On Standardization in the RoboCup Soccer Humanoid Leagues

Norbert Michael Mayer [#1], Joschka Boedecker [*2], Minoru Asada [#*3]

[#]*Asada Synergistic Intelligence Project, ERATO JST,*
2-1 Yamada-oka, Suita, Osaka 565-0871, Japan
[1]michael@jeap.org

[*]*Department of Adaptive Machine Systems,*
Graduate School of Engineering, Osaka University,
2-1 Yamada-oka, Suita, Osaka 565-0871, Japan
[2]joschka.boedecker@ams.eng.osaka-u.ac.jp
[3]asada@ams.eng.osaka-u.ac.jp

***Abstract*— From 2008 on, the RoboCup competitions are going to host four distinct humanoid robot soccer leagues: the Standard Platform League (using the Nao robot of Aldebaran Robotics), the KidSize League (custom robots smaller than 60cm), the 3D Soccer Simulation League, and the TeenSize League (custom robots larger than 100cm). Currently, each of these leagues defines their own rule framework and research agenda, and standards are lacking that would facilitate collaborations across the individual leagues. As a consequence, solutions from one league are difficult to compare, knowledge is not easily transferred, and significant amounts of work are duplicated. This binds resources and slows down the technological and scientific progress. We therefore propose common control interfaces, and a common simulation platform across these humanoid robot soccer leagues. We discuss opportunities and difficulties for such a standardization process for the different leagues, and suggest candidates for solutions. We believe that by implementing these standards successful collaborations will be facilitated, and the overall progress towards the ultimate goal of the RoboCup project accelerated.**

I. Introduction

The stated goal of the RoboCup initiative is by 2050 having a team of humanoid robots win the game of soccer against the human champion of that year [1], [2]. To achieve necessary steps towards this goal, several different leagues have been part of RoboCup, each with characteristic hardware and software platforms, and different research agendas. It took until 2002, however, to introduce the first league sporting actual humanoid robots in the RoboCup competitions at Fukuoka, Japan. While the technology used in these early humanoid soccer games was certainly very different from the what we will see in the 2050 game, the physical shape of these robots approached the envisioned robots for the first time. This new league, called simply Humanoid League (SocHL) has been growing with each year, and the technology, the rule set, and the quality of the games improved at a fast pace.

In 2007, another humanoid soccer competition was introduced at the RoboCup competitions in Atlanta, USA. This league grew out of the Soccer Simulation League (SocSimL) and consequently, the robots are entirely simulated in software. The ground work for this league was laid in 2004 already with the development of a simulator for physical three-dimensional multiagent simulations [3] based on the generic components of the SPARK simulation framework [4], but humanoid robot models were integrated and used for the first time in 2007.

The RoboCup competitions in 2008 will see the introduction of a third league using humanoid robots (figure 1 shows a typical robot of each of the three leagues). This league will be called the Standard Platform League (SocSPL), since all of the participating teams are competing with the same type of hardware, namely the humanoid robot Nao from Aldebaran Robotics [5]. The focus in this league is on software development while still using state of the art robot systems. Like in the former 4-Legged League, which it will replace, and the SocSimL, the use of a fixed platform facilitates comparisons of different software solutions, as well as sharing code that has proven successful in the competitions. These are likely to be some of the factors for the rapid progress in the quality of the games of these leagues, and for the considerable number of contributions of their participants to the RoboCup Symposium.

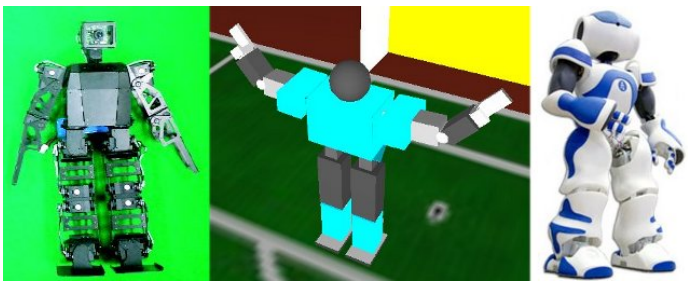

Fig. 1. Typical robots as used in the Humanoid League KidSize (left), the 3D Soccer Simulation League (middle), and the Standard Platform League (right).

