# Non-verbal Social Robot-Human Interaction

by

Paul Plasman
(student number: 1071157)

November 2005

Supervised by:
Dr. B. De Boer
Prof. A. Bonarini

**Kunstmatige Intelligentie**
**RijksUniversiteit Groningen**

# ABSTRACT

This thesis discusses the concretization and implementation of a simulation of an autonomous, mobile tour guide robot. The robot has a non-anthropomorphic shape, and is equipped with an omni-directional camera on top of it.

The robot gives its tour by sequentially visiting a list of target positions which it receives beforehand. All interactions with the public that are involved in the tour will take place by movement of the robot's body and its occupation of the space. The communication with the people will be of an implicit, yet expressive action-based type. Without having a human form, it must display certain 'gestures' that are general and at the same time convincing enough to be understood by the human observer. This form of non-verbal dialogue can make use of so-called social norms and conventions. An example of this kind is proxemics, the social use of space. Knowing how to stand in line, walking through a crowded hallway etc, are examples of this.

The behaviours are implemented on BRIAN, a multi-layered behaviour architecture that makes use of fuzzy logic in its reasoning process.

# Table of Contents

# 1. INTRODUCTION

This thesis focuses on the *psychological aspects* involved in the interaction of an artificially intelligent, autonomous robot with humans. As a robot's contact with humans becomes closer, good attention should be paid to the aspects that influence these interactions.

A practical and entertaining domain to study this is the area of robots that serve as a guide for people in a public area. The major task of these robots is to tour people around the area, while at the same maintaining the interest of these people by interacting with them. This interaction involves a two-way exchange of information. The robot reacts to the movements of the person, and the human in turn interprets the robot's actions and responds to these.

In this thesis, I sketch an overview of some important features of communication and interaction, connecting these to the tasks of a tour guide robot. The main focus lies on describing the concretization of a few of the *non-verbal* aspects of behaviour – the communication without words. The implementation of a non-verbally communicating tour guide has been realized in a fuzzy logical architecture.

# 2. ROBOT-HUMAN INTERACTION

First, a number of aspects and means of embodied communication will be discussed, and the factors that influence the way that the transmitted message can be interpreted. In the end I will summarize which aspects are relevant for the current implementation and how they have been used.

## *2.1 Communication and Interaction*

First, some important basics and characteristics of communication will be discussed together with the potential implications and difficulties that are inherent to the communication process.

### 2.1.1 Communication: An Overview

Communication is a process that involves *conveying information* from one party to the other. In the definition as formulated by Shannon, communication is a process that involves a sender, a transmission medium and a receiver (Shannon, 1948).

A distinction can be made between explicit versus implicit communication. Explicit communication is the "intentional transmission and reception of information" (Zebrowski, 2004), for example as it happens in a dialogue between two humans. Humans use verbal utterances to intentionally exchange information with each other. In other words, both parties are actively acting to exchange a message. Implicit communication to the contrary emerges from the interaction between an agent and its environment (Arkin et al., 1994). An example of this type can be found in ant colonies. Whenever an ant is foraging and finds food, it picks it up and walks back, carrying the food to the nest. While it is holding food, it leaves behind a trail of a certain substance. If another ant notices this trail, it tends to follow it thereby increasing its likelihood of finding the food source and reinforcing the trail by also leaving the substances as it ports the food back to the nest. In this way, the ants modify their environment thereby implicitly communicating the presence of food in a certain direction.

Characteristics of the communication process that have been observed by other authors involve amongst others interaction distance – the distance between the communicating agents – and sophistication of interpretation, which is an indication of the complexity of the interpretation process that gives meaning to the transmitted signal. To understand sophistication of interpretation, take for example the difference between the chain of chemical events that a simple chemical signal induces in a bacterium versus the process of interpretation of human language, which is currently the most complex example as known (Jung et al., 2000).

### 2.1.2 Meaning of communication and intersubjectivity

Meaning of communication and intersubjectivity, in other words: understanding what is being said. During a communication process, it is obvious that both parties should understand the meaning of the message that the other party is transmitting. Otherwise no logical communication would occur and the exchange of information would be fruitless. However, each agent has only access to its own thoughts and immediate experience. Without having access to the thoughts and experiences of the transmitting agent, the receiving agent should still interpret the message in the way as the sender intended. This topic is known as intersubjectivity (Dourish, 2001). The difficulty that this brings along is that an agent can only infer about another agent's internal model from what it sees from the outside. Seeing 'what is inside the box' is impossible, therefore one can only guess about the other agent's intentions by observing the totality of its actions.

### 2.1.3 User Expectations

Referring back to intersubjectivity, when two agents are engaged in conversation they both make assumptions about the other agent's internal model of the world. From these assumptions, certain expectations arise about what can be expected from each other. Take for example the difference between a human communicating with another human, and the same human communicating with his dog. Human-human communication will most likely involve the use of natural language and the use of abstract terms, while human-dog communication will be more sound- and gesture-based and the topic of communication will be much simpler. This happens because we know from experience (after all, expectations are based on assumptions developed by experience) that dogs have lower intelligence and brain capacity which does not enable them to engage in a natural language-conversation. From this we infer that we should communicate with the dog in a different way and that also should *expect* less from it (we do not expect our dog to ask for further explication when we whistle it for attention). These expectations are also influenced by a factor called *morphology*, which will be discussed later on in this chapter.

### 2.1.4 Non-verbal communication: Communication by Action

In the communication process amongst humans, the most striking method of communication is in the form of *spoken language* in which words, containing symbolic meanings, are exchanged by verbal actions. However, there is vastly more to human communication than the vocal exchange of words. Aspects as bodily postures, hand gestures, direction of gaze, etc. can enhance the message that is being conveyed. Sometimes only a gesture with the hand can express as much as an entire set of phrases (for example, a baby pointing to object to indicate that it wishes to get the object). This part of communication is *non-verbal communication*, in other words, using action to express one's intentions.

This section focuses on two forms of non-verbal communication that are used for the current implementation – communication by making use of one's body and communication by making use of the environmental space. This second type of communication is known as *proxemics*. In short, proxemics can be defined as "the study of the ways in which man gains knowledge of the content of other men's minds through judgements of behaviour patterns associated with varying degrees of [spatial] proximity to them." (Hall, 1966). Examples of proxemic norms are knowing how to stand in line, how to walk through a crowded hallway, etc. (Dautenhahn et al., 2003). Summarized, proxemics is the social use of space. Therefore, by manipulating the inter-personal space between two agents, certain intentions can be conveyed.

The other method of non-verbal communication that I mentioned - making use of one's body - can either be moving the whole of the body or moving individual parts of the body. Deliberately turning away from a person (moving the whole of the body) or pointing towards an object (moving parts of the body) are examples of this. This type of non-verbal communication is called kinetics. The entire set of gestures that humans use during conversation is part of kinetics.

A problem arising while using proxemics and kinetics as part is that the interpretation of these can differ from culture to culture. Cultural differences have a great influence on for example 'allowable' interpersonal difference. Hall has identified four bodily distances which indicate certain 'spheres' around someone in which that person tolerates different types of people, depending on the situation and the affinity with the person. These spheres and their distances are roughly – for American people – intimate (0 to 18 inches), personal-casual (1.5 to 4 feet), social-consultive (4 to 12 feet) and public (12 feet and further) (Hall, 1966). During interaction, care should be taken to take into account these distances, since for example 'trespassing' within someone's intimate distance as a public person (not a close friend of the other) might be regarded as not being appropriate and can give rise to feelings of discomfort or even aggression. Therefore, during communication the interpersonal distances should always be respected to avoid possible confusion.

Not only proxemics can be interpreted in a wrong way due to inter-cultural differences, this is also true for kinetic actions. Certain gestures that in one culture indicate approval might be regarded as highly offensive in others.

When kinetics are used in a proper way – when types of gestures are used that do not leave room for other interpretations (Nicolescu et al., 2001) – they are a powerful method of using language that is commonly interpretable and can be used in a variety of implementations. Nicolescu et al. have used this in their implementation that enabled a robot to 'ask' a human for help when it failed to perform a certain task. This calling for help was accomplished by movements of the robot's body only (Nicolescu et al., 2001).

## 2.2 SOCIALLY ACTIVE ROBOTS

In this discussion of interaction between humans and robots, the focus is on *socially active robots*. These are robots that more or less actively search the interaction with a human being, and attempt to interact with them in a 'social' way. That is, in a way that is consistent with the beliefs and expectations of the human observer.

### 2.2.1 Socially Active Robots

Breazeal et al. argue that people, when they observe and interact with an autonomous robot, generally apply a social model to this interaction (Breazeal et al., 2002). Therefore, they use the term *social robots* to describe this type of autonomous robots. If the robot's behaviour adheres to a person's social model and acts according to that person's expectations then the robot is said to be *socially intelligent*.

There are different degrees in which a robot can show social behaviour. In the simplest form of social interaction, the robot is said to be *socially evocative*. Its behaviour is simple but not much more than just that. It interacts with humans, but for both parties there is not a lot of mutual gain in the interaction. When the robot is able to communicate, it is said to be *socially communicative*. However, as Breazeal et al. argue, the social behaviour 'ends at the interface' and is often nothing more than reflexive behaviour. This means that the robot reacts to a few predefined situations but is not inherently a social entity. One step higher are the *socially responsive* robots. These robots actually benefit from their interaction with people. An example of this is when they use their interaction with a human to learn a task (Nicolescu et al., 2001). Still, these types of robots are to a certain extent socially passive since they do not actively search the interaction with humans to satisfy internal social aims. The highest degree of social behaviour is displayed by robots that are characterized as being *sociable*. Here, interacting takes place to satisfy internal aims of the robot (again for example. for learning),

but also benefits humans while doing so (for example, helping the human to perform a task).

### 2.2.2 Morphology

An important aspect that influences the way that information that is transmitted by an agent is interpreted is the *embodiment* of the communicating agent. Firstly, the agent's morphology defines the possible set of actions that it will be *able to perform*. For example, an agent that has a cube-like shape will not be able to tilt its 'head' as an animal would be able to do, and an agent that has no method of moving its face will not be able to make use of facial expressions as humans would be able to do.

More important, an agent's morphology influences the *expectation* that a human observer develops for the agent. Depending on the agent's shape, a human that encounters the agent will develop certain expectations about the intellectual and behavioural capacities of the agent. An agent resembling a human "elicits strong expectations about the robot's social and cognitive competencies." (Dautenhahn, 2002). When these expectations are not met, it is likely that the other communication party experiences confusion, frustration and disappointment (Dautenhahn, 2002).

Masahiro Mori described this correlation between anthropomorphism of a robot and emotional response to that robot in the graph as depicted in figure 2.1. The sudden drop in the line to the right is what he called the *uncanny valley*. This drop in experienced positive emotional response occurs when the intimacy between a human and robot increases as the robot's appearance and behaviour become more human-like but suddenly drops as the robot fails to meet certain expectations about its behaviour.

Summarized, with increasing humanlikeness of an agent's morphology and increasing behavioural complexity, the overall humanlikeness of the agent increases. However, when at a certain point the agent fails to show behaviour that is lifelike enough to match with its morphology, a somewhat uncomfortable feeling arises in the human spectators. At these points it seems to be interacting with a zombie-like creature, an animated being that is just not-quite human enough.

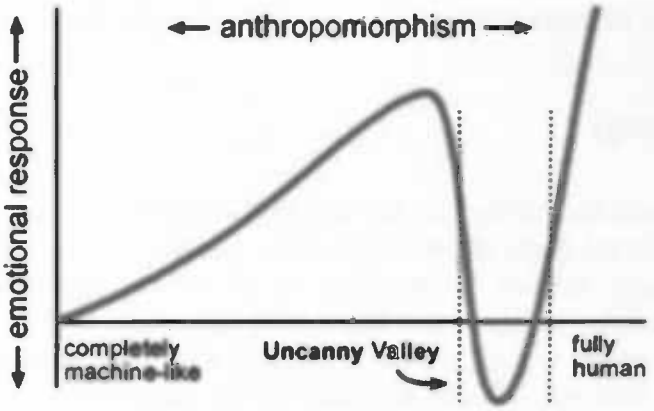



*Fig 2.1: The Uncanny Value (Bryant, 2005)*

### 2.2.3 Expressiveness

Behavioural complexity might be limited by morphology, a non-anthropomorphic agent still has a great ability to convey internal states and intentions to a human. As investigated by McAleer et al. (McAleer et al., 2004), whenever humans see an object moving in a certain way they tend to attribute a social meaning and purpose to that movement. Even when the object is nothing more than a moving square, people still have attributed a certain 'aliveness' to the object. In particular, objects that were moving faster or showed more diverse movements were regarded as more 'alive' than objects that were moving slower and in straighter lines. Therefore, expressiveness does not seem completely limited by morphology.

Another example of how simple creatures can be very expressive are the famous creatures as described by Braitenberg (Braitenberg, 1984). These types of robot are no more than simple car-like creatures that behave according to simple rules, however the behaviour patterns that emerge look highly expressive and humans tend to immediately ascribe certain attributes to these creatures (shy, exploring, etc).

### 2.2.4 (Psychological) Involvement

In order to create an interaction with the human that should continue for at least a little while, it is important that the robot creates a certain involvement by the human in its behaviour. In other words, there should be a proper type of interaction that serves to sustain the interest of the human in the robot's actions. Should the robot fail to create this interest, the communication process is less likely to be fruitful and the human may abandon the robot altogether.

The interest should be created and maintained by a type of interaction in which the robot finds a way to convey its intentions to a human and creates a certain involvement and believability from the human in the robot's actions. The information that the robot transmits should be easily understandable by the human receiver. However, not only understanding what is being communicated will lead to involvement. The robot will need to perform actions that carry intentional meanings (Nicolescu, 2001) and in this way attempt to convey information about its internal states to the human. In past research, this has been achieved by a non-anthropomorphic robot which communicates by making use of actions, whose outcomes are common regardless of the specific body performing them (Nicolescu, 2001). In this way, the communication medium leaves no room for ambiguity and it easily becomes clear what the robot intends) and so user expectations should match the robot's intentions.

## 2.3 CONCLUSION

This chapter has given an overview of communication in robots, and especially the non-verbal part of this. Non-verbal communication is limited by a few aspects such as the robot's morphology. However, this type of communication does avoid certain difficulties and can even exploit advantages of its own simplicity. For example, the human expectation towards non-anthropomorphic creatures is lower than the expectation towards anthropomorphic ones. When the transmitted message is simple and performed in an unambiguous, straightforward way, non-verbal communication can be a powerful way by which non-anthropomorphic creatures can communicate with humans. Also, since the sophistication of interpretation in these types of implementation is simple, it might also be used for robot-robot visual communication.

When we now describe the current implementation in the light of the reviewed aspect can be said that this robot is something between a socially communicative and a socially responsive robot, since no internal state of the robot is affected but it *does* benefit the robot in the current situation. The robot's morphology is extremely simple, and the robot creates a certain expressiveness by making use of proxemic and kinetic actions. In the later sections it will be made clear how this is actually implemented.

# 3. ASPECTS OF AN AUTONOMOUS ROBOT TOUR GUIDE

This chapter gives an overview of service robots that interact with people during their daily functioning. Important requirements that these robots should meet in order to function properly are discussed, also in the light of previously developed and tested implementations. The focus will be largely on robots that function as a (tour-)guide in a public area.

## *3.1 Tour Guides: Museums and other Public Buildings*

A common place where autonomous tour guides have been put to use is in museums. Museums are the kind of public buildings that provide a challenging task for any autonomous agent to give an entertaining and correct tour. Museums are likely to contain a large number of people that behave to a certain extent in an unpredictable way and have large open spaces with numerous obstacles present. They challenge the robot to be adaptive, interactive and flexible. Most of the examples of embodied tour guide robots describe museum robots, therefore this overview will be focused largely on this type of robots.

