An Evolutionary Approach To Gait Learning For Four-Legged Robots

Sonia Chernova, Manuela Veloso
Computer Science Department
Carnegie Mellon University
{soniac, veloso}@cs.cmu.edu

***Abstract*—**

Developing fast gaits for legged robots is a difficult task that requires optimizing parameters in a highly irregular, multidimensional space. In the past, walk optimization for quadruped robots, namely the Sony AIBO robot, was done by hand-tuning the parameterized gaits. In addition to requiring a lot of time and human expertise, this process produced sub-optimal results. Several recent projects have focused on using machine learning to automate the parameter search. Algorithms utilizing Powell's minimization method and policy gradient reinforcement learning have shown significant improvement over previous walk optimization results. In this paper we present a new algorithm for walk optimization based on an evolutionary approach. Unlike previous methods, our algorithm does not attempt to approximate the gradient of the multidimensional space. This makes it more robust to noise in parameter evaluations and avoids prematurely converging to local optima, a problem encountered by both of the algorithms mentioned above. Our evolutionary algorithm matches the best previous learning method, achieving several different walks of high quality. Furthermore, the best learned walks represent an impressive 20% improvement over our own best hand-tuned walks.

I. Introduction

Creating effective motion in four-legged robots is a very challenging task, as there is a large number of degrees of freedom and therefore parameters to be set. We have extensively addressed this task for several years and have developed motions for a few different versions of the Sony AIBO four-legged robots. The process of defining successful motions is tedious and rather brittle, as it strongly depends on the particular walking environment. Every new walking surface, e.g., the specific carpet or the floor, changes the performance of a walk, potentially affecting the robot's walking speed and stability. The walk parameters then need to be recalibrated, a process that can easily take several hundred trials for an expert. Nevertheless researchers using the Sony AIBO robots, in particular for robot soccer, have developed effective motions. Our own best hand tuned walk is at 235mm/sec.

With multiple changes of hardware for new robots and an increasing number of different environments where the robots need to walk, there was recently a drive to develop automated approaches for selecting motion parameters. Two recent papers, namely from the UNSW and the UT Austin robot soccer teams, [1], [2], contributed the first learning approaches for four-legged robot motion. The UNSW learned walk was impressively acquired while at RoboCup-2003 [3] and the robots reached a speed of 270mm/sec. The UT Austin effort was later successful in reaching a learned walk of 291mm/sec. Both groups used locus-based motion systems and gradient descent based learning methods. A summary of the best hand tuned and learned walks can be seen in Table I.

Following up on these recent learning accomplishments, we engaged in researching for a new learning algorithm robust to evaluation noise. Both of the gradient-based learning methods mentioned above were sensitive to noise in parameter evaluations and were prone to prematurely converge to local optima. Additionally, the motion system developed by our team is not locus-based, and the search space is less continuous with a large number of local optima.

In this paper, we contribute an evolutionary approach that was used to successfully learn effective motion parameters that have allowed the robot to sustain maximum speeds of 296mm/sec. Since we have observed that the last digit is of great sensitivity to the time computation, we feel confident presenting our results as a maximum of 290 ± 6mm/sec, closely matching the performance of the UT Austin work. Our algorithm encountered a set of high quality and fast walks, improving the speed of our robot's walk by approximately 20% over our own best hand-written walk. Most importantly we have developed an algorithm that allows us to autonomously optimize our motion parameters for various surface conditions and different robot platforms, such as the ERS-7, in a relatively short amount of time. Furthermore, the approach presented in this paper is of value not only for ourselves but also for many other teams that use our motion system which has been freely available for several years.

The paper is organized as follows. We first present in

Hand-tuned Gaits			Learned Gaits		
UNSW	UTAustin	CMPack	UNSW	UTAustin	CMPack
254	245	235	270	291	296

TABLE I

FORWARD VELOCITIES OF THE BEST HAND TUNED AND LEARNED GAITS IN MM/SEC.

detail our robot motion system, highlighting its ability to test for kinematic errors. We then present the genetic learning algorithm, including the representation, operators, and fitness used. We present in detail the experimental learning setup, providing empirical results. We finally conclude the paper.

II. Robot Motion

A. The Motion System

At the lowest level, the goal of the motion system is to define a path for the legs in three-dimensional space that will provide fast and stable locomotion. To achieve this goal the task is abstracted to a higher level, and the motion is defined in terms of parameters describing the behavior of the body and legs.

