531) connected to the sound card of the Guest3 computer. Output sound is generated from two loudspeakers (Pioneer TS-G1016) which get the sound signal from an amplifier (CEBEK ES4) connected to the sound card. See figure 7.2.

### 7.1.9 Cameras and Pan Tilt Units

On the top of the enclosure, two cameras are located. They can be controlled, independently, by two PTUs (Pan & Tilt Unit). The movement of one of them is used for gesture implementation in the robot (see the GestureServer agent section 7.3.7). The other is used for artificial vision purposes (see the VisionPeopleDetector agent section 7.3.15). Between these two cameras, a Web cam (axis 2120) has been installed which is used to transmit images via the network to other computers. Finally, next to the laptop, an iSight (Apple Macintosh) camera is installed which is connected to the Apple Macintosh G4 PowerBook laptop, it can be used to support video conference over the network to other computers. See figures 7.2 and 7.3.

### 7.1.10 Motion Control

The robot MiReLa uses four DC motors of 48 V for translational and rotational purposes: three for translation and one for rotation. A system of gears and belts makes it possible to transmit the rotation movement of the three translation motors to the four wheels of the robot MiReLa and thus make the robot able to translate forward and backward. The

movement of the rotation motor is translated via another system of gears and belts to the four wheels and the enclosure which rotate all together. See [19] for more information.

#### Translation Control – MCP/MDU

The MDU (*Motor Driver Unit*) of MiReLa is a H-bridge designed to provide a pulse-width modulation control of motors that draw high currents. A pulse-width modulation control, controls the motor speed by driving the motor with short pulses. These pulses vary in duration to change the speed of the motor. The longer the pulses, the faster the motor turns, and vice versa. One MDU controls the three translation motors. See figure 7.3.

The MDU is connected to the MCP (*Motion Control Processor*) which makes the link to the Guest2 computer of MiReLa through a serial line. There is one encoder installed in one of the translation motors. This encoder is also connected to the MCP. Thus the translation orders are transmitted from the Guest2 to the motors via the MCP and the odometry information is also obtained from the MCP.

#### Rotation Control

Originally there was a MDU connected to the rotation motor which was controlled from the MCP. In other words, the same MCP controlled the MDU for rotation and the MDU for translation. In January 2004, one MDU failed. With only one MDU operative, it was only possible to control the rotation or the translation of the robot, depending on where was the MDU placed. It wasn't possible to get a replacement from RWI as it ceased to exist as a company. After some unsuccessful attempts to repair the broken MDU, a decision was taken to change the whole rotation control system.

One motor JVL Mac95 [6] is now used for rotation. It is a DC motor of 48 V and it has an encoder integrated. It is connected to the Guest2 from a serial connection. The three PCs of MiReLa are placed in the enclosure while the motors are placed in the base. The enclosure rotates relative to the base, thus, any cable connection between them must be specially treated. B21 robots incorporate a electrical rotary connector for the pre-designed base-enclosure connections. But adding a new connection doesn't have a trivial solution in these robots. The decision was to use a wireless Bluetooth based RS232 cable. The installed device was the "AIRcable Serial to Serial" [15].

### 7.1.11 Multi Sensor Processors MSPs

The ultra-sound sensors in the enclosure, the infra-red sensors and the bumpers, in the base and in the enclosure, are connected in groups to the MSPs (Multi Sensor Processor). The MSPs read the data generated by these sensors and they supply them, with tension and clock signals. They are connected to the Guest2 computer using a *Philips* $I^2C$ (Inter-Integrated Circuit) serial computer bus.

## 7.2 Navigation System

### 7.2.1 Motion Control

The motion control architecture is a result of the PhD thesis of Iñaki Raño [83]. This subsection is designed to present a brief summary of this architecture in order for the reader to be able to understand the next subsection, which explains what has been developed in the thesis work being presented here, to make it possible to be able to instruct the robot to navigate to different places at any time. This is important to understand some of the agents of the *Society of Agents* of MiReLa presented in section 7.3.

The motion control architecture consists of a (non-hierarchical) set of *Filters*, and a motor-fusion component. It is a purely Behaviour based architecture [29], [30],[5],[28],[31]. Each *Filter* takes real time data from one or more of the different sensor systems (infra-red sensors, ultra-sound sensors, laser range finder, electronic compass, or bumpers) and outputs two (normalised) speed quantities: $\alpha_f$ and $\beta_f$; $\alpha_f$ for the robots forward speed and the $\beta_f$ for the robots rotational speed. (The position of the robot, with respect to some frame of reference, is *not* explicitly controlled.) The output $\alpha_f$ values from the different filters are combined to form a total $\alpha_m$ that determines the actual forward speed of the robot and the $\beta_f$ values are combined to form a total $\beta_m$ that determines the actual rotation speed of the robot.

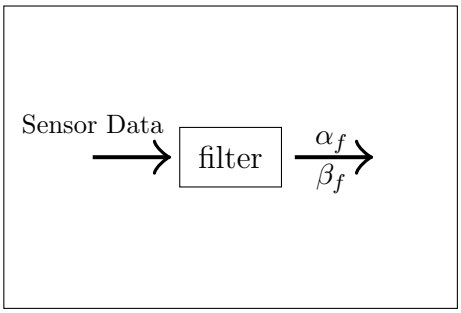



Figure 7.4: A filter gets data from the sensors and processes it to get two values, an $\alpha_f$ (forward speed) and a $\beta_f$ (rotational speed).

Each *Filter* is designed, developed, and tested to support a particular kind of capability, such as: ultra-sound-based obstacle avoidance; laser-based obstacle avoidance; door passing; etc.

The $\alpha_f$ and $\beta_f$ values from each *Filter* are combined using the *harmonic mean* expression, see [88].

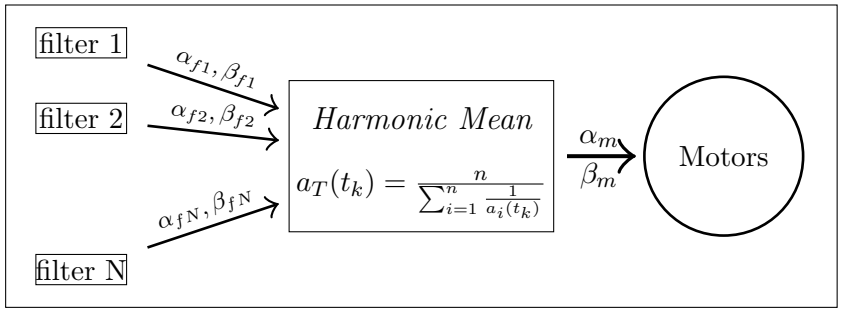



Figure 7.5: The $\alpha_f$ values $(\alpha_{f1}, \alpha_{f2}, ..., \alpha_{fn})$ are combined in a harmonic mean to get a unique $\alpha_m$, and the $\beta_f$ values $(\beta_{f1}, \beta_{f2}, ..., \beta_{fn})$ are combined in a harmonic mean to get a unique $\beta_m$. The resulting $\alpha_m$ is the forward speed sent to the translation motors and $\beta_m$ is the rotational speed sent to the rotation motors.

The motion behaviour of the robot in any given moment, depends on the filters chosen to be combined.

For example, the robot may be required to move about five metres in North direction without bumping into obstacles. This would be achieved by combining a laser based obstacle avoidance *Filter*, and ultra-sound based obstacle avoidance *Filter*, together with a distance integration *Filter*, and a compass *Filter*. Or the robot may be required to pass through a door. Then, a laser based door passing *Filter* would be used. Or it may be required to stop and wait until a specific signal is received. Then a stop *Filter*, generating a zero $\alpha_f$, would be used.

To control the time and the duration of each behavioural state, a structure called a *Transition* was created.

A *Transition* is specified in terms of a set of *Filters* and a set of *Termination Conditions*, see figure 7.6. Thus it specifies which *Filter*'s $\alpha_f$ and $\beta_f$ to add to the combination rule. *Termination Conditions* define detectable states that are generated upon successful completion of a specific condition (for example, an odometry integration value of 5 m in the specified transition direction) or by some known failure condition. When one of the *Termination Conditions* is satisfied, the *Transition* terminates and the robot starts the next *Transition*.

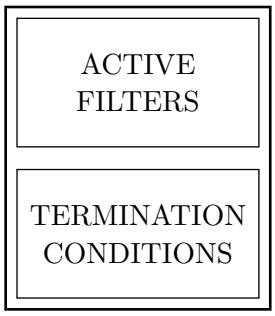



Figure 7.6: A *Transition* specifies the active *Filters*: the *Filters* whose $\alpha_f$ and $\beta_f$ values need to be combined; and the termination conditions: the conditions which, if one is satisfied, terminate the *Transition*.

### 7.2.2 Navigation

An early development in the research done for this thesis, was to implement a reliable and a simple way to instruct the robot, at any time, to go to different places in the Main Building of the San Sebastián Technology Park.

MiReLa, moves from one place to another by completing a set of *Transitions*. In order to be possible to instruct the robot, at any time, to go to different places, MiReLa must be able to select, in real time, different sets of *Transitions* to complete.

A structure called a *Transition Group* was created. This defines a sequence of *Transitions*. To do a *Transition Group* is for the robot to complete the sequence of *Transitions* it defines from the first to the last. A *Transition Group* also defines the ANTG set (Allowed Next *Transition Groups*), which are the *Transition Groups* the robot can start doing when it completes all the *Transitions* of the current *Transition Group* (See figure 7.7).

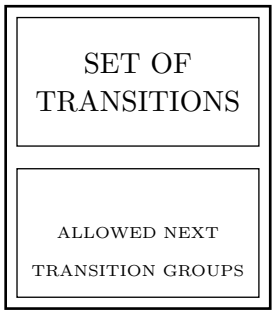



Figure 7.7: Each *Transition Group* specifies a set of *Transitions* and the *Transition Groups* the robot is allowed to start doing next, ANTG (Allowed Next *Transition Groups*).

From the ANTGs of the *Transition Groups* a directed graph can be drawn. The robot is allowed to do in sequence those *Transition Groups* connected in the graph. For example, in the illustration graph presented in figure 7.8, if MiReLa is currently doing TG5, to be

doing TG2, it must first do TG1 and then TG3. Or it can be doing TG2 by first doing TG4 and then TG6.

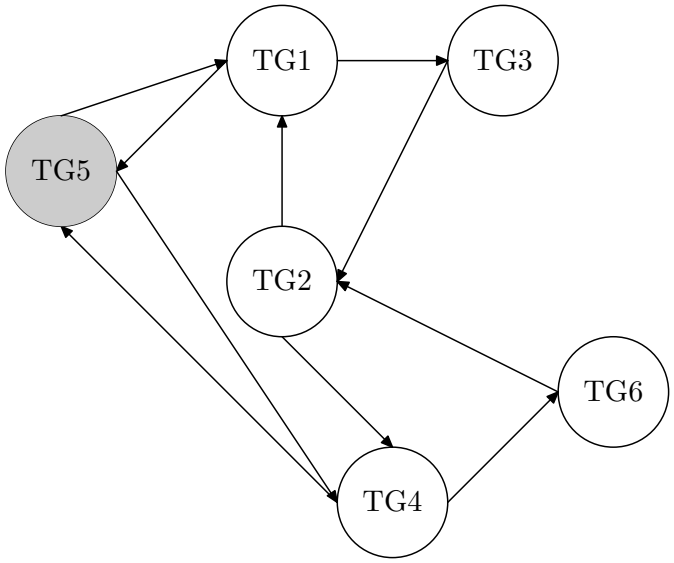



Figure 7.8: A general graph made from the ANTGs of the *Transition Groups*.

MiReLa is always *doing* a *Transition Group*, even if *appears* not to be doing anything: a MiReLa *Transition* does not necessarily have to cause MiReLa to move. The way to make MiReLa find its way to some other place, is to then tell it *to be doing* a *Transition Group* that it can *only* be doing after having completed all the *Transition Groups* in the network between its current *Transition Group* and the specified *Transition Group*, and where successful completion of all the intermediate *Transition Groups* results in the robot actually moving to the specified place. Navigation from place to place, is thus a *side effect* of telling MiReLa to be doing certain *Transition Groups*: we see MiReLa find its way to the place we want it to go, but MiReLa has no proper sense of going anywhere; it just checks each *Termination Condition* of each *Transition* that form the *Transition Groups*. When a *Transition Group* is done it starts the next *Transition Group* in the identified sequence of *Transition Groups*. It is the designer's job to build *Transition Groups* which make the robot behave in the desired way.

To facilitate this use of the *Transition Group* graph for navigation, and to support the possibility for users to "tell" MiReLa where to go (by telling it which *Transition Group* to be doing), a special type of *Transition Group* was defined, called WNTG (Wait for New *Transition Group*). WNTG type *Transition Groups* have a particular kind of completion condition which is only satisfied when MiReLa is "told" to be doing another (or the same) *Transition Group*. The way to make MiReLa always be doing a *Transition Group* is to "tell" it to do only WNTG type *Transition Groups*. In effect, what this means is that MiReLa will complete a series of *Transitions* (grouped in several *Transition Groups*) that take it to some place, and then wait until it is instructed to be doing some other *Transition Group*, that may result in it going somewhere else.

### *Transition Groups* type WNTG and CTG

Two types of *Transition Groups* are thus defined, a WNTG (Wait for New Transition Group) type, and a CTG (Common Transition Group) type. Any *Transition Group* must be one of these two types.

**WNTG type *Transition Group*:** Is one which satisfies the following two properties:

1. *All* the *Transitions* that are defined as part of it, *must* produce a zero $\alpha$ value.
2. It must include at least one *Transition* with only one active *Termination Condition*, which is set to *true* only when another *Transition Group* of type WNTG is selected to be doing.

**CTG type *Transition Group*:** This type includes any kind and number of *Transitions*, except those which satisfy property 2 of *Transition Group* of type WNTG.

MiReLa can only be asked to be doing a type WNTG *Transition Group*. The sequence of *Transition Groups* that must be completed, in order for MiReLa to go from doing one WNTG type *Transition Group* to be doing another WNTG type *Transition Group* must all be of type CTG. Having a *Transition Group* of type WNTG in this sequence, would make the robot stop and wait for a new order before starting to do the required *Transition Group*.

The most common thing is that a *Transition Group* of type CTG produces robot translation, that's to say, it results in MiReLa moving from one physical place to another. But not all the CTG type *Transition Groups* involve translation. There are configurations where the robot is required to do other things. For example a CTG type *Transition Group* might make MiReLa stop, wait until it detects a person and turn to where it detected the person.

### Generating a graph from the names of the *Transition Groups*

Up to this point, a way to generate a graph has been explained, which is achieved by completing the ANTGs of each *Transition Group*. This should be done in a configuration file from where the robot could get the information. But having to specify the *Transition Groups* it is allowed to start doing next in each *Transition Group*, makes the process, at least, time consuming and generally more opened to errors. When any new *Transition Group* is added, the need to modify, not only the ANTG of the new *Transition Group* but also the ANTGs of all the *Transition Groups* which will allow the new *Transition Group* to start doing after themselves, arises.

Some rules were defined to name the *Transition Groups* and thus help to semi-automate the generation of the graph. This way, the ANTGs of each *Transition Group* are completed from the names of the *Transition Groups*. That's to say, a graph is generated from the names of the *Transition Groups*.

The structure of the names of the two types of *Transition Groups* is defined as follows:

- CTG type *Transition Groups*: Name1:Name2. Where Name1 is a name for the place the robot is expected to be when it starts doing the CTG type *Transition Group*. And Name2 is a name for the place the robot is expected to be when it ends doing the *Transition Group*. Doing a CTG type *Transition Group* commonly makes the robot translate. But it is possible for the robot to do a CTG type *Transition Group* without translating itself. Thus Name1 and Name2 will be different, but the place still be the same. As previously said, completing the *Transition Groups* may make the robot move to different places, but MiReLa does not do any association checking between the physical place and the *Transition Group* it is doing.
- WNTG type *Transition Groups*: Name:Name. Where Name is the place the robot is expected to be in when it starts and ends the *Transition Group*. As previously explained, if the robot is doing a WNTG type *Transition Group* it doesn't translate, thus it ends the *Transition Group* at the same place it started it.

The ANTGs are completed as follows:

| Type of *Transition Group* | Name structure | Generation of the ANTG |
|---|---|---|
| CTG | Name1:Name2 | Include the *Transition Groups* of type CTG starting with Name2 |
| WNTG | Name:Name | Include the *Transition Groups* of type CTG starting with Name |

Table 7.1: How the ANTGs of the *Transition Groups* are completed.

Note that in a graph generated from these rules, it is not possible to start from doing a WNTG type *Transition Group* and end doing another WNTG type *Transition Group* in one step. This is because there isn't any WNTG type *Transition Group* included in the ANTGs of CTG type *Transition Groups*, nor in the ANTGs of WNTG type *Transition Groups*. That is to ensure that the sequence of *Transition Groups* that must be completed, in order for MiReLa to go from doing one WNTG type *Transition Group* to be doing another WNTG type *Transition Group* are all of type CTG. In other words, from the graph generated from the ANTGs created in this way, a *Transition Group* of type WNTG can only be the starting point of any sequence. Thus, a second step is needed to make the sequence end with a *Transition Group* of type WNTG.

The steps required to find the sequence to do in order to bring the robot from doing a *Transition Group* of type WNTG to another *Transition Group* of type WNTG, will be illustrated with an example:

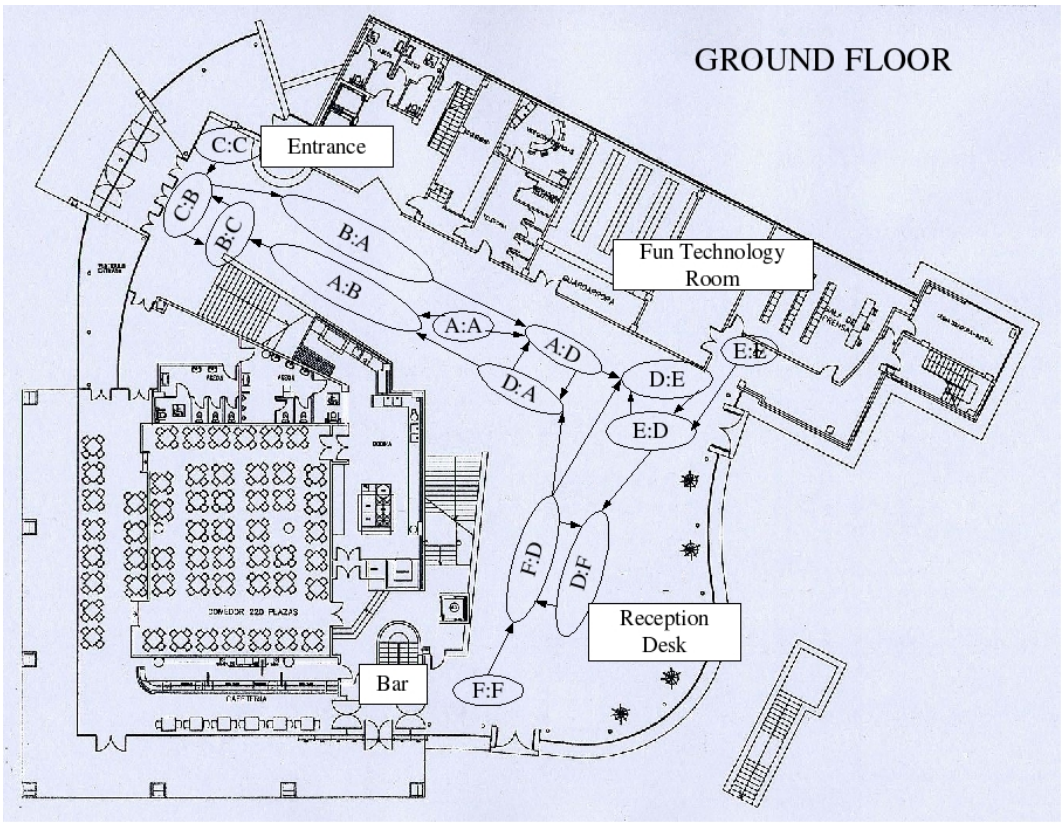



Figure 7.9: Plan of the ground floor of the Main Building of the San Sebastián Technology Park, with the representation of the *Transition Group* graph implemented in MiReLa for the 2003 Open Day (see section 8.1.2).

Figure 7.9 shows the plan of the the ground floor of the Main Building of the San Sebastián Technology Park, where MiReLa operates. It is represented the graph the robot used for the **2003 Open Day** trial. (Section 8.1.2 explains the preparation and results of it.) It is formed from the names of the following *Transition Groups*:

| *Transition Group* of type WNTG | *Transition Group* of type CTG |
|---|---|
| A:A, C:C, E:E, F:F | A:B, B:A, B:C, C:B, B:C, A:D, D:A, D:E, E:D, D:F, F:D |

Table 7.2: *Transition Groups* used for the 2003 Open Day.

Suppose the robot is located at Reception (see figure 7.9), thus, doing *Transition Group* of type WNTG C:C and it is required to go to Fun Technology Room, thus, to be doing *Transition Group* of type WNTG E:E.

The breath first search algorithm [27] is used, starting at C:C and with the specification to stop at a *Transition Group* which ends in E.

The sequence of *Transition Groups* found by the algorithm in step 1, is:

C:C→C:B→B:A→A:D→D:E

Then the *Transition Group* of type WNTG, E:E is added to it, thus the resulting sequence to do to bring the robot from Reception to Fun Technology Room is:

C:C→C:B→B:A→A:D→D:E→E:E

## 7.3 A Society of Agents for MiReLa

This section presents all the agents which form the *Society of Agents* of MiReLa. To present them, the same structure employed in section 5.2 is used. Figure 7.10 shows a representation of all the agents which run in the robot. It shows the data generated by each agent and the instructions accepted by them, and represents the connections to get data from and to send instruction to the other agents. The agents are presented in subsections of one or two pages each. One subsection for each agent. Each subsection starts with a table indicating the skill the agent adds to the system, the type of instructions it accepts, the data it generates, the type of instructions (indicating to which agents) it sends, and the data types it generates. Then the operation of the agent is explained.

The FFTServer, NoteChecker, WordChecker, WhistleHelper, SoundServer, and AmbientNoise agents form part of the *Society of Agents* implemented to control lights. They have been described in section 5.2, thus they are not described here again. But they are also part of the *Society of Agents* implemented for MiReLa, they make MiReLa able to use the *MiReLa Music Language*.

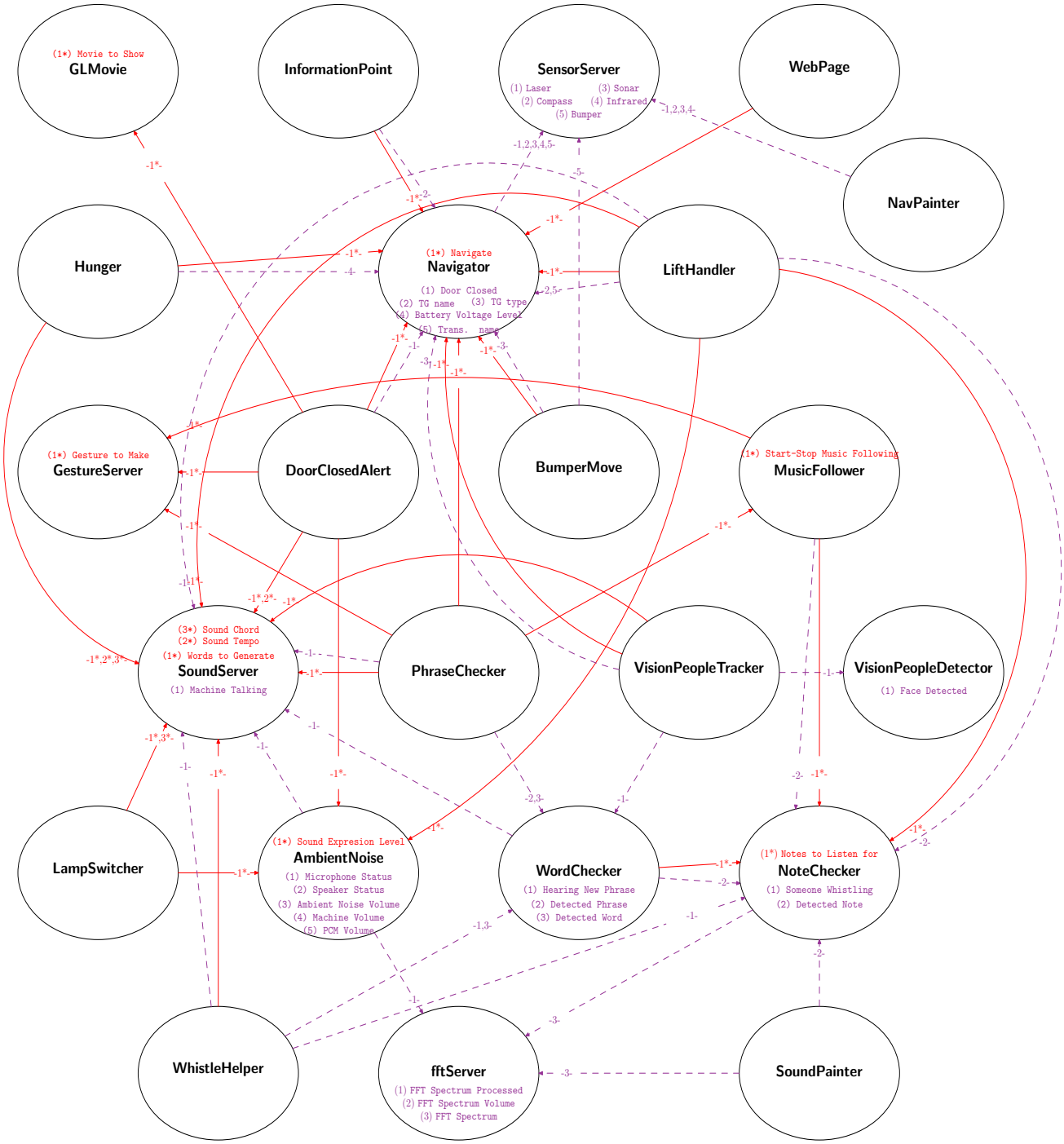



Figure 7.10: Representation of the agents of the *Society of Agents* implemented in the robot MiReLa. Section 3.2 explains the meaning of the different types of arrows (dashed or continuous), their labels and the nature of the names written at each agent (agent name, accepted instructions, and generated data).

