Available online at www.sciencedirect.com
ScienceDirect
Robotics and Autonomous Systems 56 (2008) 199–212
Robotics and Autonomous Systems
www.elsevier.com/locate/robot

# Free gait generation with reinforcement learning for a six-legged robot☆

Mustafa Suphi Erden*, Kemal Leblebicioğlu

*Department of Electrical and Electronics Engineering, Computer Vision and Intelligent Systems Research Laboratory, Middle East Technical University, 06531 Ankara, Turkey*

Received 15 June 2006; received in revised form 26 July 2007; accepted 6 August 2007
Available online 14 August 2007

**Abstract**

In this paper the problem of free gait generation and adaptability with reinforcement learning are addressed for a six-legged robot. Using the developed free gait generation algorithm the robot maintains to generate stable gaits according to the commanded velocity. The reinforcement learning scheme incorporated into the free gait generation makes the robot choose more stable states and develop a continuous walking pattern with a larger average stability margin. While walking in normal conditions with no external effects causing unstability, the robot is guaranteed to have stable walk, and the reinforcement learning only improves the stability. The adaptability of the learning scheme is tested also for the abnormal case of deficiency in one of the rear-legs. The robot gets a negative reinforcement when it falls, and a positive reinforcement when a stable transition is achieved. In this way the robot learns to achieve a continuous pattern of stable walk with five legs. The developed free gait generation with reinforcement learning is applied in real-time on the actual robot both for normal walking with different speeds and learning of five-legged walking in the abnormal case.


*Keywords:* Locomotion; Walking; Six-legged robot; Reinforcement learning; Free gait

## 1. Introduction

### 1.1. Motivation and content

Due to its superiority over wheeled and tracked systems on loose-rough-uneven terrains, legged locomotion has attracted attention of robotics researchers. The advantages of legged locomotion are mostly due to the fact that it makes use of isolated footholds [8,9]. Considering the manufacturing and control of walking, the inspirations derived from the biological systems, namely legged animals from cockroaches to camels, are most important for multi-legged robot researchers [4]. Studies to incorporate intelligent systems into robotic applications result in promising developments for learning in control of robot manipulators [6]. When legged robots are equipped with intelligent self-learning systems an autonomous system which can adapt itself to different circumstances emerges.

Mahajan et al. [21] define "gait" as follows: "The gait of an articulated living creature, or a walking machine, is the corporate motion of the legs, which can be defined as the time and location of the placing and lifting of each foot, coordinated with the motion of the body, in order to move the body from one place to another." In the literature there exist two contesting views about how gait control is achieved in the nervous system of animals, and what is the best to be applied in multi-legged robots. The "reflex model" composes of local controllers in the legs based on the sensory-motor-feedback between the local agents. The well-known example for reflex model controlled walking is the one developed by Cruse et al. [2], where the legs interact with each other via some mechanisms. The "Central Pattern Generator (CPG) model", on the other hand, is based on a feed forward central controller which generates rhythmic motions of the legs without the need of sensory feedback. The various applications of CPGs utilizing oscillatory neural networks (cf. [14]), are based on the Pearson model of insect locomotion [7].

☆ This research is supported by the research fund of Middle East Technical University as a scientific research project: BAP-2002-03-01-06.
* Corresponding author. Tel.: +90 3122104558; fax: +90 3122101261.
*E-mail addresses:* m.s.erden@tudelft.nl (M.S. Erden), kleb@metu.edu.tr (K. Leblebicioğlu).

0921-8890/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2007.08.001

Donner [3] and Klaassen et al. [18] provide brief comparisons of CPG and reflex models. The core of the discussion is how to manage adaptability and smooth passage between gait patterns during continuous walking. The CPG models generate continuous patterns, mostly by oscillators controlling the leg movements, without using any feedback. It is easy to manage smooth passages between gait patterns by smoothly changing the oscillation constants. However, the CPG models are not as adaptive as the reflex models considering the changing environment and system conditions. For example, if one of the legs is out of order the CPG model collapses. The reflex model, on the other hand, can manage such situations; because, it gets feedback from the legs and determines the commands accordingly. The problem with the reflex model, on the other hand, is the lack of a durable walking pattern, and the necessity of processing the feedback from all legs for every step. Donner [3] and Klaassen et al. [18] both conclude, CPG and reflex models should be conciliated in order to achieve the best performance. The approach presented in this paper is an attempt to conciliate CPG and reflex models using the ideas of central generation of free gaits and reinforcement learning for state transitions, respectively. Free gait generation and reinforcement learning are performed on the developed discrete model of stepping. The resulting Free Gait Generation with Reinforcement Learning (FGGRL) is applied both in simulation and to an actual robot. The FGGRL generates gaits with large stability margin while walking with different speeds and a stable gait in case of a rear-leg deficiency.

In the following, a review of gait adaptation and reinforcement learning literature that inspired this work is given. In Section 2, the discrete model of stepping and the states of the walking scheme are introduced. In Section 3, the free gait generation algorithm making use of this model is given. In Section 4, reinforcement learning is incorporated with the free gait generation algorithm for learning the state transitions that lead to more stable gaits. The developed reinforcement scheme with the free gait generation constitute the FGGRL. In Section 5, the FGGRL is modified to learn walking when one of the rear-legs is out of order. In Sections 6 and 7, simulation and experimental results, respectively, obtained by application of the FGGRL are presented. Section 8 concludes the paper.

### *1.2. Related gait adaptation literature*

In the literature reinforcement learning is widely applied to walking machines in order to obtain improvement in various robot behaviors. Among those, Ilg and Berns [11] and Ilg et al. [12] make use of reinforcement learning in order to develop suitable protraction and retraction movements with off-line learning. Kirchner [17] applies real-time reinforcement learning to a six-legged robot in order to develop, first, the elementary movements for a single leg, then, the tripod gait for the robot. Huber and Grupen [10] apply real-time reinforcement learning to a four-legged robot in order to develop a turning gait, using a hybrid discrete event dynamic system structure.

There are attempts of free gait generation, in which off-line learning or programming in advance is applied. Pratihar et al. [25] develop a genetic-fuzzy approach in order to achieve optimal path and gait generation simultaneously. In their structure the fuzzy logic controllers change the stroke lengths to handle the turning of the robot and to avoid ditches while walking. Pal et al. [22] perform free gait generation by using a graph search among the predefined states delineating the supporting and returning legs. Their search maximizes the number of leg transfers in each step. Porta and Celaya [24] propose a reactive free gait generation, in which the gait is generated and modified as a result of the interaction of the robot with the environment. If one of the legs does not find a proper position to place the tip on the front side of its stroke, it places it somewhere in the middle or rear of the stroke. The gait of the machine is modified immediately depending on the current stroke positions of the legs. In their free gait generation the authors make use of the stability rule, which is adopted also in this paper as “the rule of neighborhood”. Therefore, their algorithm, as the one in this paper, guarantees stability in every instant throughout the walk. All of the three mentioned works are simulation-based. All the three algorithms are intended for real-time modification of gaits according to the interaction of the robot with the environment. However, this modification is limited with the capabilities of the off-line developed structures. Neither the rule structure of the first, nor the recorded possible state transitions of the second, nor the rules used for free gait generation in the third are intended for continuous improvement. The superiority of reinforcement learning compared to those is that it provides a learning structure for continuous development and adaptation to unexpected situations. Porta and Celaya [23], Kimura et al. [16], Svinin et al. [27], and Karalarli et al. [15] use reinforcement learning specifically to obtain efficient gait generations.

Frequently, the applications of reinforcement learning to gait generation aim speed maximization. Kimura et al. [16], aim learning of sequential angular positions of the actuators of a four-legged robot in order to maximize the speed. The reinforcement learning algorithm is realized on an actual robot system, in which two wheels attached to the rear of the robot measure the advancement and provide the reinforcement signal. Karalarli et al. [15] use reinforcement learning to maximize the speed and stability margin using a gait generation frame with fuzzy controllers. Reinforcement learning is used to tune the centers and spreads of the membership functions that determine the transitions of legs between stance and swing. Porta and Celaya [23] use reinforcement learning with their free gait generation structure, explained extensively in their other work [24], for speed maximization. The results show that the scheme with reinforcement learning is more efficient than their hand-programmed reactive free gait generation. The superiority of their learning scheme is that it is possible to be applied on a real robot; because, as mentioned before, their free gait generation always guarantees stability and avoids the robot from falling [24]. This facility exists also in the reinforcement learning scheme developed in this paper. In the application here the free gait generation already adopts the commanded speed, therefore there is no need of speed optimization. Rather, the stability

of the gait is optimized in our application. Besides that, the application here accepts an external reward signaling if the robot falls or not due to some unexpected reasons. For example, a rear-leg might be deficient and the robot does not know it. In that case, the reinforcement learning makes the robot learn the transitions that do not result in falling.