### 3.1.1 Requirements

Robots operating in a public environment that come into close contact with people should satisfy a few important requirements. I will discuss these in the light of previously developed implementations of service robots and comments by their developers.

- Safety

The first, most important requirement is that a robot does not pose a danger to its environment (people, objects) and not to itself. It should be *safe* in its functioning and should not harm any individual in its presence.
This also implies that the robot should be capable of preserving itself. It has been discovered that people can experience great pleasure in obstructing the robot's path to let it move in another direction (Burgard et al., 1999), so the robot should be capable of avoiding that it bumps into the people and moving itself out of the way.

- Autonomy

Should a robot work unsupervised, it should possess sufficient *autonomy* – the second important requirement. Autonomy can be briefly described as the robot's ability to function without external supervision. While a robot is navigating by itself, it should not get stuck in situations where it is unable to backup from. Additionally, it should not require help to get to certain places. Crossing doorsteps, driving through doorways etc. should not pose a problem. If these

should pose a problem, the robot's design should be altered or the robot should not be required to get to these places. It is always possible to equip a robot with a certain messaging system that enables it to call for help (Nourbakhsh et al., 2003) but this is not desirable as it obviously reduces its autonomy.

- Be Interesting / Be Aware

When a robot is supposed to interact with the people around it, it should be *interesting*. Together with the requirement of autonomy, this is one of the most important requirements that a robot should fulfil. Since it is a service robot, without developing and maintaining a proper interaction with humans its existence has no purpose.

An important factor that contributes to making a robot interesting is the fact that the robot is able to show *awareness*. A robot should make apparent that it is aware of the people around it and with whom it is interacting. Awareness could be made clear by adapting the robot's position and orientation to the position of the people, or a camera mounted on the robot can be used to indicate the robot's current point of focus, signalling to which persons or objects currently its attention is directed (Burgard et al., 1999).

As noticed by Burgard and colleagues in their implementation of an interactive museum tour guide, people believed it more important that a robot was sufficiently able to interact with people than its ability to navigate through the building. (Burgard et al., 1999). The same observation has been done by Tomatis et al. that "awareness has been found as one of the most relevant characteristic for man-machine interaction for the human" (Tomatis et al., 2002).

### 3.1.2 Tasks

Roughly, the job of an autonomous tour guide can be divided into three tasks that it attempts to fulfil during its functioning. These points have been used by Schulte et al to summarize the important tasks of their museum guide robot (Schulte et al., 1999).

- *Travelling*: Navigating around the building from goal to goal.
- *Attracting people*: Convincing people to follow it while it gives a new tour, attempting to draw their attention.
- *Engaging people's interest and maintaining their attention*: While touring or explaining specifics of certain exhibits, it should be sufficiently interesting for people to follow the robot (see also the requirement of being interesting).

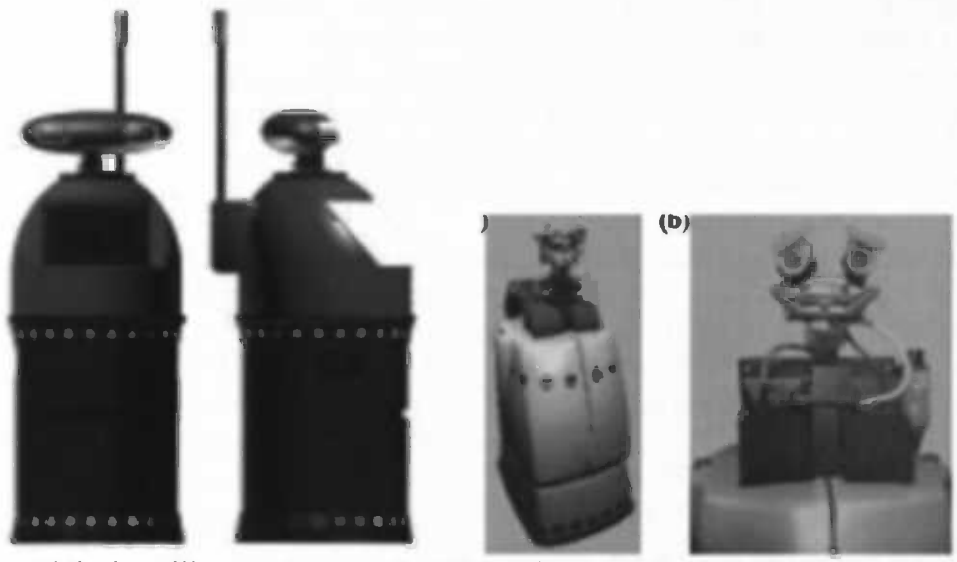
For the current implementation they have been used as reference to make sure a sufficiently complete set of behaviours was constructed.

### 3.1.3 Sensors

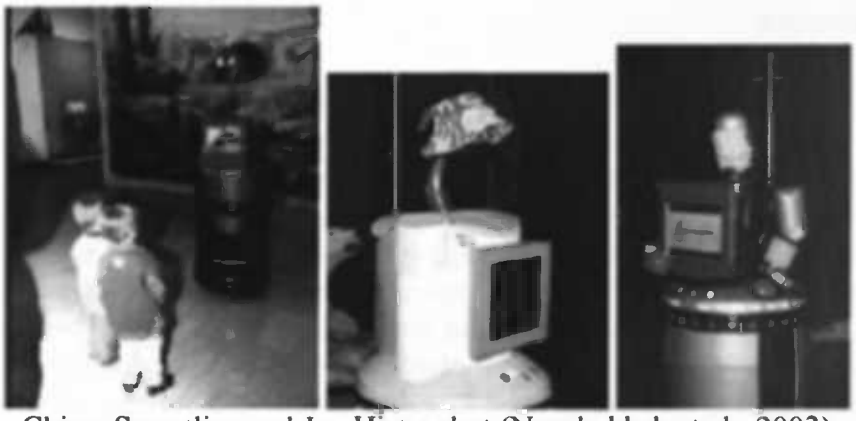
Needless to say, a robot that works in close contact with humans should be equipped with a proper set of sensors. Detection of possible hazardous obstacle should be infallibly present to guarantee a safe functioning. For the interaction with humans, proper detection of human presence should be available. One way of doing this can be realized by extracting features from camera images (Tanawongsuwan, 1999). The exact choice of sensors however is not of great importance for the problem that is being investigated in this project, as long as it provides sufficiently unambiguous information about the environment (i.e. a man-sized plant should not be recognized as a human and distract the robot from its task).

### 3.1.4 Appearance

Many robots that have been to use in a museum possess a human face to appeal more to the visitors' intuition (Castrillón Santana et al., 2001; Thrun et al., 1999; Nourbakhsh et al., 2003). As observed, visitors showed a tendency to initially treat the robot as a human being instead of a machine. Treating it as a human being means going out of its way, verbally responding to it, et cetera (Nourbakhsh et al., 2003).
However, as mentioned in the previous chapter about interaction, as long as the robot is sufficiently capable to fruitfully interact with a human this requirement is not of highest importance. Still, most of the museum robots that are discussed in the literature possess humanlike faces. A few of these can be seen below.




Eldi (Castrillón Santana et al., 2001) Minerva (Thrun et al., 1999)



Chips, Sweetlips and Joe Historybot (Nourbakhsh et al., 2003)

Each of these robots has a human-like face as focal point. The rest of the body contains either a screen that might display information about what the robot is currently performing or consists of a metal casing that covers the robot's hardware.

### 3.1.5 Ways of Interaction

A robot can interact with people in a great variety of ways, ranging from advanced speech interfaces to non-verbal motion-directed interaction. In museums, service robots are often equipped with a speaker, with which it expresses information or its current intentions to the crowd (i.e. Nourbakhsh et al. 2001). In a public building like a museum, the interaction between the robot and visitors will often be of short duration, since most people visit the building with other intentions than just playing and interacting with a robot. Therefore, the need to implement a hugely diverse and ingenious communication mechanism does not arise. In certain implementations a small set of pre-recorded speech messages is sufficient to convey the robot's wishes to the crowd and at the same time entertain them by giving auditive comments about the current situation.

If the interaction is of short duration, the robot does not need to have a full set of social capacities but can also make use of certain tricks. Pre-recorded speech that gives information or reacts to certain situations (Burgard et al., 1999) and special movements provide sufficient entertainment when presented to a new visitor but when these repeat after a certain period of time they will become predictable and maybe even boring at the end. However, as mentioned earlier, most likely this situation will not occur since normally the people are not present in the building to just interact with the robot – the robot's presence is only an extension to their presence which just adds to the excitement.

In the current implementation, all methods of communication by use of sounds or light signals have been abandoned to solely focus on the non-verbal aspects of the human-robot interaction.

### 3.1.6 Navigation

Guiding people through a building can happen either in a previously defined fashion – every tour the robot will follow the same pattern – or it can happen dynamically as visitors can indicate where the robot needs to go. In all implementations, obstacle avoidance has taken up a very important place. In existing implementations (i.e. Thrun et al., 1999) navigation is accomplished by recognizing the robot's current position within a building and using that information calculating the path to take. My robot uses a simpler approach since it knows its location by a GPS-like sensor.

## *3.2 Other service bots*

So far, this chapter has focussed solely on museum tour guide robots. Although this thesis is aimed at an implementation like these, there are a great variety of other applications that have been developed to take on close and intensive interaction with human beings. I will describe a few of these here.

In the entertainment industry two great examples of highly interactive autonomous 'creatures' can be found in the form of Sony's robot dog AIBO and NEC's Papero. These robots can be described as toy-like creatures that have a complex set of behaviours with which they can take on interaction with humans. AIBO's dog interface has even been completed with a possibility of petting the dog by which the robot gets feedback about its current actions.

A more serious approach has been developed in the form of 'Roball' (Michaud, 2000) which serves to interact with children with autism. The ball/robot interactively plays with the child by moving around autonomously through the environment and asking the child it to pick it up or spin it. In this way the researchers have attempted to open the child up to its surroundings.

## *3.3 Conclusion*

Robots that interact closely with humans need to appeal to these humans. A method of doing this is often realized by enabling the robot to show that it is aware of the people around it. Apart from this, to be put in a public space crowded with humans, the robot should possess the ability to work alone (autonomously) and safely. The major part of tour guide robots that have been put to use are morphologically somewhat similar to humans. This is not one of their main requirements. Anthropomorphism might appeal more to humans in a social setting, but as shown in the rest of this thesis, it is not limited to this aspect and could also have negative drawbacks.

# 4. THE ARCHITECTURE

This chapter will give a detailed overview about the architecture in which the robot's behaviour has been implemented and the type of (fuzzy) reasoning that it uses to produce output.

## *4.1 Behaviours*

Before focusing on how the implementation of behaviours can take place, I will take a closer look what exactly *is* a behaviour and how it can be described within a fuzzy architecture.

A distinction can be made between reactive versus planned behaviour. Reactive behaviour deals with the current environmental condition, for example avoiding obstacles, whereas planned behaviour tries to fulfil predefined goals such as reaching a target position. Generally, an autonomous agent needs to fulfil both types of goals.

Many implementations have adopted an approach that uses two levels of control to tackle this problem – one higher-level planner that formulates the goals to be achieved and at the lower level is a controller which produces physical movements and attempts to fulfil these goals while dealing with the environment. A problem using this approach is that the controller needs to be rather complex, since the physical movements need to match both types of goals to the highest degree possible.

To tackle this problem, Saffioti et al decomposed the controller into small *units of control* or *control schemas*. Each of these 'units' describes a specific motor skill which results in a certain type of low-level physical movement such as panning a camera, moving a limb forward, etc. Their idea is now that different commands can, to a greater or lesser degree, generate the same type of movement. Depending on the current internal state of the robot and the environmental state, some commands (or movements) have greater possibility to reach the robot's current goals. A control schema can then be seen as a preference or *desirability function* (Saffioti et al., 1995) over the space of all possible commands. These preferences or desirability functions are then represented by predicates in a multi-valued (fuzzy) logic.

Take for example the bodily movement *'turn left'*, whose execution in a specific environment leads to the state '*person is present in front of the robot* ', thereby satisfying the goal of '*turn your face to the person*'. In the specific environment under the current goal conditions, the *preference* for this movement will be higher and in this way a behaviour is described in terms of the robot's *preferences among possible actions*.

Combining this approach with a multi-valued logic controller, behaviours can be combined in a trade-off fashion by which each behaviour expresses its preferences among the possible actions which are mathematically combined to a weighed output (for example when behaviour A desires *GO_VERY_FAST_FORWARD* and behaviour B desires *GO_FORWARD* their combined weighed output might result in *GO_FAST_FORWARD*).

## 4.2 Fuzzy Logic

In traditional logic, a statement can be either completely false (0) or completely true (1). However, when dealing with a real-life environment and human reasoning, this approach turns out to be too limited. When, for example, considering concepts like height and statements like 'the boy is tall' there is no hard (height-)limit that defines the term 'tall'. Explicit reasoning about the truth of a statement as vague as the one above is therefore difficult in the language of traditional logic.

Fuzzy logic (Zadeh, 1965) differs from traditional logics at the point that it can deal with this type of vagueness. By making use of the entire interval [0..1], the 'truthness' of a statement can be discussed in finer detail. In contrast to *crisp* predicates which can only be true or false, *fuzzy predicates* can be partially true and thereby can represent uncertainty and approximation.

Continuing with the example of height, in fuzzy logic it is possible to define a number of *fuzzy sets* that range on values of the category height, in which the fuzzy set *tall* is one of a number of overlapping fuzzy sets. This will be described in more detail in the following section.

### 4.2.1 Fuzzy Sets

A fuzzy set is an extension of the classical set theory. First, consider the definition of a crisp set X which is defined by the following Boolean membership function μ:

$$\mu(x): \begin{cases} \textit{true}, & \text{if } x \subset X \\ \textit{false}, & \text{if otherwise} \end{cases}$$

where x is an element within a given universe U.
In contrast, the membership function of a fuzzy set is but a continuous one, returning real values:

$$\mu(x): \begin{cases} 0, & \text{if } x \not\subset X \\ 1, & \text{if } x \subset X \\ [0..1], & \text{if x has a } \textit{partial} \text{ belonging to X.} \end{cases}$$

Therefore, the fuzzy function returns the 'degree of belonging' of an element in a set. (Bonarini, 1996).

Different fuzzy sets for age groups are shown in figure 4.1. In this case, a person of 20 years old has a .3 degree of belonging to the group *child*, .75 degree of belonging to the group *adult* and no belonging (0 degree) to the group *elderly*.

