Modeling and Learning Walking Gaits of Biped Robots

Matthias Hebbel, Ralf Kosse and Walter Nistico
Robotics Research Institute
Information Technology Section
Universität Dortmund
Otto-Hahn-Str. 8, 44221 Dortmund
Email: *forename.surname*@uni-dortmund.de

***Abstract*— This paper describes an open loop modeling of a walking gait by mimicking the human walking style. A parameterizable model for the leg and arm movement will be developed. For finding the parameters of these problem classes often machine learning approaches are used. Thus, several optimization techniques are discussed and finally Evolution Strategies chosen for the optimization process. The best fitting parameters like population size or the selection operator are then found out by doing walk evolution with different configurations of the strategy in a robot simulator. Finally the best performing strategy is used to evolve a forward walk on a real robot.**

I. Introduction

The ability for robots to operate in human made environments is a very challenging task. The movement of the robot in human environments requires the possibility to adapt to the ground like a flat surface, soft surface or stairs. To overcome these problems, a lot of research is done in the area of legged robots. The task of walking becomes more and more difficult with a smaller amount of legs. While robots with six or four legs could always remain stable on the ground while walking, the problem of generating a walking gait for biped robots is much harder.

In this paper we first will describe a parameterized model which generates trajectories for the feet and arms of the robot. This model contains variables for the trajectories like step size, step height, timing etc. Finding the walking parameters by hand is not feasible, because the search space is too large and the parameters strongly depend on each other, i.e. optimizing them separately is not possible, but machine learning approaches can be used to optimize these parameters. Thus an overview of possible techniques to find these parameters is given. Afterward, chosen strategies with different population sizes and selection operators are used to optimize the walk in the simulator. Finally, we present the results of a walk learning experiment which was done on a real robot. For these experiments we have developed a system which allows the robot to learn to walk fully autonomously.

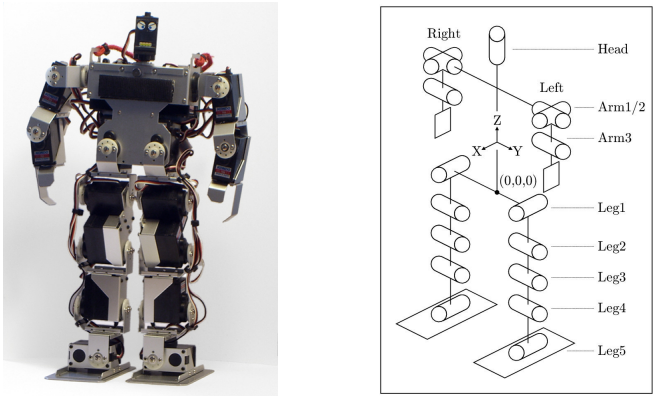
II. The Experiment Platform

First we will explain the used humanoid robot model including the degrees of freedom, afterward the used simulator for the evaluation of the learning strategy will be explained.

A. The robot

The used robot is the KHR-1 from the Japanese manufacturer Kondo and is presented in Fig. 1(a). It is being delivered as a construction set and has been partially modified for this work. The KHR-1 has 17 degrees of freedom, all of them are rotatory joints realized by servo motors. Each leg has 5 degrees of freedom, each arm has 3. The head of the robot can also be rotated but is not considered in the walking model.

A particularity is the arrangement of the servo joints in the legs: the robot does not have the ability to rotate the legs.



(a) The "Kondo KHR-1" robot.

(b) Kinematic structure including the defined coordinate system.

Fig. 1. The used robot with the available joints.

Small hardware modifications had to be done to let the walk evolution run on the KHR-1. First a controller board has been developed to control the 17 servos of the robot. The controller board uses a serial communication to receive requested joint values and send the current sensor readings. The board can either communicate with an external PC or a PDA. Additionally the board is equipped with an accelerometer and a gyroscope to be able to detect the orientation in space.

Further the small plastic gears in the servos have been replaced by gears of metal. The small plastic gears have to stand the biggest torsion forces and tended to break frequently. The metal gears turned out to have a much longer lifetime and less play but created also a bigger friction inside the servos which resulted in slightly weaker servos. The controller board

Proceedings of the Workshop on Humanoid Soccer Robots
of the 2006 IEEE-RAS International Conference on Humanoid Robots
Genoa, Italy, December 4, 2006
pp. 40-48
ISBN: 88-900426-2-1

can update the servos at a freuency of up to 100 Hz but tests have shown that an update rate of 50 Hz shows no apparent difference. Thus we controlled the servos with 50 Hz.

We define a coordinate system shown in Fig. 1(b). For an upright standing robot the X axis points forward, the Y axis to the left side of the robot and the Z axis points upwards.

B. The simulator

To find an appropriate learning strategy we used a simulator developed by Bremen University [1]. It is based on SimRobot including the OpenDynamicsEngine [1] for the physical simulation. The simulator is equipped with a model of the KHR-1, but the physical simulation is not accurate and realistic enough to do walk evolution and use the found parameters on the real robot. However, the general behavior is comparable with the real robot. For that reason we used the simulator only to compare different learning strategies and did the real walk evolution afterward on the real robot with the learning strategy proven to be best in the simulation.