It would be desirable to have similar effects also across different leagues within RoboCup, especially with three leagues now sharing similar robot morphologies. So far, however, collaborations have been rather sparse, and substantial amounts of work were duplicated. Currently, for instance, basically every team in the hardware leagues develops their own simulation environment, often with considerable effort. These resources

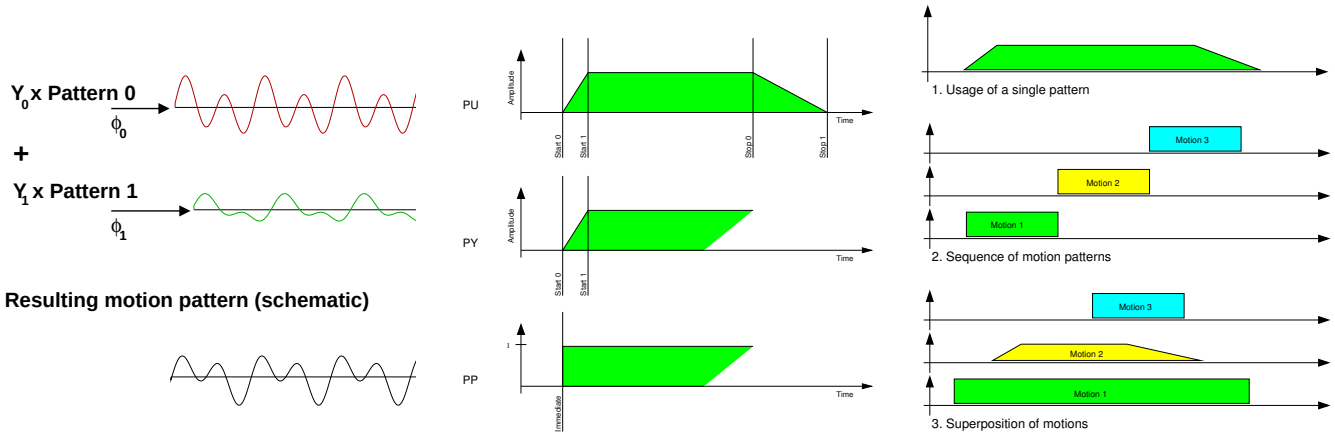

Fig. 2. Left: Motion superposition: By using HMDP two or more motions can be superposed by defining the amplitudes Y_i and the phase shift ϕ_i. The resulting motion pattern is the sum of both initial patterns. Middle: HMDP commands for different usages of the motion patterns. The PU command can trigger a motion pattern with end. PY triggers the motion pattern without giving a point in time for the end. The PP command is for testing and starts a motion pattern immediately without end. Right: Examples for possible implementations of motion patterns. The third example is making use of the superposition principle.

could be freed if a standard simulation platform was available that could be used by all of the participants of different leagues.

As a first step, however, it would be important to develop standards for robot sensing and control. The harmonic motion description protocol (HMDP) [6], [7] which we present in the next section could serve this purpose. After the HMDP description, we review some of the concepts of the SPARK simulator in section III, highlighting the features that make it suitable as a standard simulation platform. Next, the 3D2Real project is presented in section IV before we close with a discussion of the problems and opportunities of our proposal in section V.

II. Example: HMDP as a Control Protocol

In the following we outline the Harmonic Motion Description Protocol (HMDP) [6] that could serve as an interface in the 3D2Real project. The initial motivation of the HMDP has been to introduce a flexible communication program on the microcontrollers of the robots of RoboCup SocHL KidSize Team JEAP (the team of the JST ERATO Asada project). The so-called ”motion machine” program that takes the role of a real-time OS on the microcontroller side (tested on an ARM 4 controller) is self-contained in the sense that it does not require any external libraries. Even a native float type is not necessary, since an emulated float is provided by the software. In comparison to present day micro-controller specifications the demand on RAM memory is relatively high though.

