Reliable and Precise Gait Modeling for a Quadruped Robot

Uwe Düffert and Jan Hoffmann

Institut für Informatik
LFG Künstliche Intelligenz
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany
http://www.aiboteamhumboldt.com

Abstract. We present a parametric walk model for a four-legged robot. The walk model is improved using a genetic algorithm, but unlike previous approaches, the fitness is determined in a run that closely resembles the later application. We thus not only achieve high speeds, but also a high degree of flexibility. In addition to the walking model being flexible, we present a means of automatically calibrating the walking engine. This allows for highly precise robot control and greatly improved odometry accuracy. Lastly, we show how the motion model can be extended to integrate specialized motions to further increase locomotion speed without compromising flexibility.

1 Introduction

Legged robots operate in areas that wheeled robots have trouble accessing. On the other hand, creation and optimization of quadruped robot locomotion is a challenging, highly complex task. The main reason for this is the difficulty to cope with the many degrees of freedom of a legged robot even in a relatively simple robot design. Such designs (may it be software and/or hardware) are often inspired by animal locomotion in terms of anatomy (number of limbs and joints, proportions, etc.), gait patterns (Central Pattern Generators) and concepts such as reflexes [5, 1, 13, 16].

Inverse kinematics is commonly used to calculate the motor commands necessary for robot motions: trajectories of the robot's paws are defined and then the corresponding limb movements and thus joint angles as a functions of time are calculated. Inverse kinematics is usually based purely on geometry, neglecting physical properties such as friction, weight of the robot as a whole and of individual components, moments of inertia, motor strengths, etc. These simplifications in the modeling tend to impair performance in real world environments.

Experience can help to come up with experimental setups which take these pitfalls into consideration while not explicitly modeling them, e. g. by conducting experiments on especially difficult surfaces [11]. Limiting possible motions to statically stable ones, robust robot gaits can be developed. These motions

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 49–58, 2006.

are reversible at any given point in time and require three of the robot's four legs to touch the ground at any time [6]. These motions trade off speed for robustness whereas dynamically stable motions can produce faster locomotion [15, 4, 8]. Gaits can be found by trying to model all physical aspects of the robot, but bridging the gap between simulation and the real world remains challenging [7, 3]. An alternative gait representation in frequency space is described in [10] yielding smooth motions and gait transitions.

Evolutionary approaches include the evolution of the controller alone (with fixed robot morphology [11, 17]) and simultaneous evolution of robot morphology and controller [19, 18]. Other approaches include teaching, learning, and inverse kinematics [12]. Many machine learning approaches focus on optimizing a single criterion such as forward or turning speed which results in highly specialized motions. From these specialized motions it tends to be difficult to generalize towards gait patterns that can handle 'mixed' motions, e. g. moving forward and sideways at the same time. These mixed motions are of great importance in real robot control in dynamic environments where it is desirable to be able to adjust the robot's position and orientation quickly ('omnidirectional' movements).

Outline. This paper describes a combination of different strategies to improve the overall performance of four-legged walking. We present a parametric walk model extending [9]. Using this 'wheel model' increases the robustness of walk of the legged robot. A means of automatically optimizing the gait pattern is presented, focusing on flexibility rather then speed alone. We then show how the gait pattern can be calibrated to achieve reproducible performance and odometry data of high quality. Lastly, we show how the walking engine can be extended to integrate specialized motions without sacrificing flexibility.

2 Method

2.1 The Wheel Model

Legged robots can access areas which wheeled robot are unable to cope with. Unfortunately, robot control for a legged robot is extremely complex due to the many degrees of freedom involved. For the four-legged Sony Aibo ERS-210, [9] introduced the *wheel model* which assumes that the robots paws are performing circular motions and robot control works much like that of a differential drive robot (fig. 1). Using this model, dynamic stability of the robot can be achieved easily by constraining the phase shift between the movement of individual legs. In our experiments, we used both Aibo ERS-210s and ERS-7s.

Each movement (consisting of a forward, a sideways, and a turning speed) results in a circular movement around a common center of rotation (see fig. 1 a). This movement is realized by making steps tangential to the circle with the speed a wheel would have at the same position.