Various approaches to robot motion have resulted in different parameterization methods. One popular approach is to define the path of each leg using loci of various shapes ranging from rectangles to ellipses. Variations of this method were used by the UNSW and UTAustin teams for their gait optimization algorithms [4], [5].

Our motion system focuses on the problem from a different perspective by approaching it from the point of view of the body instead of the legs [6]. The trajectory of the body is represented by an acceleration model which maintains the desired direction of motion. The behavior of each leg depends on whether the foot is in the air or on the ground. While the foot is on the ground, the leg moves relative to the body to satisfy the kinematic constraint of body motion without slipping. While in the air, the leg moves toward a specified target location where it will be set down.

The air path and the target position for foot placement are key factors in keeping the robot stable and maintaining the leg within reachable space. Unlike other approaches which aim to finely control the air path of the leg, our approach leaves the exact path unspecified, and the only requirement is that the foot clear the ground while it is moving forward. The target position is chosen such that once the robot's foot is set down, and is following the body's trajectory, it will pass through the neutral position of that foot at a specific time in the walk cycle (usually about half way through the ground cycle). When calculating the target position it is assumed that the body trajectory will remain unchanged for this short period of time. The position of the body is then evaluated at a future point when the foot is to be set down, and at the time when the foot will be passing through the neutral position. The target position can then be calculated from this data. The target point for setting the foot, the projected velocity of the body along the ground plane, and the current position and velocity of the foot are then used to specify a spline path for the leg to follow through the air.

In some cases our assumption of a constant body trajectory does not hold because the direction of motion is changed in the middle of a step cycle. When this occurs the foot passes close to, but not through the neutral position. Since the path remains smooth, this method performs well in the majority of cases.

Calculations of the air path of the legs and target position also take into account additional specified parameters. These include the desired height and angle of the body, the total cycle time of the walk, and velocities of the legs. Additional hop and sway parameters control the movement of the shoulder and hip joints in vertical and lateral sinusoidal patterns.

The complete set of the 54 motions parameters used to control the walking motions is as follows:

- Neutral positions of the legs (12 parameters: x-pos.,y-pos.,z-pos. for each leg)
- Lift velocities of the legs (12 parameters: x-vel., y-vel., z-vel. for each leg)
- Set velocities of the legs (12 parameters: x-vel., y-vel., z-vel. for each leg)
- Leg pickup time (4 parameters)
- Leg set-down time (4 parameters)
- Body angle
- Body height
- Body hop amplitude
- Body sway amplitude
- Time of one walk cycle
- Maximum height of airpath (2 parameters: front and rear legs)
- Maximum velocities allowed (3 parameters: forward, lateral and angular)

By hand-tuning these parameters we have achieved walks with speeds of up to 235mm/s.

B. Kinematic Test of Parameters

Part of the process of generating new motion parameters is the motion system's kinematic test. Before being executed on the robot, each parameter set is tested to see if the resulting walk maintains the legs within reachable space using the forward kinematic model. If the planned path of the legs is beyond the physical limits of the robot, the motion of the legs can become unpredictable. In many cases this can lead to motions that put excessive strain on the motors and are otherwise dangerous for the robot. The kinematic test procedure calculates the number of 8ms motion frames during which each leg is beyond the physical bounds of the joints. This data allows the user to decide whether the parameter set should be tested on the robot. A small number of error frames are usually allowed for testing.

III. The Learning Algorithm

With the parameterization described above, the problem of optimizing the gait speed becomes a parameter optimization problem in multi-dimensional space. A variety of algorithms exists for solving this type of problem, but the selected approach must possess specific desirable characteristics. The algorithm must:

- Handle non differentiable search space since no gradient information is available.

Algorithm III.1: GA($Population, F, G$)

$F \leftarrow FitnessFunction$
$F_t \leftarrow TerminationFitness$
$G \leftarrow NumberOfGenerations$
$M \leftarrow SizeOfPopulation$
$p_c \leftarrow FrequencyOfCrossover$
$p_m \leftarrow FrequencyOfMutation$
$Population \leftarrow$ RANDOM POPULATION()
CALCFITNESSOFEACHINDIVIDUAL($Population$)
while F($BestIndividual$) $< F_t$ **and** $GenerationNum < G$
do
- **if** $RadiationOn =$ **true**
  - **then** CHECKFORRADIATION()
- $Parents \leftarrow Population$
- $Children \leftarrow$ CROSSOVER($Parents, p_c$)
- $Children \leftarrow$ MUTATION($Parents, p_m$)
- CALCFITNESSOFEACHINDIVIDUAL($Children$)
- $Population \leftarrow$ MERGE($Parents, Children$)
- $Population \leftarrow$ CROPPOPULATION()

return ($BestIndividual$)

TABLE II

PSEUDOCODE FOR THE MODIFIED GENETIC ALGORITHM USED TO OPTIMIZE THE ROBOT'S GAIT.

