Developing Robot Motions by Simulated Touch Sensors

Fabio Dalla Libera[1], Takashi Minato[3], Hiroshi Ishiguro[2,3] Enrico Pagello[1], and Emanuele Menegatti[1]

[1] Intelligent Autonomous Systems Laboratory, Department of Information Engineering (DEI), Faculty of Engineering, University of Padua, Via Gradenigo 6/a, I-35131 Padua, Italy
[2] Department of Adaptive Machine Systems, Osaka University, Suita, Osaka, 565-0871 Japan
[3] ERATO, Japan Science and Technology Agency, Osaka University, Suita, Osaka, 565-0871, Japan

Abstract. Touch is a very powerful but not much studied communication mean in human-robot interaction. Nonetheless many robots are not equipped with touch sensors, because it is often difficult to place such sensors over the robot surface or simply because the main task of the robot does not require them. We propose an approach that allows developing motions for a real humanoid robot by touching its 3D representation. This simulated counterpart can be equipped with touch sensors not physically available and allows the user to interact with a robot moving in slow-play, which is not possible in real world due to the changes in the dynamics. The developed interface, employing simulated touch sensors, allows inexperienced users to program robot movements in an intuitive way without any modification of the robot's hardware. Thanks to this tool we can also study how humans employ touch for communication. We then report how simulation can be used to study user dependence of touch instructions assuring all the subjects to be in exactly the same conditions.

Key words: touch, robot teaching, motion development, simulation

1 Introduction

Observing how dance [1] or sport instructors teach motions, we can note that, with simple touches, the teacher intuitively conveys plenty of information on how to modify the trainee's movement. Touch is particularly appealing as an intuitive method for humans to interact with robots, and has been employed to program robot arms [2] [3] and humanoid robots [4]. It then seems plausible to use touch to develop motions for humanoid robots, and we therefore aim at studying how touch can be employed for human-robot communication. Many humanoid robots are available on the market for an affordable cost, but usually these devices are not equipped with touch sensors. The most straightforward solution would then be to customize the robot by covering it with tactile sensors.

However several difficulties arise, first of all because the humanoids available on the market are quite small and the wiring becomes complex. If the sensors provide an analog output, (multiplexed) A/D converters must be employed and buses with sufficient bandwidth must be designed. These problems had been tackled in several works, for instance in [5]. If, as in [6] air actuators are employed it is possible to read the error between the target position and the actual position to estimate the force applied by the user. Though air actuators are ideal for the human-robot interaction because of their compliance, their control is very difficult (the response is highly non-linear) and are not usually available on off-the-shelf humanoid robots. One alternative solution would be to use a shadow robot. This technique, [7], consists in having two identical robots, placed in the same position. The user interacts with one of the two robots, and comparing the torques with the ones of the second robot it is possible to distinguish the force applied by the user from the other forces (gravity, friction forces, etc.). Though interacting with a physical robot is probably more intuitive, simulating the touch sensors is a very cost effective solution to allow tactile interaction with a robot. Furthermore, it is possible to simulate sensors not currently available with the current technology in terms of size, bandwidth, signal to noise ratio, etc. Then this technique is general, applicable to any kind of robot. Interacting with a virtual world also allows to view the robot's movement in slow play or stop the motion with no effect on the dynamics, something not feasible in the real world (for instance, slowing down a jump motion in the real world to better observe the motion execution is not possible). In the simulated world additional information can be easily displayed, for example the zero moment point [8]. As a drawback much information measurable by advanced touch sensors, such as the intensity and the direction of the applied force, cannot be obtained by virtual touch sensors which are simulated, for instance, by mouse clicks. We therefore must assume that the user's touch direction is normal to the touched surface. In the case a standard mouse is employed the user cannot either touch multiple sensors simultaneously, as would be possible employing a real robot. We present here results obtained using simulation at different levels. First we will show how it is possible to simulate the touch sensors while employing the real robot to obtain the motion dynamic (therefore preventing any simulation-reality gap). In this case simulation is used to provide a sort of "augmented reality" that enhances the existing robot by providing touch sensors and allows us to study touch interaction. We then show an application of simulation to analyze user dependence of touch instructions in human-robot interaction. The main advantage in using simulation when studying user dependency is that simulation allows all the users to be in exactly the same conditions (same robot, same calibration, same friction of the floor) and to run the experiment in parallel (multiple PCs are more easily available than multiple robots). Obviously simulation also allows a safer interaction, both for the user and the robot. In section 2 we introduce the idea of extending the robot capabilities by creating virtual models equipped with sensors not available on the real robot, and present in detail the simulation of touch sensors. In section 3 we describe a possible algorithm to develop