III. Moving the Limbs

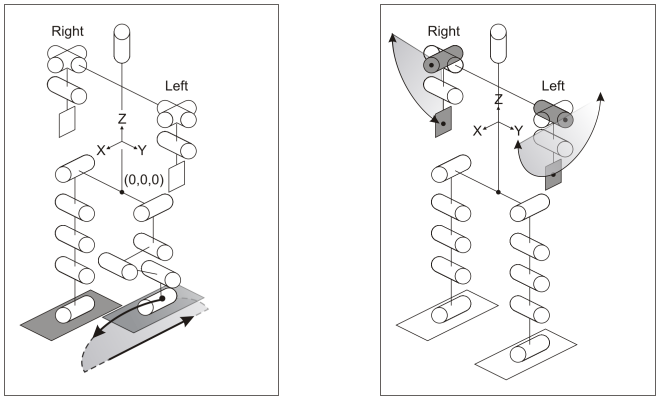
In this section we explain a parameterized walking model which moves the legs and arms in order to walk. At first only straight forward walking is explained and afterward extended to an omnidirectional movement. The walking model is based on the human walk to put as much previous knowledge as possible in the model in order to keep the number of parameters and the search space for parameter optimization as small as possible.

A. Trajectories for the legs

All existing walking gaits in nature have in common that the legs move in two phases: The phase when a foot is on the ground to push the body forward and the phase when it is in the air to prepare the next step on the ground. In the period when the leg touches the ground, it moves relative to the body contrary to the walking direction; during the air phase it moves into the walking direction. The movement of the feet on the ground has to be a straight line. The movement in the air however can nearly be arbitrarily chosen. For our walking model we apply the constraint that the time for the foot being in the air equals the time when being on the ground.

The movement of a single foot while walking can be described as a trajectory in the three dimensional space. However, for efficient walking a two dimensional description is sufficient. The trajectory is spanned by a vector pointing into the walking direction and a vector pointing opposite to the force of gravity. A third dimension could only cause a movement which does neither contribute to move into the requested direction nor have an effect on the height that the feet has to be lifted during the air phase. The trajectory has to be a closed path which a foot travels completely along during a whole step cycle.

The way the feet are controlled by the *walking engine* along the specified trajectory is illustrated in Fig. 2(a). The required joint angles to reach the position of the feet on the trajectory are calculated by means of *Inverse Kinematics*. A closed form solution for the inverse kinematic calculation for legs with more than three joints usually is not existent. But by making specific assumptions, e.g. the knee always bends to the front, the system of equations becomes well-defined. Techniques which iteratively approximate the according joint angles can also be used but are generally more computationally expensive than the direct solution [2]. Here for simplicity we assume that the feet always have to be parallel to the ground. For robots with toe-like feet [3] with a joint in the sole of the foot like the robot "Toni" from the University of Freiburg, this approach has to be extended to guarantee a human-like movement of the feet. The trajectory on which both feet move on is identical, but the phase movement is shifted in time by half of the duration of a whole step, i.e. while one foot starts to touch the ground, the other one is about to leave it.



(a) The movement of a foot along the trajectory.

(b) Possible trajectories of the arms in the $X-Z$ and the $Y-Z$ plane.

Fig. 2. Trajectories of the feet and the arms.

Due to the fact that the movement of the feet in the air can be nearly arbitrarily chosen, an overview and evaluation of possible trajectories will be presented. In the following trajectory explanations we assume that the timing parameter is always in the interval $0 \leq t < 1$. The ground phase movements for all trajectories are defined to be in the time interval $0.5 \leq t < 1$ and follow the equation

$$\vec{f(t)} = \begin{pmatrix} x(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} \frac{s_l}{2} - 2s_l(t-0.5) \\ 0 \end{pmatrix}$$

with $x(t)$ and $z(t)$ describing the foot position and the time s_l defining the step size.

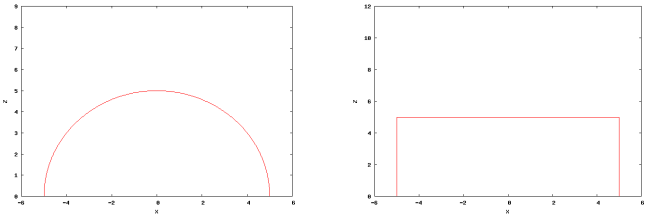
1) Semi-circle: An exemplary semi-circle trajectory is presented in Fig. 3(a). In consequence of the circular movement, the height the feet are lifted while walking is half of the length of the step. Assuming a constant speed of the feet on the trajectory, one parameter is enough to describe the semi-circle. Experiments with the semi-circular locus have shown that the fact that the foot touches and leaves the ground perpendicular to it is very helpful, but the coupling of step height and

[1] http://www.ode.org/

step length leads especially for bigger step sizes to stability problems since high lifted feet cause a lifting of the center of gravity of the robot.

The semi-circle trajectory with the step length s_l for $0 \leq t < 0.5$ is given by:

$$f(\vec{t}) = \begin{pmatrix} x(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} -\cos(2\pi t) \cdot \frac{s_l}{2} \\ \sin(2\pi t) \cdot \frac{s_l}{2} \end{pmatrix}.$$



(a) The semi-circle trajectory. (b) The rectangular trajectory.

