Walking in Unstructured Natural Environments

Josep M. Porta and Enric Celaya
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Gran Capità 2-4, 08034, Barcelona, SPAIN

Abstract

In this paper we present a complete behavior-based controller that allows a six legged robot to walk in unstructured environments. The task of walking on rough terrain is decomposed in two subtasks: terrain adaptation and movement generation. The first one maintains the stability of the robot while the second is in charge of the advance movement. We present solutions to each of these two tasks and show how they can be integrated into a single controller so that further improvements can be made on each of them without modifying the other. The results of a series of evaluation test performed on a real robot are presented.

1 Introduction

One of the main objectives in the field of mobile robots is that of building systems able to operate in natural environments, which usually are dynamic, unstructured, and (too often) hazardous, making this objective really hard to achieve.

Legged robots are a special case of mobile robots. Their control involves more aspects than that of wheeled robots. While in a wheeled robot just two parameters (speed and direction) are enough to command the robot (and usually these parameters can be directly translated to motor orders), to reach an equivalent functionality with a legged robot we have to take into account aspects such as leg coordination, gait generation, turning strategies, etc. As a counterpart, legged robots are much better suited for traversing highly unstructured rough terrain, where obstacles of any size can appear at every step. This kind of terrain should be considered as the natural habitat of a legged robot.

The task of walking on rough terrain has two components: a static and a dynamic one. The first can be defined as follows:

> *Given the current position of the robot, adopt a posture as stable as possible.*

The dynamic component is in charge of the advance movement of the robot and can be thought as:

> *Issue the appropriate sequence of leg movements so that the robot advances with the desired speed and direction.*

We call these two components the *Terrain Adaptation* and the *Movement Generation* problems, respectively. To build a complete controller for a legged robot, both problems have to be solved. The common approach is to develop a movement generation module and enlarge it to include some aspects of terrain adaptation [2], [6]. In [5] we proposed to use the inverse approximation: solving the terrain adaptation problem first and then include aspects of movement generation, and we argued that this approach lends to simpler and more robust controllers.

Nevertheless, there is no need, in principle, to impose the primacy of one problem over the other. Both problems can be solved separately, provided the resulting solutions can be properly integrated at the end. In the next sections we present our solutions to these two problems and the way in which they can be integrated so that they can work together with a good performance. The level of performance achieved with the proposed control structure is illustrated with quantitative results obtained from tests performed on a real robot.

2 Work Frame

2.1 The Robot

As a testbed for our experiments we have used Genghis II, a commercially available six-legged robot. Its body is about 35 cm length, legs measure 10 cm and when they are completely vertical the body is 8 cm separated from the ground. The maximum lift position of a leg maintaining the body horizontal is 13 cm (the leg is completely raised while the other legs are vertical). Each leg is provided with two motors, one for advance (the α motor) and one for lift movements (the β motor). A force sensor is provided for each motor. Other sensors of the robot are, two whiskers, a belly contact sensor, four infrared emitter-receiver sensors, five passive pyro sensors, and one inclinometer in the advance direction of the robot.

For more details on Genghis hardware you can see [8].

2.2 The Software Architecture and the Programming Language

In rough terrain, a legged robot must react quickly to unforeseen problems and it is difficult to construct and maintain an exact model of the environment since it does not present regularities and tends to be dynamic. These characteristics discourage the use of symbolic architectures to program legged robots.

The behavior-based approach seems to be more useful for this task. It does not require the construction of an environment model. It is based on reactive modules that operate in parallel providing quick responses to the terrain features detected by the robot. For this reason we use the subsumption architecture [4] (a behavior-based architecture) to implement our controller. The basic ingredients of this architecture are:

1. Controllers are composed of processes (called behaviors) that work in parallel and that can share information through message passing.

2. Precedence between behaviors is achieved with message suppression and inhibition.

3. The task is decomposed in layers. The philosophy behind subsumption is that the activity of lower layers should contribute to the workings of the higher ones. This provides a methodology to build and test controllers incrementally.

Brooks designed a special language [3] (called Behavior Language, BL) to implement subsumption-based controllers. Programs for Genghis' BL compiler are developed on a personal computer and transferred to the robot's memory (of 32 Kb) for its execution. This imposes a short limit to the size of the program that controls the robot. So we decided to translate the execution to a personal computer and communicate sensor readings from the robot and commands to the motors through serial line. This supposes the lost of autonomy of the robot and the slow down of reactivity due to serial line delays, but as a counterpart, we can use all the personal computer resources (a file system, a large main memory, a floating point unit and pre-existing software).