### 7.3.1 The Navigator Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Make the robot able to Navigate | `Navigate` |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| `Bumper` →7.3.2 SensorServer | `Door Closed` | |
| `Compass` →7.3.2 SensorServer | `Transition name` | |
| `Sonar` →7.3.2 SensorServer | `Transition Group name` | |
| `Infrared`→7.3.2 SensorServer | `Transition Group type` | |
| `Laser` →7.3.2 SensorServer | `Battery Voltage Level` | |

Table 7.3: Skill Table for the Navigator agent.

#### Description:

The Navigator agent makes the robot able to navigate from one place to another. It controls the motors in the way explained in section 7.2 (sending them an $\alpha_m$ forward speed and a $\beta_m$ rotational speed), and it uses a *Transition Group* graph created from a text file to navigate, as explained in the same section 7.2.

It requires data from the sensors. These data are served by the SensorServer agent (see section 7.3.2), thus the Navigator agent requests them from it.

If the robot intends to pass through a door, the Navigator agent can detect if it is closed. If the door is closed, variable (a boolean) `Door Closed` is set to true. If the door is opened `Door Closed` is set to false.

The Navigator agent also generates the `Transition name`, `Transition Group name` and `Transition Group type` data. They are strings of characters. `Transition name` indicates the name of the *Transition* the robot is currently doing. `Transition Group name` indicates the name of the *Transition Group*, and `Transition Group type` the type (`Transition Group type`: WNTG or CTG) of the *Transition Group*, currently being done.

The Navigator agent also calls to a base controller function to get the current battery voltage level. This is served to any agent requiring it in the form of the `Battery Voltage Level` data.

The Navigator agent can be instructed by sending it `Navigate` instructions of the following type:

- Be_doing *Transition Group*: *Transition Group* is the WNTG type *Transition Group* an agent may require the robot to end up doing (section 7.2 explains how is this

calculated).

- Be *Transition Group*: Makes the robot immediately start to do the *Transition Group* of type WNTG requested. This is useful, for example if the Navigator agent starts up doing a *Transition Group* which doesn't correspond with the place the robot is in the given moment.
- Pan_+: The robot stops and starts rotating in clock-wise direction, that's to say, it starts panning.
- Pan_-: The robot stops and starts rotating in counter-clock-wise direction.
- Pan_0: If the robot was doing a pan movement, it stops doing it and continues with the *Transition* it was doing before starting to do the pan.
- turn_<degree>: Makes the robot change the *Transition* it is doing to a *Transition* to turn the degrees specified. For example, turn_90 makes it turn $90^o$ left. Then it continues with the *Transition* it was doing before.
- Finish Transition: When this instruction is received, if the *Transition* being done has the *Termination Condition* Finish *Transition* when signal received activated, the *Transition* is terminated.

### 7.3.2 The SensorServer Agent

| **Skill** | | **Accepted Instructions** |
|---|---|---|
| Serve Sensor data | | — |
| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
| | `Bumper`<br>`Compass`<br>`Sonar`<br>`Infrared`<br>`Laser` | |

Table 7.4: Skill Table for the SensorServer agent.

#### Description:

The SensorServer agent accesses the bumper, ultra-sound, infra-red, laser range finder, and compass sensors of the robot. As explained in section 7.1, the MSP (Multi Sensor Processor) accesses the ultra-sound sensors in the enclosure and the infra-red and the bumper sensors in the enclosure and base. The SensorServer agent accesses the MSP and also the laser range finder and the compass sensors. It gets data from the ultra-sound, infra-red, laser range finder, and compass sensors at a frequency of 10 Hz maximum, and if the speed of the data transfer rate of the connection permits, they are served, to the agents requesting them, also at 10 Hz (refer to section 3.3 for a detailed explanation about how are data served). The `Bumper` data are served when a bump event is detected in any of the base and enclosure panels.

Table 7.5 shows the data types obtained from the different sensors.

| **Sensor** | **Data** |
|---|---|
| Laser range finder | Array of 180 integers |
| Ultra-sound | Array of 24 integers |
| Infra-red Enclosure | Array of 24 integers |
| Infra-red Base | Array of 24 integers |
| Infra-red Base Down | Array of 8 integers |
| Bumper | Array of 2 integers indicating the column and the row the bump happened |

Table 7.5: The data types generated by the different kinds of sensors of the robot MiReLa.

### 7.3.3 The BumperMove Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Turn towards bump | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| `Bumper` →7.3.2 SensorServer<br>`Transition Group type`→7.3.1 Navigator | | `Navigate`→ 7.3.1 Navigator |

Table 7.6: Skill Table for the BumperMove agent.

**Description:**

When the robot is stationary, doing a *Transition Group* of type WNTG, the BumperMove agent makes it able to turn towards where it receives a bump. This imitates the behaviour of human beings when touched, for example on a shoulder. We turn to look towards where we felt the touch. The BumperMove agent only makes the robot behave this way when it is doing a *Transition Group* of type WNTG. As explained in section 7.2.2, the $\alpha_m$ sent to the motors has a value of zero in every *Transition* which form a *Transition Group* of type WNTG. This ensures the detected bump wasn't due to MiReLa bumping into something.

To add this skill, the BumperMove agent requires the `Bumper` data from the SensorServer agent (see section 7.3.2) and the type of *Transition Group* being done at the moment, which it obtains from the Navigator agent (see section 7.3.1) in the form of the `Transition Group type` data. It also has to sends `Navigate` instructions to the Navigator agent.

When a bump is received, if the robot is doing a *Transition Group* of type WNTG, agent BumperMove instructs the Navigator agent sending a `Navigate` instruction of the form turn_<degree>. For example if it detects a bump on the back panel (see figure 7.1 of MiReLa) it sends the instruction turn_180 to the Navigator agent.

### 7.3.4 The DoorClosedAlert Agent

| Skill | Accepted Instructions |
|---|---|
| Make an alert when a door is closed | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `Door Closed` →7.3.1 Navigator | | `Words to Generate`→ 5.2.5 SoundServer<br>`Sound Tempo`→ 5.2.5 SoundServer<br>`Navigate`→ 7.3.1 Navigator<br>`Movie to Show`→ 7.3.8 MovieServer<br>`Gesture to Make`→ 7.3.7 GestureServer<br>`Sound Expresion Level`→ 5.2.6 AmbientNoise |

Table 7.7: Skill Table for the DoorClosedAlert agent.

#### Description:

The DoorClosedAlert agent generates an alert when a door in the path of the robot is found to be closed. It requests the `Door Closed` data from the Navigator agent (see section 7.3.1), when it is set to true, the DoorClosedAlert agent makes an alert. Table 7.8 shows the different instructions sent to each agent depending on the time elapsed from when the door was found closed. Every eight seconds MiReLa says the phrase open door in *MiReLa Music Language*, shows a sequence of pictures on the screen indicating to a person looking at the screen to open the door, and moves the camera gesturing to look at the screen. If the door remains closed, because nobody opens it for the robot, MiReLa repeats the phrase open door, each time a bit slower and a bit louder, giving the impression that it is getting annoyed with the situation. It gets to turn 180°; as if to try to find some help. Finally it returns to normal volume and speed of sound generation and it repeats the phrase every minute, as if it got bored of trying to get help to open the door.

| Elapsed time after last alarm | SoundServer `Words to Generate` | SoundServer `Sound Tempo` | MovieServer `Movie to Show` | AmbientNoise Sound Expr. Level | Navigator `Navigate` | *GestureServer* `Gesture to Make` |
|---|---|---|---|---|---|---|
| — | Open Door | Normal | Start Door | Normal | — | look at the screen |
| 8 | Open Door | Normal | — | Normal | — | look at the screen |
| 8 | Open Door | Low | — | High | — | look at the screen |
| 8 | Open Door | Low | — | High | turn_180 | look at the screen |
| 8 | Open Door | VeryLow | — | VertHigh | — | look at the screen |
| 60 | Open Door | Normal | — | Normal | — | look at the screen |

Table 7.8: The DoorClosedAlert agent alerts. Instructions sent by the DoorClosedAlert agent to other agents depending on the time elapsed since a door in the path of the robot was detected to be closed.

When the door gets open it sends instructions to the SoundServer agent for `Sound Tempo` to be set to normal again, to the MovieServer agent to stop showing the movie, and to the AmbientNoise agent for `Sound Expresion Level` to be normal again.

### 7.3.5 The PhraseChecker Agent

| **Skill** | **Accepted Instructions** | |
|---|---|---|
| Interpret phrases | — | |
| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
| `Detected Phrase`→5.2.3 WordChecker<br>`Detected Word`→5.2.3 WordChecker<br>`Machine Talking`→5.2.5 SoundServer | | `Words to Generate`→ 5.2.5 SoundServer<br>`Navigate`→ 7.3.1 Navigator<br>`Gesture to Make`→ 7.3.7 GestureServer<br>`Start-Stop Music Following`→ 7.3.6 MusicFollower |

Table 7.9: Skill Table for the PhraseChecker agent.

#### Description:

This agent gets the `Detected Phrase` data from the WordChecker agent, interprets the meaning of the phrase, and interacts with other agents in the society depending on the meaning of the phrase.

When the value of `Detected Word` is the name of the robot, it sends to the SoundServer agent the instruction to generate the music for the word "Yes", and sends to the GestureServer agent the instruction "question" to move the cameras gesturing a question (see section 7.3.7).

When the `Detected Phrase` data contains a phrase with navigation meaning, it sends the correct instructions (of the type `Navigate`) to the Navigator agent to move the robot in the required way.

When `Detected Phrase` contains a phrase meaning to play a song, it sends the requested song name to the SoundServer agent, to play it. Depending on the song name, it may instruct other agents to do some actions. For example, when it detects a phrase meaning "play the melody for a good-bye song", the PhraseChecker agent sends the name of this song to the SoundServer agent and when it detects that the `Machine Talking` data has been set to 1 (indicating the robot is generating sound), it sends instruction "start_bye" to the GestureServer agent. When it detects `Machine Talking` has been set to 0 again, it sends instruction "stop_bye" to the GestureServer agent.

When `Detected Phrase` contains a phrase meaning "follow music", it instructs the MusicFollower agent to start to follow the music.

### 7.3.6 The MusicFollower Agent

| **Skill** | **Accepted Instructions** | |
|---|---|---|
| Make the robot able to dance | `Start-Stop Music Following` | |
| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
| `Detected Note`→5.2.2 NoteChecker<br><br>`Someone Whistling`→5.2.2 NoteChecker | | `Notes to Listen for`→ 5.2.2 NoteChecker<br>`Gesture to Make`→ 7.3.7 GestureServer |

Table 7.10: Skill Table for the MusicFollower agent.

#### Description:

The MusicFollower agent makes the robot follow the music by moving one camera (sending instructions to the SensorServer agent (section 7.3.2)). It makes the robot change its behaviour towards the music it listens and instead of trying to find a meaning in the music, it makes some different movements depending on the notes detected. If the music is played in the key of C Mayor: one movement when the tonic $c$ is listened (send instruction "yes" to the SensorServer agent), and one different movement when the dominant note $g$ is detected (send instruction "no" to the SensorServer agent).

When it receives an instruction of the type `Start-Stop Music Following` indicating "start music following", it instructs the NoteChecker agent to listen for some specific notes. To do this, it locks the instruction type `Notes to Listen for` of the NoteChecker agent in level 4. The most common thing is for this instruction to be locked in level 3 by the WordChecker agent, thus, when the MusicFollower agent locks it in level 4 the NoteChecker agent ignores the instructions sent by the WordChecker agent and the WordChecker agent is not able to detect any MML words in the music until the MusicFollower agent unlocks the instruction. Depending on the notes detected, the MusicFollower agent instructs the GestureServer agent to perform a different gesture with the camera.

When a long silence is detected (no `Detected Note` and `Someone Whistling` false for more than five seconds), or an instruction indicating "stop music following" is received, the MusicFollower agent unlocks the `Notes to Listen for` instruction type in the NoteChecker agent.

The MusicFollower agent makes possible a different kind of interaction, by a person playing music and the robot, were the person can make the camera move in different ways depending on the music. It is a way to associate music to the movement, or movement to the music.

The MusicFollower agent has been developed with the idea that it could also be used to aid

to learn the *MiReLa Music Language*. This is not the actual purpose of the MusicFollower agent, but it is a suggestion for a future development. This would be implemented as a new agent which would associate a camera movement to each of the notes of the *MiReLa Music Language*. Thus a *MiReLa Music Language* phrase would cause a sequence of camera movements. Being able to make the camera move in a certain way by whistling, would mean the robot understood a certain *MiReLa Music Language* phrase.

### 7.3.7 The GestureServer Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Generate gestures with the cameras | `Gesture to Make` |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| | | |

Table 7.11: Skill Table for the GestureServer agent.

**Description:**

This agent adds to the robot the skill of moving one of its cameras to make gestures. The GestureServer agent accesses the pan-and-tilt drive of one camera of the robot MiReLa (see figure 7.2). It moves it in different ways depending on the received instructions (see table 7.12).

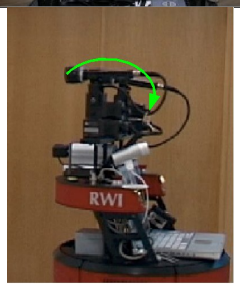
| Instruction | Gesture | |
|---|---|---|
| yes | Move camera up and down once |   |
| no | Move camera left and right once |   |
| question | Direct the camera in the upper left direction, wait 1 second and return to the original position |   |
| look_screen | Turn the camera towards the laptop's screen, wait 1 second and return to the original position |   |
| start_bye | Direct the camera towards the ceiling and start moving left and right. |   |
| stop_bye | Stop moving the camera left and right and return to the original position | |

Table 7.12: Shows the gestures the robot MiReLa does when a specific instruction is received.

### 7.3.8 The MovieServer Agent

| Skill | | Accepted Instructions |
|---|---|---|
| Show Movies on Screen | | `Movie to Show` |
| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
| | | |

Table 7.13: Skill Table for the MovieServer agent.

#### Description:

The MovieServer agent adds to the robot the ability to show videos on a computer screen. In the case of MiReLa on the screen of it's laptop computer, the Guest4 (see figures 7.1 and 7.3). This skill is useful to present a visual form of certain MML phrases. It offers the possibility to define in a text file the frames a movie may be formed from and the frames per second rate. The graphical part was programed by Iñigo Martínez in C++, using the OpenGL libraries. It was then embedded in a common agent structure to give it communication skills.

When an instruction of what `Movie to Show` is received, a window is opened in the left part of the laptop screen with the requested movie.

Only one movie has been used by robot MiReLa so far. This shows a person opening a door. This was formed by some frames consisting of a sequence of real photos taken in different stages of one person opening a door. It was changed for the unique frame shown in figure 7.11, which was considered to be a clearer illustration of the action of opening a door, and thus provide a better visual form of the MML phrase "open door".

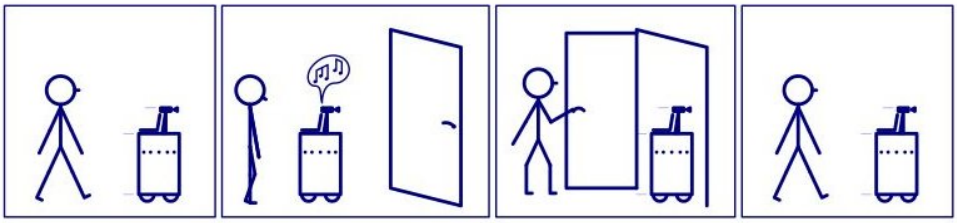


Figure 7.11: OpenDoor movie. Picture designed and produced by Tim Smithers.

When the agent receives the string "`Start Door`", it shows this movie formed by one frame. When it receives "`Stop Door`", it stops showing it.

### 7.3.9 The NavPainter Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Show sensor data on screen | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| `Compass` →7.3.2 SensorServer<br>`Sonar` →7.3.2 SensorServer<br>`Infrared`→7.3.2 SensorServer<br>`Laser` →7.3.2 SensorServer | | |

Table 7.14: Skill Table for the NavPainter agent.

**Description:**

The NavPainter agent is specialised in showing sensor data on a computer screen. As the MovieServer agent this agent runs in the laptop computer of MiReLa. It shows sensor data on the screen of the computer it is running in. It requests the `Compass`, `Laser`, `Sonar`, and `Infrared` data from the SensorServer agent and shows it on the screen. The graphical part of this agent was programed by Iñigo Martínez in C++ using the OpenGL libraries, which was embedded in a common agent structure to give it the required communication skills.

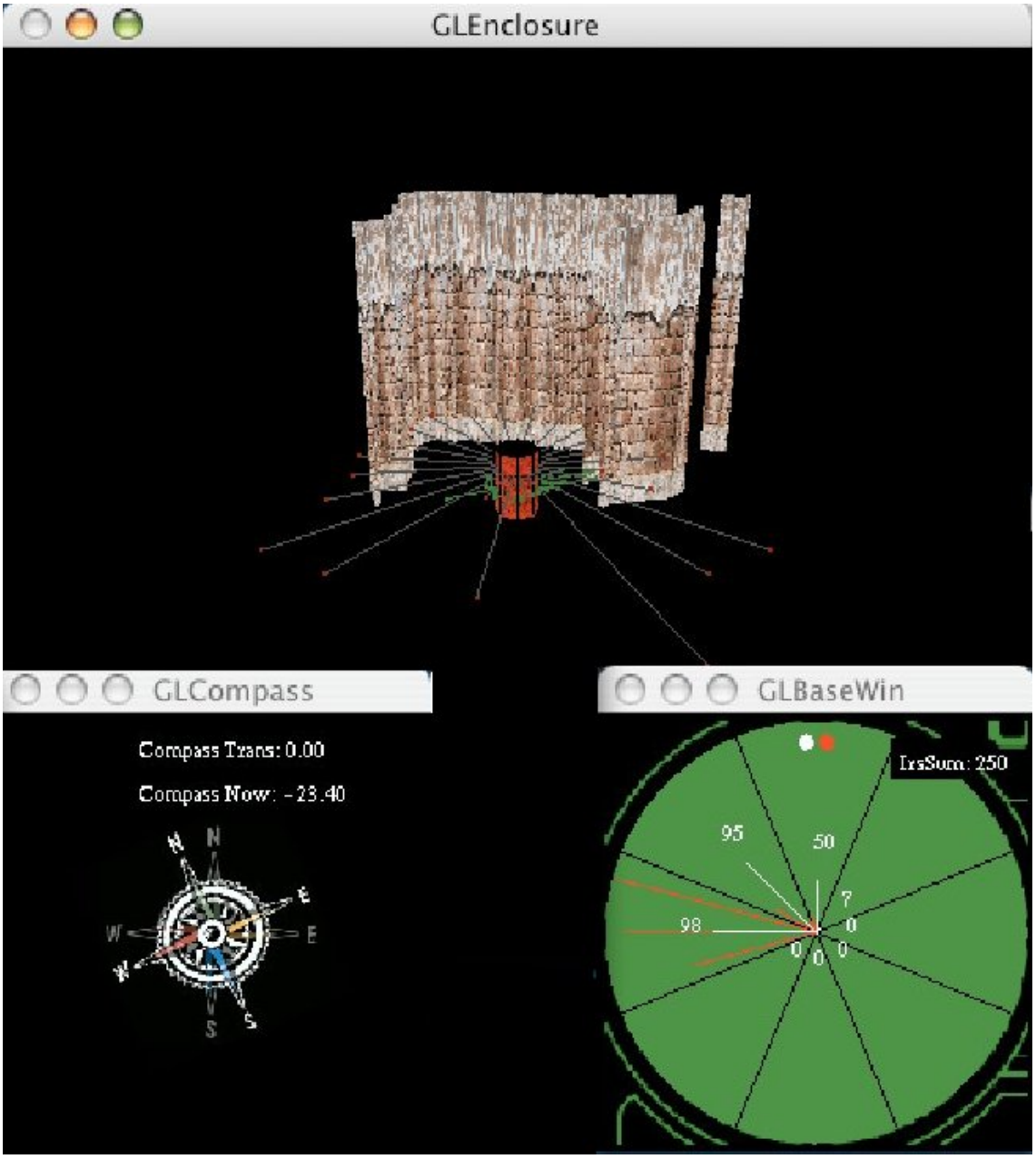



Figure 7.12: The NavPainter agent's screen-shot. On the top part of the figure, titled GLEnclosure, a representation of the `Laser`, `Sonar` and `Infrared` (enclosure) data is shown. The cylinder represents the robot MiReLa and the walls are painted from the `Laser` data values. The `Sonar` data and the `Infrared` data are painted in the form of beams. On the window on the lower left corner titled GLCompass, the `Compass` data is represented and on the window on the lower right corner titled GLBaseWin the `Infrared` data got from the base is represented.

### 7.3.10 The LiftHandler Agent

| Skill | Accepted Instructions |
|---|---|
| Handle the situation at the lifts | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `Transition Group name`→7.3.1 Navigator<br>`Transition name`→7.3.1 Navigator<br>`Machine Talking`→5.2.5 SoundServer<br>`Detected Note`→5.2.2 NoteChecker | | `Navigate`→ 7.3.1 Navigator<br>`Notes to Listen for`→ 5.2.2 NoteChecker<br>`Words to Generate`→ 5.2.5 SoundServer<br>`Sound Expresion Level`→ 5.2.6 AmbientNoise |

Table 7.15: Skill Table for the LiftHandler agent.

#### Description:

The ElevHandler agent makes it possible for the robot to call the lift and instruct it to go to a different floor using DTMF tones (see chapter 6). It gets data from the Navigator agent (section 7.3.1), the SoundServer agent (section 5.2.5), and the NoteChecker agent (section 5.2.2) for it.

As explained in section 7.2, navigation is achieved by making the robot complete a set of *Transition*s, grouped in *Transition Group*s. A special type of *Transition* was designed for entering the lift. It makes the robot to stop ($\alpha_m = 0$ and $\beta_m = 0$) until a Finish Transition instruction is received. A *Transition* of this type is placed where the ElevHandler agent must interact with the lift and the Navigator agent must wait until the interaction is finished.

The ElevHandler agent has access to the `Transition Group name` and `Transition name` data . When it detects that the robot is doing the *Transition Group* to bring it into the lift, it connects to the Navigator, NoteChecker, SoundServer, and AmbientNoise agents, to send them instructions.

When the Navigator agent makes the robot stop waiting for the instruction Finish Transition, the ElevHandler agent, depending on the `Transition Group name` and the `Transition name` data, instructs the AmbientNoise agent to control the `Sound Expresion Level`. Due to the material and the geometry of the building outside the lifts, sound generates big echo and the detection is more reliable if the `Sound Expresion Level` is low. Inside the lift, echo is not that big and the detection is more reliable if the `Sound Expresion Level` is high. It also instructs the SoundServer agent to generate the specific DTMF sound that

requires the situation (call the lift, keep the doors opened, or go to the required floor). It instructs the NoteChecker agent to listen to the specific notes the lift may respond with if it decodes correctly the DTMF sound signal sent. If the expected sound pattern is detected by the ElevHandler agent (checking the `Detected Notes` served by the NoteChecker agent) in response of a DTMF tone, it concludes the communication was successful and sends to the Navigator agent the instruction Finish Transition requesting it to continue with the next *Transition* in the *Transition Group* it is doing.

### 7.3.11 The WebPage Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Control the robot from a Web page | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| | | `Navigate`→ 7.3.1 Navigator |

Table 7.16: Skill Table for the WebPage agent.

**Description:**

The WebPage agent has been designed to control the navigation of the robot from a Web page. It is designed to be used by a person in order for him or her to be able to tele-operate the robot. Different places of the building can be selected to send the robot to. To be at one of these places for the robot is to be doing a certain *Transition Group* of type WNTG. Thus, when one of these places is selected, a `Navigate` instruction is sent to the robot of the form "be doing *Transition Group*".