Fig. 3. Trajectories for the feet with step length 10 and step height 5 in the X-Z plane.

2) Rectangle: The rectangular movement of the feet can be described by two parameters for the shape of the trajectory, the length of the step s_l and the height of the foot lifting s_h (see Fig. 3(b)). Additionally, relative timing parameters have to be defined: t_1 is the time a foot is in the lift phase, t_2 defines the time to move a foot forward and t_3 is the time needed to lower a foot. As defined, the time of a foot in the air has to be equal to the time on the ground, the relative ground time is 0.5 and $t_1 + t_2 + t_3 = 0.5$. With these parameters, the rectangular trajectory is given by:

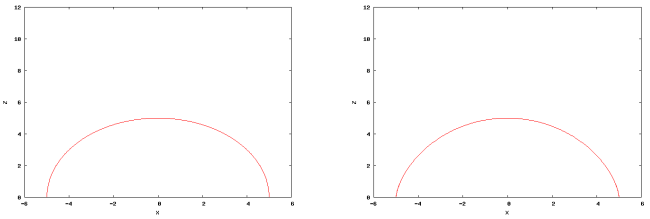
$$\begin{pmatrix} x(t) \\ z(t) \end{pmatrix} = \begin{cases} \begin{pmatrix} -\frac{s_l}{2} \\ t \cdot \frac{s_h}{t_1} \end{pmatrix} & \text{for } 0 \leq t < t_1 \\ \begin{pmatrix} 2s_l(t - t_1) - \frac{s_l}{2} \\ s_h \end{pmatrix} & \text{for } t_1 \leq t < t_1 + t_2 \\ \begin{pmatrix} \frac{s_l}{2} \\ s_h \cdot (1 - 2t) \end{pmatrix} & \text{for } t_1 + t_2 \leq t < 0.5 \end{cases}$$

The rectangular locus has, like the semi-circle locus, the advantage of a plain perpendicular movement of the feet when the feet leave the ground and start to touch it, but the jerky movement in the corners of the trajectory resulted in unwanted shaking of the whole robot body.

3) Semi-ellipse: The semi-elliptical movement (see Fig. 4(a)) can be described by two parameters, step length s_l and step height s_h. For $0 \leq t < 0.5$ it is defined by:

$$f(\vec{t}) = \begin{pmatrix} x(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} -\cos(2\pi t) \cdot \frac{s_l}{2} \\ \sin(2\pi t) \cdot s_h \end{pmatrix}$$

This trajectory has the same properties as the semi-circular one but the shortcomings of the coupling of step height and step length is not existing here.



(a) The semi-elliptical trajectory. (b) The sinusoidal trajectory.

Fig. 4. Trajectories for the feet with step length 10 and step height 5 in the X-Z plane.

4) Sinusoidal: For $0 \leq t < 0.5$ the sine trajectory (see Fig. 4(b)) is defined by:

$$\begin{pmatrix} x(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} \frac{s_l}{2}\left(2 \cdot (2t - 1) - \frac{\sin(\pi(1 + 2 \cdot (2t - 1)))}{\pi}\right) \\ \sin(2\pi t) \cdot \frac{s_l}{2} \end{pmatrix}.$$

The trajectory corresponds to the sine function in the range of values $0 \leq \pi$ in the x-$f(x)$-plane. The shape is very similar to the semi-ellipse but it has the disadvantage that in the moment of lowering a foot on the ground, the movement of the foot has still a component converse to the subsequent movement on the ground. Tests on a carpet have shown that this caused our robot to stumble and fall over more often than with the semi-elliptical movement.

B. Trajectories for the arms

Like the human walk shows, the arm movement synchronized with the leg movement supports the walk. The arms can help to balance or could even give a momentum into the walk direction, e.g. joggers or runners make extensive use of the arms. For the arm movement we dispense with inverse kinematic calculations to follow a defined trajectory. We only consider the upper shoulder joint and directly control the joint of the shoulder movement. As shown in Fig. 2(b) the arms can be moved in the X-Z and the Y-Z plane. The arm movement is firmly coupled with the leg movement, using the same timing parameter t. The time to swing an arm forth and back is equal to the time of a whole step.

For controlling the arm movement we tried a linear movement, swinging the arm forth and back and also a sinusoidal movement. As expected, the later one caused a much smoother movement, since it slowly accelerates and decelerates the arms at the turning points. Another advantage is that the arms stay longer near the turning points where they can contribute as balancing weights.

The angle ω_{arm_x} of the arm movement in the X-Z plane over time is defined as

$$\omega_{arm_x}(t) = \frac{\pi}{2} x_{arm_{max}} \cdot \sin\left(2\pi\left(t + \varphi_{arm_x}\right)\right)$$

with the maximum amplitude $x_{arm_{max}}$ and a phase shift of φ_{arm_x} relative to the leg movement. Like for human walking, we defined the phase shift to be 0.25 relative to the leg movement, i.e. the left arm swings forward while the right

leg moves backward and vice versa. The movement of both arms of course is shifted in phase by 0.5.

