Distributed Real-Time Processing for Humanoid Robots

Toshihiro Matsui[*1], Hirohisa Hirukawa[*2], Yutaka Ishikawa[*3], Nobuyuki Yamasaki[*4]
Satoshi Kagami[*1], Fumio Kanehiro[*2], Hajime Saito[*2], Tetsuya Inamura[*3]

**1 Digital Human Research Center, AIST, Tokyo Japan*
**2 Research Institute of Intelligent Systems, AIST, Tsukuba, Japan*
**3 Department of Computer Science, The University of Tokyo, Tokyo, Japan*
**4 Department of Information and Computer Science, Keio University, Yokohama, Japan*
t.matsui@aist.go.jp *http://drtp.dip.jp/*

Abstract

A humanoid robot is a real-time system controlled by a complex computer system that requires huge computing power for perception and planning, high energy efficiency for self-contained control, reduction of physical dimensions, and high reliability. This paper proposes a distributed architecture for the humanoid robot control substituting conventional centralized control architectures. In addition to the parallelism that provides scalable computing power at low clock namely at low energy, the distributed architecture contributes to reliable operations by replacing many fragile analog signal wires with a digital network with redundant routes. In order to accomplish a real-time control over the network, RMTP (Responsive Multi-Threaded Processor) for parallel and real-time computation has been newly designed. RMTP can synchronize more than thirty nodes distributed over a robot body in less than 5 micro second with a real-time network called the Responsive Link (RL). Architectures of RMTP, RL and Linux-based real-time system software are presented.

1. Introduction

Recent advances of robot technologies greatly depend on the development of embedded computing technologies as well as advances in mechanisms, sensors, and actuators. In 1996, Honda-P2 revealed biped walk of a life-size humanoid robot with a totally self-contained computer system [1]. This was enabled by microprocessors backpacked on the robot. One of the innovations there was an application of workstation processors to embedded robot control with a modern real-time operating system.

Most following humanoid robots have taken the similar control architecture, for the real-time control of the centralized computer has been the only feasible way to realize 1ms servo loop of more than 30 joints of a humanoid robot. Biped walking, however, is not the only function for a humanoid robot to perform meaningful tasks, and the computer system is requested to respond to a growing demand of huge computing power. Although processor technologies following Moore's law could have provided the computing power, they also raised the problem of growing energy consumption. In addition, increasing length of signal cables from sensors and actuators to the central processor spoils the reliability of the total robot system.

This paper describes a project to break through these problems by introducing distributed real-time computing system in a humanoid robot with newly designed processor, Responsive Multi-Threaded Processor (*RMTP*)[1]. An actual installation will be made as an upgrade of HRP-3 inheriting HRP-2 mechatronics (Fig. 1) [2]. Processor nodes for servo, real-time communication, a real-time operating system based on Linux, and distributed robot software for perception and control are presented.

Figure 1.
Humanoid Robot, HRP-2

Height 1539 ㎜, Weight 58Kg, Number of joints 30 (6 DOF for each leg and arm), walking speed 2km/h.

[1] This project is supported by Japan Science and Technology Agency (JST) as a CREST project titled *Distributed Real-Time Processing for Humanoid Robots* in the field of *New High-performance Information Processing Technology.*

IEEE COMPUTER SOCIETY

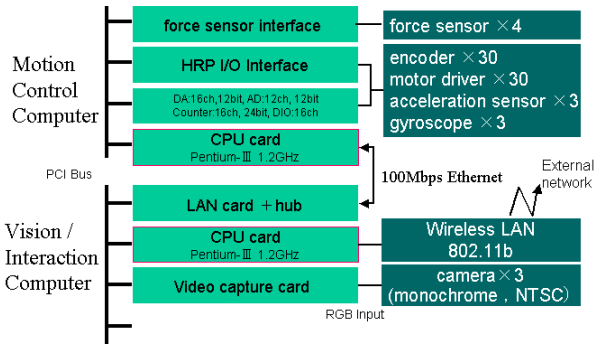

Figure 2. HRP-2 Computer Configuration

2. Problems of Humanoid's Computer System

Four micro-sparc processors piggybacked on the Honda P-2 realized its smooth biped walking [3]. HRP-2 developed by Kawada and AIST with other contractors in 2002 was controlled by two Pentium-3s. HRP-2 is estimated to utilize 20-times greater computing power than P-2. Although this computer power is enough for blind walk on a flat floor, capabilities for task planning and visual recognition are limited.