Maes and Brooks [20] apply a behavior-based learning algorithm for walking of the two-joint six-legged robot, named Genghis. In their scheme learning corresponds to adapting the preconditions of some given behaviors in order to control their activation to result in straightforward walking of the robot. Similar to the Robot-EA308 used in this paper, Genghis is also equipped with touch sensors on the bottom. Additionally a trailing wheel measures the forward advancement. Any forward movement measured by the trailing wheel signals a positive feedback, while any touch of the sensors at the bottom signals a negative feedback. The robot Genghis learns the tripod gait in 10 min with a non-intelligent search of the conditions, and in 1 min 45 s with an intelligent search. The authors do not provide a detailed result of the learning process. They also do not report application of their scheme for adaptation to any other condition than the straight walk with all legs in order. They mention adaptation to different speeds and to the case of deficiency in one of the legs as future work. In comparison to the work of Maes and Brooks [20], the gait generation scheme developed in this paper guarantees straightforward walking of the robot with any given speed when all the legs are in order. The learning scheme adapts a gait pattern to maximize the stability, which is not a concern for Maes and Brooks [20], throughout the walk. Moreover, adapting to the case of deficiency in one of the rear-legs, which is a much more harder task than walking with all legs, is realized in this paper. The learning scheme of Maes and Brooks [20] does not seem to be flexible enough to adapt to unexpected conditions; because, their scheme is based on preplanned behaviors. It is not possible to learn new behaviors (different ways of stepping), which would necessitate in some unforeseen conditions. The algorithm in this paper, on the other hand, allows emergence of new ways of stepping by developing a discrete model. In this way it is possible to learn a gait even for the condition of a rear-leg deficiency.

Svinin et al. [27] use reinforcement learning in order to learn the sequential action patterns that result in a stable straight walk. In this application, also, falling of the robot results in a negative reinforcement, and the algorithm learns to avoid those. However, the application is purely simulative. It is strongly emphasized that learning procedure may take too much time if it is applied to an actual robot in real-time. In fact, this difficulty is pointed out also by Ilg and Berns [11], Ilg et al. [12], and Kirchner [17], noting that real-time application of reinforcement learning to actual robotic systems is extremely difficult because of the necessity of large number of training cycles. It is recommended to incorporate a priori knowledge or to make use of a biased learning strategy in order to simplify and speed up the learning process. The application of a real-time learning to an actual walking robot in this paper is noteworthy with those of Maes and Brooks [20], Kimura et al. [16], Kirchner [17], and Huber and Grupen [10]. In order to cope with large number of training cycles, this paper applies the reinforcement learning, not starting from a tabula rasa, but with the peculiar free gait generation algorithm that guarantees stability for all states. Such an approach, does not only guarantee stable walk in normal conditions, but also makes learning faster even when stability is not guaranteed when one of the legs is disfunctioning.

The necessity of handling the case of deficiency in one of the legs is addressed by Karalarli et al. [15] and Inagaki [13]. Karalarli et al. [15] content with observation of the behavior of their gait generation algorithm, making use of reinforcement learning, when one of the legs is deficient. Although the algorithm manages to make the simulative robot walk with instants of instability, it does not end up with a stable pattern. Inagaki [13], on the other hand, dwells upon generating a stable periodic gait for five legs. The analysis reveals that, with the settings of the paper, there exists no stable periodic gait with five legs for the general rectangular leg arrangement of hexapod robots; stable periodic gaits for five legs can be achieved only with an assymmetric pentagonal leg arrangement. However, in order to switch to such an arrangement the workspace of the legs should be quite large. The analysis and results of these two papers reveal that static-stable five-legged walking is a hard task. This paper does not claim to have solved the five-legged walking problem. Rather, five-legged walking is used as a case in which adaptability of the robot to an unexpected failure is tested. Moreover, the five-legged walking in this paper is limited to a pseudo-static-stable walk, which will be introduced in the following, with deficiency in one of the rear-legs. The peculiarity of this paper, on the other hand, is that the learning of five-legged walking is realized on an actual robot.

Lastly, it is worth mentioning about the frequency adaptation of compliant mechanism walking robots, in relation to gait adaptation. Compliant mechanism walking robots adopt oscillatory behavior in which the kinetic and potential energies are transformed into each other in different phases of the gait [19]. Weingarten et al. [28] perform an off-line gait adaptation for an energy efficient walk of the robot RHex. “Gait adaptation” for the robot RHex corresponds to arranging the oscillation parameters of a pre-given tripod gait. Brambilla et al. [1] develop a real-time adaptive gait control of a four-legged compliant robot, based on couple oscillators, in order to minimize the energy dissipation. They also adopt a single gait pattern, the “walking gait”, and aim to adapt the frequency of stepping within this gait pattern. The feedback in their system adapts the frequency of the controlling oscillators to the resonant frequency of the system, which is determined by the mass of the robot and the compliant mechanisms on the legs. This adaptation results in less energy dissipation. The results of these three papers stressing the frequency adaptation of compliant robots are not directly applicable to rigid-legged robots. Moreover, since they use a single gait plan and adapt only its variants by changing oscillation frequencies, their adaptation schemes are not flexible to handle unexpected conditions necessitating different gait plans, such as a leg deficiency.

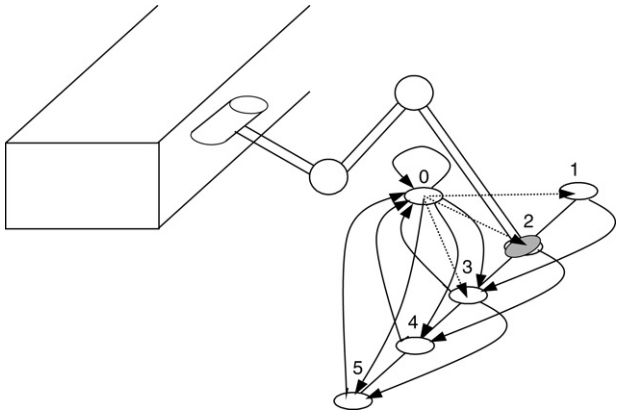


Fig. 1. Discrete positions of stepping, and the possible transition for a velocity of two units per iteration.

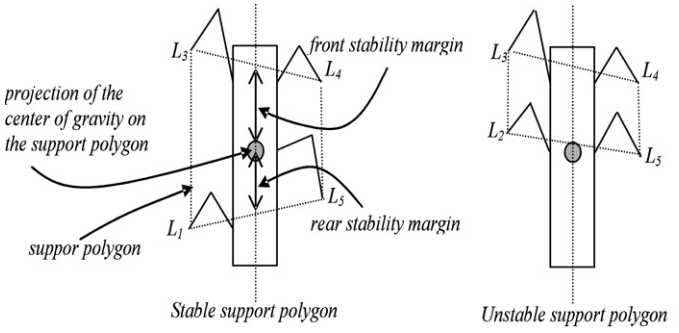


Fig. 2. Examples for stable and unstable support polygons.

## 2. Discrete model of stepping and explanation of states for the learning task

The model of walking is developed based on a discrete model of the possible positions of the tip point of a single leg (Fig. 1). There are five support positions (leg-states) on the ground labeled from 1 to 5. Considering the walking direction as positive, the position 1 corresponds to the maximum possible anterior extreme position, while the position 5 corresponds to the minimum possible posterior extreme position. The position 0 is the single return position at which the tip point stays when the leg is not supporting. In a regular stepping, which is not the case here, there is a single tip point trajectory. Therefore the anterior and posterior extreme positions are fixed. According to such a regular scheme the transitions in Fig. 1 would be from 0 to 1, 1 to 5, and 5 to 0. However, in order to generate a free gait, the stepping must also be free, namely the leg must be able to start supporting on any position on the support line and to pass to the return position from any support position. The discretization in Fig. 1 prepares the frame for such a free stepping.

In free stepping the transitions of the leg-states are conditioned by the speed of the walk that determines how much unit the tip point is to be retracted on the support line. In the model here the tip point has to be in one of the leg-state positions at the end of every iteration. For the simulation and actual robot used here, one unit of distance corresponds to 2 cm. In Fig. 1 the transitions for a walk with two units per iteration

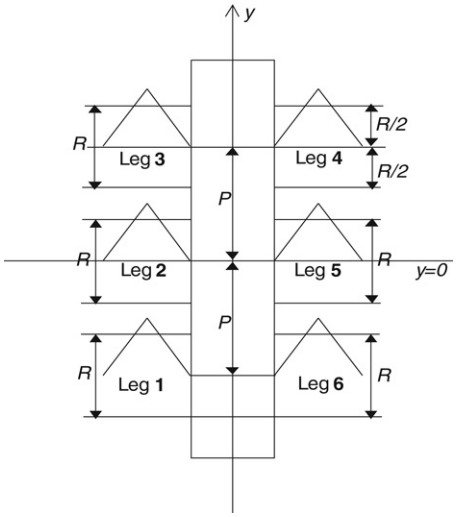


Fig. 3. Figure showing the pitch ($P$) and stroke ($R$) lengths.

are given, which corresponds to *4 cm per iteration* or *4 cm per step*. With this speed the leg has to retract two units if it is supporting. Accordingly, the transitions from 1 to 3, 2 to 4, and 3 to 5 are allowed. The transition from 1 to 2, for example, is not allowed; otherwise the body would not be advanced by two units. The leg cannot pass to any supporting state from the state 0. The first supporting state it visits has to satisfy the condition that it allows the required amount of advancement in the following iteration. Accordingly, for the iteration of two units the first visits from 0 to 4, or from 0 to 5 are not allowed. The first visit from 0 has to be to either 1, 2, or 3.

