Parameter Optimization of a Signal-Based Omni-Directional Biped Locomotion Using Evolutionary Strategies

Barış Gökçe and H. Levent Akın

Boğaziçi University, Department of Computer Engineering, 34342, Bebek, Istanbul, Turkey
{sozbilir,akin}@boun.edu.tr

Abstract. The ultimate goal of RoboCup depends heavily on the advances in the development of humanoid robots. Flexible walking is a crucial part of playing soccer and bipedal walking has been a very active research topic in RoboCup. In this paper a signal-based omnidirectional walking algorithm for the Aldebaran Nao humanoid robot is presented. Inspired from the existing methods in the literature, the proposed method models the omni-directional motion as the combination of a set of periodic signals. The parameters controlling the characteristics of the signals are encoded into genes and *Evolutionary Strategies* is used to learn an optimal set of parameters.

1 Introduction

Locomotion is one of the main modules for a humanoid robot expected to work in human inhabited environments. Consequently, research on biped walking have become one of the most exciting topics in robotics. Most of the researchers prefer to use precise knowledge about the robot dynamics; mass, location of center of mass, inertia of each link to generate reference trajectory for a stable walking motion. In this approach, the most popular method is to keep the point on the ground where these active forces are zero which is called as Zero-Moment Point(ZMP) into the support polygon [1]. Another approach for biped walking, Passive Dynamic Walking(PDW), is inspired from a bipedal toy that can walk on a slope without any explicit control [2]. This approach is mainly based on gravity and inertia. In this approach, motor control is used to generate the effects of gravity in level floors or shallow up slopes. Recently usage of Central Pattern Generators(CPG) have become popular. The CPG approach is inspired from the symmetric characteristic of vertebrate animals' locomotion[3]. In these animals, locomotion is generated by synchronized rhythms of different gaits. In the application of this idea, a signal is generated for each joint and these signals are synchronized by a central signal. There are mainly two approaches; model-driven and model-free. While the properties of the signals are determined in the model-driven approach, they are left to the training procedure in the model-free approach. Because neither stability nor feasibility is considered in CPG based biped walking, the training procedure is a very crucial part of this type of walking.

In [4], a model for a CPG based biped walking is proposed. In this work, the position of the leg is modeled by three features and motion is defined as a combination

J. Ruiz-del-Solar, E. Chown, and P.G. Ploeger (Eds.): RoboCup 2010, LNAI 6556, pp. 362–373, 2011.

of five movements. The features are calculated with respect to these movements and synchronization is obtained by a central clock. They have applied feedback by using gyroscopes and foot pressure sensors in a later study[5]. While the foot pressure sensors are used to manage the central clock, the gyroscopes are used to reduce the angular velocity by using P-controller. Additionally, policy gradient reinforcement learning and particle swarm optimization is used to optimize the walking speed.

The Standard Platform League (SPL) in RoboCup has changed the platform from 4-legged robot to a humanoid robot *Nao* produced by Aldebaran Robotics [6] in 2008. Since then many studies have been carried on the locomotion module of this robot. In [7], preview control is used to keep the ZMP in the desired path and it requires precise modeling of the robot. On the other hand, Shafi et al. [8] use *Truncated Fourier Series* to generate angular velocities. After the implementation of biped locomotion, *Genetic Algorihm* is used for optimization.

In this work, our main concerns are two important constraints for the locomotion in soccer domain: stability, and the speed to approach a point. For this purpose, we propose an omni-directional model-driven CPG based biped walking algorithm with a training procedure to satisfy these constraints. In the model of locomotion, the position of the leg is represented by two terms; *leg extension* which represents the distance between the foot and the trunk, and *leg angle* representing the angle between the pelvis plate and the line from hip to ankle. In addition to the representation of leg with respect to the trunk, the main motion is divided into four components; shifting, shortening, swinging and loading. Each component is defined as a continuous and periodic signal and the whole motion is generated by summing up these signals. Since the balance and the speed of the bipedal walking algorithm highly depends on the synchronization of the signals, coefficients and constants of the signals are defined as the parameters of the system and the whole system is trained by *Evolutionary Strategies*[9] engine. As a test platform, we used Aldebaran Nao humanoid robot. Firstly, we tested the algorithm in a simulator, Webots and then realized it on real robots.