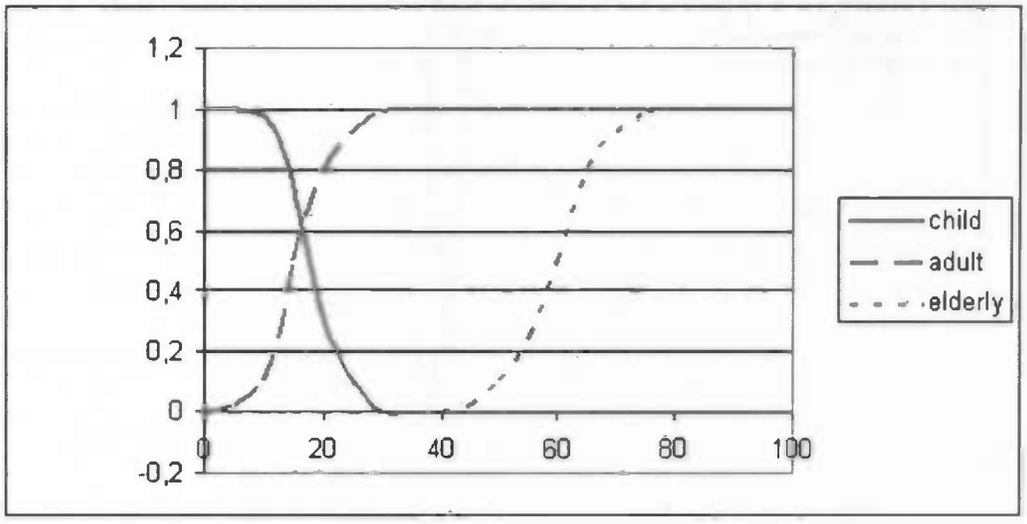



*Figure 4.1 Fuzzy sets describing age groups*

### 4.2.2 Fuzzy Operators

A number of fuzzy operators, which are extensions of the traditional logic operators *and*, *or* and *not*, can be used to build logical statements. There are a number of approaches to interpret such extensions (Dubois, Prade, 1980), one common interpretation for *and* is to take the minimum value of the membership functions that state the truth of a proposition, for *or* to take the maximum and the value for *not* can be obtained by subtracting the value of the statement from 1. An example of the fuzzy logical operators is visualized in figure 4.2.

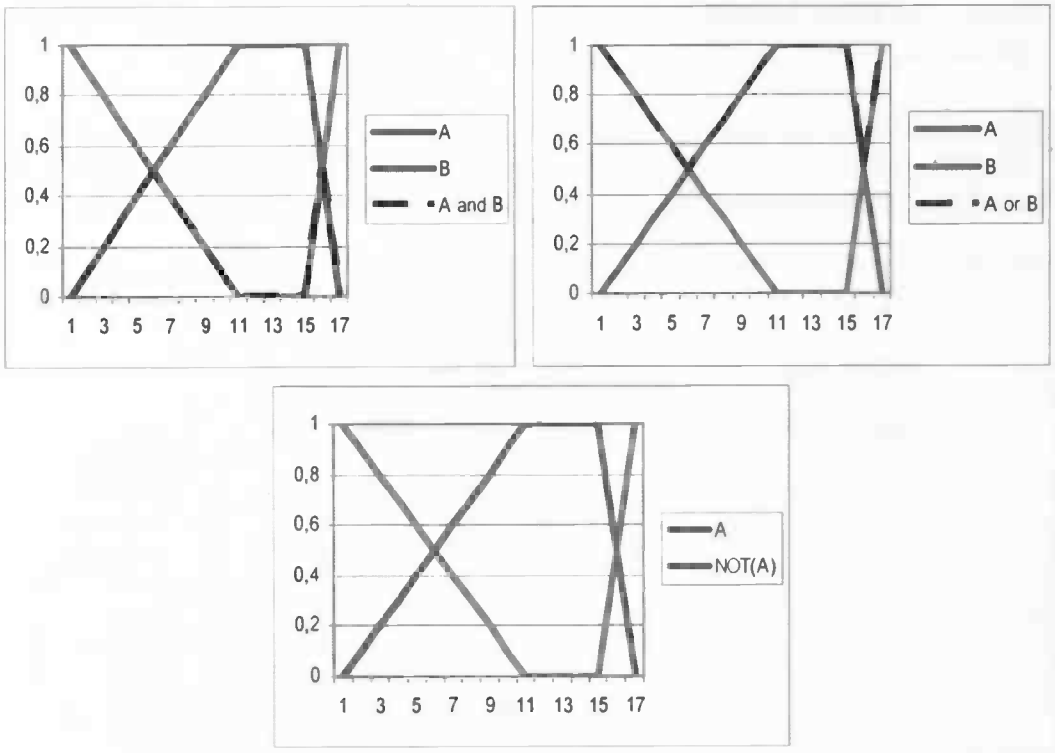



*Figure 4.2 The fuzzy logical operators and, or and not*

### 4.2.3 Fuzzy Rules

Reasoning with these fuzzy sets and operators happens by use of *fuzzy rules*. A fuzzy rule is a rule of the type

IF *premise* – THEN *actions*

The premise has the structure *<fuzzy_variable> IS <label_of_fuzzy_set>* or a combination of these, combined by fuzzy logical operators to form one statement. The action part has the same structure.

So, let's consider the fuzzy rules

IF *(X is NN) and (Y is NE)* THEN *(O is MR)*
IF *(X is NE) and (Y is NN)* THEN *(O is MS)*.

Here X and Y are the input values, O is the output. NN, NE, MR and MS are fuzzy numbers.

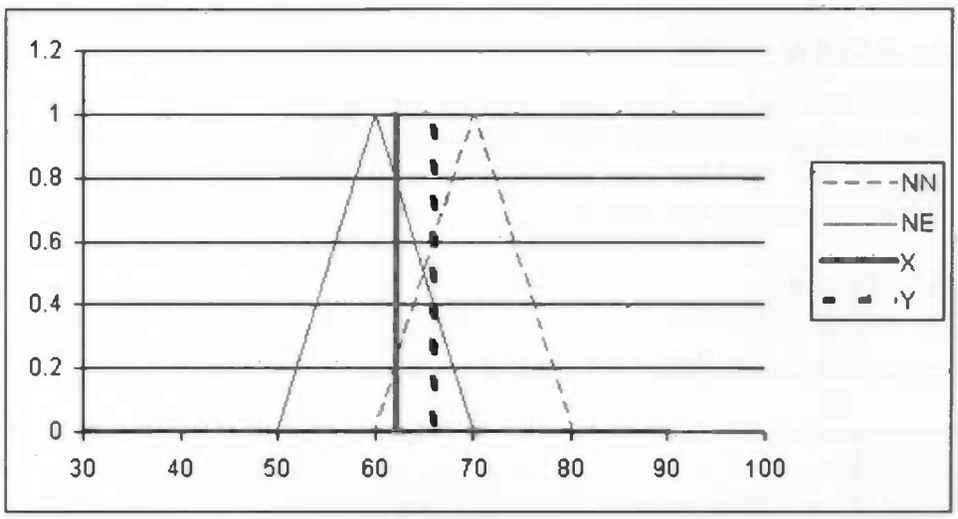



*Figure 4.3 Fuzzy numbers NE and NN with input values X and Y*

If the rule gets as crisp input

X = 62
Y = 68

then the rule will be interpreted as follows:

IF *(X is NN) AND (Y is NE)* THEN *(O is MR)*
0.2 0.4
0.2 0.2

IF *(X is NE) AND (Y is NN)* THEN *(O is MS)*
0.8 0.6
0.6 0.6

Since mapping X and Y onto the fuzzy sets leads to the values (X is NN) = 0.2, (X is NE) = 0.8, (Y is NE) = 0.4 and (Y is NN) = 0.6.
Having obtained these values and inserting them in the first rule, the *and* operator generates a result of 0.2. Therefore, the output *O is MR* that is implicated by this rule is weighed by 0.2.
Similarly for the second rule, the output *O is MS* is weighed by 0.6. The total final output will be the weighted sum of these two values of *O*.
The values of MR and MS are not shown in this example, but these indicate values within another fuzzy set. See figure 4.3 for a graphical explanation of the fuzzy sets and mapping of the input on these sets.

## 4.3 Mr. BRIAN

A practical implementation that makes use of fuzzy logic is Mr. BRIAN (Bonarini et al., 2004). Mr. BRIAN (Multilevel Ruling BRIAN) is an extension for a previously build architecture BRIAN (BRIAN Reacts by Inferring ActioNs).

### 4.3.1 Overview

The general overview of BRIAN's structure is given in the figure below.

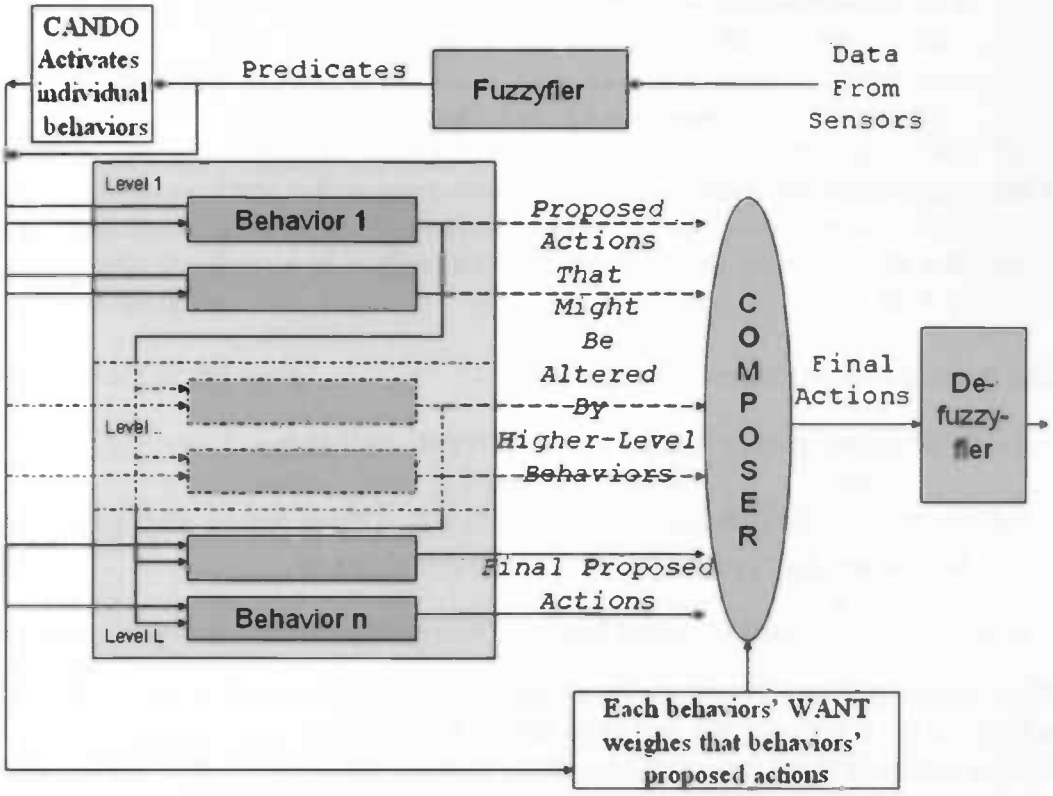



*Figure 4.4 Multilevel Ruling BRIAN*

In BRIAN, behaviours are ordered hierarchically, according to their priority. A behaviour is a set of fuzzy rules aimed at attaining one single goal. Each individual behaviour receives input in the form of fuzzy predicates. Higher-level behaviours also receive as input the proposed actions as suggested by behaviours that lie in the lower levels below them. The higher-level behaviours have the possibility to suppress the actions that are suggested by lower-level behaviours. Therefore, whenever a behaviour receives as input proposed actions that are in conflict with its current goal, it can inhibit these actions and propose its own.

### 4.3.2 Fuzzyfication

Input is as received by the agent's sensors, and needs to be submitted to the architecture in the form of a *crisp data list*. This is a list of crisp data, in which each value is of the form

<name, value, reliability>

here name refers to its unique name, value to its crisp value and reliability is a real number in the range [0..1].

As the list is entered into the architecture, the set of crisp data is mapped onto a list of *fuzzy shapes*, to obtain fuzzy values that are used in the reasoning process. These fuzzy shapes are a combination of predefined fuzzy sets that can be used to describe a certain data range. In the architecture, there are 7 sets available, as shown in figure 4.5.

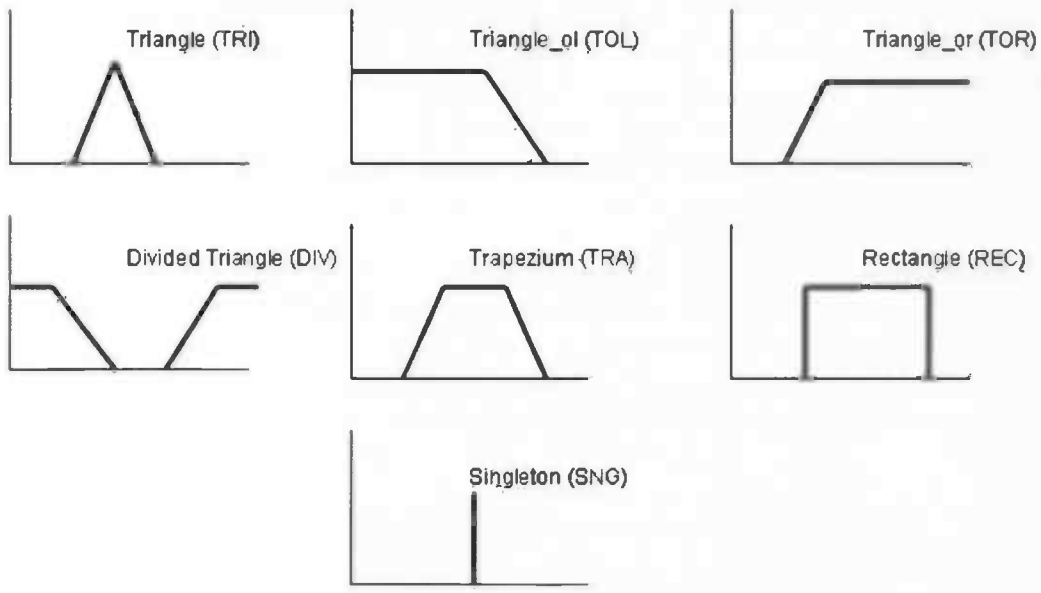



*Figure 4.5 The list of fuzzy sets available in BRIAN*

The **bold** lines indicate the value (ranging from 0 to 1) that the fuzzy set $\mu(x)$ returns for each value of the crisp data x. An example of a fuzzy set that can be constructed using the above sets is the following:

```
( DISTANCE
      (SNG (OUT_OF_RANGE -1))
      (TRA (CLOSE 1 1 20 40))
      (TRA (MEDIUM 20 40 70 90))
      (TOR (FAR 70 90))
)
```

The names DISTANCE and the names of the individual subsets (OUT_OF_RANGE, CLOSE, etc) can be defined by the implementer. SNG, TRA and TOR refer to the abbreviations that are used in figure 4.5.

After mapping the input variables on these fuzzy shapes (the *fuzzyfication*), the predicates are calculated.

### 4.3.3 Predicates

Predicates are the variables that are used within the architecture during the reasoning process. They can be interpreted as giving a truth value about aspects of the environmental context that is given as current input. Predicates are calculated by examining the degree of truthness of the input values belonging to a certain predefined fuzzy set.

Predicates are of the form

<name, value, reliability>

where name is its unique name, value its truth value, and reliability a real number in [0..1].

Take for example the predicates PersonClose and PersonMedium, which might be defined as

```
(P PersonClose)          = (D PersonDistance CLOSE )
(P PersonMedium)         = (D PersonDistance MEDIUM )
(P PersonCloseOrMedium) = (OR (P PersonClose)
                              (P PersonMedium) )
```

P indicates that this is a Predicate, D indicates that it says something about the value from (input-)Data. Therefore, PersonClose reflects the truthness of the fuzzy variable PersonDistance having the value CLOSE. PersonCloseOrMedium is evaluated by combining the previously calculated predicates PersonClose and PersonMedium.

In this example, PersonDistance is an instantiation of the previously declared DISTANCE.

Now, if the aforementioned variable PersonDistance is inserted into the architecture with value 30 and reliability 1, the following fuzzy data will be calculated:

```
<DISTANCE, CLOSE,  0.5, 1>
<DISTANCE, MEDIUM, 0.5, 1>.
```

leading to the predicates

```
<PersonClose,  0.5, 1>
<PersonMedium, 0.5, 1>.
```

These predicates are then given as input to the behaviours.

For the sake of simplicity, reliability is set to 1 in this example which indicates that the distance has been sensed with 100% certainty. Under degraded environmental situations the reliability would drop.