motions using touch. In section 4 we present an example of how simulation was employed to conduct tests on user-dependence of touch instructions. In section 5 we present some preliminary experimental results and in section 6 we conclude summarizing the ideas presented in the paper.

2 Developing Motions With Virtual Sensors

Recently many robots, and in particular small humanoid robots actuated by servomotors, are becoming available on the market for a lower and lower cost. One of the main issues when dealing with these robots is the development of robot movements, which is difficult given the high number of joints (usually around 20). Robot movements are often developed employing slider-based interfaces that require the user to define the robot motion as a set of keyframes, that is, a set of instants in time for which the position of each and every joint is provided.

This approach is very time consuming so many alternatives for automatic motion generation had been proposed in literature [9–14]. Nonetheless handcrafted motions are still much diffused [15, 16]. It appears then necessary to devise intuitive ways to program robot motions by scratch, and we decided to rely on touch interaction with the robot. Using touch to develop motions has advantages in terms of intuitiveness for unexperienced users, but obviously for tasks for which optimization criteria exist specifically devised algorithms can obtain better performances.

Small humanoid robots available on the market are usually not provided with touch sensors, but as stated in the introduction, these sensors can be simulated, and the user can interact with a simulated robot equipped with such sensors. In detail we can imagine the following development cycle, depicted in figure 1:

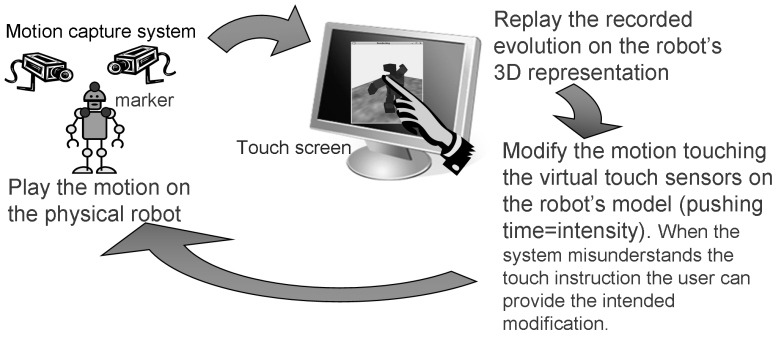


Fig. 1. Phases of the motion development.

1. the robot moves in the real world. Its orientation in the world is captured with a motion capture system and recorded;

2. users watch the recorded motion evolution in a virtual world.
3. users choose an instant in time where the motion should be modified and touch the virtual touch sensors to modify the posture at that time.
4. the motion is modified given the user instructions
5. if the obtained motion is not satisfactory, the development cycle is repeated

The trial and error development process of classical slider-based interfaces is therefore maintained, but instead of requiring the user to set each joint by a slider with the proposed approach she/he is able to develop the motion just by touching the robot. Using simulated touch sensors even eases the task with respect to an interaction in the real world, since the movement can be watched in slow-play or stopped. In fact we imagine the user to select an instant in time, stop the execution, touch the robot, eventually choose other instants where the motion should be edited, change the posture by touching the virtual sensors and then play the modified motion on the real robot. In the realized system the virtual sensors can however be touched during motion playback as well.

As previously stated the robot moves in the real world, and we acquire its position and orientation by a motion capture system. To acquire this information three markers placed on the robot's torso revealed to be sufficient. Given the position and orientation of the robot in the world we are able to reconstruct the motion evolution afterwards, since knowing the motion and a model of the servomotors response it is possible to calculate the angle of each joint. If the servomotors support position reading, accurate joint angle information could also be recorded during the motion evolution. However approximated joint angles are sufficient, since their value is only needed to show the user the robot's posture in the virtual world.

