FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Humanoid Robot NAO: Developing Behaviors for Soccer Humanoid Robots

Luís Miranda Cruz

State of the Art Review

Master in Informatics and Computing Engineering

Supervisor: Luis Paulo Reis (PhD.)

Supervisor: Armando Sousa (PhD.)

12th February, 2011

Abstract

The topic of humanoid robot control has driven many research in areas such as robotics, artificial intelligence, computer vision, among others. More and better technologies have been developed, providing a wide range of humanoid robots and simulators which is very useful in this task. Nowadays, humanoids are equipped with several degrees of freedom leading to very versatile movements but also more complexity to the inherent problems.

Robocup is a robotic soccer international competition that promotes the research in these areas, including humanoid leagues, both on simulated and real environments. The FCPortugal team has already a good record of wins in RoboCup. The team intends to study these problems due to its participation in these humanoid leagues. In this context, this dissertation proposes the development of automated processes for optimizing behaviours and generate sequences of behaviours for a given task with regard to the environment.

This document presents the state of the art of the dissertation, describing different optimization methods, such as *Hill Climbing*, *Simulated Annealing*, *Tabu Search* and *Genetic Algorithms*. It also describes the topic of *Machine Learning*, the *planning problems*, and AI search methods, such as the *greedy best-first search* and the *A* search* algorithms. Humanoid control is introduced, explaining the behaviours at motor, skill and task level, and presenting the work already developed by FCPortugal in behaviours generation, namely the *Step-Behaviours*, *Slot-Behaviours* and *Central Pattern Generators*. The humanoid that will be used in the scope of this project will be the NAO and the robotic simulator will be the SimSpark, both the official technologies for RoboCup.

Resumo

O controlo de um robô humanóide é um tópico que tem impulsionado muita pesquisa em áreas como a robótica, inteligência artificial, visão por computador, entre outras. Mais e melhores tecnologias têm vindo a ser desenvolvidas estando disponível uma vasta gama de robôs humanóides e simuladores robóticos que ajudam a esta tarefa. Cada vez mais os humanóides são equipados com um elevado número de graus de liberdade, o que apresenta grande versatilidade de movimentos mas acrescenta a complexidade dos problemas de controlo.

O RoboCup é uma competição internacional de futebol robótico que fomenta a pesquisa nestas áreas, integrando ligas de humanóides, tanto em ambiente simulado como real. A equipa FCPortugal apresenta já um bom historial de vitórias no RoboCup e dada a sua participação nestas ligas de humanóides, pretende estudar estes problemas. Nesse âmbito, esta dissertação propõe o desenvolvimento de processos automáticos de optimização de movimentos e de geração de sequências de movimentos para realização de tarefas de robôs humanóides.

Neste relatório é apresentado o estado da arte da dissertação, apresentando os diversos algoritmos de optimização como *Subir a Colina*, *Arrefecimento Simulado*, *Pesquisa Tabu* e *Algoritmos Genéticos*. São também introduzidos os temas de *Aprendizagem de Máquina*, *Problemas de Planeamento*, e algoritmos de pesquisa, descrevendo os algoritmos de pesquisa gulosa e pesquisa A*. O controlo em humanóides é explicado, descrevendo os comportamentos ao nível de motor, habilidade e tarefa, assim como o trabalho desenvolvido pela FCPortugal na geração de comportamentos, nomeadamente os *Step-Behaviours*, *Slot-Behaviours* e *Central Pattern Generators*. O robô humanóide que será utilizado neste projecto será o NAO e o simulador será o SimSpark, ambos tecnologias oficiais do RoboCup.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Chapter 1

Introduction

This section briefly describes the motivation and goals behind this project, as well as its context and the document outline.

1.1 Context

This dissertation is related with humanoid robot agents designed to play soccer. The idea of soccer-playing robots was introduced in order to stimulate and challenge the research in many scientific areas.

In 1996, took place in Japan the first robotic soccer competition – the preRoboCup-96. Since then, hundreds of researchers yearly throughout the world to engage themselves and their students in RoboCup [CFH+02]. RoboCup, originally named *Robot Soccer World Cup* is an international competition where several researchers come together to prove the concepts they developed and show them applied in soccer matches. The best robot team wins the contest. There are different leagues, including leagues beyond the soccer domain, related to rescue or domestic applications.

Currently, RoboCup includes the following leagues:

- RoboCup Soccer
  - Standard Platform League
  - Small Size League
  - Middle Size League
  - Simulation League
    - 2D Soccer Simulation
    - 3D Soccer Simulation
    - 3D Development
    - Mixed Reality Soccer Competition

    - Humanoid League

- RoboCup Rescue
    - Rescue Robot League
    - Rescue Simulation League
        * Rescue Agents
        * Virtual Robots
- RoboCup@Home
- RoboCupJunior
    - Soccer Challenge
    - Dance Challenge
    - Rescue Challenge
    - General

In one of the many leagues of RoboCup, is the humanoid league. In the humanoid league the game is played by teams of humanoid robots. Humanoid robots are robots with a human-like appearance designed to behave like humans. A very popular humanoid is Nao. Nao has several sensors and joints which make him physically capable of playing soccer. Further details about humanoid robots will be presented in 2.1.

Since 2000, a Portuguese team, known as FCPortugal[1], has been participating in Robocup competition. It is a joint project of the Universities of Porto and Aveiro and has already won several awards, including 3 World RoboCup International Competitions. FCPortugal has more than 60 international publications related with RoboCup, intelligent simulation and cooperative robotics. [RL10]

The team Research Interests include:

- Multi-Agent Systems (MAS) and Coordination in MAS;
- Intelligent Cooperative Robotics and Robotic Soccer (RoboCup);
- Intelligent Simulation, Agent Based Simulation;
- Soccer, Game Analysis, Strategic Reasoning and Tactical Modelling;
- Humanoid walking and skills development using optimization methodologies;
- Bridging the gap between simulated and real robotics.

[1]Official website available at: http://www.ieeta.pt/robocup/

1.2 Motivation

Robotics plays an important role in many levels of our society. Robots can be used in military operations, spacial missions, industrial tasks like painting or assembling, education, art, and so on. Usually, Robotics is applied for tasks that humans don't like to do, some of them because they are dangerous, repetitive, dirty, or boring. Besides, they are useful due to their robustness, precision and/or autonomy. When well programmed, they can perform some tasks better than any human.

Recently, the humanoid robots have been becoming increasingly popular. Several research projects are being made in order to make them dance, play musical instruments, do the household chores, and so on.

Humanoids can be a powerful way of human-machine interaction. It's natural that humans tend to freely interact with human-like entities. Historically, humans had constrained themselves to the technology, for instance, with the monitor, keyboard and mouse. With humanoids, humans can experience new kinds of interactions, not too different from the usual ones on their social lives [WF11].

Robocup as also integrated humanoids in some of its leagues. What makes RoboCup really interesting is the fact that in one competition researchers have to solve problems from different subjects related to Robotics, Artificial Intelligence, Computer Vision, Multi-Agent systems, Real-Time Reasoning, Machine-Learning, and so on. Its popularity also revealed to be a way to attract public to these research subjects. This means that the robot soccer is a really attractive application of technology. However, these technologies can be applied in many other problems, creating an huge social and economic impact.

The soccer game requires humanoid players to be capable of walking in many directions, to kick the ball with high precision, to get up when necessary, or even to defend the ball, if we're talking about a goal keeper. The better the robot executes these behaviors, the higher the performance of the team will be. So, it is important for a team to have behaviours that are robust, efficient and adaptable to different situations.

Even with high quality behaviours, the humanoid has to autonomously choose the sequence of behaviours that fits the best a given task. This is a real challenge because it might have a large number of behaviours, and with that, there are even more possible combinations for the sequence, so it has to reason and choose on the fly the one that could give a faster and more robust conclusion of the desired task.

1.3 Objectives

Based on this, the main goals for this dissertation are:

- Develop high quality behaviours for humanoid robot.

- Develop a capable of evaluate the performance of behaviours

- Develop an integrated method capable of predict different sequences of behaviors for the same tasks, and choose the one which gives the best results at real-time.
- Ensure that the approach works properly on both real and simulated humanoids.
- Execute experiments in different game situations and explore the results with simulated games against well-known RoboCup teams.
- Participate in robotic competitions, events, conferences and journals to validate the approach and prove the concept.

1.4 Document Outline

This report is organized in N chapters, in which the first one is this introduction. The second chapter presents the literature review, showing what is being done in the area and different approaches and methodologies used to solve related problems.