Actual gait patterns are described by a set of parameters of the walking engine. Several approaches to parameterize these patterns where developed in the context of gait optimization allowing the trajectories to differ from the first used trapezoidal shape (e. g. *canter action* in [9] and *free form quad* in [2]).

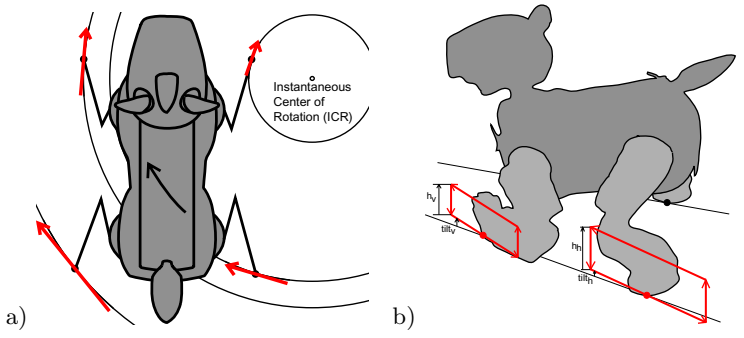


Fig. 1. a) Simultaneous walking and turning using the wheel model: The resting positions of the feet move on circles around a common rotation center. This is realized by steps (red) that are deduced from imagined wheel movements and tangential to the optimal circles. **b)** Parameters of the foot movements: Feet are moved in parallelograms (rhomboids), the direction and size of which is determined by the wheel model.

We chose the parameter set shown in table 1 which allowed us to fully describe the characteristics of different gait patterns while still being reasonably small for genetic optimization.

Table 1. All used parameters of a parameter set and their meaning

- x_f, y_f, z_f: position of a front foot relative to middle between the front shoulders
- x_h, y_h, z_h: position of a hind foot relative to middle between the hind shoulders
- h_f, h_h: maximum height of the feet above ground during a step
- $tilt_f, tilt_h$: tangent of the the arc between the theoretical foot trajectory and the ground; influences the intensity of touching the ground and can avoid sliding on it
- gp_v, gp_h: fraction of time a front or hind foot has ground contact (in theory)
- $l = (l_{fl}, l_{fr}, l_{hl}, l_{hr})$: relative time of lifting each leg, (0, 0.5, 0.5, 0) describes the usually used trot and means lifting left front and right hind foot at the beginning of a full step (0) and lifting the other two feet half a full step later (0.5)
- T: duration of a full step in frames à 8 milliseconds.

2.2 Localization

One of the goals of our work was to allow fully automatic gait optimization. We wanted to be independent of external hardware (such as an external camera and computer to track the robot) and wanted everything to run on the robot. This requires the robot to be well localized in order for it to determine its current speed and performance. The standard beacons on a RoboCup field were used in previous work, but they have two disadvantages: 1) if only one beacon is visible, the robot has no way of determining its lateral position, and 2) tracking multiple beacons is error prone and the localization error is often bigger than the covered distance. We therefore devised a special bar code like pattern (see fig. 2) that allows the robot to precisely monitor its x-y-position and orientation from a single camera image from a wide range of positions. The black and white

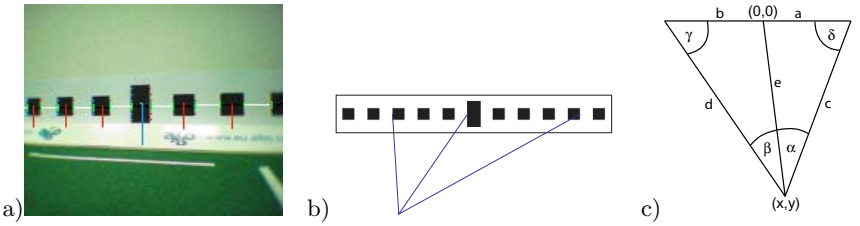


Fig. 2. The b/w pattern used for localization: **a)** A camera image of an ERS-210 with detected pattern parts highlighted, **b)** Schematic view of the pattern, **c)** Variables used to calculate the robot pose

pattern has the further advantage of being clearly detectable in various lighting conditions. Size and structure of the pattern are chosen such that equally good localization performance can be achieved from the majority of distances to it.

As shown in fig. 2 c, the current position can be calculated from the known distances a and b and the recognized angles α and β.