In the next section, we analyze the proposed biped walking algorithm. Section 3 describes the training procedure for the proposed parametric walking algorithm and Section 4, the experimental results in the simulation and real-world environments. Conclusions and future works are given in Section 5.

2 Proposed Biped Locomotion Approach

In this study, we developed a model for CPG-based omni-directional biped walking algorithm with variable speed. While modeling the system, we use a modified form of the approach given in [4]. The tests on bipedal walking have shown us that changing the angle between the foot plate and pelvis plate causes disturbances on the balance when an unexpected landing of the swinging leg occurs because of the rigidness and existence of sharp edges on the foot. This problem is solved by keeping that angle zero. That way, if the foot touches the ground before or after the expectation, the foot plate will be parallel to the ground and the reaction force will be distributed to not a single point but the whole surface. Therefore in the modeling of the leg we used two features to determine the position and orientation of the foot with respect to the pelvis. These features are

- *Leg Extension*: The distance between the hip joint and the ankle joint. It determines the height of the step while moving.
- *Leg Angle*: The angle between the pelvis plate and the line from the hip to the ankle.

In addition to the model of the leg, we divide the main motion into four components; *shifting, shortening, loading,* and *swinging*. Each of these correspondnds to a component of the motion. They are defined as continuous and periodic signals. Since the algorithm is developed for omni-directional and variable speed walking, in addition to the central clock, the speed values also have effects on the signals. Speed components are forward (v_{fwd}), sideways (v_{strafe}) and turn (v_{turn}).

Central Clock: While modeling such a system, a central clock is required to synchronize the signal. Therefore, a central clock (ϕ_{trunk}) is generated for the trunk which is between $-\pi$ and π. Each leg is fed with different clocks (ϕ_{leg}) with $lid \times \frac{\pi}{2}$ phase shift where lid represents leg ID; -1 for the left leg and +1 for the right leg. This way, the phase difference between legs is preserved as π. In the calculations of the motion features for each leg, the corresponding phase is considered and the features depend on that value.

- ***Shifting Motion:*** In the shifting motion, lateral shifting of the center of mass is obtained. For this purpose, a sinusoidal signal given in Equation 1 is used. In the equation, a_{shift} determines the shifting amount and it depends on the speed values as shown in Equation 2.

$$\theta_{shift} = lid \times a_{shift} \times sin(\phi_{leg}) \tag{1}$$

$$a_{shift} = c_{shift} + \frac{c^f_{shift}}{|v_{fwd}|} + \frac{c^s_{shift}}{|v_{strafe}|} + c^t_{shift} \times |v_{turn}| \tag{2}$$

- ***Shortening Motion:*** The second important motion is the shortening signal. The shortening signal is applied in only a part of the shifting phase. A constraint to prevent both legs from being in the shortening phase at the same time is also added. Therefore, another clock is generated from the clock of the leg with the Equation 3. According to the clock generated for the shortening phase, a part of leg extension is calculated with Equation 4. This phase difference determines the time to shorten the leg.

$$\phi_{short} = \frac{2 \times \pi}{\pi - 2 \times p_{short}} \times (\phi_{leg} + \frac{\pi}{2}) \tag{3}$$

$$\gamma_{short} = \begin{cases} -(c_{short} + \sqrt{v^2_{fwd} + v^2_{strafe}}) \times (cos(\phi_{short}) + 1) \; if - \pi \leq \phi_{short} < \pi \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad otherwise \end{cases} \tag{4}$$

- ***Swinging Motion:*** Swinging is the most important part of the motion. In this part, the leg is unloaded, shortened, and moved along the way of motion which reduces the stability of the system considerably. Additionally, we have another constraint for this part. Normally, the swinging signal is always applied. In the double support phase, the weight of the body is shifted through the motion direction, the distance

between the legs for further steps is changed in the single support phase. In the Aldebaran Nao humanoid robot [6], which is our development platform, the yaw joints at the hip of each leg are connected to each other and work in the opposite direction. It is impossible to change the orientation of the body in the double support phase. Because of changing the behaviour of the leg angle feature and the explained additional constraint, two additional clocks are generated; one for the yaw component of the leg angle feature (Equation 5), one for the pitch and roll components (Equation 6). This phase differences determine the time to swing the swinging leg. Therefore, the calculations of the yaw component and other components of leg angle feature are different (Equations 7, 8 and 9).

