CrudeScientists Team Description

N. Micheal Mayer[1], Alan Liu[1], Chia-Hao Hsieh[1], Chun-Ming Chen[1], Hsien-Chang Lin[1], Jheng-You Lin[1], Jia-Heng Zou[1], Juris Klonovs[1], Meng-Xuan Jia[1], Pin-Chih Lin[1], Tsung-Lin Kuo[1], Wei-Lien Sung[1], Yu-Hua Xiao[1], Joschka Boedecker[2]

[1]National Chung Cheng University, Chia-Yi, Taiwan, R.O.C.
[2]Osaka University, Osaka, Japan
email: mikemayer@ccu.edu.tw

Abstract. We implemented the Harmonic Motion Description Protocol (HMDP) as a module into SimSpark. Initially, HMDP was developed in real robots. It served there as a framework for the communication between the behavior system and the real time motor controller. In case of SimSpark it serves for the same purpose between the agents (clients) and the server. Thus, it is possible to upload pre-designed motions before the game starts. The superponability of those motions allows for an infinte set of possible motions. We see several advantages of HMDP in contrast to the current usage of the effector control and thus suggest to add the HMDP module to the standard competiton server. In addition, we provide a tool for motion design, of which we add the sources to the present competition contribution.

1 Introduction

One important feature of SimSpark is its extensibility using dynamical loadable plugins that can contain sensors and effectors. In our contribution we present a plugin that basically runs on software for motor controllers. The very same software had been implemented in real robots before; the first robot has been a Vision 3G robot of VStone Inc. in Osaka. The design of the 3G robot is typical for robots that participate at the RoboCup Humanoid League and resembles also the design of the Nao robot of Aldebaran robotics: It consists of 2 CPUs (excluding the very small embedded systems on each servo motor): While vision processing and higher level behavior control are done by the first CPU, real time motion processing is done by the other one. Both communicate over serial bus with each other.

The task of the motor controller CPU is to react on abstract motion commands and translate this in a real-time sequence of postures that are send in precise timing to the servo motors. The implementation of the Harmonic Motion Description Protocol (HMDP) has been suggested earlier [1][2] as a flexible software

on the motor controller side. The parsing of the messages and the execution of the resulting trajectories has been implemented by using a self-sufficient C-code, that does not require external libraries and even refrains from using float type variables because those require external libraries in some CPUs for embedded systems. One important feature of HMDP and the underlying motor controller software is that several motions can be linearly superposed on top of each other and thus parametrizable motions can be created. In addition, several libraries have been programmed that support the communication on the client side. Last but not least a motion design tool with a GUI has been programmed that is called Qmotion2.

The implementation of HMDP into SimSpark had also been suggested earlier as part of the 3D2Real project [3]. The current contribution is that we have actually implemented HMDP including the above mentioned –though further developed– self-sufficient C-code. The role of the motor controller part is performed by the plugin, while the external client communicates over an effector and perceptor class with the module. The code is encapsulated in a namespace. Global data entities that contain information about motion patterns, timing and others are comprised in 2 structures, that have to be instantiated once for each agent that is connected. At present the HMDP parts have been added to module soccer.so in the rcssserver3D part of SimSpark.

SimSpark/HMDP have successfully been used in a class about open source software for and in embedded systems. The students were able to very quickly design motion patterns such as standing up of the robot and walking. They were able to create parameterizable motions and search the parameter space for optimal motions (e.g. as fast as possible walking). A report about their progress has been added to the contribution material.

2 HMDP

The name HMDP originates from the fact that –instead of a sequence of postures– harmonic functions (i.e. several cosine and sine functions) are used to describe motions. In this way both repetitive motions (walking) as well as non-repetitive motions are possible. Non-repetitive motions are realized in the way that a start and a stop point are given, also a slow fade in and fade out of a motion is possible. Finally, the motions can be run with different amplitudes. Thus, by superposing several motion patterns with different amplitudes, a parametrizable motion can be created, in which the amplitudes of the underlying motions serve as parameters of the resulting motion of the robot. In this way parametrizable motions can be designed in standard way: First one extreme example is designed by hand (e.g. walking on spot) and then another extreme example (e.g. walking as fast as possible forward), provided that both motions use the same base frequencies, and further constraints, a parametrizable motion between these 2 extremes can be achieved by superposing those with the first motion having the

