A Framework for Learning in Humanoid Simulated Robots

Esther Luna Colombini[1], Alexandre da Silva Simões[2], Antônio Cesar Germano Martins[2], and Jackson Paul Matsuura[1]

[1] Itandroids Research Group
Technological Institute of Aeronautics (ITA), Brazil
[2] Automation and Integrated Systems Group (GASI)
São Paulo State University (UNESP), Brazil
{esther,jackson}@ita.br,
{assimoes,amartins}@sorocaba.unesp.br

Abstract. One of the most important characteristics of intelligent activity is the ability to change behaviour according to many forms of feedback. Through learning an agent can interact with its environment to improve its performance over time. However, most of the techniques known that involves learning are time expensive, i.e., once the agent is supposed to learn over time by experimentation, the task has to be executed many times. Hence, high fidelity simulators can save a lot of time. In this context, this paper describes the framework designed to allow a team of real *RoboNova-I* humanoids robots to be simulated under *USARSim* environment. Details about the complete process of modeling and programming the robot are given, as well as the learning methodology proposed to improve robot's performance. Due to the use of a high fidelity model, the learning algorithms can be widely explored in simulation before adapted to real robots.

1 Introduction

In recent years, there has been much discussion concerning how knowledge can be acquired and used by autonomous agents. Through learning an agent can interact with an unknown environment and improve its performance over time by focusing its sensors on parts of the environment that are relevant to the task at hand.

In this scenario, the RoboCup® has created in the last years a set of realistic and simulated leagues to stimulate developments in the robotic field. One of these leagues is the *Humanoid League*, where autonomous mobile robots with a human-like appearance play soccer against each other. Humanoid League rules follow FIFA soccer laws in general lines. However, currently, some simplifications are assumed. Differently from conventional soccer, for example, each team consists of two players, in which one can be designated as a goalkeeper.

Another recently created league is the RoboCup *Rescue Simulation Virtual Robots*, in which a team of heterogeneous robots is asked to look for victims in

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 345–352, 2008.

a urban search and rescue (USAR) task. This category also aims to fill the gap between real and simulated environments by using a high fidelity simulator, the USARSim [1], recently extended by the work of Zaratti et al. [2] to work with legged robots. It is also an important research tool for studies of learning, Human Robot Interaction (HRI) and multi-robot coordination, given the possibility of simulating commercial and self-developed robot platforms.

Recently, some simulated models of real legged robots have been proposed. The Sony Aibo and Sony QRIO [2] are some of these models. One of the main constraints for using these robots as the basis for researching learning relies in the fact that they are no longer commercially available. In fact, there is a commercial platform that has been modelled for USARSim, Robovie-M [3]. However, its model is not yet fully available.

This paper presents the details about the complete process of modeling and programming the commercially available version of *Robonova-I* humanoid robot, as well as the learning methodology proposed to improve its performance on the Humanoid league task. The rest of this paper is structured as follows. Section 2 explains the main characteristics of autonomous Learning. Section 3 presents the proposed approach for building the robot model, describing, in details, the complete high fidelity geometric model of the robot and the set of script files needed to configure this model in the USARSim RoboCup simulator. The learning framework is presented in Section 4. Finally, Section 5 summarizes with the main conclusions and presents some lines for further work.

2 Learning

Reinforcement Learning (RL) [4] is a class that lies between the extremes of supervised learning, where the policy is taught by an expert, and unsupervised learning, where there is no evaluative feedback. It is a technique that allows an agent to adapt to its environment through the development of an action policy, which determines the action that should be taken in each environmental state in order to maximize (or minimize) a function over a cumulative reinforcement. The reinforcement is a real value that defines the desirability of a state and can be expressed both in terms of rewards or punishments. In RL systems, the a priori domain knowledge incorporated by the designer is minimal and is mostly encapsulated in the reinforcement function.

Q-learning [5] is the preferred RL algorithm because it provides good experimental results in terms of learning speed and it is a model-free learning for optimal policies. It learns the values of all actions in all states, rather than only representing the policy.

3 Proposed Approach

The use of Learning, more specifically RL, is wide spread on RoboCup. In the development of the simulated robots' plan, DAMAS Rescue team used Jack

Intelligent Agent programming language [6], decision tree algorithms and reinforcement learning [7]. Moreover, F180 champions CMUDragons are known to use RL techniques. Furthermore, Soccer Simulation 3D champion FC-Portugal can be cited as another successful example [8].