The processors of P-2 were connected on VME bus, while processors of HRP-2 on separate PC motherboards are connected via 100Mbps Ethernet (Fig. 2). HRP-2 has 30 actuated joints and 180 I/O cables to connect them to the central computer. Most cables run through narrow gaps in joints where heating motors sit. Under these circumstances, arrangement of the cables and connectors often causes troubles. Robot's continuous vibrations and sudden impacts occasionally make cables break. While usual acceleration is at most 1.5G when HRP-2 is walking, it reaches 10G when it falls down on the buttocks covered by an absorbing cushion.

Growing energy consumption of the processors is also a problem. Two Pentium-3 processors of HRP-2 running at 1.2GHz consume 60W while four processors of Honda P-2 consumed 20W. Pentium-4 at 3GHz consumes 80W, i.e. 288KJ per an hour. A humanoid robot can carry 2000-3000KJ of battery energy. To keep practical operational hours, the computer system can never be too conservative.

To summarize the above, the centralized control architecture of current humanoid robots suffers from shortage of computing power, inability of expansion due to physical and energy limitations, and reliability caused by crowded cables.

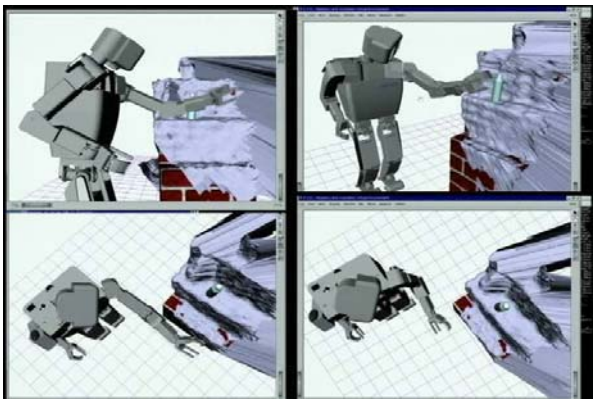

Figure 3. Motion planning to reach a bottle avoiding obstacles and its visualized simulation

3. Information Processing for Humanoid Control

We estimate computing power needed for improving humanoid behaviors discussing the state of the art of the motion planning, perception, real-time control, and communication.

3.1 Motion planning

In order for a humanoid robot to walk stably, footsteps to reach a goal and collision-free trajectories of all joints are computed before an actual action. At run time, each trajectory is traced and torque commands are sent to 30 actuators at 5ms interval. To keep track of the trajectory, sensory information from joint angle encoders, touch sensor on the foot bottom, and gyroscopes are monitored.

Vision and tactile sensors are used to build environment model, on which global and longer term planning is performed. The planning task is basically a search in a huge configuration space. If we divide each joint angle into 360, the search space formed by 30 joints ends up with 10^{77}, disabling a complete search. To choose natural solutions if not the best, heuristic approaches such as a genetic algorithm or a randomized planning must be taken into considerations. Figure 3 shows the result of a task planning to grasp an object without hitting obstacles.

Kagami [5] succeeded implementing an on-line biped motion planning, which controls the H-5 humanoid robot according to a simple joystick command. In there, footstep planning and torque pattern generation of 15 actuators in the lower body are computed within 100ms. The robot requires a complete flat floor since the motion plan is generated every second and any environmental change or external disturbance is not allowed for this duration. Replanning for changing situation is not allowed after a step motion starts. For more robust walk, 10 times faster planning is desirable.

IEEE COMPUTER SOCIETY

H-7 robot's longer term planner generates a sequence of footsteps in a particular gait pattern. A floor of 5m x 5m is divided into 256 x 256 tiles, each of which is assigned with the existence information of obstacles. 50,000 search branches are tested for one-meter walk spending 1 second on 1.4GHz Pentium-3 [6]. For a longer path planning, time roughly equivalent to the path length raised to the second power is needed.

3.2 Perception

Among five sensations, vision is the most useful to understand the environment. Although vision's response does not need to synchronize to the 5 ms interval of the joint trajectory samples, it is desirable to respond in the 33ms frame interval.

