HMDP: A New Protocol for Motion Pattern Generation Towards Behavior Abstraction

Norbert Michael Mayer, Joschka Boedecker, Kazuhiro Masui, Masaki Ogino, and Minoru Asada

Dept. of Adaptive Machine Systems,
Graduate School of Engineering, Osaka University, Osaka, Japan and
Asada S.I. Project, ERATO JST, Osaka, Japan
{michael,joschka,masui,ogino,asada}@jeap.org

Abstract. The control of more than 20 degrees of freedom in real-time is one challenge of humanoid robotics. The control architecture of an autonomous humanoid robot often consists of two parts, namely a real-time part that has direct access to the motors or RC servos, and a non-real-time part, that controls the higher-level behaviors and sensory information processing such as vision and touch. As a result motion patterns are developed separately from the other parts of the robots behavior. In research, particularly when including developmental processes, it is often necessary that the design or the evolution of motion patterns is integrated in the overall development of the robot's behavior. This is indeed one of the main principles of the embodied intelligence paradigm. The main aim of this work is to define a flexible way of describing motion patterns that can be passed to the motion controller which in turn executes them in real-time. As a result, the Harmonic Motion Description Protocol (HMDP) is presented. It allows the motions to be described as vectors of coefficients of harmonic motion splines. The motion splines are expressed as human-readable ASCII strings that can be passed as a motion stream. Flexibility is achieved by implementing the principle of superposition of several motion patterns. In this way also closed loop control is achievable in principle. Moreover, the HMDP can be implemented into the (deleted for blind review) project of the 3D soccer simulation league as a standard way to communicate motion patterns between the agent and the simulation interface and/or real humanoid robots.

1 Introduction

Many developers of autonomous robot systems experience difficulties when designing a control system that is at the same time capable of high level sensor processing, in particular vision sensors, and motor control. The solution is in most cases a hybrid design using 2 CPUs, one for motor control and one for sensor processing. The sensor data processing is often done by a PC like system with a broadband multitasking operating system (Windows, Linux) that usually does not have real-time capabilities. The motor control is done by a micro-controller

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 184–195, 2008.

that performs predefined motion patterns. The demanded motion pattern is communicated between both CPUs in some way, e.g., by serial bus. In particular in humanoid robots the motion control part has to be a real-time system in order to avoid jerky motions.

During the development process of the robot's behavior usually problems arise from this hybrid design. Whereas a PC-like system is always accessible, ready for changes, the motor controller can only be accessed via specialized editors and development tools, debuggers etc. Changes of motor controller programs can only be realized by flashing the limited memory that is available on the controller board. The programming of the motor-controller is mostly in done in C by using many custom definitions that depend on the design features of the motor controller which vary in dependence of the product line and the manufacturer. Moreover, the real-time behavior is managed by a series of interrupts that are again dependent on the type of the controller.

The problems usually result in a development process in which the motion patterns are developed separately from the design of the overall behavior. This seems acceptable in robot systems that do not require a big set of motions and do not have many degrees of freedom.

In humanoid robots, however, this design principle is not really satisfying. This is particularly true for soccer playing humanoid robots. Whereas humans have an infinite number of motion patterns available, the typical motion number of patterns of robots that participate a the RoboCup is normally less then 10, e.g. strong kick, soft kick, walking, turn, several goal keeper behaviors. A first step would be to allow for the activation of several motion patterns at the same time. This can be used for looking for the ball and walking forward independently in parallel. Furthermore, it can be used to balance out perturbations from the walking process. Thus, in addition to the normal walking process a weak pattern can be added that can stabilize the motion pattern. For this purpose it is necessary that the exact phase relation between both motion patterns is controllable. This is one requirement for the protocol.

Splines and harmonic functions have been used in various projects in different fields so far. Greszczuk and Terzopoulos [1] describe learning of muscle-actuated locomotion through control abstraction in order to generate realistic animations for computer graphics applications. They employ artificial animals with many degrees of freedom and abstract learned controller functions using Fourier analysis. These compact controller representations are then synthesized in learning of higher-level behaviors which benefits from the dimensionality reduced form of these controllers.