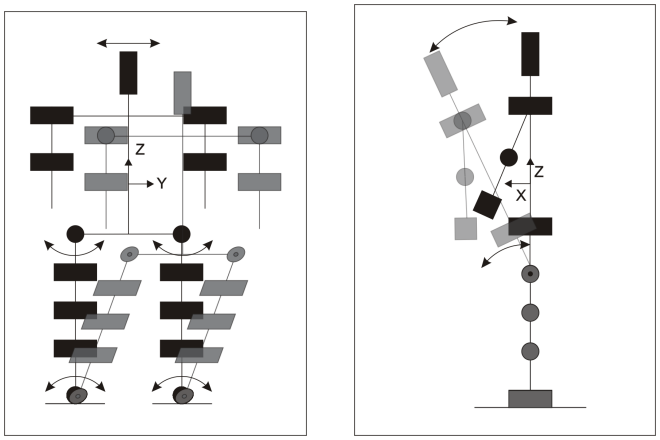
Since the movement in the Y-Z plane is restricted by the robot body, the arm movement ω_{arm_y} is defined as

$$\omega_{arm_y}(t) = \frac{\pi}{4} y_{arm_{max}} \cdot (\sin(2\pi(t + \varphi_{arm_y})) + 1)$$

with a fixed phase relative to the leg movement of $\varphi_{arm_y} = 0.5$.

C. Moving the body

To countervail the shifting of the center of mass of the robot while walking, we allow additionally to the arm movement a movement of the upper part of the body. Fig. 5(a) and 5(b) show the movement: the robot can "swing" sideways into the Y direction and forward into the X direction.



(a) The body movement in the Y-Z plane.

(b) The body movement in the X-Z plane.

Fig. 5. Schematic illustration of the two body trajectories. The black color represents the upright standing robot while the gray color shows the robot moved into the according direction.

1) Swinging sideways: The sideways movement of the robot is justified due to the fact that a shifting of the center of mass of the robot into the direction of the foot which is still on the ground gives a much better stability of the robot. With a foot size big enough even a statically stable walk can be achieved by this movement. This sideways shifting is achieved by only using the leg joints 1 and 5 (see Fig. 1(b)). Like the arm movement, this movement is also sinusoidal and is also coupled with the timing parameter t. The sideways movement is defined as:

$$\omega_{body_y}(t) = \frac{\pi}{2} y_{body_{max}} \cdot \sin(2\pi(t + \varphi_{body_y}))$$

For a phase shift of $\varphi_{body_y} = 0$ relative to the leg movement, the body moves in a way that it leans most to the right side while the left leg is in the air and just reached the center of the air phase movement. The upper body is in the center position when both feet touch the ground. The calculated value $\omega_{body_y}(t)$ is simply added to the previously (according to the foot trajectory) calculated value of the leg joints 1. To guarantee a parallel movement of the feet to the ground, it has to be subtracted from the leg joint 5.

2) Swinging forward: For completeness also a forward swinging has been implemented. Birds for example swing the body slightly forward for each step. The frequency of this movement has to be double of the rest of the movements, since it has to be done for each step. It is described by:

$$\omega_{body_x}(t) = \frac{\pi}{4} x_{body_{max}} \cdot (1 + \sin(4\pi(t + \varphi_{body_x})))$$

Like the sideways swinging, the angle $\omega_{body_x}(t)$ is added to the leg joint 2. A phase shift of $\varphi_{body_x} = 0$ leads to the described behavior.

D. Controlling the walking direction

By now only straight forward walking has been considered. But like humans, the robot should be able to walk sideways and to rotate. Best would be to be able to combine a translative movement vector (in the X-Y plane) with a rotational component, i.e. an omnidirectional movement to be able to walk on a circle for example.

As previously mentioned, the Kondo KHR-1 does not have a rotational joint in the legs. For robots with rotational joints along the Z axis the rotational movement can be done by rotating the foot which remains on the ground while the other one is in the air.

A translational movement into a direction α in the X-Y plane however can be achieved easily by simply rotating the defined trajectory of the feet by the angle α: the calculated x coordinate is multiplied by the cosine of α and the y coordinate by the sine of α. This positions the feet on a circle around the robot center and leads to a problem: for sideways walking, i.e. $\alpha = \pm 90°$ the legs would continuously collide under the robots' body since there is not much space between the legs in the y direction. To overcome this problem, the legs get spread for sideways walks proportionally to the sine of α:

$$\omega_{offset_y} = \frac{s_l}{2} |\sin(\alpha)|$$

with the step length s_l. This offset ω_{offset_y} is added to the y coordinate before the inverse kinematic calculations. The right foot is shifted by ω_{offset_y} to the side, the left one by $-\omega_{offset_y}$.

In [4] it has been shown that different walking trajectories for the different walking directions can improve the walk of four legged robots. The same approach has been used here too, but is not explained in detail.

IV. Optimization Techniques

In this section the optimization techniques used to let the robot learn to walk are explained. The focus lies on Evolutionary Algorithms which are well-tried for problems with an unknown structure. The choice of the optimization techniques is, a problem due to the huge amount of different approaches. However by analyzing the problem to optimize, several techniques can be excluded.