Images are served from the robot's Web camera to the WebPage agent. This enables the operator to see the places where the robot is navigating through. The operator can also make the robot to stop and do a pan (stop and start rotating on its axis) at any time. This is done by clicking on the button StartPan, at the bottom of the Web page. This is useful for the user to be able to observe more of the environment the robot is navigating through. Clicking on the button StopPan, the robot stops panning and continues doing the current *Transition Group*.

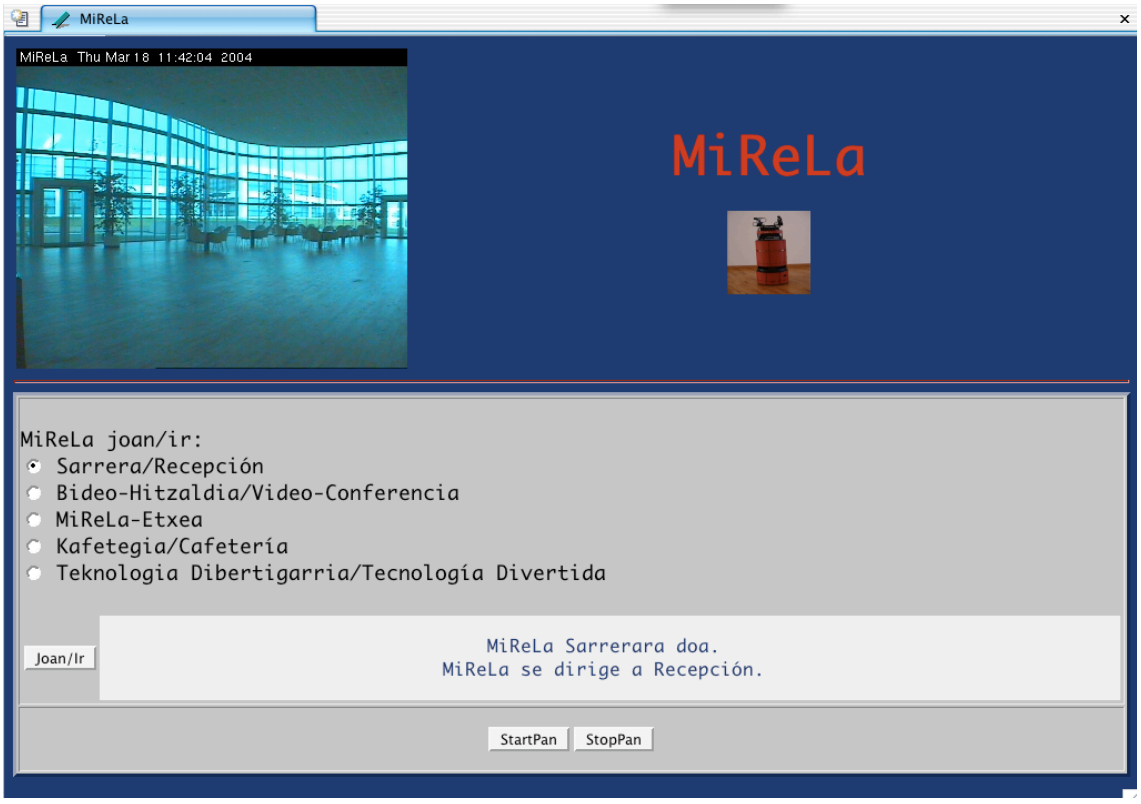



Figure 7.13: Screen-shot of the Web page to operate the robot MiReLa. It offers the possibility to tell the robot to go to five different places in the Main Building of the San Sebastián Technology Park. They are Reception, Video-Conference room, MiReLa's home, Bar, and the Fun Technology Room. The place is selected between the five different options and when the button Joan/Ir (Go) is clicked the petition to navigate to the selected place is sent to the robot MiReLa. The Web page also offers the possibility to send a request to the robot to stop and do a pan motion by clicking on the button StartPan and to stop doing the pan clicking on the button StopPan.

### 7.3.12 The InformationPoint Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Information about the Building | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| `Transition Group name`→7.3.1 Navigator | | `Navigate`→ 7.3.1 Navigator |

Table 7.17: Skill Table for the InformationPoint agent.

**Description:**

The InformationPoint agent offers general information about the San Sebastián Technology Park, and detailed information about the Main Building, where it has been designed to be installed and where the robot MiReLa operates. It was designed and implemented in JAVA by Iñigo Martínez and Jon Azpiazu. It shows a plan of the Main Building of the San Sebastián Technology Park. Clicking on different points of this plan, it offers detailed information about them (Auditorium, Lifts, Hall, etc), which is presented on the left side of the screen (see figure 7.14), and also shows in the plan the route to be taken to go from where in Information Point is located to the selected place and explains the steps for it in part the section behind the plan. It also offers the possibility of video conferencing with the secretary of the Main Building of the San Sebastián Technology Park.

The link of the InformationPoint agent with the robot is that the information point offers the possibility for a person, a visitor to the Main Building of the San Sebastián Technology Park, to be guided by the robot. For this, the InformationPoint agent sends `Navigate` instructions to the Navigator agent. It also requests the `Transition Group name` data served by the Navigator agent, this allows the InformationPoint agent to show the position of the robot in the plan. Images are served from the robot's Web camera to the InformationPoint agent. This enables the user to see the places where the robot is navigating through.

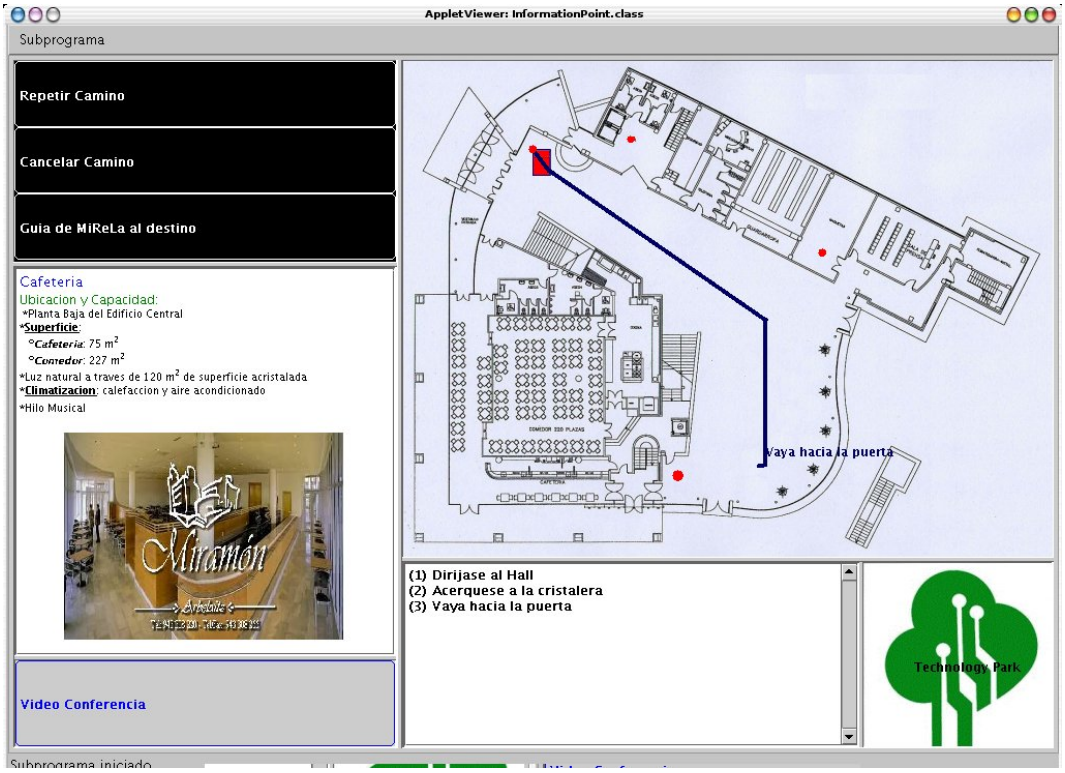



Figure 7.14: Screen-shot of the Information Point. The top right part of the window shows a plan of the Main Building of the San Sebastián Technology Park. Clicking on different points of it, shows information of the selected part of the building. It offers the possibility to be guided to the selected point by the robot MiReLa. The plan also shows where is the robot located at each moment.

### 7.3.13 The Hunger Agent

| Skill | Accepted Instructions |
|---|---|
| Control robot's behaviour depending on the battery voltage | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `Battery Voltage Level`→7.3.1 Navigator | | `Navigate`→ 7.3.1 Navigator<br>`Words to Generate`→ 5.2.5 SoundServer<br>`Sound Tempo`→ 5.2.5 SoundServer<br>`Sound Chord`→ 5.2.5 SoundServer |

Table 7.18: Skill Table for the Hunger agent.

#### Description:

The Hunger agent checks the battery voltage level served by the Navigator agent. As explained in section 7.1 the robot is equipped with four batteries of 12 V. When they all are charged, the total voltage measured is about 50 V. This value drops slowly as the batteries diss-charge. While it is larger than 46 V it is safe for the robot to operate. When it goes below 46 V, it is better to change the batteries, but the robot can continue operating. If the value gets to 43 V or 42 V, the batteries should be changed immediately and the robot should stop moving.

When the `Battery Voltage Level` gets to 46 V, the Hunger agent sends the words "Finish Battery" to the SoundServer agent to generate them in *MiReLa Music Language*, every five minutes (remember MML offers limited structures to create phrases, this is a phrase composed by a verb and an argument).

If the `Battery Voltage Level` is 45 V, it locks the instruction types `Sound Chord` and `Sound Tempo` of the SoundServer agent in level 4, and sends them the instruction of type `Sound Chord` "Minor" and the instruction of type `Sound Tempo` "Slow" to the SoundServer agent. All the sounds are then played in a minor chord and in a slow tempo, giving the impression of tiredness. It sends "Finish Battery" instruction for `Words to Generate` in level 3 every three minutes. It also locks in level 3 the instruction `Navigate` and instructs the Navigator agent to go where the battery charger is located.

If the `Battery Voltage Level` is 44 V, it behaves in the same way as when the `Battery Voltage Level` is 45 V, but it locks the instruction `Navigate` of the Navigator agent in

level 4 instead of in level 3 and sends instruction of type `Sound Tempo` "Very Slow" instead of "Slow" to the SoundServer agent.

If the `Battery Voltage Level` drops to 42 V, the Hunger agent stops all the other agents running in all the computers of MiReLa, and switches the computers off.

### 7.3.14 The LampSwitcher Agent

| **Skill** | **Accepted Instructions** | |
|---|---|---|
| Switch the lamp On/Off by music | — | |
| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
| `Transition Group name`→7.3.1 Navigator | | `Sound Chord`→ 5.2.5 SoundServer<br>`Words to Generate`→ 5.2.5 SoundServer<br>`Sound Expresion Level`→ 5.2.6 AmbientNoise |

Table 7.19: Skill Table for the LampSwitcher agent.

#### Description:

The LampSwitcher agent makes the robot able to switch on or off the lamp when it arrives to the place where it is located.

It checks the `Transition Group name` data served by the Navigator agent and when it detects the robot is doing the *Transition Group* associated with the place the lamp is located in, it instructs the SoundServer agent to generate the name of the lamp (sending a `Words to Generate` type instruction) in a monophonic form (sending instruction "Single" `Sound Chord` type). This is for the lamp to detect its name more reliably. It also instructs the AmbientNoise agent to set the `Sound Expresion Level` to high (sending the instruction "high").

After the lamp is switched on or off, the LampSwitcher agent instructs the SoundServer agent to return `Sound Chord` to the state it was in before (sending the instruction of the type `Sound Chord`: "Last State") and the AmbientNoise agent to return `Sound Expresion Level` also to the state it was before (sending the instruction of the type `Sound Expresion Level`: "Last State").

### 7.3.15 The VisionPeopleDetector Agent

| **Skill** | **Accepted Instructions** |
|---|---|
| Detect and track people faces using artificial vision | — |

| **REQUIRED DATA** | **GENERATED DATA** | **SEND INSTRUCTION** |
|---|---|---|
| | `Face Detected` | |

Table 7.20: Skill Table for the VisionPeopleDetector agent.

#### Description:

The VisionPeopleDetector agent has been designed and developed by Jon Azpiazu in C and C++. It controls a camera (not the one used for gesture making) which moves scanning to find people faces. When it detects a face it tracks it, that's to say, it moves the camera in order to keep the face focused in front of it.

It serves the data the `Face Detected` data, which is a boolean. It is set to true when a face is detected and is set to false when no face is detected.

### 7.3.16 The VisionPeopleTracker Agent

| Skill | Accepted Instructions |
|---|---|
| Aid making a more reliable artificial vision detection | — |

| REQUIRED DATA | GENERATED DATA | SEND INSTRUCTION |
|---|---|---|
| `Transition Group type`→7.3.1 Navigator<br>`Hearing New Phrase`→5.2.3 WordChecker<br>`Face Detected`→7.3.15 VisionPeopleDetector | | `Words to Generate`→ 5.2.5 SoundServer<br>`Navigate`→ 7.3.1 Navigator |

Table 7.21: Skill Table for the VisionPeopleTracker agent.

#### Description:

If the robot is stopped waiting for a navigation instruction, that's to say, doing a WNTG type *Transition Group*, and if the VisionPeopleDetector agent doesn't detect any face, it helps it scan a wider area by making the robot turn on its axis. For this, the VisionPeopleTracker agent checks the `Transition Group type` data served by the Navigator agent and the `Face Detected` data served by the VisionPeopleDetector agent. If the `Transition Group type` data indicates that the robot is doing a WNTG type *Transition Group*, and the `Face Detected` data remains in state "false" for more than five seconds, it instructs the Navigator agent to move on its axis slowly in one direction, that's to say, to make a pan movement. This is done sending to the Navigator agent the `Navigate` type instruction "Pan_+" or "Pan_-". If no face is detected in two seconds doing the pan movement it sends the instruction "Pan_0" to the Navigator agent to stop panning. If other five seconds pass with no detections, it sends a new instruction to the Navigator agent to pan in the direction opposite to which it panned the time before. The VisionPeopleTracker agent keeps sending these instruction (do a pan in one direction, stop doing it, do a pan in the other direction, stop doing it, etc) until the `Face Detected` data indicates a face has been detected.

When the `Face Detected` data indicates a face has been detected, and if the robot was panning, the VisionPeopleTracker agent sends the instruction of type `Navigate` "Pan_0" to the Navigator agent to stop the movement. It also sends the instruction of type `Words to Generate` "yes" to the SoundServer agent. But it only does this if the `Hearing New Phrase` data obtained from WordChecker is set to true, that's to say, if the robot is waiting to listen a new MML phrase. This is important, since if a person is saying something to MiReLa in MML, if the robot generates a word before the person finishes the phrase, it does not listen for the rest of the phrase because the `Machine Talking` data of SoundServer is set to true (see section 5.2.3). Thus, VisionPeopleTracker only sends an instruction to say something if the `Hearing New Phrase` data is set to true.

## 7.4 Concluding Comments

The service robot MiReLa has been presented. MiReLa is being developed to guide people through the Main Building of the San Sebastián Technology Park. Its hardware has been described (section 7.1), and also how it navigates and can be asked, at any time, to go to different places (section 7.2). Then the *Society of Agents* implemented for MiReLa has been presented.

It has been explained in chapter 3 that there are no restrictions to communicate between agents, that's to say, that any agent is allowed to get data from any other agent, and to try to instruct any agent at any time. It can be seen in figure 7.10 that the agents in the *Society of Agents* of MiReLa make use of this property, and that each agent interacts with those which can help it to perform its skill. The *Society of Agents* implemented in MiReLa gives to the robot some skills. More skills can be implemented by adding to the society new agents which exploit the agents already in the society. As the *Society of Agents* gets bigger, there are more skills available to help new agents to perform their skills, thus it typically becomes easier to add new skills.

# Chapter 8

# Trials, Results, and Evolution

This chapter presents the experimental trials which were used to develop, test, and evaluate the *Society of Agents* system, and the *MiReLa Music Language* (MML) system presented in this thesis.

Section 8.1 explains the Open Day trials. These trials have been done with the robot MiReLa, where the *Society of Agents* system and the MML were implemented and tested. It shows how the way to build *Societies of Agents* and the MML have evolved based on the evaluation and conclusions taken from the results of the Open Day trials. It also presents the motivation to install the system to control the lifts using DTMF sounds.

Apart from the Open Day events, numerous other MiReLa demonstrations have been used to test aspects of the robot's behaviour, in particular the MML and the lift use. They have been demonstrations done to the people visiting the Main Building of the San Sebastián Technology Park, such as groups of school students, scientists, and business people. The evaluation of the results of all these demonstrations, also contributed to develop the work presented in this thesis.

Section 8.2 presents a trial done at the "Moncloa Palace", the Presidential Palace, at Madrid, with the robot MiReLa and the lamp MiFaRe. For this latest trial, the *Society of Agents* and the *MiReLa Music Language* were in the state explained in this thesis.

Section 8.3 shows the results of a test done to check how far a person whistling MML phrases could be located from MiReLa and MiFaRe for them to still reliably recognise the phrases.

Section 8.4 shows the results of a test done to check how a person new to the *MiReLa Music Language* can manage to whistle some words and make MiReLa or the lamp understand them.

## 8.1 MiReLa at the Open Day Trials

The Main Building of the San Sebastián Technology Park is a public building. It hosts many conferences, workshops, visits, and other events throughout the year. Each year, in November, an Open Day event is held. The aim of the Open Days is to present the work of the San Sebastián Technology Park, and the various research centres, companies, and other organisations situated in the Technology Park, to a general public. An Open Day is set to last one morning, from 10:00 to 14:00. To date, there have been five such Open Days, the first in 2000, and on such occasions about 2,000 people visit the main building, during the morning. MiReLa was presented and demonstrated during all of these five Open Day events.

These Open Days present a very challenging scenario for the robot. The Main Building, specially the Hall is transformed by many stands and presentations from the companies in the Technology Park. It also fills up with people and noise. These Open Day events have therefore been very important occasions for testing the robot MiReLa in some tough but realistic conditions. Many decisions about subsequent work and developments have been strongly influenced by the evaluations and outcomes of these Open Day trials.

For the 2000 Open Day, MiReLa did a demonstration where it moved around the Hall and in the 2001 Open Day, MiReLa was prepared to play a kind of game where it had to try to locate some children wearing green hats in the Hall of the Main Building of the San Sebastián Technology Park. In both these trials people were not allowed to be close to the robot.

The 2002, 2003, and 2004 trials will be explained in detail, as they have been essential in the development of the work presented here. Nothing was controlled in any of these last three Open Day trials, not the environment (sound, furniture) nor the amount of people around the robot.

### 8.1.1 The 2002 Open Day

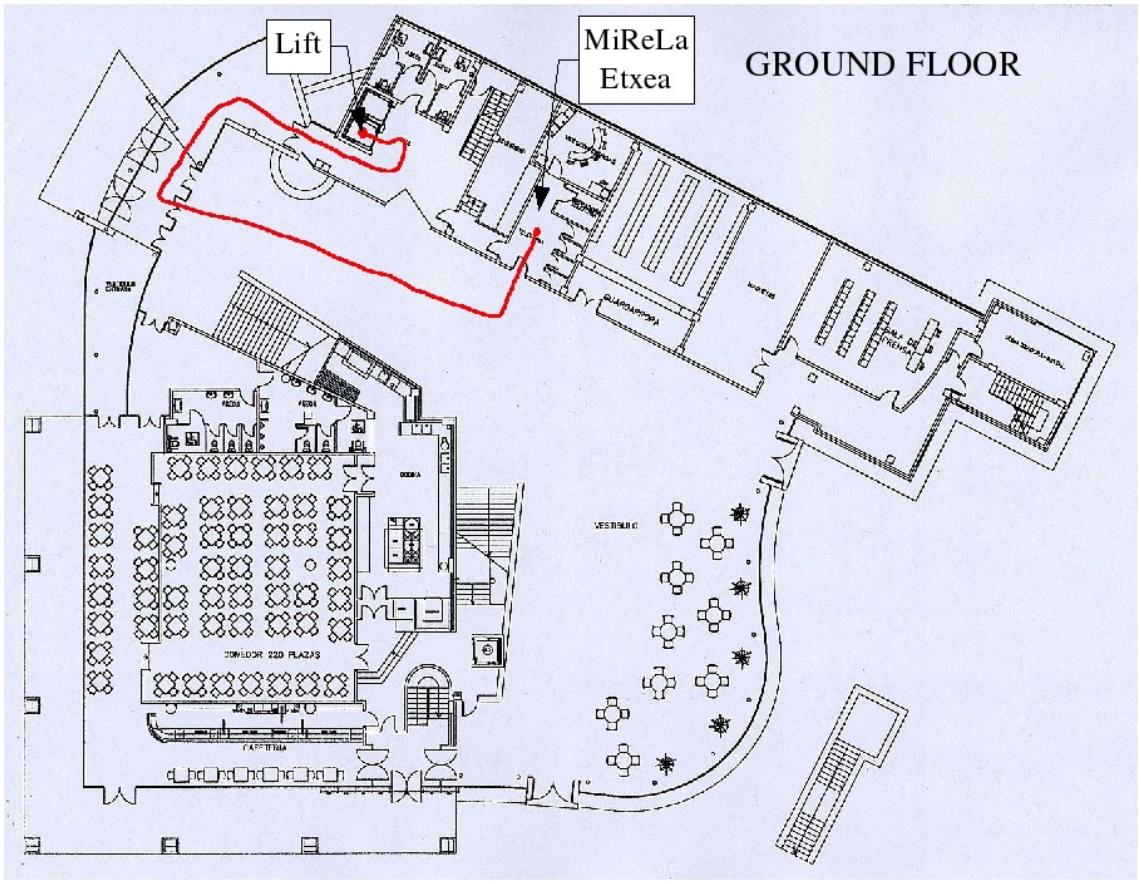



Figure 8.1: Plan of the ground floor of the Main Building of the San Sebastián Technology Park. The trajectory followed by the robot MiReLa in this floor at the 2002 Open Day is marked. MiReLa navigated from "MiReLa Etxea" (MiReLa's home: the room where MiReLa is usually kept) to the lift on the ground floor, on its way to the Auditorium on the first floor, and it navigated from the lift to "MiReLa Etxea" in the way back from the Auditorium.

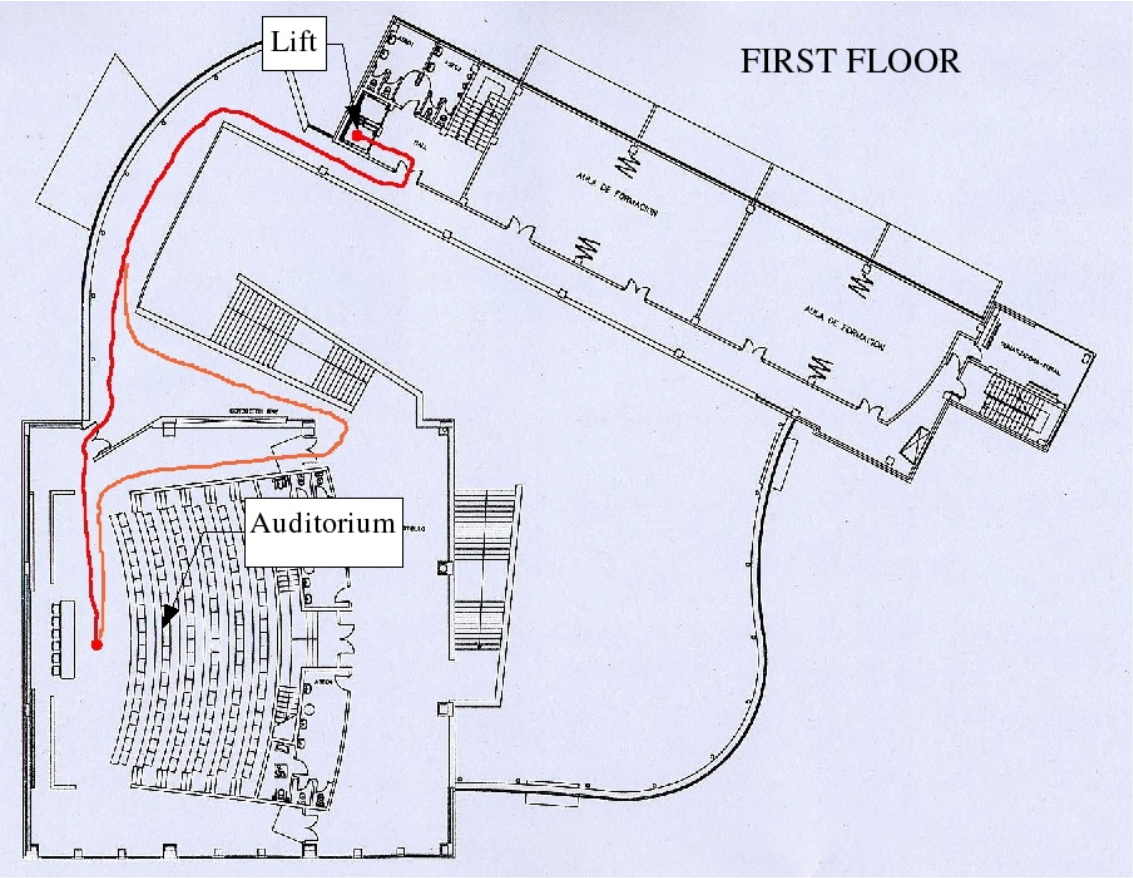



Figure 8.2: Plan of the first floor of the Main Building of the San Sebastián Technology Park. The trajectory followed by the robot MiReLa on this floor at the 2002 Open Day is marked. MiReLa navigated from the lift to the Auditorium and returned back to the lift on this floor.