Another example from the field of computer graphics is given in [2]. Here, Fourier expansions of experimental data of actual human behaviors are taken as a basis to interpolate and extrapolate locomotion for an animated human figure. The authors describe how rich variations of human locomotion can be achieved by superposition of different Fourier expansions. Furthermore, the abstraction of the movements allows different parameters of the animation (like

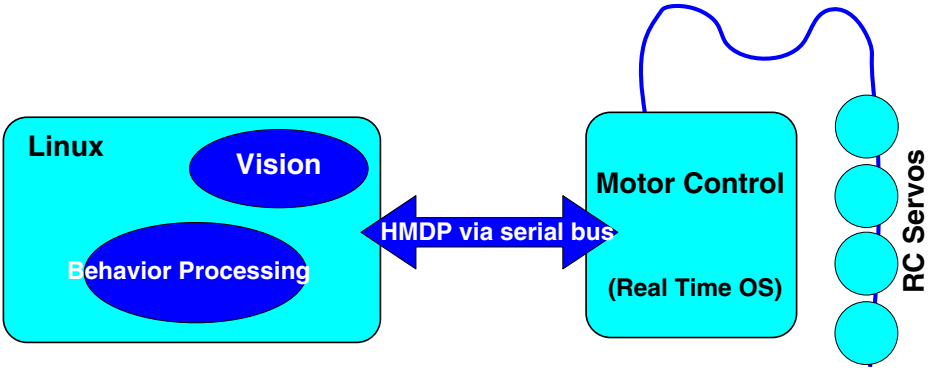

Fig. 1. Possible implementation of HMDP in a robot environment: Higher level behaviors are processed in a Linux micro PC (e.g. Geode). The PC sends motion patterns over the serial bus to micro controller. They are executed in real time.

e.g. step-length, speed, or hip position) to be controlled interactively to tweak the resulting animation.

In robotics, we find application of splines e.g. for trajectory generation of mobile robots (see [3] for the case of controlling an all-wheel steering robot). The idea of control abstraction for a more compact representation of movements is realized with different methods, for instance Fourier analysis for cyclic motion patterns as in [4] or using hierarchical nonlinear principal component analysis to generate high-dimensional motions from low-dimensional input [5].

In the following section we outline the requirement specifications that follow from the above mentioned motivations. We then outlined the principle of superposition of motion, its advantages and potential problems in section 3. The syntax of the HMDP as implemented in our software is described in section 4. To illustrate the work with the protocol, we provide an example using an experimental graphical user interface in section 5. Finally, we present a possible role of the HMDP in the 3D2Real project [6], and close with a discussion.

2 HMDP Requirement Specifications

To define the specifications of the HMDP more precisely:

- The HMDP includes messages that are submitted from the PC to the micro-controller 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

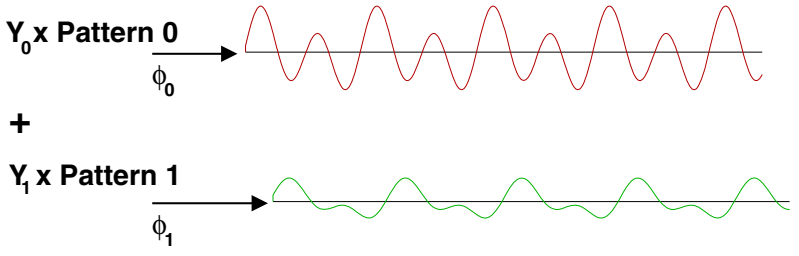

Fig. 2. Motion superposition: By using HMDP two or more motions can be superposed by defining the amplitudes Y_i and the phase shift ϕ_i. The resulting motion pattern is the sum of both initial patterns.

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 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 the information of the 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 is sent from the micro-controller to the PC.

3 Harmonic Motion Splines and Motion Superposition Principle

In this section we outline the principle with which motion patterns can be expressed in terms of motion splines; how they can be superimposed and under which circumstances the superposition of motions is useful.