The angle δ can be calculated by applying the sine theorem twice. This yields the relative position (x, y) w.r.t. the center of the pattern:

$$x = -c\sin(\delta)$$
$$y = -a + c\cos(\delta)$$

with $c = (a\sin(\pi - \alpha - \delta))/\sin(\alpha))$. For a distances from the pattern of 60 cm to 260 cm, the position error from single images during walking is $\Delta p = (16\,\text{mm}, 38\,mm)$. The noise caused by camera vibrations can be reduced by using simple PID smoothing. This decreases the average position error and allows meaningful speed calculations even for short durations.

2.3 Evolution Run

Most previous optimization approaches have in common that they are focused on improving a particular motion, such as walking forward at high speed. They fail, however, to take into account the overall performance of the robot in real world situations, where constant adjustments of the robot's direction and orientation are necessary.

To evaluate a parameter set, the robot's performance following a path is determined. Unlike other experiments, this is not a simple straight line, but a rather complex path that requires the robot to strafe and turn too to follow it (see fig. 3). Deviations from the path are penalized by the fitness function (see below). Such a course is a sequence of target positions and robot orientations. An evaluation run consists of two parts, a forward part with desired robot orientation changing over time as shown in fig. 3, and a backward part with constant robot orientation. Instead of executing a fixed sequence of steps,

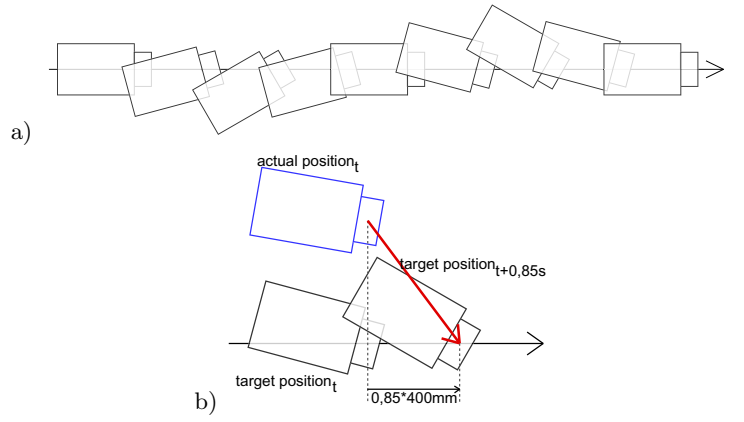


Fig. 3. a) The path for evaluation omnidirectional gait pattern defines the target position and orientation of a robot. **b)** The activated movement (red) results from the distance to the target position 0.85 seconds later.

the robot is forced to correct its previous mistakes to stay on the course. Such a task was chosen because it is easily reproducible and quite typical for actual applications where the robot constantly changes the direction while walking forward (e. g. chasing a ball). The control algorithm is very similar to the one used by us in RoboCup games to steer the robot towards the ball or some other target position.

The runs are performed fully autonomous. No manual replacement of the robot is necessary since it is generally able to localize using the pattern described above. The start and end point of the path are fixed position relative to the localization pattern.

The obvious performance criteria for a gait are speed and accuracy. The fitness function F of a parameter set P used in the experiments favors more stable gaits and can be interpreted as walk speed corrected by unwanted vibrations and position deviations:

$$F(P) = \dot{x} - \Delta y/6 - 33\Delta\varphi - \left(10^{-5}\ddot{z} - 5\right) - 40p_{\text{blind}}$$

where $\dot{x}$ is the average speed, Δy and $\Delta\varphi$ the deviation from the course, $\ddot{z}$ describes vertical vibrations and p_{blind} is the percentage of images without recognition of the localization pattern, e. g. because of vibrations or totally wrong gaze direction. As walking forward is more important, the fitness of the forward walking part contributes more than that of walking backward.

Evaluating a single parameter set takes about 30 seconds: 10s for walking forward with turning, 10s for walking backward and 10s for positioning for the next run. If the performance of a parameter set is below a threshold, the run is aborted and the robot returns to the starting point using the standard gait. Typically, a complete optimization run with 50 to 60 parameter sets takes about 30 minutes.

2.4 Evolution

Since it is very difficult to model the interdependencies between parameters and the resulting speed and quality of a gait pattern, genetic algorithms were used for optimization (see [14]). A population of parameter sets is exposed to evolution. Each parameter set corresponds to an individual and each parameter to a gene. All genes of an individual are stored on a single chromosome.

