Robotics and Autonomous Systems 57 (2009) 870–876

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Robot motion description and real-time management with the Harmonic Motion Description Protocol

Norbert Michael Mayer [a,*], Joschka Boedecker [b], Minoru Asada [b]

[a] Department of Electrical Engineering, National Chung Cheng University, 168 University Road, Min-Hsiung, Chia-Yi 62102, Taiwan

[b] JST ERATO Asada Project for Synergistic Intelligence and Emergent Robotics Laboratory, Graduate School of Engineering, Department of Mechanical Engineering, FRC-1, Osaka University, 2-1 Yamada-oka, Suita, Osaka 565-0871, Japan

ARTICLE INFO

Article history:
Available online 23 April 2009

Keywords:
Humanoid robots
Motion design
Harmonic functions

ABSTRACT

We describe the Harmonic Motion Description Protocol (HMDP), that can serve as a part in tools for rapid prototyping of behaviors in a hybrid simulation real robot environment. In particular, we are focusing on the RoboCup 3D Soccer Simulation League server that is currently evolving rapidly, and becomes a more and more useful open source, general purpose simulation environment for physical Multiagent systems. HMDP uses harmonic functions to describe motions. It allows for superposition of motions and is therefore capable of describing parametric motions. Thus, typical open loop motions (walking on spot, forward, turning, standing up) of small humanoid robots are readily available and can be optimized by hand. In conjunction with the HMDP some software tools and a small real-time motion generator (called Motion Machine) have been developed. The current implementation is very flexible to use and can easily be implemented in rather small embedded systems.

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. This is particularly true for humanoid robots that can have around 20 degrees of freedom. For advanced systems (e.g. Honda's Asimo or Sony's Qrio robots) multitasking real-time OS systems (often with several CPUs) are available, that manage the entire sensor and actuator processing in real time. Low cost designs usually lack the sensor processing and are restricted to a remote controlled embedded CPU or even an analogue system (e.g. RoboSapiens, [1]). These systems completely lack autonomous behavior capabilities. For the medium to low cost designs (for example in RoboCup [2,3], where autonomous behaviors are demanded), the solution can be a hybrid design using 2 CPUs, one for motor control and one for sensor processing. On the one hand the sensor data processing – in particular vision – is done by a portable IBM-PC i386 like system with a custom broadband multitasking operating system (Windows, Linux, BSD) that does not have real-time capabilities (see for example [4–7]). The drivers for cameras and other devices are cheap and do not need further development. The motor control on the other hand is done by a micro-controller that performs pre-defined motion patterns. The demanded motion pattern is communicated between both CPUs by a serial pipe or bus system, sometimes wireless. In particular, in humanoid robots, the motion control part has to be a real-time system. Motion patterns such as standing up need to be precisely executed in the range of a time-span of 100 ms, in order to produce a reliable performance. For this reason direct positional control from the PC side should be avoided.

During the development process of the robot's behavior 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 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. Thus, for prototyping, the development of motion patterns may turn out to be a bottleneck, and developers may be reluctant to change a working motion pattern. The problems can 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.

* Corresponding author. Tel.: +886 5 2720411; fax: +886 5 2720862.
E-mail address: nmmayer@gmail.com (N.M. Mayer).

0921-8890/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2009.03.014

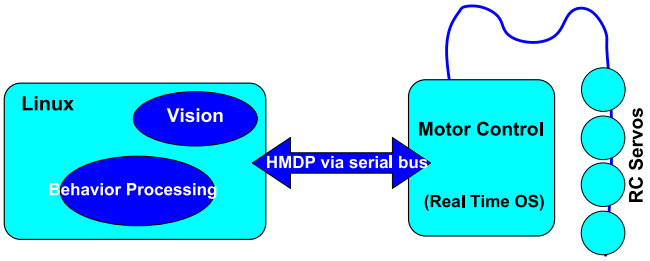

Fig. 1. Possible implementation of HMDP in a robot environment: Higher level behaviors are processed in a Linux micro-PC (e.g. Geode). The PC sends motion patterns over the serial bus to micro-controller. They are executed in real time.

In humanoid robots, however, this design principle is not really satisfactory. This is particularly true for soccer playing humanoid robots. Whereas humans have an infinite number of motion patterns available, the typical number of motion patterns of robots that participates, in the RoboCup competitions for instance is less than 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 walking motion. For this purpose it is necessary that the exact phase relation between both motion patterns is controllable.

