Intuitive Humanoid Motion Generation Joining User-Defined Key-Frames and Automatic Learning

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

[1] Intelligent Autonomous Systems Laboratory, Department of Information Engineering (DEI), Faculty of Engineering, University of Padua, Via Gradenigo 6/a, I-35131 Padova, 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. In this paper we present a new method for generating humanoid robot movements. We propose to merge the intuitiveness of the widely used key-frame technique with the optimization provided by automatic learning algorithms. Key-frame approaches are straightforward but require the user to precisely define the position of each robot joint, a very time consuming task. Automatic learning strategies can search for a good combination of parameters resulting in an effective motion of the robot without requiring user effort. On the other hand their search usually cannot be easily driven by the operator and the results can hardly be modified manually. While the fitness function gives a quantitative evaluation of the motion (e.g. "How far the robot moved?"), it cannot provide a qualitative evaluation, for instance the similarity to the human movements. In the proposed technique the user, exploiting the key-frame approach, can intuitively bound the search by specifying relationships to be maintained between the joints and by giving a range of possible values for easily understandable parameters. The automatic learning algorithm then performs a local exploration of the parameter space inside the defined bounds. Thanks to the clear meaning of the parameters provided by the user, s/he can give qualitative evaluation of the generated motion (e.g. "This walking gait looks odd. Let's raise the knee more") and easily introduce new constraints to the motion. Experimental results proved the approach to be successful in terms of reduction of motion-development time, in terms of natural appearance of the motion, and in terms of stability of the walking.

1 Introduction

It is widely accepted that robots will become part in everyone's life in the near future and that their use will not be limited to manufacturing chains in factories. Emblems of this tendency are the humanoid robots, which structure and purpose

L. Iocchi et al. (Eds.): RoboCup 2008, LNAI 5399, pp. 13–24, 2009.

is very different from classic robotic arms. Small size humanoids have recently become more and more popular among robots, mainly for entertainment and research purpose. This is due to the strong decrease in their cost in the last few years. Small humanoids are usually less than 50 cm tall and actuated by low voltage servomotors (around 5V) at each of the joints. See [11], [12] or [13] for some examples. Though these robots are often simpler than big ones, the number of their degrees of freedom (d.o.f.) is very high. For instance Kondo's KHR-2 HV has 17 d.o.f. while VStone's Robovie-M has 22. Because of the high dimensionality of the configuration space, generating motions for this kind of robots is a complex task.

In general, three major strategies are adopted. The first one is about generating the motion off line, and then replaying it on the robot. The second one is about calculating the motion online, so that all the information available can be employed to produce an optimal movement. The third approach, that is placed in between the previous two, consists in calculating the motion off line and then adjusting it online depending on the data coming from the sensors, for instance to assure stability. While of course online motion planning and generation usually allows a better movement, nowadays off line motion generation (with or without online correction) is widely employed, mainly due to the restrictions posed by the limited computing power available on board of the robot. Several approaches were proposed to generate the motion, for instance by splines [1], by truncate Fourier series [2] or by central pattern generators. Nevertheless, the most basic but, surprisingly, the most widespread way of realizing motions is still to specify a motion as a set of "frames", that is a set of time instants for which the position of each joint is provided. The position to be assumed by each motor at each time is usually obtained by a simple linear interpolation of the positions it must assume in the previous and in the following frames. Figure 1 shows a commercial motion editor based on this design principle, while [3] or [4] provide examples of recent papers where keyframe-based interfaces are used. As easily observable in all these examples, this kind of interfaces usually presents one slider for each of the joints, which allows to choose the rotation angle assumed by the servomotor at the frame being edited.

Some improvements were proposed, for instance [5] introduced an interface where the motion is represented by Fourier coefficients calculated from control points provided by the user and [6] introduced a system by which the user develops movements by intuitively pushing the robot parts instead of acting on sliders.