$$\phi^{y}_{swing} = (\frac{\frac{\pi}{2}}{2 \times p_{swing} - \pi}) \times (2 \times p_{swing} + \phi_{leg}) \tag{5}$$

$$\phi_{swing} = c_{swing} \times (\phi_{leg} + \frac{\pi}{2} + p_{swing}) \tag{6}$$

$$\theta^{y}_{swing} = |v_{turn}| \times (lid \times cos(\phi^{y}_{swing} - (sign(v_{turn}) + lid) \times \frac{\pi}{4})) \tag{7}$$

$$\theta_{swing} = \begin{cases} sin(\phi_{swing}) & if -\frac{\pi}{2} \leq \phi_{swing} < \frac{\pi}{2} \\ b \times (\phi_{swing} - \frac{\pi}{2}) + 1 & if \frac{\pi}{2} \leq \phi_{swing} \\ b \times (\phi_{swing} + \frac{\pi}{2} - 1) & otherwise \end{cases} \tag{8}$$

$$b = -\frac{2}{2 \times \pi \times p_{swing} - \pi} \tag{9}$$

According to the speed and direction of the motion, θ_{swing} is distributed to the components of leg angle features with the Equations 10 and 11.

$$\theta^{r}_{swing} = -v_{fwd} \times \theta swing \tag{10}$$

$$\theta^{p}_{swing} = v_{strafe} \times \theta swing \tag{11}$$

– ***Loading Motion:*** The last motion of the step is loading which is also not always applied. As in shortening phase, we added the same constraint which prevents both legs from being in the loading phase at the same time. The formula to calculate the clock for the loading phase is given in Equation 12. This phase difference determines the time to load the weight of the robot to the stance leg. This phase affects only the leg extension feature as given in Equation 13.

$$\phi_{load} = \frac{2 \times \pi}{p_{load}} \times \phi_{leg} + \pi; \tag{12}$$

$$\gamma_{load} = \begin{cases} (c_{load} + 0.5 \times (1 - cos(|v_{fwd}|))) \times cos(\phi_{load}) + 1) & if -\pi \leq \phi_{load} < \pi \\ 0 & otherwise \end{cases} \tag{13}$$

$$\theta^r_{leg} = \theta^r_{swing} + \theta_{shift} \tag{14}$$

$$\theta^p_{leg} = \theta^p_{swing} \tag{15}$$

$$\theta^y_{leg} = \theta^y_{swing} \tag{16}$$

$$\gamma = \gamma_{short} + \gamma_{load} \tag{17}$$

There is a direct relation between the joints of the robot and motion features. In other words, by using these features, the values for hip, knee, and ankle joints can be calculated easily as summarized below:

- There is only one joint that changes the distance between the hip and ankle which is the leg extension. For this reason, the angle of the knee joint is determined according to the value of only the leg extension feature of the motion. From the five sub-motions of a step, the leg extension feature is calculated and it is between -1 and 0. While calculating the angle of the knee joint from the extension, the normalized leg length, *n*, is calculated as $n = n_{max} + (1 - n_{min}) \times \gamma$. The knee joint value is directly proportional to the arccosine of leg length; $\theta_{knee} = -2 \times acos(n)$.
- Another special case for the motion is its turn component. If we consider the physical construction of the robot, there is only one joint which turns the body; the hip yaw joint. Therefore, the turn component of the motion directly determines the value for the hip yaw joint which means $\theta_{hip^y} = \theta_{leg^y}$.
- There are two more joints at the hip. By definition, the roll and pitch components of the leg angle feature directly determines the angles of the hip joints. This means that $\theta_{hip^r} = \theta_{leg^r}$ and $\theta_{hip^p} = \theta_{leg^p}$. In order to compensate the effects of the leg extension on the pitch component of the leg angle feature, half of the knee should be rotated around the turn component of the motion and added to the hip pitch and roll joints. In other words, the pitch and roll components of hip joints can be calculated by the Equation 18.