### 4.3.4 CANDO and WANT

To control the selection and activation of behaviours, there are two conditions that are used – the WANT and CANDO conditions. These conditions are fuzzy logical statements containing fuzzy predicates. Each behaviour has its own WANT and CANDO condition.

A behaviours *CANDO condition* is its activation condition that needs to be fulfilled in order for the behaviour to become active.
For example, when defining the CANDO condition for the rule GotoTarget, the rule with which a tour guide agent can travel to its current target position. It can only go to its target if it has not reached it yet. Therefore, a possible CANDO condition in this case would be

```
GotoTarget = ( NOT ( TargetReached ) )
```

where TargetReached is a predicates that reflects the distance of the agent from the target.

The *WANT condition* of a behaviour is a motivational condition, which defines whenever a designer wants the behaviour to be active. This condition is dependent on the agent's current context, which can be the agent's internal goal, the current environmental situation, etc.
Continuing with the example of the GotoTarget behaviour, the WANT condition could be dependent on the fact whether there are people close to the agent.
This could lead to the following WANT condition

```
GotoTarget = ( OR ( PeopleClose ) ( PeopleMedium ) )
```

OR is the fuzzy logical or-operator.

### 4.3.5 Activation of Behaviours

To activate a behaviour, the behaviour's CANDO value needs to be greater than a given threshold $\tau$, whose value can be defined beforehand. If the behaviour is activated, the actions that are proposed by it are calculated, associating each with their respective CANDO value. After a behaviour proposes its actions, its WANT condition is evaluated and the result is used to weight the proposed actions.

The final set of actions comes from the weighted average of all proposed actions. These actions are still in fuzzy form.

### 4.3.6 Defuzzyfication

Before the actions can be sent to the agents actuators, they need to be transformed back into a crisp data state. This d*efuzzyfication* happens in a way that is similar to the fuzzyfication of data, the data values are again mapped onto a number of fuzzy sets. The difference here is, that these sets only contain singleton subsets. Therefore, mapping a value onto the *defuzzyfication sets* always results in one (crisp) value. The final crisp data list is than given as BRIAN's output.

# 5. THE SIMULATOR

The simulated tour guide, the person that it is guiding and the environment in which it is operating are simulated in a 3D simulator, called MarsOde – a Multi-Agent Robot Simulator based on ODE – which was developed by the Dipartimento di Elettronica e Informazione of the Politecnico di Milano.

## *5.1 Specifications*

This simulator is a platform-independent robot simulator, in which robots and objects interact with each other according to the rules of rigid body physics. By rigid bodies is meant that the simulated bodies do not deform – they are completely stiff. Each simulated object has six degrees of freedom, two degrees about each axis (x, y and z).

Two important external libraries are used by the simulator, respectively Xith3D and ODEJava. Xith3D is a scenegraph-based 3D programming APID for Java. It allows for viewing the simulated world in 3D perspective in a window. ODEJava is a Java-wrapper around a previously developed open source physics engine *ODE* (Open Dynamics Engine). This physics engine can be used for simulating articulated rigid body dynamics. Examples of these are ground vehicles, legged creatures and moving objects in VR environments. Built-in collision detection is available.

Simulation happens in the form of sequential simulation steps. In each step, the position of all objects is updated and the collision engine calculates the effect of collisions that have taken place.

Figure 5.1 shows the main components of the simulator.

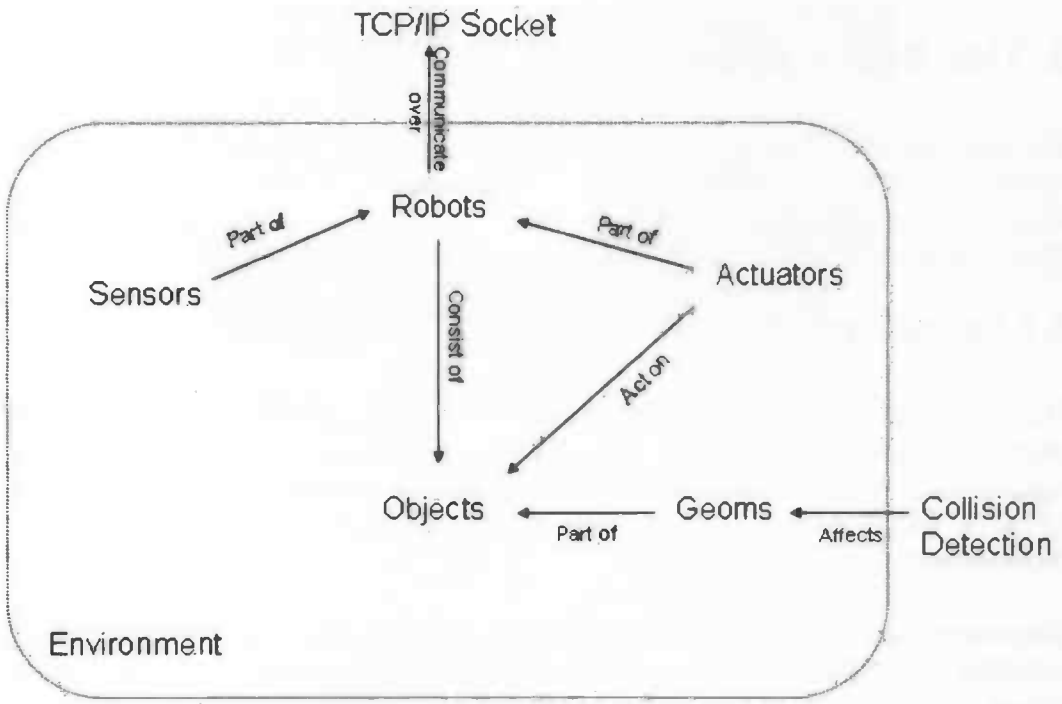



*Figure 5.1 Important components of the MARSOde simulator*

## 5.2 The Simulated Environment

Without going too much into detail about the different variables and data types that are used within the simulator, a description will be given of how the physics simulation is implemented.

All objects that make up the simulated world are parsed from an external xml file. This file lists the objects and their properties such as size, shape, weight. The XODE (XML ODE data interchange format) is used to specify the objects. Each object has an embedded *geom* for collision detection (see also the next section).

The simulated robots are parsed from separate xml files, also in XODE format. Each robot definition consists of the robot's 'bodily parts' (which are all individual objects) which can be linked by *joints* and moved by means of *activators*, more on this later. An external TCP/IP port needs to be specified through which the robot communicates with its external controller.

### 5.2.1 Bodies, Joints and Geoms

The visible, rigid objects within the simulated world are referred to as *bodies*. The rigid body types that are available to use are sphere, box and cylinder.

A body has the following properties

- mass
- shape
- position
- rotation
- angular velocity.

These properties have no specific units such as kg and meters, but are all expressed in the units of the simulator. For example, the robot can be of size 2x2x2.5 and move with a velocity of 2. Units need to be defined by the person that is running the simulation. For this simulation one unit of length is equal to 1 dm, and velocity is expressed in dm/sec.

Another type of object, however not visible, are *geoms*. A geom is a geometry object, and it is used for the detection of collisions in the simulated world. Geoms can collide with other geoms to yield one or more contact points on which then act certain forces. A geom itself does not have dynamical properties such as velocity, but only geometrical properties as shape. To use the collision detection system, geoms are associated with bodies. This is done to obtain their position and rotation from the bodies.

A geom and a body together describe all the properties of a simulated object.

Bodies can exist as single entities, or they can be connected to other bodies by means of different types of *joints*. Joints are the same as they are in real life – they connect two objects. By doing this, they pose a restriction on the bodies' position and rotation relative to each other. An example of joint is the *hinge* joint-type. A hinge connects two bodies to each other. These bodies are then subjected to certain restriction regarding their mutual distance and position. The two bodies are connected by an anchor point, to which the hinge keeps them together. Movement relative to this anchor point is only allowed as rotation around an axis on the anchor point. Figure 5.2 shows an example of the hinge type, which can be controlled by the *actuator* HingeController. More about actuators is explained in the following *actuators* section.

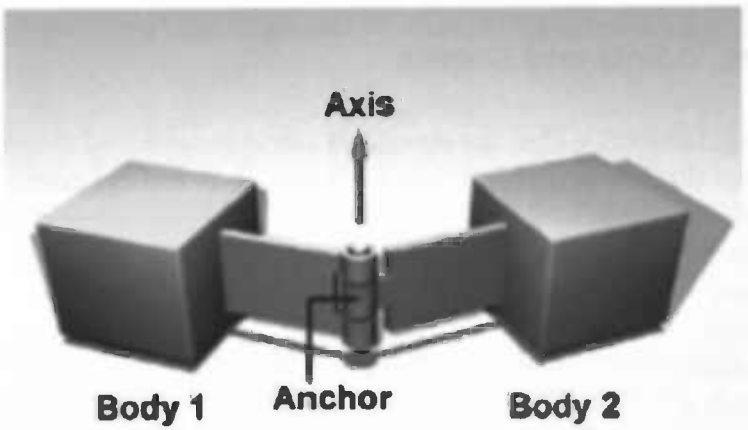



*Figure 5.2 A Hinge Joint (Smith, 2004)*

### 5.2.2 Actuators

Simulated robots can be moved by applying an activation value to their *actuators*. An actuator can be moved by changing its rotational speed, position or velocity. These are expressed in floating point values. An example of an actuator is the actuator-type `mars.actuators.HingeController` which acts on the hinge-type joint (see above). By applying a rotation around the axis of the hinge, the two connected bodies can be propelled forward together.

### 5.2.3 Sensors

As an extension to the set of sensors that is already available in the simulator, I have implemented two extra sensors. Refer to the controller protocol section for the form of data that they return.

#### *5.2.3.1 Vision*

The Vision sensor approximately simulates an omni-directional vision camera. It is an approximation since its method of object recognition does not have anything to do with parsing of images. Its actual way of sensing objects happens similar to that of a sonar – by casting out a number rays around the robot and detecting their collisions it senses the objects surrounding the robot. The name vision is taken since, for each object detected, it returns information about the object's relative distance (the contact point between the ray and the object hit by the ray), the object's relative angle to the robot and the name of its geom. Therefore similar to 'seeing' and interpreting the object.

Currently, 180 different rays are used for detection of objects. Using this and the chosen range of maximum 6 metres, the smallest object that can be detected at minimum and maximum range can be calculated as following:

The Vision sensor is positioned in the center of the robot's body, from where the rays point outwards. The body measures 40 by 45 cm. Collisions with this body are ignored.

On a distance of 20 cm, the distance between two consecutive rays is

$$(2*\pi*r)/180 = (2*\pi*0,2)/180 = 0,0698m$$

so objects with size about 7 cm or bigger can be detected.

At maximum range, this yields

$$(2*\pi*r)/180 = (2*\pi*6)/180 = 0.166m$$

There is only one limitation to the sensor, namely the fact that it can only detect one object for each ray that is cast out. Therefore, although other objects might still be visible behind an object that has been sensed (for example when they are larger in height) they will not be detected by the sensor.

#### 5.2.3.2 Position Sensor

The Position Sensor returns the agents current absolute position and its rotation in relation to the y-axis. The simulators coordinates are defined in a right-handed coordinate system. The value of x and y is expressed in diameters.

### 5.2.4 Robots

In my simulation, there are two types of robots available, the *TourGuide*- and *Person*-robot.

#### *5.2.4.1 TourGuide*

There is only one instantiation of the *TourGuide* robot present. This is the robot on which the experiments are conducted.

*Appearance*

The TourGuide robot is a non-anthropomorphic robot, which consists of a box-shaped body with three wheels attached to it – 2 to the side of its body, and one at the bottom center. The robot body measures 40 cm in length, 40 cm in width and is 15 cm high. It weighs 35 kilos and the robot has a maximum forward speed of

40 cm/sec. A picture of the robot can be seen on the screenshot at the end of this chapter.

*Sensors*

The TourGuide is equipped with a Vision sensor and a Position sensor.

*Actuators*

Locomotion of the agent can be considered as being realized by a set of two identical motors. In the simulator this is obtained by applying a rotation to the HingeController that acts on the hinge joints that connect the wheels to the robot's side to its body. Forward and backward movement is generated by giving the same activation to the side wheels, turning speed is generated by giving different activations to these wheels. The bottom wheel has a 360 degrees turning possibility and is merely used to support the robot's body.

#### 5.2.4.2 Person

Several instantiations of the *Person* robot are used to simulate a crowd of people. For the sake of simplicity a Person robot has nearly the same physical characteristics – only the colour of its robot body differs – actuators and sensors as the TourGuide robot. The two robot-types only differ in the fact that a Person robot does not have a GPS sensor (and, outside of the simulator is connected to a different instantiation of the behavioural architecture).

### 5.3 Static Objects

In my simulation, the static objects in the robot's environment are recognized as either 'obstacle' or 'door'. No further specification of their type is done. For implementational simplicity, all objects have a rectangular shape with no protruding edges and they are all equally high.

### 5.4 Targets

The TourGuide robot gives its tour by sequentially visiting a list of targets in the environment.
The set of target positions are not represented anywhere in the simulator. Instead they are a list of coordinates that is saved in the Controller connecting the simulator to the architecture. Refer to the 'Experiments' section for more details about this.

## 5.5 Controller Protocol

Each robot within the simulator is connected to an external controller. This controller will receive the data robot's sensor and it will calculate and return the activation values that are applied to the robot's activators. Communication between an agent and its controller happens over a TCP/IP socket

At first, on receiving the starting string `#BEGIN_SIMULATION#`, the simulation starts and simulation takes place in sequential steps.

A typical simulation step happens in the following sequence:

- The controller receives sensor data from the simulated agent
- The controller receives the reinforcement observations (to be used when it is connected to a learning architecture, i.e. a neural network)
- The controller sends the calculated actuator values
- The controller sends control information

For the current simulation purpose, the reinforcement information is simply ignored. The only focus is on sensor values and the calculated actuator values.

The sensor data is transmitted in the form:

```
#BEGIN_SENS#
<variable_assignment_list>
#END_SENS#
```

Actuator values are sent in the following way:

```
#BEGIN_ACT#
<variable_assignment_list>
#END_ACT#
```

Control information is sent as:

```
#BEGIN_CTRL#
<control_info_list>
#END_CTRL#
```

The *variable_assignment_list* as used for transmitting sensor- and actuator-data is a list containing strings of the following form:

```
name_variable=value
```

where

`name` is the name of the sensor or actuator

`variable` is the variable that is sensed or actuated

`value` is the value that is returned by the sensor or imposed to the actuator.

For example, the variable_assignment_list as returned by the Vision sensor looks like this:

```
Vision:Object1=11.297735,321.92807
```

In this case, the sensor has sensed the object `Object1` at a distance of about 11 units from the robot and makes an angle of about 322 degrees with the robot's front side.

The `control_info_list` consists of information which can communicate to the simulator to put the robot back to its original position or the processing time. This processing time is a floating point value (float) that the simulator dynamically takes into account to represent the time that the controller needed to process its received input. It waits ... for a number of time before applying the activation values to the activators.

At any moment, on receiving the `#END_SIMULATION#` string, the simulator and interface terminate.

## 5.6 Screenshot

The figure below shows a screenshot the simulator in which a TourGuide robot (grey body) and a Person robot (coloured body) are visible in a straight hallway with a turn to the right. The block in the centre of the hallway represents an obstacle to the robot and in theory should be avoided.