However, these approaches force the user to specify an exact configuration of the joints, usually obtained through a trial and error process. On the other hand many articles, like [7], [8], and [10] present results achieved specifying the motion in a parametric way and letting a genetic algorithms determine good parameter configuration. As a drawback when CPGs are employed the effect of the parameters on the motion becomes often difficult to identify. This means that the user cannot easily modify a resulting motion by directly varying the parameters or impose constraints on the motion. This work aims at merging

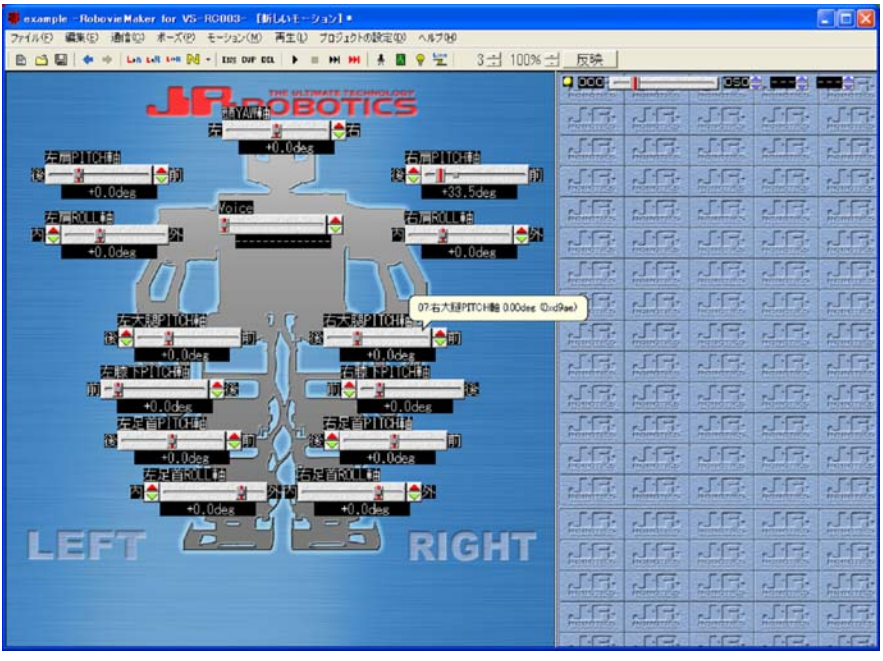

Fig. 1. Robovie Maker, a commercial motion development software developed by VS-tone

the classical, time consuming but easily understandable slider-based approach, with genetic algorithms that automatically search for a good motion. In the developed interface the user can specify a range of possible values instead of a fixed joint position and relationships that should be maintained between the joints. A genetic algorithm tries then to increase the value of a specified fitness function - that describes the quality of the motion - identifying good values that obey the constraints. If users notice bad features in the resulting motion, they can easily modify the constrains of the movement and run the genetic algorithm again. This paper is organized as follows. Section 2 introduces the advantages of the proposed approach and illustrates the ideas underlying the implementation. Section 3 describes how a walking pattern was obtained through the developed method and the performed tests. Section 4 presents the results of the experiment and is followed by a short section commenting these results. Finally,section 6 will summarize the presented concepts and discuss the future work.

2 Main Idea

A motion of a robot with n degrees of freedom can be defined by a function $f: \mathbb{R}^{\geq 0} \rightarrow M$, where $M \subseteq \mathbb{R}^n$ i.e. that given a real positive or null number, i.e. time, provides an n dimensional vector that specifies the position of each motor, that is some areas of M are not reachable, as the ones representing self collision

positions. Discretization of this space could also be considered to reduce the search space. Nevertheless this space remains huge considering that the values of n is around 20 for small humanoids available on the market. Let us consider the space F of functions describing the movements of the robot, i.e. functions f mapping time to motor positions. More formally, F is the space of functions having $\mathbb{R}$ as domain and M as codomain. In this case too, it is possible to reduce the search space considering domain and codomain discretization. We can also note that the variation between consecutive positions is limited by the motor speed. Nevertheless the dimension of F is, by definition, exponentially higher than M. Therefore search algorithms (in particular local search algorithms like hill climbing, simulated annealing or genetic algorithms) can explore just a very little part of this space. Thus, strong constraints must be introduced in the search space. With classical interfaces the user needs to completely specify the f function by a trial and error process. When search algorithms are employed, instead, the f function is nearly completely defined and it depends on few parameters which values are determined by the search algorithm. In other terms, the search is not done directly in F, but in a p dimensional space P, where p is the number of parameters and $|P| << |F|$. The mapping between the points in P (parameter sets) and the points in F (trajectories in M) can be very complex and highly non linear, as it happens, for instance, when CPGs are employed and P dimensions are the connection weights [8]. This fact has advantages as small variations in P might correspond to big variations in F so that very distant points in F can be sampled. The drawback deriving from this technique is that the meaning of the parameters is difficult to understand so in many cases nearly no predictions can be done regarding consequences of changing the value of one of them.