At the 2002 Open Day, MiReLa guided a person from "MiReLa Etxea" (its home), on the ground floor, to the Auditorium, on the first floor of the building, and then returned home again. Figures 8.1 and 8.2 show the plans of the ground floor and first floor of the main building, and the trajectory followed by MiReLa. The total distance of the whole tour was more than 200 m. To complete it, the robot have to go through 10 doors and it needed to go up and then down one floor in the lift.

A Webcam was installed on the robot to transmit images via wireless network to an audience in the Auditorium. This gave the possibility for people to watch from a Web page, projected in the large screen in the Auditorium, what the robot was seeing.

Two runs of this experiment were presented; one at 11:20 and the other at 13:20. For the first run there were not so many people wandering around the building, for the second run it was much more crowded. On both occasions MiReLa successfully navigated its way from its home, through the Hall, round to the lifts, up the lift, and then into the Auditorium and out of it, and return back to its home. On the first run, MiReLa bumped into a wall, and, on the second run, it bumped an edge of a door, but neither of these events caused

it to get lost, or fail to complete the tour.

In this trial, the complete tour, from home to Auditorium and back, was defined as one *Transition Group* containing the complete sequence of *Transitions*. The possibility to tell the robot to go to different places at any time was not implemented. MiReLa could only do one fixed set of *Transitions*. However, MiReLa's navigation behaviour proved to be robust and reliable in these realistic but difficult conditions, involving lots of people: conditions that MiReLa had never been tested in before the 2002 Open Day event!

The *MiReLa Music Language* was created for this event. Although the robot could generate phrases with name, verb, and arguments, it could only recognise phrases of one word as its name ("MiReLa"), "hello" and "continue". This kind of interaction was used with the robot in the lifts scenario, or in front of a door. Recall that MiReLa doesn't have any kind of devices to help it operate the lift, nor to open and close doors. Thus it needed the help of a person to open and close doors and to operate the lift for it, at this time.

For the doors situation, the robot, could detect when a door was closed and it said "open door" in MML, so that the person following it could open the door.

For the lifts scenario, MiReLa stopped in front of the lifts, and waited to listen the word continue in the *MiReLa Music Language*. The person accompanying the robot called the lift and kept the doors open, then he played the notes for the continue instruction on a harmonica. The robot went into the lift and said which floor number it wanted, again using the *MiReLa Music Language*. It then waited for a continue instruction again. The person operated the lift to bring it to the required floor. When it had arrived, the person kept the doors open again to let the robot exit. Then he played the notes for the word continue and the robot found its way out of the lift.

The *MiReLa Music Language* was presented in the Auditorium to the general public. When MiReLa arrived in the Auditorium, a Xylophone was used to play the words meaning hello, and how are you, to which the robot responded hello, and OK. To finish the interaction the instruction to continue was played and the robot continued executing the sequence of transitions to take it out of the Auditorium and back down to "MiReLa Etxea".

Opening a closed door may be obvious for any person following the robot, if it stops in front of it and starts saying the same thing repeatedly. In fact, people following the robot during the two demonstrations made comments such as "look it is saying that the door is closed" or "it is saying: open the door!". Thus it might be expected a person who met the robot for the first time would understand the needed action (to open the door) from the situation. In the case of the lift, this was not obvious. It was concluded that it would be better for the robot to operate the lift itself. It was also concluded that the interaction between the robot and the lift should be a *MiReLa Music Language* type of interaction to keep to the style or machine-people interaction being investigated. This way, even if people following the robot were not able to understand the meaning of the sounds generated by the robot and the lift, they would understand that a robot-lift communication was taking place between the two machines.

For the 2002 Open Day trial, no society of agents was implemented. Two main processes were run in the two PCs of MiReLa, one for navigation and one for sound recognition

and generation. The laptop was only used as interface to the two PCs of MiReLa. They communicated via UDP sockets [2]. The navigation program was single threaded, where data acquisition from the sensors and processing in the filters was done sequentially in one process. The experience acquired in the preparation of this trial helped to make a clear decision to move from single process, single threaded, applications to distributed, multi-process, multi-threaded applications. One important disadvantage of the single process, single threaded implementation was that if something went wrong in the data acquisition process the whole program had to be halted. It happened with some low periodicity in the tests and demonstrations previous to Open Day of 2002 that the laser range finder had to be restarted as it stopped working. As the robot could only do one fixed sequence of *Transitions* it meant that any failure of this type would require to bring the robot back to the starting point (to "MiReLa Etxea") or change the sequence of *Transitions* to fit to the place the failure happened and rerun the program.

The sound recognition and generation process was formed by two threads, one for doing the recognition from the sound signal to a Fast Fourier Transform Spectrum and then to notes and the other to show the Fast Fourier Transform Spectrum values on the screen.

### 8.1.2 The 2003 Open Day

At the 2003 Open Day, MiReLa was demonstrated as an aid for the reception service of a large building. It offered video-conference capabilities to contact a receptionist and a Web page interface was created to provide a way for this person to control the robot from his or her desk [51].

For this trial two very important goals were achieved towards flexibility and reliability:

1. A *Transition Group* based graph system (see section 7.2.2) was implemented, which gave flexibility to the navigation behaviour making it possible to tell the robot, at any time, to navigate to a different place in the building.
2. An early version of the *Society of Agents*, explained in chapter 3 was implemented in the robot MiReLa to integrate in a convenient and efficient way, the different aspects involved in the operation of the robot, such as navigation, interaction from an early version of the *MiReLa Music Language*, and interaction from a Web page.

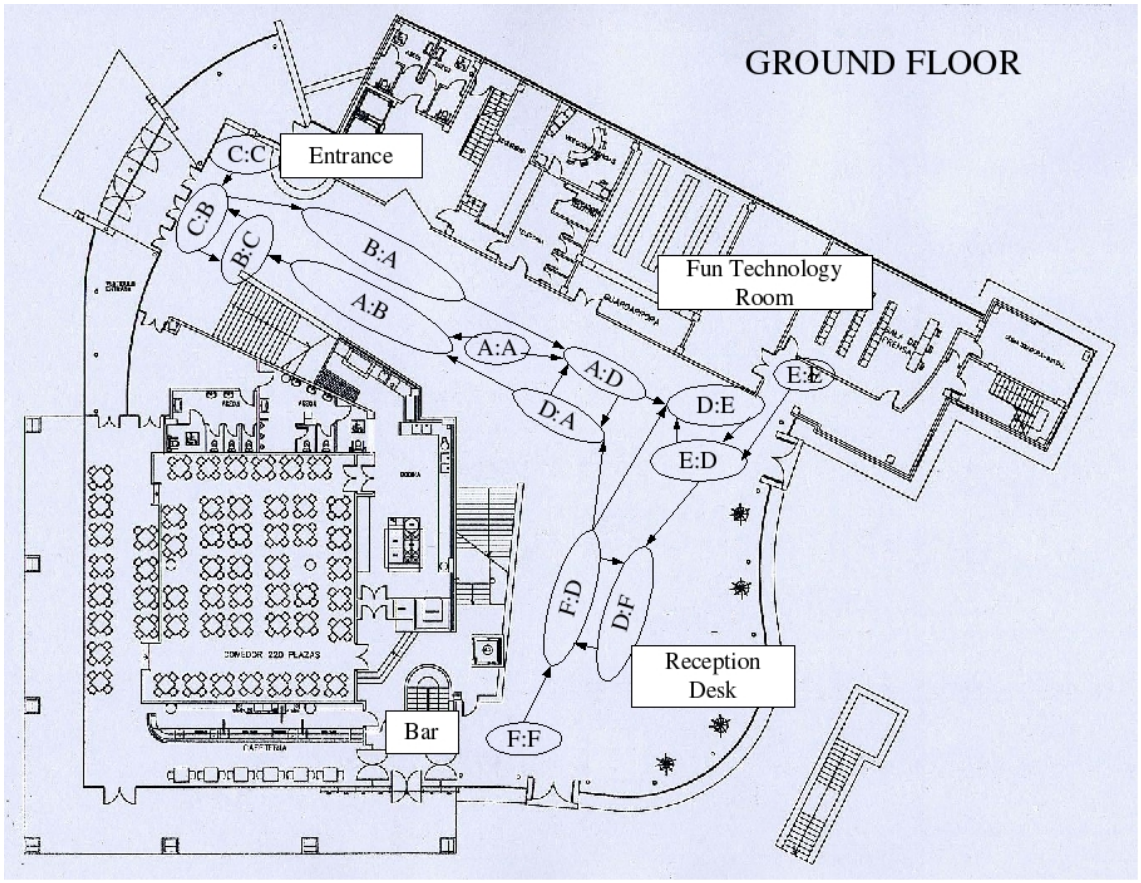



Figure 8.3: Plan of the ground floor of the Main Building of the San Sebastián Technology Park, with a representation of the *Transition Group* graph implemented in MiReLa to navigate in the ground floor at the 2003 Open Day.

At the 2003 Open Day, MiReLa operated in the Hall of the Main Building. Figure 8.3 shows the plan of the ground floor of the main building and the representation of the *Transition Group* graph which made possible for the robot to navigate in it.

The robot was placed near to the entrance of the main building (labelled "Entrance" in figure 8.3), doing *Transition Group* C:C, and it offered the possibility of a video conference connection with another person using a computer placed at the point labelled "Reception Desk" in figure 8.3.



Figure 8.4: Picture showing a person video-conferencing with the person at the "Reception Desk" through MiReLa.

The person at the "Entrance", interested in being guided by the robot, could video-conference with the person at "Reception Desk" and tell him or her which place he or she would like to be guided to by the robot.



Figure 8.5: Picture showing the computer with the Web page control interface at the "Reception Desk".

The person at "Reception Desk" would then use the Web page command interface (see agent WebPage, section 7.3.11), to instruct MiReLa to guide the person to his or her desired place. In other words, MiReLa was instructed to be doing either *Transition Group* A:A, *Transition Group* E:E, or *Transition Group* F:F.

In this experiment, MiReLa operated continuously from 10:00 to 14:00. It always reached the places it was asked to go to. At around 12:00, the Hall became so crowded that it was necessary to slow down the general speed of the *Transitions*, to prevent MiReLa from bumping into people in front of it, because they did not have enough room to make way for it. Figure 8.6 shows two images taken of the occupancy level of the Hall between 12:00 and 13:30.

As said at the beginning of this section, for this trial, two very important goals were achieved towards flexibility and reliability. One was the implementation of the *Transition Group* based graph system, and the other was the implementation of an early version of a *Society of Agents*. The fact that it was possible to tell MiReLa to go to different places, also resulted in the need of a more elaborated version of the *MiReLa Music Language*.

The society of agents, made it possible to split what was done by a single process before into several tasks done by different agents. It gave flexibility and robustness to the software implementation as it could be developed incrementally, simplifying also failure detection. In this early version of the society of agents, there was not a clear distinction made between instruction and request/send data communication and the priority based instruction system was not implemented yet.

Navigation was split into two agents, one for sensor data acquisition (the SensorServer agent), and one for controlling the motors (the Navigator agent).

Thus, the problem explained in the previous section, of having a failure, for example, in laser data acquisition became a less critical issue, as it was solved on two fronts: if this failure happened, the SensorServer agent would stop, but the Navigator agent would continue running, in which case it would notice the lack of laser data information as the connection would be closed. In the case MiReLa would be using the laser data in that moment it would stop, but it would be able to continue when laser restarted. But even having to restart the Navigator agent was not as critical as before thanks to the flexibility given by the *Transition Group* based graph system. It was enough to indicate what *Transition Group* corresponded more or less to the place the failure happened. Anyway, the robot was not required to be brought to the starting point at "MiReLa Etxea".

Sound recognition and generation was also split into several agents: FFTServer (section 5.2.1), SoundServer (section 5.2.5), NoteChecker (section 5.2.2), WordChecker (section 5.2.3), and PhraseChecker (section 7.3.5).

As the *Transition Group* based graph system started to be used, the *MiReLa Music Language* had to handle the possibility to tell the robot at any time to navigate to different places. Phrases of the form `Be doing a` *`Transition Group`* (of type WNTG) or start doing (Be) at a *Transition Group* (of type WNTG) (see sections 7.2 and 7.3.1) were needed to be generated by and recognised using the *MiReLa Music Language*. In other words, phrases composed by a verb and argument(s) were required. A decision about the order of these elements in a phrase was taken. The decision was to place first any arguments a verb may have and finish the phrase with the verb. When a verb was recognised, the phrase was finished, thus the words recognised before were treated as the argument(s) of the verb. If more than one argument was recognised and the verb was a one argument verb, the argument last heard was treated as the valid one for the verb. This was thought to be a good way to handle verbs with variable number of arguments and a clear way to separate phrases.

A phrase was considered to have one of the following forms:

*Arguments Verb*

or

*Adverb or Interjection*

- *Verb:* The action.
- *Adverb:* For example, Yes, no, OK.
- *Interjection:* For example, Hello, Bye.
- *Arguments:* Arguments for the action. Could be Nouns, Adjectives.

Some simple rules were defined for the construction of words:

- *Noun:* Formed by 3 notes.
- *Verb, Adverb, Interjection:* Formed by 2 notes.

Practice showed that, for the tasks the robot is able to do, the most common thing is to use verbs with no arguments or with one argument and in all of the cases are verbs with a fixed number of arguments.

For sound recognition, there was no directed listening done (see section 4.5 for details about directed listening), that's to say, the WordChecker agent didn't instruct the NoteChecker agent about what notes to try to find in the FFT data. The algorithm for note detection was also different from the one explained in the NoteChecker agent (see section 5.2.2): a fixed threshold value was chosen for the amplitudes of the input sound frequencies served by the FFTServer agent. If an amplitude larger than this threshold was detected at some frequencies, the largest among them was chosen. If the same frequency was chosen from the `FFT Spectrum`s received as input, for a fixed amount of time, it was concluded a note of that frequency had been detected.



Figure 8.6: Pictures showing the occupancy level of the Hall of the Main Building of the San Sebastián Technology Park between 12:00 and 13:30 during the 2003 Open Day.

Using a fixed threshold, required a precise control of the input volume selected for the microphone to capture sound. If it was too high, there were many false detections, and if it was too low, no detections occurred. Ambient noise also had a large influence on the performance of the note detection algorithm. If it was high, there were many false detections (notes detected from the ambient noise). The fact that the phrases started with an argument increased the probability of recognising a valid word from the false detections. A verb can have many different arguments, for example, a phrase to tell the robot to navigate to a place would have the following form in this early version of the *MiReLa Music Language*:

| `PlaceName` | `Be_Doing` |
|---|---|
| Argument | Verb |

For each place the robot was able to navigate to, a different noun formed by three notes was created. With the robot able to navigate to many different places and starting the phrases with the arguments, almost any three notes could form an argument. In places with high ambient noise, arguments could be detected from the ambient noise, and sometimes even a whole MML phrase.

The *Society of Agents* of MiReLa supports multimodal interaction; through the *MiReLa Music Language*, Web page, and Information Point. For the 2003 Open Day two forms of interaction with the robot were developed: interaction through a Web page and interaction using the *MiReLa Music Language*. As the wireless network worked properly, the Web page interaction didn't show any problem. But for the *MiReLa Music Language* interaction, between 12:00 and 13:30 it was not possible to make a reliable communication with the robot using the *MiReLa Music Language*, due to the high density of people in the Hall (see figure 8.6) and the high level of ambient noise they generated. Video-conferencing was hard too. (It was also noted that at this time, normal conversation between people was quite hard too, due to the number of people in the Hall.) It happened that the robot sometimes understood a navigation order from this high level of ambient noise, thus it was decided to switch the microphone off. This experience served to enforce the determination to continue investigating the use of the music language approach, and avoid any attempt to use any Human Natural Languages, as it would not be possible to achieve any reliable speech recognition in such conditions. But it was found a clear need to develop a more reliable sound detection system for the next 2004 Open Day.

The experimental results showed the predisposition of people to interact with the robot was different depending on their age. But even for young people, it was necessary for one of the investigators to be near the robot to explain how they could use the video-conference link through the robot.

A better solution was thought to develop an Information Point to offer information about the San Sebastián Technology Park, and include the options of video conferencing with the reception desk, and being guided by the robot to any part of the building.

### 8.1.3 The 2004 Open Day

For the 2004 Open Day a better visitor-robot interface was developed in the for of an Information Point from which it was possible to tell MiReLa to guide a person to any part in the ground floor and first floors of the Main Building of the San Sebastián Technology Park. Also the system to operate the lift using DTMF sounds was installed and tested.

MiReLa operated from 10:00 to 14:00 and it could travel to any place on the ground floor and the first floor of the Main Building of the San Sebastián Technology Park. MiReLa served as an aid for the Information Point presented as an agent of the *Society of Agents* of MiReLa in section 7.3.12. Following the conclusions taken at the 2003 Open Day it was considered a better option not to offer video-conference capability from the robot. Instead this was implemented as part of the Information Point features.

Interaction with MiReLa could be done from the Information Point and via *MiReLa Music Language*. Based on the results of 2003 Open Day, directed listening was developed (see section 4.5) and the sound recognition algorithm explained in section 5.2.2 (the NoteChecker agent) was implemented in the robot. This algorithm removed the dependence on a threshold value for note detection, and so the volume level for sound acquisition, or the ambient noise level didn't influence in the note detection. It also removed the need for defining a fixed amount of time to determine that a valid note had been detected. The syntax of the musical language was also reformed, but it didn't yet have the form of the syntax presented at section 4.

In order to use directed listening, a better option was to change the order of the elements of the phrases. The verb preceded its arguments so that after a verb was detected the listening could be directed only to the notes that form its possible arguments. The decision to start any phrase directed to the robot with its name was also taken for this Open Day event. This way the robot directs its listening only to one note (note $e$ in this case) most of the time. When it detects an $e$, it directs its listening to trying to detect only a $d$, and if it detects a $d$ it directs its listening to try to detect only an $A$. If the sequence of these three notes is detected, it opens its listening for the possible notes that can fit in each position of a verb and its possible arguments, an adverb or an intersection. That's to say, after the name of the robot is detected (the notes $edA$) the robot tries to detect any note that would fit as a first note of a verb, an adverb, or an interjection, defined in the vocabulary. When one is detected, it tries to detect a second note that could fit as the second note of a verb, an adverb or an interjection which starts with this note, and so on. This is explained in section 4.5. It makes more unlikely for the robot to detect a meaningful phrase from the ambient noise. It can happen for the robot to detect the three notes of the name from an ambient music, but, thanks to the new note detection algorithm, not from the ambient noise as used to happen before. Even if it detects these three notes, it is very unlikely for a melody being played in the ambient to have the same sequence of notes to form a MML whole phrase. The probability exists, but it never happened for MiReLa yet; not at the 2004 Open Day neither in any test done after it during the years 2004 and 2005. It has to be noted that the notes of the *MiReLa Music Language* alphabet are formed by high frequencies which are not very common in the music melodies. Starting the phrases with the name of the robot, acts as an effective filter against detecting valid phrases in *MiReLa Music Language* from any melody. It can also be thought that after whistling the name of

the robot, the probability for it to get a valid sequence of notes from an ambient music is bigger. This is true, but when someone is whistling or playing a *MiReLa Music Language* phrase (or any melody for other people to listen) he or she tries to do it in a louder voice than any music being played in the ambient. Thus most probably the robot detects the *MiReLa Music Language* phrase without getting interfered from any music being played in the ambient.

The new syntax of the sentences was as follows:

*NAME Verb Arguments*

or

*NAME Adverb or Interjection*

Where,

- *Noun:* Was formed by three notes, in the range c to A (see table 4.2).
- *Verb:* Was formed by three or more notes. They started and end with the note g which was the Dominant (V th) note in the C mayor scale. Given a key, a person can whistle or play the Dominant note quite easily.
- *Adverb* and *interjection*: Was formed by two notes starting with the note A, which is the Super-dominant (VI th) note in C mayor scale.

The improvement achieved in the sound recognition system was significant. This time the system worked reliably even between 12:00 and 13:30, when the building was very full of people: MiReLa didn't receive any false positive detections.

For the 2004 Open Day, the system to operate the lifts using DTMF sounds was installed for one lift (this is presented in section 6). Since then on, the robot can operate the lift itself. The ElevHandler agent (refer to 7.3.10) makes it possible for MiReLa. As this was done as a prototype project, it was installed in only one of the two lifts of the Main Building of the San Sebastián Technology Park. There is only one button on each floor to call the lift. The control system of the lifts, according to an internal control algorithm, decides which of them responds to the call. This is a handicap for the robot as it is able to enter any of the two lifts. For the moment, and until the system is installed in both of the lifts, or until MiReLa is able to choose the lift to get in, it is necessary to make sure the lift responding to the robot's call is the one where the sound control boards have been installed. (For the Open Day, we disabled the lift which could not be operated from DTMF tones).

The spaces in front of the lift on each floor, due to its geometry and material, make the sound generate large echos. During testing of the robot for the 2004 Open Day, it

was noticed that in this environment, the *MiReLa Music Language* recognition was not as reliable as desired for recognising words containing consecutive repeated notes (for example, *ccd*). From these results a new reform of the music language was done which resulted in the version of the MML explained in chapter 4.

More agents were developed and used for this Open Day event. The priority based instruction system (see section 3.2) was not yet implemented and some problems started to arise as the probability of two agents instructing the same agent at the same time increased. From these result, the need to handle conflicts was identified.

The priority based instruction system was implemented after the 2004 Open Day. One paper was published in [52] explaining this version of the *MiReLa Music Language* used in 2004 Open Day and the *Society of Agents* implemented (with the priority based instruction system already in use) to make the recognition and generation of this version of the *MiReLa Music Language*. Only the WordChecker agent was modified to go from this version of the MML to the version presented in this thesis. If the old version of the WordChecker agent is run, the old version of the MML is used, while if the new version of the WordChecker agent is run the new version of the MML is used. In a future, if there are more than one MML type languages, the way for a machine to be able to recognise them all will be to have more than one WordChecker type agents and choose which of them to be remained active.

## 8.2 MiReLa and MiFaRe at the Moncloa Palace

On the 23th of June 2005, the robot MiReLa and the lamp MiFaRe were transported to the Moncloa Palace, the official residence of the President of Spain, in Madrid. (They were shown together with seventeen other research works from different places in Spain, for the presentation of the project INGENIO 2010 [108].) Three hundred people, politicians, business people, and research directors, were invited to the event. After the presentation of the project INGENIO 2010, the visitors had time to see the different research projects being presented. MiReLa operated for two hours and a half during this public presentation.

For the main presentation, MiReLa was placed at the entrance of the "Consejo de Ministros" building. It had a 2 $m^2$ area to move in. The lamp was placed on one side of the area. The *Transition Group* graph created for MiReLa to navigate in this very small area consisted on four *Transition Group*s. Two *Transition Group*s of type WNTG, one for the robot to stop next to the lamp waiting for a new navigation order, and one for the robot to stop about one and a half metres far from the lamp, waiting for a new navigation order, and two *Transition Groups* of type CTG to connect the two *Transition Group*s of type WNTG (travel one and a half metres in one direction and in the reverse direction).

The robot used a camera installed to try to detect and track people's faces, using a vision system being developed by Jon Azpiazu. It was embedded in the *Society of Agents* by the VisionPeopleDetector (section 7.3.15) and VisionPeopleTracker(section 7.3.16) agents. This agents made it possible for the robot to play the notes for the adverb "yes" when a person's face was detected, and to scan (slowly turning the upper part of the robot's body right and left after some time with no face detection).




Figure 8.7: Picture taken at the entrance of the "Consejo de Ministros" building of the Moncloa Palace, of MiReLa and the three components of the team which travelled for the event INGENIO 2010. The names of the persons from left to right are Urko Esnaola, Jon Azpiazu, and Tim Smithers.

MiReLa could be commanded from the *MiReLa Music Language* to do different things, such as rotate a specified number of degrees, play a song or navigate to the lamp or away from it. The sound detection system presented in this thesis was implemented for this test. When it reached the place next to the lamp, it switched it on or off (depending on the state of the lamp), by playing the notes of the name of the lamp (*efd*,*MiFaRe*), the agent LampSwitcher (section 7.3.14) was created for it.

The "Moncloa Palace" was a completely unknown and unprepared environment for MiReLa, so it constituted a new kind of test for it. At the end of the presentation, MiReLa wandered around one part of the ground floor of the "Consejo de Ministros" building, avoiding obstacles and people around it. It never bumped into anything, or anyone. This was a motion control test of MiReLa in a new and quite different place.

Apart from the ambient noise generated by the three hundred guests, there was music being played in the background. This was not a handicap for MiReLa and the lamp to reliably understand the more than forty commands given to them from different music sources, such as a xylophone, a harmonica, a mobile phone, or whistling. The lamp was switched on and off by MiReLa more than fifteen times and it never failed in any of these attempts.