We want to find a parameter set for our walking model which lets the robot walk satisfyingly. The problem itself cannot be specified mathematically which means that the only

way to get the "function value" (from now on called *fitness*) for a certain parameter set is to try them out. This means we want to optimize a "black box" system: the black box outputs a value for a couple of input values. The process inside the black box is not known and is for the optimization technique not of interest. Furthermore the measurement affected by noise, i.e. measuring the fitness of the same parameter set n times can generally lead to n different values. Finally the role of the different parameters is not quite clear, e.g. the movement of the arms or the upper body in the model might be useless or counterproductive.

These considerations already exclude analytical methods because of the lack of a mathematical description. Analytical methods make use of certain properties of the objective function at the position of the optimum, this can for example be the slope of the graph at this position. For the slope calculation typically derivations are used which are for our problem not existent.

Another class of optimization techniques are the iterative methods: the principle is a stepwise approach toward the optimum. Unlike the analytical methods, these methods do typically not find the exact optimum. Similarly to the analytical methods, some iterative techniques use beside the value of the objective function also information about the first or second derivative of the current point. These are especially all gradient strategies or the methods of conjugate gradients [5] which make also use of the second derivative. Due to the lack of a mathematical description these methods are also not directly usable.

Still a lot of methods exist which conform to a black box optimization. The "Hill Climber" approaches for example follow during the search for the optimum a path of steadily growing fitness values (for a maximization problem). One of the biggest shortcomings of these techniques is the problem of getting stuck in local minima, i.e. points in the search space whose direct neighbors all have a worse fitness values. Additionally, these methods are typically very vulnerable to measurement noise. A point with wrongly measured high fitness for example can only be left if another point with a higher fitness can be found. An overview of hill climbing methods is given by Schwefel [6].

Another very widespread class of optimization algorithms are heuristics which are inspired by nature. The eminent category are the Evolutionary Algorithms [7]. They can solve black box problems and can in certain variants deal with measurement noise of the fitness. These algorithms operate on populations of individuals and are based on the paradigm *survival of the fittest*. A parent population is creating an offspring population by making use of the operators *replication*, *mutation* and *recombination*. Due to the *mutation* and *recombination* operators the offspring individuals "differ" from their parents, e.g. they have different properties. The *selection* then decides which individuals will form the new parent generation, all other individuals will die out. When the *selection* operation is based on the mentioned paradigm *survival of the fittest*, in the course of the evolution process the properties of the parent generation will be optimized with respect to the *fitness* criteria of the selection operator. A modification of the evolutionary algorithms brought up particle swarm and simulated annealing algorithms.

A. Genetic Algorithms

Genetic Algorithms have been developed by Holland [8] and originally operate in $\mathbb{B}^n$, the n-dimensional search space of binary numbers, i.e. the recombination and mutation operations on the individuals are manipulations of bits. Later versions of Genetic Algorithms however can also operate in $\mathbb{R}^n$. The fitness of an individual is typically in $\mathbb{R}$. The algorithm uses a population which has a size bigger than one to make the recombination of individuals for the generation of new individuals possible. The "encoding of the chromosomes" maps the search space of the given optimization problem to the search space $\mathbb{B}^n$. To find this encoding is the most challenging problem when using Genetic Algorithms.

B. Evolution Strategies

Evolution Strategies (ES) have been developed in the beginning of the sixties by Schwefel [9] and Rechenberg [10] to optimize technical problems. The most simple ES uses one parent and generates by mutation an offspring. The mutation is realized by adding Gaussian distributed noise to the parameters of the parent. In case of a better fitness of the offspring, it becomes the parent, otherwise the parent remains and a new offspring is generated by mutation. This loop is done until a termination criteria stops the evolution. This evolution strategy is called $(1+1)$ strategy. The first number depicts the number of parents, the second number the amount of offsprings. The $+$ specifies that is it an elite strategy what means that only fitter offsprings can replace parents. Ingo Rechenberg analyzed the $(1+1)$ strategy and developed the $1/5$ rule which controls the mutation strength. He proved for two models that the mutation strength is optimal if 1 out of 5 generated offsprings is fitter than the previous parent [10]. In case of a bigger success rate the mutation strength is too small and has to be increased, otherwise it is too big and has to be decreased.

Schwefel developed in his dissertation [9] Evolution Strategies with a parent population of a size of $\mu \geq 1$ which generate $\lambda \geq 1$ offsprings by recombination and mutation. Additionally he improved the control of the mutation strength: each parameter which has to be optimized (object parameter) obtains its own mutation strength (strategy parameter). These strategy parameters are also included in each individual and are selected and inherited together with the individual's assignments for the object parameters. Thus, they have a higher probability of survival when they "encode" object parameter variations that produce fitter object parameters. This adaption of the mutation strength is called self-adaption.

An offspring is created by means of recombination of the parents. Both the object parameters and the strategy parameters of the parents are recombined. Principally one can differentiate between discrete and intermediate recombination. Let $\mathbf{p} = (p_1, \ldots, p_k)$ be an individual of the parent population which

includes the object and endogenous strategy parameters, then the single components n_i of the offspring individual $\mathbf{n} = (n_1, \ldots, n_k)$ get randomly chosen out of the p_i components of randomly chosen parents for the discrete recombination. The intermediate recombination on the other hand creates the components n_i out of the mean value of all p_i of the parents.