Using real robots for the evolution is time consuming, therefore choosing the following evolution parameters turned out to be a good compromise between fast advance and avoidance of unusable parameter sets.

A population consists of only ten individuals, starting with a known parameter set (a manually tweaked one used by our team in previous years) and nine mutations of it. In every generation, half of the population with the worst fitness is selected and replaced by mutations and recombination of the better half. 40% of the descendants are created by mutation and 60% by recombination. Mutation changes single genes with a probability of 30%, equally distributed up to ±6% of its original value. Recombination interpolates the value of each gene randomly between the parent values of that gene or even extrapolates into the direction of the better parent.

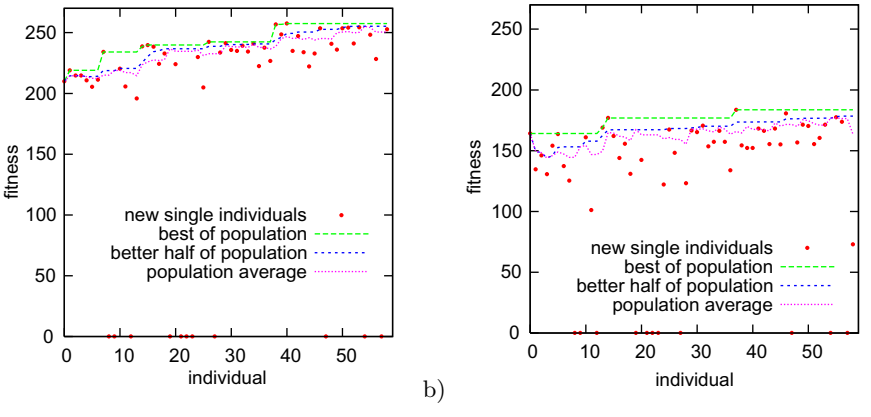


Fig. 4. ERS-7: Evolution of walking parameter sets with two separated populations for the forward and the backward part of the course, **a)** Development of the fitness in the population of forward walking parameters, **b)** Development of the fitness in the population of backward walking parameters

Using these values results in visible and measurable differences between single individuals without getting unusable parameters. This method is useful for local optimization in a sensible part of the search space without prior knowledge about correlation between parameters and with only a few abortive attempts (fig. 4).

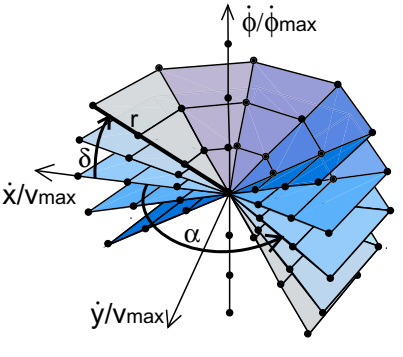
2.5 Odometry Calibration

Experience shows that the executed motion does not always match the one intended by the calculated step sizes. The actual speed depends non-linearly on the target speed.

A walk request commonly consists of forward, sideways, and turn speed. It turned out, however, that for our application this is not a very good way of describing motions. We therefore devised a different means of describing a walk request consisting of the walk direction α, the 'turn-walk-ratio' δ, and the 'overall speed' r as defined in fig. 5. To approximate the non-linear dependency of target and actual speed, the target speed for each combination of walk direction and turn-walk-ratio is divided up into three ranges [0, small[, [small, med[, and [med, max]. Within each of these ranges, the dependency is assumed linear. The values for the boundaries are determined in the calibration process. The calibration is done for all combinations of forward, sideways, and turn speed which results in 127 boundary points to be calibrated. Luckily, this can be done automatically:

In a first run, an autonomous behavior measures the influence of increasing the step size on the walk speed for each combination of walk direction and turn-walk-ratio for a certain parameter set. If increasing the step sizes does not increase the overall speed any more, the behavior starts to measure the next walk direction. This first calibration run determines the respective minimum and maximum speeds and enables to chose a medium speed minimizing the deviations when using linear interpolation in between.