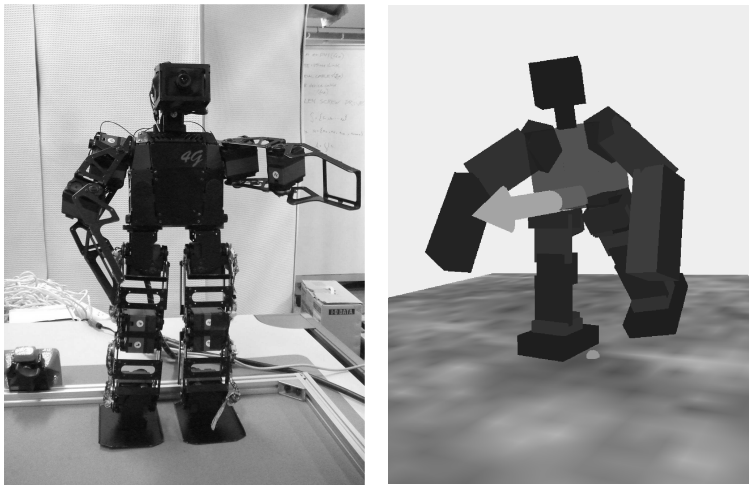
Once data are collected they can be used to replay the movement in a virtual world. Using data from execution of the motion in the real world frees us from the necessity of simulating the robot's dynamic. Very approximate robot models can be employed, as long as the simplification does not prevent user-robot interaction. The capabilities of the robot can be extended in the virtual world, providing it with touch sensors, noiseless gyroscopes, virtual cameras and so forth. Additional information can also be displayed, for instance in the developed interface the projection of the (approximated) center of gravity on the floor and the velocity vector of the center of gravity are displayed[4]. Fig. 2(b) depicts a view of the virtual 3D world as rendered by the interface. The robot links are simplified by parallelepipeds, and each face simulates a touch sensor.

These simulated sensors can be clicked with the mouse, allowing the user to "push" the robot links by clicking. Since with conventional devices as mice or touch screens it is not possible to measure the applied force, the output is binary and the interpretation of the intensity in the developed interface is based on the pushing time. Another limitation in employing conventional devices is that just one sensor at time can be clicked. This problem can be solved assuming that

[4] These visual hints are introduced to allow expert users to improve the motion performance. They are not essential and thus to realize a model for the presented touch interaction interface no information on the mass distribution is actually required.

if the clicks occur when the robot motion is stopped than all clicked parts are touched together.

To give the user feedback while pushing the robot parts haptic devices could be employed. While this would probably make the interaction more natural, for simplicity we chose to provide visual feedback, i.e. the sensors gradually change color from green to red while being pushed.



(a) (b)

Fig. 2. The robot employed in the experiments and its simplified 3D representation. The projection of the center of gravity onto the ground (represented by a sphere on the ground) and its velocity (represented by an arrow) are given as additional information.

3 Interpreting Touch Instructions

Employing touch to develop motions seems straightforward, in particular if we observe human-human interaction. Nonetheless decoding the meaning of touch instructions is not trivial. Often no direct mapping from touched part to modified joint angle is perceived as natural, intuitive by the users. In fact, this one-to-one mapping would be quite similar to the one used in classic slider-based interfaces. Furthermore while with existing methods like teaching playback or compliant joint control the robot moves passively and the user always need to apply forces to the robot, what happens with humans and what we aim to obtain is to have the robot gradually interpret touch meaning and understand how to modify its motion.

One of the issues to be tackled when dealing with touch instructions is the strong context dependence of the touch meaning. For example if the robot is

standing, touching the upper part of one leg could mean that the leg should bend further backwards. However if the robot is squatting, the same touch could mean that the robot should move lower to the ground by bending its knees (see Fig. 3).