The transition from state 0 to the first supporting state has nothing to do with advancement of the body; therefore, such a transition will be mentioned as a sub-transition. A sub-transition can be considered to be bound to the succeeding transition, which takes place in between the supporting states, within one iteration: First the sub-transition from state 0 to the anterior supporting state takes place, then the transition from that anterior supporting state to the next supporting state takes place. As a result, in one iteration, considering two units of advancement, transitions from 0 to 3, 4, or 5 are possible. A leg in state 0 can stay in return position, so the transition from 0 to 0 is also possible.

The leg-states of the six legs, together, construct the state of the robot. There are $6^6 = 46\,656$ possible states for the robot. The supporting legs in any state of the robot correspond to a supporting polygon. For the state to be statically stable, the projection of the center of gravity of the robot on the horizontal plane should remain within this supporting polygon (Fig. 2). The stability margin of a support polygon, and that of the corresponding state of the robot, is defined as the minimum of the front and rear stability margins. Theoretically, a state is stable if this stability margin is greater than zero. However, with the actual Robot-EA308 used for this paper, the minimum margin for a stable walk is 2 cm. Therefore, throughout the paper the minimum stability margin will be taken as 2 cm, rather than zero. In Figs. 2 and 3 the convention of numbering the legs throughout the paper is revealed.

According to the 2 cm minimum stability margin assumption, 41 510 of the 46 656 states are statically stable. However, not all of these stable states can be used during an actual walk. This is because some of them are either dead-end states, or states that are impossible to be reached. For example the state 5-5-5-5-5-5 is a dead-end state, since the next state after one iteration has to be 0-0-0-0-0-0, which is unstable. The state 2-2-2-2-2-2 is impossible to be reached, because the robot has to be in the unstable state of 0-0-0-0-0-0 in advance of the iteration, considering the minimum speed of one unit per iteration. The state 1-1-1-1-1-1 is impossible to be reached even with a sub-transition due to the same reason. While the states that are impossible to be reached do not create a problem while learning state transitions, the dead-end states do.

The learning of walking corresponds to learning the possible transitions from any visited stable state to another stable state which is not dead-end. Although the total number of unstable states (5146) is much less than the number of stable states (41 510), many of the unstable states are always reachable while only some of the stable states conditioned by the walking speed are reachable from any given stable state. For example the state 0-0-0-0-0-0 is reachable from any given stable state. Based on the computer calculations, for the state of 3-0-3-0-3-3, with the walking speed of 2 units per iteration, there are $2^6 = 64$ possible transitions (the legs in leg-state 3 can go to the leg-state of either 5 or 0; the legs in leg-state 0 can go to the leg-state of either 0 or 2). Among these only 21 of them are statically stable, considering a minimum of 2 cm stability margin. Among the stable ones only 8 of them are not $-1$ or $-2$ states (to be explained in the following). Therefore, the machine has to learn to choose 8 of the 64 possible transitions from this particular state. This ratio is as low as 2/64 for the state 3-3-3-3-3-3 with the same speed. These are low ratios for real-time learning applications. Therefore, learning of walking in the very general sense is a tedious task [27,11,12,17]. In order to cope with the problem of high dimension of possible transitions, this paper adopts the idea of stable free state generation, which guarantees generation of statically stable next states which are not dead-end.

## 3. Free gait generation (FGG)

The FGG developed here is based on successive stable free state generations (FSG). This FSG is performed by a random choice of a state from the subset of stable non-dead-end states that satisfy the rule of neighborhood and fulfill the advancement of the robot with the commanded velocity. In this generation satisfaction of the rule of neighborhood is critical in order to guarantee the stability.

### 3.1. The rule of neighborhood

The idea of "satisfying the rule of neighborhood" is used also by Song and Waldron [26], and to our knowledge, first applied in the generation of free next states by Porta and Celaya [24]. We have previously given the definition of the rule of neighborhood and analytically proved that any gait satisfying the rule of neighborhood is statically stable, provided the pitch length (the distance between the middle points of stroke lines) is greater than the stroke length [5]. Here for convenience we repeat the definition and give the proof of stability.

**Definition.** Considering all the legs in clockwise direction, namely in the circle of 1_2_3_4_5_6_1, if a leg is lifted, then both the legs in the neighborhood should be supporting. Any support polygon, and the corresponding state of the six-legged robot, satisfying this rule is said to satisfy the *rule of neighborhood*.

**Theorem 1.** *Any support polygon, and the corresponding state of the six-legged robot, satisfying the rule of neighborhood is statically stable provided that the pitch length is greater than the stroke length* ($P > R$). *The minimum possible stability margin of a support polygon satisfying the rule of neighborhood is* $(P - R)/2$.

**Proof.** Due to the rule of neighborhood the front stability margin is determined by the support of one of the following three couples of legs: 2-4, 3-4, 3-5. Due to the geometry in Fig. 3 the minimum front stability margin occurs when the couples 2-4 or 3-5 are supporting. For both cases the minimum margin occurs when the front-leg (3 or 4) and rear-leg (2 or 5) of the couples support at their posterior extreme positions. In this case the line connecting the support positions of the couples crosses the center line at a point where $y = (P - R)/2$. Then the minimum front stability margin is $(P - R)/2$. The same arguments go for the rear stability margin, whose possible minimum can again be calculated as $(P - R)/2$. Then the possible minimum of the stability margin of the polygon, given as the minimum of these two margins, is $(P - R)/2$. Due to the assumption of $(P > R)$, the minimum possible stability margin of a polygon satisfying the rule of neighborhood is always greater than zero, hence the polygon is stable. ■

In the case of the Robot-EA308, the pitch length is 17 cm, and the stroke length is 8 cm. The minimum possible stability margin of a polygon satisfying the rule of neighborhood for the actual robot is then $(17 - 8)/2 = 4.5$ cm, which is greater than the required minimum of 2 cm. Therefore the theorem applies to the Robot-EA308, and in its discrete stepping model any state satisfying the rule of neighborhood (SSRN) is statically stable.

As stated before the dead-end states are a problem for learning of walking. Therefore, the free gait generation should avoid generation of any dead-end states. The following definitions and theorems deal with avoiding dead-end state generation.

**Definition.** Given the speed of walk in units per iteration, a SSRN is designated as a $-1$ *state* if it does not lead to any SSRN; in other words, a dead-end state is a $-1$ state.

**Definition.** Given the speed of walk in units per iteration, a SSRN is designated as a $-2$ *state* if it does not lead to any SSRN which is not $-1$.

**Definition.** Given the speed of walk in units per iteration, a SSRN is designated as a $-3$ *state* if it does not lead to any SSRN which is not $-2$.

**Theorem 2.** *Given the speed of walk in units per iteration as v_unit, where v_unit takes one of the values from* {1, 2, 3, 4}*, a −1 state has to have two successive supporting leg states greater than or equal to* (6-*v_unit*)*.*

**Proof.** A −1 state, $P_{-1}$, has to lead only to states that violate the rule of neighborhood. This means, the state of $P_{-1}$ is such that it cannot avoid two of the legs to have the leg state of 0 in any of the succeeding states. Without violating the generality, let us assume that the pair of legs $L_i$–$L_{i+1}$ have to take the values 0-0 in the succeeding state of $P_0$. Since $P_{-1}$ satisfies the rule of neighborhood we know that at least one of these two legs is supporting in state $P_{-1}$. If the remaining is not supporting, namely if it is in state 0 in $P_{-1}$, then that leg can take any leg state in accordance with the speed in the succeeding state; namely, that leg does not have to be in state 0 in $P_0$. However, this contradicts with the condition that $L_i$–$L_{i+1}$ have to take the values 0-0 in $P_0$. Therefore, both the legs are supporting in $P_{-1}$. In order to generate 0-0 in the succeeding state, these two legs have to have the state of at least (6-*v_unit*) in $P_{-1}$, otherwise they could take another supporting state. For example if one of the legs was (5-*v_unit*), it could take the leg-state 5 in the succeeding state. As a result, any −1 state, $P_{-1}$, has to have two successive leg-states greater than or equal to (6-*v_unit*). ■

**Theorem 3.** *Given the speed of walk in units per iteration as v_unit, where v_unit takes one of the values from* {1, 2, 3, 4}*, a −2 state has to have the pattern of* $s_{i-1}, s_i, s_{i+1}, s_{i+2}$ *in any of its four successive legs, where* (6-2*v_unit*) $\leqslant \{ s_i, s_{i+1} \} <$ (6-*v_unit*) *and* $\{ s_{i-1}, s_{i+2} \} \geqslant$ (6-*v_unit*) *holds.*