However, most of the techniques known that involves RL are time expensive, i.e., it takes time to find a policy to successfully accomplish the proposed task and difficult to configure. In these cases, as the agent is supposed to learn over time by experimentation, the task has to be executed many times. Hence, high fidelity simulators can save a lot of time.

In our case, the study of real robots in a simulated environment only makes sense if the resultant study can be sent back to the real robot. For this purpose, the construction of the *Robonova-I* humanoid robot simulated model in USARSim is proposed.

4 Bulding Robonova-I Model

The construction of a robot model in the USARSim environment is a very complex process. Since documentation for this task is extremely rare, we will detail in the following sections the construction of the *Robonova-I* model. In this approach, two main steps were adopted: *i)* robot geometric model construction and *ii)* robot scripting.

4.1 Geometric Model

The construction of the geometric model of the robot makes necessary the steps mentioned next.

Creating the static meshes. In our approach a tridimentional model of the robot was made in a CAD (Computer Aided Design) environment. We adopted the *AutoCad® 2007* software, which provides a rich set of 3D creation and management tools, necessary to reproduce the complex forms of the robot.

One important remark must be done with respect to the XYZ coordinate system. Autocad environment is well known to use the XYZ positive axis arranged according to the LHR (left hand rule). The final assembly of the robot in the Unreal engine is assumed to arrange XYZ axis according to the RHR (right hand rule). In order to convert between systems, the orientation of the X axis must be changed. This situation forces the feet of our CAD robot model to be constructed above the XY plane, with the robot front oriented to the -X axis.

Accomplishing the first step of the process, a high fidelity model of the Robonova-I robot was generated in the CAD environment using exclusively static meshes. It is also important to remark that other kinds of primitives (like surfaces or regions) are not recognized in Unreal engine. Each robot material (in this case golden metal, black plastic and servomotors plastic) was represented in a different layer. The complete robot drawing was splited, and one new drawing was created to each rigid part of the robot (without joints). The complete robot model and its parts are shown in Figure 1.

Fig. 1. Geometrical model of the Robonova-I robot. a) Assembled robot; b) Exploded main parts of the robot.

Converting the static meshes. In next step, the static meshes must be converted to its preferred file format to be imported in USARSim. The ASE (ASCII Scene Exporter) file format is the standard, since it stores identifiers for all file objects materials. In our approach, we adopted the software *3D Studio MAX® 8* to realize this task. After a CAD file was imported, three different materials (simple color patterns) were created using the material editor tool. One material was assigned to each layer of the original CAD drawing and, so, all static meshes were attached together forming a single body composed by different materials. The body was so rendered to texture, in order to generate a texture map. The texture properly was discarded, and the file was exported to the ASE format. This process was repeated to each one of the rigid parts of the robot.

Creating textures. In order to allow the use of simple textures in the robot into the USARSim environment, in proposed approach three standard 256x256 Bitmap files with 8 colors depth were created in the *paint* software and filled with the three different colors of the textures.

Assigning textures. In the next step, the textures must be assigned to the materials specified in the ASE file. This process was realized inside of the *Unreal Editor 2004*. First, all three textures were imported and a unique UTX texture package was created. After this, each of the static meshes of the rigid parts of the robot were imported into a unique USX package. Still using the Unreal Editor, each material of each component of the meshes in the USX package was linked to one of the textures in the UTX package. The USX package now stores all information about robot geometry, except the information concerning the position to correctly assemble this parts, which will be informed in the robots configuration script.