Conversely the approach adopted in this research is to work directly on M so that the meaning of the parameters is straightforward. In details, the f function is defined as a linear interpolation of key-frames (points in M), that is the movement is given by linear transitions between consecutive motor configurations m_i. To make things even more easily understandable, each position m_i is given by a sum of basic activations of some joints a_j, where the role of each activation a_j is very simple, like "raise the left arm" or "bring the right leg backwards". An activation is then simply a point in M having most of the coordinates equal to zero (the joints that do not need to be moved for the intended purpose) and the few coordinates used for each basic joint activation are defined in terms of constants, parameters or sum of parameters. Of course it would be possible not to introduce the concept of activations a_j, and define directly the frames m_i, but, as will be clear in the experiment section, the introduction of basic activations allows their reuse in several frames and makes the process of defining frames nearly trivial. Thanks to the vector nature of activations and frames, each m_i can also be expressed as a sum of $k_{i,j} * a_j$, where each $k_{i,j}$ is a scalar value. Figure 2 provides an example on what composing "activations" to create "frames" means.

For each parameter appearing in the activations the user specifies the range in which its value should be used. This allows the user to easily constrain the

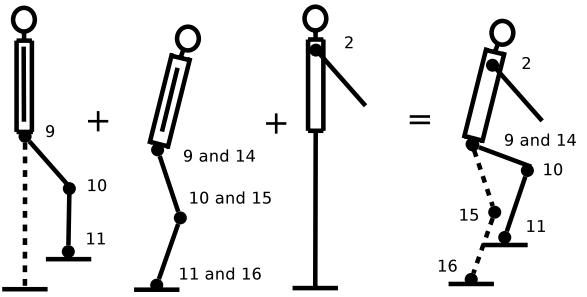

Fig. 2. Three "activations" (i.e. basic joint movements) are composed to create a "frame" (i.e. a posture at a certain time). Note that the distinctions between "activations" and "frames" is introduced just for clarity, since both of them are simple vectors describing the angle of each joint.

movement directly, which is often a big advantage. In fact when search algorithms are employed, if the parameter meaning is unknown the only way to constrain the final movement is to introduce penalizing factors in the score/fitness function. For instance, when developing a walking motion, an obvious fitness function would be the distance covered in a certain amount of time. The result of the algorithm could anyway be a motion where the robot moves fast by sliding its feet without lifting them and without falling. It would be then necessary to introduce a reward for lifting the feet in the fitness function, but deciding how to weight the covered distance and the feet lift is very difficult. With the approach taken in this paper, it is just necessary to choose a proper range for the parameters (actually, joint angles) in the activation a_j that has the role of lifting the foot. Certainly it is possible to provide many other examples regarding situations in which constraining the movement by using the fitness function is difficult. For instance, imagine to define a fitness function for a "human-like" gait. As it should be clear, instead, with the approach taken in this paper the user can employ an easy fitness function, run the genetic algorithm, observe the resulting motion and constrain the activations a_j (or introduce new ones) to correct the motion in the desired way. The genetic algorithm can then be run again and this process can be carried on until the motion determined by the genetic algorithm (and constrained by the user) is satisfactory. It is interesting to note the generality of this method, since no assumptions are made on the robot structure or on the desired movement. Therefore, even if the proposed approach had been applied to a very standard task, walking, we'd like to stress that the method we describe can be used for the realization of any kind of motion. We also feel that reporting a comparison of this method with other techniques specifically devised for the generation of walking motions is out of the scope of this paper. Finally it is worth to state that the proposed method aims at allowing the user to develop in an intuitive way and in short time a variety of motions, without the need to devise any mathematical description of the trajectories. Indeed, the maximization of the performance is not one of the purposes of this work.