In the following we discuss the harmonic motion splines for a robot with A actuators. Currently a mere position control is considered. We have spline functions that describe motion patterns $f_{p,a}(t)$. These are expressed in terms of discrete finite series of sine and cosine functions:

$$f_{p,a}(t) = c_0 + \sum_{0 \leq n \leq max} c_{2n+1,a} \sin(\rho \times \omega_{n,p} \times t) + c_{2n+2,a} \cos(\rho \times \omega_{n,p} \times), \quad (1)$$

where $\rho = \frac{\pi}{2N}$, $0 \leq p < P$ is the index of the pattern and $0 \leq a < A$ the index of the actuator. The set wave numbers $\omega_{n,p}$ is specified when the pattern is initialized. As a convention of the current HMDP for a specific pattern p it is identical for all actuators a.

The vector $\mathbf{f}_p(t) = \{f_{1,p} \ldots f_{a,p}\}$ expresses then the state vector of the robot, i.e., the positions of all servos, given only the pattern p is active with an amplitude of 1. The final position that is sent to the servo is then:

$$\mathbf{F}(t) = \sum_{p<P} R_p(t)\mathbf{f}_p(t - \phi_p) \quad (2)$$

Where the amplitude $R_p(t)$ and the offset ϕ_p are transmitted when the pattern is activated. Before the onset or change of the amplitude of a motion pattern the value ϕ_p, $Y_{new,p}$, T_{start0},T_{start1} have to be transmitted, $R_p(t)$ is then determined by

$$\begin{array}{ll} t < T_{start0} & : R_p(t) = Y_{old,p} \\ t \in [T_{start0}, T_{start1}] & : R_p(t) = (Y_{new,p} - Y_{old,p})/(T_{start1} - T_{start0}) \times t \\ t > T_{start1} & : R_p(t) = Y_{new,p} \end{array} \quad (3)$$

In other words the amplitude is changed in a linear way from the previous value to the current value. However, the value of ϕ_p changes at the time T_{start0}.

The design makes several types of messages necessary:

- **Pattern initialization message:** This message determines the ID of the pattern p. It also sends information about the used wave numbers $\omega_{n,p}$ for all n.
- **Coefficient transmission:** Since this can be a large set of information and since the buffer of of the receiving device is limited it seems useful to separate this type of message from the first message. So in this second type of message all coefficients $c_{i,a}$ with $0 \leq i \leq 2n + 1$ and $0 < a < A$ have to be submitted in "digestible" message sizes.
- **Use Pattern messages:** For using the patterns the on set times T_{start0} and T_{start1} as well as the the new amplitude Y_{new} and ϕ_p have to be used.
- **Sensor reading commands:** Those commands should contain a time value that specify for the motion controller at what phase of the motion the sensor value i.e. that potentiometer value of the joint should be read.
- **Time managing commands:** The time management has to be covered, in some way. The higher level controller should roughly know the current time value of the motor controller. The overflow of the time counter needs to be managed. The increment of the time counter should be changeable.

In the current approach the structures that manage the patterns are organized in a static manner. Also dynamic ways to storage the patterns seem possible.

The total motion of the control output according to equation 2 is the superposition of all active motion patterns in their current amplitude. The virtues of this superposition might not be directly obvious in the general case. In the following we go into three different examples where the superposition of motion patterns is useful.

First, examples for periodic movements: Independent movements concern non overlapping sets of actuators and are applied by simply running both patterns at the same time. With respect to humanoid robotics this can be done by looking for the ball – that is: moving the head and walking at the same time. Both patterns can have different wave numbers and can be applied completely independent from each other. Here it is necessary that both movements do not interfere with each other, e.g., that the limbs collide under certain circumstances. In addition it is only possible to have one pattern with dynamic effects on the whole body of the robot active at any time. In the case of walking and looking, only the walking would have an effect on the dynamic of the whole body of the robot. This kind of combination is only possible if both movements concern completely non overlapping sets of joints and one movement pattern leaves the joint that concerns the other movement in a default position.

The second example would be two movements at the same frequency. Where the first movement is the default behavior and second movement is a response of the control to some perturbation. As an example, take vibrations during walking; these can be damped by adding a regulatory movement on top of the standard movement.