The HMDP specifications are:

- The HMDP includes messages that are submitted from the PC (acting as master) to the micro-controller (acting as slave) and response messages from the micro-controller to the PC.
- The protocol allows for the PC side to set the current time as an integer and also to set the maximal time value after which the current time on the micro-controller is set to zero again.
- The protocol defines motion patterns in terms of splines. In order to allow for periodic motion patterns that can be repeated an arbitrary number of times the set of base functions is defined as a set of sines and cosines.
- The protocol activates motion patterns including the information at what time the motion pattern is activated, and its amplitude. It also defines which step of the motion pattern is assigned to what time step of the motion controller (motion phase assignment).
- The design of the HMDP includes the management of the motion patterns on the micro-controller side. It is possible to activate several motion patterns at the same time. The resulting motion pattern is the superposition (see fig. 2) of all activated motion patterns (motion superposition principle).
- The protocol allows to read out values of sensors that are connected to the micro-controller. In particular, it allows to read out the the angle of the servo positions at a particular time step. The message for a sensor request consists of a tuple of a time at which the sensor value should be read out and the name of the particular sensor. As soon as the time for read out is reached, the time value, the sensor name, and the sensor value are sent from the micro-controller to the PC.

In addition the HMDP syntax was designed to be relatively easy to parse. Below we describe the set of messages and the way parameters and numbers are defined in the protocol.

The HMDP has been used for the Team JEAP robots during the RoboCup 2007 competitions, and proved to be practically applicable there (post competition analysis showed that the unsatisfying results in the competition were caused by other components.). Figure 3 shows the a part of the design tool where the standing up motion as it has been used during the RoboCup has been loaded. The standing up motion proved