4.2 Robot Configuration

After preparing the robot parts geometric model and textures, it is necessary to add these new models to the USARSim file structure. However, the robot

Table 1. Robonova-I parts and parameter values: weight, rotation angle, static friction rupture with robot up and down

Robot part	Quantity	Weight	Rotation	Up SFR	Down SFR
				-	-
Head	1	27g		-	-
Chest	1	337	-	-	-
Hand	2	65g	180^0	-	-
Elbow	2	65g	180^0	-	-
Shoulder	2	6g	360^0	-	-
Thigh	2	23g	90^0	-	-
Knee	2	135g	90^0	-	-
Superior ankle	2	44g	180^0	-	-
Inferior ankle 2	2	23g	180^0	-	-
Foot	2	83g	90^0	-	-
Spins	-	8g	-	-	-
Robonova-I	-	1.260g	-	260 Kgf	600Kgf

Fig. 2. Frontal view (out of scale)

Fig. 3. Back view (out of scale)

physical parameters and dynamics still have to be configured. In this phase, scripts written in the Unreal Script language are prepared for each part of the robot, as well as for the complete robot model. For the individual parts, parameters such as torque, mass, angular velocity, friction, restitution, etc., are described. These parameters are used by the Karma engine [9] that is responsible for modeling the USARSim system dynamics. As for the complete robot model script, it contains the static meshes assembling and relative movements (i.e. axis spin) information.

To keep the fidelity of the model, some experiments were carried out with the real *Robonova-I* in order to obtain some of the Karma parameters. Based on the data of these experiments, well-known physical constants and robot geometry, one can estimate static and dynamic friction, maximum and minimum joint aperture, motor torque, etc. Some of the acquired values are shown in table 1.

Finally, scripts were compiled in order to generate the robot model into the USARSim environment. The final robot model built and imported in the virtual environment (out of scale) is presented in Figures 2 and 3.

5 *Robonova-I* Learning Framework

The design of architectures composed of very simple skills is not easy, nor is the learning of its sequence, as producing an adequate combination of these behaviours is not straight-forward. Furthermore, the controller decomposition introduces the need for determining when to trigger control, i.e. when to re-evaluate the previously selected behaviour and choose a new one.

Considering these constraints, the framework described proposes the introduction of learning in two levels. In the first level, the information provided by sensors (gyroscopes, camera and pressure sensors) is used to build the controllers responsible for movements. These basic controllers represent the set of sequential servo commands that a robot may perform to execute a movement. We can divide the controllers in four main groups: *i)* Walking Controllers; *ii)* Precise Positioning Controllers; *iii)* Special Actions Controllers; and *iv)* Goalkeeper Controllers.

In the *Walking Controllers* the three main controllers are the shift-right, the shift-left and the forward walk controllers. As the names propose, they are responsible for the shift-sideways movements and for the forward walk movement of the robot. Other Walking Controllers are the backward walk, the diagonal walks (forward left, forward right, backward left and backward right), the turns (left and right) and the forward run.

In the *Precise Positioning Controllers* there are just three controllers that are smaller and more precise versions of the three main Walking Controllers. They are step-right, step-left and step-forward. The steps are small movements sideways or forward executed to allow a precise positioning of the robot.

There are four *Special Actions Controllers*, two of them responsible for the interaction with the ball, the kick right and the kick left controllers, used to kick the ball with the right leg and with the left leg respectively. The other two are the stand-up controller and the bend controller. The stand-up controller is used when the robot falls to get back to the upright condition while the bend controller is used to allow the camera to track objects near the feet of the robot.

Finally, the *Goalkeeper Controllers* are specific actions for the goalkeeper, such as: defend-right, defend-left and defend-mid. Each of them is used to defend a ball kicked to the right, left or in the direction of the robot respectively.

As for the second level, once defined the basic controllers, it is possible to apply the learning approach to automate the process of choosing a controller to execute in a specific environment situation. Figure 4a presents an overview of the proposed system architecture.

5.1 State and Action Space Modeling

For using RL algorithms, one has to guarantee that the problem can be modelled as a MDP (Markov Decision Process), i.e, the problem has to be represented as a finite set of actions and states and a discrete time model where the states should be available for measurement. However, real robot tasks have infinite state and action spaces, continuous time and due to sensorial limitation are not always