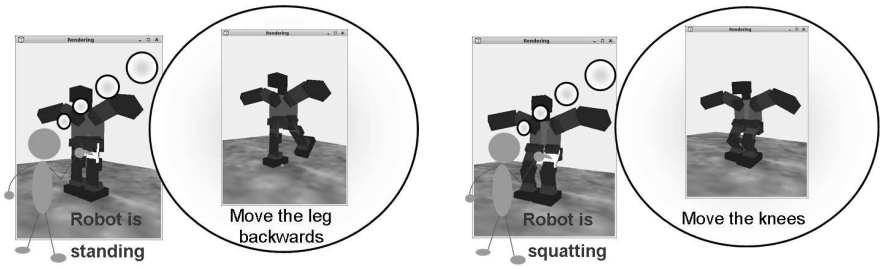


Fig. 3. Context dependence of the meaning of touch instructions. The user touches the robot in the same way, but the desired posture modification (bend the leg and bend the knees, respectively) is different because the robot posture is different.

We can then easily imagine user dependence on the meaning of touch instructions, since if no protocol is fixed different users will tend to touch the robot differently to give the same instructions. Avoiding to force the user to employ a certain protocol can enhance the intuitiveness of the interface. Suppose a user has a desired posture modification she/he would like to apply (for instance, raise the leg). If the protocol is fixed the user must identify which sensors, according to the touch protocol, should be pushed to have the desired modification. On the other hand if the system is capable of adapting to the user and estimate his/her intention, it is sufficient for the human operator to touch the robot spontaneously with no mental effort.

One simple way to have an interface which is able to adapt to the user is to ask the user to provide examples of the mapping between touches and corresponding posture modifications and use a supervised learning technique. The role of the learning algorithm is to realize a mapping between the tuple $(touchinformation, context)$ and expected intended modification of joint angles. Currently the context consists of the posture of the robot (represented as the angle of each of the joints) the orientation (roll, pitch and yaw) of the robot's torso, and the velocity of the center of gravity. The posture is needed because the meaning of touches may depend on the posture, as in the provided example in which touching the lap means different things depending on whether the robot is standing or squatting. Likewise, the meaning of the instructions may vary depending on the orientation, for instance the meaning could be different if the robot is standing or is lying down. Finally, touch meaning could also depend on the velocity, especially if the robot is moving fast, for example if it is falling down.

Given the limited number of examples compared to the dimensions of the input space, among the numerous available supervised learning algorithms, like neural networks or Gaussian Mixture Models, we decided to employ k-Nearest Neighbor with a specifically devised metric. While this paper focuses on the simulation aspects the details on the metric and how it was derived are provided in [17]. Briefly each example provided by the user consists of an input I_i and an associated intended joint modification vector M_i (that we assume to be the class to which it belongs). Given an input I_*, the system output vector M_* can be obtained by weighting the joint modifications present in the collected examples M_i, with weights ω_i calculated employing the distance (in the high dimensional space) between the system input I_* and each example coordinates I_i. Concretely, indicating with E the number of collected examples

$$M_* = \sum_{i=1}^{E} \omega_i M_i \tag{1}$$

Directly employing k-Nearest Neighbor with Euclidean or Mahalanobis distance based weights presents two problems. First of all touch information (pushing time of each of the sensors) should be prioritized over the context. This is to avoid the output being determined mainly by the context instead of by the pushed links, as would happen if touch information is given no priority over the other features of the input (i.e. if the input vector I_i components are all treated equally). As a trivial example, suppose the human operator designed an arm motion and therefore only provided examples involving the arm, then when she/he will push the legs this will cause the arm to move, while in such cases of no available knowledge it would be intuitive not to apply any posture modification.

To solve this problem given an input vector I_* and in particular the touching information T_*, the output M_* is calculated considering only the examples having a set of pressed sensors (i.e., sensor having a pushing duration greater than zero) the same set of sensors pressed in T_* or a subset of them. In other words, indicating with n the number of sensors and with the notation $T_*[s]$ and $T_i[s]$ as the pushing duration of the s-th sensor in the system input T_* and in the i-th example touch information vector respectively, the i-th example is considered if and only if

$$\bigvee_{s=1}^{n} (T_i[s] > 0)) \wedge (T_*[s] = 0) \tag{2}$$

is false. In other terms $\omega_i = 0$ in equation 1 for the examples in which equation 2 holds.