The priority based instruction system was implemented for this trial which avoided conflicts to happen.

## 8.3 Whistle From Far Test

Tests have been done with the robot MiReLa and the lamp MiFaRe, to see the distance from which they can recognise phrases whistled to them.

For the robot MiReLa a person was positioned on the balcony of the Hall of the Main Building of the San Sebastián Technology Park. Background music was being played as is usual in the building. The robot was first placed in the middle of the Hall, in front of the person. The tester whistled to it in *MiReLa Music Language*, to tell to go to the entrance of the Hall, where the reception desk is located. This is the place where the robot appears in the figure 8.8. The tester then whistled to the robot to go to "MiReLa Etxea", this is reached from the door located just under the person in the figure 8.8. At first the tester missed the pitch, as it is common to happen, more so if there is music being played in the background, but MiReLa detected that someone was whistling and tried to help by playing its own name *edA*, as all the sentences directed to MiReLa start with the name MiReLa (see section 4.3). This was done by the WhistleHelper agent (presented in section 5.2.4). The tester got the pitch and could tell MiReLa to go to "MiReLa Etxea".

The distance between the person and the robot was of fifteen metres, and the interaction succeed even with music being played in the background.

In the case of the lamp MiFaRe, a test was done to whistle from outside of the office where it is located. For this trial, music of the group "Rage against the machine" was put on as background music. There was no problem to make the lamp switch on and off using

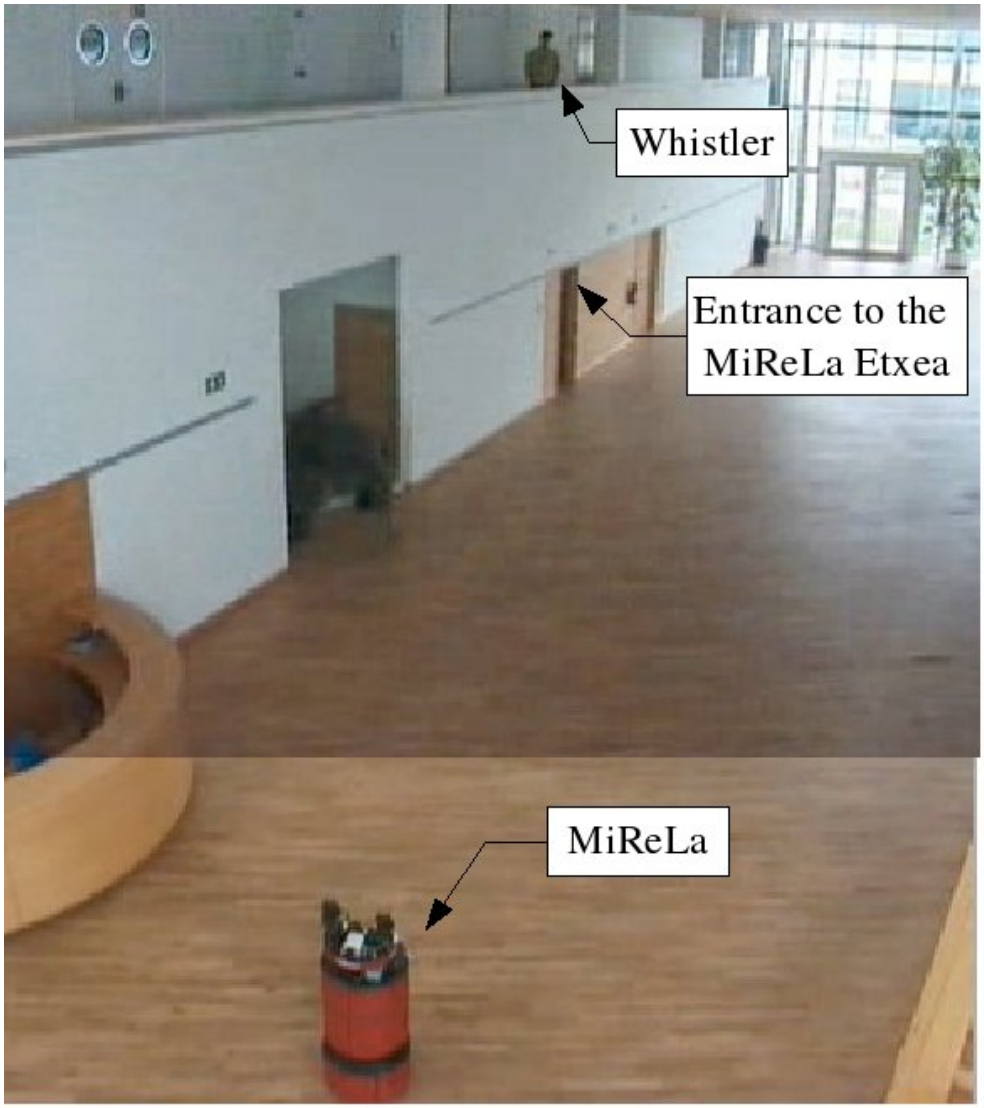



Figure 8.8: Picture of a person whistling to MiReLa from the balcony of the Hall of the Main Building of the San Sebastián Technology Park to tell it to go to "MiReLa Etxea". (Figure built from two shots of a video-clip.)



Figure 8.9: Picture of a person whistling to the lamp MiFaRe from outside of the office. (Figure built from three shots of a video-clip.)

the *MiReLa Music Language*. Figure 8.9 shows a tester whistling from the corridor to the lamp inside the office.

## 8.4 Whistling the Name Test

From the 2004 Open Day a test has been set to check the difficulty for any person to whistle words in the *MiReLa Music Language*. It consists of trying to encourage people, in contact with the robot MiReLa or the lamp MiFaRe for the first time in their life, to whistle the name of the machine. MiReLa plays the notes for the adverb "yes" when it detects its name. The lamp MiFaRe changes its state (switch On/Off) and plays the notes for the adverb "yes" when it detects its name.

From November 2004 to June 2005, twenty-one people have tried whistling to the robot MiReLa or the lamp MiFaRe. The WhistleHelper agent (see section 5.2.4) helped them for it. When it detects someone is trying to whistle something, it helps by playing the name of the machines (the notes *edA* in the case of the robot MiReLa and the notes *efd* in the case of the lamp MiFaRe). Eighteen people succeed in their attempts. Nine of the people who whistled to the machines were children, they all could make the machine detect the name. During the event INGENIO 2010 at "Moncloa Palace", eight people whistled to the lamp MiFaRe or to the robot MiReLa. Six of them succeeded in their attempts.

Some people could whistle the name at the first attempt after listening to the notes given by the WhistleHelper agent. Most of them could do it at the second or the third try. No person tried to whistle the name more than three times. If they couldn't do it by the third time they gave up.

## 8.5 Concluding Comments

This chapter have shown the trials which helped to develop the research work presented in this thesis. The Open Day trials, together with the demonstrations done for people visiting the Main Building of the San Sebastián Technology Park have had an important influence in the evolution of the work (see section 8.1). *Societies of Agents* implemented in the way presented in the chapter 3 and the *MiReLa Music Language* explained in the chapter 4 were developed based on the evaluation of the results of the three Open Day trials presented in this chapter, and (as said in the beginning of this chapter) the smaller demonstrations done during the years 2002, 2003, and 2004. They were also tested in several demonstrations done during the year 2005 and in the trial done in the "Moncloa Palace", explained in section 8.2. In this trials the recognition of the *MiReLa Music Language* demonstrated to be reliable and robust in noisy environments and in the test presented in section 8.3 was demonstrated that *MiReLa Music Language* phrases whistled by a person, can be recognised by a machine from a long distance (about fifteen metres from the emitter to the receptor).

During this year 2005, several people tried to whistle the names of the lamp MiFaRe and the robot MiReLa to check the acceptance the *MiReLa Music Language* would have from the people. The results have been encouraging.

# Part IV

# Evaluation, Conclusions and Future Work

# Chapter 9

# Evaluation

The research presented in this thesis, has been focussed on two main topics. They have been presented in section 1.2. One is the research into a simple and extensible way, to put together all the different aspects involved in the operation of a service robot of Type II. The other is the research into an effective people-machine interaction that uses a music language.

This chapter presents the evaluation results of each of these topics.

## 9.1 A *Society of Agents* for Developing Complex Behaviours in Machines

*Societies of Agents* built in a specific way (see chapter 3), have been investigated to serve to put together all the different aspects involved in the operation of a service robot of Type II. They support the development of complex behaviours by adding simple skills which progressively make the machines able to achieve more sophisticated tasks.

As said in the section 1.2.1, one common practice to achieve a complex goal (such as the resolution of a problem, to make a robot able to perform a task, etc), is to identify different parts, split the goal into simpler subgoals, and try to achieve each of the subgoals. But achieving each of the subgoals doesn't necessarily mean to achieve the original goal. It is necessary to find a way so that each of the achievements complement the others and all together help achieving the original goal. Some tasks have been identified for the robot MiReLa (such as navigate to different places, and interact using an artificial music language), and then the subtasks have been identified which could lead the robot to succeed in performing the defined tasks. Then a way has been investigated to combine the solutions to the subproblems so that they serve not only to achieve the tasks originally defined, but also new ones needed later. It has been done in the form of a *Society of Agents*.

The goal when implementing a *Society of Agents*, has been to create a system formed by small parts, simple, for the probability they fail doing their task, to be low, and

independent, not controlled but influenced from others; and to find a way to connect all these parts so that all together they can make the system able to achieve progressively more complex tasks. This section analyses the *Societies of Agents* implemented for the robot MiReLa (chapter 7) and for the desk-lamp MiFaRe (chapter 5) and evaluates to what extent this goal has been achieved.

Minsky's *The Society Of Mind* has been used as source of inspiration for the development of an agent based control system. Some of the terminology used through this thesis have been taken from Minsky's book, for example the meaning of the term "agent", which has been introduced in section 3.1. This does not necessarily mean that the end result is anything like what Minsky describes. It is necessary to take a careful look at the resulting *Society of Agents* system, and to compare it with what Minsky describes, in order to establish how alike the two are. This section also evaluates if the *Society of Agents* system shares the basic characteristics of the *The Society Of Mind*.

- Subsection 9.1.1 analyses the data exchanged by the agents to evaluate the communication requirements.
- Subsection 9.1.2 compares the basic characteristics of *The Society Of Mind*, presented in section 3.1 with the final implementation of the *Society of Agents* system.
- Subsection 9.1.3 evaluates the aspects which make the robot MiReLa seem to have some kinds of emotions.
- Subsection 9.1.4 presents some properties of the *Society of Agents* system.
- Subsection 9.1.5 presents some conclusions.

### 9.1.1 Agent Data Analysis

This subsection analyses data exchanged by the agents running in MiReLa and the lamp.

- Table 9.1 shows data generated by the agents.
- Table 9.2 shows the instructions admitted by the agents.
- Table 9.3 shows the agents requiring each data type.
- Table 9.4 shows the agents using each instruction.

| | Agent | Data | Bool | Int | Float | String (char array) | Other |
|---|---|---|---|---|---|---|---|
| 1 | Navigator | Transition Group name | | | | x | |
| | | Transition Group type | | | | x | |
| | | Transition name | | | | x | |
| | | Door Closed | x | | | | |
| | | Battery Voltage Level | | | x | | |
| 2 | Sensor-Server | Bumper | | | | | int array 2 |
| | | Sonar | | | | | int array 24 |
| | | Laser | | | | | int array 180 |
| | | Infrared | | | | | int array 56 |
| | | Compass | | | x | | |
| 3 | FFTServer | FFT Spectrum | | | | | int array 1024 |
| | | FFT Spectrum Volume | | | x | | |
| | | FFT Spectrum Processed | | | | | float array 3 |
| 4 | Note-Checker | Detected Note | | | | | struct{ string; float;} |
| | | Someone Whistling | x | | | | |
| 5 | Word-Checker | Hearing New Phrase | x | | | | |
| | | Detected Word | x | | | | |
| | | Detected Phrase | | | | x | |
| 6 | Sound-Server | Machine Talking | x | | | | |
| 7 | Ambient-Noise | Ambient Noise Volume | x | | | | |
| | | Microphone Status | | x | | | |
| | | Speaker Status | | x | | | |
| | | Ambient Noise Volume | | | x | | |
| | | Machine Volume | | | x | | |
| | | PCM Volume | | x | | | |
| 8 | Relay-Controller | Relay Status | x | | | | |
| 9 | Vision-People-Detector | Face Detected | x | | | | |

Table 9.1: Data generated by the Agents.

| | Agent | Data | Bool | Int | Float | String (char array) | Other |
|---|---|---|---|---|---|---|---|
| 1 | Navigator | `Navigate` | | | | | *Instruction Message* |
| 2 | NoteChecker | `Notes to Listen for` | | | | | *Instruction Message* |
| 3 | SoundServer | `Words to Generate` | | | | | *Instruction Message* |
| | | `Sound Tempo` | | | | | *Instruction Message* |
| | | `Sound Chord` | | | | | *Instruction Message* |
| 4 | AmbientNoise | `Sound Expresion Level` | | | | | *Instruction Message* |
| 5 | MusicFollower | `Start-Stop Music Following` | | | | | *Instruction Message* |
| 6 | GestureServer | `Gesture to Make` | | | | | *Instruction Message* |
| 7 | MovieServer | `Movie to Show` | | | | | *Instruction Message* |

Table 9.2: Accepted instructions data.

Twenty-four agents have been created. Robot MiReLa implements a society formed by twenty-three of them (section 7.3) and the lamp MiFaRe implements a society formed by eight of them (section 5.2). Table 9.1 shows that there are nine agents which generate data, they generate twenty-seven different types of data all together. Table 9.2 shows seven agents accept instructions; they all accept only one instruction type except the SoundServer agent which accepts three types of instructions. Table 9.3 shows that from the twenty-seven data types available to the agents:

- Eight of them are not required by any agent, thus nineteen different data are exchanged between different agents.
- Six data types are required by one agent; they are:
  - `Transition name`, `Door Closed`, and `Battery Voltage Level` from the Navigator agent.
  - `FFT Spectrum Processed` from the FFTServer agent.
  - `Detected Phrase` from the PhraseChecker agent.
  - `Face Detected` from the VisionPeopleDetector agent.
- Eight data types are required by two agents; they are:

| | Agent | Data | Requested from |
|---|---|---|---|
| 1 | Navigator | `Transition Group name` | ElevHandler, InformationPoint, LampSwitcher |
| | | `Transition Group type` | BumperMove, VisionPeopleTracker |
| | | `Transition name` | ElevHandler |
| | | `Door Closed` | DoorClosedAlert |
| | | `Battery Voltage Level` | Hunger |
| 2 | SensorServer | `Bumper` | Navigator, BumperMove |
| | | `Sonar` | Navigator, NavPainter |
| | | `Laser` | Navigator, NavPainter |
| | | `Infrared` | Navigator, NavPainter |
| | | `Compass` | Navigator, NavPainter |
| 3 | FFTServer | `FFT Spectrum` | NoteChecker, SoundPainter |
| | | `FFT Spectrum Volume` | — |
| | | `FFT Spectrum Processed` | AmbientNoise |
| 4 | NoteChecker | `Detected Note` | WordChecker, SoundPainter, MusicFollower, ElevHandler |
| | | `Someone Whistling` | WhistleHelper, MusicFollower |
| 5 | WordChecker | `Hearing New Phrase` | WhistleHelper, VisionPeopleTracker |
| | | `Detected Word` | WhistleHelper, RelayController, PhraseChecker |
| | | `Detected Phrase` | PhraseChecker |
| 6 | SoundServer | `Machine Talking` | WordChecker, WhistleHelper, AmbientNoise, PhraseChecker, ElevHandler, |
| 7 | AmbientNoise | `Ambient Noise Volume` | — |
| | | `Microphone Status` | — |
| | | `Speaker Status` | — |
| | | `Ambient Noise Volume` | — |
| | | `Machine Volume` | — |
| | | `PCM Volume` | — |
| 8 | RelayController | `Relay Status` | — |
| 9 | VisionPeopleDetector | `Face Detected` | VisionPeopleTracker |

Table 9.3: Table showing which agents request the data generated by each of the nine agents which generate data.

| | **Agent** | **Data** | **Receive instructions from** |
|---|---|---|---|
| 1 | Navigator | `Navigate` | BumperMove, DoorClosed-Alert, PhraseChecker, ElevHandler, WebPage, InformationPoint, Hunger, VisionPeopleTracker |
| 2 | NoteChecker | `Notes to Listen for` | WordChecker, MusicFollower, ElevHandler, |
| 3 | SoundServer | `Words to Generate` | WhistleHelper, Relay-Controller, DoorClosedAlert, PhraseChecker, ElevHandler, Hunger, LampSwitcher, Vision-PeopleTracker |
| | | `Sound Tempo` | DoorClosedAlert, Hunger |
| | | `Sound Chord` | Hunger, LampSwitcher |
| 4 | AmbientNoise | `Sound Expresion Level` | DoorClosedAlert, ElevHandler, LampSwitcher |
| 5 | MusicFollower | `Start-Stop Music Following` | PhraseChecker |
| 6 | GestureServer | `Gesture to Make` | DoorClosedAlert, Phrase-Checker |
| 7 | MovieServer | `Movie to Show` | DoorClosedAlert |

Table 9.4: Instructions received by the agents.

  - `Transition Group type` from the Navigator agent.
  - `Bumper`, `Sonar`, `Laser`, `Infrared`, and `Compass` from the SensorServer agent.
  - `FFT Spectrum` from the FFTServer agent.
  - `Someone Whistling` from the NoteChecker agent.
  - `Hearing New Phrase` from the WordChecker agent.

- Two data types are required by three agents; they are:
  - `Transition Group name` from the Navigator agent.
  - `Detected Word` from the WordChecker agent.

- One data type is required by four agents; it is:
  - `Detected Note` from the NoteChecker agent.

- One data type is required by five agents; it is;
  - `Machine Talking` from the SoundServer agent.

Thus the most common thing is for one data type to be required by one or two agents. Table 9.4 shows from which agents each agent receives instructions. It can be seen that instructions `Navigate` (from the Navigator agent) and `Words to Generate` (from the SoundServer agent) are used by eight different agents. The priority based instruction system (see section 3.2) has proved to be enough to avoid any conflict resulting from two or more agents sending different instructions to the same agent at the same time. This issue is discussed in the next section (9.1.2).

The SensorServer agent serves `Sonar`, `Infrared`, `Laser`, and `Compass` data with a frequency of 10 Hz maximum (see section 7.3.2) and the FFTServer agent serves the `FFT Spectrum`, `FFT Spectrum Volume` and `FFT Spectrum Processed` data with a frequency 21.5 Hz (see section 5.2.1) if the connection allows, to any agent requesting them (as it is explained in sections 7.3.2 and 5.2.1 respectively). They are the agents which transmit the most data, in the biggest frequency, through their connection. If an agent requests all the `Sonar`, `Infrared`, `Laser`, and `Compass` data, 260 integers and 1 float will be transmitted by the connection (see table 9.1) 10 times per second. An integer and a float need 4 bytes, a byte is formed by 8 bits, thus it requires to transmit 8352 bits (261 $int*4\ bytes/int*8\ bits/byte$) each time; 83.5 Kbits/s (8352 $bits * 10Hz * \frac{1}{1000} Kbit/bit$). In the case of the FFTServer agent, if the `FFT Spectrum` data are requested in one connection, 1024 integer numbers are sent 21.5 times per second. This is 704.5 Kbits/s (1024 $int*4\ bytes/int*8\ bits/byte*$ $21.5\ Hz*\frac{1}{1000} Kbit/bit$). The maximum data transmission rate of the Ethernet connection between the computers in the network of the robot MiReLa is 100 Mbits/s, thus data transmission speed is not a problem for the agents of the society.

There are two agents which require the `FFT Spectrum` data from the FFTServer agent, they are the NoteChecker (section 5.2.2) and the SoundPainter (section 5.2.7) agents. There are other two agents which request the `Sonar`, `Infrared`, `Laser`, and `Compass` data from the SensorServer agent. They are the Navigator (section 7.3.1) and the NavPainter (section 7.3.9) agents. It is important for the NoteChecker agent to get all the data served by the FFTServer agent in time and the same is true for the Navigator agent about the data served

by the SensorServer agent. For the SoundPainter and the NavPainter agents, as their skill is to paint the data received, it is not critical to receive the data in the frequency they are generated in the server agents. For this reason, to avoid any problem related to the network connection (such as video images being transmitted through it), it is recommended to place the NoteChecker agent in the same computer than the FFTServer agent, and the Navigator agent in the same computer as the SensorServer agent. In case the network becomes saturated, an agent in a different computer from the SensorServer or FFTServer agents, would receive data in a lower frequency than the frequency they generate them, but it would receive them actualised. Section 3.3.1 explains how data are transmitted.

Apart from the SensorServer and FFTServer agents, the rest of the agents generate data at a very low frequency and, as is shown in table 9.1, in most cases, they are data formed by a single boolean, a single integer number, a single float number, or a short string of characters. The same is true of the instruction data received by each agent. They are of type *Instruction Message*. A *Instruction Message* has been defined in section 3.3.1. It is a data structure composed by one string of characters and three integer numbers. Agents don't send instructions very often, only in punctual occasions.

### 9.1.2 Comparison with The Society of Mind

This subsection compares the *Society of Agents* presented here, with the basic characteristics of Minsky's *The Society Of Mind* explained in section 3.1.

#### Mind Separated

In his book, Minsky suggests the mind is formed by many smaller processes, called agents, doing very simple things by themselves and joined in societies. This is presented in the section 3.1 of this thesis. Minsky uses the terms "agent" and "society" in a particular sense, which is explained in this section 3.1. From these definitions a society of agents can be understood as a collection of processes doing simple things that organised in specific ways form the human mind.

This thesis, suggests one way to implement agents and defines some simple and uniform interaction rules to allow them to be joined together to form a society (section 3.2). Thus it fits in the definition of society of agents given in *The Society Of Mind*. But, it is important to make clear that it does not suggest a way to explain how human mind works! The goal when designing this particular way to build societies of agents has been, as explained in the introduction, to find an approach to put together all the different aspects involved in the operation of a service robot of type II which allows to make the robot progressively more capable for doing different tasks. A *Society of Agents* has been implemented in the robot MiReLa to make the robot able to do many different task (including interaction using the *MiReLa Music Language*), as it is explained in section 7.3. A *Society of Agents* has also been implemented in the lamp MiFaRe to make it possible to operate it using the *MiReLa Music Language*. This is presented in section 5.2. The lamp MiFaRe is not a service robot, but the *Society of Agents* approach showed to be valid also for these kinds of applications.

The concept of the mind separated is presented in section 3.1: There is not one centralised place or entity which controls the whole system, instead each part controls itself locally, typically with little knowledge about others. Modifications, adding, or removing any of these parts influences the behaviour of the entire system. A *Society of Agents* is a distributed system. In the implementation done for the robot MiReLa and the lamp MiFaRe, there was nothing implemented to control all the agents. Each agent has its own control and it can influence others.

It is very important for Minsky that the agents which form a society achieve their tasks by exploiting other agents. The agents presented in this thesis achieve their goal by exploiting the abilities of other agents. This, in most of the cases, is to get data generated by others and in some cases includes instructions to other agents about how to perform their skill.

### *A-brain* and *B-brain*

The terms *A-brain* and *B-brain*, introduced in subsection 3.1.2, can be used to classify the agents. The *A-brain* consists of those agents which have direct relation with the outside world. They get information from the environment using sensors such as ultra-sound, infra-red, laser range finder, compass, or microphone and they act on it, making the robot MiReLa to move, switching light on and off (in the case of the lamp MiFaRe), generating sound, or painting into a computer screen. The *B-brain* consists of those agents which only interact with other agents. Table 9.5 shows a classification of all the agents presented in the part III indicating if they form an *A-brain* or a *B-brain*.

| | **Agent** | ***A-brain*** | | ***B-brain*** |
|---|---|---|---|---|
| | | Sensor | Actuator | |
| 1 | Navigator | Odometry | Robot Movement | |
| 2 | SensorServer | Sonar, Laser, Infrared, Bumper, Compass | | |
| 3 | FFTServer | Sound Input | | |
| 4 | SoundServer | | Sound Output | |
| 5 | AmbientNoise | | Sound Expression Level | |
| 6 | GestureServer | | Gesture camera movement | |
| 7 | MovieServer | | Paint on a computer screen | |
| 8 | NavPainter | | Paint on a computer screen | |
| 9 | SoundPainter | | Paint on a computer screen | |
| 10 | InformationPoint | | Paint on a computer screen | |
| 11 | WebPage | | Paint on a computer screen | |
| 12 | RelayController | | Activate/Deactivate relay | |
| 13 | VisionPeopleDetector | Video Image Input | Vision camera movement | |
| 14 | BumperMove | | | x |
| 15 | DoorClosedAlert | | | x |
| 16 | NoteChecker | | | x |
| 17 | WordChecker | | | x |
| 18 | PhraseChecker | | | x |
| 19 | WhistleHelper | | | x |
| 20 | MusicFollower | | | x |
| 21 | ElevHandler | | | x |
| 22 | Hunger | | | x |
| 23 | LampSwitcher | | | x |
| 24 | VisionPeopleTracker | | | x |

Table 9.5: Classification of all the agents in *A-brain* and *B-brain* terms.