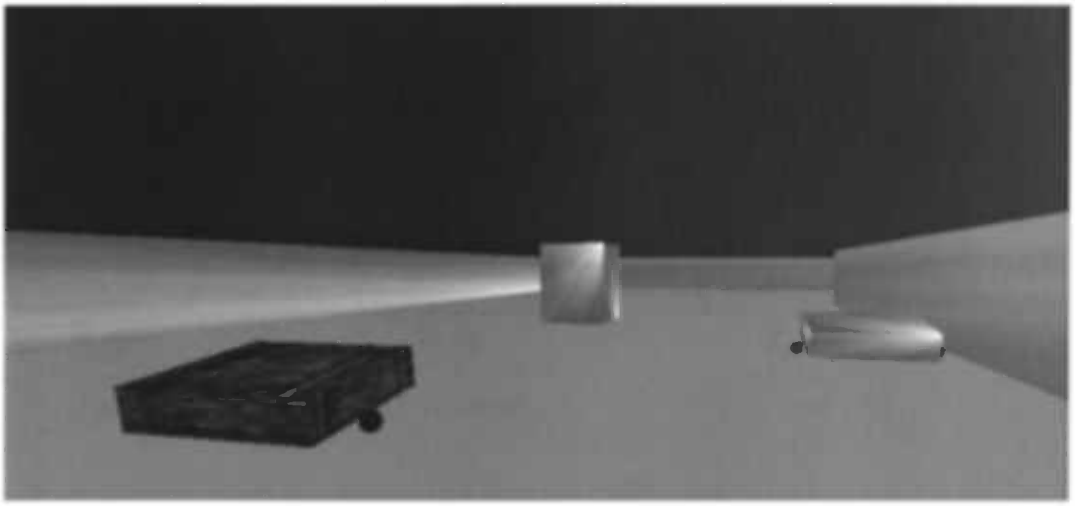


*Fig 5.3 Screenshot of the MARSOde simulator*

# 6. IMPLEMENTATION

This chapter will give a description of the actual implementation of the behaviours.

The implementation can basically be split into the Interface part – which controls communication with the simulator and calculates certain variables from the sensor input – and BRIAN, the part that describes what happens in the logical engine Mr. BRIAN. Figure 6.1 shows an overview of all components involved in the implementation and in which ways they communicate with each other.

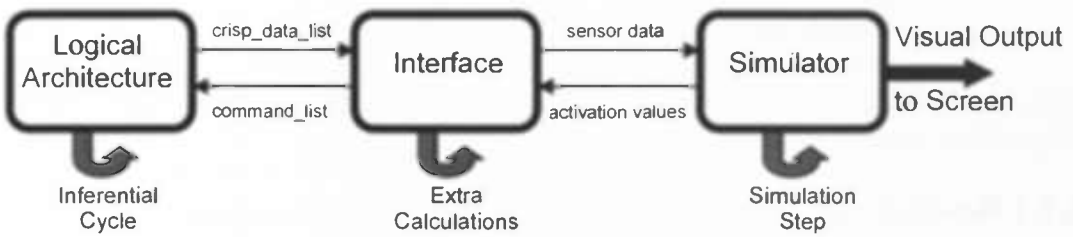



*Figure 6.1 Components Overview*

I will start out by discussing the type of data that the Interface receives and what subsequently is done with it.

## 6.1 The Interface

The sensor data that is received from the simulator is parsed into a list of variables by the interface. This data contains information about the current condition of the environment as perceived by the robot.

The variables that are used for calculation in the interface and in BRIAN are variables that

- Tell something about the robot's position and orientation
- Tell something about objects relative to the robot
- Tell something about the position of the person that is guided
- Tell something about the location of the robot's current Target

The main operations performed in the Interface with these variables are:

- Determine a 'free path' along which the robot can drive without bumping into any object, using the environment-data
- Transform person-data into the same coordinate system as the robot
- Keep track of the person's current and past position, and calculate his/her speed
- Put the environment data into BRIAN's *crisp_data_list*, run BRIAN and subsequently read activation data from its *command_list*

Some of the required data can be easily obtained from the Vision- and GPS-data and will not need further computation to obtain their values. These data are

- Robot position
- Robot orientation
- Object angle and –distance
- Person angle and –distance

The target points that the robot needs to visit are already present in the list of Target Coordinates within the Interface.

### 6.1.1 Navigation

Although it may sound trivial, navigation happens by turning the robot's front towards the current target and then driving towards it. However, the current target position is not always a fixed one. The coordinates that the robot sequentially visits to give its tour are indeed fixed, but while touring, the robot can choose between two types of targets. These targets can be the current Target Coordinate out of the Target Coordinate List or the Person. Which of the two is chosen depends on the robot's internal state and the environmental situation.

With use of the coordinates of the chosen target, the desired direction can be calculated easily using the robot's position and rotation and Pythagoras.

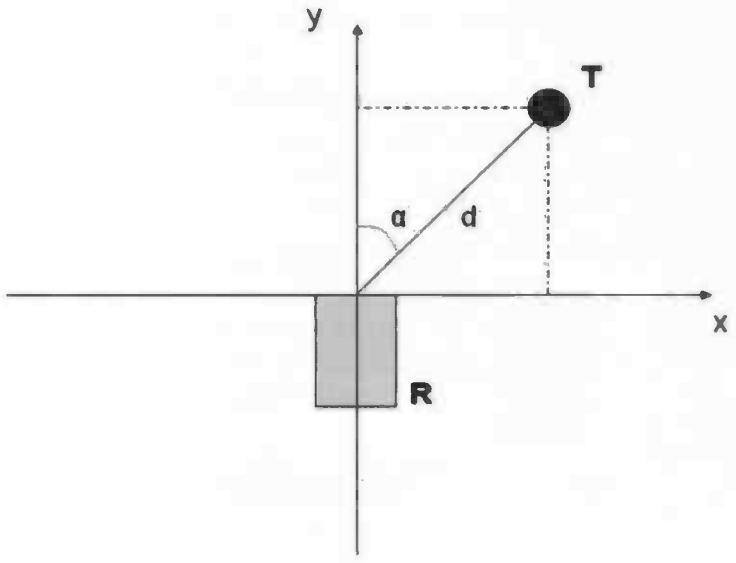



*Figure 6.2 Calculation of target coordinates*

Distance over the x- and y-axis between robot and target position is given by:

$$Distx = Rx - Tx$$
$$Disty = Ry - Ty$$

With these, the total distance d is calculated.

$$d = \sqrt{Distx^2 + Disty^2}$$

Finally, the angle α is obtained using

$$\alpha = \sin^{-1}(\frac{Distx}{d})$$

### 6.1.2 Calculating Free Passage

In case there is an obstacle between the robot and its target position it is necessary to calculate a free passage around it. Since the robot does not possess some complex path-planning algorithm, another approach needs to be considered. It is assumed that the environment is simple and that no extremely big obstacles will suddenly appear that block the robot's path. In this "simple" environment it will be sufficient for the robot to determine whether the direction in which it attempts to drive provides sufficient free space for its body and wheels to pass. If there is no sufficient passage space in the intended direction, then an alternative direction is calculated.

Free space is calculated by checking the available space left and right from the straight line that can be drawn from the front of the robot. Figure 6.3 shows the part to the right of the robot containing the minimum amount free space that is

required for the robot to drive straight forward. This 'free space' is checked from the robot's Vision sensor data.

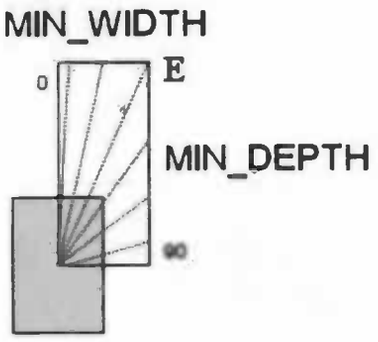



*Figure 6.3 Free path calculation*

Calculation happens as follows:

Checking of the Vision sensor rays starts at the front of the robot (relative angle 0). From that part on, for each angle α up to 90 degrees to the right it checks whether the sensor data lists objects that lie within or outside the robot's path. The following checks are made:

For α-values from 0 to phi, where phi is $\tan^{-1}(MIN\_WIDTH / MIN\_DEPTH)$

$$VisionData[\alpha] =< (MIN\_WIDTH / \cos(\alpha))$$

And for α-values from $\tan^{-1}(MIN\_WIDTH / MIN\_DEPTH)$ to 90

$$VisionData[\alpha] =< (MIN\_DEPTH / \cos(90 - \alpha))$$

The same checks are made for Vision data at 0 to 90 degrees to the left of the robot.

When insufficient free space at this angle is found, the same checks are done for the angle that lies 18 degrees to the left. If still there is insufficient space, the process continues until all 360/18=20 steps are completed. On finding sufficient free space, the angle of possible passage is returned. If, after having checked all 20 possibilities, no angle of passage is found, the function returns the fault value 1.

### 6.1.3 Calculating the Coordinates of the Person

In order to obtain the speed of the person, their (x, y) position is calculated and stored. Data that is known about the person is the distance *d* from the person to the robot and the relative angle $\alpha$ that the person makes with the front of the robot.

- Distance *d* of the person to the robot
- Relative angle $\alpha$ of the person to the robot
- Robot's current position R

Now, from the robot's point of view, we have this situation:

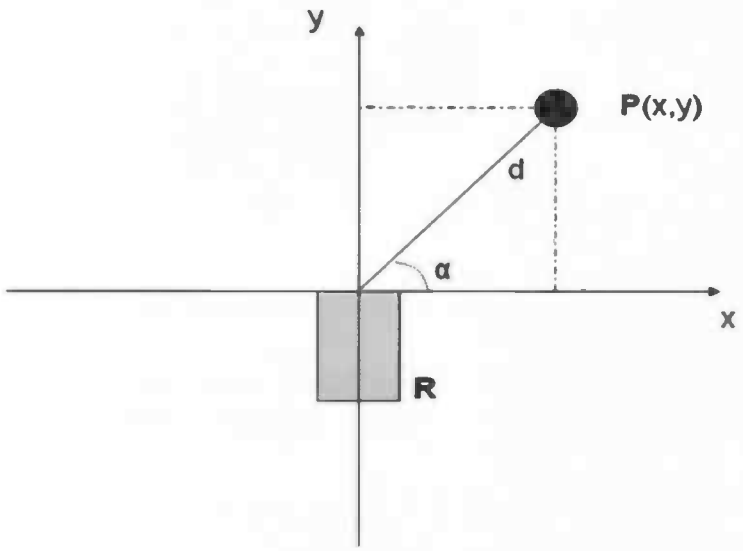



*Figure 6.4 Determining the person's position relative to the robot*

With these variables, the coordinates of point P relative to R can be expressed as

$$X_P = d * \cos(\alpha)$$
$$Y_P = d * \sin(\alpha)$$

Now we have the coordinates, as expressed from the robot's point of view and with the robot's position as point (0, 0). So, the coordinates are expressed in the *robot's* coordinate system and by rotation and translation we need to convert these into the *simulator's* coordinate system.

The robot's body might be rotated with regard to the x-axis within the simulator with an angle β, so that the person's position P needs to be rotated around the robot's position R with an angle of β degrees. This is done by

$$X'_P = X_P * \cos(\beta) - Y_P * \sin(\beta)$$
$$Y'_P = X_P * \sin(\beta) + Y_P * \cos(\beta)$$

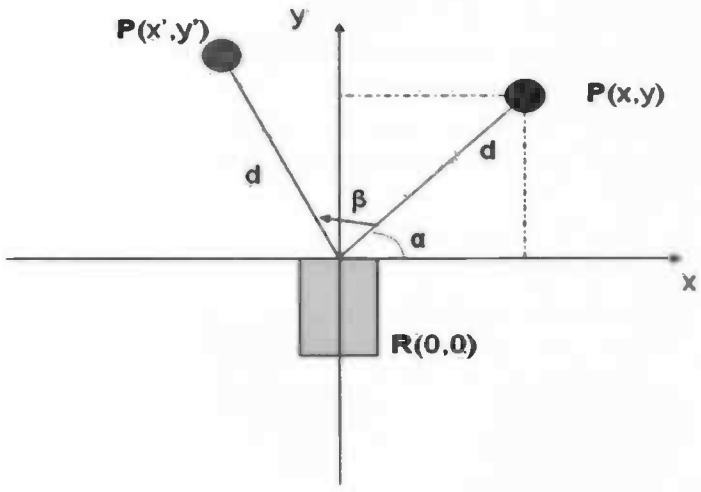



*Figure 6.5 Rotating the person within the robot's coordinate system*

By adding the robot's x- and y-coordinates from this result, the person's coordinates within the simulators coordinate system are obtained.

$$X_{FINAL} = X'_P + X_R$$
$$Y_{FINAL} = Y'_P + Y_R$$

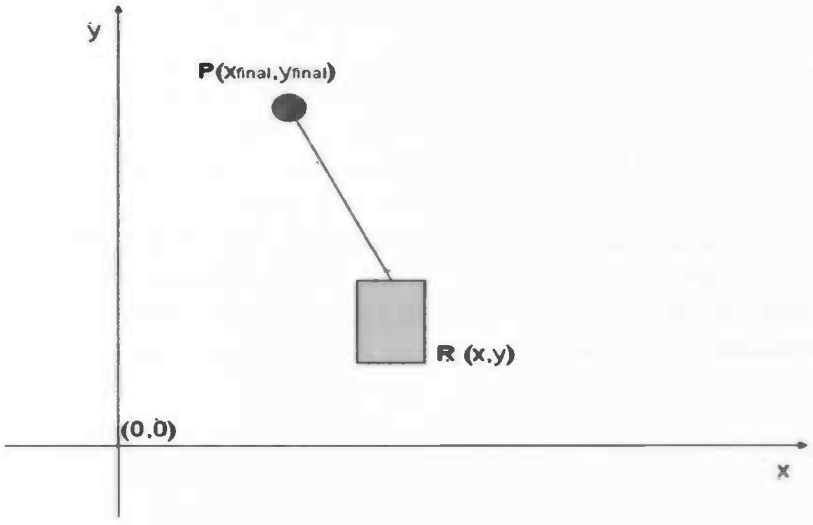



*Fig 6.6 The person's coordinates within the simulator's coordinate system*

### 6.1.4 Calculate the Person's speed

Determining how fast a person is moving is done by calculating the distance $d$ that the person has travelled over a certain period of time $t$ and dividing this by $t$ ( $v_P = d / t$ ). The distance $d$ is calculated by determining the difference between the current position of the person and his/her position $n$ cycles ago. This value is logged into a list that keeps track of the person's current and past speed.

Since the person might have changed its speed in the elapsed cycles, an estimation of the average speed of the person is obtained by taking the average of its speed over 25 cycles.

## 6.2 BRIAN

After these calculations, it is up to the logical engine to formulate the proposed actions that fulfil the current goals. I will describe all the behaviours that are implemented and how they are related to each other. Furthermore, for each of the behaviours I will give a brief overview about their WANT- and CANDO-conditions and the predicates that make up their fuzzy logical expressions.

### 6.2.1 Behavioural Hierarchy

The complete set of behaviours and their hierarchy is displayed in the figure below.