The selection operator elects the parents for the next generation. In case of the $+$-strategy it elects the μ best individuals out of the offsprings and the parents while in the case of the ,-strategy only the best μ individuals out of the offsprings are chosen to be the parents of the next generation. For the $+$ strategy the lifetime of an individual can be limited to κ generations. This creates a mixture of the $+$ and the , selection.

C. Simulated Annealing

For the optimization process, the simulated annealing mimics the effects of cooling down metals. The principle is similar to the $(1+1)$ Evolution Strategy. The search points are changed with a mutation-like operation. In case of a better fitness value of an offspring individual, it replaces its parent. In case of a worse fitness, it can even replace its parent with a certain probability; this probability gets lower during the optimization process and is called "annealing schedule". It simulates the increasing coherence of molecules in metals during the cool down process. Thus, the probability of leaving a local minimum by accepting a worse individual decreases during the optimization.

D. Particle Swarm Algorithms

Particle Swarm Algorithms mimic the behavior of fish or bird swarms. Each individual of the swarm (also called particle) acts fully autonomously, i.e. there is no central control of the swarm movement. The particles move into a direction which is on the one hand guided by own preferences and on the other hand also look about the movement of the swarm. This creates a combination of individuality and coherence.

For optimization a swarm out of virtual particles is used, comparable to the population of an Evolutionary Algorithm. The particles move in the search space and carry the function value (fitness) of their current position. Based on the best found fitness value in the whole swarm and the best found fitness value of the particle itself, a new position can be calculated with the aid of a random component. Each particle also orients itself toward the other particles in the swarm, it "flies" into the direction of more promising values with a higher fitness.

The difficulty in using the particle swarm algorithms is in finding appropriate values for constants which affect the behavior of the algorithm.These constants are very problem specific and have a big effect on the performance of the algorithms. More details and variants of particle swarm algorithms are described by Kennedy et al. [11].

V. Learning to Walk

In this section we first present results of several walk evolution processes in the simulator, each with a different configuration of the Evolution Strategy. Since several tests have shown that the semi-elliptical foot trajectory gives best results, we focus our work only on this trajectory. Overall 11 parameters have been optimized. After having found the best performing strategy, we apply this strategy to real walk evolution on a physical robot.

A. Selection of an optimization technique

The use of evolution strategies for this problem is quite straightforward. Especially the measurement noise and the long time to evaluate the fitness led to this choice, additionally very good experiences regarding the progress rate and the found optimum have been made in the RoboCup Four-legged-league [4]. For us it was important that the strategy should adapt automatically to the problem. The class of self-adapting Evolution Strategies was very auspicious here.

B. Fitness evaluation

The major goal of the required walking patterns is the maximum speed. In robot soccer the team with the faster robots has an invaluable advantage: the faster robots reach the ball first and can keep it under control while the opponent is automatically in a defensive position. Other side requirements for the walk are stability, to be robust against pushing of other robots, and smoothness, to prevent the camera from making blurry images.

As a first approach we decided to use only the maximum speed of the walk resulting out of the parameter assignments of an individual as its fitness. To determine the fitness we let the robot run for a certain distance and measured the time it needed to travel from the starting to the end point. In case of falling, the individual was assigned the fitness 0 which in our implementation is declared to be the worst achievable value, the same is true for a walk which did not reach the goal within a maximum period of time.

C. Comparison of Evolution Strategies

To find an appropriate population size and selection operator, we ran several walk evolution processes with different configurations of Evolution Strategies. In the following we present four results.

1) The $(1+1)$ Strategy: At first we ran the walk evolution with a $(1+1)$ strategy with adaption of the mutation strength according to the $1/5$-th rule. This strategy immediately revealed the problems of getting stuck in a local minimum. As soon as the robot found one walk which was slightly better than the starting walk (this could have also happened due to measurement errors), the mutation strength went down because in the next generations too many lethal individuals have been created, the robot fell down or the offsprings simply had a lower fitness. Since all these offsprings have been classified to have a worse fitness, the algorithm "thought" that a maximum must be reached and decreased the mutation strength which led to the impossibility to leave the locally found optimum. A real optimization with this technique was not possible.

To overcome these disappointing results, we implemented a "reset" of the mutation strength if it became too low and

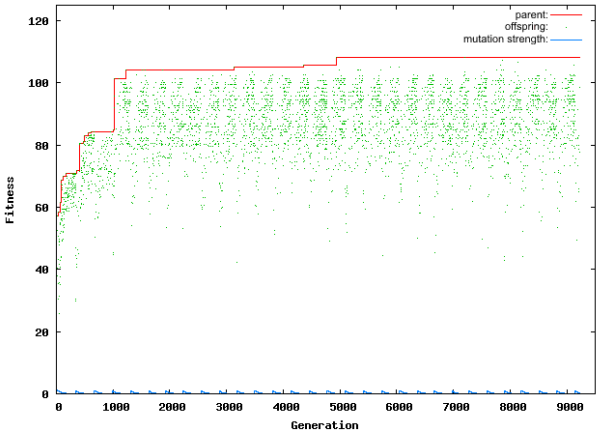


Fig. 6. The results of the modified $(1+1)$ ES in the simulator.