In a second calibration pass, all chosen 127 boundaries are adjusted independently to match their requested forward, sideways, and turn speed by changing the target step size proportional to half of the detected speed difference. These adjustments minimize the gap between target and actual motion without risking to alternate only the sign of the difference. Iteratively running the calibration further decreases the error, as shown in fig. 5 b.



walk direction $\alpha = \arctan(\dot{x}, \dot{y})$
$\rightarrow \left[-\pi, -\frac{3\pi}{4}, -\frac{\pi}{2}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right]$

turn-walk-ratio $\delta = \frac{2}{\pi} \arctan\left(\frac{v}{v_{max}}, \frac{\dot{\varphi}}{\dot{\varphi}_{max}}\right)$
$\rightarrow \left[\underbrace{-1}_{\text{turn right}}, -\frac{3}{10}, -\frac{1}{10}, 0, \frac{1}{10}, \frac{3}{10}, \underbrace{1}_{\text{turn left}}\right]$

overall speed $r = \sqrt{\left(\frac{v}{v_{max}}\right)^2 + \left(\frac{\dot{\varphi}}{\dot{\varphi}_{max}}\right)^2}$
$\rightarrow$ [slow, medium, fast] .

Fig. 5. The position of the 127 parameter sets used (black dots: standing, 6× turning only, 8×5×3× with walking): The azimuth α denotes the walk direction, the declination δ denotes the normalized turn-walk-ratio and the radius r the normalized overall speed

Calibration is done for constant walk requests only. A constant walking motion always results in an arc. Knowing time, position and orientation of the robot at a starting and an ending point, the average walking and turning speed inbetween can be calculated.

This calibration process is repeated when the robot has to operate on a new surface.

2.6 Using Multiple Parameter Sets

Once a good omnidirectional gait pattern was found, we explored ways of including specialized gaits to further increase performance. This brings about the problem of switching or interpolating between gait patterns. That problem can be solved by allowing different parameter sets for the 127 boundary points described in the previous section. For interpolation to work, neighboring parameter sets must not differ *too much*.

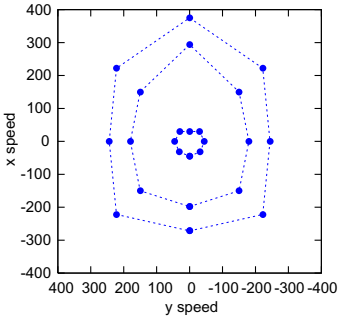
If e. g. a particularly good (and similar enough) parameter set was found for turning, it can be used to replace the parameter set(s) associated with the boundary points (walk direction $= 0$, turn-walk-ratio $= \pm 1$, overall speed $= v_{\max}$).

Whether a parameter set is suitable for interpolation is evaluated manually. For example, we extended the walking engine by including a parameter set for fast turning. This was derived manually from the parameter set for the omnidirectional walk by decreasing the distance between front and hind feet.

2.7 Performance

The walking engine consisting of the omnidirectional parameter set and specialized motions for 'walking backwards in a straight line' and 'fast turning (only)'

a)



b)

	$\lvert\Delta\dot{x}\rvert$ in mm/s	$\lvert\Delta\dot{y}\rvert$ in mm/s	$\lvert\Delta\dot{\varphi}\rvert$ in rad/s
uncalibrated	12,9	12,1	0,086
measured	7,7	11,4	0,070
calibration 1	6,4	7,6	0,038
calibration 2	5,8	8,0	0,030
calibration 3	4,4	8,7	0,021
2 weeks later	7,9	7,5	0,046
different robot	6,9	7,8	0,039

Fig. 6. a) The used minimum, medium, and maximum speed for an ERS-7 in all 8 walk directions without turning: By using several optimized and calibrated parameter sets, a much higher speed range can be covered than by using a single parameter set and the (necessary) speed limitation e. g. to an ellipse. **b)** The average difference between assumed and real speed of the 127 parameter sets can be decreased significantly with a few calibration steps.

proved to deliver highly reliable and reproducible performance. The speed ranks amongst the highest that have been achieved on the Aibo ERS-7 to date (fig. 6 a). The precision was evaluated qualitatively: the robot was able to stay on a rectangular path only using odometry for localization. In contrast, using uncalibrated motions, the robot would turn farther than desired at every corner resulting in a triangular trajectory.