We have developed a compiler [11] to generate executable code for the personal computer. In this compiler, the basic structure of BL has been maintained, but a number of features have been improved and some restrictions have been removed:

- While in BL there are just two levels of priority for messages, in our language the programmer can use up to 256 levels of priority, so he can avoid artificial constructions to get a similar effect [5].
- In BL the inhibition of messages can only affect output messages and in our language it can affect both outputs and inputs.
- Behaviors can be grouped into modules so that its activation can be controlled as a whole.
- There is the possibility to share memory between behaviors since the modules can include variable declarations.
- The user can define its own data types (as in other behavior based languages [7]) and functions.
- In BL messages can only contain simple bytes and in our language they can contain any data type.
- In our language, the suppress/inhibit time is variable while in BL it was prefixed.

With our programming language one can program following the subsumption architecture principles but one can also choose to use other philosophies since the language does not restrict the programmer in this aspect. It is the programmer responsibility to use a coherent programming style.

3 The Terrain Adaptation problem

The purpose of terrain adaptation is to adopt a stable stance in any surface. This is accomplished by including mechanisms to keep legs in contact with the ground, the body in an appropriate position with respect to the local surface, and the vertical projection of the center of mass well centered with respect to the supporting legs. All these tasks can be seen as environment driven tasks, so we have implemented them with reflexes. A reflex is a simple and pre-programmed process that is executed every time a specific sensorial pattern is detected.

According to its sensorial inputs we have three kinds of terrain adaptation reflexes:

- Leg reflexes: They involve individual legs and respond to the signals of their own local sensors.
- Proprioceptive reflexes: They respond to the posture adopted by legs at any time.
- Body reflexes: They react to perceptual conditions related with the body as a whole.

3.1 Leg reflexes

These reflexes are in charge of the primary terrain adaptation mechanism that consists in trying to keep all legs in contact with the ground. Each leg has three reflexes:

- The *Land* reflex: Whenever a foot is unloaded (indicating that it has lost contact with ground) the leg is moved down until a new contact with the ground is detected.

- The *Search* reflex: If a leg can not reach the ground with a descending movement, it steps forward and backward in order to find a support point in the surrounding area.

- The *Skip* reflex: If a leg collides with an obstacle while moving horizontally, then the leg is retracted and the movement is re-issued but at a higher position with the hope of placing the food over the obstacle.

All these reflexes have been observed in insects [9], [10]. The combination of these reflexes results in a good likelihood that all legs find a supporting surface, increasing the stability.

3.2 Proprioceptive reflexes

If we let the leg reflexes work without control then the robot can evolve towards inappropriate *postures* (or configuration of leg positions with respect to the body). The purpose of the proprioceptive reflexes is to restrict the posture according to several criteria. These criteria are controlled by specific reflexes that we call *balances* and that we describe next. For those aspects of the posture controlled by the lift motors we consider three balances (that we call β balances):

1. Height balance.
   The average lift position of legs should be maintained such that legs stay vertical enough to avoid body contact with ground and to reduce the energy expended by motors against gravity, but not too much to improve stability and to allow force readings to be measured in the lift motors.

2. Roll balance.
   The average lift position of legs on the right side of the robot should be balanced with the average lift position of legs on the left side to avoid a lateral (roll) decompensation of the robot.

3. Pitch balance.
   The average lift position of the anterior legs of the robot should be balanced with the average lift position of the posterior legs to avoid a longitudinal (pitch) asymmetry of the robot.

For those aspects of the posture controlled by the advance motors we consider two balances (called α balances):

1. Advance balance.
   The average advance position of legs should be maintained so that the robot's center of mass is centered with respect to the body.

2. Yaw balance.
   The average advance position of legs on the right side of the robot should be balanced with the average advance position of legs on the left side to maintain the orientation (yaw) of the body.

When the posture does not fulfill one of these criteria, then the corresponding balance issues a simultaneous movement of all legs, that we call *gesture*, to correct the situation. A gesture can be seen as a higher level form of motor control, in which legs are not moved individually but in a coordinated way.

The six advance and the six lift positions of legs define two six-dimensional spaces of

