SimSpark: An Open Source Robot Simulator Developed by the RoboCup Community

Yuan Xu[1] and Hedayat Vatankhah[2]

[1] DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7, Berlin
D-10587 Germany
yuan.xu@dai-labor.de

[2] Department of Computer Engineering and IT
Amirkabir University of Technology
Tehran, Iran
hedayatv@gmail.com

Abstract. SimSpark is an open source robot simulator developed by RoboCup Community. This paper briefly describes the development of SimSpark since 2008. Furthermore, some new features are proposed and implemented for the next RoboCup, including realistic motor, heterogeneous robots, and agent proxies. As a powerful tool to state different multi-robot researches, SimSpark has been successfully used in RoboCup simulation league, standard platform league and humanoid league.

1 Introduction

The development on robots may be severely limited by the constrained resources. This is especially true in the research of multi-robot systems in areas such as RoboCup. Using simulation for algorithm development and testing makes thing easier.

SimSpark, a multi-robot simulator based on the generic components of the Spark[7] physical multi-agent simulation system, has been used in the RoboCup Soccer Simulation League since 2004. The project was registered as open source project in SourceForge in 2004, it has an established code base with development increasing year-over-year. As the result, RoboCup soccer simulations have changed significantly over the years, going from rather abstract agent representations to more and more realistic humanoid robot games[2,3]. Thanks to the flexibility of the Spark system, these transitions were achieved with little changes to the simulator's core architecture.

In this paper we describe the recent development of SimSpark project, which make the SimSpark possible to simulate 11 vs. 11 humanoid robot soccer games in real time. In section 2, we will give an overview of the SimSpark project since 2008. After that, we will describe the development of the Spark simulation platform in section 3 and the implementation of RoboCup 3D Soccer Simulator in section 4. We will introduce some new features for RoboCup 2013 in section 5. Furthermore, we will give the application examples of SimSpark in section 6. Finally, we will outline future development plans in section 7.

2 Project Overview

Until 2008, soccer simulation and SimSpark simulator were developed and released as a single project called rcssserver3d. In late 2008, the SimSpark project migrated from CVS repository at `http://sserver.sourceforge.net` to the new project Subversion repository at `http://simspark.sourceforge.net`. As part of this process, the project was broken to two main projects (Spark simulation platform and RCSSServer3D soccer simulation server) and two auxiliary projects (RSGEdit and simspark-utilities). The separation clarifies SimSpark is a generic simulation environment rather than only a robot soccer simulator.

SimSpark was never a single-platform project, but it practically didn't support Windows. In recent years, Windows support was added after migration from Autotools to CMake, and windows installers are now released.

3 Spark

As a generic simulation environment, Spark provides a rich set of features to create, debug and modify multi-robot simulations. It has three main components, including the simulation engine, the object and memory management system, and the physics engine. Details about architecture and concepts of Spark have been described in [2,7]. In this section, we describe the necessary changes for simulating 11 vs. 11 humanoid robot soccer game. However changes to the simulator core were never specialized for the soccer simulation.

Sensor Plugins Sensors of a robot allows awareness of the robot's state and the environment. Humanoid robots like the NAO usually have many different sensors installed. We have implemented new plugins to simulate sensors of humanoid robots. Some of the sensors delivers information from physics engine, such as joint position, gyro, accelerometer, and force resistance. Furthermore, lines can be sensed by virtual vision. Additionally, a more realistic camera which delivers images rendered via OpenGL hardware accelerated offscreen buffers is implemented. Details of these sensors can be found in [3].

Multi-threads Supporting One great feature of SimSpark is switching between single thread mode and multi-threads mode. The multi-threads mode can improve the performance in computer with multi-cores processor, but the single thread mode is also useful for developing the simulator.

The implementation of multi-threads loop is based on two conditions. First, different tasks are assigned to different *SimControlNode*s in SimSpark. For example, *AgentControl* is a node that manages the communication with agents. For each simulation cycle, *SimControlNode*s are executed one by one in the single thread mode, but they can run in parallel. Second, all data of simulation state is stored in a tree called *active scene*, the physics engine and *SimControlNode*s interact through the *active scene*. As we know, the physics computation is the most time-consuming, and the physics engine does not need to access

the active scene during physics computation. So the physics computation and *SimControlNode*s can run in parallel. Furthermore, the physics engine, e.g. Open Dynamic Engine[9], is modified to run in parallel by using Intel Threading Building Blocks[1].