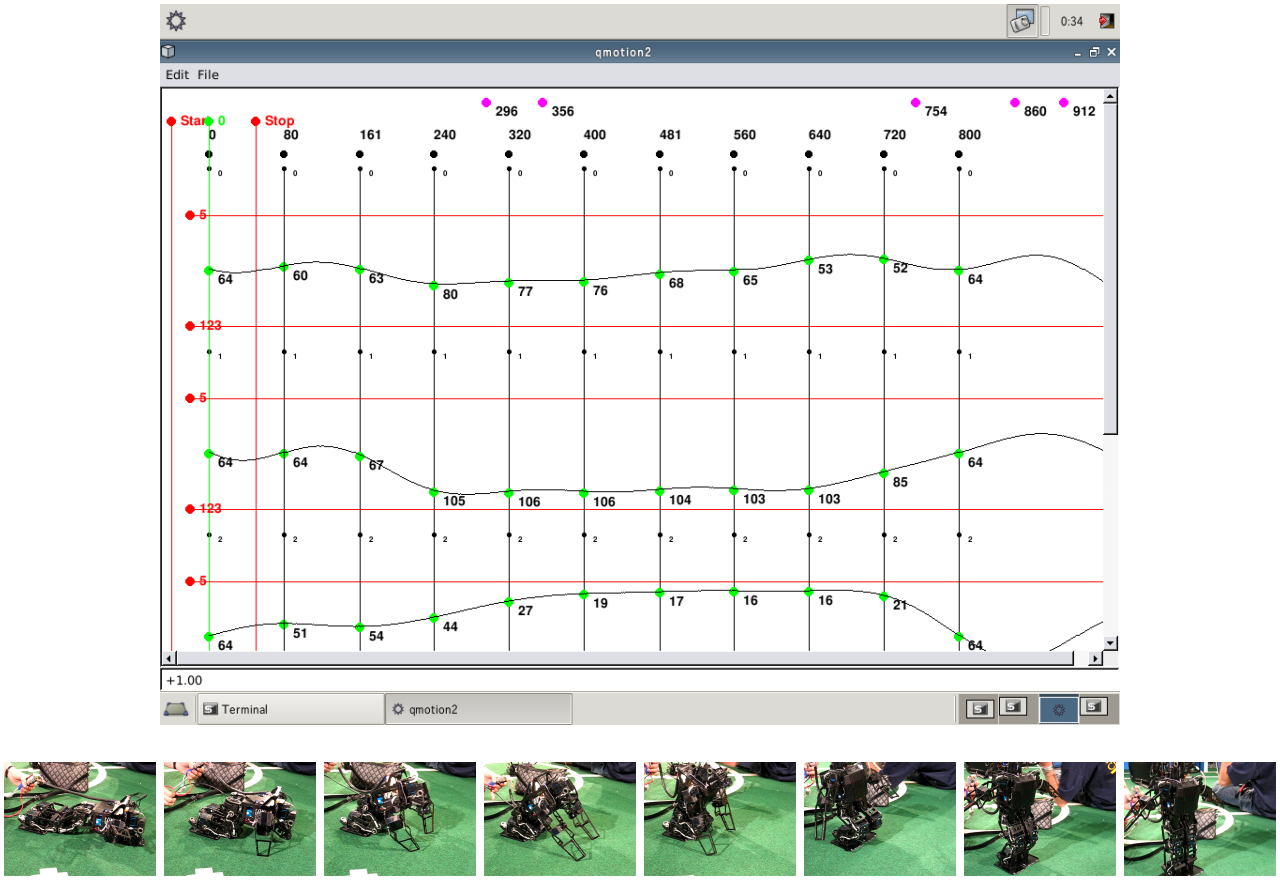

Fig. 3. Standing up motion designed by a tool to design HMDP motion patterns (screenshot). The screen shows in the top the activated frequencies of sines and cosine functions that perform the motion intierpolation (dots in purple). Below the resulting interpolated motion functions for selected joints. The time increases from left to right. The first row depicts the left hip joint in roll direction, the second row the first left knee joint (4G has 2 knee joints) and (half occluded) the motion of the second left knee joint. Below: The resulting motion for standing up of the robots (8 subsequent stages of the motion).

to be one of the few motions where the exact calibration is particulary essential. This might be caused by the fact that the arms of the robots of type 4G are relatively short.

The walking motion is the most prominent example for motion superposition. In the case of the JEAP team the walking motion has been achieved by the superposition of three basic motions: walking on spot, forward-backward, and turn. (In addition a moving sideward motion would have been very useful.)

In the next section, we describe SimSpark, the 3D simulator currently used in the SocSimL, and point out the features that make it suitable as a possible standard simulation platform.

III. SimSpark – A flexible simulator for physical multiagent simulation

The 3D simulator currently used in the SocSimL, called *SimSpark* was designed to be extremely flexible. It owes its flexibility to an underlying application framework called *Zeitgeist* which relies on two key concepts. These and other features of the system are summarized below, based on the detailed specifications in [8] and [4].

A. Key concepts

The first key concept is the implementation of a variant of the reflective factory pattern [9] which allows for factory based instantiation of objects at runtime while storing information about the creating factory in the object. This can be used to access class names and supported interfaces once the class objects are created, and is the basis for the scripting language interface for Zeitgeist (currently only implemented for Ruby).

The second key concept is the organization of object factories and created objects in a kind of virtual file system. Every node in this tree-like structure stores its own path name and a reference to its parent and child nodes. This way, services and objects can be easily located during runtime simply by providing a path expression. The object factories are stored at well-defined locations in this file system which makes it possible to instantiate objects of classes that are unknown at compile time, e.g., through the scripting language interface. This way, it is easy for instance to add new sensors, actuators, robot model formats, etc. as plugins to the simulator.

B. Physical simulation

The SimSpark simulator in its current implementation uses the Open Dynamics Engine (ODE) [10] for physically realistic

dynamics simulations. ODE offers fast rigid body simulations, collision detection, and supports the use of articulated body structures. Furthermore, it has integrated joint motors that help to stabilize the simulation. The most important ODE concepts are available im SimSpark, encapsulated in classes to fit into the overall object-oriented design of the simulator.

In practical, complex applications, ODE based simulations have sometimes turned out to be hard to stabilize. Extensive knowledge of the numerous ODE parameters is necessary in order to ensure stability, and it is easy to make mistakes. Therefore, the use of other freely available physics engines like for instance Bullet [11] as alternatives will be investigated in the future.

C. Core simulator

The core of the simulation binds together processes like timing, event management, and communication with external processes. It contains the main runloop of the simulator. From the beginning the design allowed for a customization of the runloop with replaceable components. Two runloop modes were built in: a simple, straightfoward loop that would realize actions sent by the agents as soon as they arrive, and a more elaborate one using the SPADES middleware [12], designed for maximum reproducability of distributed simulations.

It turned out that the SPADES based runloop was too complex and system setup not mature enough to be used in competition situations. The simple runloop model was, however, too indeterministic for accurate control, exhibiting large variations in execution time based on the load of the server. Therefore, a new timer was implemented that runs the control loop at 50 Hz, the rendering at 25 Hz, and can implement different delays for different sensor and actuator types. Furthermore, it can run in single-threaded or multi-threaded modes (with the latter still being an experimental feature though). Some problems remain even with this new improved runloop though, as we will point out in the discussion in section V.

In order to customize this generic simulation core for specific simulations, the game or application logic can be implemented as plugins. Furthermore, descriptions of the environment and the robot models can be specified using the *RubySceneGraph* language (see [4] for details), designed especially to facilitate hierarchical descriptions. Partial support for other formats such as RoSiML [13] has been implemented and can be used as well.