**Proof.** A −2 state, $P_{-2}$, has to lead to only −1 states. This means, according to Theorem 2, the state $P_{-2}$ is such that it cannot avoid two of the successive legs to have leg-states greater than or equal to (6-*v_unit*) in the succeeding state. Without violating the generality, let us assume that the pair of legs $L_i$–$L_{i+1}$ have to be in this condition in the succeeding state of $P_{-1}$. Using the same reasoning as in the proof of Theorem 2, these two legs have to be supporting with the states $s_i, s_{i+1}$ in state $P_{-2}$, where (6-2*v_unit*)$\leqslant\{s_i, s_{i+1}\}<$(6-*v_unit*) holds. However, this condition is necessary but not sufficient to have the states greater than or equal to (6-*v_unit*) in the succeeding state. This is because any supporting leg can take a supporting leg-state or the leg-state 0 in the succeeding state. There must be a condition that forces these two legs to stay in support in the succeeding state. This condition is due to the satisfaction of the rule of neighborhood by the succeeding state of $P_{-1}$. The couple of legs $L_i$–$L_{i+1}$ have to be supporting in state $P_{-1}$ only if the two legs in the vicinity of the couple are in leg-state 0. Again following the reasoning in the proof of Theorem 2, if these two legs have to be in leg-state 0 in state $P_{-1}$, they have to be supporting with a state greater than or equal to (6-*v_unit*) in state $P_{-2}$. ■

**Theorem 4.** *There does not exist any −3 state.*

**Proof.** A −3 state is defined as a state which has to lead to a −2 state. By Theorem 3, a −2 state, $P_{-2}$, has to have the pattern of $s_{i-1}, s_i, s_{i+1}, s_{i+2}$ in any of its four successive legs, where (6-2*v_unit*)$\leqslant\{s_i, s_{i+1}\}<$(6-*v_unit*) and $\{s_{i-1}, s_{i+2}\}\geqslant$ (6-*v_unit*) hold. Following the reasoning used in the previous theorems, in order to have to be supporting with these specific states in $P_{-2}$, the legs must be supporting in the previous state. Any four successive supporting legs, whatever their supporting states are, cannot be forced to support in the successive state with the rule of neighborhood; since, any of the two legs in between can pass to the leg-state 0 without violating the rule of neighborhood. Then a SSRN, which is not −2 is achieved. Therefore, it is impossible to have a −3 state which has to be succeeded by a −2 state. ■

**Theorem 5.** *If a free gait generation algorithm avoids any −1 or −2 states in successive state generations, the gait will visit no dead-end states.*

**Proof.** If −1 states are avoided, there will be no visits to any direct dead-end states. Then, −2 states will become the dead-ends of the new situation. If those are also avoided, there will be no visits to the states that have to lead to a −1 state. Once, −2 and −1 states are avoided there will be no dead-end states of the new situation; since, there are no −3 states. As a result the free gait generation will be free of any dead-ends. ■

### 3.2. Free state generation algorithm

Based on the above theorems, the following state generation algorithm is developed. This algorithm guarantees the satisfaction of the rule of neighborhood and avoids any −1 or −2 states. It takes the current state, which is neither −1 nor −2, and the speed of walk in units per iteration as the input, and generates the next state as the output.

*Given:* Current state $P = [s_1, s_2, s_3, s_4, s_5, s_6]$ and the speed of walk $v\_unit = v$.

*Step* 1: Find a succeeding state in which all the legs that can support are supporting:

$$P' = [s'_1, s'_2, s'_3, s'_4, s'_5, s'_6], \quad \text{where}$$

$$s'_i = \begin{cases} \text{mod}\,(6, s_i + v_{\text{unit}}), & \text{if } s_i \neq 0 \\ \text{random}\,\{1, 2, \ldots, 5 - v_{\text{unit}}\}, & \text{if } s_i = 0 \end{cases}.$$

*Step* 2: Construct the state $P''$ by transferring the supporting legs in state $P'$ to the leg-state 0 with a probability of 0.2, 0.3, 0.4, 0.5, and 0.6 for the leg-states 1, 2, 3, 4, and 5, respectively (These probability values are determined so that the leg is more likely to stay in support if its tip point is in a front position, and it is more likely to transfer to the return state if its tip point is in a rear position on the support line.).

*Step* 3: Check if $P''$ satisfies the rule of neighborhood. If it does not satisfy, change the state of the legs with leg-state zero to their values in $P'$ with a probability of 0.5. Repeat this process till a state satisfying the rule of neighborhood is obtained. This state will be $P'''$.

*Step* 4: Check if $P'''$ is a −1 or −2 state. If it is either −1 or −2, goto step 1. Otherwise $P'''$ is a succeeding state of $P$, which is neither −1 nor −2 and satisfies the rule of neighborhood.

*Output:* $P'''$.

The random generation of the next state is guaranteed to stop in finite number of cycles; because, the initial state $P$ is already a non-dead-end state generated by the algorithm and there exists at least one SSRN to be reached by random generation. The successive usage of the free state generation algorithm results in the FGG of this paper. With the FGG the machine can walk with a given speed without any fall. However, the states generated by free gait generation do not have any specificity rather than being a SSRN. Among the possible gaits with a given speed there are some with more static stability, while there are some that hardly satisfy the stability condition. Improvement of stability can be achieved by a learning with which the best state transitions are memorized and utilized. The next section introduces the application of reinforcement learning for this purpose.

## 4. Free gait generation with reinforcement learning (FGGRL)

The reinforcement learning adopted here is based on updating the utilities of state transitions according to a reinforcement signal. A similar application of reinforcement learning of walking to an eight leg robot is performed by Svinin et al. [27].

### 4.1. Updating the utilities of state transitions

After every $(n)$th iteration, the transition from a state $P^{(n)}$ to $P^{(n+1)}$ is assigned a reinforcement signal depending on the stability margin of the next state $P^{(n+1)}$. For this purpose a nominal stability margin value of 7 cm is used, and the reinforcement takes a value in comparison to this nominal margin as in Eq. (1). The subscript $sm$ in the reinforcement term indicates that it is used to maximize the stability margin.

$$r_{sm}^{(n)} = \begin{cases} \dfrac{2}{1+\mathrm{e}^{(7-s_m^{n+1})}} - 1, & \text{if } s_m^{n+1} \geq 7 \\ \dfrac{2}{1+\mathrm{e}^{(7-s_m^{n+1})}} - 1.5, & \text{if } s_m^{n+1} < 7 \\ -1.5, & \text{if } v_{\text{unit}} \neq v_{\text{commanded}} \end{cases}. \quad (1)$$

In Eq. (1), $s_m^{(n+1)}$ stands for the stability margin of the next state $P^{(n+1)}$. The third line of the reinforcement signal might be encountered during speed transitions, which will be explained in the end of this section. The visited states, and the experienced transitions within these visited states are recorded with their corresponding utilities in a lookup table. After every iteration the utilities of memorized transitions, which can take values in the range of [0, 1], are updated. In the following the items of updating are explained.

*Utility initialization:* If the transition is experienced the first time it is assigned the utility of 0.5.

$$u_{P^{(n)} \to P^{(n+1)}} = 0.5. \quad (2)$$

*Evaporation:* After every iteration, all the utilities in the lookup table are discounted with a factor of 0.9. In this way, non-used utilities and states are eliminated in time.

*Utility elimination:* The transitions with utilities less than 0.02 are cleaned from the lookup table.

*Memorized state elimination:* If the sum of the utilities of all experienced transitions from a state are less than 0.1, that state is totally eliminated from the lookup table.

*Reward distribution:* The reward $r_{sm}^{(n)}$ achieved by the transition from $P^{(n)}$ to $P^{(n+1)}$ at iteration $(n)$, is used to update the utilities of the last five transitions that lead to $P^{(n+1)}$. The updating of the utilities is as in Eq. (3). The learning rate denoted by the term $\gamma$ is taken to be 0.8.

$$u_{P^{(i)} \to P^{(i+1)}} := u_{P^{(i)} \to P^{(i+1)}} + \gamma^{(i-n)} r_{sm}^{(n)} (1 - u_{P^{(i)} \to P^{(i+1)}}), \quad i = n-4, n-3, \ldots, n. \quad (3)$$

### 4.2. Selection or generation of transitions

The next state is determined either by selection from the memorized transitions, or by generation of a new transition, depending on the utilities of the current state $P^{(n)}$. If there are $k$ memorized transitions from the state $P^{(n)}$ each with utility $u_{P^{(n)},j}$, one of these transitions is realized with probability $p_u$, and a new transition is created with probability $(1 - p_u)$, where $p_u$ is given as in Eq. (4).

$$p_u = \sum_{j=1}^{k} \frac{u_{P^{(n)},j}}{k}. \quad (4)$$

If one of the memorized transitions is to be realized, a random selection is performed with a distribution proportional to the utilities of the transitions. Otherwise a new transition is created.

A new transition can be created in two ways. First, the FSG algorithm can be used to generate the next state. The generated next state might still coincide with one of the memorized states; then the table is updated with the existing states, otherwise the generated state is considered as a new state. Second, one of the memorized states can directly be used as the next state, although it does not exist within the transitions experienced from the current state $P^{(n)}$. Such a direct utilization makes it easier for the algorithm to catch cycles of transitions, which correspond to a patterned periodic walk. In case of existence of such a transition to a memorized state, the two possibilities of new transition generation are performed with equal probabilities of 0.5. If there are more than one possible transitions to a memorized state, and it is decided to perform a transition to a memorized state, then one of them is selected randomly with a distribution proportional to the sum of the transition utilities of the candidates. If there does not exist any memorized state proper for transition, then the FSG algorithm is directly used.