The third chapter describes the approach and work plan, decomposing the problem in specific tasks during the 20 weeks.

In the last chapter are exposed some conclusions related with the state of the art and with the work done so far.

Chapter 2

Literature Review

This chapter presents the literature review of this project providing background knowledge and some analysis of related work.

2.1 Humanoid Robots

An humanoid robot is a robot with a human-like appearance. This is, it's a robot which structure has two legs, two arms and a head. Besides, its movement is expected to be human-like, using legged locomotion.

We are familiar with humanoids as they've been inspiration for many fictional stories. Examples of these are *C-3PO* and battle droids in *Star Wars* films, or even *NS-5* in *I, Robot*. In addition, humanoids can be used as a research tool in several scientific areas. This section presents the most important humanoid robots as well as some related projects.

2.1.1 Honda ASIMO

In 2008, the Honda robot ASIMO conducted the Detroit Symphony Orchestra. It was a remarkable fact in the robotics community. The name ASIMO is an acronym for "Advanced Step in Innovative MObility". As seen in figure 2.1, it looks like a small astronaut, standing 130 centimetres and weighing 54 kilograms. The 34 DOF allows it to do things like walking with a tray, or handle a cart. It is capable of walking fast, up to 3km/h forward, change direction and walk up/down stairs smoothly. It can even reach 7km/h by adopting a special running gait. [Hon07, SWA$^+$02] Although its popularity, ASIMO is still a really expensive technology. [Hes02]

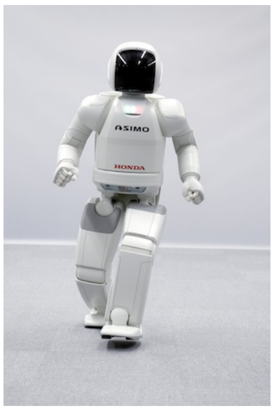
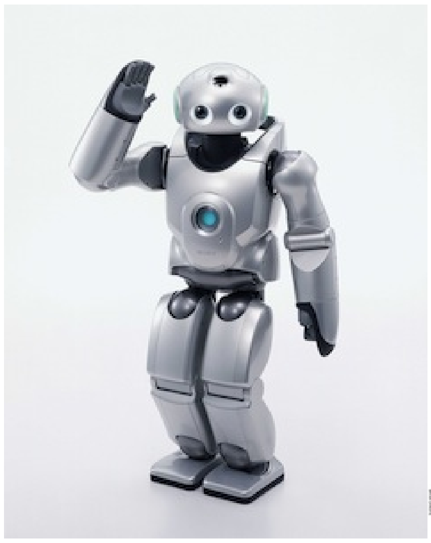


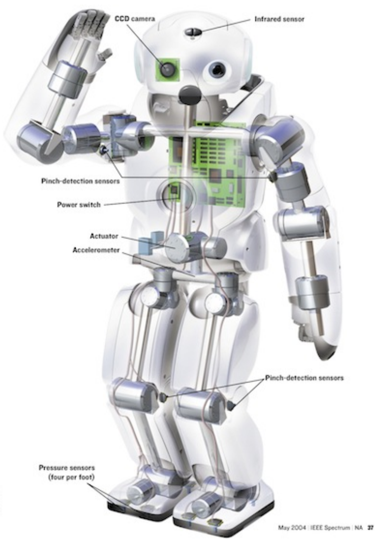
Figure 2.1: Honda ASIMO.

2.1.2 Sony QRIO

Qrio, which stands for "Quest for Curiosity", was an humanoid robot designed by Sony for use within the home. It could sing, dance and its successful development of integrated walking, jumping, and running. Its movement control technology lead to the world record of the first running humanoid robot [Gui04]. In one of its performances Qrio showed up conducting the Tokyo Philharmonic Orchestra in a rehearsal of Beethoven's Fifth Symphony. Nevertheless, in 2006, Sony announced the end of its Aibo robot line as well as the development of Qrio. Qrio stood at 58 cm tall and weighted 7.3Kg. It was able to interact with humans, thanks to its face and voice recognition systems, its speech synthesis and its remarkably fluid motion due to the equipped 38 DOFs. [Gep04]



(a) Normal view of Qrio.



(b) Qrio in detail. [Gep04]

Figure 2.2: Sony QRIO.

2.1.3 TOPIO - TOSY Ping Pong Playing Robot

TOPIO is an Vietnamese humanoid robot designed by TOSY Robotics to play ping-pong. It has 1.88m in height, 120kg in weight and 39 degrees of freedom. Ping Pong is challenging because it requires understanding of dynamic environments, accurate real-time vision, fast actuation, and intelligence to play the game with a winning strategy [Rai08]. It has an Artificial Intelligence module that allows to continuously improve itself while playing.

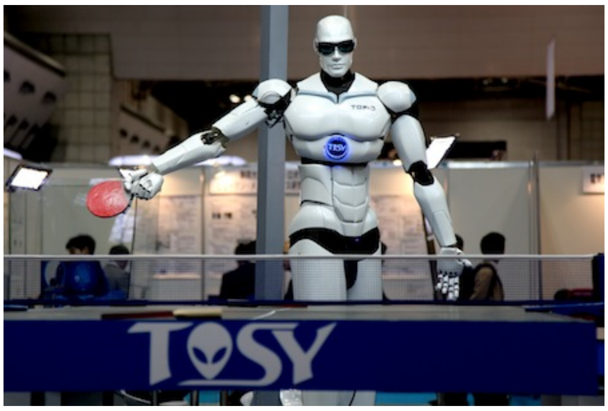


Figure 2.3: Tosyo ping-pong playing robot TOPIO.

2.1.4 iCub

The iCub is an humanoid robot created for research in embodied cognition. It has the dimensions of a 3.5 year old child. It is distributed as Open Source under the GPL/FDL licenses both for software and hardware and the entire design is available for download at the project's official website [1]. This is very useful in research projects because users and developers can freely exploit, customize and benefit from the work of other users. [MSV+08]

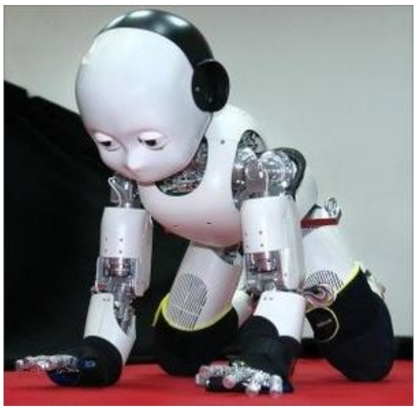


Figure 2.4: iCub Crawling - ©2010 RobotCub Consortium.

[1]Available at `http://www.robotcub.org.`

2.1.5 RoboNova Platform

RoboNova (cf. fig. 2.5) is a low cost humanoid developed by Hitec Robotics Company in Japan. It's original version comes with 19 DOF. It has two programming interfaces: the *RoboScript*, a graphical programming interface for beginners, and *RoboBasic*, for more advanced users, offering a programming language based on BASIC designed for controlling humanoid robots.

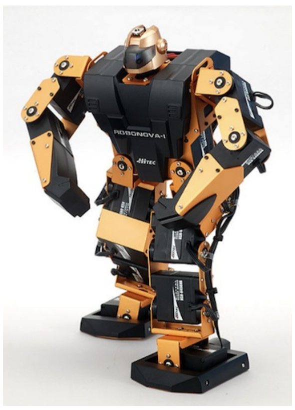


Figure 2.5: Robonova humanoid robot.

2.1.6 Aldebaran NAO

The humanoid robot NAO (cf. fig. 2.6) is built by the French company Aldebaran-Robotics[2]. It weighs 4.5 kg and stands 57 cm high. It was created with the goal of making available a performant biped robot to the maximum people. Humanoids like Asimo and Qrio were either not available to researchers or only available to the few teams that have enough funding to support the cost and maintenance.

It is cappable of recognizing features and human faces in the environment, to self-localize and to operate in the environment. Besides, this can be made while walking which is a concern for the participants of RoboCup humanoid league. Another feature is the open architecture. Most of the embedded software, including the operating system can be changed by the user. Low level hardware access is also open to allow users to change joint control laws. [GHB+08]

NAO has a total of 25 DOF. 11 DOF for the lower part, that includes legs and pelvis, and 14 for the upper part, that includes trunk, arms and head. Figure 2.7 presents the kinematics details.

Since 2008, NAO is the official robot for the RoboCup standard league.