D. Advanced 3D Visualization

SimSpark includes an advanced 3D visualization application. It supports internal rendering of the simulated scenes, as well as streaming of scene description data over the network for visualization in external monitor applications. It also supports recording of log files for later (view-independent) playback, or MPG video files for playback in a standard video player. In addition to the built-in library *Kerosin* [14], the open-source rendering engine OGRE 3D [15] has recently been integrated and allows for state of the art, hi-fidelity visualization.

Fig. 4. A screen shot of the current soccer simulation setup in SimSpark.

E. Network support

Agents as well as external monitors for visualization can be connected to the simulator via network. TCP and UDP connections are supported, although only TCP is used at the moment. The implementation of the network communication is very modular so that different kinds of protocols can be supported easily. Furthermore, there is an interface for a so-called *trainer* application. They can connect to the server and make use of a special protocol in order to move objects in the simulation, set game states, and repeat and evaluate game scenarios. This is especially useful for debugging and machine learning applications.

F. Cross-platform support and tools

SimSpark implementations exist for Linux, Windows, and MacOS X operating systems. Currently, they differ somewhat in the status of their implementation, but eventually, they should be unified. In addition to these ports for different operating systems, the SocSimL community has contributed an increasing number of tools to support the simulator use, among them things like visual editors that allow easy construction of new robot models with export functions into the supported file format.

Simulators based on the SPARK framework have been used in the 3D SocSimL since 2004 and the community of users has grown to a considerable size. At RoboCup 2007, humanoid robot models were used for the first time, and while stability problems were an issue at times due to non-optimal setting of physics parameters, the overall experience was positive and encouraging. Figure 4 shows a screen shot of the simulation as used at the competitions in Atlanta.

The sum of the features described above, with flexibility being the most important one, make SimSpark generally very suitable for a wide variety of physical multiagent simulations, and therefore, as a standard simulation platform across the different RoboCup soccer humanoid leagues. It is a central

component in the 3D2Real project described in the next section.

IV. The 3D2Real Project

The 3D2Real project [16] was initiated in 2006. The main idea of this project is to try and use synergy effects from a collaboration between researchers in the Humanoid and the Soccer Simulation League (SocSimL). The concept could be easily extended to the SPL, if this would be seen as useful by the organizers of this league. The possible collaboration includes a joint roadmap for the near future of both leagues, as well as the specification of standards and the development of tools that can be used in both leagues.

Humanoid robot simulations have been performed in the RoboCup 3D SocSimL at the 2007 competitions for the first time. In the HL, on the other hand, the first multi-robot games have been held, and the great progress in controlling the robots allows researchers to approach issues of collaboration and coordination which have been extensively studied in the SocSimL. In short, both leagues are beginning to come closer to each other, and joint efforts in the development of tools and architectures that allow easier transfer of knowledge and technologies could speed up the mutual progress.

The goal we envision for the 3D2Real project is to have the finals of the simulation league using real robots by the year 2009 or 2010. For this ambitious goal several steps are necessary in the next years to create the required infrastructure and tools. First, real robot prototypes have to be implemented as a simulation model in the 3D simulator. According to the proposed road map, a technical challenge would be held at RoboCup 2008 to test the ability to use the agent code of SocSimL participants on a pre-determined real robot.

The HMDP introduced in this work could be used as a standard for the motion description of the simulated and the real robots. In the simulation, the agents are connected to the simulator over the network. This means that they have to send messages back to the simulator in order to specify, e.g., desired velocities for the motors. Since many agents connect to the simulator at once during a game this can lead to a high volume of network traffic that can cause severe problems for the server. This in turn, can cause clients to get out of sync with the server, and their control to be jerky. If the HMDP were used for the description of motion patterns, longer messages describing the motions would only have to be sent sporadically when new patterns have to be set. Actually the transmission of all occurring patterns before the beginning of the game is possible and probably the best solution. Certain times in the simulation when movements should be carried out can be specified, thus abstracting from real time, which facilitates smooth control .In the case of the real robots the motion patterns can be stored in the flash memory of the microcontroller and would readily be available at every reboot until they are replaced by a different set of motion patterns.