If the command of speed is changed at an instant, the memorized transitions cannot be used. Therefore, the memorized state transition intended to be utilized has to be checked if it is fitting to the commanded speed. It is easy to achieve this check by comparing the state of supporting legs in the next state candidate and the current state. If the speed requirement is not satisfied the memorized transition selected for realization cannot be used; then, a new state is generated using the FSG algorithm according to the commanded speed.

### 4.3. Smooth transition of gait patterns for different speeds

The FSG described above guarantees SSRN which are not $-1$ or $-2$ for the given speed. However, when the speed command is changed at an instant, the machine might be in a state which is $-1$ or $-2$ according to the new commanded speed. Such a situation might occur when the speed command is increased. In the case of a decrease in the speed command, it does not occur; because, any state which is not $-1$ or $-2$ for a given speed, is also not $-1$ or $-2$ for any speed lower than that. (Consider that the supporting legs will retract less in lower speeds. Less retraction is always less likely to create a dead-end state.) The current state input to the FSG is tested if it is a $-1$ or $-2$ state regarding to the given speed. If it is so, the speed is temporarily reduced to 1 unit per iteration till a state which is not $-1$ or $-2$ is reached with respect to the newly commanded speed. It is clear that, any state visited by the robot, no matter according to which speed it is generated, is neither $-1$ nor $-2$ with respect to the speed of 1 unit per iteration. Moreover, it is possible to reach every SSRN which is not $-1$ or $-2$ for any given speed, with generation according to the speed of 1 unit per iteration. Once a non $-1$ and non $-2$ SSRN for the commanded speed is reached with generations according to 1 unit per iteration, the robot can go on with the commanded speed. The robot changes the speed only with slowing down the walk, not with stopping. The state transitions get the negative reward of $-1.5$ when the speed is lowered for transition purposes. In this way memorization of the transition states are avoided. That is why there is the third line in the formula of the reward in Eq. (1).

## 5. FGGRL when there is the external stability problem of a rear-leg deficiency

In the above introduced FGGRL, the generated transitions are guaranteed to be stable, and only learning of state transitions with larger stability margins is performed. When there is an external stability problem, as a defect in one of the legs, a SSRN cannot guarantee static stability any more. Moreover, the detections for $-1$ and $-2$ states developed till here, do not work for the five-legged case. When a rear-leg is out of order the FSG algorithm does not know it and continues to generate states as if all the six legs were in order. Then the task of learning stable transitions with states neither $-1$ nor $-2$ for the five-legged case is loaded on the reinforcement learning part of the FGGRL. The robot falls down with many of the transitions, stands up again and tries new transitions. During all these, learning is performed. For the case of five-legged walking only the speed of 1 unit per iteration is used, since any other speed is impossible with five legs for the Robot-EA308.

The stability margin with six-legged walking is assured to be larger than 4.5 cm in any instant of the walk, due to the satisfaction of the rule of neighborhood. For the five-legged case such a stability margin cannot be guaranteed. Moreover, even satisfying static stability in all instants is impossible for the Robot-EA308. Therefore, the concept of quasi-static-stability is developed for five-legged walking.

### 5.1. Quasi-static-stability for five-legged walking

In explaining the discrete model of stepping the transitions from leg-state 0 to the first supporting states were mentioned as sub-transitions, which do not have anything to do with advancement of the robot. It was again stated that these sub-transitions could be considered to be bound to the succeeding transition within the supporting states within a single iteration. In walking of the Robot-EA308 in control of the FGGRL, this is exactly realized. In this way, the legs that transform to a supporting position from the return position are put in support position, before the legs that transform to the return state of 0 are raised. Then, in the very beginning of retraction, more legs than the actual ones in the next state are supporting the robot. Since the retraction happens in a short time, a statically unstable situation in the beginning of retraction is easily compensated. Then it is enough for a state to sustain the minimum static stability margin of 2 cm at the end of the retraction. For that reason, any state that sustains the minimum static stability margin of 2 cm after retraction is designated as quasi-static-stable for the Robot-EA308. In the simulations for learning of five-legged walking, and also in the calculation of the reinforcement signal for the six-legged walking in Eq. (1), this quasi-static-stability margin is used.

Since the quasi-static-stability is calculated only at the instant the legs are retracted, there is an asymmetry considering the front and rear stability margins. This is because when the legs are retracted, the front stability margin is more likely to be less than the rear stability margin. The quasi-static-stability margin is limited mostly by the positions of the front-legs. Therefore, it becomes very hard for the Robot-EA308 to walk even with a quasi-static-stable gait when one of the front- or middle-legs is out of order. Because of this reason, the five-legged walking adopted here is limited to one of the rear-legs being out of order, rather than any of the six.

In the learning scheme for the five-legged case the reinforcement signal and the probability of using an existing state for transition generation are modified. For the learning of five-legged walking, it is not meaningful to try to improve the static stability margin. What is learned for five-legged walking is just the transitions to quasi-static-stable states. The reinforcement signal is modified as in Eq. (5). The subscript $sw$ in the reinforcement term indicates that it is used to achieve a stable walk.

$$r_{sw}^{(n)} = \begin{cases} 1, & \text{if } P^{(n+1)} \text{ is quasi-static-stable} \\ -1.5, & \text{if } P^{(n+1)} \text{ is statistically unstable} \end{cases}. \tag{5}$$

Only the transitions to quasi-static-stable states get positive rewards, and transitions to unstable states are cleaned from the memory just after they are experienced. As a result, the memorized states are all quasi-static-stable. It is reasonable to make use of the memorized states whenever it is possible. For the five-legged case, the probability of using an existing state for transition generation is made 1, rather than 0.5. The robot creates a transition to a memorized state whenever it is possible. Consequently, it is easier to catch a cycle of state transitions that result in five-legged walking.

Table 1
Successive epochs and commands sent to the simulation robot within one episode

| Epochs | A | B | C | D | E |
|---|---|---|---|---|---|
| Commands speed : (cm/step) | $v_{\text{unit}} = 1$ | $v_{\text{unit}} = 4$ | $v_{\text{unit}} = 2$ | $v_{\text{unit}} = 3$ | Deficiency on rear left leg ($v_{\text{unit}} = 1$) |
| Step numbers | 1–100 | 101–200 | 201–300 | 301–400 | 401–500 |

The following algorithm is adopted for learning walking when one of the rear-legs is out of order:

*Initialization state:* $P_i = [\underline{1}11111\underline{1}]$. This state is used to initialize the walk if the robot cannot find a stable next state succeeding its current state. It should be noted that, because of the rear-leg deficiency, either the first or the last leg-state is realized as 0, although it is commanded as 1.

*Current state at iteration n:* $P^{(n)}$ This is equalized to the initialization state at the beginning of learning.

*Number of learning steps: N*

*for n=1 to N,*

*Step* 1: Determine the next state candidate $P^*$.

If there is any memorized state possible to make a transition, it will be realized. Such candidates might or might not have been memorized as a possible transition from $P^{(n)}$.

*Step* 1.i: Check if there is any memorized transition from $P^{(n)}$. If not goto *Step 1.ii*. If there is, the next state will be selected among them with probability $p_u$ (Eq. (4)). The selection will be performed randomly according to a distribution proportional to the transition utilities of $P^{(n)}$. If this is performed, goto *Step 2*. With probability $1 - p_u$, goto *Step 1.ii*.

*Step* 1.ii: Check if there is any memorized state that is possible to make a transition. If not goto *Step 1.iii*. If there is, the selection will be performed randomly according to a distribution proportional to the sum of the transition utilities of the memorized next state candidates. Then, goto *Step 2*.

*Step* 1.iii: Use FSG algorithm to generate a new next state from $P^{(n)}$.

*Step* 2: Make the transition from $P^{(n)}$ to $P^*$.

If the transition is stable, memorize the transition, update the lookup table according to the reinforcement signal with value 1, assign $P^{(n+1)} := P^*$, $n := n + 1$, and goto Step 1.

*Step* 3: Transition is unstable. The robot falls down. Make the robot stand at the state $P^{(n)}$.

*Step* 4, 5: Repeat *Step 1*, *Step 2*, and *Step 3*.

In this way try to find a stable transition from $P^{(n)}$ for two more times.

*Step* 6: Three attempts have been performed to make a stable transition from $P^{(n)}$, but none of them has created a stable next state. Give up trying to find a stable transition from $P^{(n)}$. Assign $P^{(n+1)}$ to the initialization state, update the lookup table according to the reinforcement signal with value $-1.5$, assign $P^{(n+1)} := P_i$, $n := n + 1$, and goto Step 1.

## 6. Simulation results

The success of the FGGRL is evaluated first with computer simulation. The simulation is performed for 500 steps of the robot. Table 1 shows the successive commands that are sent to the simulation robot during these 500 steps. In the first four epochs the reinforcement in Eq. (1), for the last epoch the reinforcement in Eq. (5) are used. In the first four epochs the FGGRL is used as it is described in Section 4, and for the last epoch it is used with its form given in Section 5.