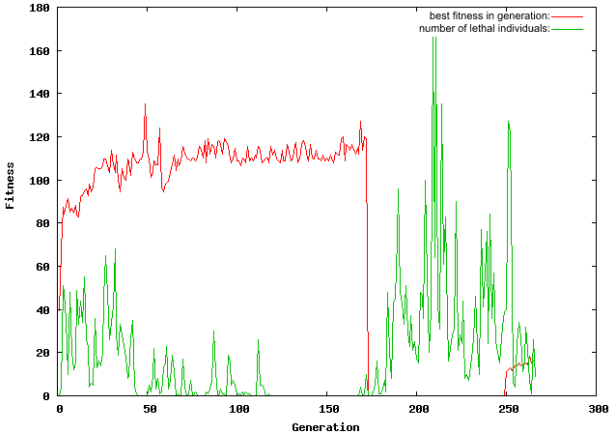


Fig. 7. The results of the $(1, 30)$ ES in the simulator.

reached a threshold. This approach is comparable to a multi start approach, which simply means that the evolution process restarts at the so far found local optimum. Fig. 6 shows the course of the fitness during the evolution process. The maximum speed of about 110 mm/s has been reproducibly found with this strategy.

2) The $(1, 30)$ *Strategy:* To overcome the problem of getting stuck in a local optimum we then used a non elite strategy which means that the parents of the next generation are only chosen out of the offsprings. The used strategy was a $(1, 30)$ strategy, i.e. one parent generates by mutation 30 offsprings and the best of the 30 offsprings becomes the next parent regardless if its fitness is better than the parents' fitness. Fig. 7 shows the graph of the fitness. The big fitness discontinuity around generation 170 immediately attracts attention. This phenomenon appeared repeatably. It can be explained by the fact that all offsprings which have been generated could not walk and so an "invalid" individual became parent and obviously only generated invalid offspings again, the fact that the number of lethal generated offspings increases at this point drastically supports this explanation. The best found fitness with this strategy was 120 mm/s.

3) The $(5, 30)$ *Strategy:* The consequential conclusion was to allow more parents to open the possibility to not only search around one point. This led to the $(5, 30)$ strategy. In the fitness graph in Fig. 8 for this strategy, the upward trend is clearly visible. Now it appears that the population size $\mu > 1$ brings the advantage that none of all simulations with this strategy created the discontinuity like the $(1, 30)$ strategy, i.e. the diversity was big enough that always individuals have been used which could walk without falling. The fact that the bigger number of parents led to this is supported by the fact that the other parameters of the strategy did not change from the $(1, 30)$ strategy. The graph also visualizes that the mean fitness values are clearly above the found fitness values of the $(1, 30)$ strategy.

4) The $(5 + 30)$ *Strategy with* $\kappa = 3$*:* Maybe the reader could think at this point that it might be smarter to use an

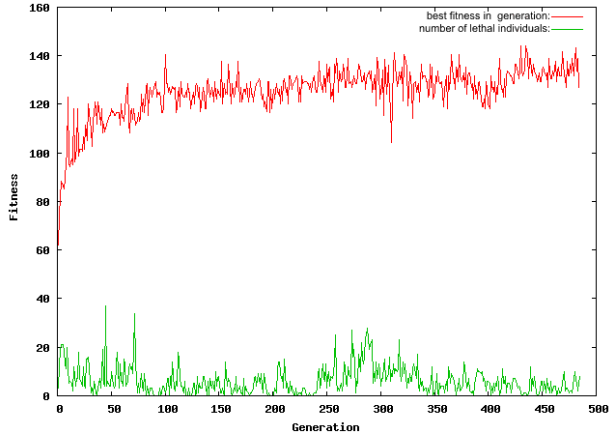


Fig. 8. The results of the $(5, 30)$ ES in the simulator.

elite strategy to guarantee a steadily growing fitness, but the principle of the self-adaption usually does not work well with the $+$ strategies [6] and the "forgetting" of found values makes the , strategy less susceptible toward the measurement noise. However we tried a mixture of the $+$ and the , strategy to be either able to deal with the noise while trying to stabilize the found values. We created the $(5+30)$ strategy with $\kappa = 3$, i.e. the individuals for the next generation are chosen out of the best offsprings **and** parents, but the parents' lifetime is limited to 3 generations. Fig. 9 shows the course of the fitness of this strategy. The step-like shape shows that obviously a parent individual often survived the maximum three generations before it died out. Furthermore several fitness values above 200 mm/s are existent, rechecks of these parameters have shown that these walks were not faster than 150 mm/s, so they occurred due to measurement noise. This also shows that the value of κ should not be chosen bigger because in case of measurement errors this would disturb the evolution process.

Despite the longer existence of wrongly measured individuals, the $(5+30)$ with $\kappa = 3$ created in all test the best results.

The found walks of all evolutions with this strategy reached more than 150 mm/s in the simulator.