The second problem arises because every distance function is symmetric. Suppose to have just one training example, where a sensor was pushed for 300 milliseconds, and this corresponded to a desired modification of increasing a certain joint angle by 40 degrees. A user might naturally expect that pushing for less time will cause a smaller change in that joint, while a longer press should produce a larger joint angle change. Nonetheless the system behavior with a

distance based weighting is such that any touch on that sensor with duration different from 300ms, either longer or shorter, results in a smaller angle change. For example if $\omega_i = 1/\left(1 + \|I_* - I_i\|\right)$ is used as a weighting function pressing the sensor for 200ms or for 400 ms would give the same modification. To avoid this unnatural behavior the weight ω_i (see Eq. 1) is calculated as $\omega_i = \alpha_i \beta_i$. Given T_*, the touch information components of the input vector, and the various example touch information vectors T_i, α_i is calculated as

$$\alpha_i = \prod_{s:T_i[s]>0} T_*[s] / T_i[s]$$

This value keeps increasing linearly as the pushing time increases. The second factor β_i accounts for all information not used in the calculations of α_i

- the sensor information $T_*[s]$ and $T_i[s]$ for the sensors s such that $T_i[s] = 0$;
- the joint angles of the robot in the system input (P_*) and the angles recorded in the i-th example P_i;
- the orientation present in the system input O_* and the one of the i-th example O_i;
- the center of gravity velocity vectors V_* and V_i, relative to the system input and to the i-th example respectively.

It was chosen to calculate each β_i as

$$\beta_i = \frac{1}{1 + d_i} \tag{3}$$

where d_i provides a measure of the diversity of the current input I_* and the i-th example input I_i. Denoting the Euclidean norm by "$\|\|$", d_i is given by

$$d_i = \sqrt{\sum_{s:T_i[s]=0} (T^*[s] - 0)^2 + \|P - p_i\|^2 + \|O - o_i\|^2 + \|V - v_i\|^2}$$

where each vector component is normalized scaling by its variance in the example data set since the units are heterogeneous.

The structure of Eq. 3 emerges from practical experiments: several decreasing functions were tested and the one which appeared to give the most intuitive behavior, $f(x) = 1/(1 + x)$ was chosen. A deeper and more formal analysis will be conducted in future works.

It must then be decided how to acquire the examples of the mapping used by the algorithm. We chose to collect them on-line, during the development of robot motions. This brings two advantages. First of all no special session where the user is required to provide how she/he would touch the robot to express certain pre-defined modifications is required. Secondly the human operator can identify when the system fails to predict her/his intention, and can provide, by the shared protocol, the intended joint modification, so that the mapping between touch instructions and estimated modification intentions can be refined where it needs to be. Ideally the system keeps improving its knowledge base

during the motion development and users need to teach the meaning of the touch instructions less and less frequently.

It is therefore required to provide a method for the user to communicate the desired posture modification when the system does not estimate the intention correctly. In the current implementation a classical slider based interface was used.

4 Assuring Identical Conditions

As previously stated we can suppose user dependence of the meaning of touch instructions. In detail the same intended posture modification could be expressed in with different touch instructions and the same touch could have different meanings for different users even within the same context. In order to compare the instruction provided by different users we should put all of them in exactly the same conditions, otherwise differences in the results could derive by those factors. In detail we would need the users to employ identical robots, assure the same servomotor calibration, the same motion capture calibration, the same friction of the ground surface and so forth. These differences can be overcome using a simulator. In this case, the behavior of the robot is identical for any execution and every user. If identical PCs are used the experiment can also be run in parallel with different users assuring them to be in exactly the same condition. Even if a simulator is used to replace the real robot, the development cycle described in section 2 does not need to be altered. In detail, the users keep developing the motion using exactly the same interface employed for the control of the real robot and the simulator can provide a virtual motion capture system that sends the information to the interface simulating an ideal motion capture system that return the exact position of virtual markers.