In the epochs of A and C the FGGRL is expected to generate gaits that have a stability margin close to or larger than the nominal value of 7 cm. There is a single gait with which the robot can walk with the speed of 4 units per iteration: the tripod gait with the successive states of 0-5-0-5-0-5 and 5-0-5-0-5-0. Therefore there cannot be any improvement in the stability margin in epoch B. The reinforcement in epoch B is always a large negative value, so that no state transition is memorized. In other words, the reinforcement learning part of FGGRL is not functioning. The epoch B might demonstrate only the smooth transition from the speed of 1 to 4 units per iteration, and the successful generation of the single choice of states by free state generation algorithm. The passage from epoch C to D might also demonstrate the smooth transition feature of the FGGRL. Within epoch D, although there are a few different gaits that can be followed, very few of them have a stability close to 7 cm. Although a slight improvement can be expected in stability margin, it is impossible to come close to the nominal value in the average.

In the end of epoch D and start of epoch E, the rear right leg is disfunctioned. The FGGRL algorithm is not aware of this situation. In the simulation the robot falls while walking with the speed of 3 units, and it is assumed that it gets an external signal. With this signal the robot only understands that there is a stability problem without knowing its reason. Then it passes to the speed of 1 units per iteration and adopts the form of FGGRL for external stability problem situations. In this epoch the robot tries successive states making state generations as if all the legs were in order. Many of these states result in falling of the robot, while some result in stable transitions. What is expected from the robot is to memorize the stable transitions it experiences and come up with a five-legged gait pattern that results in a continuously stable walk.

The success of the FGGRL is evaluated based on its performance in epochs A, C, and E. If the robot achieves a walk with continuous stability margin larger than 6 cm in the last ten steps of the epochs A and C it is considered to be successful, respectively, in these epochs. If the robot can achieve a continuous stable walk with five legs in the last ten steps it is considered to be successful in epoch E. In computer simulation the described 500 step walk was performed for 1000 episodes. The FGGRL was successful 891 times in epoch A, 996 times in epoch C, and 685 times in epoch E. Among those, in 604 of the trials the algorithm was successful in three of the epochs together. The algorithm is more successful with the

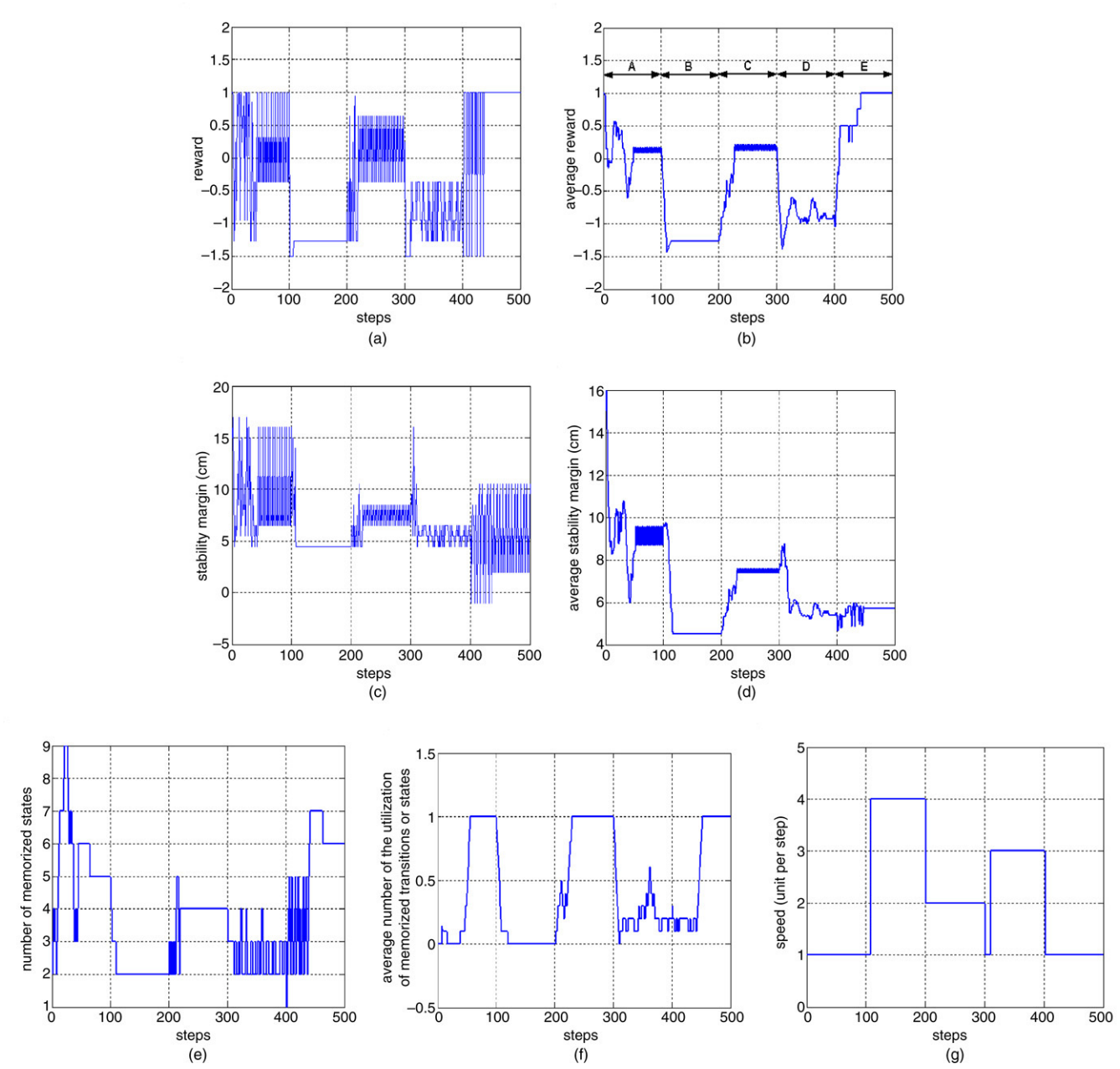


Fig. 4. Results for a successful FGGRL in computer simulation. (a) Reward for each step, (b) average reward of the last ten steps for each step, (c) stability margin for each step, (d) average stability margin of the last ten steps for each step, (e) number of memorized states for each step, (f) average number of utilization of the memorized transitions or states, (g) speed of the robot at each step.

speed of 2 units per iteration compared to 1 units. This result inclines one to comment that the ratio of the number of states with stability margin larger than 7 cm to all the possible stable states is larger for the speed of 2 units per iteration compared to 1. Although the computer calculations for the state 3-0-3-0-3-3 given in Section 2 support this comment, it needs a mathematical proof, which will not be dealt here.

In Fig. 4, the results of a successful trial is shown. In (a) the actual reward returned after each iteration is plotted. In (b) the average reward in the vicinity of each step is given. The averages in the figures are calculated using the results of preceding ten steps. In the epochs A, C, and E one can observe the increase and stabilization of the value of rewards. In epochs A and C the average reward is managed to be raised above 0 in the last 40 and 60 steps respectively. In epoch E, the average reward is 1 in the last 50 steps. This shows that the robot learned to walk with five legs in about 50 steps. In this figure, the number of attempts performed after falling, which are explained in the 4th and 5th steps of the five-legged learning algorithm, are not included. The number of these attempts for this result is 23. The pattern of successive states learned for five-legged walking is as follows,

[4 3 0 4 0 X] → [5 4 0 5 2 X] → [0 5 3 0 3 X]
→ [2 0 4 2 4 X] → [3 2 5 3 5 X] → [4 3 0 4 0 X]


Fig. 5. Experimental setup for the Robot-EA308.

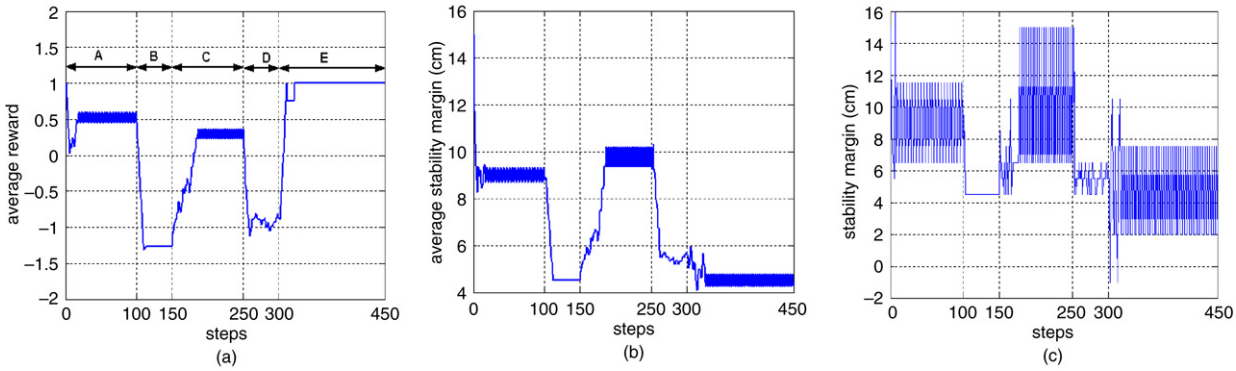


Fig. 6. Results for a successful FGGRL on the actual Robot-EA308. (a) Reward for each step, (b) average reward of the last ten steps for each step, (c) stability margin for each step.

with the respective stability margins of 2, 5.5, 2, 8.5, 10.5, and again 2 cm. The sixth leg is crossed, because it is not used by the robot, although commanded by the FGGRL.