Vision sensing requires huge computing power. Unlike feature-based vision systems for object recognition, the depth map of environment is important for walking robots. A depth map is composed by binocular stereovision, which searches for corresponding pixels in right and left images. The match is computed by summing every absolute value of difference of pixel intensity in a window (SAD method). Depth map generation for a 320 x 240 image with a window size of 11 x 11 finishes in 20ms on 1.4GHz Pentium-3. If we use a 35mm semi-wide lens, 320 x 240 resolution corresponds to eye sight 0.1 (20/200). 35mm lens coverage is one fifth of human view at best.

Audio processing of a humanoid is used to find directions to the sound sources, to recognize spoken language, and to generate artificial voice. Among these, recognition is the heaviest. Current speech recognition technology can produce results almost in real time with a 1GHz Pentium-3. Recognition score trades off with the speed. Thus 1-2Gips would be required for humanoid's audio processing.

3.3 Servo control

HRP-2 uses two 1.2GHz Pentium-3 processors (Fig. 2). The first processor performs real-time servo tasks for the biped walk control, while the second processor is assigned to non real-time tasks such as vision and speech processing. Joint trajectory is referenced every 5 ms and the motor servo runs every 1 ms.

The shorter the servo cycle, the more stable and stiffer the control. All the humanoid robots so far do position control. Though compliance control is difficult because of the high gear reduction ratio, shorter control cycle and force sensing can give similar effect. We plan to speed up the servo cycle five times faster than the current HRP-2 to enable compliant and flexible motion. Joint angle reference will be generated every 1ms and the servo is performed at every 0.2ms.

3.4 Communication and system management

The operating system is requested to provide the following functions: communication and synchronization between tasks, I/O device drivers, resource management, exception handling, file systems, etc. Since users of these services are real-time tasks, these services also need to guarantee to finish within predictable periods.

The operating system of the H-7 humanoid robot, which is a prototype to HRP-2, bases upon RT-Linux and HRP-2 on ART-Linux [12, 13]. Both operating systems are extensions of Linux toward real-time control to wake up kernel tasks (RT-Linux) or user processes (ART-Linux) according to external events (RT-Linux) or in synchronization with fine grained timer intervals (ART-Linux). In order to accomplish 0.2ms servo synchronized over the whole body, communication latency should be maintained under 0.1ms at the system software layer, and 0.05ms at the hardware communication level.

4. Distributed Real-Time Humanoid Control

The above considerations lead to the following conclusions:

1. For low-level servo, synchronized task control within 0.2ms over dozens of joints is required.
2. For high-level recognition and planning tasks, computing power equivalent to several 3GHz Pentium-4 is required. However, since these are open problems, extensibility (scalable computation power) is important.
3. Maximum power affordable to processors is limited to 100W.
4. Reduction of signal wire length is necessary for higher reliability.

To cope with these statements, we propose a distributed control architecture based on multi-threading and a real-time network. Distribution enables addition of computation nodes, i.e. scalable computation power. Inherent parallelisms in image processing and geometric computation can take advantages of multithread and vector processing. Reliability can be improved by replacing crowded analog signal wires with a simple digital network connecting computation nodes placed in the vicinity of joints.

While centralized control could enjoy low latency communication between multiple tasks under the control of the single operating system sharing memory, distributed control demands real-time communication and real-time task scheduling among processors over the network. Since it seems very difficult to achieve these functions with existing processors and networks,

IEEE COMPUTER SOCIETY

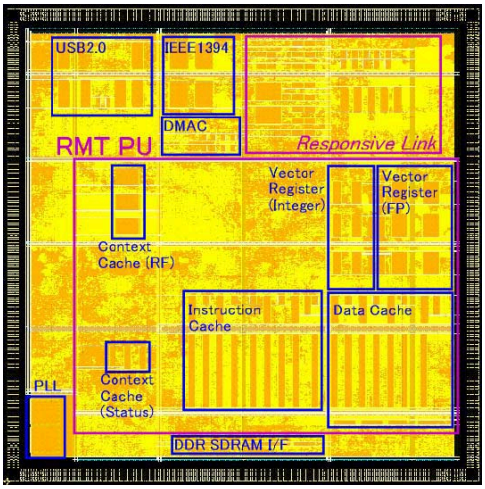

Figure 4. Responsive Multi-Threaded Processor (RMTP). 130 nm, CMOS 8 layers, Cu wiring, approx. 10 million gates, 10x10 ㎜, 1000 pins

we have designed a new processor with real-time communication capability.