- Have high convergence rate since every evaluation is expensive.
- Be able to find the true optimum solution independent of the initial parameters.
- Be resistant to noise in the evaluation function.
- Allow a parallelized approach.

Our chosen solution for this problem is an evolutionary approach based on genetic algorithms. The algorithm is based on the conventional GA [7], [8] but has several important modifications[1].

A. Genetic Representation

A summary of our genetic algorithm is presented in Table II. Each walk parameter set is represented by an integer vector with one element per walk parameter. All parameter values are bounded to loosely represent the physical constraints of the robot. For example we bound the height to be in the range of 85-120mm since values outside of this range are impossible to satisfy using a crouched walk.

The crossover operation uses two-point crossover to form two new individuals from two parents. Two crossover points are randomly selected and the vector elements between these positions on the parent vectors are exchanged to form the child vectors. P_c controls how many new individuals are created by applying the crossover operator.

The mutation operator uses Gaussian mutation to create one new offspring from a single parent. The procedure consists of adding a random integer value from a Gaussian distribution to some elements of the parent vector. P_m effects the number of individuals mutated, as well as the number of mutated elements per vector.

The fitness function aims to maximize the forward velocity of the robot and for the purposes of this experiment depends only on the forward distance traveled during the alloted time period.

The algorithm uses overlapping populations in order to always preserve the best individuals in the population. During each generation the algorithm creates a temporary population of children from the parent population by using the crossover and mutation operators. Each new child is tested using the kinematic parameter test and only individuals that pass the test are admitted into the population. Crossover and mutation is repeated until the desired number of valid children is reached. Each new individual is then evaluated and the populations are combined into one large population. The worst M individuals are then removed in order to return the population to its original size. This assures that the population will continue improving or will plateau when no better individuals can be found.

An additional improvement over traditional GAs is an optional method to prevent premature convergence to suboptimal extremes. This method targets individuals that are in a local extreme by adding radiation to the neighboring area [9]. If a large group of individuals is clustered within the same locality, radiation is placed into the middle of that region. In effect this increases the mutation rate in the area dramatically, causing all of the individuals to mutate during the next generation and to disperse to other areas of the space. The influence of the radiation falls off with distance from the radiation point, and the level of radiation decreases over time. We found this method to be useful in controlling the learning behavior of the algorithm as will be described in the next section.

B. The Learning Process

All experiments took place on a robot soccer field designed for RoboCup competitions. This allowed us to reuse many of the features present in CMPack code, such as the vision and localization systems.

During each evaluation the robot walks across the field for a specified amount of time, calculating its velocity based on how far it has traveled. The localization system of the robot uses the uniquely colored landmarks located around the field to triangulate the robot's position and track its progress. Since the uneven movement of the robot's camera during the walk can introduce noise into the measurements, the robot evaluates its starting and ending positions while standing still.

The learning algorithm itself is executed on an off-board computer which communicates with the robots over a wireless network. Each parameter set is sent to an available robot to be evaluated. Once evaluation is completed the robot replies with an evaluation score. The distributed setup provides greater processing power, allows for parallel evaluation by using several robots, and assures that all results are logged without loss of data in case of robot failure. The algorithm can easily scale to an arbitrary number of robots. In our case we were able to simultaneously use four robots on the same field (see Figure 1). The learning process itself is completely autonomous; the only human intervention required is to replace discharged robot batteries.

[1]Implementation based on the GAlib package developed by Matthew Wall at the Massachusetts Institute of Technology.