In RoboCup 2012, the 11 vs. 11 humanoid robot (22 degree of freedom) soccer games were simulated in real time by computer with Intel Core i7-975 @3.33GHz CPU and 4G DDR3 RAM.

4 RCSSServer3D

As the competition environment for the Soccer Simulation 3D League at RoboCup, RCSSServer3D simulates the soccer field where two team of robots play soccer game. In its initial version players were modeled as spheres in a physical three dimensional world. Now it supports humanoid players with articulated bodies.

Soccer Simulation Most rules of the soccer game are judged by an automatic rule set that enforces the basic soccer rule set. However more involved situations like detection unfair behavior still require a human referee. Noticeably, the size of soccer field is increasing every year, since the number of robots in each team is increasing. In RoboCup 2012, each team has 11 robots. This makes coordination between multi-robots more important.

Robot Model The NAO robot is currently used in the competitions of the Soccer Simulation 3D League at RoboCup. Its height is about 57cm and its weight is around 4.5kg. Its biped architecture with 22 degrees of freedom allows NAO to have great mobility. The NAO robot model is also equipped with a powerful selection of the sensors described in section 3, to provide a widespread information base for agent development. There are differences between simulated NAO and real NAO. For example, the hip joints are physically connected by one motor in the real NAO. Furthermore, the simulated robot gets positions of objects via virtual vision sensor, while the real robot has to process images to understand the world. Nevertheless, this NAO model enables RCSSServer3D to be a good humanoid robot research platform.

5 Experimental Features

For getting more realistic simulation, some new features are proposed and implemented, including realistic motor, heterogeneous robots, and agent proxies. These experimental features probably will be used in RoboCup 2013 for the first time.

5.1 Realistic Motor

NAO robot has twenty-one motor joints as its actuators. The simulation engine, i.e. ODE, provides a simple model of real life servos: the motor brings the body

up to speed in one step; and provides force that is not more than is allowed. The simple motor model is one reason for the unrealistic simulation results. In this section, we proposed and implemented a realistic motor model. More details and experimental results are described in [11].

Stiffness The stiffness determines how strong the motor is. In the real robot, this percentage is the maximum electric current applied to the motor. For a DC motor, the electric current, I, determines the output torque, $\tau = K_\tau I$; where K_τ is the torque constant of the motor, and can be found in the specification. The simulation engine can specify the maximum torque of the servo, therefore the stiffness control can be easily implemented by setting the maximum torque of the simulated servo:

$$\tau_{max}(t) = k_s(t) T_{max} \tag{1}$$

where $\tau_{max}(t)$ is the maximum torque set in the simulated servo at time t; $k_s(t)$ is the stiffness at time t; and T_{max} denotes the maximum torque of the servo when stiffness is 1.

Power Consumption Another important aspect of real motor is power consumption. Because the robot is powered by a battery with limited energy, and has to walk during the game. Furthermore, the motor can overheat if it consumes too much energy and becomes too hot. In real robots, the temperature of each motor is measured, and the motor shuts down when the temperature is too high.

DC motors can be modeled by the following equation: $U = U_e + IR = K_e\dot{\theta}$; where U is the voltage of input, U_e is the back electromotive force (EMF), I is the electric current, R is resistance, $\dot{\theta}$ is the speed, and K_e is the speed constant of the motor. The value of R and K_e can be found in the specifications of motor again; and the simulation engine provides the value of τ and $\dot{\theta}$; therefore we can calculate the power consumption:

$$P = UI = U_e I + I^2 R = \frac{K_e}{K_\tau}\dot{\theta}\tau + \frac{R}{K_\tau^2}\tau^2 \tag{2}$$

And the total energy used by the motor is:

$$E = \sum_t P_t \Delta t \tag{3}$$

where Δt is the time step of the simulation, and P_t is the power consumed at time t. For the overall power consumption, the energy consumed by devices other than motors, e.g. main board, CPU, camera, etc. has to be counted.