As for a magnitude of distribution, we can take either per-joint processor architecture or per-limb architecture. In other words, number of distributed processors can either be 30 or 5. We chose per-joint distribution to minimize the signal wire length and to maximize redundancy. In this architecture, each peripheral processor is integrated with a motor driver interfacing to an encoder and to an amplifier.

5. Processor Design

5.1 RMT Processor (RMTP)

We designed two types of new processor chips of the single architecture; one for servo and the other for planning and recognition. The former is called *micro-RMTP*, which focuses on the conservation of space and power. The latter is *HP-RMTP*, which focuses on processing power. *RMTP* is the abbreviation for *Responsive Multi-Threaded Processor* [8,10].

Micro-RMTP and *HP-RMTP* are based on the common *RMT* architecture (Fig. 4)[2]. Its instruction set is compatible with MIPS-II and includes some of MIPS-III, and extends to support vector and SIMD instructions that operate on 512 vector registers.

The most prominent feature of RMTP is that up to eight prioritized threads run in parallel in a single CPU.

[2] The RMT processor was developed by Keio University in the distributed real-time network project (2000-2004) subsidized by Ministry of Education of Japan.

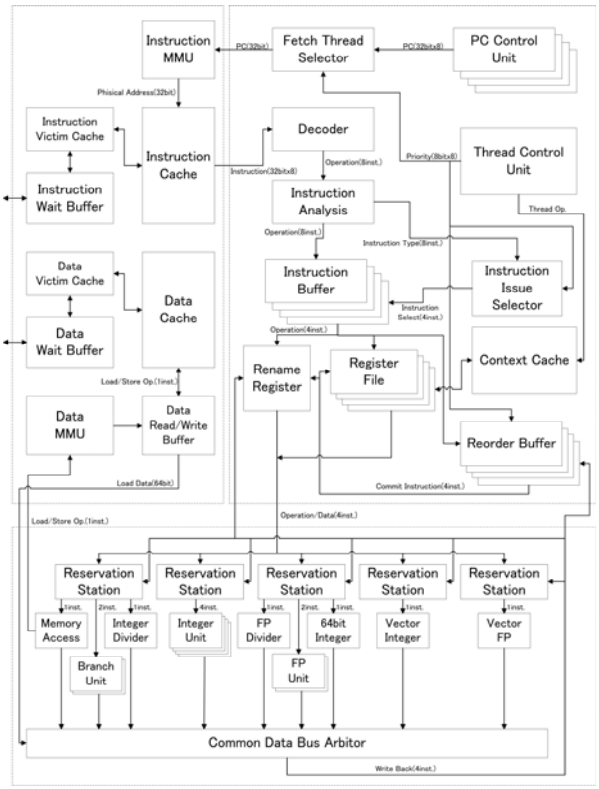

Figure 5. RMTP architecture

In addition to the eight active threads, 32 threads are maintained in the thread context cache, enabling thread switching in 4 clocks. Four instructions out of above 8 threads are issued simultaneously.

Instructions from eight threads are chosen by the thread control unit and the instruction issue selector according to thread scheduling policy described in the next section. Ready instructions are buffered in the reservation stations waiting for operands (Fig. 5). When operands become ready, they are sent to one of the fourteen execution units. If the instruction is a vector instruction, the integer vector unit executes 8 operations and the FP vector unit executes 4 operations in parallel. The peak performance of RMTP is described in Table-1. When all the parallel computation gadgets work simultaneously, RMTP outperforms Pentium-4.

Since above performance is mostly brought by parallel computation, the clock speed can hold back to 100-300MHz, reducing the power consumption down to 8W at maximum. Further low-power operation by cutting off power supply to idling units and dynamic clock control is available as well.

For real-time communication with external nodes, one RMTP has four *Responsive Links.* Besides a wide-band memory interface, a number of I/O interfaces, such as PCI bus slave, IEEE-1394 video layer, PWM

IEEE COMPUTER SOCIETY

Table-1 Peak computing performance of RMTP@300MHz

Scalar Integer	1.2 GIPS
Scalar Floating point	0.6 GFLOPS
Vector Integer	9.6 GIPS
Vector Floating point	4.8 GFLOPS

generator, PWM interpreter, etc. are implemented on-board [8, 9, 10].