$$\begin{pmatrix} \theta^r_{hip} \\ \theta^p_{hip} \end{pmatrix} = \begin{pmatrix} \theta^r_{leg} \\ \theta^p_{leg} \end{pmatrix} + R_{\theta^y_{hip}} \times \begin{pmatrix} 0 \\ -0.5 \times \theta_{knee} \end{pmatrix} \tag{18}$$

- Ankle joints are calculated by applying the inverse of the rotation around the turn component of the motion to the negative of the leg angle feature. In order to compensate for the effects of the leg extension on the pitch component of the foot angle feature, half of the knee should also be added to the ankle pitch joint. In other words, the pitch and roll components of ankle joints can be calculated by the Equation 19.

$$\begin{pmatrix} \theta^r_{ankle} \\ \theta^p_{ankle} \end{pmatrix} = \begin{pmatrix} 0 \\ -0.5 \times \theta_{knee} \end{pmatrix} + R^{-1}_{\theta^y_{hip}} \times \begin{pmatrix} -\theta_{leg^r} \\ -\theta_{leg^p} \end{pmatrix} \tag{19}$$

3 Parameter Optimization Using Evolutionary Strategies

After the successful realization of the proposed model, we focus on the stability and speed of the locomotion. We have defined an open-loop model of motion which does not consider stability. Therefore, there is no guarantee for the stability. In addition to the stability of the motion, it is difficult to define the speed by using only the signals. Even if we solve these problems, the effects of the signals on stability and speed may change with the properties of the floor surface like friction, and softness. Therefore, training is an essential part of the locomotion for stability, speed and adaptability. In order to define an optimization on the signals, we use the characteristics of the signals as the parameters of the locomotion and problem is reduced to a parameter optimization problem. As an optimization method, we used *Evolutionary Strategies* [9] because it works on the parameters with real-values. Additionally, the *population size* is an important hyper-parameter in the optimization algorithm which determines the number of individuals existing in a generation. Our model consists of thirteen parameters which determine the walking pattern. These parameters are given in Table 1.

3.1 Fitness Function

The most crucial component of an evolutionary algorithm is the fitness function which determines the goodness of the parameter set. While constructing the fitness function, we analyzed the problem we are trying to optimize. In the soccer domain, gaining possession of the ball is the most important task to be successful in the game. From this point of view, we can conclude that our fitness function should have two components:

- **Reaching target:** The first component evaluates the success of the parameter set in reaching the target object. We measure the distance the robot traveled and divide it by the distance it should have taken. This value should be around 1 if the robot can accomplish the task with success. The importance of this part arises when the robot can not reach the target. The parameter set which can get closer to the target will get more reward from this part.
- **Speed:** The second component considers the speed of the robot. For this purpose, a maximum speed is defined as 20 cm/s and closeness of the obtained speed is used as the second part of the fitness function. That way, slower parameter sets can be punished.

By multiplying these two parts, the fitness value for the corresponding parameter set is calculated. At that point, another important property of the fitness function comes into question. The parameter sets which can accomplish the task should always be better than those which fail. If we analyze our fitness function, it should always be between 0 and 1, but faster parameters sets which cannot reach the target position can have better fitness values than the successful but very slow parameter sets. In order to give reward success, 1 is added to the fitness value if the robot can reach the target. As a result of this correction, while fitness values of successful parameter sets are between 1 and 2, that of failed parameter sets are in between 0 and 1. We can formulate the fitness function as in Equation 20.