Temperature Regulation We model the temperature and heat of the motor with the following equations:

$$\Delta Q = \Delta Q^+ + \Delta Q^- = I^2 R \Delta t - \lambda(T - T_e)\Delta t \tag{4}$$

$$\Delta Q = C \Delta T \tag{5}$$

where T is the temperature of the motor, T_e is the environment temperature inside the motor, so it is higher than outside and differs from motor to motor, ΔQ is the heat changing, ΔQ^+ is the heat produced by the motor, ΔQ^- is the heat transferred from the motor to the environment, λ is thermal conductivity which indicates the ability of a motor to conduct heat, and C is the heat capacity of the motor, which can be seen as constant. Finally, the temperature of the motor at time $t + \Delta t$ can be solved as:

$$T_{t+\Delta t} = T_t + \Delta T = T_t + \frac{[I^2R - \lambda(T_t - T_e)]\Delta t}{C} \tag{6}$$

In this model, we need to determine T_e, λ, and C by experiments. A sequence values of Δt, T_t, and I^2R can be measured by experiment, therefore the optimum parameters of eq. (6) is determined.

The whole process of joint simulation is summarized in Fig. 1. When the battery is empty, the maximum torque τ_{max} is set to 0 to turn off the motor.

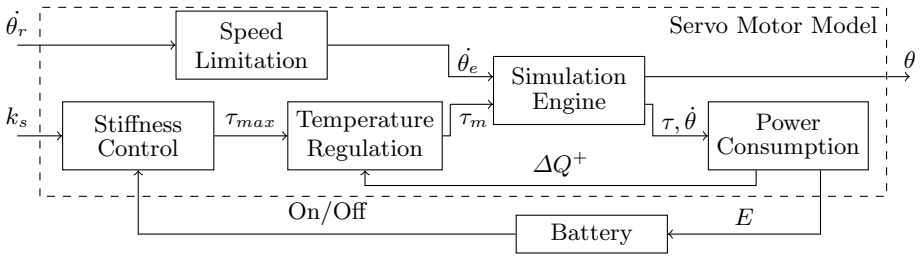

Fig. 1: Pipeline of the servo motor simulation

5.2 Heterogeneous Robots

Heterogeneous systems are emerging as effective solutions to many multi-robot tasks, it is also true for the soccer team. The physical difference between robots are important when they serve diverse roles in a cooperating team. In simulation, the physical properties of the robot can be easily changed, so it is a good platform for researching on heterogeneous systems.

Heterogeneity also creates new challenges for robot AI developers. The robots with different mechanical properties will be generated at running time, different robots are provided to teams, so teams have to adopt their developed behaviors to unknown robots.

SimSpark already provided the ability to define different robot models and use them in its simulation environment. However, each model was fixed and if you wanted to have several variations of a robot, you had to defined several separate robot models. To better support games with heterogeneous robots, we have implemented the parametric robot model. This model has a number

of parameters and can have different values for different robot variations. For example, the length of the legs can be different for different robots. In order to launch this model, only these parameters have to be specified additionally.

5.3 Agent Proxies

Initially, SimSpark used SPADES[8] for managing external agents. It managed the simulation time and the time spend in agents for thinking. Later it was dropped because of its complexities and agents communicated with the server directly. The cycle duration was managed by the server without considering agents and the network latency. Therefore, if agents were unable to deliver their commands to the server in a cycle because of network latency, their commands were executed by the simulator in the next simulation cycle. As the number of soccer players in soccer simulation increased, this problem became more apparent.

To solve this problem, the concept of agent proxies was proposed. Agents communicate with the proxies instead of communicating directly with the simulator. In this model, the cycle time is managed by the proxies on the client side, and they communicate with the simulator in Sync Mode. Therefore, the simulator waits for all proxies to signal the end of a cycle and then proceeds with the next cycle. This model ensures that agents can use the allowed cycle time to think in a cycle when new information is delivered to them.

This model is not perfect because the agents can have some spare time if network latency occurs, but it is more predictable and fairer than the old model. The network latency can be reduced by using binary network protocol between proxies and simulator. Nevertheless, no new information is arrived to agents in that spare time and they cannot send new commands for that cycle.

Currently, an initial version of agent proxies is under development outside SimSpark project using Java.

6 Applications

In addition to be used in RoboCup Soccer Simulation 3D League, SimSpark is gaining popularity in other leagues, including Standard Platform League and Humanoid League. It is also widely used to teach artificial intelligence and robotic lectures.

RoboCup Soccer Simulation 3D League Since 2004, SimSpark has been used as a official competitions environment. Research teams have developed useful research tools based on SimSpark. For example, RoboViz[10] is designed to assess and develop agent behaviors in SimSpark. And these tools are also released in open source. Combined with these tools, SimSpark becomes a featured platform to develop and test new algorithms for multi-robot systems.