There are thirteen *A-brain* type agents and eleven *B-brain* type agents. The *A-brain* type agents *act* into the outside world in several ways: five of them paint on a computer screen (showing movies, data obtained from sensors, sound data, information about the San Sebastián Technology Park, and information about the robot MiReLa); two of them control the movement of two cameras (one each), one controls a relay board, one generates sound, one controls the sound expression level, and one controls the movement of the robot.

They *sense* from the outside world in different ways: one gets the odometry values given by the motors of MiReLa, one gets data from the sonar, laser, infra-red, bumper, and compass sensors of the robot, one gets sound input data from a microphone, and one captures video images. The rest of the agents interact exclusively with other agents. Sections 5.2 and 7.3 show which agent does each agent interact with.

The difference between the agents of the *A-brain* and the *B-brain* is that *A-brain* agents are the agents in direct contact with the hardware, while *B-brain* agents are in contact only with other agents. But there is no explicit distinction done to identify each agent to be part of the *A-brain* or the *B-brain*. In other words, when an agent connects to another agent to get data, or send an instruction, the procedure is the same if they are from the *A-brain*, from the *B-brain* or one from the *A-brain*, and the other from the *B-brain*.

### Conflict Handling

A conflict arises when two or more agents need to exploit the same ability of the same agent at the same time. In other words, when two or more agents need to instruct the same agent at the same time, a conflict arises. A conflict may also arise when an agent is doing a task in response to an instruction from an agent and receives a new instruction from another agent before it finishes the task it is doing.

To resolve these situations, the priority based message scheme explained in section 3.2 has been developed. Five priority levels have been defined. The message with highest priority is accepted. It offers the possibility for an agent to lock in a specific priority level, a specific instruction type accepted by another agent. This makes the other agent unable to accept instructions of this type in the specified priority level or lower priority levels, from other agents. The agents are notified if their instruction message has been accepted or not so they can decide what action to take.

This conflict resolution mechanism was developed at quite a late stage in the development of the *Society of Agents* (see section 8). In the early stages of development of the society, no control was implemented over the exploitation of the agents. In other words, in the beginning when an agent received an instruction it processed it. This started to show some problems. For example, it sometimes happened that an agent sent an instruction of the form `Words to Generate` to the the SoundServer agent containing one word of a phrase and before it could send the next word in the phrase, the SoundServer agent received another word from a different agent. Remember that as table 9.4 shows in the previous section, the SoundServer agent receives instructions from eight agents, thus the probability exists for this situation to happen. The result was a *MiReLa Music Language* phrase with no sense. Also, as explained in section 3.2.4, some instructions need a temporal effect, that's to say, an agent needs the effect of the instruction to be maintained for a certain amount of time. As directed listening was used (see section 4), the fact that the NoteChecker agent could be instructed by the WordChecker, MusicFollower, and ElevHandler agents started to be problematic, as their instruction needed a temporal effect. The same happened with the Navigator agent, which like the SoundServer agent, receives instructions from eight different agents. It sometimes happened that the Navigator agent received an instruction to navigate to a place and before reaching the place it received

another instruction, from another agent to navigate to a different place.

The use of the priority scheme solved these problems. Some care must be taken when choosing the priority level in which to send an instruction. The common rule is to send instructions in levels two or three. Priority level five is used for critical instructions. In the *Society of Agents* implemented in robot MiReLa only the Hunger agent uses priority level 5 to send instructions. To avoid the instructions of an agent to be blocked, a good practise can be in some cases to make the agent which receives the instructions to unlock any locked instruction type if a specified amount of time passes without receiving any instruction.

The policy in the research done for this thesis, has been to try to use the simplest solution for concrete problems and not make it more complex until it clearly needs to be. The simplest solution was not to use any control for exploiting other agents. Only when not using any conflict control started to make more complicated to keep the system working reliably was it decided to implement some kind of control. The use of the priority system reliably solved all the conflicts detected in the *Society of Agents* of MiReLa, and in the *Society of Agents* of MiFaRe. It is expected that it will allow to keep conflicts controlled also for bigger agent societies. The aim is to continue with it until the need for a more sophisticated solution becomes clear.

None of the techniques proposed by Minsky (they are presented in section 3.1.3) have been used for conflict handling. They can be interesting when implementing a society of competitive agents. Conflict handling using the priority based system proposed in this thesis, can only be effective in societies formed by fair agents. That's to say, societies where the agents don't compete to get their instructions accepted, but they respect the priority level assigned to it. A competitive agent would end up sending all the instructions in priority level five, and would try to lock the level in order for the instruction accepting agent not to accept instructions from any other agents. It is the designers task to decide the priority level an agent sends each instruction in, for the agent to be helpful for the system and not harmful.

#### Ideas for future developments

Only the basic ideas of Minsky presented in *The Society Of Mind* have been used to design and build the societies of agents presented in this thesis. It has not been investigated how to add memory, or how the societies can make to learn new things. It would be interesting if a robot or a machine could learn new capabilities by itself. Minsky proposes some ideas for these and they have been considered interesting to help for future developments. Section 10.2.3 presents these as possible future work.

### 9.1.3 Emotions

Although investigation about how to implement emotions was not the aim of this thesis, the robot MiReLa gives the impression sometimes to have emotions.

When the robot MiReLa is in front of a closed door; the DoorClosedAlert agent helps it to make an effective alert. Section 7.3.4 explains how the DoorClosedAlert agent instructs other agents to make this alert. The robot starts saying in the *MiReLa Music Language* that the door is closed, shows a sequence of pictures on the screen indicating to a person looking at the screen to open the door, and moves the camera gesturing to look at the screen. After repeating the sentence twice, if nobody opens the door it repeats the phrase a bit slower and a bit louder. If the door continues closed, MiReLa keeps on repeating the phrase slower and louder, giving the impression it is getting nervous. It finishes saying the phrase in a loud voice and very slowly and it also turns back, like looking for someone to help it to open the door. It looks like getting angry. If the door continues closed after some time, it returns to the initial volume and sound generation speed giving the impression that it got bored of trying to engage somebody to open the door. These emotions were not implemented in MiReLa, but it acts as if it is getting nervous, angry and then bored, as a person or an animal might act in this situation, thus the person following the robot, may associate these emotional states to the robot.

The Hunger agent, makes the robot seem to be sad when the battery level drops. This is how Damasio explains how humans feel hungry:

> "Several hours after a meal your blood sugar level drops, and neurons in the hypothalamus detect the change; activation of the pertinent innate pattern makes the brain alter the body state so that the probability for correction can be increased; you feel hungry, and initiate actions to end your hunger; you eat, and the ingestion of food brings about a correction in blood sugar, this time an increase, and the appropriate neurons place the body in the state whose experience constitutes the feeling of satiety."
>
> –Damasio, DISPOSITIONS FOR SURVIVAL, *Descartes' Error* pp. 116.

The drop in the sugar level can be compared with the drop of the battery level, then the Hunger agent, alters the state of some agents so that probability for correction can be increased. This is explained in section 7.3.13. If the battery level drops to a certain level, it sends an instruction to the Navigator agent in priority level three to return home where the battery is recharged. It also instructs the SoundServer agent to talk in minor chords. This gives the impression for the robot to be sad. If the robot doesn't go home and the battery level drops more, it increases the priority of the message to send to the Navigator agent to return home. When the battery level reaches a critical level, the Hunger agent makes all the agents to stop and shuts the computers down.

Damasio in his book *Descartes' Error* explains his point of view about how emotions work in humans. This ideas have also been considered to be interesting for a future development of the *Society of Agents*, and they have been presented in section 10.2.3.

### 9.1.4 Some Properties

**Multi-language:** The InformationPoint agent (see section 7.3.12) was built in Java which wasn't a problem to exploit other agents in the society (request data and send instructions)

programmed in C and C++. As long as agents use the same communication protocol, they can be built in any programing language.

**Multi-Platform:** The robot MiReLa is equipped with two PCs running Linux and an Apple Macintosh PowerBook G4 laptop, running Mac OS X —a version of Unix. Using the Internet standard communication protocol TCP/IP made the communication of agents running in any Operating System possible, thus there were no big problems found to communicate between agents running in the Apple Macintosh laptop with the ones running in the PCs. The fact that both Linux and Apple Macintosh OS X are Unix based systems, allowed to develop in C++ code which could be compiled in both the operating systems without modifications required. This makes the code developed for any agent valid to be run in an Apple Macintosh computer with OS X or in a PC with a Linux Operating system.

**Multi-Developer:** The *Society of Agents* is a good way to separate a problem into smaller problems and solve them one by one. Once these smaller problems are identified, they can be developed separately. Different developers can develop them and the *Society of Agents* offers a simple and easy way to integrate all the different developments and make a system (such as a robot) progressively more capable to perform more complex tasks. Concretely the VisionPeopleDetector and InformationPoint agents were entirely programed by other developers. The only thing to agree was what kind of data they needed, what kind of data they were expected to generate, as well as what kinds of instructions they should send or receive. It was also necessary to explain the other developers the simple communication protocol used by the agents.

### 9.1.5 Concluding Comments

The goal presented at the beginning of this section has been achieved in a large extent. Most of the agents do very simple tasks by themselves and the communication is also very simple. The results of the trials presented in chapter 8 have demonstrated that organised in a society they can support a system doing quite complex tasks. This has been demonstrated especially in the robot MiReLa which is able to navigate to any part of the Main Building of the San Sebastián Technology Park, operate the lift, engage in an effective interaction using the *MiReLa Music Language*, helping the people to whistle the language, make gestures with the camera and pan-and-tilt to help in the interaction, allow to be interacted from an Information-Point and from a Web-Page, integrate artificial vision with some navigation or sound features, make an alert when a door is found closed and find a way to "survive" when the voltage level of its battery is getting down.

Even if almost all the agents do very simple tasks, one of them, the Navigator agent, could yet be split in some simpler agents. As explained in section 8.1 the system originally designed to do navigation has been split into two agents, the Navigator and SensorServer agents. Many features have been added to the Navigator agent, as some new filters and termination conditions, or the possibility of telling it to navigate to a different place at any time (this is explained at section 7.2). But they have not been added as new agents,

but as part of the Navigator agent. The main reason to add these new features not as new agents but as part of the Navigator agent, has been that they were done during the early part of the research work. New filters were added in order to make the navigation more robust, and the possibility to be able to tell MiReLa to navigate to a different place at any time was identified as one thing important to have implemented as soon as possible. A way to build societies of agents was being designed and developed in parallel, and the ability of allowing to choose the place to navigate to in real time was essential for the robot in order to grow the *Society of Agents* of MiReLa and test it. Table 9.4 shows in section 9.1.1 that the Navigator agent gets `Navigate` instructions from eight agents in the society, this demonstrates the importance of this ability in terms of growing the *Society of Agents*.

Now that the *Society of Agents* system is well defined, it would be very useful for the Navigator agent to be divided into a number of agents. Section 10.2.2 shows a detailed explanation about how this could be done.

The comparison done in subsection 9.1.2 shows that the basic characteristics of the *The Society Of Mind* are shared by the *Society of Agents* system proposed in this thesis. It also identifies the next steps to be done in a future development. They are to make a robot or a system, able to learn. It would be useful for a service robot to be able to learn new capabilities.

Subsection 9.1.3 identifies some behaviours of the robot MiReLa which may make a person in contact with the robot think it has some emotional characteristics. It showed this helps people to better understand the needs or intentions of the robot, and thus it is worth to further investigate this effect.

## 9.2 A Music Language for Machines

### 9.2.1 Goal

The second goal of this thesis was to investigate a people-machine interaction using an artificial language based upon musical notes. The result has been the *MiReLa Music Language*.

### 9.2.2 The MiReLa Music Language

#### Hypothesis

The investigation started with the hypothesis that musical sounds are a way almost not explored but offering very interesting features for people-machine and machine-machine interaction. Instead of pushing machines into engaging humans in spoken language, music is an invitation to meet in the middle ground as it is suggested in [17].

There isn't any system nowadays which can reliably recognise speech in environments

with some ambient noise (people talking for example). Instead, it was thought that as musical note recognition is simpler than spoken words recognition, there are more chances to do reliable recognition in noisy environments. It is possible to play music using a very wide set of instruments (including electronic devices such as mobile phones or PDAs) and humans are good whistling sequences of musical notes.

Even if nowadays most of the effort in robotics is being directed towards human like robots, it is not clear that the end user really wants this. Many people don't feel comfortable with these robots which move, speak, and look like humans. In the other hand, machines not looking like humans, and not talking like humans give more the impression they are designed to serve us and not to control us, or be like us.

#### Robustness in noisy environments

The results shown in chapter 8 "Trials, Results and Evolution" demonstrate it is possible to reliably recognise the *MiReLa Music Language* in noisy environments. From the end of 2004 on, a reliable *MiReLa Music Language* recognition system has been achieved, the results of the 2004 Open Day and the Moncloa trial demonstrate this.

The tests done to determine the distance from which a person can whistle the *MiReLa Music Language* show the detection is also robust from long distances. The robot MiReLa could recognise phrases to tell it to navigate to different places in the Hall, from a person placed on a balcony. The robot recognised the *MiReLa Music Language* phrases from a distance of fifteen metres from the person whistling them. In the case of the lamp MiFaRe, it could recognise its name from a person whistling from the outside of the office. In both of the cases there was ambient music as well.

These results show the *MiReLa Music Language* recognition is very robust even if it is whistled in noisy environments and from long distances. It is not possible to find a spoken speech recognition system showing any similar robustness.

#### Evaluation of the structure of the musical phrases

Machines using the *MiReLa Music Language* can be easily understood to be using a language, even if people don't understand what they are saying. The common behaviour people showed when the robot talked in the *MiReLa Music Language* was to try to guess what it was saying. This shows that people understands the robot is using a language.

The longest phrases used in the application experiments have been of three words: Name Verb Argument. The whole interaction with the robot MiReLa could be done using phrases of at most three words, the lamp MiFaRe was designed to switch on and off at detection of its name.

The interaction necessary with most of the machines (including robots), doesn't require long phrases. We don't want to interact with them to have long conversations, we interact with them when we want something from them: in the case of the lamp MiFaRe, to switch the lights on and off, and in the case of the robot MiReLa to go to a specific place, do a

specific movement or play a song. Most of this interaction can be handled with very short phrases formed by a name, a verb and an argument, like for example: MiReLa Be_doing One (to tell to MiReLa to be doing the *Transition Group* one which results with the robot moving to a specific place). In the case of the lamp MiFaRe, for switching the lamp on or off it only requires to recognise its name. These kinds of phrases can be formed by short sequences of notes in *MiReLa Music Language*, they are short *MiReLa Music Language* phrases which are quite easy to learn with little practice.

#### Easy to whistle

As whistling is a way humans have to create musical sounds naturally, the *MiReLa Music Language* has been designed to be easy to whistle. Chapter 4 explains the syntax and the grammar of the language and the motivation which leaded to set each of the rules of the syntax. The *MiReLa Music Language* went through a lot of changes since it was started testing to what it is today. They are reported in section 8.1. These changes have been directed to make the music language more reliable for recognition and easier to whistle.

The tests reported in section 8.4, where almost all the people new to the robot or to the lamp could whistle their name by the third try are very encouraging. The attitude from the people towards the *MiReLa Music Language* changes drastically when they see it is very simple, for example to switch the lamp on and off by whistling. Many of them were surprised with themselves as they didn't expect a priory they should be able to make the machine understand their whistling; the WhistleHelper agent helps a lot in this aspect.

### 9.2.3 Evaluation of the DTMF Sound Interaction With the Lift

DTMF [46] sound signals are widely used for signal transmission via the telephone line. The use of DTMF sound signals is extended for domotic applications such as switching the heater on, starting the oven, or the washing machine, pulling up/down the blinds, or connecting/disconnecting the alarm remotely from the telephone. Many of these applications use a set of boards which translate DTMF sound signals into relay activation/deactivation. This relays are connected to the machine to be controlled remotely.

The fact of having a large variety of such boards available in the market and that the communication is sound based, helped in the decision to install a system based on DTMF tones into the lifts. This has been taken as an initial step towards using the *MiReLa Music Language* to operate the lifts.

It has served to learn about the acoustics of the lifts environment and also about how to install a parallel system (sound based) to operate a lift.

#### DTMF boards and a microphone input signal

The biggest uncertainty before installing a DTMF board into the lift, was if it would be possible to find any which could reliably detect DTMF tones from a sound signal obtained

from the ambient, and not from a telephone line. Many products were examined and many vendors were asked about this issue before choosing the NORCOM NC400 board. All the examined products were designed to get the DTMF sound input from a telephone line connection, so that, the DTMF sounds generated pulsing the phone buttons are transmitted to the board from the telephone line, making the board receive a very clean signal with almost no noise. The NC400 board was the board offering best guaranties to detect sound reliably from a microphone input, although it was designed, as the rest of the boards, to get the sound signal from the telephone line.

The experiments done with the NC400 boards in the office before installing them in the lift, showed them to be able to detect signals formed by sequences of three DTMF sounds from a microphone input. That was not the case (as explained in chapter 6) outside the lift. Due to the geometry and material characteristics of the spaces, a large echo is generated and the board couldn't decode reliably signals formed by sequences of three DTMF sounds. Inside the lift cabin the detection worked reliably.

#### Reliability of the MiReLa-Lift interaction

The final decision was to use DTMF sound signals formed by one DTMF tone outside the lift and inside the lift. As the boards can decode sixteen DTMF sounds, there is no problem to find different signals for the different options to operate the lift of the San Sebastián Technology Park. The lift was prepared to be able to operate it from sound in the ground floor and the first floor of the building, but sixteen signals are enough to extend the system to operate the lift to cover the five floors of the building.

The interaction MiReLa-lift is 100% reliable. The lift never showed any false positive detection and the robot MiReLa always gets the lift to understand its orders. As the lift can generate sound in response to a detection, the communication is bidirectional. When the robot MiReLa generates a DTMF signal it waits for the lift's response, if it doesn't detect any, it repeats the signal until it detects a response.

Inside the cabin, the interaction succeeds almost always at the first try, that's to say, MiReLa generates a signal, the lift detects it and MiReLa also detects the response of the lift at the first try. Outside the lift, the interaction doesn't always succeed at the first try. Sometimes the robot has to repeat the signal more than once until the interaction succeeds. Most of the times it is the lift that doesn't detect the signal, but sometimes it is the robot which doesn't detect the response from the lift. But as said before the interaction always succeeds if not at the first try, at the second or the third try.

### 9.2.4 Concluding Comments

The results of the trials presented in chapter 8, show that it is possible for the detection of the *MiReLa Music Language* to be robust and reliable in noisy environments. They also show that whistling the *MiReLa Music Language*, can be detected from long distances and that it is not difficult to learn whistling some words in it.

The use of DTMF tones to communicate have been tested in the lift. They are well defined tones, formed by two frequencies. It is unusual to get a DTMF tone detection from the ambient noise resulting from the background music or people talking. It needs to be deliberately generated. Thus the recognition of DTMF sounds could be done in a way rather simple, robust and effective. Even a musical language could be thought formed by an alphabet of DTMF sounds. But the fact that it is not possible to generate DTMF sounds by most of the musical instruments, neither whistling, makes the idea not to be attractive.

# Chapter 10

# Conclusions and Future Work

## 10.1 Conclusions

Some of the work presented in this thesis, has been reported in [51], [50], [52], and [53]. An extended version of the last paper [53] is being prepared as a chapter in a forthcoming volume of the Springer Lecture Notes in Artificial Intelligence series.

### 10.1.1 A *Society of Agents* for Developing Complex Behaviours in Machines

The relation proposed between the agents is a Server-Client relation. For the *Request/Serve Data* communication type, each agent acts as server to serve the data it generates and as client to get data it needs from other agents (see 3.2.1). For the *Send/Receive Instructions* communication type, to send an instruction of a specific type, an agent connects as a client to the agent which accepts that type of instructions (see 3.2.2). This is the only architecture principle applied to build the societies of agents for the robot MiReLa and for the lamp MiFaRe.

Although no restrictions have been defined about the complexity of the agents, the goal when building societies of agents should be to create agents as simple as possible. Most of the agents developed for the *Societies of Agents* of MiReLa and MiFaRe do fairly simple things by themselves. May be some of them could be split into agents doing simpler things. To identify them and to split them in agents doing simpler things is a task identified for future work. The most complex agent among these implemented is the Navigator agent. Section 10.2.2 presents a suggestion about how to split it in a society formed by simpler agents.

If there is an agent which needs a certain capability which is implemented by another agent, it should not re-implement it, but make use of the already implemented one by the other agent. This way, while the society grows, and there are more capabilities available, it is easier, faster and safer to add new ones. It is safer for an agent to use a capability already implemented by another agent, which has been tested and shows guaranties of

working properly than re-implementing it by itself.

In the way proposed to create *Societies of Agents*, all the agents are of the same level of importance and there isn't any centralised entity to control them. This makes the *Societies of Agents* to be fully extensible, where, adding a new agent does not necessarily imply any changes for the other agents already in the society. It also showed to help integrate easily development done by different people in different programing languages, this has been evaluated in section 9.1.4.

The way to build societies of agents proposed in this thesis shares the basic characteristics of *The Society Of Mind*. Section 9.1.2 evaluates this aspect.

The agents of the *Societies of Agents* proposed in this thesis are similar to the layers proposed by the Subsumption architecture [31] in:

- Both (an agent and a layer) add one new capability to the system.
- They can be added incrementally; adding new capabilities doesn't necessarily require any changes in the system.

As said in section 9.1.5 the *Society of Agents* built and implemented in the robot MiReLa make it able to navigate to any part of the Main Building of the San Sebastián Technology Park, operate the lift, engage in an effective interaction using the *MiReLa Music Language*, helping the people to whistle the language, make gestures with the camera and pan-and-tilt, allow to be interacted from an Information Point and from a Web-Page, integrate artificial vision with some navigation or sound features, make an alert when a door is found closed and find a way to "survive" when the voltage level of its battery is getting down. This demonstrates that building and implementing the *Societies of Agents* in the way it is proposed in this thesis (see chapter 3) is a valid way to put together all the different aspects involved in a service robot of Type II. This was one of the two goals to be achieved in this thesis research (see section 1.2).

### 10.1.2 A Music Language for Machines

The *MiReLa Music Language* (MML) is a proper language. It is simple and this makes the machines where it is implemented in able to understand any construction in MML. The robots using Human Natural Languages, can't have this property. A Human Natural Language is very rich in vocabulary and in constructions. It is possible to form many kinds of phrases with many names, verbs, adverbs, adjectives, and interjections. It is very difficult to make a robot to understand every kind of sentence which can be constructed in a Human Natural Language. What the approaches using Human Natural Language recognition do is to limit the kinds of constructions a machine can recognise (see for example the case of the robot Hermes presented in section 2.2.3). They don't recognise every meaningful phrase of a Human Natural Language but a sub-group of phrases. But in the case of the MML it is not difficult to make a machine understand any sentence possible to build in this language using the words defined in the vocabulary.

The results of the trials presented in chapter 8, demonstrate the detection of the *MiReLa Music Language* can be done reliably in noisy environments. They also show that a machine can detect phrases in MML whistled by a person even if the person is located at a distance considerably far (about fifteen metres) from it.

The test done encouraging people new to the *MiReLa Music Language* to whistle the name of the robot MiReLa or the lamp MiFaRe in the music language, showed that generally people are good at whistling MML words in tune (with the help of the WhistleHelper agent). This suggests they would also be able to whistle full MML phrases with a bit of a practise. The *MiReLa Music Language* has been designed to be easy to whistle.

The communication we require with machines is usually simple. As explained in chapter 5, the *MiReLa Music Language* has been implemented and tested in the lamp MiFaRe to switch it on and off. It showed to be easy and reliable to use. This suggests the *MiReLa Music Language* is also appropriate to be used, (with proper installations done for each case) to open and close doors, operate window blinds, or operate lifts, for example. They are things we do every day, and proper implementations of the *MiReLa Music Language* for them, would offer the possibility to operate them remotely in a reliable way, without the need of any extra devices, only the ability to whistle short MML phrases. This is useful not only for impaired people, but also for any person who, for example, carrying bags or a big box in his or her hands, would need to operate the lift, open doors and switch the lights on or off; or for a person who sitting down in a sofa needs to roll down the window blinds to watch the TV better and close the door of the room to listen the sound better.

It can be concluded that the use of the *MiReLa Music Language* is an effective, efficient and acceptable way for people-machine and machine-machine interaction; which was the second goal to be achieve in this thesis (see section 1.2).

## 10.2 Future Directions

As imagination always goes farther and faster than real life, many ideas for future development have been thought during this research. It is gratifying to see that many of the ideas which once were part of the future work plans of the imagination became to be part of the real life. And, it is interesting to finish the thesis, presenting some ideas for farther development together with suggestions about how to make them to be real, or at least about where to start the investigation.