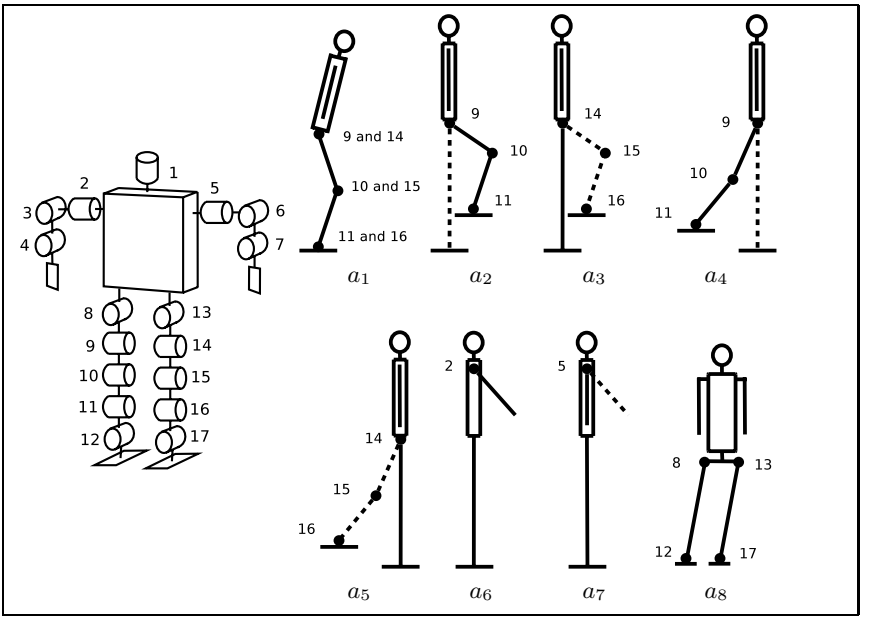

Fig. 3. Activations used to define the walking motion. The figure on the left shows the disposition of the Kondo's KHR-2 degrees of freedom, each one labeled with a number. The same numbers are used in the activation pictures to indicate the joints employed. Dashed lines represent the left part of the body, continuous ones the right side.

3 Realization of a Walking Motion

In our experiments a Kondo KHR-2HV was used. This is a small humanoid robot with 17 degrees of freedom, 34 cm tall and weighting about 1.3 Kg. For time reasons and to avoid hardware damages, preliminary tests were conducted using a simulator, namely USARSim. See [9] for a description of this simulator. A walking motion was developed for this robot using the proposed approach. In detail, first some "activations" a_i, that is basic set of joint modification used to create reusable features of postures were defined. Later, using these "activations", "frames" m_j,

Eight basic activations, also depicted in figure 3, were defined.
More precisely

- a_1 defines the basic posture of the robot during walking, consisting in slightly bending the knees (turning therefore the ankles too) and turning the hip joints so that the trunk is slightly bent forward. This activation defines three parameters, $p_{posture,thigh}$, $p_{posture,knee}$, $p_{posture,ankle}$ corresponding to the three joints mentioned (their values are equal for both legs).
- a_2 and a_3 define, respectively, the action of lifting up the right and the left foot. Two parameters, $p_{lift,thigh}$ and $p_{lift,knee}$ are defined and represent,

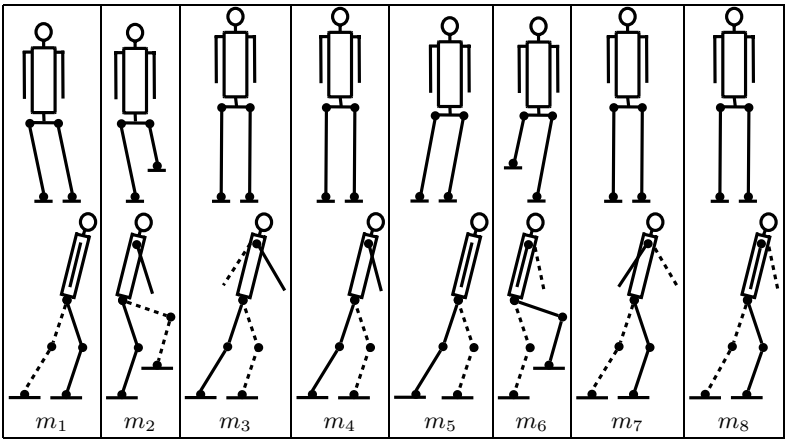

Fig. 4. Frames designed to have a walking motion

respectively, the thigh angle and the knee angle. The ankle is moved so that it maintains parallel to the ground (it is rotated by the opposite of the sum of the knee and thigh angles). For these two activation,as well as for the ones described hereafter the same parameter defines the position of the corresponding joints in the two legs.