2.2 Humanoid Simulators

In general, humanoid robots are a really expensive technology, as well as its maintenance. The use of simulation environments provides many advantages for research, development and test. For

[2]Official web site at: `http://www.aldebaran-robotics.com/`

Figure 2.6: The NAO developed by Aldebaran Robotics, now used in the RoboCup standard platform.

instance, in real environments every time a simple change is made in the agent, it's needed to reinitialize the robot and reinstall the new version all over again. Besides, all tests can be done without damaging the real robot and it's easier to retrieve detailed information about the execution.

The following sections will present some of the most popular simulators.

2.2.1 SimRobot

SimRobot is a simulator developed in the context of RoboCup by the B-Human team[3]. Unlike most of simulators, which have a client/server architecture, SimRobot consists of several components linked in a single application, as shown in figure 2.8. This approach has been chosen to offer a more comprehensive debugging, allowing to halt or stepwise the simulation without any concurrences.

The simulator has the following components: the *SimRobot core*, the *simulation scene*, the *user interface* and the *controller*.

The *SimRobot Core* models the robots and the environment, simulates sensor readings and executes commands given by the user or the controller. The *simulation scene* is a description of the robots and the environment, modeled using a XML specification language called RoSiML[4]. It consists of an external file that can be loaded at runtime.

The *user interface* (cf fig. 2.9) provides the graphical visualization and user interaction of the simulation. It allows the user to move objects around the scene, so that different situations can

[3]B-Human's Official Website available at http://www.b-human.de

[4]Robot Simulation Markup Language. XML Schema is available at: http://www.informatik.uni-bremen.de/spprobocup/RoSiML.html

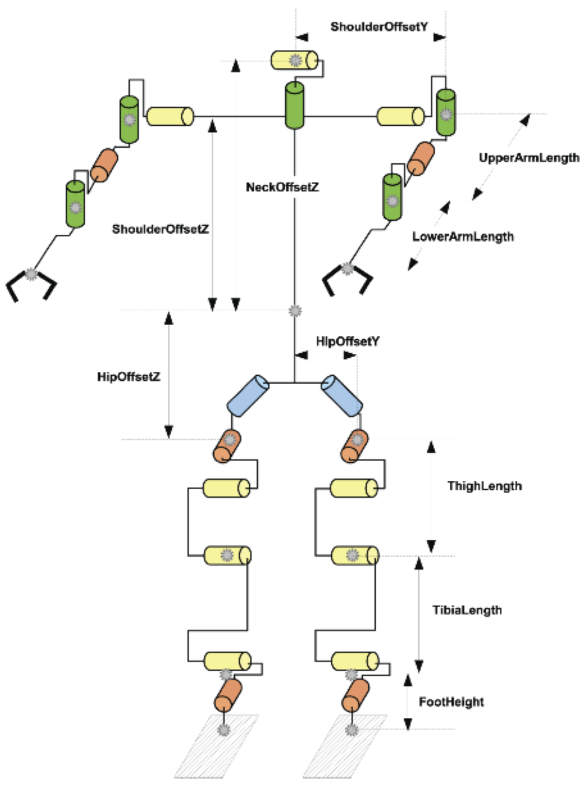


Figure 2.7: Detailed kinematics of NAO. Wrist joint not represented.

be easily modelled, gives a control for direct manipulation over the actuators and has a graphical visualization for sensors.

The *controller* is the component that implements the sense-think-act cycle. In each simulation step, it reads the sensors, plans the next action, and sets the actuators to the desired states. This controller has to be provided by the user, it contains the control software of the robots which usually is identical to the software running in the real robots. [LR08]

2.2.2 Webots

Webots^{TM} is a robotic simulation environment used to model, program and simulate mobile robots. A large set of sensors and actuator is available to equip each robot and an integrated IDE is provided to program the controllers. Once tested in simulation, the controllers can be transferred to real robots, including the humanoid robot Nao. In figure 2.10 can be seen the Webots^{TM}'s user interface.

2.2.3 OpenHRP3

OpenHRP3 (Open Architecture Human-centered Robotics Platform version 3) is a simulator and motion control library for humanoid robots developed at the National Institute of Advanced In-

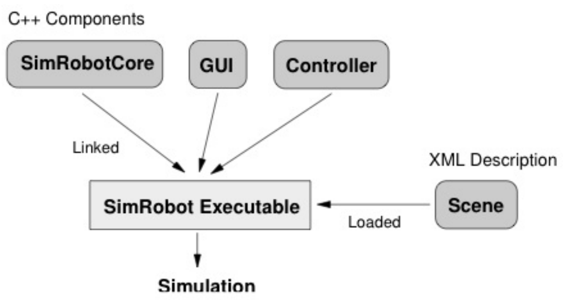


Figure 2.8: The modules of SimRobot and their dependencies. Source: SimRobot – Development and Applications [LR08].

dustrial Science and Technology, Kanehiro et al. (2004). It's a free tool that can simulate the direct dynamics of open and closed chains of multi-body systems. It includes a library of position, force/torque, vision, and inclination sensors, which are essential for developing control systems for the robots. The software also includes simple controllers for locomotion, balancing, a walking pattern generator as well as collision detection and avoidance schemes. Advanced controllers can be developed by the users. Besides, standard tools for plotting simulation results, OpenHRP has a 3D visualization interface to display the simulation.

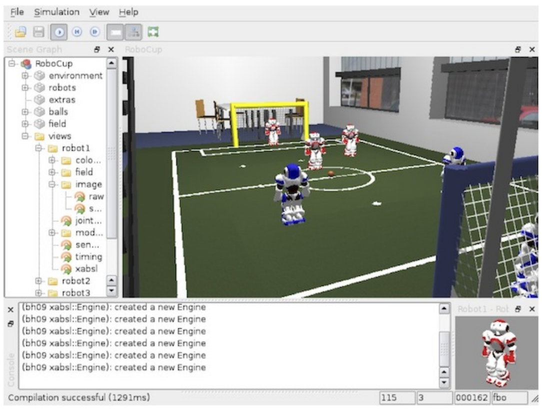
The software developed in OpenHRP can be directly used in the HRP humanoid robots [KHK+08] but this is not possible for other humanoid robots. The main reason is that several features in OpenHRP cannot be modified by the user. For example, the actuator models are fixed, and compliance in the robot joints or the robot links cannot be added to the model. Furthermore, there is very little documentation available for OpenHRP and since this is a free tool there is also no support from the developers. [MCDB+10]

2.2.4 Microsoft Robotics Studio Simulator

Microsoft Robotics Studio (MSRS) is a Windows-based environment for robot control and simulation. It provides a set of useful tools, including a three-dimensional simulator. In the simulator, entities can be created in order to include robots and objects like walls, obstacles, floors, etc. Each entity is provided with a set of callback functions for dealing with events such as closions and frame updates. It integrates the physics simulator PhysX created by Ageia. Entities in the simulation environment specify a physical description with parameters such as friction coefficient, mass and center of gravity. Screen shots of the simulator can be seen in figure 2.11. [Jac07]

2.2.5 SimSpark

SimSpark is a multi-agent simulation system and is used as the official simulator for the RoboCup Simulation League competition. Although it was created for robotic soccer simulation, the simulator core is the most generic possible and all soccer specific details are contained in a set of



```
(bh09 xabsl::Engine): created a new Engine
(bh09 xabsl::Engine): created a new Engine
(bh09 xabsl::Engine): created a new Engine
(bh09 xabsl::Engine): created a new Engine
(bh09 xabsl::Engine): created a new Engine
(bh09 xabsl::Engine): created a new Engine
```

```
Compilation successful (1291ms)    115    3    000162    fbo
```

Figure 2.9: SimRobot with the tree view on the left and two scene views on the right. [RLM+09]

plugins [BDR+10]. The simulator is designed in a client/server architecture having three main kinds of components: agents, the server, the monitor (cf fig. 2.12).

An agent is the software that drives the behaviour of a robot. Usually, they communicate with the simulation server via TCP sockets. The server is the component that hosts and manages the simulation process. It is responsible to constantly update the simulation state. Every interactions between the objects, collisions, gravity, and so on, are modelled by the server. Another responsibility of the server is to keep track of all the connected agents. The server provides perceptors to report the information about the senses of an agent, and provides effectors which allow agents to perform actions within the simulation.