The third example would be parametric non-periodic movements, like kicking. Here, the kicking direction can be superposed to a standard kicking behavior.

Limitations of the HMDP approach are closed loop control tasks that require inevitably control reactions below the limit of the reaction time of the combined system motor controller higher behavior controller.

4 HMDP Syntax

In the following we describe the set of messages and the way parameters and numbers are defined.

4.1 Characters Used in HMDP Syntax

It is a subset of the ASCII characters. At the current stage the HMDP uses: numbers ([0..9,a..f]), capital letters ([A..Z]) and the symbols $*$, $<$, $>$, +,-, @ ,! and &. Carriage return (decimal code 10,13) defines the end of a command line, after which the line is parsed in the processor.

4.2 Check Sum Feature

All commands can be sent in a check sum mode. The check sum is calculated as: $A = (\sum_i X_i)\ \ modulo\ 16$, where A is itself expressed as a hex-value([0-9,a-f]).

4.3 Long Command Feature

In some microprocessors command lines over 10 digits may become unsafe since the hardware serial-bus chips provide only a buffer of around 10 characters. Therefore it seems useful to break long messages down into shorter parts that are written into the program buffer. An & character at the end indicates that the current command line is continued after the carriage return. In combination with the check sum feature messages up to a defined length limit can be sent safely.

4.4 At-Time Execute Command Feature

With this feature a command - usually a command to read out sensor values - is executed at a specified time. In the current implementation the slave assumes that the at-time commands are sent in sequence, i.e., the command that should be executed first is sent first etc. The slave does not sort the commands and would wait until the time for the next command in its batch is reached and then look for the start time of the following at-time command. Thus, if one sends an at-time command for time 100 and then an at-time command for time 50, both commands are executed at 100 and 101, respectively. If the time counter is already behind the time given in the at-time command (like in the previous example) the command is executed in the next time cycle.

4.5 Types of Numbers

Numbers are transmitted as hex numbers that include the numbers [0..9] and the letters [A..E]. Currently, three types of numbers are used: integers, rational numbers, and real values.

- Integers are transmitted as conventional hex numbers.
- Rational numbers are used to express wave numbers. Since most motor-controllers can only emulate floating arithmetic by software, rationals seem to be faster than real values. They are expressed by a set of two subsequent integers.
- Floats are described in a standard semi logarithmic way by transmitting the exponent and the mantissa as integer values including their signs.

4.6 Internal States

The robot can be set into 3 distinct internal states. Depending on the activated state different groups of commands have different effects. It is important to note that if a command is used in the wrong state the real-time property may be disrupted, the command may have an undesired effect, or no effect at all.

- State 0: The motion machine is deactivated and commands can be directed towards the motors. Command groups 0,I can be used.

– State 1: HMDP state. The motion machine is on and overwrites motor commands (group I). Instead the HMDP commands have to be used that control the motion machine. At-time commands are possible but may interfere with the timer and therefore the motion may not be precise (lost ticks may appear).
– State 2: "Plastic" state of the robot. The robot can be set manually into a state, and keeps the current posture, while changes in the current posture are possible by applying force to the servos.

4.7 Group 0 Administrative Commands

This group communicates state variables and other information of the current state of the system. The syntax is usually $> XX$ for a command to the slave and $< XX$ for requested information. In addition, some commands for copying motion patterns into the flash memory are provided. Servos can be turned on and off. For the protection of the servos allowed ranges of position values of the servos are defined. Zero positions[1] can be re-defined, and the list of available servos can be requested. These commands are typically used when the robot is in state 0, but can also be used in the other two states. However, in state 2 they affect the real-time property, and ticks may be lost.

4.8 Group I Static Posture Commands

The commands affect the posture of the robot directly. In state 0 these may be used to control postures, which is useful in the development phase of the the motions and in order to calibrate the zero positions. In addition, positions can be read from the controller. There, the controller can distinguish between the actual current position and the position that is targeted by the controller.

4.9 Group II HMDP Commands

An HMDP message starts with a systematic set of key characters which simplifies the parsing of the protocol. An additional initial character in front of every message can be custom defined in order to make it possible to add HMDP to already existing messaging systems. Apart from the custom defined header the first character of each HMDP messages can be either a P, a T, or S, indicating a time-related, pattern-related or sensor-related command.