- a_4 and a_5 are used to bring backwards, respectively, the right and the left leg, by turning the thigh and the knee. Also in this case, two parameters $p_{back,thigh}$ and $p_{back,knee}$ are defined for these joints while the ankle degree of freedom is used to keep the foot parallel to the ground.
- a_6 and a_7 provide the actions of rotating, respectively, the right and left shoulder joints to swing the arms. Each of these activations has just one parameter, $p_{armrotate,shoulder}$ which defines the shoulder angle.
- a_8 consists in rotating the thigh joints by an angle set by a parameter, $p_{swing,thigh}$, and the ankles by the opposite angle, so the legs keep parallel and the body moves in the frontal plane.

Given these activations, the definition of the frames was straightforward:

$$\begin{array}{ll} m_1 = a_1 + a_5 - a8 & m_2 = a_1 + a_3 + \frac{1}{2}a_6 - a_8 \\ m_3 = a_1 + a_4 + a_6 - \frac{1}{2}a_7 & m_4 = a_1 + a_4 + \frac{1}{2}a_6 \\ m_5 = a_1 + a_4 + a_8 & m_6 = a_1 + a_2 + \frac{1}{2}a_7 + a_8 \\ m_7 = a_1 + a_5 - \frac{1}{2}a_6 + a_7 & m_8 = a_1 + a_5 + \frac{1}{2}a_7 \end{array}$$

For simplicity, in the current implementation the time between any two consecutive frames (m_i and m_{i+1}) is the same and is a parameter determined by the genetic algorithm (in other terms, the speed is a parameter too).

3.1 Genetic Algorithm

As understandable from the previous subsection, the genetic algorithm must determine 10 parameters, 9 of which are appearing in the activations and the

remaining one is the time between frames. Each parameter was coded as a real value, and for each of them a search range was provided. We used a population of $S = 20$ individuals, let evolving for 50 generations. To evaluate each individual a quite simple fitness function was employed. Assume the robot is standing in the XY plane, headed toward the X direction at the beginning of the evaluation. The fitness function of individual P_i is given by $f(P_i) = \sqrt{\max\left\{0, \max\{0, x_T\}^2 - |y_T|\right\}} + 50 * s_T$, where x_T and y_T are the coordinates of the robot, expressed in cm, reached in 20 seconds and s_T is 0 if the robot felt down within these 20 seconds and 1 otherwise, i.e. it is a bonus for individuals that do not fall down. The term $\sqrt{\max\left\{0, \max\{0, x_T\}^2 - |y_T|\right\}}$ is used to give higher scores to individuals that walk further, giving less credits to the individual that do not proceed straight (on average, in 20 seconds) or proceed backwards. A simpler function like $x_T - |y_T|$ could have been used. Anyway in this case the evolution prefers individuals that just stand at the initial position over individuals that moved far but have $y_T > x_T$. The employed function instead gives an higher fitness to walking patterns that make the robot move forward, unless $y_T > x_T^2$, which could be considered as a side walk. For evolution between successive generations a ranking selection was chosen; in details, the Z individuals of the population are ordered by decreasing fitness. Let us denote by P_i the individual of the population appearing as the i-th in the ranking. The individuals $P_1 \dots P_k$ with fitness equal to at least half the the fitness of P_1 are then identified (i.e. k, $1 \leq k \leq Z$, is determined) and only these are considered to produce the following generation. Tests were conducted applying both crossover and then mutation. More precisely, when taking individuals at random the probability of taking P_i is given by

$$\wp(P_i) = \frac{k - i + 1}{\sum_{j=1}^{k} j} = \frac{k - i + 1}{k * (k+1)/2}$$

Firstly two individuals P_a and P_b were picked up at random, and the resulting individual P_c was obtained taking each gene from P_a with probability $\frac{f(P_a)}{f(P_a)+f(P_b)}$ and from P_b with probability $\frac{f(P_b)}{f(P_a)+f(P_b)}$. Mutation was then applied on P_c, and particularly each gene was modified (with probability $\wp_{mutation} = 1$) multiplying its value by a random number in the $[0.75, 1.25)$ range.