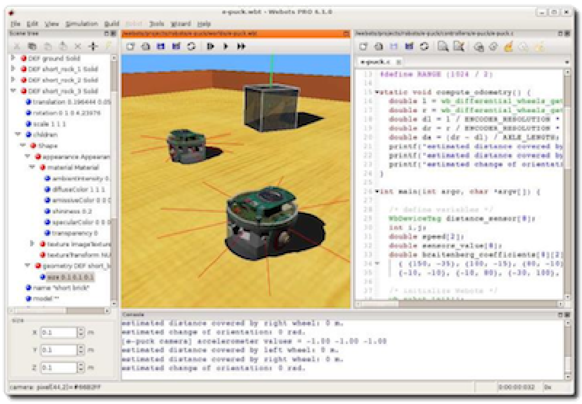
Finally, the monitor is the component responsible for the graphic visualization of the simulation. It connects to server and continuously receives a data stream with the information needed to render the scene. SimSpark provides different robot models for use by agents. The RoboCup 3D Soccer Simulation League currently uses a model of Nao humanoid. As shown in figure 2.13, the model is quite similar to the real Nao. [BDR+10]

2.3 Humanoid Behaviours

In this context, an humanoid behaviour is a set of trajectories of robotic joints composed in order express the control of the robot. Once specified it can be executed as many times as necessary. Behaviours can be expressed at the motor, skill or task level.

At motor level, behaviours are expressed by prescribing commands directly to system's actuators. At skill level, behaviours express the capabilities of the agent, for instance, walking, getting up, and so on. Skills are models expressed as parametrized programs for motor level controllers, without the ability to strategize about objectives or encode semantics about the world. Finally, task level behaviours are skills performed in order to achieve specified goals. [JM03]

Some related research, including the work done by FCPortugal team, is presented in the following sections.



```
#define RANGE (1024 / 2)

static void compute_odometry() {
  double l = wb_differential_wheels_get
  double r = wb_differential_wheels_get
  double dl = l / ENCODER_RESOLUTION *
  double dr = r / ENCODER_RESOLUTION *
  double da = (dr - dl) / AXLE_LENGTH;
  printf("estimated distance covered by
  printf("estimated distance covered by
  printf("estimated change of orientati
}

int main(int argc, char *argv[]) {

  /* define variables */
  WbDeviceTag distance_sensor[8];
  int i, j;
  double speed[2];
  double sensors_value[8];
  double braitenberg_coefficients[8][2]
    { {150, -35}, {100, -15}, {80, -10}
    {-10, -10}, {-10, 80}, {-30, 100},

  /* initialize Webots */
```

```
estimated distance covered by right wheel: 0 m.
estimated change of orientation: 0 rad.
[e-puck camera] accelerometer values = -1.00 -1.00 -1.00
estimated distance covered by left wheel: 0 m.
estimated distance covered by right wheel: 0 m.
estimated change of orientation: 0 rad.
```

Figure 2.10: Webots simulating e-puck robot with a tree-view of the objects on the left and a source code editor on the left.

2.3.1 Skill Level Behaviours Generation

Defining a skill by composing several commands at motor level controllers commands can be a very difficult task. In order to make things easier, a skill can be defined with the trajectory for each involved joint. These trajectories can be generated based on a sequence of start and end points in the joint space during some amount of time. [Pic08]

Some of the methods used by FCPortugal to define skill level behaviours are described in this section.

2.3.1.1 Step-based Behaviours

The step-based method was the first method used by FCPortugal. It generates behaviours using step functions. A step function (or staircase function) is a discontinuous function consisting of a series of constant functions, each one defined in some interval of time [Pic08].

So a joint trajectory will be defined as a step function in which the amplitude is the desired angle on each interval of motion. The function is described as follows:

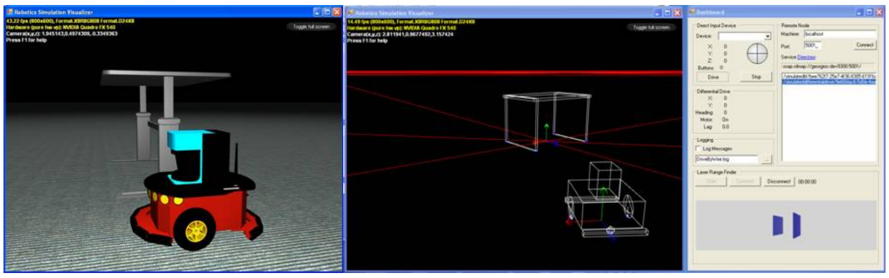
$$f(t) = \sum_{i=0}^{n} \theta_i \cdot \mathbf{1}_{A_i}(t), \forall t \in \mathbb{R} \tag{2.1}$$

where n is the number of intervals, A_i is the interval i, and θ_i is the desired angle for joint at interval i. $\mathbf{1}_{A_i}(t)$ is the indicator function of A_i and is defined as follows:

$$\mathbf{1}_{A_i}(t) = \begin{cases} 1, \text{ if } t \in A_i \\ 0, \text{ if } t \notin A_i \end{cases} \tag{2.2}$$

The main advantages of this method are [Pic08]:

- Simple to understand;



(a) Modular robot base with differential drive, laser range finder and bumper array.

(b) Physics primitive view.

Figure 2.11: Screen shots of Microsoft Robotics Studio Simulator.

- Simple to implement;
- Simple to define target trajectories (target angles and tolerances).

The main drawbacks of step-based method [Pic08]:

- Time from current angle to target angle is unpredictable;
- No control over the angular velocity trajectory;
- The same gain is used for all joints;
- Sensitive to overshoot reactions at the control level;
- The syntax of the configuration file is not user-friendly;
- The model is not flexible.

In FCPortugal, a step-based behaviour is specified in a XML file, consisting of a sequence of steps, and each of them has a sequence of joint positions, corresponding to the target angles that might be achieved in the step. One step finishes its execution when all the joints are approximately at the specified angle.

2.3.1.2 Slot Based Behaviours

Another approach to generate behaviours used by FCPortugal is the slot-based method. Slots are intervals of time where several joints are moved in parallel. This allows to define not only the trajectory points, but also the time in which those angles should be achieved.

Comparing with step-based method, this approach allows to have control over the transition time of the current point to the target point. So, instead of defining the states at each interval of time, in slot-based methods transitions are specified.

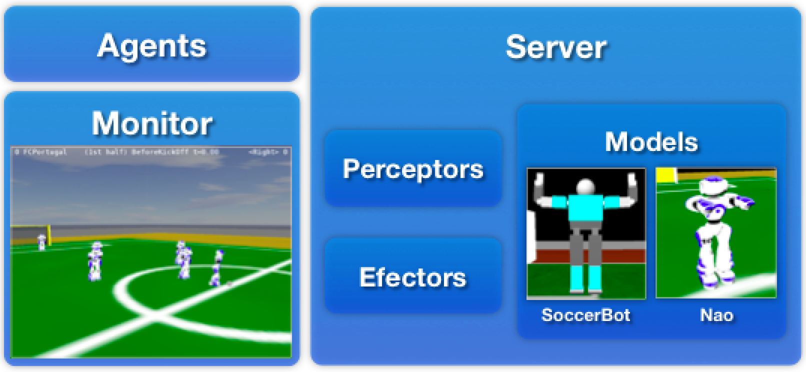


Figure 2.12: General architecture of SimSpark architecture.

In order to have smooth transitions between the current point and target point, sine interpolation is used. This works as follows: in each slot, the controller will interpolate between the current point and next point by performing a sine-like trajectory generated by the following expression:

$$f(t) = A\sin(\frac{\phi_f - \phi_i}{\delta}t + \phi_i) + \alpha, \forall t \in [0, \delta] \tag{2.3}$$

where $f(t)$ is the trajectory function, δ is the duration of slot in miliseconds ϕ_i is the initial phase (which will influence the initial angular velocity), ϕ_f is the final phase (which will influence the initial angular velocity), A is the amplitude and α is the offset. A and α are calculated using the following expressions:

$$A = \frac{\theta_f - \theta_i}{\sin(\phi_f) - \sin(\phi_i)} \tag{2.4}$$

$$\alpha = \theta_i - A \cdot \sin(\phi_i) \tag{2.5}$$

where θ_i and θ_f are the initial and final angles, respectively, and should be defined between $-\pi$ and π.

In the FCPortugal agent, this behaviours are described in XML files. In each behaviour is described a sequence of slots. Each slot has a time parameter and is composed by a series of moves for the involved joints, which will be executed at the same time. The table 2.1 describes these classes.

An example of a behaviour description can be seen in listing 2.1. Several slots are defined and each of them have a set of joint moves.