5 Experiments

Experiments had been conducted using Vision4G, a humanoid robot produced by Vstone[5]. This robot is 445mm high and has 22 degrees of freedom actuated by DC servomotors (Fig. 2(a)). For capturing the robot's position we used the Eagle Digital System developed by Motion Analysis Corp. The robot's 3D representation, used to provide the robot with virtual touch sensors, is presented in figure 2(b). Strong simplifications had been introduced, for instance parallel links, present in the robot legs, had been modeled by a single link actuated by two motors that rotate synchronously. For the reconstruction of the motion the response of the servomotors had been approximated by a simple delay of 200 ms. We'd like to recall that the interface does not provide any dynamics simulation, and the position and orientation of the robot for each time instant is calculated interpolating the motion capture system data.

[5] `http://www.vstone.co.jp/`

As a first validation of the feasibility of the developed interface, a stand up motion, a jump motion[6] and a walking motion were successfully realized. In detail the jump motion was realized both with the touching approach and with a classical slider-based interface for comparison. Similar motions were obtained, respectively, in 17 minutes and in over 40 minutes. Though this is just a preliminary results this can provide support to the thesis that motion development time can be reduced introducing touch-based interface. The examples of the mapping between touch instructions and posture modification provided by the user were studied. Analysis of the collected data is presented in [17].

User dependence was investigated asking six subjects to develop the same motions (a walking and a kicking motion) using the interface connected to a simulator. All the subjects are all Italian male computer science students, and their age is in the range 23-27 (mean 24.5, standard deviation 1.87). The simulator was developed using ODE, an open source library designed to simulate rigid body dynamics[7]. Figure 5 shows a screenshot of the rendering provided by the simulator. The robot had been modeled by 31 rigid bodies, each of which consists of one or more parallelepipeds(totally 39) linked by 34 joints.The number of joints is higher than the number of DOFs for the presence of parallel links, which in this case has been modeled directly as free hinge joints. The inertia matrix of each rigid body was calculated using the following approximations:

- each parallelepiped has uniform density and weights 35g;
- the real position and weight (63g) of the servomotors was identified and the density inside each servomotor was assumed constant;
- the robot's weight not accounted in the previous terms was assumed to be located in the robot's torso, uniformly distributed.

The main finding is that different users gave different abstraction levels in providing touch instructions:

- a nearly fixed mapping from a small set of sensors to the joints; the context has little or no influence
- a mapping based on physical considerations (the joints are imagined to be "elastic"); in this case, the context, for instance the position of the ground, becomes crucial
- a very high level representation of the motion, where for instance just the limb that should be moved is indicated by touching; at this level of abstraction a single touch corresponds to a motion primitive.

As previously stated since a simulator was employed the differences in the meaning of touch instruction is guaranteed to be due to the user-dependence of the mapping, and not by different environmental conditions during the tests.

[6] To ease the task a rubber-band pulling the robot from the top was employed. Details and pictures are reported in [17].

[7] See `http://www.ode.org/` for more information on this library.

Fig. 4. Screenshot of the rendering provided by the simulator.

6 Discussion and Future Works

Touch is a very intuitive mean of communication, and can be used in human-robot interaction for teaching motions, first of all for tasks like dance or sport movements for which it is difficult to provide a mathematical definition of the performance and for which, therefore, a user evaluation and tuning is very important. However most of the available robots are not provided with touch sensors. One cost effective solution, presented in this paper, consists in interacting with a 3D representation of the robot which extends the robot capabilities simulating sensors not present on the real one. We then presented an application of simulation on studying user dependence of touch instructions. In this case simulating the robot allows us to study the differences in the teaching method of different users when they interact in exactly the same conditions with a (simulated) humanoid robot.

Future works will aim at making the interaction more direct and natural. For instance, we can imagine to employ more advanced virtual reality devices. Another limitation of the current approach is that touch instructions are interpreted just observing the physical context, while we can imagine that knowledge of the task could be exploited to improve the meaning estimation.