4 Experimental Results

The approach presented in this paper, with the activations and frames reported in the previous section, successfully led to a walking pattern on simulation. During the development of the motion we modified the motion constrains, and in detail the parameter ranges, three times. Initially $p_{lift,thigh}$ and $p_{lift,knee}$ were given a range of $[0, \pi/2]$ but the algorithm always chose very low values for these 2 parameters and the robot just slided the foot, giving a stable but

Table 1. Values of the parameters

Parameter	Range	Best,simulation	Best,real robot
$p_{posture,thigh}$	$\left[\frac{\pi}{2} - \frac{\pi}{16}, \frac{\pi}{2}\right]$	1.388	1.4801
$p_{posture,knee}$	$\left[\frac{\pi}{2} - \frac{\pi}{16}, \frac{\pi}{2}\right]$	1.428	1.4403
$p_{posture,ankle}$	$\left[\frac{\pi}{6} - \frac{\pi}{16}, \frac{\pi}{2} + \frac{\pi}{16}\right]$	0.3316	0.5195
$p_{lift,thigh}$	$\left[\frac{\pi}{8}, \frac{\pi}{3}\right]$	0.3927	0.3826
$p_{lift,knee}$	$\left[\frac{\pi}{8}, \frac{\pi}{4}\right]$	0.7854	0.2826
$p_{back,thigh}$	$\left[\frac{\pi}{8}, \frac{\pi}{4}\right]$	0.7854	0.2489
$p_{back,knee}$	$\left[\frac{\pi}{8}, \frac{\pi}{6}\right]$	0.1992	0.0981
$p_{armrotate,shoulder}$	$\left[\frac{\pi}{16}, \frac{\pi}{4}\right]$	0.6882	0.4961
$p_{swing,thigh}$	$\left[\frac{\pi}{32}, \frac{\pi}{8}\right]$	0.1470	0.2240
$time$	$[0.1, 0.4]$	117.2	140.3

not human like walking pattern. The lower value was then increased to $\pi/8$. Surprisingly after this modification the algorithm tended to select values close to the maximum of the range. The evolution tended also to prefer very fast and small steps over slower and longer ones. To account for this we increased the minimum time between frames from 10ms to 100ms. The maximum value of the lateral swing ($p_{swing,thing}$parameter), instead, was decreased from $\pi/4$ to $\pi/8$. These modifications of the range show the advantage of the proposed method, i.e. the clear meaning of the parameters allows to constrain and improve the motion very easily. Table 1 reports the ranges of each parameter in the initial population, their values on the best individual obtained in simulation during 50 generations and the values for the best individual in the real world determined with the last 10 steps of evolution on the real robot. Angles are expressed in radians, times in seconds. Note that for all the individuals of the first generation each parameter is set to the medium value of its range. The fitness of the best individual found using the simulator is 181.38, corresponding to an individual proceeding for 131.38 cm in the X direction and 4.03 in the Y direction. On the real robot the distances reduced to 48.9 cm and 4.3 cm in the X and Y direction respectively. This is due to the increase of the inter-frame time and to the reduction of friction between the feet and the ground in the real world compared to the simulated case.

5 Discussion

Observing section 3 it is possible to understand how easy it is to define motions in terms of "activations" and "frames". Employing the method for the development of a walking pattern, a very important movement for the Robocup competition, we could verify that the approach is, as expected, less time consuming (at least for operator time) than directly setting each joint. In fact, while designing a walking pattern by a slider based interface took about eight hours, the design of frames and the adjustments on the range took less than three hours to the same user. The achieved speed is not very high, anyway the walking is very stable, in

Fig. 5. Snapshots of the walking pattern achieved on the real robot. The time between successiva images is 150 ms.

fact the robot walked for over seven minutes without falling. We strongly believe that inserting new frames to refine the motion improvements of the motion could be easily achieved. While these tests are of course too limited to completely prove the efficacy of the presented method, these preliminary results are encouraging and suggest this could be a good direction for further research.