Figure 2.13: The humanoid robot Nao: real vs simulated. Adapted from [BDR+10].

Listing 2.1: Extract from a slot behaviour XML specification (KickLeft Behaviour of the FCPAgent) - February 2011.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<behavior name="KickRight" type="SlotBehavior">
    <slot delta="0.170" />
    <slot name="shiftWeight" delta="0.400">
        <move id="&lleg2;" angle="-11" />
        <move id="&lleg6;" angle="11" />
        <move id="&rleg2;" angle="-11" />
        <move id="&rleg6;" angle="11" />
        <move id="&lleg3;" angle="8" />
        <move id="&rleg3;" angle="8" />
        <move id="&lleg4;" angle="-14.5" />
        <move id="&rleg4;" angle="-14.5" />
        <move id="&lleg5;" angle="12.5" />
        <move id="&rleg5;" angle="12.5" />
    </slot>
    <slot name="kickLegUp" delta="0.500">
        <move id="&rleg3;" angle="20" />
        <move id="&rleg4;" angle="-40" />
        <move id="&rleg5;" angle="20" />
        <move id="&lleg3;" angle="0" />
        <move id="&lleg4;" angle="0" />
        <move id="&lleg5;" angle="0" />
    </slot>
</behavior>
```

The main advantages of this method are [Pic08]:

- Simple to understand and implement;
- Time from current angle to target angle is controlled;

Slot Behavior	
Class	**Properties**
Slot	delta(δ)
Joint	joint identifier target angle (θ) kp, ki, kd

Table 2.1: Description of the classes in a Slot Behaviour specification. Adapted from [Rei10].

- Some control over the angular velocities trajectories;
- The model is flexible;
- The motion description language is user-friendly and well structured.

The main drawbacks are [Pic08]:

- It is not possible to define more complex sinusoidal shapes;
- The angular velocities trajectories are not completely controlled.

2.3.1.3 Central Pattern Generator

Central pattern generators are biological neural networks that can produce coordinated multidimensional rhythmic signals, under the control of simple input signals. They are found both in vertebrate and invertebrate animals for the control of locomotion [RI06]. The concept of neural networks can also be applied in robot control in order to generate a suitable pattern for a walking motion.

The input parameters are described by the vector $a = (a_r, a_p, a_y)$ which corresponds to the lateral swing amplitude, forward swing amplitude and rotational amplitude, respectively. A gait phase, ϕ_{gait}, varies between $-\pi$ and π. ϕ_{leg} represents the phase of a leg and will correspond to $\phi_{gait} - \frac{\pi}{2}$ for the left leg and $\phi_{gait} + \frac{\pi}{2}$. The step of one leg during the whole gait is divided into five complex stages: *Shifting*, *Shortening*, *Swinging*, *Loading* and *Balance* [Rei10]. Further details can be found in [Pic08].

The main advantages of CPG behaviours are [Pic08]:

- More complex shapes are possible;
- Angular velocities trajectories are completely controlled;
- Allows several gaits with just one generator (forward walk, backward walk, sided walk, curved walk, curved walk and turn on the spot);
- The generated gait is very similar to the natural human behaviour.

The main drawbacks are [Pic08]:

CPG Behaviour Parameters	
Class	**Properties**
Behavior	delta(δ)
Pattern	joint identifier target angle (θ) period phase amplitude

Table 2.2: Description of the classes in a CPG Behaviour specification. Adapted from [Rei10].

- Requires complex knowledge about human behaviours
- The shifting stage leads to slower movements.

A behaviour specification file for a CPG behaviour consists of a series of patterns for each specified joint [Rei10]. The table 2.2 describes the parameters used to specify the behaviour.

2.3.2 Task-level Behaviours

In task-level behaviours it's intended to combine the known behaviours in a sequence ensuring that the task is successfully completed. Several researches are being made on this subject.

An interesting approach is in [OTI$^+$00], in which the tasks are learned by watching them being performed by the human hand. The idea is to represent actions in symbolic task models, acquired through observation. For each task, in the acquisition process is specified a set of *attention points* (APs), dividing the task in the steps that requires attention and analysis. Through observation, the system creates a model of the task classifying the actions between the APs, taking account of attributes like in which hand is the performing, the absolute position in 3D space, the kind of action (if it's grasp, release, pour, hand over), and many others.

Another interesting feature in this approach is the assignment of priorities to the attributes. For instance, the information of which was the hand that performed the task is expected to be less important than the kind of action. First, the robot tries to perform the task exactly as the described in the model. If it fails, then it will iteratively ignore the attributes with lower priority. This task model is flexible enough to apply an acquired task in different environments. [OTI$^+$00]

2.4 Optimization

As long as humankind exists, we strive for perfection in many areas. We want to reach a maximum degree of happiness with the least amount of effort. In our economy, profit and sales must be maximized and costs should be as low as possible. [Wei09]

Therefore, as usually, there is a science to deal with this problems - Optimization. It refers to find the best elements from a set of possible alternatives, according to a criteria.

Thus, an optimization can be stated by a set of parameters, known as **decision parameters**, describing the alternatives, and a mathematical function, known as **objective function**, that analyses each alternative according to these parameters. The objective function gives a quantity to classify a candidate solution of the problem. If the goal is to maximize the function, it is called **utility function**, otherwise, if it's to minimize, then it's called **cost function**.

There are many approaches on solving optimization problems. In the following sections, a classification of optimization solving methods is presented, as well as the most popular algorithms.

2.4.1 Classification

2.4.1.1 Deterministic and Stochastic Algorithms

Optimization algorithms can be divided in two basic classes: deterministic and stochastic algorithms. Deterministic algorithms are used when the utility of a possible solution can be efficiently calculated. In each step of a deterministic algorithm exists at most one way to proceed.

On the other hand, stochastic algorithms contain instructions that use random numbers in order to decide what to do or how to modify data. Such methods apply in the usual case where a closed-form solution to the optimization problem of interest is not available and where the input-information ito the optimization methods may be contaminated with noise. [Wei09, WW04]

2.4.1.2 Individual and Population-based Algorithms

Another kind of classification that can be made it's individual or population-based methods. Individual-based methods deals only with one possible solution in each iteration. Conversely, in optimization-based methods multiple alternatives are handled at the same time. [Pic08]

2.4.1.3 On-line and Off-line Optimization

According with the use case, methods can be classified with regard to the required speed. It's possible to distinguish **on-line optimization** in which problems need to be solved quickly in a time span between ten milliseconds to a few minutes. In order to find a solution in this short time, optimality is normally traded in for speed gains. With **off-line optimization**, time constraints are less important and the user can wait even days if that means obtaining an optimal or close-to-optimal solution. [Wei09]

2.4.2 Hill Climbing

The Hill Climbing (HC) algorithm is simple to implement and performs well in several situations, achieving reasonable results. It starts with a arbitrary guess of the solution, and then attempts to find a better solution by analysing the neighbour alternatives, which are obtained by making minor changes to the guess. It terminates when reaches a state where no neighbour has a better value.

Hill-climbing algorithms typically choose randomly among the set of best successors if there is more than one. It often makes rapid progress towards the solution because it is usually easy to improve from a bad state. Unfortunately, hill-climbing is not so good when [RN10]:

- There is a local maximum, which is a peak that is higher than each of its neighboring states but lower than the global maximum.
- There is a plateau, which is a flat area of the state-space landscape. It can be a flat local maximum, from which no better state, or a shoulder, from which progress is possible. A hill-climbing search might get lost on the plateau.

Hill climbing is good for finding a local optimum, but it is not guaranteed to find the best possible solution (the global optimum) out of all alternatives. The algorithm is shown in algorithm 1.

Algorithm 1 Hill Climbing search algorithm. [RN10]

```
1: $current \leftarrow GetInitialSolution()$
2: loop
3:     $neighbour \leftarrow$ a highest-valued successor of $current$
4:     if $neighbour.VALUE < current.VALUE$ then
5:         return $current.STATE$
6:     end if
7:     $current \leftarrow neighbour$
8: end loop
```

2.4.3 Simulated Annealing

Simulated Annealing (SA) is a stochastic method to avoid getting stuck in local, non-global minima, when searching for global. minima [DA89]. Annealing, in metallurgy, is the process used to temper or harden metals and glass. This procedure hits the material to a high temperature so that the material is liquid and the atoms can move freely. The temperature of the material is then slowly decreased so that, at each temperature, the atoms can move enough to begin adopting the most stable orientation. If the material is cooled slowly enough, the atoms are able to achieve the most stable orientation.

The Simulated Annealing algorithm is presented in 2. When it's time to choose the next state, instead of choosing the best state, however, it picks a random state. If the random state has a better value, it is accepted. Otherwise, it accepts the move with some probability less than 1. The probability decreases as the state becomes worse, and also as the "temperature" T goes down. In the beginning of execution the temperature is high, and decreases over time. So worse states are more likely to be accepted at the start, but more unlikely as T decreases.

2.4.4 Tabu Search

Tabu Search (TS) is a variant of HC method, proposed by Fred Glover in 1986. This algorithm maintains a list of the k last visited states in order to prevent a previous state from being repeated.

Algorithm 2 Simulated Annealing. [RN10]