3 Conclusion

Using the presented experimental setup, we were able to perform automated evolutionary optimization of gait patterns on a legged robot. Unlike most other approaches, the gait pattern found performs well in actual applications where target speed and direction of the robot continuously change. This was achieved by having the robot follow a path that closely resembles a real life situation for evaluation. The gaits found are calibrated to allow for remarkably accurate odometry which greatly improves localization. Lastly, the walking engine was extended to allow interpolation from one parameter set to another. Using this approach, we were able to use highly optimized/specialized motions in combination with the general, highly accurate gait pattern found in the evolution. The walking engine was successfully used in the RoboCup 2004 world championships Sony Four-Legged League. Its outstanding performance and precision was one of the key advantages of the GermanTeam over other teams and helped to win the championship.

Acknowledgments

The program code used was integrated into the development of the GermanTeam, a joint effort of the Humboldt University of Berlin, University of Bremen, University of Dortmund, and the Technical University of Darmstadt. Source code is available for download at `http://www.germanteam.org`.

References

1. A. Billard and A. J. Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. 2000.
2. J. Chen, E. Chung, R. Edwards, N. Wong, E. Mak, R. Sheh, M. S. Kim, A. Tang, N. Sutanto, B. Hengst, C. Sammut, and W. Uther. runswift 2003. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004.
3. X. Chen, K. Watanabe, K. Kiguchi, and K. Izumi. Optimal force distribution for the legs of a quadruped robot. *Machine Intelligence and Robotic Control*, 1(2):87–93, 1999.
4. U. Düffert, M. Jüngel, T. Laue, M. Lötzsch, M. Risler, and T. Röfer. GermanTeam 2002. In *RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.)*, number 2752 in Lecture Notes in Artificial Intelligence. Springer, 2003. More detailed in http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf.

5. J. Duysens, H. V. de Crommert, B. Smits-Engelsman, and F. V. der Helm. A walking robot called human: lessons to be learned from neural control of locomotion. *Journal of Biomechanics*, 2000.
6. M. Fujita, S. Zrehen, and H. Kitano. A quadruped robot for RoboCup legged robot challenge. In *Proceedings of the second RoboCup Workshop*. Springer, 1998.
7. M. Hardt and O. von Stryk. The role of motion dynamics in the design, control and stability of bipedal and quadrupedal robots.
8. B. Hengst, D. Ibbotson, S. B. Pham, J. Dalgliesh, M. Lawther, P. Preston, and C. Sammut. The UNSW RoboCup 2000 Sony Legged League team. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 70–72 (64–75). Springer, 2001.
9. B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut. Omnidirectional locomotion for quadruped robots. In *RoboCup 2001 Robot Soccer World Cup V, A. Birk, S. Coradeschi, S. Tadokoro (Eds.)*, number 2377 in Lecture Notes in Computer Science, pages 368–373. Springer, 2002.
10. J. Hoffmann and U. Düffert. Frequency Space Representation and Transitions of Quadruped Robot Gaits. In *Proceedings of the 27th conference on Australasian computer science*, volume 26, pages 275 – 278. Australian Computer Science Society, Inc., 2004.
11. G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with Aibo. *IEEE International Conference on Robotics and Automation*, pages 3040–3045, 2000.
12. V. Hugel and P. Blazevic. Towards efficient implementation of quadruped gaits with duty factor of 0.75. In *Proceedings of the IEEE International Conference On Robotics and Automation*, 1999.
13. H. Kimura, Y. Fukuoka, Y. Hada, and K. Takase. 3d adaptive dynamic walking of a quadruped robot by using neural system model. 2001.
14. J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.
15. S. Lenser, J. Bruce, and M. Veloso. CMPack: A Complete Software System for Autonomous Legged Soccer Robots. 2001.
16. M. A. Lewis. Gait adaptation in a quadruped robot. *Autonomous Robots*, 12(3):301–312, 2002.
17. S. Nolfi and D. Floreano. Learing and evolution. *Autonomous Robots*, 7(1):89–113, 1998.
18. S. Nolfi and D. Floreano. *Evolutionary Robotics*. MIT Press, 2000.
19. K. Sims. Evolving 3D morphology and behavior by competition. In *Proceedings in Artificial Life IV, R. Brooks and P. Maes (editors)*, pages 28–39. MIT Press, 1994.