Table 1. Parameters of the walking algorithm

Parameter	Description
p_0	Frequency
p_1	c_{shift}(Shifting Constant)
p_2	c^f_{shift}(Shifting Forward Speed Coefficient)
p_3	c^s_{shift}(Shifting Strafe Speed Coefficient)
p_4	p_{load}(Phase Difference of the Loading with the Leg)
p_5	p_{short}(Phase Difference of the Shortening with the Leg)
p_6	c_{short}(Shortening Constant)
p_7	c^t_{shift}(Shifting Turn Speed Coefficient)
p_8	c_{load}(Loading Constant)
p_9	c_{swing}(Swinging Speed)
p_{10}	p_{swing}(Phase Difference of the Shortening with the Swinging)
p_{11}	n_{max}(Maximum Height of the Hip)
p_{12}	n_{min}(Inverse Proportion for the Step Height)

$$Fitness\ Value = \begin{cases} (\frac{d_{traveled}}{d_{target}} \times \frac{v}{v_{max}}) + 1 & if\ task\ is\ accomplished \\ \frac{d_{traveled}}{d_{target}} \times \frac{v}{v_{max}} & otherwise \end{cases} \tag{20}$$

In addition to the definition of the fitness function, the way to calculate fitness value is also very important. In order to get rid of noisy calculations, each parameter set is evaluated five times. The resulting fitness value is the average of three calculations by excluding the minimum and the maximum values.

4 Experiments and Results

Since the fitness function defined in the previous section summarizes the expectations from the bipedal walking in the soccer domain, the training procedure requires the robot to move between two targets which are the beacons used in the 2008 set up of the 4-legged league. They are placed at a distance of 450 cm. The important drawback of the beacons is the difficulty of perceiving these objects if the robot is very close to them. Therefore, the robot gets as close as 100cm to the target and turns to go to the other one as the next trial. This gives us a path with length of 250 cm which is sufficient to measure the success of the parameter set.

All processes are handled by two threads on the robots. While one, called *cognition*, is used to perceive the environment and make plans, the other one, called *motion*, is used to run the motion engine with the speed values calculated by the *cognition* thread. For the perception of the environment and low-level decisions(*search_object*, *goto_object* etc.), we used the code developed in our team. Because there is no role in the match to go between two objects continuously, to realize this behaviour we developed a 7-state high level finite state machine, shown in Fig.1.

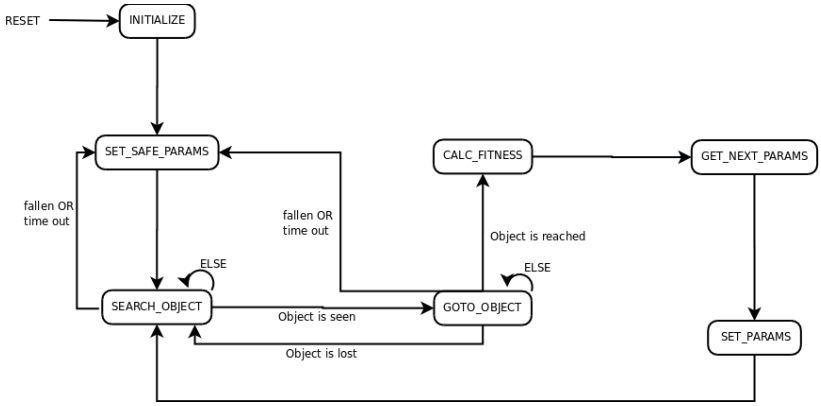

Fig. 1. Finite State Machine(FSM) used as training behaviour

4.1 Experiment Environments

Experiments are conducted in two different environments; simulation and real-world. We aim to obtain two different results with two different experiments. In the first experiment, we aim to conduct a wider search in the domain and we distributed the initial population uniformly in the whole search space. Because the population covers the domain, we should have a bigger population and continue for more generations. Therefore, this experiment is considerably time-consuming. On the other hand, our goal for the second experiment is to fine-tune the parameter set obtained at the end of the first experiment for a different environment in the real-world with a limited time. For this purpose, the initial population is distributed around the base parameter set.

Experiments in the Simulation Environment: As the simulation environment, we used the field constructed by Cyberbotics[10] for the Webots' RoboCup package. There are mainly two important motivations to use this tool. The first one is that it is one of the two supported 3D simulators by RoboCup and the second one is to execute the same binary on the real robot. In the experiments on the simulation environment, we worked on the controllers of two nodes; *Nao* and *Supervisor*. *Supervisor* is the master node which has access to each node in the environment and permission to move the robot from one point to another and change the orientation. In the simulation experiments, this node is used only to correct the posture of the robot when it falls down. *Nao node* is the model of the robot we are using. This node is used to run the *NaoQi* middleware which is developed by Aldebaran and links the shared libraries including our methods. As controller, we have used the one provided by Aldebaran called *nao_in_webots*. The main advantage of this controller is that it lets us to load the same module implemented for the simulation to the real robot.