In (c) and (d), in Fig. 4, the actual and average stability margins are given, respectively, for each iteration. One can observe the affinity between the average stability margin and reward traces for the epochs A, B, C, and D. Such an affinity is already not expected for the epoch E, because the reward does not depend directly on the stability margin as in the others. In (e) and (f), the average number of utilization of memorized states or transitions and the number of states memorized are given respectively. As it is observed, in the beginning of the epochs A, C, and E the number of memorized states increases, and in the middle or towards the end it decreases and remains at a constant value. Accordingly, the utilization of the memorized states is low at the beginning of the three epochs, and full utilization is performed after the middle. For the epoch B the utilization is always in the zero level, while in D it remains in very low values. The two states memorized in epoch B are the initialization state which always exists in the memory, and the current state; namely, no new state memorization is performed in epoch B. In figure (g) the actual speed of the robot is given for each step. One should observe that immediately after the 100th and 300th steps, the actual speed is not the commanded speed but 1 unit per step. This is because the robot advances with the speed of 1 unit per step in order to catch a suitable state from which it can change to the commanded speed.

## 7. Results with the actual Robot-EA308

The FGGRL is applied to the Robot-EA308 with the experimental setup shown in Fig. 5. The Robot-EA308 is a kinematically controlled three-joint six-legged robot. The servo-motors on the joints of the legs are controlled via the servo-controllers on the robot. For the learning of five-legged walking the robot is equipped with four switches beneath its body frame, and the necessary hardware to send the signal of falling to the computer. When one of these switches is closed, the robot is considered to fall down, and the controlling computer can detect this through the connection between the hardware on the robot and its parallel port. The successive joint angle commands, which result in motions of the legs, are calculated by the controlling computer according to the output of the FGGRL algorithm. These comments are sent to the servo-controllers on the robot via the serial port of the computer.

It should be noted that in epochs A to D, the reinforcement is based on the stability margin; therefore, it is calculated internally by the computer. It is only in epoch E, namely after one of the rear-legs is out of order, that a real feedback from the actual robot is used as the reinforcement signal. In Fig. 6, the results of a successful FGGRL application on the actual

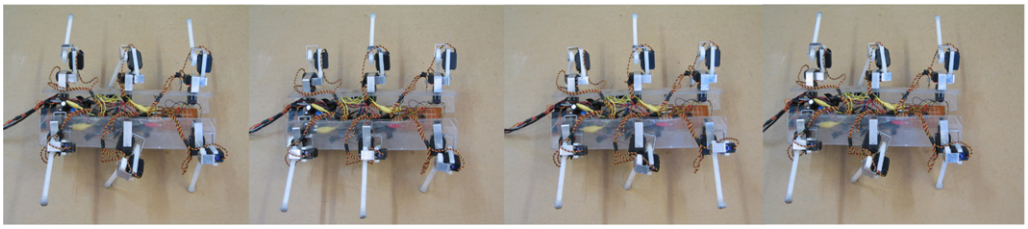

Fig. 7. Slides showing the Robot-EA308 while walking with the five-legged gait pattern composed of the sequential states [0 5 0 3 5 X] → [4 0 2 4 0 X] → [5 0 3 0 4 X] → [0 5 0 3 5 X], from left-to-right.

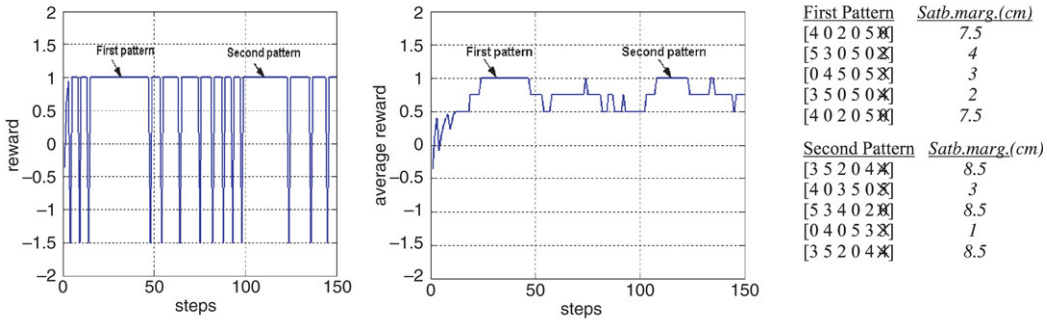


Fig. 8. Results of a five-legged learning episode, in which the "on-the-border" problem is observed. (a) Reward for each step, (b) average reward of the last ten steps for each step, (c) the first and second patterns of states with their stability margins.

Robot-EA308 are shown. For this result, again the right rear-leg (sixth leg) is out of order in epoch E. Since the results obtained on the real robot are very similar to the ones obtained from the computer simulations, only the figures for average reward, average stability margin, and stability margin are given. In the application on the real robot, the length of epochs B and D are limited to 50 steps, because the gait patterns generated in these epochs do not differ much in time. The epoch E, on the other hand, is enlarged to 150 steps, in order to see the robot walking with five legs for some time.

For the epochs A and C, the robot manages a walk with large stability margins towards the middle of the epochs. It was possible to observe the smooth passage from 2 units to 3 units per step in the very beginning of the epoch D: The robot first made steps with 1 unit advancement, and passed to 3 units after three steps.

One can observe in Fig. 6, that the robot learned to walk with five legs after about 23 real steps. The number of attempts due to the falling throughout these 23 steps was 18. The states that constitute the five-legged walk after the 23 steps are

[0 5 0 3 5 X] → [4 0 2 4 0 X] → [5 0 3 0 4 X]
→ [0 5 0 3 5 X]

with stability margins of 4, 2, 7.5, and again back to 4 cm respectively. The slides that show the robot in these states are given in Fig. 7.

For the application in Fig. 6, it is observed that although the robot found a pattern of five-legged walking, it occasionally lost its stability on the state with the minimum stability margin of 2 cm. However, in its first attempt it managed to have a stable step with the same state. (Since the attempts are not considered as real steps, the falls are not observed on the figures.) Therefore, these falls did not alter the learned pattern in this example. However, in some other episodes of learning of five-legged walking with the actual robot, the irregular stability of some states created problems for learning. For example, a state with stability margin of 2 cm resulted in falling in some cases; or, a state with stability margin of 1 cm, which was considered to be unstable in the simulation, resulted in a stable standing in some cases, but falling in others. Such "on-the-border" states result the following: First the robot considers the state as stable, and starts to establish a pattern with it. But after some time the state either becomes totally unstable, or sometimes stable and sometimes unstable. In such a case, if the robot manages to make the state stable in its attempts, the learned pattern is preserved, with occasional falls and recoveries. Otherwise, the learned pattern is lost with cleaning of the problematic state from the memory. In Fig. 8, the results of a five-legged learning episode is given as an example in which the problem of on-the-border state is observed. During this episode a pattern (First Pattern) is established and used between the 24th and 47th steps. In step 48, the state with stability margin of 2 cm, behaved unstable and the pattern was lost. The robot managed to establish another pattern (Second Pattern) with a state of 1 cm stability margin, and used it between the 118th and 123rd steps. However, at the 124th step this state behaved unstable and the pattern was again lost.

With the actual robot 30 episodes of 150 steps were performed for learning of five-legged walking. In half of those the right, and in the other half the left rear-leg was deficient. No difference was observed in this regard. The result of each episode is categorized as "perfect", "acceptable", "late", and

"not-learned". "Perfect" means that the robot learned a five-legged walking pattern in the first 100 steps and managed to finish the total 150 steps with the same pattern. "Acceptable" means, the robot managed to learn a five-legged walking pattern in the first 100 steps, but because of "on-the-border" states, lost the learned pattern. "Late" means, the robot could learn a five-legged walking pattern only in the last 50 steps. And lastly, "not-learned" means the robot could not learn a pattern in all 150 steps. Among the 30 episodes, the results of 9 were "perfect", 10 were "acceptable", 5 were "late", and 6 were "not-learned". With the perfect results the robot learned five-legged walking in about 5–10 min and finished the 150 steps in less than 15 min. With the acceptable results, although a pattern is generated again in 5–10 min, the fulfillment of 150 steps lasted up to 20 min. The late and not-learned episodes were finished in between 20 and 25 min.

If having established a five-legged walking pattern in the first 100 steps is considered as "successful", as done for the simulation results, 19 of the episodes ("perfect" or "acceptable") were successful, and the remaining 11 ("late" and "not-learned") were unsuccessful, with a success ratio of 63%. This is only slightly less than that of the five-legged learning in the simulation. Why isn't there a considerable decrease despite the negative effects of the "on-the-border" phenomena? Because, with the actual robot some states with a stability margin of 1 or even 0 cm, result in stable standings and are used in the patterns. This is not possible with the simulation, where any state with a stability margin less than 2 cm is consistently unstable. Existence of such states in the real robot application, slightly enlarges the set of stable states in the search space. It should be noted, however, that the negative effect of "on-the-border" states might be larger in other samples of episodes. Therefore, "on-the-border" effect should be stressed here as a severe problem to be considered in application of reinforcement learning to learning of walking.