Finally while in this work the user defines the target of the robot's motion and the performance of the result is evaluated by the user's subjective criterion, to allow a better comparison with other works definition of a set of measurement other than the development time here employed should be considered.

References

1. Takeda, T., Hirata, Y., Kosuge, K.: Hmm-based error recovery of dance step selection for dance partner robot. In: ICRA, IEEE (2007) 1768–1773

2. Voyles, R., Khosla, P.: Tactile gestures for human/robot interaction. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems. (August 1995)
3. Grunwald, G., Schreiber, G., Albu-Schffer, A., Hirzinger, G.: Touch: The direct type of human interaction with a redundand service robot. In: IEEE Int. Workshop on Robot and Human Interactive Communication RO-MAN '01, Bordeaux/Paris, France. (2001)
4. Yoshikai, T., Hayashi, M., Ishizaka, Y., Sagisaka, T., Inaba, M.: Behavior integration for whole-body close interactions by a humanoid with soft sensor flesh. In: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids 2007), Pittsburg, USA. (December 2007)
5. Ohmura, Y., Kuniyoshi, Y., Nagakubo, A.: Conformable and scalable tactile sensor skin for curved surfaces. In: ICRA, IEEE (2006) 1348–1353
6. Minato, T., Yoshikawa, Y., Noda, T., Ikemoto, S., Ishiguro, H., Asada, M.: Cb2: A child robot with biomimetic body for cognitive developmental robotics. In: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids 2007), Pittsburg, USA. (December 2007)
7. Katsura, S., Ohnishi, K., Ohnishi, E.: Transmission of force sensation by environment quarrier based on multilateral control. **54**(2) (April 2007)
8. Vukobratovic, M., Borovac, B., Potkonjak, V.: Zmp: A review of some basic misunderstandings. Int. J. Human. Robot **3**(2) (2006) 153–175
9. Nakaoka, S., Nakazawa, A., Yokoi, K., Hirukawa, H., Ikeuchi, K.: Generating whole body motions for a biped humanoid robot from captured human dances. In: ICRA, IEEE (2003) 3905–3910
10. Okumura, Y., Tawara, T., Endo, K., Furuta, T., Shimizu, M.: Realtime zmp compensation for biped walking robot using adaptive inertia force control. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2003). Volume 1. (2003) 335–339
11. Furuta, T., Yamato, H., Tomiyama, K.: Biped walking using multiple-link virtual inverted pendulum models. Journal of Robotics and Mechatronics **11**(4) (1999) 304–309
12. Iida, S., Kato, S., Kuwayama, K., Kunitachi, T., Kanoh, M., Itoh, H.: Humanoid robot control based on reinforcement learning. In: Proc. of the 2004 Int. Symposium on Micro-Nanomechatronics and Human Science and The Fourth Symposium Micro-Nanomechatronics for Information-Based Society. (2004) 353–358
13. Peters, J., Vijayakumar, S., Schaal, S.: Reinforcement learning for humanoid robotics. In: Third IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids 2003), Karlsruhe, Germany. (September 2003)
14. Yamasaki, F., Endo, K., Kitano, H., Asada, M.: Acquisition of humanoid walking motion using genetic algorithm - considering characteristics of servo modules. In: ICRA, IEEE (2002) 3123–3128
15. Wama, T., Higuchi, M., Sakamoto, H., Nakatsu, R.: Realization of tai-chi motion using a humanoid robot. In Jacquart, R., ed.: IFIP Congress Topical Sessions, Kluwer (2004) 59–64
16. Baltes, J., McCann, S., Anderson, J.: Humanoid robots: Abarenbou and daodan. In: RoboCup 2006 - Humanoid League Team Description, Bremen, Germany. (June 2006)
17. Dalla Libera, F., Minato, T., Fasel, I., Ishiguro, H., Menegatti, E., Pagello, E.: Teaching by touching: an intuitive method for development of humanoid robot motions. In: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids 2007), Pittsburg, USA. (December 2007)