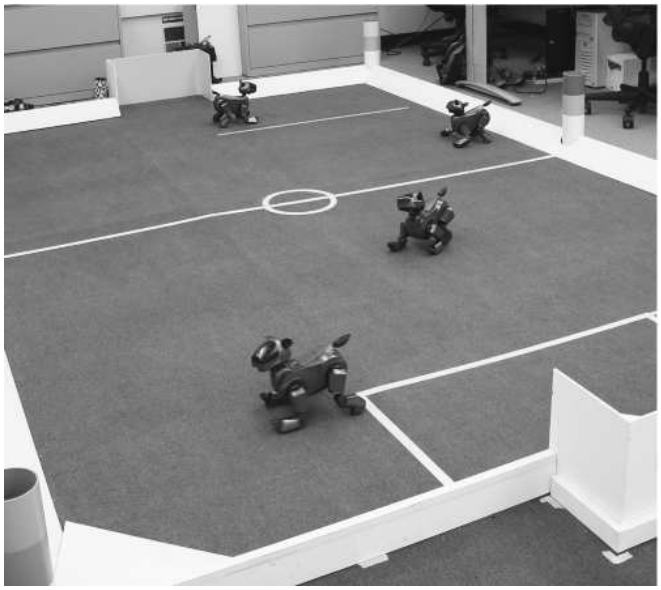

Fig. 1. The training environment. Four robots evaluate different parameter sets simultaneously. The colored markers around the field are used for localization.

For our setup we limited the number of motion parameters being varied to 12. The parameter space is partially simplified by the fact that since the AIBO robots are fairly symmetric, the same foot offset values are used for the left and right legs. Other parameter values were set to fixed values in order to simplify the search space. Since previously motion parameters had always been done by hand, safe values could easily be established for these parameters. The twelve parameters used in the learning process are the ones that have the greatest effect on the walking motion. The parameters that were learned are: body height, body angle, cycle time, front x offset, front y offset, rear x offset, rear y offset, front lift-time offset, front set-time offset, rear lift-time offset, rear set-time offset, and hop amplitude.

The lift-time and set-time offset parameters represent offsets from the original lift and set times of the legs. The traditional gait for the AIBO robots is a trot which involves diagonal pairs of legs moving in unison. The robot has only two feet in contact with the ground at any one time. The lift and set-time offsets allow the learning algorithm to modify the timing of the legs so that front and rear legs move slightly out of synch. Changes to these parameters can cause the robot to have all four feet on the ground for longer periods of time, or to attempt to lift all four feet off the ground at the same time. Previous results have shown that the fastest learned walks tries to keep each foot on the ground only 43% of the time [2].

We found that for the populations size, 30 individuals was a good compromise between a fast algorithm and a diverse population. A larger population increases the evaluation time of each epoch. Since one of our goals is to be able to run the algorithm during competitions and demos where the time before presentation is limited, it is not practical to have very large population sizes since this will limit us to running fewer generations of the algorithm and slowing down the learning. A smaller population size may not provide enough variation, causing the algorithm to converge to local extremes more often than necessary.

The population can be initialized with random or hand selected parameters. The learning process takes place in two phases:

Phase1: The goal of Phase1 is to explore a wide range of parameter values, never focusing too much on one area. The mutation and crossover rates are set high, $P(m) = 0.5$ and $P(c) = 0.6$, causing the parameters to vary over a large range. The radiation method is turned on during this phase, so that if by chance a population does become too homogeneous radiation is placed in that neighborhood. During this phase we avoid converging to any optimum at all but spend the time exploring a wide range of parameters.

Phase2: During Phase2 the goal is to converge to the optimal walk. A handful of the best individuals discovered during Phase1 forms the initial population in this phase. Now the algorithm is changed to explore in detail the neighborhoods surrounding these few parameters. Radiation is turned off, and mutation and crossover probabilities are turned down to $P(m) = 0.3$ and $P(c) = 0.2$.

An important decision point in this approach is the termination of Phase1 and initiation of Phase2. Staying in Phase1 too long will slow down the algorithm because the large mutation rate combined with the radiation factor force the algorithm to constantly find new areas to explore even if it has found a global optimum. Switching into Phase2 too early could also have a slowing effect if none of the parameters from Phase1 are very good. The algorithm would then be limited to exploring a small set of local regions of optimality; since the mutation rate is fairly low, it may take a while for a random mutation to find another area with better values.

During testing we found that this decision is not difficult if the scope of the evaluation function is known. In other words, with the knowledge that walks with velocities upward of 260mm/sec were considered good we terminated the algorithm once several such promising walks had been found. Although it is not possible to guarantee that the algorithm converges to the global optimum since the entire space has not been searched, the algorithm finds highly optimized walks with speeds that have not been previously achieved through hand tuning.

C. Dealing With Noise

Even though during the evaluation the robots remain stationary while calculating their positions, some error due to noisy sensors still remains in the system. Several steps were taken to minimize the effect of these errors.