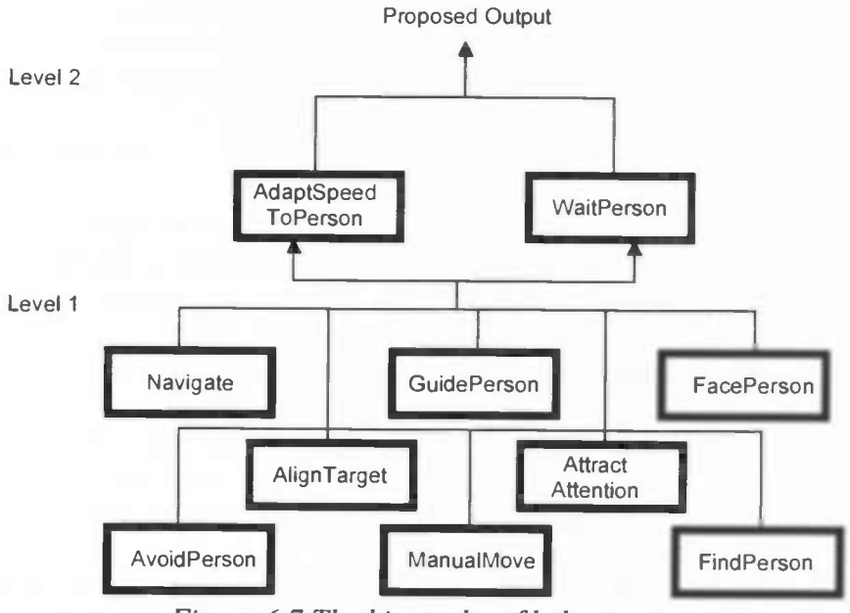



*Figure 6.7 The hierarchy of behaviours*

As can be seen in figure 6.7, the behaviours have been split up into two levels only. Since most of behavioural control is implemented by making use of flags that display the robot's internal state (explained later on this chapter about WANT and CANDO) there was only need of two higher-level behaviours that are able of modifying or deleting the proposed forward speed.

### 6.2.2 Behaviour Description

In this section, all individual behaviours from the hierarchy will be listed together with a description of their goals, input and output and how they attempts to fulfil these goals.

The lower-level behaviours will be described first, since the higher-level behaviours receive as input the combined proposed output from these behaviours. The conditions by which the behaviours are activated will be given later on, in the WANT/CANDO section.

- Navigate

This behaviour aims at navigating the robot to its current target position, taking into account the angle at which it has free passage.

Goal: Driving to the current target position
Input: The suggested angle of passage and the distance to the target.
Output: Rotational and Forward Speed.

- AvoidPerson

The task of the AvoidPerson behaviour is to make sure that the robot does not bump into the person that it is guiding. The Navigate behaviour already makes sure that the robot only chooses a free direction to drive to and basically already avoids driving into the person. However, this is only accomplished on very short range. The AvoidPerson behaviour works on a slightly larger range and provides extra safety.

Goal: Avoid driving into the person
Input: Distance and angle to the person
Output: Rotational and Forward Speed.

Combining the Navigate and AvoidPerson behaviours allows for a relatively smooth passage through narrow hallways and a safe navigation around the person.

- AdaptSpeedToPerson

This is one of the behaviours that serve to maintain the interest from the person in the robot. While the robot is touring (that is, while it is not performing other tasks such as attracting attention), it attempts to stay within proper reach of the person. This is done by adapting its speed to the speed in which the person is walking. By doing this, the robot gives an impression of being aware of the person – one of the aspects that were said to be of great importance for a tour guide robot (see the chapter about service robots). Also, by adapting its own speed to that of the person, it will reduce the number of occasions in which the person is too far away or even lost from sight.

Goal: Keep the attention from the person by adapting speed to the person's speed
Input: Combined proposed input from the lower-level behaviours
Forward speed of the person
Output: Forward speed of the motors

- AlignTarget

When a hallway or room provides sufficient space for the robot and the person to have a significant distance from each other, the robot attempts to drive somewhere in the straight line that can be drawn between the person and the target.
Goal: Keep the attention of the person by driving in a proper position between the target and the person
Input: Distance to the person, angle of the person
Distance to the target position, angle of the target position
Output: Rotational speed of the motors

- WaitPerson

Whenever a person is too far behind the robot, the robot does not immediately initiate its attractive behaviour. Instead, it first waits for a person for a given amount of time and then, if nothing has changed in the situation, enables attractive behaviours.
Goal: Stand still at the spot to wait for the person that is lagging behind
Input: Distance to the person
Output: Deletion of forward speeds that are proposed by lower-level behaviours
The flag *Waiting* to indicate that this behaviours is now active

There are three different ways in which the robot can attract the attention of the person. From the way in which they are implemented it becomes immediately clear that the behaviour is not part of the navigation of the robot, but that it attempts to convey something else with it. This is either because of the rigid movements that make up the behaviours or the fact that the behaviours are 'directed' at the person. This has been done by pointing the front of the robot towards the person while it exhibits these behaviours.
In the behaviour hierarchy as shown in figure 6.2 there is only one AttractPerson behaviour visible while there are three mentioned here. This is done because the experiments have been run with only one attractive behaviour active for easier visualisation and interpretation of the results.

- AttractWagTail

The first way in which the robot attempts to attract the attention of someone that lags behind is by 'wagging its tail'. Since the robot does not possess a tail, this behaviour results in shaking its complete body in a rigid way while standing still on the spot.

Goal: Convey to the person that it should approach the robot
Input: Distance to the person, angle of the person
Output: Rotational speed of the motors
The flag *Attracting* to indicate that this behaviour is now active

- AttractLookBackDriveForward

If the person keeps lagging behind, the robot can turn around to face the person, turn to the target again, drive forward a small distance and then turn back around to face the person again. By facing the person the robot communicates that it is trying to convey something to that person. Continuing its tour and repeating the facing behaviour, it attempts to communicate its intention to get the person to follow it.

Goal: Communicate to the person that the robot wants it to follow it
Input: Distance to the person, angle to the person
Output: Forward and rotational speed of the motors

- AttractShakeBody

This type of attractive behaviour has also been used in the implementation of Nicolescu to indicate that the robot needs the attention of the person to whom it is showing the behaviour (Nicolescu, 2001). Whenever the robot loses sight of the person and finds him/her back, this behaviour is shown.

Goal: Attract the attention of the person
Input: Distance to the person, angle to the person
Output: Forward speed of the motors

- FindPerson

Losing a person should be unacceptable for a tour guide robot, since losing the person could mean that it failed to give its tour correctly. With this behaviour the change of finding back the person and successfully continuing the tour increases.

Goal: Try to find back the person that the robot was guiding by looking at the last position where the person was detected by the robot's sensors.
Input: Distance and angle to the position where the person was last seen
Output: Forward and rotational speed of the motors

- FacePerson

While waiting for a person, the robot attempts to turn its front to the person, indicating that it is the person that it is waiting for (and not some other environmental condition or inner state of the robot).

Goal: Turn the robot's front to the person
Input: Angle to the person
Output: Rotational speed of the motors

- GuidePerson

The proposed output of GuidePerson does not contribute to the movement of the robot but consists of certain flags which in their turn make up parts of the WANT- and CANDO-conditions of some of the other behaviours.

Goal: Control the internal states of the robot
Input: Environmental conditions (person/robot data), internal states of the robot
Output: Internal states of the robot

- ManualMove

This behaviour has nothing to do with the robot's autonomous guidance and has only been used to control it manually by use of a keyboard.

Goal: Control the robot with a keyboard
Input: Keyboard commands
Output: Forward and rotational speed of the motors.

### 6.2.3 Proxemics and Kinetics

So far, nothing has been mentioned about the way in which proxemics are incorporated into the design of the behaviours. As explained previously, behaviours are selected and activated according to their WANT- and CANDO-conditions. These conditions will be described in more detail in section 6.2.4 but here it suffices to say that they are mainly constructed by use of two types of variables, being the robot's current state/activity (flags) and the proxemic distance that the robot currently occupies in relation to the person that it is guiding

The proxemic distance has been divided into four sections, as already mentioned in the chapter about non-verbal communication.

0 to 18 Inches (0 – 45,72 cm) – Intimate
1.5 to 4 ft (45,72 to 121,92 cm) – Personal
4 to 12 ft (121,92 cm to 365,76) – Formal/Business
12 ft < – Public

Figure 6.8 gives an impression of the length of these distances. The black dot in the centre is the person to whom the distances are applicable.

*Fig 6.8 Proxemic distances*

Regarding the fuzzy nature of BRIAN's reasoning and the fact that proxemic distance differs between cultures I made the spaces overlap a little, resulting in the following fuzzy sets:

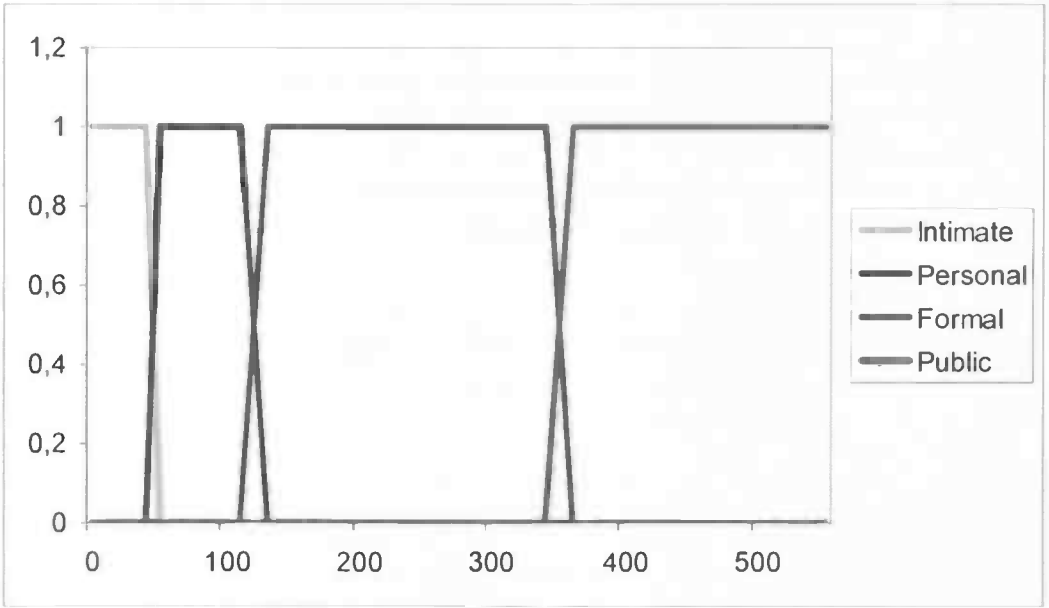



*Fig 6.9 Fuzzy sets describing proxemic distance*

The proxemic space that the robot tries to occupy during the tour will be the formal/business sphere. This has been chosen for the reason that this distance corresponds the best to the relation between a tour guide and its follower. Personal or intimate distances would be somewhat inappropriate in a public building and the public space would be too distant. Also, if the robot staid in the *public* proxemic space too frequent this could result in losing the person during the tour.

Most of the behaviours that serve to display awareness are active in the *formal* proxemic distance. In contrast, most of the behaviours that attempt to attract the attention of the person are active in the public distance or the border between the business and public space.
The intimate distance is explicitly avoided by the robot, which is accomplished by the AvoidPerson behaviour. This was not mentioned in the behaviours description which only focused on safety issues, but a side effect of this behaviour is that the robot actively steers out of the person's intimate space.

Now I will discuss kinetics, which is nothing more than the movement of the robot's body. In the implementation has been chosen to make use of rigid, fast movements to attract the attention of the person, in contrast to smooth overlapping movements as the robot moves around or when it tries to convey an intention to the person. These rigid movements are used to immediately make apparent that something has occurred and that the robot attempts to indicate that attention needs to be paid to something. Apart from this, it becomes immediately clear that these behaviours are not part of touring but should serve to indicate something.

However, there are two 'smooth' behaviours that serve to maintain the attention – being AdaptSpeed and AttractLookBack. Since they have been implemented to make the robot act in a natural kind of way and because of the fuzzy nature of BRIAN, the robot performs these behaviours smoothly. Also, AdaptSpeed would not serve much if it was adapting the robot's speed rigidly and make the robot bump its way through the tour.

### 6.2.5 WANT/CANDO

In this last section I will give an overview of the conditions that serve to active the behaviours.

Table 6.1 shows the WANT- and CANDO-condition for each behaviour as they are used in BRIAN.

| Behaviour | WANT | CANDO |
|---|---|---|
| *Level 2* | | |
| WaitPerson | ( OR<br>( P PersonFar )<br>( P PersonVeryFar ) ) | ( AND<br>( P AutoMode )<br>( P PersonPresent ) ) |
| AdaptSpeedToPerson | ( AND<br>( P TargetN )<br>( P Touring ) ) | ( AND<br>( AND<br>( P AutoMode )<br>( P Touring ) )<br>( P PersonPresent ) ) |
| *Level 1* | | |
| AvoidPerson | ( P PersonVeryClose ) | ( AND<br>( P AutoMode )<br>( P PersonPresent ) ) |
| FindPerson | ( P PersonLost ) | ( AND<br>( P AutoMode )<br>( NOT<br>( P PersonPresent ) ) |
| AttractAttention | ( P WaitingAttract ) | ( AND<br>( P AutoMode )<br>( P PersonPresent ) ) |
| FacePerson | ( NOT<br>( P PersonN )<br>) | ( AND<br>( P AutoMode )<br>( AND<br>( P PersonFar )<br>( P Waiting ) ) ) |
| Navigate | ( OR<br>( P InitTour )<br>( NOT<br>( P TargetReached )<br>) ) | ( AND<br>( P AutoMode )<br>( OR<br>( P InitTour )<br>( P Touring ) ) ) |
| AlignTarget | ( NOT<br>( P TargetAligned<br>)<br>) | ( AND<br>( P AutoMode )<br>( AND<br>( P BetweenPersonTarget)<br>( P Touring ) ) ) |
| GuidePerson | ( P AutoMode ) | ( P AutoMode ) |
| ManualMove | ( NOT<br>( P AutoMode ) ) | ( NOT ( P AutoMode ) ) |

*Table 6.1 The WANT- and CANDO-conditions for the TourGuide robot*

In table 6.1, the words in **bold** are fuzzy logical operators that act on the predicates which are indicated with a 'P'.

The AutoMode predicate is not part of the robot's behavioural control directed towards guiding a person but it is used to switch the robot from manual control to automated control.

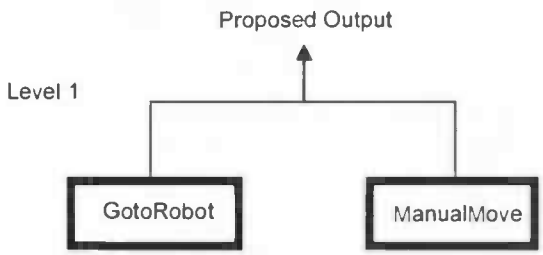
Some conditions consist of flags which are controlled by the GuidePerson behaviour. These are used to indicate which action (higher level goal) the robot is performing. The following flags are used

- Touring — When the robot is guiding the person and nothing special needs to be done.
- Waiting — When the person is lagging behind and the robot is waiting for him/her.
- Attracting — This flag is enabled when the robot is showing behaviour to attract the person's attention.
- InitTour — This flag is active when the tour is starting. It will also be activated when the robot will 'reset' the situation and drive back close to the person to get it to follow.
- PersonLost — The Person has been lost from view and needs to be found back.

In the table, the difference between the function of the WANT- and CANDO-modules can be seen by looking at their predicates. The CANDO-conditions are more defined by the overall environmental situation (i.e. if certain factors are present or absent) and the WANT-conditions are defined by more specific aspects (am I facing the person/my target?).