5.2 Multi-thread scheduling

RMTP performs multi-thread scheduling by the hardware, thus eliminating an intervention of the operating system at task switching. The system can choose one of three scheduling policies: dynamic prioritized scheduling, high performance scheduling, and fixed slot scheduling. In the dynamic prioritized scheduling, tasks with higher priorities are favored: the task with the highest priority can use multiple execution units simultaneously. The high performance scheduling tries to use as many execution units as possible: at each cycle, each thread can get the equal favor. The fixed slot scheduling allocates fixed number of execution units to specified tasks. The fixed slot scheduling is hard-real-time oriented since it can guarantee the allocation of processing units to the highest priority tasks. For most applications, the dynamic priority scheduling is a good compromise both for real-time processing and total throughput.

5.3 Responsive Link (RL)

RMTP has real-time and high speed communication links, called *Responsive Link*[3] [9, 10]. We proposed the *Responsive Link* in 1998 and released the first ULSI implementation, *Responsive Processor* with the *Sparc* core in 1999. The *Responsive Link* has the following features:

- Each link is composed of two channels: high-throughput *Data channel* with 64-byte packets and low-latency *Event channel* with 16-byte packets (both are bi-directional).
- Every packet is assigned with a priority, which controls packet overtaking at *RL* routers in *RMTP*.
- Packets of different priorities can take different paths.
- A packet has per-byte ECC for 1-bit error correction and 2-bits error detection.
- The maximum speed is 800MHz and applications can choose lower speeds for lower power operation.

[3]The Responsive Link and the Responsive Processor have been developed by Electrotechnical Laboratory, former AIST. The protocol is proposed as an international standard to ISO JTC-1 SC25/WG4.

Since each RL router can perform routing by reading the first 32 bits in a packet, delay in one stage routing is 480ns at 100Mbps, and overall communication in an entire humanoid body finishes in 5 micro second for a 8-byte payload on the Event channel. Hundreds of *RL* nodes can configure a low-latency and high-bandwidth network.

5.4 Micro RMTP Node

As depicted in Fig. 6, a *Micro-RMTP* is integrated in 30x80x10mm space including power amplifiers so that it is attached to a motor. The *micro-RMTP* will be packaged on SIP (system-in-package) with the memory subsystem to reduce number of pins for external interconnection.

Micro RMTP's main task is the motor servo, which performs frequent I/O but is not accelerated by parallel or vector computation. However, as member nodes of the total humanoid system, these small processors execute parallel computations for recognition and planning allocated by the HP-RMTP nodes.

The specifications of the micro *RMTP* follows.

1. Design rule: 130 nm
2. Size: die 10x10 ㎜, SIP30 x 45 mm
3. Power : core 1V, memory 2.5V, I/O 3.3V, total power 1W
4. CPU performance: 133MHz, eight threads
5. On-chip I/O
   (i) *Responsive Link* x 4
   (ii) encoder counter and PWM generator x9
   (iii) PWM interpreter x 3
   (iv) Serial 4 ch
6. On SIP functions
   (i) memory: 64MB 32bit DDR SDRAM without ECC
   (ii) Flash ROM: 16 MB
7. On-board functions
   (i) ADC: 12bit x 8ch
   (ii) DAC: 12bit x 8ch
   (iii) Current sensor x 3
   (iv) Motor drive amplifier x 1

5.5 HP-RMT Processor（HP-RMTP）

HP-RMTP will be designed in 2005 with finer design scale. Clock speed will be 3-500MHz to provide higher performance together with higher parallelism. We expect at least twice faster performance than our current *RMTP*. *HP-RMTP* will implement on-chip IEEE-1394 and PCI-bus interfaces to connect digital TV cameras and other devices.

6. System Software for Real-Time Control

The operating system of the RMTP is based on Linux-2.6 for common programmers' benefits and to

IEEE COMPUTER SOCIETY

allow easy porting of CORBA. So far, we have confirmed booting of Linux on the clock level simulator and on the instruction level simulator.

Since RMTP supports the simultaneous execution of multiple threads, the low-cost thread switching, the priority-based multi-thread scheduling, and the priority-based real-time communication network, the operating system does not need to implement sophisticated scheduling algorithms. However, in order to avoid sporadic insertion of irresponsible time by interrupts and device drivers, interrupt generation is limited to a timer running at relatively short ticks. Device drivers run as normal threads, not in the interrupt contexts.

In addition, a deadline scheduling is applied to hard real-time tasks. Although it is well acknowledged that robot control requires real-time scheduling, conventional robot software only relies on cyclic task invocations, that is, the capability to start a task at a specified time, not to finish by the deadline. A challenge of our new operating system is to precisely implement this deadline scheduling and to impose robot applications to specify the deadline.