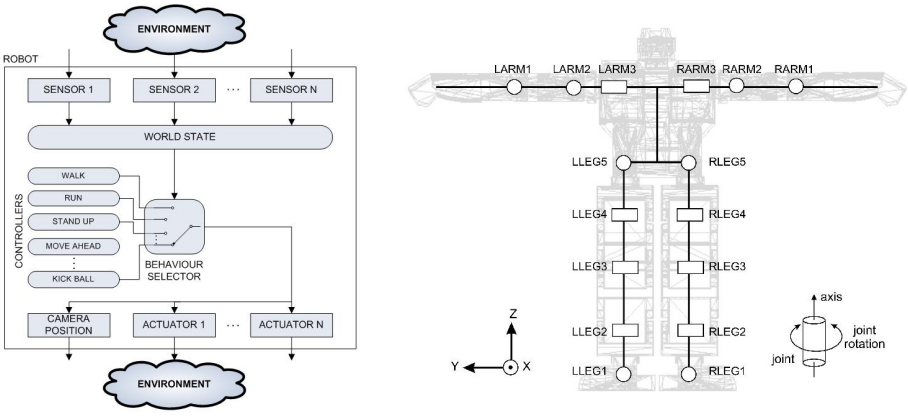

Fig. 4. a) System Architecture. b) Joints position and spin.

measurable. To deal with this problem, a discretization model for the state and action spaces is proposed.

First, consider the joints robot structure presented in Figure 4b. For each arm there are three joints with one DOF each, while for each legs these number reaches five, what give us sixteen DOFs. The state and action space discretizations are divided into two groups: 1) for low level learning or controllers learning and 2) for high level learning or switching controller determination.

Low level discretization. The low level action space is composed of sixteen elements, each representing a servo. Each action corresponds to change a servo angle by adding or subtracting 15 degrees to its actual state.

If the state vector, defined by the gyroscopes, camera and pressure sensors values indicate a falling, the robot is assigned with a null value reward and for each action performed, it receives an unitary reinforce. We work in all cases with a minimization criterium and with a step corresponding to a change in the servos configuration. In these cases, the goal is to perform the movements without much changing in the robot servos and without falling down.

High level discretization. For the high level learning, after fine tuning the individual controllers, one can apply the high level learning, using the same state vector defined for the low level phase, to decide which basic controller to execute. The reward structure considers that each action performed costs a unit to the learning agent, while accomplishing the goal gives it a null reinforcement value. This criterium can help the robot to save battery. To help configuring the learning parameters, a graphical interface was implemented.

6 Conclusion

This paper presented a complete framework for a *Robonova-I* humanoid robot, composed by: *i)* A complete high fidelity geometric model of the robot; *ii)* A set

of scritp files to configure this model in the USARSim RoboCup simulator; *iii)* An architecture model to be used in robot learning, and *iv)* A graphical interface for learning parameters settings.

At the best of our known, this set of features represents the first available framework of a commercially available humanoid robot. In this way, this framework is expected to work as an important tool in robots dynamics research and also to contribute to reduce time required to test learning algorithms.

The paper also presented a detailed description of the robot modeling and configuration process for USARSim environment, filling some gaps in the related technical literature and expecting to reduce the amont of time required to create new robots and models in this environment.

As main ongoing works, there is a set of experiments to establish the confidence degree between proposed model and real Robonova-I robot with respect to dynamics, sense and acts. As future work, we point the implementation of a large number of RL algorithms in order to extend the framework capabilities.

References

1. Wang, J.: USARSim V2.0.2 Manual: A Game-based Simulation of the NIST Reference Arenas (2006)
2. Zaratti, M., Fratarcangeli, M., Iocchi, L.: A 3d simulator of multiple legged robots based on usarsim. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 13–24. Springer, Heidelberg (2006)
3. Greggio, N., Silvestri, G., Antonello, S., Menegatti, E., Pagello, E.: A 3d model of a humanoid for usarsim simulator. In: First Workshop on Humanoid Soccer Robots, Genova, pp. 17–24 (2006)
4. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, USA (1998)
5. Watkins, C.: Learning from delayed rewards. PhD thesis, King's College (1998)
6. Howden, N., Renquist, R.: Jack intelligent agents - summary of an agent infrastructure. In: 5th Int. Conf. on Autonomous Agents, Montreal, Canada (2001)
7. McCallum, A.K.: Reinforcement Learning with Selective Perception and Hidden State. PhD thesis, University of Rochester, New York (1996)
8. Reis, L.P., Lau, N.: Fc portugal team description: Robocup 2000 simulation league champion. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 29–40. Springer, Heidelberg (2001)
9. The Unreal Engine Site (2007), `http://wiki.beyondunreal.com/wiki/Karma`