Some of the ideas for the future work have been mentioned in the previous chapter 9 of Evaluation and in the previous section 10.1 of Conclusions. They are:

1. To split the Navigator agent into a *Society of Agents*. A suggestion about how to make this to be possible is presented in the section 10.2.2. It presents the agents to implement for it, explaining the data each of them should generate and request from others, and the instruction types each should accept and send to others. A graph of this *Society of Agents* is presented as it has been done in section 5.2 for the agents

forming the *Society of Agents* of the lamp MiFaRe, and in section 7.3 for the agents forming the *Society of Agents* of the robot MiReLa. A brief explanation is given of the role each of the agents involved in the *Society of Agents* of the Navigator agent should have.

2. Make a machine able to memorise and learn using a *Society of Agents*. For a service robot of Type II, like MiReLa, it is interesting to have the ability to learn new capabilities. It would be interesting for it to be able to learn how to navigate from one place to another, or to learn the meaning of new *MiReLa Music Language* words and phrases, by itself, or being taught for somebody or some other machine. The capabilities, are now added to it in the form of new agents, that's to say, a new agent adds to the robot a capability or a skill. A goal for a future work should be the robot to have some agents to help it learn new things, or even to make the robot able to create new agents itself. The first step in this process is to be able to have some kind of memory of the states (generated data) of some agents of its *Society of Agents*. Section 10.2.3 is a collection of some suggestions from Minsky, in *The Society Of Mind*, about how to do it.

3. Investigate more about emotions. It has been reported in many investigation works that giving the impression, from a robot, of having emotions helps the interaction between people and the robot. Some of these reports have been presented in the chapter 2 (robot Sage, and robot Minerva). As explained in section 9.1.3, also the robot MiReLa gives the impression of being translating from one emotional state to another in certain occasions. It is thought to be worth to investigate in more detail what human emotions are, and how useful would be for a robot to have some kinds of emotions. Section 10.2.4 presents some ideas from *Descartes' Error* [44], of Damasio, which explain what he thinks emotions and feelings are, and the role they have in reasoning. This intends to be useful as a starting point for the investigation suggested about emotions.

Apart of the three directions mentioned above, some ideas to extend the work presented here, have been built in the world of the "imagination". Section 10.2.1 summarises them.

### 10.2.1 Future Development Ideas

This subsection summarises some suggestions for future work:

1. Add the possibility to be able to ask to the robot to make a tour. This in *MiReLa Music Language* could be said as "MiReLa Tour <list of places to visit>" where MiReLa is the name of the robot, Tour would be the verb which would be followed by the places the robot would be required to visit in the tour. A new agent (the TourMaker agent, for example) would get the names of the places and send to the Navigator agent instructions of the type `Navigate` to make the robot visit them. This agent would check the `Transition Group name` data generated by the Navigator agent to know which *Transition Group* the robot is doing. It would instruct the Navigator agent to go to the first place on the list, when the `Transition Group`

`name` data would indicate the place was reached, it would send an instruction to navigate to the next place in the list, and would continue this way until all the places in the list would be visited. To add this capability to the robot, would not require any changes for the Navigator agent.

2. One of the conclusions, based on the results of the trials done for the *MiReLa Music Language*, has been that it could be used to switch the lights on and off, operate a lift, open and close doors, and operate window blinds. Thus a next step is to implement and test it in such ambient intelligence environment.

3. It would be interesting to install two, or three, or even four directional microphones in a machine, to detect where the MML notes come from. This would be done creating FFTServer type agents, one for each microphone. A new agent would check the `FFT Spectrum` data served by them and would combine them to calculate an estimation of the sound source direction. This agent would serve the combined `FFT Spectrum` data and also another data indicating the direction from which it calculated an MML sound came from. The NoteChecker agent would get data from this new agent. The WordChecker agent would request the data indicating the sound source direction, so that it could estimate the direction from which the notes served by the NoteChecker agent came from. In the case two consecutive notes came from two different directions, the WordChecker agent would be able to determine they came from two different sources thus would not understand them as forming a MML word. This will become useful as more things use MML to communicate.

4. The *MiReLa Music Language* recognition is based on absolute pitch. In other words, the exact note frequencies of the *MiReLa Music Language* words, must be whistled or played by an instrument, in order for a machine to recognise them. This has been shown to be effective with the help of the WhistleHelper agent. But it would be interesting to investigate how MML words could be recognised based on note intervals rather than in absolute pitch. For example, to recognise the name of the robot MiReLa (the notes $edA$), instead of checking if the frequency of the note $e$ (1,318.52 Hz) is detected, then the frequency of the note $d$ (1,174.64 Hz) is detected, and then the frequency of the note $A$ (1,760.00 Hz) is detected, it would be to check the note intervals. That is to say, to look for a sequence of three notes where the relation between the frequencies of the first and the second note would be:

$$\frac{1,318.52(e)}{1,174.64(d)} = (\sqrt[12]{2})^2 = 1.1225 \tag{10.1}$$

which means the second note is two semitones lower than the first. And the relation of the frequencies of the second and the third note would be:

$$\frac{1,760.00(d)}{1,318.52(A)} = \frac{1}{(\sqrt[12]{2})^7} = \frac{1}{1.4983} = 0.6674 \tag{10.2}$$

which means that the thirds note is seven semitones higher than the second [82].

5. In the introduction chapter has been mentioned that there are a number of whistled languages all around the world. One of them is “el Silbo” [93] which is still used in Gomera, one of the Canary islands (Spain). This language is being taught to

children at school, and it is a particular whistled version of Spanish. An interesting project would be to investigate this language to find if it could be reliably detected by a machine, and if this would be possible, compare it with the *MiReLa Music Language*.

### 10.2.2 The Navigator agent as a *Society of Agents*

Section 7.2 explains how navigation is achieved in MiReLa and section 7.3.1 explains the Navigator agent. The movement control of the robot is translated to the generation of two values $\alpha$ (forward speed) and $\beta$ (rotational speed) to send periodically to the motors. The procedure followed for it was to create structures called *Filter*s which get data from one or more sensors and generate an $\alpha$ and a $\beta$ each. The $\alpha$ values of the *Filter*s are combined in a harmonic mean to get a unique $\alpha_m$. The $\beta$ values of the *Filter*s are combined to get a unique $\beta_m$. The navigation behaviour of the robot depends on the *Filters* chosen to be combined. A *Transition* defines the *Filter*s to be combined and some termination conditions. When one termination condition is satisfied the *Transition* is considered to be finished. A *Transition Group* defines a set of *Transitions*. To do a *Transition Group* is for the robot to complete the sequence of *Transitions* it defines from the first to the last. A *Transition Group* also defines the ANTG (Allowed Next *Transition Groups*), which are the *Transition Groups* the robot can start doing when it completes all the *Transitions* of the *Transition Group*. The effect of doing *Transition Groups*, for the robot MiReLa, is to navigate to different places. This is done by the Navigator agent, but the Navigator agent can be split into different agents arranged in a society.

Figure 10.1 presents a representation of a possible *Society of Agents* in which Navigator could be split in. Each *Filter* is an agent. CompassFilter, SonarFilter, IrFilter, DriveFilter, LaserFilter, DoorFilter, and StopFilter are *Filter* agents. A *Filter* agent gets sensor data, thus gets data from the SensorServer, TransMotor or RotMotor agents, and transforms it into the values $\alpha_f$ and $\beta_f$. Some of them accept instructions, for example, the CompassFilter agent can be instructed with the `Compass Heading` the robot is required to turn in.

The Combinator agent is the agent which combines the outputs ($\alpha_f$ and $\beta_f$) of the *Filter*s. It accepts instructions about which *Filter*'s outputs to combine, it gets the $\alpha_f$ and $\beta_f$ served by these *Filter*s and serves the results ($\alpha_c$ and $\beta_c$) for any agent requiring them. This happens when ever one of the outputs of the filters chosen to be combined is updated, and thus the frequency in which $\alpha_c$ and $\beta_c$ are generated, depends on the frequency the sensor data is served (10 Hz maximum in the case of the SensorServer agent).

The Driver agent gets the $\alpha_c$ and $\beta_c$ served by the Combinator agent and instructs the TransMotor and RotMotor agents with them if no instruction is received ($\alpha_d$ and $\beta_d$) from any other agent. If it receives instructions of the type $\alpha_d$ and $\beta_d$ it sends them to the RotMotor and TransMotor agents instead of the $\alpha_c$ and $\beta_c$ obtained from the Combinator agent. This is useful, for example, to implement an emergency stop for the robot.

The RotMotor agent makes the robot able to rotate. It accepts instructions about rotation speed ($\beta_m$) and serves data obtained from the rotation motor encoder of the motor.

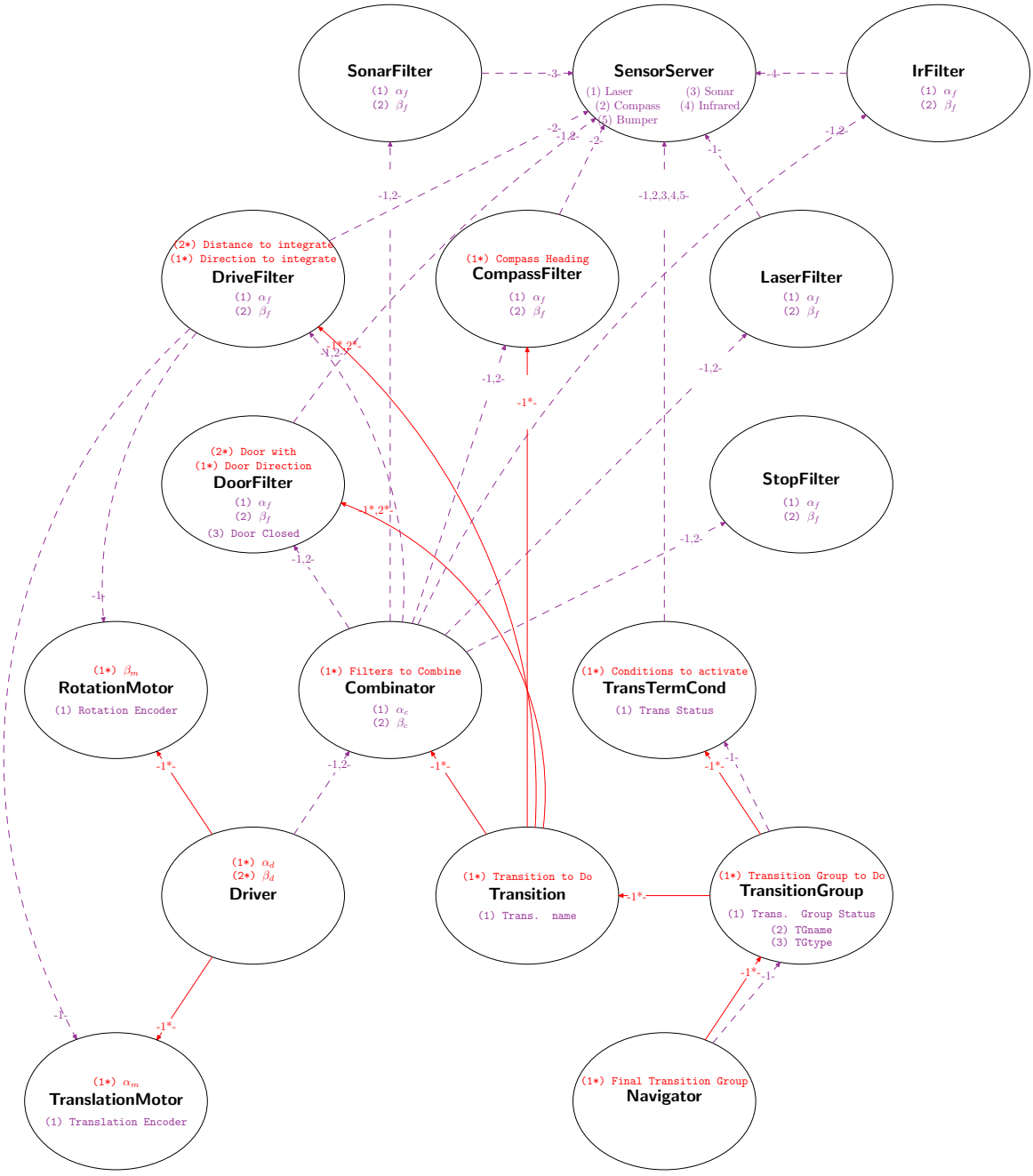



Figure 10.1: The Navigator agent split in many simpler agents arranged in a *Society of Agents*.

The TransMotor agent makes the robot able to translate. It accepts instructions about translation speed ($\alpha_m$) and serves data obtained from the translation motor encoder of the motor.

The Transition agent is the agent which instructs the Combinator agent about which *Filter*'s outputs ($\alpha_f$ and $\beta_f$) to combine in each moment. It also instructs the *Filter*s that accept instructions; for example, if the DoorFilter agent's output is required to be combined it instructs it with the width and the orientation of the door. It accepts instructions about the *Transition* to do in each moment. The Transition agent serves the name of the *Transition* being done.

The TransTermCond agent checks the conditions to determine when a *Transition* may be considered to be finished. It serves the status of the transition (`Transition Status`). It can be instructed with the the conditions to activate. When one of these conditions is completed `Transition Status` indicates the transition is finished. This data may serve for other agents to decide when to start doing a new *Transition*.

The TransitionGroup agent gets the `Transition Status` data served by the TransTermCond agent. When it indicates the actual *Transition* has been finished, it instructs Transition to do the next *Transition* of the *Transition Group* being done. It also instructs the TransTermCond agent about what conditions to activate for each *Transition*. It serves the name and the type of the *Transition Group* being done (`Transition Group name` and `Transition Group type`). It also serves the status of the *Transition Group* being done (`Transition Group Status`). It is not required a different agent to determine when a *Transition Group* has finished as the *Transition Group* finishes when finishes the last *Transition* of the group. (See section 7.2.2).

The Navigator agent can be instructed with the *Transition Group* (of type WNTG) the robot is required to be doing. It decides the *Transition Group*s to do in order to achieve this goal and instructs TransitionGroup to do them.

This arrangement of the Navigator agent as a *Society of Agents*, allows to get many data from each different agent which might be useful for other agents not necessarily designed for navigation. It also offers flexibility to test different navigation approaches. For example, another agent could be created to combine the filter's output in a different way and make driver get the $\alpha$ and $\beta$ from this agent. And it allows to update or repair the agents without stopping the whole system.

### 10.2.3 Remembering and Learning

For the robot MiReLa would be useful to be able to learn new capabilities. It would be useful for it to be able to learn how to navigate from one place to another. In other words, to learn to build the *Transitions* and the *Transition Groups* which would bring it from one place to another. It would also be useful if it could learn the meaning of some *MiReLa Music Language* phrases. The ability from the robot to learn such things would save people from having to program them.

Minsky explains some techniques and kinds of agents which can help in the process of

learning. They could be applied to the *Societies of Agents* presented in this thesis. To be able to learn, the machine needs to be able to remember. To remember, in the case of the agents of the *Societies of Agents* would be to store some specific data generated by themselves or others in concrete moments. Minsky, defines a type of agents that he calls *K-line*s, which may be used by a system to store some of the states of other agents, and *Polyneme*s as a sub-type of *K-line* agents which can be useful to help learning things.

For example, a future starting point could be to make possible to teach MiReLa that the phrase in MML "MiReLa Turn Forty" means we want the robot to turn forty degrees. In other words, make possible for us to teach MiReLa that the first word is its name, the second word is a verb which involves some kind of movement, and the third word is an argument which specifies some quality of the movement.

### *K-line*s

In *B-brain* type agents, a very useful type of agent is a *K-line* (Knowledge-line). Minsky defines it as follows:

> "A *K-line* is a wirelike structure that attaches itself to whichever mental agents are active when you solve a problem or have a good idea.
>
> When you activate that *K-line* later, the agents attached to it are aroused, putting you into a "mental state" much like the one you were in when you solved that problem or got that idea.
>
> ... we "memorize" what we're thinking about by making a list of the agents involved in that activity"
>
> —8.1 K-Lines: A theory of memory pp. 82

A *K-line* agent in the *Societies of Agents* presented in this thesis, would monitor some specific agents, that is to say, it would obtain some specific data from certain agents. This agent would have the ability to send some kind of signal (an instruction) to the agents it monitors, to make them generate the data they generated in a specific moment.

### Remembering and Creating Fantasies

Remembering or generating a fantasy or an illusion, is quite the same and it can be done using *K-line*s. It is to bring certain agents to some specific states. In the case of remembering, it is to bring some agents to a state they already were before. Minsky says that:

> "A fantasy need not reproduce the fine details of an actual scene. It need only reproduce that scene's effect on other agencies."
>
> —16.8 Stimulus vs. simulus pp. 170

and Damasio says that:

> "If you have a dispositional representation for the face of Aunt Maggie, that representations contains not her face as such, but rather the firing patterns which trigger the momentary reconstruction of an approximate representation of Aunt Maggie's face, in early visual cortices."
>
> —Storing images and forming images in recall pp. 102

For example, remembering Aunt Maggie's face, may be done by one or more *K-line* agents which send certain signals to some specific agents in early visual cortices and bring them to a state were they reconstruct an approximate representation of Aunt Maggie's face.

### Learning and Developing

Mind is an always changing entity. A society may grow, may change, and may develop. This must be done without stopping the whole system. In Minsky's words:

> "How could a brain continue functioning while changing and adding new agents and connections? One way would be to keep each old system unchanged while building a new version in the form of a detour around or across it—but not permitting the new version to assume control until we're sure that it can also perform the older system's vital functions. Then we can cut some of the older connections."
>
> —10.9 Learning a hierarchy pp. 107

To solve new problems, to achieve new goals, we have to develop new skills. This may be done in different ways. We can start anew to try to find a solution to the problem, in other words, we can start developing the new skill from scratch. But this is not usually what we do.

> "When attempting a new task, we never like to start anew: we try to use what has worked previously. So we search around inside our minds for old ideas to use."
>
> —8.10 Levels and classifications pp. 91

Thus when we create a new skill, that's to say, a new agent, we make it exploit other agents to help to develop the skill. The way to build *Societies of Agents* proposed in this thesis (see chapter 3) has been designed to help in this aspect.

A very interesting way to develop new agents comes from the concept of *Functional Autonomy*:

> "**Functional Autonomy.** In the course of pursuing any sufficiently complicated problem, the subgoals that engage out attentions can become both increasingly more ambitious and increasingly detached from the original problem.

> Suppose a baby's initial goal was to reach a certain cup. This could lead to the subgoal of learning how to move the arm and hand efficiently, which, in turn, could lead to sub-subgoals of learning to move around obstacles. And this could keep growing into increasingly general and abstract goals of learning how to understand and manage the physical world of space and time. Thus one can begin with a lowly goal, yet end up with some sub-subgoals that lead our mind into the most ambitious enterprises we can conceive."
>
> —17.4 Functional autonomy pp. 177

The earliest goal, for the baby, in this example is the goal of reaching the cup. The other subgoals are later stages. This early stage can teach (say if it is right or wrong) the other subgoals in some aspects. Let's make a picture: The baby's goal is to reach a certain cup, thus an agent will be created to perform this skill. It must learn to exploit other agents, those which help in obstacle avoidance for example. But they may not even exist, in that case they might be created, or they may exist but don't perform the task correctly for the goal of reaching a cup. Thus, in any of the cases, this agent (the one destined to control the skill of reaching a cup) can teach the obstacle avoidance agents to better perform their skill by indicating to them when they did it right and when they did it wrong. Later, when another agent needs obstacle avoidance as subgoal to perform it's task, as the agents involved in obstacle avoidance learnt from the previous experience, they will perform the task more reliably. This new agent will also be able to teach them to do it better by indicating if they did it right or wrong. And thus,

> "... each method helps the others, and all of them together make a mass of skills that grow in power and efficiency."
>
> —18.7 What is a number? pp. 192

Each new agent has something to teach the others. In the case of the baby reaching the cup the way of teaching is obvious: if the baby gets to reach the cup, every thing went right other wise something went wrong. But there are many cases when we learn from the society we leave in, we learn from the persons we live with. And we don't learn the same way from a person we trust in than from a person we hardly know. Minsky says:

> "Many people dislike the thought of being dominated from within by the image of a parent's wish. Yet, in exchange, that slavery is just what makes us relatively free (as compared with other animals) from being forced to obey so many other kinds of unlearned, built-in instinct goals."
>
> —17.8 Attachment-images pp. 181

When we lose a person we were attached to, only time can reconstruct the normal functioning of all the agents involved in this attachment.

> " It is an injury more terrible than loss of sight or limb, to lose the normal use of the agencies with which one builds one's own identity."
>
> —17.9 Different spans of memories pp. 182

### Polynemes

When something new is learnt, it is necessary to store this experience in a way we can easily access our newly acquired capability when it is needed. As Minsky explains, a *Polyneme* is a type of *K-line* agent which helps for it:

> "A *Polyneme* is a type of *K-line*; it sends the same, simple signal to many different agencies: each of those agencies must learn, for itself, what to do when it receives that signal."
>
> —19.3 Words and ideas pp. 198

> "To understand a polyneme, each agency must learn its own specific and appropriate response, Each agency must have its private dictionary or memory bank to tell it how to respond to every polyneme."
>
> —19.5 Polynemes pp. 200

Minsky presents *Polyneme*s as helpful to store experience. To achieve a goal, such as the resolution of a problem or the acquisition of a skill (for example, learn how to make an aerial while surfing), what we generally do is to try different things, until we get to achieve the goal. This in terms of agents, could be seen as sending different instructions to different agents to make them do different things. One specific set of instructions sent to one specific set of agents, may result in the goal being achieved. A *Polyneme* agent then may be created and send to these specific set of agents the same signal. When they first get this signal, they should store their state related to this signal. When the *Polyneme* agent sends this same signal to them again, they should bring themselves to the stored state and make possible to achieve the goal again.

### Concluding Comments

For a robot which has implemented a *Society of Agents*, new capabilities are added by new agents. To make a robot learn new things, is for a new agent to learn how to use or exploit the agents already in the society, to make the robot able to achieve a given goal. Minsky suggests *K-line*s and *Polyneme*s to help for it.

As a starting point for future development of the *Society of Agents* of MiReLa to make it capable to learn new things, the goal presented at the beginning of this subsection is suggested: make possible to teach MiReLa that the phrase in MML "MiReLa Turn Forty" means we want the robot to rotate forty degrees. This could be done by starting to whistle the phrase and rotating the robot forty degrees by hand. After, continuing to whistle the phrase, but not rotating it by hand, and leaving the robot to learn to use, or exploit the agents in its *Society of Agents* to replicate the effect of the movement.

### 10.2.4 Emotions and reasoning

Many approaches are being developed to simulate emotions in robots. Many of them intend to help to achieve a more effective people-robot interaction. The cases of the robots Sage and Minerva were presented in section 2.2. Also for the robot MiReLa, some kinds of behaviours and changes in behaving give the impression that the robot experiences changes in it's mood. This is the case when it is in front of a closed door asking for help to open it, or when the battery level starts to drop and MiReLa seems to be tired and sad. As said in section 9.1.3, in the case of MiReLa no "mood transition scheme" (robot Sage) nor "emotional states" (robot Minerva) were implemented. The goal was to make a more effective alert when a door was found closed and for the case of low battery level, to make the people understand what was going on with the robot without the robot having to say repeatedly in it's music language that the battery had dropped. The result was to give the impression that the robot experiences different moods.

The conclusion is that this result suggests an investigation about what human emotions are, why do we have them, and if it would be useful for a certain type of robot to have human like emotions. As a suggestion to start the investigation, some directives have been collected in this subsection from the book *Descartes' Error* of Damasio. Damasio explains how is composed the human brain, and how does he understand emotions and feelings (to feel emotions). He presents the *Somatic-Marker Hypothesis* to explain the important role emotions play when trying to understand how mind works. He presents them as an important part of the decision making process, for which, Damasio says that we would need a very long time using only reason and that we would probably decide wrong. He says that emotions help us to decide fast and many times correct. Both Minsky and Damasio think that reason and emotion goes together, they can't be separated. The following two quotes have been taken from *The Society Of Mind*:

> "... we'll propose to regard emotions not as separate from thoughts in general, but as varieties or types of thoughts, each based on a different brain-machine that specializes in some particular domain of thought."
>
> —16.1 Emotion pp. 163

> "We like to think of reasoning as rational—yet we often represent our arguments as fights between adversaries positioned to determine which can wield the greater strength or force.
>
> ... we rarely need to know that anything is absolutely wrong or right; instead, we only want to choose the best of some alternatives."
>
> —18.6 Magnitude from multitude pp. 191

Damasio sees that, emotions and feelings help us reduce drastically the possible alternatives to choose among, when we have to take decisions.

> "... our brains can often decide well, in seconds, or minutes, depending on the time frame we set as appropriate for the goal we want to achieve, and if they

> can do so, they must do the marvelous job with more than just pure reason. An alternative view is needed."
>
> —Rationality at work pp. 172

The alternative view is given by emotions, feelings and the *Somatic-Marker Hypothesis*.