In the simulation experiment, the population has 20 individuals. Simulation results for the average and the overall best fitness values are given in Fig.2. As a result of the experiment, the best fitness value is increased from 1.23 to 1.29. In order to calculate performance improvement, we should consider the frictional part which determines the

proportion of the average speed to the maximum speed, 20 cm/s. Therefore, the average speed to go to the target is increased from 4.6 cm/s to 5.8 cm/s which can be interpreted as an improvement of more than 25 percent. Although, we have optimized the parameter set for omni-directional walking, we also obtained an improvement for the forward walking which is the most dominant motion to reach the target. While the forward speed of the best parameter set in the initial population is 5.2 cm/s, that of the best parameter set obtained at the end of the training applied for omni-directional walking is 7.1 cm/s which is an improvement more than 35 percent.

Average fitness value of a generation shows us the fitness of the generation to reach the target. When we analyze the average fitness values of the population during the training procedure, it started around 0.1 and increased sharply until the 7th generation. In the later generations, average fitness value started to oscillate and the overall best fitness value did not change. From this point of view, we concluded that the population has converged. This result was important in order to decide when to finish the training in real-world experiment where we have more restricted time limitation.

Experiments in the Real World Environment: In the first experiment, we made a wide search in the domain of the parameter set. However the results obtained from the simulation does not exactly fit to the real world environment. So, we need another experiment in the real world. Because of hardware and time limitations, we aim to fine-tune the parameter set in the real world. For this purpose, we generate initial population in the neighborhood of the parameter set which will be fine-tuned. As the experiment environment, we constructed the same field as the simulation environment with the carpet and beacons used in the 2008 set up of 4-legged league. One important difference with the simulation experiment is that no supervisor is used to correct the posture. Its own *'getUp'* movement is used when the robot falls down. Therefore, time spent to get up is automatically added as a punishment.

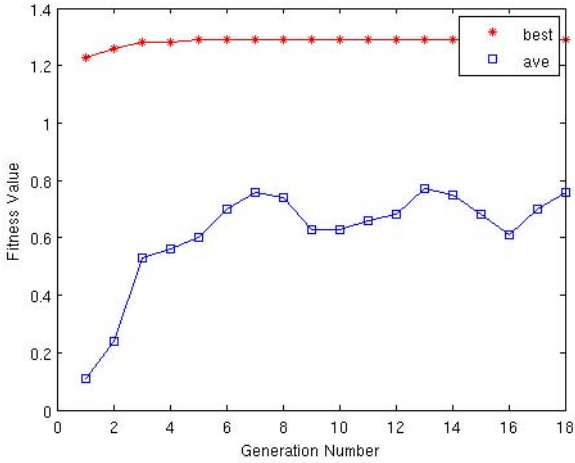

Fig. 2. Results on simulation experiments for the parameter optimization

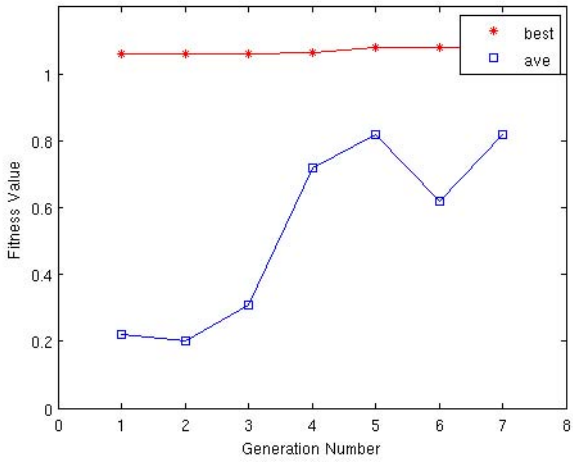

Fig. 3. Results in the real world experiments for the parameter optimization

The results of the experiment for the average and the overall best fitness values are given in Fig.3. In the experiment, we obtained results similar to the simulation experiment with smaller improvements. At the end of the experiments in the real-world, the best fitness value is increased from 1.061 to 1.08045. When we calculate the average speed to go to the target as in the simulation experiments, we find that it is increased from 1.220 cm/s to 1.609 cm/s which is an improvement more than 30 percent. When we analyzed the effects of the fine-tuning on the forward speed, we see an increase from 4.6 cm/s to 6.5 cm/s which is an improvement more than 45 percent.