To make the deadline scheduling possible, the system has to be able to estimate a time required to finish a given task. For this, we are developing a static WCET analysis tool referring to the RTL representations in the GNU Compiler Collection. Modeling of the cache behavior is also incorporated. Accordingly the operating system bounds the time required for every system call. Programmers of hard real-time tasks are requested to provide the worst case parameters such as the maximum values of loop counts.

7. Conclusion

In order for the real-time distributed control of a humanoid robot, *RMTP* will be attached to each of 30 joints and connected by the *Responsive Link* network. The humanoid robot will be controlled in the 200 micro second servo loop which traces trajectory points generated every 1 ms based upon the motion plans updated every 100 ms. Thread management of the Linux-2.6 based real-time operating system is facilitated by the hardware multithread scheduler of *RMTP*. WCET analysis is used for hard real-time programming. As results, coming HRP-3 humanoid robot will be able to perform tasks with force control, irregular terrain walking, whole body motion, and quick recovery from communication failures.

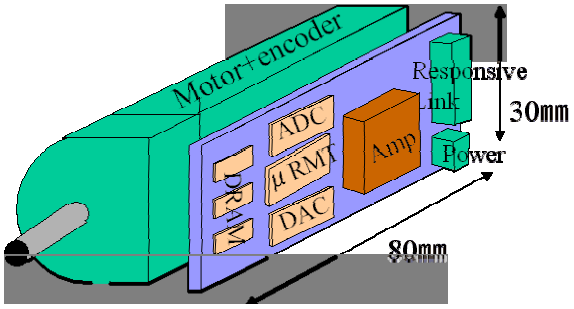

Figure 6 μRMTP node integrating I/O and power amplifiers attachable to a motor

Acknowledgement

Authors express sincere gratitude to Professor Hidehiko Tanaka and other advisors of the new high performance computing field, CREST, JST.

Reference

[1] K. Hirai, "Current and Future Perspective of Honda Humanoid Robot," Proc. IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS-97), pp. 500-508, 1997.

[2] T. Isozumi, K. Akaike, M. Hirata, K. Kaneko, S. Kajita and H. Hirukawa, "Development of Humanoid Robot, HRP-2," Journal of Robotics Society of Japan (in Japanese), Vol. 22, No. 1, 2004.

[3] http://www.honda.co.jp/robot/

[4] Hans Moravec, Robot: Mere Machine to Transcendent Mind, Oxford University Press,1998.

[5] Satoshi Kagami et al., "A Fast Dynamically Equilibrated Walking Trajectory Generation Method of Humanoid Robot," Autonomous Robots 12, pp. 71-82, Kluwer Academic Publishers, 2002.

[6] Joel Chestnut, James J. Kuffner, Koichi Nishiwaki, Satoshi Kagami, "Planning Biped Navigation Strategies in Complex Environments," IEEE Int. Conf. on Humanoid Robots (Humanoids2003), Oct, 2003

[7] Kanehiro, Fujiwara, Kajita, Kaneko, Hirukawa, Nakamura and Yamane, "Humanoid Robot Software Platform, OpenHRP," Journal of Robotics Society of Japan (in Japanese), Vol. 21, No. 7, pp. 89—97, 2003.

[8] http://www.ny.ics.keio.ac.jp/

[9] Nobuyuki Yamasaki, "Responsive Processor for Parallel/Distributed Real-Time Control," in Proc. Of 2001 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS-2001), pp. 1238-1244, 2001.

[10] Nobuyuki Yamasaki, "Responsive Multithreaded Processor for Distributed Real-Time Systems", Journal of Robotics and Mechatronics, Vol. 17, No. 2, pp. 130-141, 2005.

[11] http://www.itscj.ipsj.or.jp/ipsj-ts/02-06/toc.htm

[12] http://rtlinux.lzu.edu.cn/index.html, http://www.fsmlabs.com/index.html

[13] S.Kagami, K.Nishiwaki, J. Kuffner, K.Okada, Y.Kuniyoshi, M.Inaba, H.Inoue, "Low Level Autonomy of the Humanoid Robots H6&H7", Robotics Research, R.Jarvis and A.Zelinsky (Eds), Springer, pp. 83--98., 2003.

IEEE COMPUTER SOCIETY