By keeping only the best members after every generation, it is assured that the next generation will be created from the strongest individuals. Unfortunately, if due to accidental noise one individual receives an abnormally high score, this individual could remain in the population forever

even though it may not be very good. To counteract this problem, walks with reported velocities over 240mm/sec were evaluated two times, and the final score is the average of the two runs.

Additionally, all the individuals in the population are reevaluated every ten to fifteen generations. Although this slows the algorithm slightly because repeated calculations are made, it limits the effect noise has on the system. Individuals that accidentally received high scores because of noise will be reevaluated and will drop down in ranking.

A segment of the learning process showing the effects of this method can be seen in Figures 2 and 3. During the 10th generation one parameter set is evaluated at a very high value due to noise from the sensors. During the following generation the entire population is reevaluated and the individual is ranked correctly. Learning progresses until a plateau is reached.

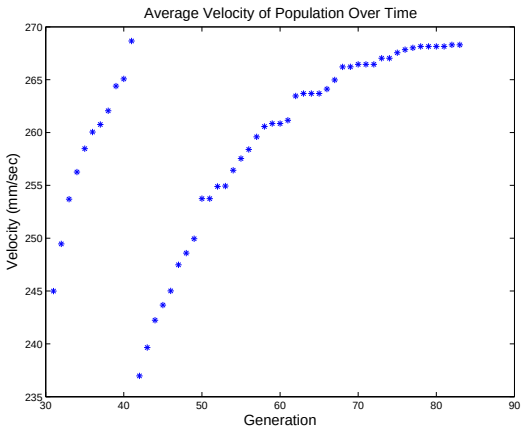

Fig. 2. The average fitness of the population over time. Due to our approach always maintaining the best individuals from combined parent-child population, the average fitness of the populations monotonically increases except when the members of the population are reevaluated during the 11th epoch.

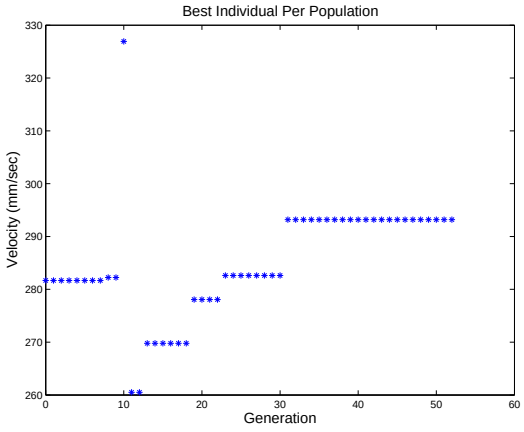

Fig. 3. The fitness of the best individual in each population over time. This is an example of why keeping only a single best individual can lead the algorithm astray if the evaluation function contains noise.

Parameter	Walk1	Walk2
Body Height(mm)	108	107
Body Angle(deg)	11	12
Cycle Time(ms)	679	673
Front x Offset(mm)	108	116
Front y Offset(mm)	60	61
Rear x Offset(mm)	89	93
Rear y Offset(mm)	60	60
Front Lift-Time Offset(%)	0	2
Front Set-Time Offset(%)	4	5
Rear Lift-Time Offset(%)	1	1
Rear Set-Time Offset(%)	-1	-1
Hop Amplitude(mm)	0	0

TABLE III

PARAMETER VALUES FOR THE TWO BEST WALKS LEARNED BY OUR ALGORITHM.

IV. RESULTS

Using the algorithm described above we were able to find two different parameter sets that allow the robot to move at 290 ± 6mm/sec. These results are a dramatic improvement over our previous hand-tuned walk which had the speed of 235mm/sec. The optimal velocity achieved by the robot closely matches the fastest known walk for the AIBOs achieved by the UT Austin Villa team [2] with their learning method.

The optimal parameters learned by the algorithm are shown in Table III. A negative lift or set-time offset percentage represents the event happening earlier than the original lift or set time for that leg. Positive percentages represent events happening slightly later than normal.

Images of the robot walking using these parameters are shown in Figure 4. Both parameter sets resulted in very similar walks. The legs move parallel to the body with very little side velocity. The legs are always picked up cleanly and there is no sliding motion along the ground. Both walks seem to be optimized to take long fluid strides to cover as much ground as possible. An apparent side effect of this is that occasionally the elbows and knees come close enough together that they touch. For one of the motions this contact is fairly rare while in the other walk it is very regular. Since the contact is brief and not very strong, it does not seem to interfere with the walking motion or damage the joints.