> "I see the *essence* of emotion as the collection of changes in body state that are induced in myriad organs by nerve cell terminals, under the control of a dedicated brain system, which is responding to the content of thought relative to a particular entity or event."
>
> —Secondary emotions pp. 139

> "I leave out of emotion the perception of all the changes that constitute the emotional response. ... I reserve the term feeling for the experience of those changes."
>
> —Secondary emotions pp. 139

> "That process of continuous monitoring, that experience of what your body is doing while thought about specific contents roll by, is the essence of what I call a feeling. If an emotion is a collection of changes ... that have activated a specific brain system, the essence of feeling an emotion is the experience of such changes in juxtaposition to the mental images that initiated the cycle."
>
> —Feelings pp. 145

> "In short, feeling your emotional states, which is to say being conscious of emotions, offers you flexibility of response based on the particular history of your interactions with the environment. Although you need innate devices to start the ball of knowledge rolling, feelings offer you something extra."
>
> —Primary emotions pp. 133

Depending on some external events (the presentation of an article in a conference, for example), and some internal events (how well we prepared it, which persons are expected to be in the conference, which kinds of questions were done to the other presenters, etc), the nerve cell terminals induce some kinds of changes in many of our body organs. It is the consequence of the emotions. We have feelings to fell these changes. They offer a way for us to detect the changes and depending on the situation we are in, act in one way or another. When we have to make a choice, we tend to think in the consequences each of the alternatives may have, and we experience different feelings about each of them. Damasio introduces the term somatic-marker:

> "Because the feeling is about the body, I gave the phenomenon the technical term somatic state ("soma" is Greek for body); and because it "marks" an image, I called it a marker. Note again that I use somatic in the most general sense (that which pertains to the body) and I include both visceral and non visceral sensation when I refer to somatic markers."
>
> —The somatic-marker hypothesis pp. 173

And he defines what does a somatic-marker achieve:

> "It forces attention on the negative outcome to which a given action may lead, and functions as an automated alarm signal which says: Beware of danger ahead if you choose the option which leads to this outcome. The signal may lead you to reject, *immediately*, the negative course of action and thus make you choose among other alternatives."
>
> —The somatic-marker hypothesis pp. 173

> "Somatic markers are thus acquired by experience, under the control of an internal preference system and under the influence of an external set of circumstances which include not only entities and events with which the organism must interact, but also social conventions and ethical rules."
>
> —Somatic markers: where do they all come from? pp. 179

Damasio presents emotions and feelings as a way to store experience in our body. They make us to relate some thoughts or events to body states and this helps in the decision making process by reducing the possible alternatives to choose among (those with which we feel better with). Thus they help to decide faster and generally better. He says that if we had to decide using reason only, the process of decision making would be so long and the final decision would probably be wrong or we would never reach any at all. The sector designed to make emotion feeling useful for the decision making process is located in the ventromedial prefrontal cortices of our brain. An experience with a patient with damage in this sector, is presented next. This patient lost the ability to apply emotions to the decision making process. Damasio explains:

> "An example from my experience will help clarify the ideas discussed above. Not too long ago, one of our patients with ventromedial prefrontal damage was visiting the laboratory on a cold winter day. Freezing rain had fallen, the reads were icy, and the driving had been hazardous. I had been concerned with the situation and I asked the patient, who had been driving himself, about the ride, about how difficult it had been. His answer was prompt and dispassionate: It had been fine, no different from the usual, except that it had called for some attention to the proper procedures for driving on ice. The patient then went on to outline some of the procedures and to describe how he had seen cars and trucks skidding off the roadway because they were not following these proper, rational procedures. He even had a particular case in point, that of a woman driving ahead of him who had entered a patch of ice, skidded, and rather than gently pulling away from the tailspin, had panicked, hit the brakes, and gone zooming into a ditch. One instant later, apparently unperturbed by this hair-raising scene, my patient crossed the ice patch and drove calmly and surely ahead. He told me all this with the same tranquility with which he obviously had experienced the incident.
>
> There is not much question that in this instance not having a normal somatic-marker mechanism was enormously advantageous. Most of us would

have had to use a deliberate overriding decision to stop us from hitting the brakes, out of panic or out of sheer feeling for the unfortunate driver in front of us. This exemplifies how automated somatic-marker mechanisms can be pernicious to our behaviour, and how under some circumstances, their absence can be an advantage.

The scene now changes to the following day. I was discussing with the same patient when his next visit to the laboratory should take place. I suggested two alternative dates, both in the coming month and just a few days apart from each other. The patient pulled out his appointment book and began consulting the calendar. The behaviour that ensued, which was witnessed by several investigators, was remarkable. For the better part of a half-hour, the patient enumerated reasons for and against each of the two dates: previous engagements, proximity to other engagements, possible meteorological conditions, virtually anything that one could reasonably think about concerning a simple date. Just as calmly as he had driven over the ice, and recounted that episode, he was now walking us through a tiresome cost-benefit analysis, an endless outlining and fruitless comparison of options and possible consequences. It took enormous discipline to listen to all of this without pounding on the table and telling him to stop, but we finally did tell him, quietly, that he should come on the second of the alternative dates. His response was equally calm and prompt. He simply said: "that's fine." Back the appointment book went into his pocket, and then he was off.

This behaviour is a good example of the limits of pure reason. It is also a good example of the calamitous consequence of not having automated mechanisms of decision making. An automated somatic-marker mechanism would have helped the patient in more ways than one. To begin with, it would have improved the overall framing of the problem. None of us would have spent the amount of time the patient took with this issue, because an automated somatic-marker device would have helped us detect the useless and indulgent nature of the exercises. If nothing else, we would have realized how ridiculous the effort was. At another level, sensing the potentially wasteful approach, we would have opted for one of the alternative dates with the equivalent of tossing a coin or relying on some kind of gut feeling for one or the other date. Or we might simply have turned the decision over to the person asking the question and replied that it really did not matter, that he should choose.

In short, we would picture the waste of time and have it marked as negative; and we would picture the minds of others looking at us, and have that marked as embarrassing. There is reason to believe that the patient did form some of those internal "pictures" but that the absence of a marker prevented those pictures from being properly attended and considered."

—The help of emotion for better or for worse pp. 192-194

**Concluding Comments**

This is a very different point of view about what do we humans use emotions and feelings for (decision making), from what most of the robotic approaches simulate them for (more effective people-robot interaction). Damasio says that emotions lead our body to real physical changes, these physical changes influence other mental processes which may lead to more physical changes, etc, and the whole process makes us able to decide (generally) right in a short time. He presents emotions, and feelings, as a mechanism to store experience in our body.

This translated to the robotic field can be understood as emotions leading to changes in the hardware. In a *Society of Agents* scheme, would be understood as changes to the agents controlling the hardware (agents from the *A-brain* (see sections 3.1.2 and 9.1.2)) which would lead the hardware change its operation, which would affect other agents in the *Society of Agents*, and help to take decisions.

The robot MiReLa did not yet had to make any complex decisions. It doesn't have any agent analysing the consequences of any decision that it takes. For the simple tasks it does for the moment this has not been considered to be needed.

As the *Society of Agents* of the robot MiReLa grows bigger, and thus, the robot has more capabilities, it may come the time when it will be found useful for MiReLa to analyse further consequences when making some decisions. It may happen to be useful for it to guess (based on experience) the consequences of one given action. Then, Damasio's suggestions will be worth to investigate and tried to implement.

# References


[1] Jon B. Allen. How do human process and recognize speech. *IEEE Transactions on Speech and Audio Processing ASP*, 2:567 – 577, October 1994. 64

[2] J.M. Alonso. *TCP/IP. Programación the aplicaciones distribuidas*. ra-ma, 1998. 148

[3] American acoustic society octave designation system. http://www.music.vt.edu/musicdictionary/texto/Octavedesignation.html. 16, 21, 60, 61

[4] P. Aramendia. Memoria prácticas en empresa. Technical report, SEIM, 2004. 87

[5] R.C. Arkin. *Behavior-Based Robotics*. MIT Press, 1998. 21, 107

[6] JVL Industri Elektronik A/S. *mac50, mac95, mac140, mac141, Integrated Servo Motors, user manual*. Blokken 42, DK-3460 Birkerod, Denmark, 2004. 106

[7] CNN associated press. Nearly extinct whistling language revived. *Science and space*, November 2003. 7

[8] B. Astrand and A.J. Baerveldt. Design of an agricultural mobile robot for mechanical weed control. In *4th European Workshop on Advanced Mobile Robots, Eurobot 01*, september 2001. 3

[9] Yoseph Bar-Cohen. Worldwide electroactive polymer actuators webhub. JPL's NDEAA Technologies Lab. http://ndeaa.jpl.nasa.gov/nasa-nde/lommas/eap/EAP-web.htm. 13

[10] R. Bischoff. Advances in the development of the humanoid service robot hermes. In *Second international conference on field and service robotics*, August 1999. 31

[11] R. Bischoff. *Design concept and realization of the humanoid service robot HERMES*, pages 485–492. Springer, 1999. 31

[12] R. Bischoff and V. Graefe. Integrating vision, touch and natural language in the control of a situation-oriented behaviour-based humanoid robot. In *IEEE conference on systems man and cybernetics*, pages 999–1004, October 1999. 34

[13] R. Bischoff and V. Graefe. Demonstrating the humanoid robot hermesat an exhibition: A long-term dependability test. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '02*, Lausanne, September 2002. 32

[14] R. Bischoff and V. Graefe. Dependable multimodal communication and interaction with robotic assistant. In *IEEE International Workshop on Robot and Human Interactive Communication ROMAN 02*, Berlin, September 2002. 34

[15] AIRcable Serial to Serial. AIRcable. http://www.aircable.net/AIRcable/serial.html. 106

[16] M. Bohlen. Universal whistling machine. http://www.realtechsupport.org/new_works/uwm.html See also, Bohlen, M. and Rinker, J. T. Concurso Internacional sobre Arte y Vida Artificial, U.W.M (2004),http://www.fundacion.telefonica.com/at/vida/paginas/v7/whistling.html. 18, 59

[17] Marc Bohlen. Experiments with whistling machines. *LEONARDO Music Journal*, 2005. 18, 181

[18] Graham Booker. *Sensors and Signals, Course Notes*. The University of Sydney, 2005. 101

[19] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions On Robotics and Automation*, 1996. 106

[20] Alan Boyle. Robots are getting more sociable. *MSNBC News*, 2004. 13

[21] Cynthia Breazeal. Regulation and entrainment in human-robot interaction. *I. J. Robotic Res.*, 21(10-11):883–902, 2002. 18

[22] Cynthya Breazeal. Emotion and sociable humanoid robots. *Int. J. Hum.-Comput. Stud.*, 59(1-2):119–155, 2003. 18

[23] Cynthya Breazeal. Toward sociable robots. *Robotics and Autonomous Systems*, 42(3-4):167–175, March 2003. 13

[24] R. Brega, N. Tomatis, K.O. Arras, and R. Siegwart. The need for autonomy and real-time in mobile robotics: a case study of xo/2 and pygmalion. In *International Conference on Intelligent Robots and Systems (IROS 2000)*, Takamatsu, Japan, 2000. 37

[25] S.A. Brewster. *Providing a structured method for integrating non-speech audio into human-computer interfaces*. PhD thesis, University of York, UK, 1994. 18

[26] S.A. Brewster, P.C. Wright, and A.D.N. Edwards. An evaluation of earcons for use in auditory human-computer interfaces. Technical report, 1993. 18

[27] P Brna. *Introduction to AI Programming*. World Wide Web, http://www.comp.lancs.ac.uk/computing/research/aai-aied/people/paulb/old243prolog/243notes96.html, 1995. 113

[28] R Brooks. A robust layered control system for a mobile robot. In *IEE Journal of Robotics and Automation*, pages 14–23, 1986. 107

[29] R Brooks. The behaviour language; user's guide. Technical report, Technical report AIM-1227, 1987. 107

[30] R Brooks. Planing is just a way of avoiding figuring out what to do next. (Also published as [31], chapter 6.). *MIT Artifitial Intelligence Laboratory Working Paper 303*, 1987. 107

[31] R Brooks. *Cambrian Intelligence. The early history of the new AI.* MIT Press, 1999. 107, 188, 207

[32] Allison Bruce, Illah Nourbakhsh, and Reid Simmons. The role of expressiveness and attention in human-robot interaction. In *Proceedings, 2001 AAAI Fall Symposium*, 2001. 13

[33] W. Burgard, A.B. Cremers, Dieter Fox, D. Hhnel, G. Lakemeyer, D. Schulz, W. Steiner, and Sebastian Thrun. Experiences with an interactive museum tour-guide robot. Technical Report CMU-CS-98-139, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1998. 15, 26, 29

[34] C-LAB. *mexi - machine with emotionally extended intelligence.* Paderborn University, http://www.c-lab.de/home/en/news/2001/01-02-21_emotional_engine_lang.html. 13

[35] R. Caughley. Chepang whistle talk. *Speech surrogates: Drum and whistle systems*, pages 997–1022, November 1976. 7

[36] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica K Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005. 11

[37] CEBEK Electronic circuits. *E-2 Etapa de potencia mono de 5 W.* Quetzal, 17-21. (08014) Barcelona, 2004. 92, 96

[38] CEBEK Electronic circuits. *PM-3 microfono con preamplificador.* Quetzal, 17-21. (08014) Barcelona, 2004. 92, 95

[39] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive dynamic walkers. *Science Magazine*, 307:1082–10852, February 2005. 17

[40] Commercial speech recognition. http://www.speechstudio.com/commercial/speech.htm#RECO. 18

[41] Compiled vs interpreted languages. http://www.vanguardsw.com/dphelp4/dph00296.htm. 57

[42] John D Cutnell and Johnson Kenneth W. Johnson. *Physics.* Wiley, New York, 6 edition, 2004. 60

[43] Maslin G. D. A simple ultrasonic ranging system. In *102nd audio engineering society*, 1983. 101

[44] Antonio R. Damasio. *Descartes' Error: Emotion, Reason and the Human Brain.* Grosset/Putnam, 1994. 5, 43, 190

[45] B. L. Davies, R. H. Taylor, S. Lavallee, G. C. Berdea, and R. Mosges. A discussion of safety issues for medical robots. *Computer Integrated Surgery*, pages 287–296, 1996. 3

[46] Dual-tone multi-frequency (DTMF). see DTMF. Wikipedia, http://www.music.vt.edu/musicdictionary/texto/Octavedesignation.html. 87, 183

[47] Prassler E., Ritter A., Schaeffer C., and Fiorini P. A short history of cleaning robots. *Autonomous Robots*, 9(3):211–226, 2000. 3

[48] Intelligent Earth. *Home Page*. World Wide Web, http://www.intelligent-earth.com/html/index.html. 13

[49] Medonis Engineering. *Home Page*. World Wide Web, http://www.medonis.com. 13

[50] U. Esnaola, I. Rañó, and T. Smithers. A behaviour-based navigation system. In *2004 (TAROS'2004)*, SEPTEMBER 2004. 99, 187

[51] U. Esnaola, I. Rañó, and T. Smithers. Robot MiReLa: Helping the secretary to attend visitors. In *2004 2nd International Workshop on Advances in Service Robots (ASER 2004)*, MAY 2004. 3, 99, 148, 187

[52] U. Esnaola and T. Smithers. Mirela: A musical robot. In *6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'2005)*, JUNE 2005. 158, 187

[53] U. Esnaola and T. Smithers. Whistling to machines. In *Ambient Intelligence and (Everyday) Life*, JULY 2005. 187

[54] D. Everett. Cultural constraints on grammar and cognition in pirahã: Another look at the disgn features of human language. *Current Anthropology*, October 2005. 7

[55] C. Fedor. *TCX. An interprocess communication system for building robotic architectures. Programer's guide to version 10.xx.* Carnegie Mellon University, Pittsburgh, PA 15213, 12, 1993. 28, 30

[56] I. Fumiya and R. Pfeifer. "Cheap" rapid locomotion of a quadruped robot: self-stabilization of boundin gait. In *The 8th Conference on Intelligent Autonomous Systems (IAS-8)*, March 2004. 17

[57] see Whistled Language. Wikipedia, http://wikimediafoundation.org/wiki/Whistled_language/. 7

[58] Boehme H.-J., Wilhelm T., Hempel T., Schroter C., and Gross H.-M. An approach to multimodal human-machine interaction for intelligent service robots. *Robotics and Autonomous Systems*, 44:83–96, 2003. 3

[59] M. Hillman. Rehabilitation robotics from past to present - a historical perspective. In *ICORR 2003(The Eighth International Conference on Rehabilitation Robotics)*, April 2003. 3

[60] Boehme H.J., Brakensiek A., Braumann U.D., Krabbes M., and Gross H.-M. Neural architecture for gesture-based human-machine-interaction. *Gesture and Sign-Language in Human-Computer Interaction. Lecture Notes in Artificial Intelligence*, pages 219–232, 1998. 6

[61] Gross H.M., Boehme H.J., Key J., and Wilhelm T. The perses project - a vision-based interactive mobile shopping assistant. *Kunstliche Intelligenz*, 4:34–36, 2000. 3

[62] Honda. *Asimo*. World Wide Web, http://asimo.honda.com/index.asp?bhcp=1. 16

[63] Geoff Husband. *What's in your Music*. World Wide Web, http://www.tnt-audio.com/topics/frequency_e.html, 1999. 60

[64] Kawada Industries. *Humanoid robot HRP-2P Prototype*. World Wide Web, http://www.kawada.co.jp/ams/hrp-2/index_e.html. 16

[65] ISP glossary. 13

[66] J. Jones and A.M. Flynn. *Mobile robts. Inspiration to implementation*. A K Peters, 1993. 104

[67] O. Karolyi. *Introducing Music*. Penguin Books, 1965. 21, 61

[68] Z. Karu. *Signals and sstems. Made Ridiculously simple*. ZiZi Press, 1995. 72

[69] Faulder L. Space-age vacuum cleaner senses dirt, potential disasters. *Edmonton Journal*, 2004. 3

[70] E. Lazkano, A. Astigarraga, J.M. Martinez-Otzeta, and B. Sierra. Postrob: a behaviiour-based post deliverer. In *2004 (TAROS'2004)*, SEPTEMBER 2004. 23

[71] W. S. Lee, D.C. Slaugther, and D.K. Giles. Robotic weed control system for tomatoes. *Precision Agriculture*, 1:95–113, 1999. 3

[72] J. Liebler. *Examinations on the Use of an Automatic Milking System on a Dairy Farm*. PhD thesis, Technische Universitat Munchen, 1997. 3

[73] MICROPIK. *PS6E8S, manual de instructiones*. Profesor Blanco 18 Bajos, 46014-Valencia, Spain, 2003. 16, 69, 70, 84

[74] Marvin Minsky. *The Society of Mind*. Wiliam Heineman Lta, 1987. 5, 43

[75] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *AAAI National Conference on Artificial Intelligence*, 2002. 3

[76] C. Morris. Whistling turks. *BBC News*, November 1999. 7

[77] U. Nehmzow. *Mobile robotics: a practical introduction*. Springer, 2000. 104, 105

[78] David Ng. *Robotic Head by David*. World Wide Web, http://members.shaw.ca/davidng/robot/index.html. 13

[79] Nomad xr400. http://www.cbcis.wustl.edu/about/xr4000/XR4000.htm. 21

[80] NORCOMM. *Model NC400 Multi-Function DTMF Decoder Intruction Manual*. 15385 Carrie Drive, Grass Valley, CA 95949, 2004. 87, 88, 92, 95

[81] Illah R. Nourbakhsh, Judith Bobenageb, Sebastien Grangec, Ron Lutzd, Roland Meyerc, and Alvaro Soto. An affective mobile robot educator with a full-time job. *Artif. Intell.*, 114(1-2):95–124, 1999. 3, 13, 20, 23

[82] J.R Pierce. *The Science of Musical Sound*. Scientific American Books. W.H Freeman and Company, New York, 1983. 191

[83] I. Rañó. *Investigación de una arquitectura basada en el comportamiento para robots autónomos en entornos semi-estructurados*. PhD thesis, San Sebastián Technology Park, 2004. 107

[84] Patric M. Rael. *Robot Maximilian*. World Wide Web, http://www.howtoandroid.com. 13

[85] Real World Interface Inc. *B21 User's guide*, 1996. 3, 8, 100

[86] Nicholas Roy, Gregory Baltus, Dieter Fox, Francine Gemperle, Jennifer Goetz, Tad Hirsch, Dimitris Margaritis, Michael Montemerlo, Joelle Pineau, Jamieson Schulte, and Sebastian Thrun. Towards personal service robots for the elderly. In *Workshop on Interactive Robots and Entertainment (WIRE 2000)*, 2000. 3

[87] Nicholas Roy, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. In *Proc. IEEE Conf. Robotics and Automation (ICRA)*, May 1999. 28

[88] P Sanchez. *Harmonic Mean*. PlanetMath.org, http://planetmath.org/encyclopedia/HarmonicMean.html, 2001. 107

[89] M. Schofield. "neither master nor slave ..." a practical case study in the development and employment of cleaning robots. In *IEEE 7th Int. Conf. on Emerging Technologies and Factory Automation ETFA'99*, pages 1427–1434, 2000. 3

[90] M. Schofield and H. Grunke. Cleaning robots from concepts to product - the users point of view. In *25th Int. Symp. on Industrial Robots ISIR'94*, pages 233–234, 1994. 3

[91] Jamieson Schulte, Chuck Rosenberg, and Sebastian Thrun. Spontaneous short-term interaction with mobile robots in public places. Technical report, Computer Science Department, Carnegie Mellon University, 1999. 27, 28, 29, 30

[92] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A.B Cremers. Web interfaces for mobile robots in public places. *IEEE Robotics and Automation Magazine*, 7(1), 2000. 26

[93] J. Schwarz. Shepherds whistle while they work and brains process sounds as language. *Science and Tech*, January 2005. 191

[94] W. Shetter. Whistling a language. In *Language miniatures: A selection of essays*. 1998. http://home.bluemarble.net/~langmin/miniatures/whistle.htm. 7

[95] Y. Shi-Dong. Birth of Korean humanoid robot marks brilliance advance in Korea robotics. *Biz Tech*, December 2004. 16

[96] Kaho Shimizu. Toyota trunpets walking robots. *The Japan Times*, March 2004. 16

[97] Roland Siegwart, Kai O. Arras, Samir Bouabdallah, Daniel Burnier, Gilles Froidevaux, Xavier Greppin, Bjrn Jensen, Antoine Lorotte, Laetitia Mayor, Mathieu Meisser, Roland Philippsen, Ralph Piguet, Guy Ramel, Gregoire Terrien, and Nicola Tomatis. Robox at expo.02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42(3-4):203–222, March 2003. 3, 39

[98] Reid Simmons, Joaquin Lopez Fernandez, Richard Goodwin, Sven Koenig, and Joseph O'Sullivan. Lessons learned from xavier. *Robotics and Automation Magazine*, pages 733 – 39, 2000. 3

[99] D.C. Slaugther, R. Curly, P. Chen, and C. Brooks. Development of a robotic system for a nonchemical weed control. In *Annual California Weed Conference*, 1992. 3

[100] Eric Smalley and Susanna Space. Interactive robot has character. *Technology Research News*, March 2002. 13

[101] W. Smart, S. Tejada, B. Maxwell, A. Stroupe, J. Casper, A. Jacoff, H. Yanco, and M. Bugajska. The 2004 mobile robot competition and exhibition. *AI MAGAZINE*, 2005. 40

[102] S.W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, 1997. 81

[103] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B. Cremers, Frank Dellaert, Dieter Fox, Dirk Hahnel, Charles R. Rosenberg, Nicholas Roy, Jamieson Schulte, and Dirk Schulz. MINERVA: A tour-guide robot that learns. In *KI - Kunstliche Intelligenz*, pages 14–26, 1999. don't have the paper. 13

[104] Sebastian Thrun, Arno Buecken, Wolfram Burgard, Dieter Fox, Thorsten Froehlinghaus, Daniel Hennig, Thomas Hofmann, Michael Krell, and Timo Schmidt. Map learning and high-speed navigation in RHINO. Technical Report IAI-TR-96-3, Computer Science Department, Carnegie Mellon University, 15, 1996. 28

[105] N. Tomatis, G. Terrien, R. Piguet, D. Burnier, S. Bouabdallah, Kai O. Arras, and R. Siegwart. Designing a secure and robust mobile interacting robot for the long term. In *ICRA 03*, 2003. 35

[106] Joschen Triesch. *Antropomorphic Robot Head*. YFX Studio, http://cloudbreak.ucsd.edu/~triesch/robotHead/. 13

[107] Trilobite. http://www.trilobite.electrolux.se (referred on 31/08/2005). 3

[108] Rosa M. Tristan. Zapatero se compromete a duplicar la inversión en I+D+i para el año 2010. *El Mundo*, 24, June 2005. 158

[109] P. Vicker and J. L Alty. Siren songs and swan songs, debugging with music. In *Communications of the ACM*, pages 87–92, July 2003. 17

[110] Matthew Walker and Peter S. Longyear. *Robot Inkha*. World Wide Web, http://www.inkha.net/. 13

[111] watchdog. http://www.webopedia.com/TERM/W/watchdog.html. 55

[112] David Whitehouse. Japanes develop 'female' android. *BBC News website*, 2005. 11

[113] Wikipedia. Network port. http://en.wikipedia.org/wiki/Computer_port. 53

[114] Chris Willis. *Android World*. World Wide Web, http://www.androidworld.com/. 12

[115] Chris Willis. *The Changing Face of Robots*. JPL's NDEAA Technologies Lab, http://www.androidworld.com/prod50.htm. 13