The reader might access the web site for the Robot-EA308 at the address http://www.eee.metu.edu.tr/~suphi/robot_ea308.html. In this site short movies of the robot can be found. Among these one can find movies showing the smooth transitions between different speeds, and movies showing the learning of five-legged walking. These movies are intended to better demonstrate the real-time application of the FGGRL on the Robot-EA308.

## 8. Conclusion

In this paper the problems of free gait generation, continuous improvement of gaits, and adaptation to the unexpected condition of a rear-leg deficiency are addressed for a six-legged robot. While the free gait generation corresponds to the central part of gait generation, the reinforcement learning incorporated with this corresponds to the reactive part. The possible walking patterns are generated by the free gait generation and among these the ones most successful for adaptation are memorized and utilized by the reinforcement learning. In this way the approaches of central pattern generation and reflex model are somewhat conciliated and their efficiency are utilized in a single gait generation scheme.

The reinforcement learning makes the robot adapt to more stable gaits in normal conditions of no external effect of instability. As a result, the stability of walking with a commanded speed is continuously increased till a pattern having good stability margin with each state is achieved. When the speed command is changed, the robot performs a smooth passage with probably slowing down to the speed of 1 units per step, and then adapting to the new speed in a few steps. This smooth passage between commanded speeds should be noted as a peculiar feature of the algorithm here. Both increasing the stability margin and smooth passage between different speeds are successfully observed on the real Robot-EA308.

The adaptation capability of the learning scheme is tested on an unexpected case for the robot, namely when there occurs a deficiency on one of the rear-legs. In this application the robot is not aware of the cause, but experiences falls although it generates statically stable gaits. The falls result in negative reinforcement and the robot learns to avoid them by memorizing and making use of the stable states. The learning of five-legged walking is realized with the actual Robot-EA308. The switches beneath the robot body enable the computer to realize the falls.

The main difference between applying the reinforcement learning with simulation and with the actual robot is designated as the effect of "on-the-border" situations. This problem rises especially because the reinforcement signal taken from the actual robot has a binary character. The robot either falls or not. The scheme here cannot handle the situations when a state behaves indefinitely. If the reinforcement signal could designate a continuous stability value it would be better for the algorithm to avoid on-the-border states. However, for the case of avoidance of falling, such a reinforcement signal is not easy to be generated with simple switches. One way to overcome this problem might be to incorporate the internal reinforcement, designating the stability margin depending on the simulation model, with the reinforcement taken from the actual robot. Future work will dwell upon developing the reinforcement signal in order to generate five-legged stable gaits that preserve their stability in case of considerable external indefiniteness.

## References

[1] G. Brambilla, J. Buchli, J. Ijspeert, Adaptive four legged locomotion control based on nonlinear dynamical systems, in: From Animals to Animats — Proceedings of the Ninth International Conference on the Simulation of Adaptive Behavior (SAB'06), vol. 4095, in: Lecture Notes in Computer Science, Springer Verlag, 2006.

[2] H. Cruse, T. Kindermann, M. Schumm, J. Dean, J. Schmitz, Walknet—a biologically inspired network to control six legged walking, Neural Networks 11 (1998) 1435–1447.

[3] M.D. Donner, Real Time Control of Walking, Birkhäuser, Boston, 1987, pp. 7–16.

[4] M.S. Erden, K. Leblebicioğlu, Multi legged walking in robotics and dynamic gait pattern generation for a six-legged robot with reinforcement learning, in: Mobile Robots: New Research, Nova Publishers, New York, ISBN: 1-59454-359-3, 2006.

[5] M.S. Erden, K. Leblebicioğlu, Analysis of wave gaits for energy efficiency, Autonomous Robots 23 (3) (2007) 213–230, doi:10.1007/s10514-007-9041-z.

[6] M.S. Erden, K. Leblebicioğlu, U. Halıcı, Multi-agent system based fuzzy controller design with genetic tuning for a service mobile manipulator robot in the hand-over task, Journal of Intelligent and Robotic Systems 38 (2004) 287–306.
[7] C. Ferrell, A comparison of three insect-inspired locomotion controllers, Robotics and Autonomous Systems 16 (1995) 135–159.
[8] F. Hardarson, Locomotion for difficult terrain, Technical Report TRITA-MMK 1998:3, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, April 1998, 1998, ISSN 1400-1179, ISRN KTH/MMK-98/9-SE.
[9] Q.J. Huang, K. Nomani, Humanitarian mine detecting six-legged walking robot and hybrid neuro walking control with position/force control, Mechatronics 13 (2003) 773–790.
[10] M. Huber, R.A. Grupen, A feedback control structure for online-learning tasks, Robotics and Autonomous Systems 22 (1997) 303–315.
[11] W. Ilg, K. Berns, A learning architecture based on reinforcement learning for adaptive control of the walking machine LAURON, Robotics and Autonomous Systems 15 (1995) 321–334.
[12] W. Ilg, K. Bernes, T. Mühlfriedel, R. Dillman, Hybrid learning concepts based on self-organizing neural networks for adaptive control of walking machines, Robotics and Autonomous Systems 22 (1997) 317–327.
[13] K. Inagaki, Gait study for hexapod walking with disabled leg, in: Proceedings of the 1997 IEEE/RSJ — International Conference on Intelligent Robots and Systems (IROS'97), vol. 1, 1997, pp. 408–413.
[14] S. Inagaki, H. Yuasa, A. Tamio, CPG model for autonomous decentralized multi-legged robot system—generation and transition of oscillation patterns and dynamics of oscillators, Robotics and Autonomous Systems 44 (2003) 171–179.
[15] E. Karalarli, A.M. Erkmen, I. Erkmen, Intelligent gait synthesizer for hexapod walking rescue robots, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, April 2004, 2004, pp. 2177–2182.
[16] H. Kimura, T. Yamashita, S. Kobayashi, Reinforcement learning of walking behavior for a four-legged robot, in: Proceedings of the 40th IEEE Conference on Decisions and Control, Oriando, Florida, USA, December 2001, 2001.
[17] F. Kirchner, Q-learning of complex behaviours on a six-legged walking machine, Robotics and Autonomous Systems 25 (1998) 253–262.
[18] B. Klaassen, R. Linnemann, D. Spenneberg, F. Kirchner, Biomimetic walking robot SCORPION: Control and modeling, Robotics and Autonomous Systems 41 (2002) 69–76.
[19] D.E. Koditschek, R.J. Full, M. Buehler, Mechanical aspects of legged locomotion control, Arthropod Structure and Development 33 (2004) 251–272.
[20] P. Maes, R.A. Brooks, Learning to coordinate behaviors, in: Proceedings of the American Association of Artificial Intelligence, Boston, MA, 1990, pp. 796–802.
[21] A. Mahajan, F. Figueroa, Four-legged intelligent mobile autonomous robot, Robotics and Computer Integrated Manufacturing 13 (1) (1997) 51–61.
[22] P.K. Pal, V. Mahadev, K. Jayarajan, Gait generation for a six-legged walking machine through graph search, in: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 2, 1994, pp. 1332–1337.
[23] J.M. Porta, E. Celaya, Efficient gait generation using reinforcement learning, in: Proceedings of the 4th International Conference on Climbing and Walking Robots, Clawar, Karlsruhe, Germany, Professional Engineering Publishing, ISBN 1-86058-365-2, 2001, pp. 411–418.
[24] J.M. Porta, E. Celaya, Reactive free gait generation to follow arbitrary trajectories with a hexapod robot, Robotics and Autonomous Systems 47 (2004) 187–201.
[25] D.K. Pratihar, K. Deb, A. Ghosh, Optimal path and gait generations simultaneously of a six-legged robot using GA-fuzzy approach, Robotics and Autonomous Systems 41 (2002) 1–20.
[26] S.M. Song, K.J. Waldron, An analytical approach for gait study and its applications on wave gaits, The International Journal of Robotics Research 6 (2) (1987) 60–71.
[27] M.M. Svinin, K. Yamada, K. Ueda, Emergent synthesis of motion patterns for locomotion robots, Artificial Intelligence in Engineering 15 (2001) 353–363.
[28] J.D. Weingarten, G.A.D. Lopes, M. Buehler, R.E. Groff, D.E. Koditschek, Automated gait adaptation for legged robots, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, April 2004, 2004, pp. 2153–2158.

**Dr. Mustafa Suphi Erden** was born in Ankara in 1977. He received the B.Sc., M.Sc. and Ph.D. degrees all from the Electrical and Electronics Engineering Department at Middle East Technical University, Ankara, Turkey. He worked as a research assistant in the same department between 1999 and 2006. Currently, he is a post-doc researcher in Technical University of Delft, The Netherlands.

**Prof. Kemal Leblebicioğlu** was born in Turkey on May 12, 1957. He received the B.Sc. degree in Electrical and Electronics Engineering in 1979, and the M.Sc. and Ph.D. degrees in Mathematics in 1982 and 1988, respectively, all from the Middle East Technical University (METU), Ankara, Turkey. From 1980 to 1988, he was with the Department of Mathematics at METU as a graduate assistant. Since 1988, he has been on the Faculty of the Department of Electrical and Electronics Engineering at METU, where he is currently a professor. His research interests include the numerical methods for partial differential equations, inverse problems, optimization, optimal control theory, neural networks, neuro and fuzzy controllers, intelligent systems, robotics, and image processing.