As we explained in the previous section, the average fitness value of the population shows us the goodness of the current population. Additionally, since fractional parts in the fitness values are very small and changes very slowly in the real-world experiment, the average fitness value can be interpreted as the proportion of the successful parameters in the population. The population has 10 individuals in the real-world experiment. When we analyze the average fitness values of the population during the fine-tuning process, it starts around 0.2 which means only two individuals, where one of them is the hand-tuned one, are classified as successful. For the next generations, the number of successful individuals in the population increases, and eventually most of the individuals can accomplish the task. After the 5th generation, average fitness value became close to the best fitness value and started to oscillate. Therefore, we concluded that the population has converged.

5 Conclusion

In this work, we proposed a model for a CPG-based bipedal walking algorithm. While constructing the model, we started with the representation of the position and orientation of the foot. In this representation, we used two features for the distance and

orientation of the foot with the torso. In the representation of the orientation, we added a constraint to keep the bottom of the foot parallel to the ground surface. This constraint is necessary because of the rigidity of the sharp edges of the sole. By holding the foot parallel to the ground, we distribute the ground reaction force to the whole surface and the stability of the robot is not affected that much when the swinging leg is landing. In addition to the representation of the foot, we divided the motion with four components and represented each component with the synchronous periodical signals. Omni-directional and variable speed movement capabilities were also obtained by embedding the properties of these capabilities into the definitions of these signals.

In addition to the implementation of the bipedal walking algorithm, we applied a parameter optimization technique, namely *Evolutionary Strategies* to optimize the parameter set of the locomotion system. We trained our bipedal walking algorithm in both simulation and real environments with different population sizes. The fitness function is the same for both environments which is the main aim of the locomotion system: approach a target as fast as possible. In the simulation experiment, the initial population is uniformly distributed to cover the parameter domain as much as possible. The optimization procedure for the real world experiment is more like fine-tuning instead of training. In this optimization procedure, the initial population is distributed in the neighborhood of the initial parameter set. Because the experiment is run in a small search space, improvement in the fitness value is not as good as the one in simulation.

Although a stable omni-directional walking is obtained with the proposed model, it is open-loop and vulnerable to the cumulative disturbances on the balance. As a future work, a feedback from foot pressure sensors can be used to increase the stability of the movement. Additionally, the training procedure can be distributed and parallelized. By the help of parallelization, we can reduce the training time and increase the search space.

Acknowledgments

This project was supported by Boğaziçi University Research Fund project 09M105.

References

1. Vukobratovic, M., Juricic, D.: Contribution to the synthesis of biped gait. IEEE Transactions on Bio-Medical Engineering 16, 1–6 (1969)
2. McGeer, T.: Passive dynamic walking. The International Journal of Robotics Research 9, 62–82 (1990)
3. Pinto, C., Golubitsky, M.: Central pattern generators for bipedal locomotion. J. Math. Biol. (2006)
4. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: ICRA, pp. 1597–1603 (2006)
5. Faber, F., Behnke, S.: Stochastic optimization of bipedal walking using gyro feedback and phase resetting. In: Proceedings of the International Conference on Humanoid Robots (Humanoids) (2007)
6. Aldebaran Robotics, `http://www.aldebaran-robotics.com/eng/`

7. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. Robotics and Autonomous Systems 57, 839–845 (2009)
8. Shafii, N., Seyed Javadi, M.H., Kimiaghalam, B.: A truncated fourier series with genetic algorithm for the control of biped locomotion. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (2009)
9. Beyer, H.-G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
10. Cyberbotics (2010), `http://www.cyberbotics.com/`