```
1: $current \leftarrow GetInitialSolution()$
2: for $t = 1$ to $\infty$ do
3:     $T \leftarrow schedule(t)$
4:     if T=0 then
5:         return $current$
6:     end if
7:     $next \leftarrow$ a randomly selected successor of $current$
8:     $\Delta E \leftarrow next.VALUE - current.VALUE$
9:     if $\Delta E > 0$ then
10:        $current \leftarrow next$
11:    else
12:        $current \leftarrow next$ only with probability $e^{\Delta E/T}$
13:    end if
14: end for
```

With this, it's preferable to visit worse states rather than repeating them. As well as improving efficiency by avoiding cycles, this improvement can allow to escape from a local optima, since it can accept worse states. [GHP+97, RN10] The pseudo-code of this method is presented in algorithm 3.

Algorithm 3 Tabu Search algorithm. [Pic08]

```
1: $current \leftarrow GetInitialSolution()$
2: $evaluation1 \leftarrow Evaluate(current)$
3: $tabuList \leftarrow \{\}$
4: while $TerminationConditionsNotMet()$ do
5:     $neighbours \leftarrow GetNeighbourhood(current)$
6:     for $i = 1$ to $Length(neighbours)$ do
7:         if $NotMember(neighbours[i], TabuList)$ then
8:             $Add(TabuList, neighbours[i])$
9:             $evaluation2 \leftarrow Evaluate(neighbours[i])$
10:            if $evaluation2 < evaluation1$ then
11:                $current \leftarrow neighbours[i]$
12:                $evaluation1 \leftarrow evaluation2$
13:            end if
14:        end if
15:    end for
16: end while
17: return $current$
```

2.4.5 Genetic Algorithms

The Genetic Algorithm (GA) was developed by John Holland (1975) it's an optimization method bases on the principles of genetics and natural selection. A GA allows a population composed of many individuals to evolve under certain selection rules to a state that maximizes the "fitness" (i.e., minimizes the cost function) [HH04]. In this case, an **individual** (also called **chromosome**) is a set of parameters, known as **genes**, and represents a possible solution to the optimization problem.

The algorithm starts by creating a new population of individuals. Then, it starts the evolution by applying several operators to the population, generating a new one. Along the algorithm, a sequence of new populations is generated.

The simplest form of GA, involves three types of operators: *selection*, *crossover* and *mutation*.

- **Selection** This operator selects chromosomes in the population for reproduction, accordingly to a given probability. The fitter the chromosome, the more times it is likely to be selected to reproduce.
- **Crossover** This operation randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring. For example, the strings 10000100 and 11111111 could be crossed over after the third locus in each to produce the two offspring 10011111 and 11100100. The crossover operator roughly mimics biological recombination between two single-chromosome organisms.
- **Mutation** This operator randomly flips some of the genes in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (eg. 0.001). [Mit98]

In addition, there is a selection method known as **elitism**, which defines the number of the best individuals at each generation, that are guaranteed to survive in the next generation. Such individuals can be lost if they are not selected to reproduce or if they are destroyed by crossover or mutation. Many researchers have found that this technique leads to better results [Mit98].

Some of the advantages of a GA include that it [HH04]:

- Optimizes with continuous or discrete variables;
- Doesn't require derivative information;
- Simultaneously searches from a wide sampling of the cost surface;
- Deals with a large number of variables;
- Is well suited for parallel computers;
- Optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum);
- Provides a list of optimum variables, not just a single solution;
- May encode the variables so that the optimization is done with the en- coded variables;
- Works with numerically generated data, experimental data, or analytical functions.

The algorithm 4 shows the pseudo code of a generic GA. The mutation function receives the parameter p_m which is the probability of occurring a mutation in a chromosome.

Algorithm 4 Genetic Algorithm. Adapted from [Pic08].