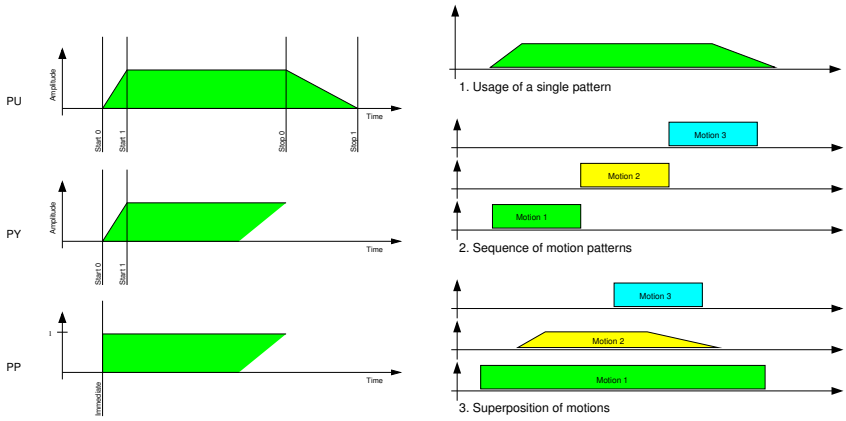

Fig. 1. Left: HMDP commands for different usages of the motion patterns. The PU command (for the HMDP syntax refer to the above mentioned publications) can trigger a motion pattern with end. PY triggers the motion pattern without giving a point in time for the end. The PP command is for testing and starts a motion pattern immediately without end. Right: Examples for possible implementations of motion patterns. The third example is making use of the superposition principle.

amplitude α and the second having the amplitude $1-\alpha$, where α is value between 1 and 0. Please refer to Fig. 1 for more useful applications of this feature. For optimization algorithms (as –for example – genetic algorithms or reinforcement learning) we see also advantages if HMDP is used: Instead of an optimization for every time step on every joint, now by using HMDP an optimization of a set of pre-defined motions can be chosen. Thus, the dimensionality of the search space can be significantly reduced. Reinforcement learning on motion patterns can become a fairly feasible task.

The syntax of HMDP allows to define a motion and transfer it once to the server-plugin and then use it for an infinite number of times. This is in several ways an advantage to the standard way in which the joints are controlled by the SimSpark client up to now. Thus, HMDP allows for pre-designed motion patterns to be transferred to the server before the game starts. Since during the competition most teams have a finite fixed repertoire of motions for standard situations (walk, stand-up, kick etc.), HMDP fits the needs of these teams very well: The teams can upload all the motions they plan to use before the game starts and then activate or deactivate these motion patterns as they like during the game. We see advantages in the reduced load of communication between server and client during the game, and the higher precision in which motion patterns can be realized.

Synchronization between the client and the simspark server has been an issue for the designer of the server in the past. If HMDP is used as a layer for motion

abstraction the client can be much more independent from the timer in the server.

Provided that the HMDP-software has been implemented on a real robot, motion patterns become exchangeable between the particular model of the real robot and the simulated robot. This may prove useful to test in the future the quality of simulations.

3 QMotion2

The motion design tool Qmotion2 (see Fig. 2) has been added to the source code of the present contribution including a detailed manual. The tool connects as a client to the HMDP server. It can connect to a server over both serial bus and a SimSpark TCP/IP connection. A motion is first defined as a sequence of postures. If a real robot is connected the posture can be modelled on the robot and then the position of each servo can be read into Qmotion2. For testing single postures can be transmitted and checked — either in SimSpark or the real robot. The set of sine and cosine waves (i.e. their frequencies) can be set by putting frequency pointers on the window. Choosing more frequency pointers results in a more precise interpolation of the set of postures given, less frequency pointers with small frequencies result in a smoother but less precise interpolation. Advanced tasks as standing up can still be achieved by trial and error. Since Qmotion2 bases on Qt3 and also at present it has only been implemented in Linux, a port to other operating systems seems easily possible. Designed motion patterns can be stored in binary files. QMotion also provides tools for setting a start and and end point for non-repetitive motions (as standing up) and finally symmetry constraints can be added for periodic motions in that way that motions of the left side are repeated after half a period of the base frequency on the right side and vice versa. This makes the design of walking easier.

4 Adding motions designed by QMotion2 into the own source code

Each motion pattern is encoded in capital letters and hexadecimal numbers. The HMDP commands that are necessary to define a motion pattern can be added by cut and paste into own C-code from the output of QMotion2. An example is provided in a small additional source code has been added for illustration.