## 6.3 Person simulation

The simulated person that is implemented to test the effectiveness of the TourGuide's behaviours is also controlled by an instantiation of BRIAN, albeit a much simpler one. It is controlled by only two behaviours, whose corresponding WANT- and CANDO-conditions are listed in the table below. Basically the task of the person is to follow the robot when it is within range of the person (accomplished by GotoRobot), and in the meanwhile avoid obstacles as it travels towards the robot (this is already calculated in the interface by the free path calculation as described earlier).
When the robot navigates out of this person's business/formal proxemic space and enters its public space the person 'loses interest' in the robot and stops following it, thereby coming to a complete halt. Whenever the person does not see the robot (when the robot is not in front of the person), it does nothing either.




| Behaviour | WANT | CANDO |
|---|---|---|
| *Level 1* | | |
| GotoRobot | ( NOT<br>( P RobotVeryClose )<br>) | ( AND<br>( P AutoMode )<br>( P RobotInSight ) ) |
| ManualMove | ( NOT<br>( P AutoMode ) ) | ( NOT<br>( P AutoMode ) ) |

*Figure 6.10 Behaviour levels and WANT/CANDO for the Person robot*

# 7. EXPERIMENTS

In order to investigate the influence of taking into account proxemics in behaviour design, a few experiments on the simulator have been run. Data of particular interest is the proxemic space that the robot occupies during the tour, and the influence that certain behaviours (seem to) have on the overall functioning of the robot during its interaction with humans.

Since there has not been a possibility to run real tests in interaction with humans, it is hard to make hard conclusions. Therefore this chapter will have a more descriptive approach on the implementation.

## *7.1 Setup*

In order to get an idea about the functioning of the robot and influence of certain behaviours on its overall function a few data have been collected during some test-runs on the simulator.

The main focus in this section is on acquiring data about the robot's occupation of the proper proxemic space given the situation. Deviations from these can occur because of changes in the 'normal tour' situation such as when the person is lagging too far behind. The tour will be given by visiting four targets for which it will need to traverse straight hallways and drive around four corners (figure 7.1). The black dots in the corners are the target positions.

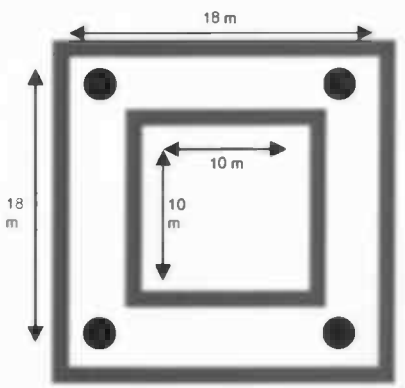



*Figure 7.1 The robot's environment*

During the second experiment the person stands still in the centre of one of the hallways doing nothing and the robot will attempt to continue its tour. While the robot is moving the environmental state will change due to its movements thereby affecting the activation conditions for the robot's behaviours. Using this 'static' situation we will obtain a good overview of the gradient change between behaviours.

## 7.2 Results

### 7.2.1 Overall

The first data sets show the robot's occupation of proxemic space for different speeds of the person that was following. The letters in the legend indicate whether the robot was touring while Adapting speed or Without Adapting speed. FF, F and SF stand for Fast Forward, Forward and Slow Forward and these represent the speed in which the person was walking.

A side note about the calculation of the person's speed needs to be made here. Since data about the person's position turned out to be rather variable it was hard to properly determine its velocity. Since extrapolation of previous data or other methods were beyond the scope of this thesis, another approach has been taken. Before running the experiment, the robot was given the person's average speed. Now, if the calculation of the speed turned out to be above (PersonSpeed + 0.25) than this value was reduced to maximally PersonSpeed + 0.25. Any value lower than PersonSpeed + 0.25 was used as is, without correction. However, these checks and calculations were only done when the AdaptSpeed behaviour was active.

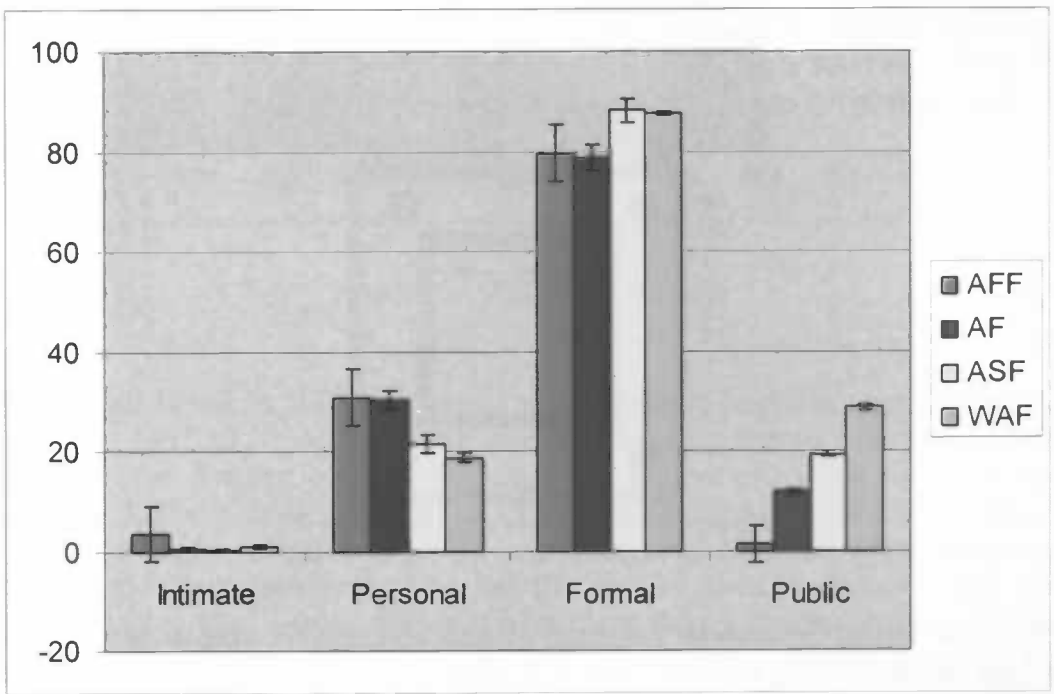



*Figure 7.2 Proxemic space occupation*

The sum of each individual line in figure 7.2 adds up to *above* 100%, however this can be explained by the fuzzy definition of the proxemic spaces. Because of

their fuzziness the spaces sometimes have partial overlapping and therefore on some occasions one data point is mapped onto two proxemic spaces.

Since we assumed that the formal proxemic space was the suitable distance in which the robot needed to be touring it seems that the robot was better capable of staying in proper range of persons moving at lower speeds. However, when regarding the amount of time that the robot spent in the public proxemic space, the situation seems reversed. Here it seems that at lower speeds of the person, the robot was more often present in the public range. This is not desirable since it means that the distance between the robot and the person was too big.

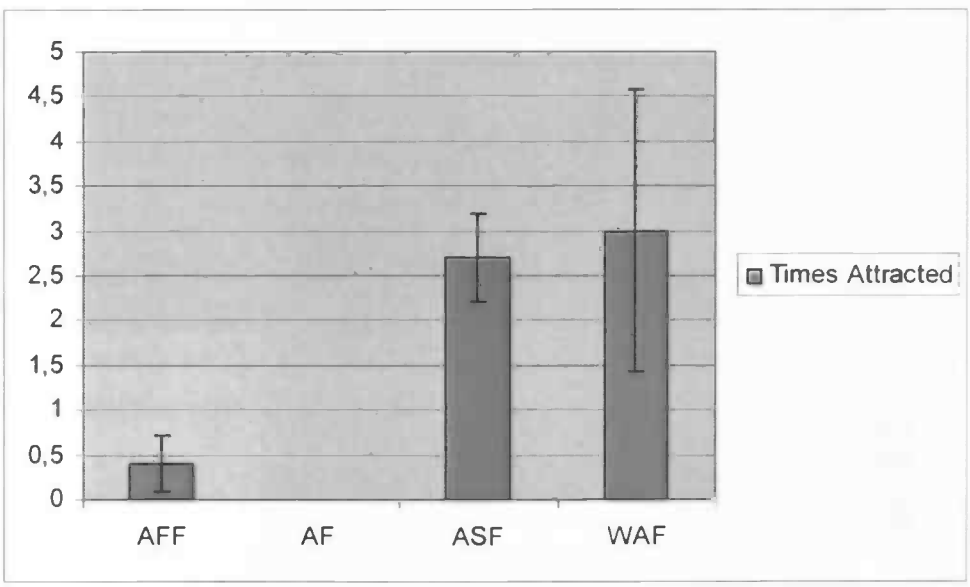



*Figure 7.3 Average times of attractive behaviour shown*

Now we take a look at the amount of occasions in which the robot needed to show attractive behaviour. During the experiments only one attractive behaviour was used and this one became active when the robot entered the public proxemic distance. In figure 7.3, it can be seen that adapting speed to that of the person did indeed have significant effect on the amount of corrective (attractive) actions that the robot needed to perform. Also, the robot seemed to have more difficulty of staying within proper reach of a slowly moving person.

From these data, it would not be fair to judge that the robot per se functioned better if the person is walking faster, it is only an indication that such situations ends up fewer times in incorrect situations. Also, whenever the robot had reached a target it needed to turn towards its next target. In the meanwhile it did not move forward so the person had a change of 'catching up' with the robot. Therefore the faster the person was moving, the closer it might have managed to come to the robot while the robot was turning.

### 7.2.2 Proxemic Space Occupation

To conclude this chapter I would like to give an indication of the proxemic space that the robot was occupying while it was performing certain behaviours. For this we need to take a look at the WANT and CANDO conditions that enable the behaviours.

#### *7.2.2.1 Flags*

Since these conditions are partially made up of the flags that give an indication about what the robot is doing, it can be helpful to first look at the activation of these flags at certain points during the tour.

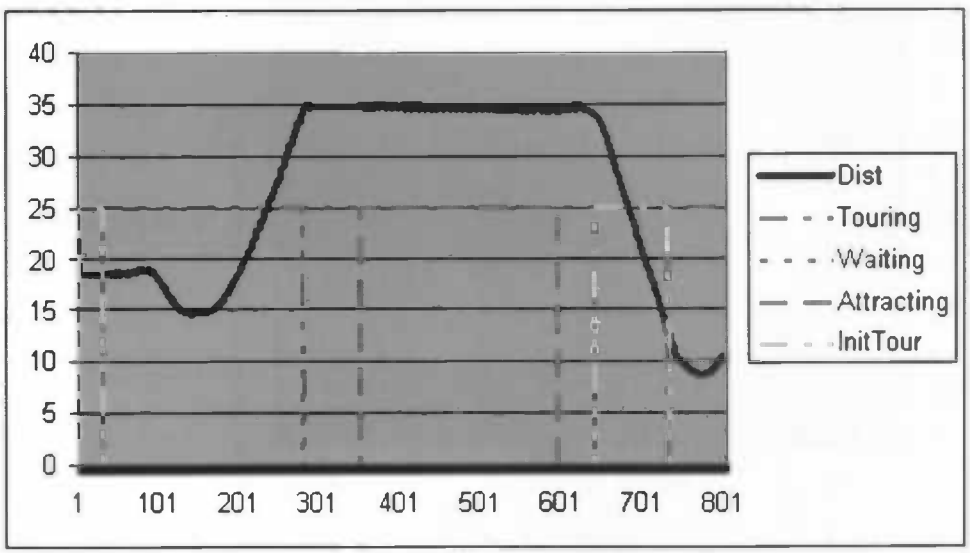



*Figure 7.4 Distance and Flags*

At the start (cycle 0), the robot stands in the person's formal proxemic distance. The flag InitTour is active which enables the GotoPerson behaviour which decreases the distance between the robot and the person. When the robot has gotten sufficiently close, the tour can start (Touring, approx cycle 25). It turns to its current target – which can not be seen in this picture – and starts navigating towards it. When it crosses the boundary between the formal proxemic sphere and the public one, the Touring flag is disabled and the robot starts Waiting for the person to come closer (approx. cycle 250). After waiting a while in which the environmental situation has not changed, the Attracting behaviour becomes active (approx. cycle 350). The robot shows its attractive behaviour for some time. If in the meanwhile still nothing has changed in the situation, the robot needs to actively 'reset' the situation by re-enabling the InitTour (cycle 630) flag and everything starts over again.

### 7.1.2.2 WANT/CANDO conditions

Displaying the entire set of behaviours in the same type of graph as has been done for the flags would create too much clutter, therefore I have chosen to only display the behaviours which will change value during the simulation. Behaviours such as Navigate and GuidePerson will be active during the entire tour, no matter the environmental condition, so they will be left out of the picture.

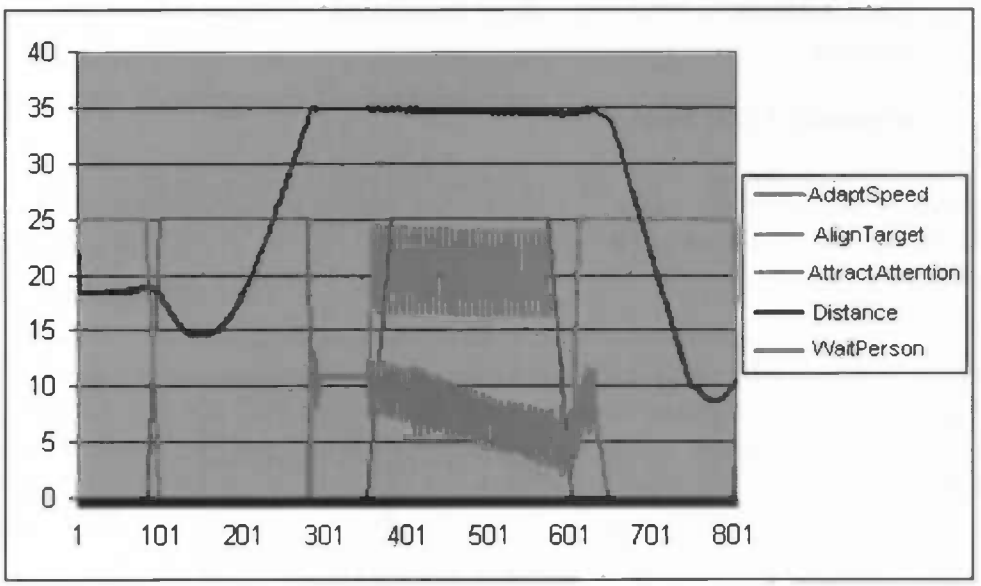



*Figure 7.5 Distance and WANT-conditions*

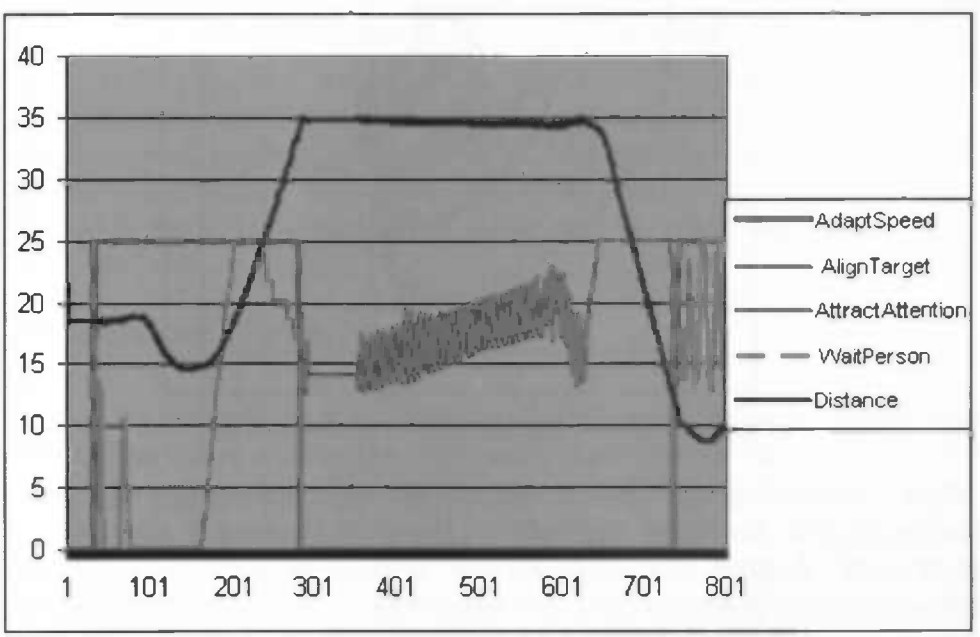



*Figure 7.6 Distance and CANDO-conditions*

The values of the behaviours have been scaled differently for a proper display in one graph so that they do not interfere with the plot of the distance. Instead of ranging from 0 to 1, their values in the figure lie between 0 and 25. The distance data has been plotted as their real values.