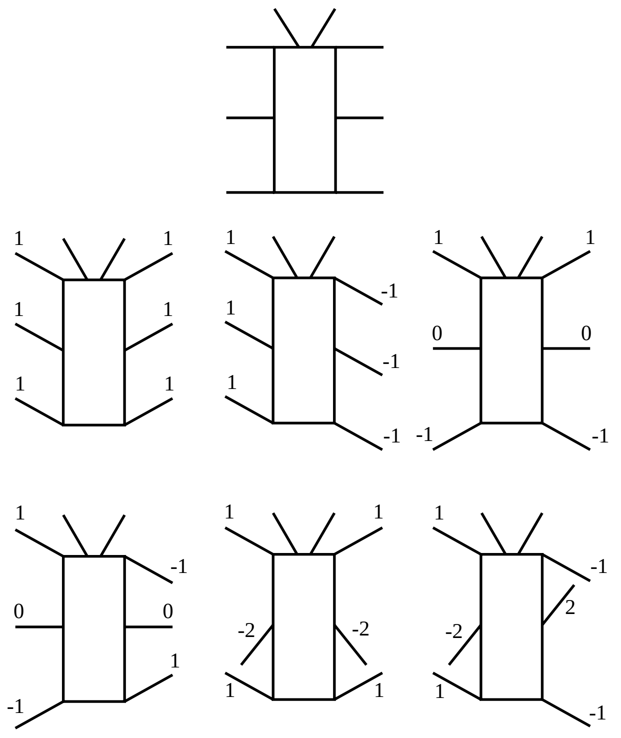

Figure 1: *Default posture and a the six orthogonal basic α gestures. The corresponding gestures applied to the lift motors constitute the basic β gestures.*

postures, and their movements define corresponding six-dimensional spaces of gestures. A convenient orthogonal basis for the space of α gestures is shown in Fig. 1. The same basis can be applied to the space of β gestures. We call these basis the α and β basic gestures, respectively. An arbitrary leg displacement can be obtained by a unique combination of the six basic β gestures for the lift motors and the six basic α gestures for the advance motors. Height, roll, and pitch balances are corrected by means of the first, second and third basic β gestures, respectively, while the advance and yaw balances are corrected by means of the first and second basic α gestures. Since the basic gestures are orthogonal, each balance is only affected by its associated gesture, and corrections to different balances can be made independently.

Note that while the first three basic β gestures control the body's height and attitude, the last three ones change the shape of the supporting surface defined by feet. To allow the accommodation of feet to arbitrary ground shapes, we should not impose further restrictions involving the last three basic β gestures. Similarly, while the first two basic α gestures control the body's advance and orientation, the effect of the remaining four basic α gestures is a slip of feet on ground with no net motion of the body. This is the reason why we do not introduce more balance reflexes.

The action of each balance is only triggered when its deviation reaches a given threshold. In order to perform smooth movements, each balance tries to reach its target value incrementally, making successive small corrections according to its associated gesture.

The posture control is a second priority adaptation task, so it only works on those legs that are not executing any leg reflex.

The use of balances is a powerful though simple way to coordinate leg movements and

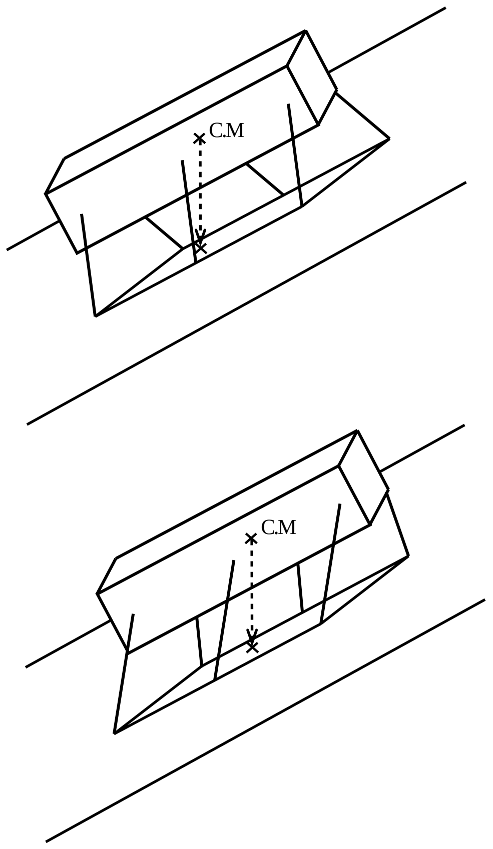

Figure 2: *A wrong position in a slope area (top) and the same situation corrected (bottom).*

to provide cooperation between them. For instance,when a leg is raised, the height balance lowers the other legs lifting the body, and the pitch and roll balances elevate the body by the side of the raised leg helping it to reach a higher position. When a leg moves forward the rest of legs are moved backward (by the advance balance) helping the first to advance further. In this way the mechanical limits of the working area of each leg are overcome by the inter-leg cooperation.

3.2.1 Body reflexes

These reflexes adjust the position and attitude of the body by modifying the default values aimed at by the balances according to several environment features detected by sensors situated in different parts of the robot's body.They are:

- The *Slope adaptation* reflex: When the robot is on a slope area (detected by the inclinometer) it moves all the legs backward (in a ascendant slope) or forward (in a descendant one) to keep the projection of the center of mass better centered in the support polygon defined by legs in contact with the ground (see Fig. 2).