```
1: $population \leftarrow CreateInitialPopulation()$
2: $Evaluate(population)$
3: while $TerminationConditionsNotMet()$ do
4:     [Elitism] $elite \leftarrow Elitism(population)$
5:     [Selection] $parents \leftarrow Selection(population)$
6:     [Crossover] $children \leftarrow Crossover(Parents)$
7:     [Mutation] $mutants \leftarrow Mutation(children, p_m)$
8: end while
```

2.5 Machine Learning

Over the years, psychologists had particular interest in studying the concept of learning. The complexity of such a natural thing has led to many theories. Some of them examine learning as a process, others as the product of a process [Smi09]. In both of them, there is a process, so in order to learn, a cognitive process of acquiring or knowledge has to be made.

Machine learning can be stated has the branch of artificial intelligence that deals with the design of algorithms that allow a machine to learn by itself thus reducing (perhaps eliminating) the need for human intervention in the learning process [Pic08].

The importance of machine learning can be seen by its real-world applications. We can find examples like speech-recognition, where the accuracy is greater if someone trains the system then if it is programmed by hand. Besides, machine learning makes possible to the system to gain more accuracy, even after the user buys the product, by learning with the user. Another application that is being successfully used is robot control, where machine learning is used for acquiring control strategies. The recent Darpa-sponsored competition, involving a robot driving autonomously for over 100 miles in the desert was won by a robot that used machine learning to refine its ability to detect distant objects (training itself from self-collected data consisting of terrain seen initially in the distance, and seen later up close). Other applications can be handwriting recognition, face recognition, systems that automatically classify microscope images of cells, data-mining, and many others. [Mit06]

The machine learning methods can be compared to the scientific method, in which there is the process of observing a phenomenon, forming a hypothesis, making predictions, and then iteratively revising the hypothesis before making further observations. Machine learning aims to automate this process, so that computer-automated predictions can make a task more efficient, accurate, or cost-effective than it would be using only a human decision. [mac]

Machine learning techniques can be broadly classified in three main categories: **supervised learning**, **unsupervised learning**, and **reinforcement learning**. In supervised learning systems is supplied a training set with input-output instances. The learner has to autonomously adapt its parameters in order to make the correct matching between a given input and the desired output. Unsupervised learning there is no such thing as a training set giving the desired outputs, and consequently the learner has to adapt its parameters autonomously given only a set of inputs. Finally, reinforcement learning. The third type, called the reinforcement learning, bridges a gap between supervised and unsupervised categories. The learner hasn't an input-output training set, however it has some kind of feedback about the environment. The feedback allows the learner to know if some action has been rewarding or punishable, allowing the learner to adapt its parameters in order to increase the rate of rewarding actions. [Kon00]

2.6 Planning Problems

Planning is the process of generating (possibly partial) representations of future behaviour prior to the use of such plans to constrain or control that behaviour. The outcome is usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents. A planning problem has an *initial state* of the world, a desired *goal state* (or set of goal states) of the world, and a *planning domain*. A planning domain is a of actions which can be executed in some particular world-states, named *preconditions* and changes the world state, having a set of *effects*. A planner solves a planning problem by producing a legal sequence of actions which takes from the initial state to the final state. [WKoT99, CIR99]

2.6.1 Defining a Planning problem

In order to provide a standard encoding language to describe planning problems, in 1998, Drew McDermott released the Planning Domain Description Language (PDDL), which encourages sharing of problems and algorithms, and provides meaningful comparison of the performance of planning methods on different problems. It separates the domain description and the problem instance description in different files, so that the same domain can be used for different planning problems. The syntax is inspired by Lisp[5], so much of the structure of a domain description is a Lisp-like list of parenthesised expressions. [FL03, GHK$^+$98]

As we can see, the description of a planning problem defines a search problem: we can search from the initial state through the space of states, looking for a goal [RN10]. So, in order to solve a planning problem, AI search methods can be used. Further details are presented in the next section.

2.7 Problem Solving with Search Methods

A solution for a formulated problem can be seen as sequence of actions, in which the solution is one particular sequence which leads to a goal state. The sequences of actions can be stated with a

[5]List processing language invented at MIT in the 50s and developed in the 60s.

search tree, where the root is the initial state, the branches are the actions, and the nodes represent states in the space of the problem.

The essence of search methods is the generation of the search tree. It starts in the initial state, which is the root, tests whether it is a goal state, and, if it is not, expands the current state applying each legal action, thereby generating a new set of nodes (states). After that, the method must choose which one of non-visited nodes must go next, and repeat the process until reach the goal state.

Search algorithms can be broadly divided in **uninformed** (also called blinded search) or **informed** search algorithms. The difference is that in informed search algorithms some guidance on where to look for solutions is given.

The most popular uninformed methods are *breadth-first search*, *uniform-cost search*, *depth first search*, *depth limited search* and *Iterative deepening depth-first search*.

In the scope of informed methods, *greedy best first search*, *A* search* are also very popular.

The performance of search algorithms can be evaluated with the following properties:

- **Completeness:** Is the algorithm guaranteed to find the solution when there is one?
- **Optimality:** Does it find the optimal solution?
- **Time complexity:** How long does it take to find a solution?
- **Space complexity**: The amount of memory needed to perform the search.

Depending on the problem to solve, some of these properties can be more important than the others.

2.7.1 Greedy best-first search

Greedy best-first search expands first the node estimated to be closer to the goal. Thus, it evaluates nodes using just an heuristic function, $h(n)$. The heuristic gives the estimated cost of the cheapest path from the node n to the goal, so the performance of the method is highly committed to the quality of the heuristic. The algorithm is called "greedy" because at each step it tries to get as close to the goal as it can. Because of that, this method can return non optimal solutions. The greedy best-first search is also incomplete in infinite spaces. With finite states it can be incomplete in the presence of loops. However, using some technique to avoid loops, the algorithm can be complete.

2.7.2 A* search

A* search (pronnounced A-star search) is a similar to method the greedy algorithm. Nevertheless, now, the evaluation function, $f(n)$, combines the heuristic, $h(n)$, with a cost-function, $g(n)$, which gives the path cost from the start node to the node n:

$f(n) = g(n) + h(n)$.

Provided that the heuristic function, $h(n)$ satisfies certain conditions, A* search both complete and optimal.

2.8 Conclusion

As presented in this chapter there are a lot of researching being made in fields related to humanoid robotics. Several technologies were developed in order to help this research, including the referred simulators and humanoid robots. Nevertheless, there is still a lot of work to do. The creation of smooth and well optimized behaviours is still a very difficult task, and the RoboCup initiative have been challenging its participants to focus on this kind of problems. Topics related to AI, such as optimization, machine learning, planning problems and search algorithms were also described. These are methods highly related to the problem of this dissertation. In robotic soccer, the humanoid players are expected to have different sequences of behaviours for the same task. However, depending on the environment, some are better than others. So, rather than having the job done, it's expected to do the job with the better approach. The hand-actions system, briefly described in 2.3.2, is not flexible enough when the goal is to reason over different ways of doing the same task. It only tries to learn them by observation and doesn't try to mix the known behaviours [OTI+00].

Chapter 3

Approach and Work Plan

This chapter briefly describes the proposed approach for this dissertation. The global architecture of the system will be presented, including a simple description of the main components. It will also present the work plan schedule estimated for the twenty weeks of dissertation, providing a description of the main tasks.

3.1 Research Approach

In order to overcome the needs presented in the previous sections it's intended to develop a software solution that can classify behaviours, generate and classify sequences of behaviours and improve the generated sequence as the humanoid gains experience in the desired task. The proposed architecture is illustrated in the figure 3.1.

The component *Behaviours Knowledge* stores all the known behaviours and the parameters related to its classification. The *Classification Engine* will be responsible of monitoring the execution of the behaviours and evaluating its performance, updating the necessary parameters in the *Behaviours Knowledge*. This component will use machine learning techniques, in order to avoid the need of studying the relative importance of all the information available about the execution.