In these figures, the effect of the flags on the behaviours is seen easily. AdaptSpeed for example is only active when the Touring flag is set. Also, note the linear increase/decrease in Wait and Attract conditions as the time that the robot is waiting increases.

### 7.2.3 Complete Tour Plot

To conclude this chapter, figure 7.7 shows the person's position relative to the robot during an entire run. The data is collected in the AdaptSpeedForward situation.

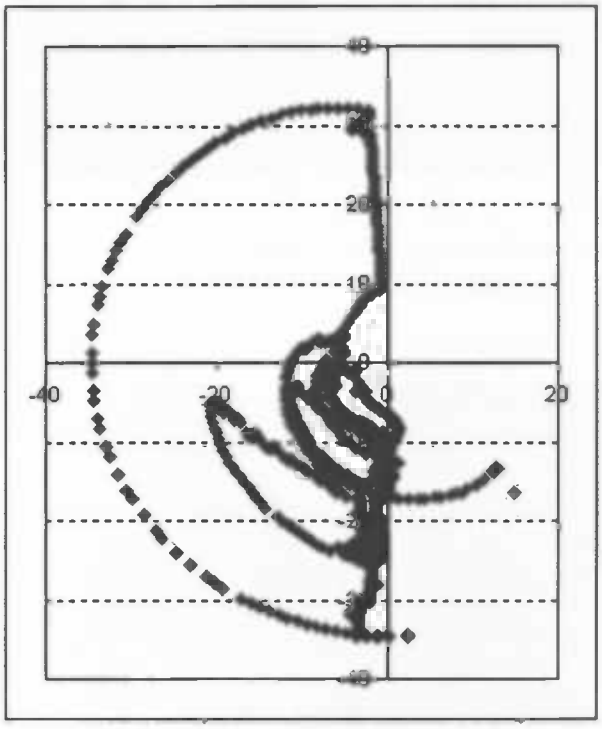



*Fig 7.7 Position of the person relative to the robot*

The robot's position in the figure is at (0,0) and the dots indicate the person's position relative to the robot. Each dot represents the person's position at one observed cycle. Positive Y coordinates are in front of the robot, negative Y coordinates indicate position behind the robot. The large curves within the figure have been caused when the robot turned its body. For example the curve at the left part of the figure happened when the robot made a 180 degrees turn. Furthermore it can be seen that most of the time the person was present at 180 degrees *behind* the robot, at a distance of 10 to 20 meters.

# 8. Conclusion

Although it is hard to generalize from findings obtained in a simulator in which also the behaviour of the people is narrowed down to a computer simulation, there are still a few things that can be concluded from this project. This chapter will be a combination of describing experiences with the architecture, the method of communication and discuss some of the findings.

## *8.1 Fuzzy Design of Behaviours*

First I will elaborate about the experiences with the fuzzy architecture and the advantages and difficulties that were encountered.

### 8.1.1 Fuzzy Control

Controlling a robot in a dynamic environment calls for a flexible controller. Fuzzy logic allows for this flexibility. With fuzzy control predicates act on a range of values instead of just a binary true/false value, resulting in a flexible implementation. In the calculation of output, the weighted sum of all proposed outputs is taken, which allows for a smooth merging of behaviours.

Difficult situations arose when behaviours cancelled each others output out, leading to a motor output that was nearly zero. This is a problem that inevitably arises when two concurrent behaviours act in opposite directions given a certain situation. The problem could be tackled by either redesigning the behaviours (tweaking their proposed output, given the current situation) or editing their activating conditions. It is a drawback of this approach, but if sufficient thought is put into the overall behaviour design this can be easily overcome. Also, when new behaviours were added that were active on these 'critic' moments, they contributed to the robot's output and made sure the combined output remained different from zero.

### 8.1.2 Behaviour hierarchy

Organizing the behaviours in a hierarchy turned out to be quite useful. Higher level behaviours that receive the proposed output from lower-level behaviours have more control in this way. By having the ability to consider the proposed output from this behaviours before it is outputted, combined with the ability to delete any unwanted output from these lower-levels, these behaviours have greater control over the robot's overall actions. Putting critical behaviours such as obstacle avoidance high in the hierarchy can create an extra layer of safety - if lower-level output conflicts too much with these critical behaviours, it can just be ignored and deleted.

## *8.2 Communication without words*

Investigating the non-verbal aspects of robot behaviour was interesting. It was surprising to discover that with simple 'tricks' such as space occupation, the robot could appeal to humans expectations and intuition and could convey certain intentions with it. Regarding this, non-verbal communication in robots can be a good method for interaction with humans. In the case of the implementations as developed in this project, future experiments that could be conducted with the help of human subjects could give further results.

### 8.2.1 Keep it simple

When 'narrowing down' an implementation to an autonomous entity that does have little in common with humans it is a good idea to keep the robot's behaviours simple to a certain degree. Humans should not be left guessing what is 'inside the box' when the robot's behaviour seemingly becomes too complex or vague/ambiguous for a straightforward implementation.

Seeing that the morphology for the robot in this project only allows for 4 degrees of freedom (forward/backward/left/right) for the robot's only body part (ignoring the wheels), the robot's behaviours remain inevitably simple. One could think of a tour guiding robot that communicates more intricately with someone by solely using non-verbal behaviour, however this robot would at least requires extra body parts such as hands to be more expressive.

In the case of this project, using the robot's body to alert people about undesired situations has been, at least in theory, sufficient. As also proven by Nicolescu et al. for one of the implemented behaviours (Nicolescu et al., 2001), the straightforward behaviours are sufficient to notify people that there is something in the current situation that is not right and that the robot attempts to convey that it wants to change this. Therefore, simplicity in morphology and behaviour did not pose too great a restriction on expressiveness and interaction.

# 9. References


Arkin R., **Behaviour-Based Robotics.** MIT Press, Cambridge, 1998.

Arkin R., Balch T., **Communication in Reactive Multiagent Robotic Systems,** *Autonomous Robots*, 1, 1-25, 1994.

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

Bonarini A., Invernizzi G., Halva Labella T., Matteucci M., **An architecture to coordinate fuzzy behaviours to control an autonomous robot**. *Fuzzy Sets and Systems*, 134: 101–115, 2003.

Bonarini A., **Fuzzy modeling and reasoning: introduction and applications**. *Cahiers du Centre Européen de Géodinamique et de Séismologie*. Luxembourg. 12, 57-66, 1996.

Brooks R.A., **A robust layered control system for a mobile robot**. *IEEE J. Robot Automation*, 2(1): 14–23, 1986.

Brooks R.A., **Intelligence without representation**. *Artificial Intelligence* 47: 139-159, 1987.

Braitenberg V., *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA, 1984.

Breazeal C., **Towards sociable robots.** *Special issue on Socially Interactive Robots, Robotics and Autonomous Systems*, 42 (3-4), 2002.

Bryant D., **The Uncanny Valley,** http://www.arclight.net/~pdb/nonfiction/uncanny-valley.html, 2005.

Burgard W., Cremers A.B., Fox D., Hähnel D., Lakemeyer G., Schulz D., Steiner W., and Thrun S., **Experiences with an Interactive Museum Tour-Guide Robot**. *Tech. report CMU-CS-98-139*, Computer Science Department, Carnegie Mellon University, 1998.

Castrillón Santana M., Cabrera Gámez J., Hernández Sosa D., Domínguez Brito A. C., Lorenzo Navarro J., Isern González J., Guerra Artal C., Pérez Pérez I., Falcón Martel A., Hernández Tejera M., et al., **Eldi's Activities in a Museum,** 2001.

Dautenhahn K., **Design Spaces and Niche Spaces of Believable Social Robots.** *Proc. IEEE ROMAN 2002*, 25-27 September, Berlin, Germany, pp. 192-197, IEEE Press.

Dourish P., **Seeking a foundation for context-aware computing**. *Human Computer Interaction*, 16, 2-3, 2001.

Dubois D., Prade H., *Fuzzy sets and systems: theory and applications*. Academic Press, NY, London, 1980.

Hall E.T., *The Hidden Dimension*. Garden City, N.Y.: Doubleday, 1966.

Huttenrauch H., Eklundh K. S., **Investigating socially interactive robots that give the right cues and make their presence felt.** *Proceedings of the CHI 2004 Workshop on Shaping Human-Robot Interaction*, 2004.

Jantzen J., **Tutorial On Fuzzy Logic,** http://www.iau.dtu.dk/~jj/pubs/logic.pdf, 1999.

Jung D, Zelinsky A., **Grounded Symbolic Communication between Heterogeneous Cooperating Robots.** *Autonomous Robots journal, special issue on Heterogeneous Multi-robot Systems*, Kluwer Academic, 269-292, 2000.

McAleer P., Mazzarino B., Volpe G., Camurri A., Smith K., Paterson H., Pollick F.E., **Perceiving Animacy and Arousal in Transformed Displays of Human Interaction.** *Proceedings for The 2nd International Symposium on Measurement, Analysis and Modelling of Human Functions and The 1st Medditeranean Conference of Measurement, Genova, Italy*, 67-71, 2004.

Michaud F., **Social intelligence and robotics**. *Proceedings from the 2000 AAAI Fall Symposium - Social Intelligent Agents: The Human in the loop*, North Falmouth, Massachusetts, USA, 127-130, 2000.

Mori M., *The Buddha in the Robot: A Robot Engineer's Thoughts on Science and Religion*. Kosei 1981.

Nicolescu M., Mataric M. J., **Learning and interacting in human-robot domains**. *IEEE Transactions on systems, Man Cybernetics, special issue on Socially Intelligent Agents -- The Human In The Loop*, 2001

Nourbakhsh I., Kunz C., Willeke T., **The Mobot Museum Robot Installations: A Five Year Experiment**. *Proceedings of IROS 2003*, Las Vegas (in print).

Nourbakhsh I., Fong T., Dautenhahn K.. **A Survey of Socially Interactive Robots**. *Robotics and Autonomous Systems*, 42: 143–166, 2003.

Saffioti A., Konolige K., Ruspini E H., **A multivalued logic approach to integrating planning and control.** *Artificial Intelligence*, 76(1-2):481-526, July 1995.

Schulte J., Rosenberg C. R., Thrun S., **Spontaneous, Short-Term Interaction with Mobile Robots.** *ICRA 1999*: 658-663.

Shannon C. E., **A mathematical theory of communication**. *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948.

Smith R., **Open Dynamics Engine User Guide.** *http://ode.org/ode-latest-userguide.html* [2005-06-21], 2001-2004.

Tanawongsuwan R., Stoytchev A., Essa I., **Robust Tracking of People by a Mobile Robotic Agent,** *Technical Report GIT-GVU-99-19*, 1999.

Thrun S., Bennewitz M., Burgard W, Cremers A.B., Dellaert F., Fox D., Hahne, D., Rosenberg C., Roy N., Schulte J., Schulz D., **MINERVA: a second-generation museum tour-guide robot**. *Robotics and Automation*, 1999.

Tomatis N., Philippsen R., Jensen B., Arras K.O., Terrien G., Piguet R. and Siegwart R. **Building a Fully Autonomous Tour Guide Robot: Where Academic Research Meets Industry**. *Proceedings of the 33rd International Symposium on Robotics*, Stockholm, Sweden, October 7 – 11, 2002.
Willeke T., Kunz C., Nourbakhsh I., **The history of the Mobot museum robot series: An Evolutionary Study**. *Proceedings of FLAIRS 2001*. Florida. May 2001.

Zadeh L.A., **Fuzzy Sets**. *Information and Control*, 8: 338-353, 1965.

Zebrowski P., **Communication In Multi-Robot Systems,** http://www.sfu.ca/~pzebrows/cmpt816/, 2004.

# Appendix I: Example Behaviour File

This is the behavior file "FacePerson". This behaviour is rather straightforward and contains instructions to turn the robot's front towards the person.

```
( PersonS1 )   => (RotSpeed VERY_FAST_LEFT);
( PersonS2 )   => (RotSpeed VERY_FAST_RIGHT);

( PersonSE )   => (RotSpeed VERY_FAST_RIGHT);
( PersonseE )  => (RotSpeed VERY_FAST_RIGHT);
( PersonE )    => (RotSpeed VERY_FAST_RIGHT);

( PersonSW )   => (RotSpeed VERY_FAST_LEFT);
( PersonswW )  => (RotSpeed VERY_FAST_LEFT);
( PersonW )    => (RotSpeed VERY_FAST_LEFT);


( PersonNE )   => (RotSpeed FAST_RIGHT);
( PersonneE )  => (RotSpeed VERY_FAST_RIGHT);

( PersonNW )   => (RotSpeed FAST_LEFT);
( PersonnwW )  => (RotSpeed VERY_FAST_LEFT);
```

## Appendix II: Simulator Config File

This is an extract from the file "RobotType_TourGuide.xml" that defines the objects of which the TourGuide robot is made up and the sensors and actuators that it contains.

```
<?xml version="1.0" encoding="utf-8"?>
<xode version="1.0r20" name="motorcycle"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://tankammo.com/xode/1.0r20/xo
        de.xsd">
   <ext ename="robotType">
      <robotType name="TourGuide">
        <actuator class="mars.actuators.HingeController" name="act1">
          <parameter name="joint" svalue="JointRight"/>
          <parameter name="controllerType" svalue="velocity"/>
          <parameter name="maxForce" value=".3"/>
        </actuator>
        <actuator class="mars.actuators.HingeController" name="act0">
          <parameter name="joint" svalue="JointLeft"/>
          <parameter name="controllerType" svalue="velocity"/>
          <parameter name="maxForce" value=".3"/>
        </actuator>

        <sensor class="mars.sensors.Telemeter" name="Vision">
          <parameter name="body" svalue="Body"/>
          <parameter name="relativeDirection">
            <vector x="0" y="-1" z="0"/>
          </parameter>
          <parameter name="relativePosition">
            <vector x="0" y="0" z="1.3"/>
          </parameter>
          <parameter name="range" value="60" />
          </sensor>

        <sensor class="mars.sensors.GPS" name="GPS">
          <parameter name="body" svalue="Body"/>
          <parameter name="relativeDirection">
            <vector x="0" y="-1" z="0"/>
          </parameter>
          <parameter name="relativePosition">
            <vector x="0" y="2" z="0"/>
          </parameter>
        </sensor>
</robotType>
</ext>
```

```
<ext ename="appearances">
    <appearance geom="BodyG">
      <texture filename="img/robot.jpg" />
    </appearance>
 </ext>
 <world>
    <space>
      <body name="Body">
                    <transform>
                          <position x="0" y="2" z="1.1" />
                    </transform>
                    <geom name="BodyG">
                          <box
                                sizex="4.5"
                                sizey="4"
                                sizez="1"
                          />
                    </geom>
                    <mass_shape total="35">
                          <box
                                sizex="4.5"
                                sizey="4"
                                sizez="1"
                          />
                    </mass_shape>
              </body>
              <body name="WheelLeft">
                    <transform>
                          <position x="2.575" y="1.5" z="0.6" />
                    </transform>
                    <geom name="WheelLeftG">
                          <sphere
                                radius="0.25"
                          />
                    </geom>
                          <mass_shape total="1">
                          <sphere
                                radius="0.25"
                          />
                          </mass_shape>
              </body>
    </space>
   </world>
</xode>
```