This is one design policy for the protocol we present. The actual HMDP protocol defines the communication between the real-time motor controller and the PC-type vision- and behavior-processing computer—initially via serial bus (of course all kinds of other communication methods are possible, see also Fig. 1 for an illustration). The software environment which we describe in this work comprises also some tools for the motion design and handling, and a real-time motion generation and management system for the motion-controller CPU called *Motion Machine*. It is designed to be small and as possible hardware independent as possible.

In the robotics literature, some of the ideas that build the foundation of our protocol have occurred before. We find application of splines e.g. for trajectory generation of mobile robots (see [10] for the case of controlling an all-wheel steering robot, or [11] for an approach using a biped robot). Contrary to other works, though, we only store the Fourier coefficients of the motion splines and use these for motion generation in the micro-controller. We do not store any sampled spline curves in order to reduce communication load. The idea of control abstraction for a more compact representation (one of the design criteria for HMDP) of movements is realized with different methods, for instance Fourier analysis for cyclic motion patterns as in [12] or using hierarchical nonlinear principal component analysis to generate high-dimensional motions from low-dimensional input [13]. However, our main motivation for the creation of this protocol was not only to represent motions in a compact way, but also to address the issues concerning timing. Ideas related to the superposition principle of different motions can be found in [14]. There the pre-generated motions are analyzed and can be mixed together in frequency space (for instance to create smooth transitions). For this purpose an interpolation of Fourier coefficients is used. A powerful motion editor for small humanoid entertainment robots is described in [15]. In this work the created joint trajectories for the different motions are finally exported into files. These files contain control points and equations to interpolate between them to generate the desired motions in real time. Finally, a good overview of the general difficulties in motion planning for humanoid robots – specifically within RoboCup – is given in [16].

With regard to the simulation of robots similar problems arise from a different point of view. The RoboCup 3D Soccer Simulation League underwent some profound changes during the last several years. Currently realistic humanoid robots are simulated (see Fig. 3 for a screen shot of the NAO simulation in the official simulator called *Simspark* [17,9]). The motions of all robots are communicated between the agent program and the simulator as instantaneous updates of velocities of simulated servos communicated via TCP/IP. The role of HMDP in this framework can be to serve as an interface between the agent and the simulator (cf. Fig. 2). The details of the current state of the 3D Soccer Simulation League are discussed in publications that relate to the 3D2Real project [8,9]. The goal that is envisioned for the 3D2Real project is to have the finals of the soccer simulation league using real robots in the near future. For this ambitious goal several steps are necessary in the next years to create the necessary infrastructure and tools. According to the proposed road map in [8], a technical challenge would be held at the RoboCup competitions to test the ability to use the agent code of SSL participants on a predetermined real robot. We propose also the development of a *central parts repository* (CPR, see also [18]). This would be a collection of real robot designs, sensor and actuator models, complete robots, as well as controllers for certain architectures.

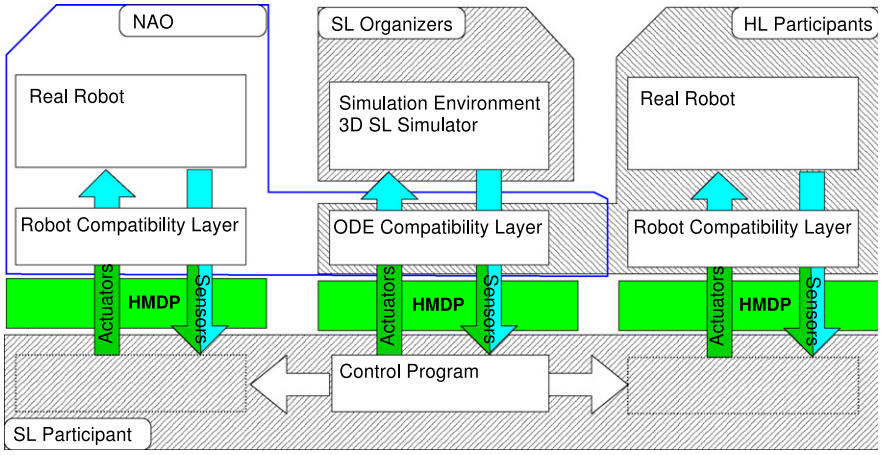

Fig. 2. Possible implementation of HMDP into a standardized software design and simulation environment (in this case 3D2Real [8]). The HMDP may serve at the same time as an interface for abstract motion description between agent and simulator as well as between agent and robot (see also [9]). The abbreviations stand for 3D Soccer Simulation League (SL), Soccer Humanoid League (HL); NAO is the name of the robot used in the new Standard Platform League. For details please see the above mentioned publications.