Several testing runs of the algorithm during which we experimented with parameters preceded the final learning run. The main learning experiment was started only a single time and ran to completion, successfully finding two optimal parameter sets. The starting population was initialized with random parameters that passed the kinematic test. During Phase1 we executed 40 generations for a total of approximately 1500 field traversals. During Phase2, 70 generations were executed for a total of over 2500 field traversals. The total running time of the algorithm was approximately 5 hours.

V. CONCLUSION

In this paper we present an evolutionary approach to autonomously optimizing fast forward gaits on quadruped robots. Our approach has proven to be very effective and

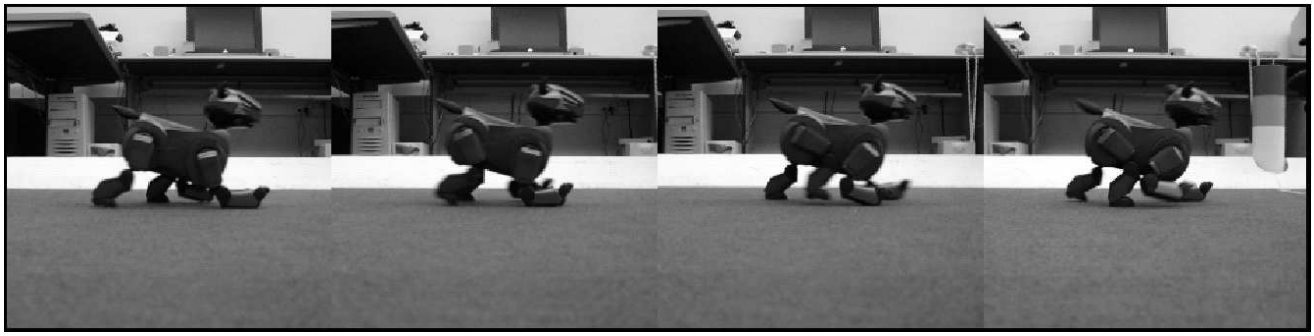

Fig. 4. The fastest learned walk.

has resulted in a 20% increase in speed over our previous walking motion. The algorithm has many strengths over alternate methods. We have shown that even if starting from a random population the algorithm is able to match the best previously known AIBO gait within a matter of hours. The GA-based approach is resistant to noise and avoids converging to local extremes. Additionally, we believe that the results presented in this paper will benefit a large number of other groups that have adopted our motion model.

VI. Acknowledgements

We would like to thank the members of the CMPack team, as well as James Bruce who originally developed the motion system. We would also like to thank Matthew Wall for making the GAlib package available.

References

[1] M. S. Kim and W. Uther, "Automatic gait optimisation for quadruped robots," in *Proceedings of the Australasian Conference on Robotics and Automation*, Brisbane, Australia, December 2003.

[2] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," November 2003.

[3] RoboCup2003, "(http://www.robocup2003.org)." [Online]. Available: http://www.RoboCup2003.org

[4] 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, "A description of the runswift 2003 legged robot soccer team," University Of New South Wales, Tech. Rep., 2003. [Online]. Available: http://www.cse.unsw.edu.au/ robocup/reports.html

[5] P. Stone, K. Dresner, S. Erdogan, P. Fidelman, N. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, D. Stronger, and G. Hariharan, "Ut austin villa 2003: A new robocup four-legged team," University Of New South Wales, Tech. Rep., 2003. [Online]. Available: http://www.cs.utexas.edu/users/AustinVilla/legged/2003/

[6] J. Bruce, S. Lenser, and M. Veloso, "Fast parametric transitions for quadrupedal locomotion," in *Prodeedings of the International Conference on Intelligent Robots and Systems.*, November 2003.

[7] T. Back, "Optimization by means of genetic algorithms," in *36th International Scientific Colloquium*, E. Khler, Ed., Technical University of Ilmenau, 1991, pp. 163–169. [Online]. Available: citeseer.ist.psu.edu/71967.html

[8] R. Storn and K. Price, "Differential evolution - a simple and effi cient adaptive scheme for global optimization over continuous spaces," Berkeley, CA, Tech. Rep. TR-95-012, 1995. [Online]. Available: citeseer.ist.psu.edu/article/storn95differential.html

[9] CERAF, "CERAF (http://klobouk.fsv.cvut.cz/˜ondra/about_ga/ceraf.html)." [Online]. Available: http://klobouk.fsv.cvut.cz/˜ondra/about_ga/ceraf.html