Subsequently, it will be integrated the *Task level Behaviours Generator* which is prompted by the agent to give the best sequence of behaviours to execute a given task in a given situation. It will use the knowledge acquired by the *Classification Engine*, in order to give the sequence that best fits the task in that situation. This is a non-trivial problem that can be stated as a planning problem and use a search method to solve it. Nevertheless, the environment is always changing and other tasks with high priority can appear, interrupting the process. Besides, a precise prediction of the effect of a known behaviour is needed in order to make a good planning.

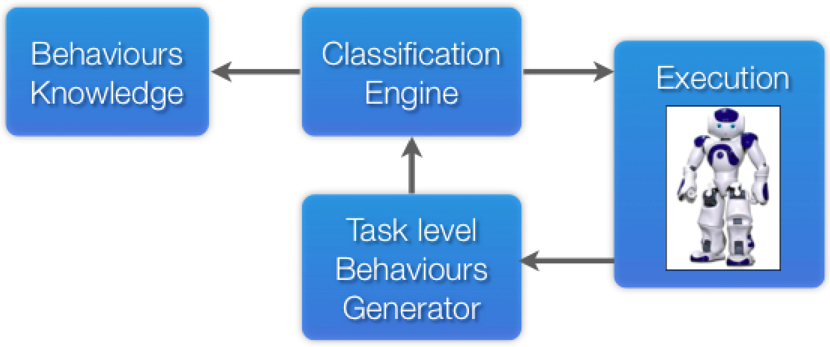


Figure 3.1: Global architecture of the system.

3.2 Work Plan

This section aims to present the work plan throughout the twenty weeks of the master's dissertation. At the end of each development module, a set of experiments will be performed both in real and simulated environments, using the humanoid Nao and the simulator SimSpark, respectively. A brief description of each task and the intended work plan are presented below.

3.2.1 Bibliographic Research

To better understand the problem, the thesis work will begin with a revision of related work. This task was initiated in the context of the course unit "Dissertation Planning". A literature review was performed on the available of humanoids and robotics simulators technologies, and on some of related problems. It was also reviewed optimization and classification techniques which are widely used in this problems. However it will be spent more time to continue the research. It will focus on the study of more research projects related to behaviour-control systems, acquisition and reasoning of task-level behaviours. During the thesis, bibliographic research will be an open topic in order to give an updated overall context of the scientific development on these areas.

3.2.2 Study of the Technologies

In order to make the best possible use of the technologies, it will be spent some time studying them, including the SimSpark simulator, the humanoid robot Nao, and the FCPortugal team's agent. It's very important to make a detailed study on FCPortugal's agent because the work and concepts developed in the scope of this dissertation will be tested using the agent as use case.

3.2.3 Classification and Test of the Behaviours

This task consists of a development module in which will be designed and implemented a system capable of classifying the behaviours performed by the humanoid. This will require a good knowledge of the behaviours generators developed by FCPortugal, and also about classification techniques. The developed component should be able to give the agent a way of reasoning about the performance of its behaviours with regard to different situations.

3.2.4 Interconnection and Sequencing of Behaviours

The resulting work of this module will be integrated with the developed system. It will focus on the application of optimization and search techniques, allowing the agent to combine different known behaviours in order to perform its task with better performance.

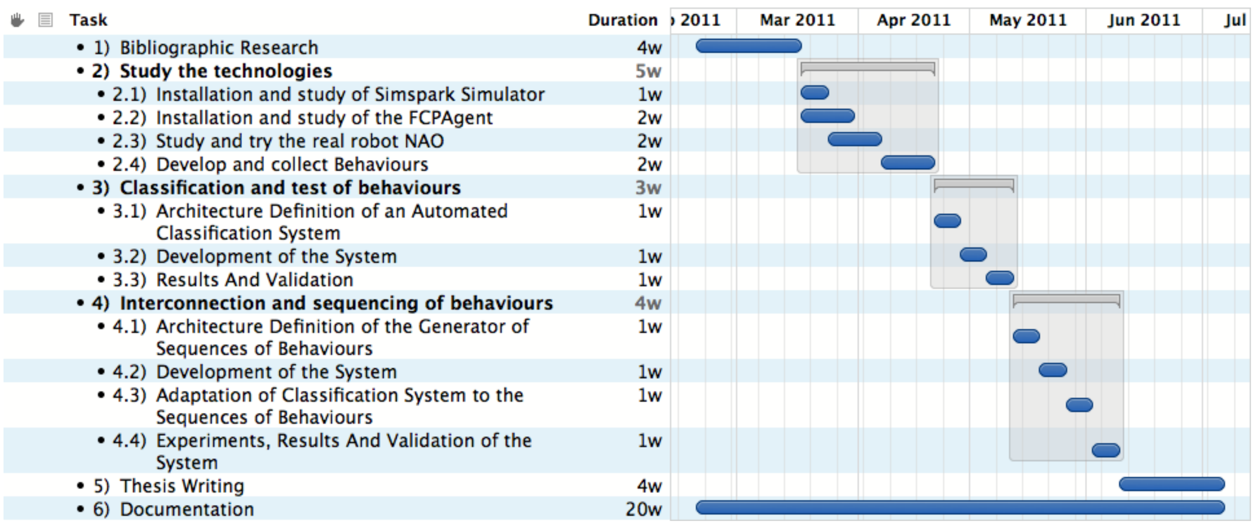
Since the environment in robotic soccer is always changing it's quite hard to find the best sequence of behaviours for a certain situation. It's not reasonable having an agent stalled in the middle of a game processing what is going to do next, so it's also required to give the sequence in real-time.

3.2.5 Thesis Writing

The final 4 weeks of the project will be dedicated to the thesis writing, which will report all the work, experiments and results made during the dissertation. The documentation written along the previous weeks will be useful in this stage.

3.2.6 Task Schedule

The tasks described in the previous sections will be performed according with the proposed schedule in figure 3.2.



Task	Duration
• 1) Bibliographic Research	4w
• **2) Study the technologies**	5w
• 2.1) Installation and study of Simspark Simulator	1w
• 2.2) Installation and study of the FCPAgent	2w
• 2.3) Study and try the real robot NAO	2w
• 2.4) Develop and collect Behaviours	2w
• **3) Classification and test of behaviours**	3w
• 3.1) Architecture Definition of an Automated Classification System	1w
• 3.2) Development of the System	1w
• 3.3) Results And Validation	1w
• **4) Interconnection and sequencing of behaviours**	4w
• 4.1) Architecture Definition of the Generator of Sequences of Behaviours	1w
• 4.2) Development of the System	1w
• 4.3) Adaptation of Classification System to the Sequences of Behaviours	1w
• 4.4) Experiments, Results And Validation of the System	1w
• 5) Thesis Writing	4w
• 6) Documentation	20w

Figure 3.2: Gantt chart of the proposed task scheduling throughout the twenty weeks of dissertation

Chapter 4

Conclusions and Future Work

This document has presented the first version of the state of the art of this dissertation. This existing research provides a solid base for future work on this subject.

Nevertheless, there are still some topics that may deserve special focus. It's the case of search methods, which might need a more detailed study. Besides, in addition to the optimization methods presented in 2.4, there is also a very popular method, *Particle Swarm Optimization*, that might be interesting in the scope of this problem. A more detailed research about machine learning methods should also be very useful, contemplating topics such as *Markov Decision Process*, *Q-Learning*.

These improvements will be performed in the very beginning of this dissertation, as estimated in the work plan.

References

[BDR+10] Joschka Boedecker, Klaus Dorer, Markus Rollmann, Yuan Xu, Feng Xue, Marian Buchta, and Hedayat Vatankhah. *SimSpark User's Manual*, 1.2 edition, January 2010.

[CFH+02] Mao Chen, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *RoboCup Soccer Server: Users Manual*. The RoboCup Federation, 2002.

[CIR99] CIRL. Planning and scheduling materials. The Computational Intelligence Research Laboratory of the University of Oregon, available at: http://www.cirl.uoregon.edu/research/overview.html, 1999.

[DA89] Anton Dekkers and Emile Aarts. Global optimization and simulted annealing. *Mathematical Programming*, 50(1-3):367–393, June 1989.

[FL03] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20(1):61–124, 2003.

[Gep04] L. Geppert. Qrio, the robot that could. *Spectrum, IEEE*, 41(5):34–37, May 2004.

[GHB+08] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. The nao humanoid: a combination of performance and affordability. *Arxiv preprint*, 2008.

[GHK+98] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. *PDDL — The Planning Domain Definition Language*. AIPS-98 Planning Competition Committee, 1.2 edition, October 1998.

[GHP+97] Perry Gray, William Hart, Laura Painton, Cindy Philips, Mike Trahan, and John Wagner. A survey of global optimization methods. URL: http://www.cs.sandia.gov/opt/survey/main.html, Sandia National Laboratories, March 1997.

[Gui04] Guiness. *Guinness World Records 2005*. Guiness, 50th annv edition, August 2004.

[Hes02] Arik Hesseldahl. Ten o'clock tech - say hello to asimo. *Forbes.com*, 2002.

[HH04] Randy L. Haupt and Sue Ellen Haupt. *Pratical Genetic Algorithms*. Wiley - Interscience, second edition, 2004.

[Hon07] Honda. Asimo technical information, September 2007.

[Jac07] J. Jackson. Microsoft robotics studio: A technical introduction. *Robotics Automation Magazine, IEEE*, 14(4):82–87, dec. 2007.

[JM03] O.C. Jenkins and M.J. Mataric. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 225–232. ACM, 2003.

[KHK$^+$08] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi. Humanoid robot hrp-3. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2471–2478. IEEE, 2008.

[Kon00] A. Konar. *Artificial Intelligence and Soft Computing Behavioral and Cognitive Modeling of the Human Brain*. CRC Press, 2000.

[LR08] Tim Laue and Thomas Röfer. Simrobot - development and applications. In Workshop Proceedings of SIMPAR 2008, November 2008.

[mac] Machine learning interaction research in the computer science department at carnegie mellon. http://www.csd.cs.cmu.edu/research/areas/machinelearning/, Consulted at February, 2011.

[MCDB$^+$10] GA Medrano-Cerda, H. Dallali, M. Brown, NG Tsagarakis, and DG Caldwell. Modelling and simulation of the locomotion of humanoid robots. In *UK Automatic Control Conference, Coventry, 7-10 September*, 2010.

[Mit98] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, February 1998.

[Mit06] T.M. Mitchell. *The Discipline of Machine Learning*. Carnegie Mellon University, School of Computer Science, Machine Learning Dept., 2006.

[MSV$^+$08] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The icub humanoid robot: an open platform for research in embodied cognition (special session on eu-projects). In *PerMIS '08 Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, pages 50–56, 2008.

[OTI$^+$00] K. Ogawara, J. Takamatsu, S. Iba, T. Tanuki, H. Kimura, and K. Ikeuchi. Acquiring hand-action models in task and behavior levels by a learning robot through observing human demonstrations. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. Citeseer, 2000.

[Pic08] Hugo Picado. Development of behaviors for a simulated humanoid robot. Master's thesis, Universidade de Aveiro, 2008.

[Rai08] R. Steven Rainwater. Ping pong playing robots. *Robots.net*, June 2008.

[Rei10] Luis Rei. Optimizing simulated humanoid robot skills. Master's thesis, FEUP - Faculdade de Engenharia da Universidade do Porto, 2010.

[RI06] L. Righetti and A.J. Ijspeert. Programmable central pattern generators: an application to biped locomotion control. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1585–1590. IEEE, May 2006.

[RL10] Luis Paulo Reis and Nuno Lau. Paper contributions, results and work for the simulation community of fcportugal. 2010.

[RLM$^+$09] T. Röfer, T. Laue, K. Müller, O. Bösche, A. Burchardt, E. Damrose, K. Gillmann, C. Graf, T. J. Haas, A. Härtl, A. Rieskamp, A. Schreck, I. Sieverdingbeck, and J. Worch. B-human team report and code release 2009. November 2009.

[RN10] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.

[Smi09] M. K. Smith. Learning theory. *Encyclopedia of informal education, http://www.infed.org/biblio/b-learn.htm*, September 2009.

[SWA$^+$02] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent asimo: System overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478–2483. IEEE, 2002.

[Wei09] Thomas Weise. Global optimization algorithms – theory and application. *URL: http://www.it-weise.de, Abrufdatum*, 2009.

[WF11] Derek Wadsworth and Doug Few. Humanoid robotics, November 2011. URL:https://inlportal.inl.gov/portal/server.pt?open=512&objID=536&parentname=CommunityPage&parentid=7&mode=2&in_hi_userid=3338&cached=true.

[WKoT99] R.A. Wilson, F.C. Keil, and Massachusetts Institute of Technology. *The MIT encyclopedia of the cognitive sciences*. MIT Press, 1999.

[WW04] M. Wetter and J. Wright. A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment*, 39(8):989–999, August 2004.