5 Example for a Visual Motion Design Tool

We programmed a graphical user interface (GUI) (see Fig. 3) in which the coefficients can be found be defining manually frequencies and support points. The

[1] Zero positions are the calibrated values of the servos of a robot in an upright position and the arms in a certain diagonal angle.

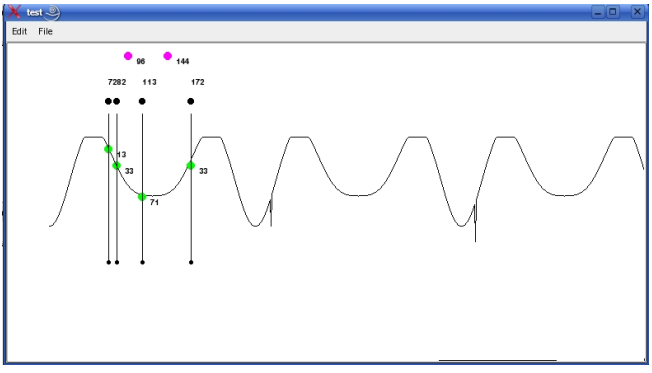

Fig. 3. Experimental graphical user interface for motion design MotionDesigner: Currently the wave numbers have to chosen by hand (pink dots). Then a function (black curve) is interpolated between support points (green dots).

program calculates the coefficients and produces HMDP messages, that define this particular motion pattern. Coefficients c_i are calculated by solving the system of equations that are defined by

$$x(t) = c_0 + \sum_{0 \le n \le max} c_{2n+1,a} \sin(\rho \times \omega_{n,p} \times t) + c_{2n+2,a} \cos(\rho \times \omega_{n,p} \times t). \quad (4)$$

Since a set $x_i(T_i)$ for a certain T_i and all $\omega_{n,p}$ are defined by the user, we have a set of linear equations that can be solved by deriving the pseudo inverse. In dependence of the ratio between the number of $x_i(T_i)$ and the number of coefficients c_j we get an under- or over-defined system of linear equations.

The program currently controls a virtual motion controller. The motion control part is C-code and can readily be implemented into a standard motion controller ICs down to the level of PIC chips or similar.

6 Possible Role in the 3D2Real Project

One problem for the RoboCup project is that throughout the leagues a lot of work is duplicated, and collaboration is rather sparse between the different leagues. This is not a desirable situation as know-how is not transferred effectively, and progress is slower than it could be since resources are bound to solve the same problems over and over again. To address this situation, the 3D2Real project 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 (SSL). This collaboration includes a joint road map 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.

Traditionally, the SSL and the HL in RoboCup have had rather different research topics. While researchers in the HL mainly worked on the design and the low-level control of their robots, participants in the SSL were concerned with high-level strategies and collaboration. In recent years, however, there have been developments which might bring both leagues closer to each other. In the SSL, there have been continuing efforts to introduce more realism into the rather abstract simulation in order to ensure that the developed strategies can be transferred more easily to real robots. Humanoid robot simulation is the preferred choice for many participants of the SSL in order to achieve this. 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 SSL. 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 towards the 2050 goal of RoboCup.

The goal we envision for the 3D2Real project is to have the finals of the soccer 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 necessary infrastructure and tools. First, the 3D simulator of the SSL [7] has to be completed, and a real robot prototype has to be implemented as a simulation model, the XML-based format *RoSiML* as used in the *SimRobot* simulator [8] seems promising. 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 SSL participants on a predetermined real robot. From 2008 until 2009, we propose the development of a *central parts repository* (CPR). This would be a collection of real robot designs, sensor and actuator models, complete robots, as well as controllers for certain architectures. Participants of both HL and SSL contribute to this repository according to their expertise and interest. The format would again be the *RoSiML* mentioned above. These contributions become a mandatory part for the HL qualification from 2009, and should be continued (at least) until 2010, even after the 3D SSL final has taken place using real robots.