Figure 5 shows the intended software concept of the 3D2Real project; it depicts also the possible position of the HMDP within this framework. The HMDP could serve as a standard interface for the motion description between the player software and the robot motor control (either in the simulation or in the real robots.

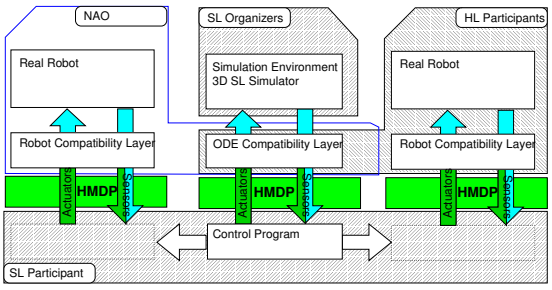

Fig. 5. 3D2Real project: Layout of the control architecture. The hatched boxes show how the different leagues contribute to the complete system architecture of the 3D2Real project. The control program for simulation system and real robot system are identical.

In the final section, we will discuss problems and benefits of a standardization for the different humanoid leagues in more detail, and give directions for further steps in the project.

V. Discussion and Outlook

One of the main characteristics of the current SocHL is that each team develops their own customized hard- and software. When the league was founded this was simply a necessity due to the lack of available commercial robot platforms. The pioneering teams of this league then explored different design ideas some of which can still be found in the robots competing in current competitions and have made their way into commercial products.

Although the SocHL continues to provide innovative robot design approaches, one point becomes more and more apparent: the fact that hardware as well as software has to be developed at the same time poses a big obstacle for newcomers to this league. Even now, the sophistication of the systems in use require large teams (up to 20 people) in order to remain competitive. In addition, the lack of standards makes a comparison of different approaches beyond the raw competition results quite difficult.

With respect to these considerations the new leagues may complement the SocHL. Even more so if some interchangeability of components (software and hardware) could be guaranteed by the organizers (cf. section IV).

The SocSimL and SocSPL would then focus on the software development while the SocHL could concentrate mainly on the development of hardware and robot design, making use of prepared software components in the software-centered humanoid leagues (SocSimL and SocSPL).

An important issue for the SocSPL will be to find a way to incorporate the best hardware solutions and control algorithms into new versions of the standard robot. Naturally, this depends to a high degree on the manufacturer, but it would be useful to improve the SocSPL robots in a yearly cycle with solutions proven successful in the SocHL competitions of the year before. Equally important would be for the SocSPL robots to adopt standard protocols like, for instance, the HMDP

presented in section II, that facilitate the transfer of software to other platforms (in SocHL and SocSimL).

One problem that has made progress in the 3D SocSimL difficult is the lack of resources for development of the shared simulation platform. In this league, the situation is such that a small group of volunteers takes care of the actual maintenance of the source code that everybody in the league uses for their research, and at the competitions. These volunteers are usually students or researchers who have many other duties, and consequently, the implementation of new features of the simulator depends entirely on the volunteers' schedules. Nevertheless, the community of course expects features to be ready for the competitions.

A possibility to alleviate this burden for the 3D SocSimL developers would be if participants of HL and SPL would provide models of their robots, as well as sensors and actuators in a standard format that can be used in the 3D SocSimL platform. As proposed in [16], this could be the XML based RoSiML [13] format, or maybe an open standard like COLLADA [17] (or an extension thereof).

As summarized in section III, many features of SimSpark make it attractive as a general simulator for the RoboCup soccer humanoid leagues. Nevertheless, several problems remain and need to be addressed to improve usability. These are namely: improved user and developer documentation, speed optimizations through consistent reference caching in Zeitgeist, and correction of the physics parameters for better stability of the dynamics in the simulation. Here the developers can learn from other open source simulators like USARSim [18]. In addition, problems with synchronization of agent programs with the server can occur if the server is under heavy load (cf. section IV). Implementing the HMDP for control of the robots in the simulator could solve this problem.

Besides the HMDP as a possible standard for robot control, other standards should be considered and evaluated. The OMG for instance has recently accepted the Robotic Technology Component Specification [19] and is currently working on the Robotic Localization Service API [20]. The Player/Stage project [21] also defines a language for robot control, and more candidates can be found.

Some of the next steps to advance the project would be to implement the HMDP in SimSpark, and specify a model of a real humanoid robot, including appropriate models for sensors and actuators. These are important preparations for the technical challenge in the SocSimL in 2008 as proposed in the 3D2Real roadmap.