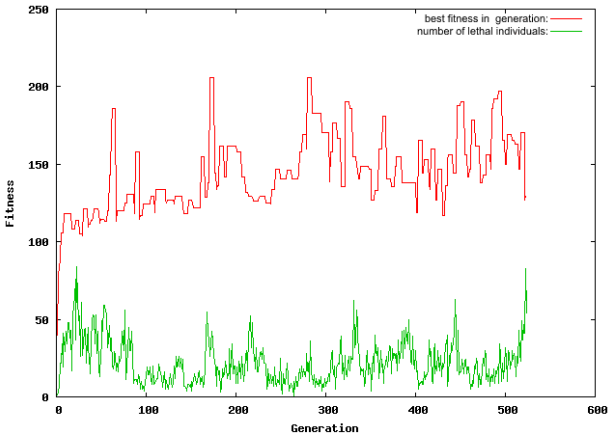


Fig. 9. The results of the $(5 + 30)$ ES with $\kappa = 3$ in the simulator.

D. Experiments on the real robot

The $(5 + 30)$ strategy with $\kappa = 3$ performed best in the simulation, thus we used this strategy to evolve a forward walk with a real robot. Fig. 10 shows the installation which has been created for this. The robot walks inside a rack which allows it to do the evolution fully autonomously. In case of falling it got up and walked back to the starting position. The speed was measured using light barriers.

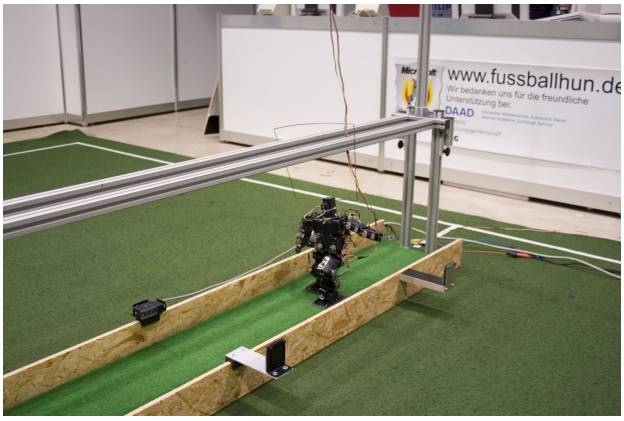


Fig. 10. The "playpen" for the autonomous walk evolution.

The fastest walk was found after 40 generations and reached 220 mm/s, it should be mentioned that for this walk the robot did neither carry a battery pack nor the on board computer. Fig. 11 shows the fitness trend of the evolution.

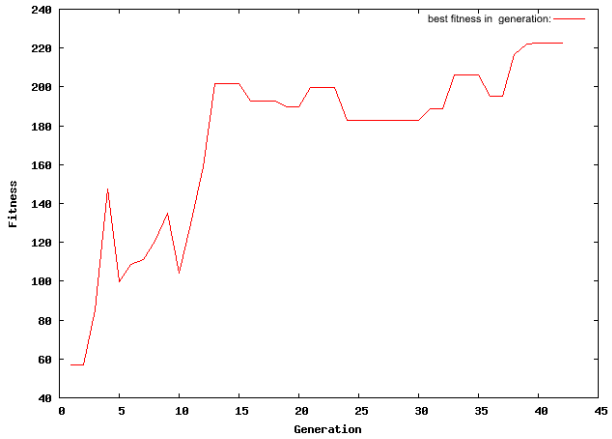


Fig. 11. The fitness trend of the walk evolution on the real robot.

VI. Conclusion

In this paper we suggested a flexible walking model for a two legged robot. Afterward several optimization techniques have been presented and discussed in respect of usability for the parameter optimization of this problem. The class of Evolution Strategies has been chosen and several variants tested in the robot simulator. The best performing strategy finally was used for walk evolution on a real robot and resulted in speeds faster than 220 mm/s.

References

[1] T. Laue, K. Spiess, and T. Röfer, "SimRobot - A General Physical Robot Simulator and Its Application in RoboCup," in *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, Springer, 2006.

[2] M. Meredith and S. Maddock, "Real-Time Inverse Kinematics: The Return of the Jacobian," tech. rep., Department of Computer Science, University of Sheffield, 2004.

[3] S. Behnke, "Human-Like Walking using Toes Joint and Straight Stance Leg," in *Proceeding of 3rd International Symposium on Adaptive Motion in Animals and Machines*, September 2005.

[4] M. Hebbel, W. Nistico, and D. Fisseler, "Learning in a High Dimensional Space: Fast Omnidirectional Quadrupedal Locomotion," in *RoboCup 2006: Robot Soccer World Cup X*, Lecture Notes in Artificial Intelligence, Springer, 2007. to appear.

[5] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[6] H.-P. Schwefel, *Evolution and Optimum Seeking*. Wiley Interscience, 1995.

[7] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[8] J. Holland, *Adaption in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press, 1975.

[9] H.-P. Schwefel, *Evolutionsstrategie und numerische Optimierung*. Dr.-ing. dissertation, Technische Universität Berlin, Fachbereich Verfahrenstechnik, 1975.

[10] I. Rechenberg, *Evolutionsstrategie*. Friedrich Fromm Verlag, Stuttgardt, 1973.

[11] J. Kennedy, R. Eberhart, and Y. Shi., *Swarm intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.