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 (currently in ASCII strings) back to the simulator in order to specify, e.g., desired positions for 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. 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. Thus, the HMDP would provide a good solution for very related problems in the 3D soccer simulation league, and the humanoid league (as described in the introduction), and might facilitate running the same code on simulated and real humanoids eventually.

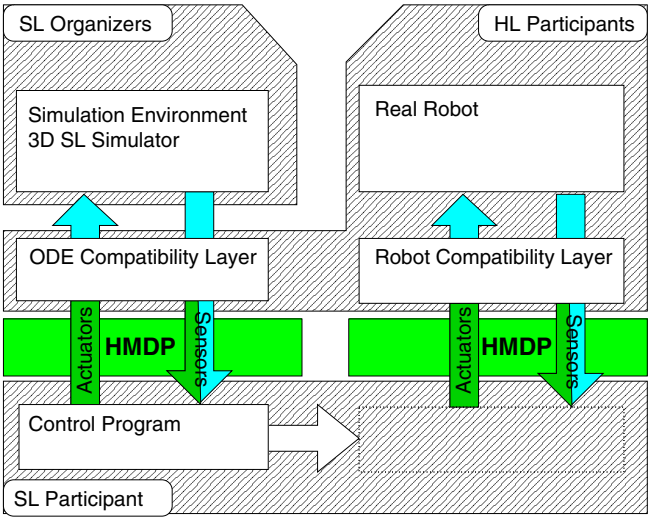

Fig. 4. 3D2Real project: Layout of the control architecture. The hatched boxes show how the different leagues contribute to the complete system architecture of the 3D2Real project. The control program for simulation system and real robot system are identical.

7 Discussion

The intention of this work is to provide a standard for the description of motion patterns for several purposes. It can be seen as an abstract but flexible motion description protocol in a server and client architecture. Moreover, it can be used for simulation purposes and at the same time for connection between a higher-level behavior control unit and a real-time motion controller. Its merits are biggest in situations where motions patterns should be changed "on-the-fly", like in a scenario that uses developmental or evolutionary methods to change the motion generation, but it also allows for a great flexibility in motion execution in general.

The representation of motions by Fourier coefficients in the protocol and the superposition principle are valuable properties for behavior abstraction, i.e., the synthesis and blending of new, higher-level behaviors from compact representations of lower-level ones. This is one aspect we want to explore further in the future, as well as the implementation of currently missing features like variable time increments, sensor readings, and overflow of time values that can happen at least theoretically.

Furthermore, the 3D2Real project is one example inside RoboCup where the HMDP seems to be an appropriate tool. The plan is now to test and improve the HMDP in dependence on results within the 3D2Real project.

Acknowledgements

We greatfully acknowledge the support of this work by the Japan Science and Technology Agency (JST), and a fellowship for young scientists from the Japan Society for the Promotion of Science (JSPS).

References

1. Greszczuk, R., Terzopoulos, D.: Automated learning of muscle-actuated locomotion through control abstraction. In: Proceedings of SIGGRAPH 1995 (1995)
2. Unuma, M., Anjyo, K., Takeuchi, R.: Fourier principles for emotion-based human figure animation. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 91–96. ACM Press (1995)
3. Howard, T., Kelly, A.: Trajectory and spline generation for all-wheel steering mobile robots. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), October 2006, pp. 4827–4832 (2006)
4. Schmidt, H., Sorowka, D., Piorko, F., Marhoul, N., Bernhardt, R.: Control system for a robotic walking simulator. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation (2004)
5. Tatani, K., Nakamura, Y.: Reductive mapping for sequential patterns of humanoid body motion. In: Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines (2003)
6. Mayer, N.M., Boedecker, J., da Silva Guerra, R., Asada, M.: 3d2real: Simulation league finals in real robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434. Springer, Heidelberg (2007)
7. Obst, O., Rollmann, M.: SPARK – A Generic Simulator for Physical Multiagent Simulations. Computer Systems Science and Engineering 20(5) (September 2005)
8. Laue, T., Spiess, K., Röefer, T.: SimRobot - a general physical robot simulator and its application in robocup. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020. Springer, Heidelberg (2006)