RoboCup Soccer Standard Platform League The special situation between Standard Platform League and 3D Simulation League is that both leagues use the same robot model — NAO from Aldebaran. So it appears to be natural to reuse the work which has already been done. Nao Team Humboldt developed a software architecture[6] which enables their control software can run both in real NAO and simulated NAO with SimSpark. This helps them to participate in both Simulation League and Standard Platform League, and achieve some good results. Furthermore, they also promotes the usage of SimSpark in the Standard Platform League by implementing its rules, see Fig. 2.

Fig. 2: Prototype of the extended SimSpark for Standard Platform League. The bottom of screen are images of robot cameras.

RoboCup Soccer Humanoid League Of course the simulator can also be extended for other leagues by adding new robot models. For example, in RoboCup Humanoid Kid Size League, FUmanoid[4] uses SimSpark to perform multi-level testing methods for archiving higher quality in each module of their robot control software and unlink the module test from the robotic hardware.

7 Conclusion and Future Work

SimSpark is a powerful tool to state different multi-robot researches. The introduction of a humanoid robot model increase the realism of SimSpark, and gains popularity in research teams who have real robots. The increased number of players per team makes research slowly get back to the high level behaviors based on solid low level behavior for realistic humanoid robot teams.

SimSpark has undergone continuous development driven by the requirement of continuous research in multi-robot system. The following extentions are planned:

Integration With Existing Robotic Frameworks To enhance the usability of SimSpark for wider usage, and also to make it easier to develop and test their robotic software on SimSpark and to use the same code on real hardware, we aim to integrate SimSpark to an existing robotic framework, such as ROS or Player.

Physics Abstraction Layer Relying on a single physics engine, i.e. ODE, hampers Simspark's flexibility. Thus, the physics abstraction layer has be developed[5], ODE and Bullet can be used as plugins of SimSpark. However, current implementation has many drawbacks and difficult to maintenance.

Robot Model Importers The usage of SimSpark will be extended when it has more robot models, however creating robot models is a time consuming task. The idea is to create model importers for different model format, so SimSpark will be able to use the robot model which is available in other simulators.

Acknowledgments

We wish to thank other members in the Maintenance Committee of the RoboCup Soccer Simulation League, especially the original authors of SimSpark: M. Kögler, M. Rollmann, and O. Obst. Furthermore, thanks go to J. Boedecker for supporting us to work in the SimSpark project.

References

1. Intel® threading building blocks, `http://threadingbuildingblocks.org/`
2. Boedecker, J., Asada, M.: Simspark – concepts and application in the robocup 3d soccer simulation league. In: Proceedings of the SIMPAR-2008 Workshop on The Universe of RoboCup Simulators. Venice(Italy) (November 2008)
3. Boedecker, J., Dorer, K., Rollmann, M., Xu, Y., Xue, F., Buchta, M., Vatankhah, H., Glaser, S.: SimSpark User's Manual, 1.3 edn. (August 2010)
4. Donat, H.: Evaluation of Simulators for Humanoid Soccer Playing Robots and Integration in an Existing System. Bachelor thesis, Department of Computer Science, Freien Universität Berlin (2012)
5. Held, A.: Creating an abstract physics layer for simspark. Tech. rep., University of Koblenz-Landau (2010)
6. Mellmann, H., Xu, Y., Krause, T., Holzhauer, F.: Naoth software architecture for an autonomous agent. In: International Workshop on Standards and Common Platforms for Robotics (SCPR 2010). Darmstadt (November 2010)
7. Obst, O., Rollmann, M.: SPARK – A Generic Simulator for Physical Multiagent Simulations. Computer Systems Science and Engineering 20(5), 347–356 (Sep 2005)
8. Riley, P.: Spades: a system for parallel-agent, discrete-event simulation. AI Magazine 24(2), 41 (2003)
9. Smith, R.: Open Dynamic Engine User Guide (2006), `http://www.ode.org`, last visited at 07.02.2013
10. Stoecker, J., Visser, U.: Roboviz: Programmable visualization for simulated soccer. In: RoboCup 2011: Robot Soccer World Cup XV (2012)
11. Xu, Y.: From Simulation to Reality: Migration of Humanoid Robot Control. Ph.D. thesis, Humboldt-Universität zu Berlin (2012)