Furthermore, it would be interesting to host demonstration games between robots of the SocHL and the SPL in order to assess the current levels of performance and identify issues for closer collaboration.

Finally, since the RoboCup is a free project, standards cannot be enforced and are if at all deliberately accepted. However, it might be worthwile to create incentives for conformance with proposed standards for the benefit of collaboration across leagues and faster progress towards the 2050 goal.

ACKNOWLEDGMENT

We thank K. Masui, S. Fuke, and M. Nuyen for their support with the screen shot material. We also thank I. Noda for helpful comments on the standardization efforts of the OMG. This reasearch was partially supported by a JSPS fellowship for young researchers which we gratefully acknowledge.

REFERENCES

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "RoboCup: A Challenge AI Problem," *AI Magazine*, 1997.

[2] H. Kitano and M. Asada, "The Robocup humanoid challenge as the millennium challenge for advanced robotics," *Advanced Robotics*, vol. 13, no. 8, pp. 723–736, 2000.

[3] M. Kögler and O. Obst, "Simulation League: The next generation," in *RoboCup 2003: Robot Soccer World Cup VII*, ser. Lecture Notes in Artificial Intelligence, D. Polani, A. Bonarini, B. Browning, and K. Yoshida, Eds. Springer, 2004.

[4] O. Obst and M. Rollmann, "SPARK – A Generic Simulator for Physical Multiagent Simulations," *Computer Systems Science and Engineering*, vol. 20, no. 5, 2005.

[5] (2007) Website Aldebaran Robotics. [Online]. Available: http://www.aldebaran-robotics.com/eng/

[6] N. M. Mayer, J. Boedecker, K. Masui, M. Ogino, and M. Asada, "HMDP: A New Protocol for Motion Pattern Generation Towards Behavior Abstraction," in *RoboCup 2007: Robot Soccer World Cup XI*, ser. Lecture Notes in Artificial Intelligence, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Springer, 2008, to appear.

[7] (2007) HMDP technical description. [Online]. Available: http://www.jeap-res.ams.eng.osaka-u.ac.jp/~michael/hmdp

[8] O. Obst and M. Rollmann, "Spark — A Generic Simulator for Physical Multi-agent Simulations," Universität Koblenz-Landau, Fachberichte Informatik 7–2004, 2004.

[9] C. Hargrove. (2000) Reflective factory. [Online]. Available: http://www.gamedev.net/reference/articles/article1415.asp

[10] R. Smith. (2007) Open dynamics engine website. [Online]. Available: http://www.ode.org/

[11] E. Coumans. (2007) Bullet physics library website. [Online]. Available: http://www.bulletphysics.com/Bullet/

[12] P. Riley, "SPADES: A System for Parallel-Agent, Discrete-Event Simulation." *AI Magazine*, vol. 24, no. 2, pp. 41–42, 2003.

[13] T. Laue, K. Spiess, and T. Röefer, "SimRobot - a general physical robot simulator and its application in RoboCup," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence, A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds. Springer, 2006.

[14] M. Kögler, "Simulation and visualization of agents in 3d environments," Master's thesis, Universität Koblenz-Landau, 2003.

[15] (2007) OGRE 3D website. [Online]. Available: http://www.ogre3d.org/

[16] N. M. Mayer, J. Boedecker, R. da Silva Guerra, and M. Asada, "3D2Real: Simulation League Finals in Real Robots," in *RoboCup 2006: Robot Soccer World Cup X*, ser. Lecture Notes in Artificial Intelligence, G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, Eds. Springer, 2007.

[17] (2007) COLLADA specification. [Online]. Available: http://www.aldebaran-robotics.com/eng/

[18] J. Wang, M. Lewis, S. Hughes, M. Koes, and S. Carpin, "Validating USARsim for use in HRI research," in *Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, 2005, pp. 457–461.

[19] O. M. G. (OMG). (2006) Robotic technology component (rtc) specification. [Online]. Available: www.omg.org/docs/mars/06-08-01.pdf

[20] ——. (2007) Robotic localization service request for proposal. [Online]. Available: http://www.omg.org/cgi-bin/apps/doc?robotics/07-06-25.pdf

[21] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the International Conference on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 30 - July 3, 2003*, 2003, pp. 317–323. [Online]. Available: citeseer.ist.psu.edu/article/gerkey03playerstage.html