5 Outlook and Discussion

The advantage of the described method to previous designs is that in comparison to streaming of robot posture information the communication load can be re-

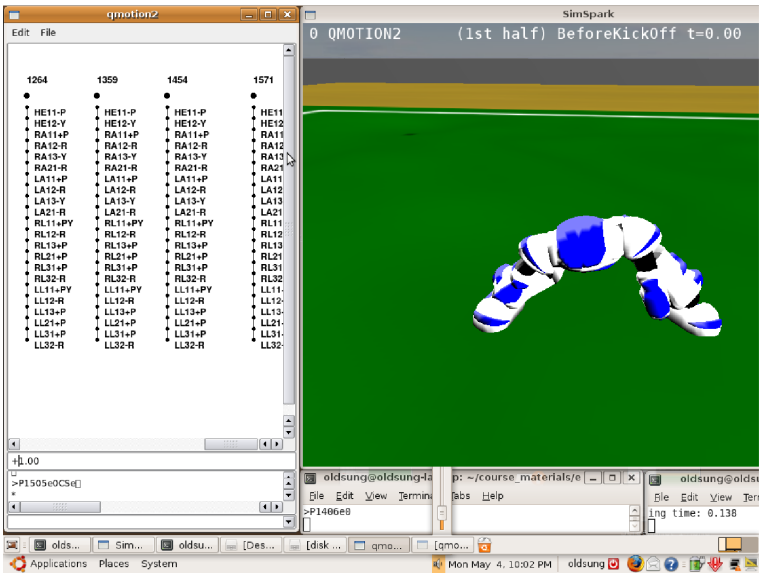

Fig. 2. Example for Qmotion2 used together with SimSpark. The x-axis in the Qmotion2 editor window represents time. The vertical black line with the dots represents the state of the robot at the time which is expressed in the number that is printed above the line. Each dot on each line represents one servo. The value of each servo can be seen and modified by mouse clicks. The widgets below allow to change the amplitude with which the motion pattern should be activated. Finally, the communication is displayed on the widget below and in the widget at the bottom of Qmotion2 editor window own HMDP commands can be written by hand and sent directly.

duced. At the same time motion design and management are still highly flexible (all kinds of parameter motions are possible). One important disadvantage to a standard control program on the motor controller is that closed loop approaches are more complicated to implement. It is nonetheless possible. A specific expected perturbation can be treated by a compensating motion pattern. Thus the control loop can look as follows: At a specific phase a sensor reading is executed. In the case of alarming sensor values a healing pattern can be activated in an appropriate amplitude. In this way although the feedback delay may be relatively large, the reaction precisely fits into the motion pattern.

Recently Python has started to be a new standard as a high level language. Powerful libraries (for example PyBrain [4]) for learning and adaptation are available. Thus, a Python interface to SimSpark (via HMDP) could be very useful.

Acknowledgements

The authors thank the Taiwanese National Science Council, the Swiss National Fund (SNF) and the Japanese JSPS and JST for their financial support. We also would like to thank K. Hwang, J. Schmidhuber, M. Asada, H. Ishiguro, K. Masui, S. Fuke, R. Silva da Guerra, and M. Ogino.

References

1. Norbert Michael Mayer, Joschka Boedecker, Kazuhiro Masui, Masaki Ogino, and Minoru Asada. HMDP: A New Protocol for Motion Pattern Generation Towards Behavior Abstraction. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, Lecture Notes in Artificial Intelligence. Springer, 2008. to appear.
2. N. Michael Mayer, Joschka Boedecker, and Minoru Asada. Robot motion description and real-time management with the harmonic motion description protocol. *Robotics and Autonomous Systems*, 2009. to appear.
3. Norbert Michael Mayer, Joschka Boedecker, Rodrigo da Silva Guerra, Oliver Obst, and Minoru Asada. 3D2Real: Simulation League Finals in Real Robots. In *RoboCup 2006: Robot Soccer World Cup X (Lecture Notes in Computer Science) (Paperback) by Gerhard Lakemeyer (Editor), Elizabeth Sklar (Editor), Domenico G. Sorrenti (E ditor), Tomoichi Takahashi (Editor)*, pages 25–34. Springer, 2007.
4. IDSIA. Pybrain: The python machine learning library. http://pybrain.org, 2009.