6 Conclusions and Future Work

This work presented a simple approach to develop motions for small sized humanoid robots. Our method is structured as follows: the user provides a set of "activations" that indicate the joints required to execute a basic action. The joint positions can be specified in terms of relationships between the joints and possible ranges. By using weighted sums of these activations the human operator is able to build "frames", i.e. key positions to be assumed by the robot during the movement. Each parameters is bounded within a range also specified by the user. Once the genetic algorithm has detected a good set of parameters (that is, parameters that provide a high value for the fitness function) the user can verify if the motion is satisfactory. If this is not the case, instead of changing the fitness function (an often complex process) s/he can directly edit the ranges to modify the constraints on the motion in a straightforward way. This actually happened three times during a walking pattern used as a test-bed of the approach. Our approach is fast to implement, since it requires nothing more than

writing a standard genetic algorithm (or any search algorithm, like policy gradient) and linear interpolation. The main drawback of this method is the time required by the genetic algorithm to calculate a good set of parameters. In the current implementation a single virtual robot is moved in the simulated world in real time. In order to have good evaluations of the fitness each individual is tested three times. The evaluation of each individual takes therefore one minute. The code has a client server architecture that allows to run multiple evaluations simultaneously. In the experimental setup two PCs were used. The time required to evolve for 50 generations was then 8 hours and 20 minutes, since it is possible to run just one instance of the simulator on one PC. Future work will therefore have the purpose of adapting the current system to a non-realtime, lightweight simulator to decrease the computation time.

Acknowledgments

We'd like to acknowledge Luca Iocchi for the fruitful discussions and Alejandro Gatto for his help in the realization of the experiments.

References

1. Ude, A., Atkeson, C., Riley, M.: Planning of joint trajectories for humanoid robots using B-spline wavelets. In: IEEE Conference on Robotics and Automation, San Francisco (2000)
2. Yang, L., Chew, C.M., Poo, A.N.: Adjustable Bipedal Gait Generation using Genetic Algorithm Optimized Fourier Series Formulation. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing (2006)
3. Wama, T., Higuchi, M., Sakamoto, H., Nakatsu, R.: Realization of tai-chi motion using a humanoid robot. In: Rauterberg, M. (ed.) ICEC 2004. LNCS, vol. 3166, pp. 14–19. Springer, Heidelberg (2004)
4. Baltes, J., McCann, S., Anderson, J.: Humanoid Robots: Abarenbou and DaoDan. In: RoboCup - Humanoid League Team Description (2006)
5. Mayer, N.M., Boedecker, J., Masui, K., Ogino, M., Asada, M.: HMDP:A new protocol for motion pattern generation towards behavior abstraction. In: RoboCup Symposium, Atlanta (2007)
6. 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 International Conference on Humanoid Robots (Humanoids 2007), Pittsburg, USA (December 2007)
7. Yamasaki, F., Endo, K., Kitano, H., Asada, M.: Acquisition of Humanoid Walking Motion Using Genetic Algorithm - Considering Characteristics of Servo Modules. In: Proceedings of the 2002 IEEE InternationalConference on Robotics & Automation, Washington, DC (2002)
8. Inada, H., Ishii, K.: Behavior Generation of Bipedal Robot Using Central Pattern Generator(CPG) (1st Report: CPG Parameters Searching Method by Genetic Algorithm). In: Proceedings of the 2003 IEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, Nevada (2003)

9. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: a robot simulator for research and education. In: Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007) (2007)
10. Hebbel, M., Kosse, R., Nistico, W.: Modeling and Learning Walking Gaits of Biped Robots. In: Proceedings of the Workshop on Humanoid Soccer Robots of the IEEE-RAS International Conference on Humanoid Robots, pp. 40–48 (2006)
11. VStone's official homepage, `http://wwsw.vstone.co.jp/`
12. Kondo Kagaku's official homepage, `http://www.kondo-robot.com/`
13. Robonova's official homepage, `http://www.hitecrobotics.com/`