- The *Height and pitch modification* reflex: If the belly contact sensor of the robot detects an obstacle, then the average height and pitch aimed at by the corresponding balances are modified to avoid the contact.

- The *Descendent step adaptation* reflex: If a front or rear leg can not find a supporting surface, then the robot lowers its body on the side of the problematic leg approaching it to the ground.

4 Movement Generation problem

This problem is usually formulated as a task without external sensorial feedback. This supposes to assume that the robot moves in the simplest of the environments: flat terrain.

A legged robot moves forwards when it steps some legs while the rest of legs (supporting legs) move synchronously backwards (in the leg retraction movement). So a legged-robot controller must decide which legs have to step and the consequent retraction velocity of the supporting legs (slow velocities would be inefficient and too fast ones would move some legs to its mechanical limit making them to be dragged).

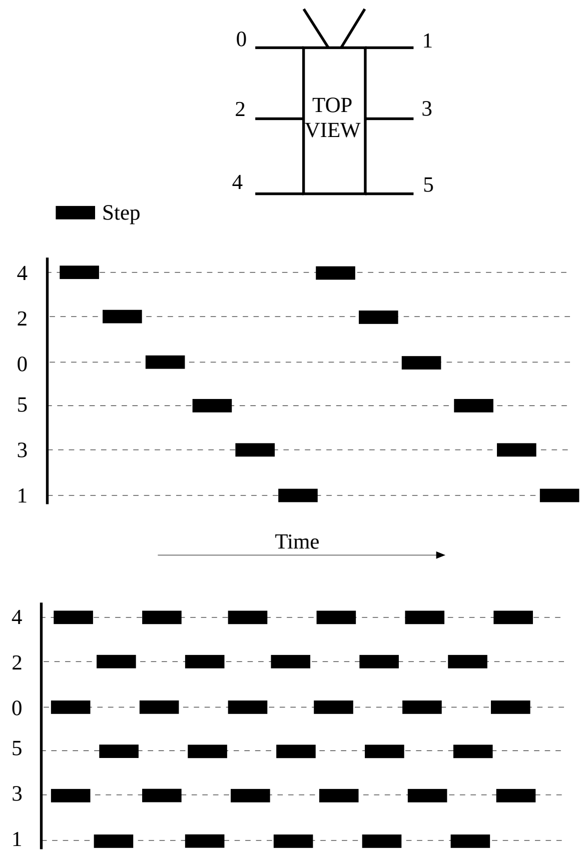

Figure 3: *The two extreme members of the metachronal family of gaits: Slow gait (top) and Tripod gait (bottom).*

Deciding which legs have to step is not an easy task. Studies of insects' gaits can be used as inspiration. In [13] we can find a description of insects' gaits on smooth horizontal surfaces. These gaits are the metachronal gaits that have been shown to be the most stable cyclic gaits [12]. For a six legged robot, one of the members of this family is the tripod gait (in which three legs are stepping while the others are supporting the body) that is the fastest statically stable gait (see Fig. 3).

But the gait selection is not the only component of the movement generation task. It also includes speed and advance direction control. These two aspects can influence both the gait selection and the movements of each individual leg.

From this overview of the movement generation problem we can conclude that a controller for a legged robot should include mechanisms for the gait generation, for adjusting the retraction movement of the supporting legs according to the current gait, for regulating the advance speed, and for changing the advance direction. Next we describe how we have solved them in our controller.

4.1 Gait Generation mechanism

The gait generation mechanism should provide stability and coordination between legs:

- Stability: Performing a step a leg should not put the robot in a unstable posture. In the case of our robot, not having neighbor legs raised at the same time (Fig. 4) is a sufficient condition to maintain a stable posture. To fulfill this condition we have to:

  - Avoid rising a leg if one of its neighbors is already on the air. This requisite is satisfied by means of a rule that only allows a leg to step if its two neighbors are on the ground. If we assume that our robot walks on flat terrain, we can use the lift position of the leg to determine whether a leg is on the ground or not.

  – Avoid stepping two neighbor legs simultaneously: This requirement is implemented with a token passing protocol between neighbor legs. Once a leg has stepped passes a token to each of its two neighbors. Only those legs that have the both tokens coming from its two neighbors are allowed to step. There is one token shared between each couple of neighbor legs and it is not the case that two neighbor legs have the token at the same time. This avoids simultaneous steps. Initially the tokens are set as shown in Fig. 5.

- Coordination: The sequence of steps should involve all legs uniformly. A minimal condition to achieve coordination is that a leg should not step again if its neighbors have not stepped. This condition is assured by the previously described token passing protocol.

The step movement consists in raising the leg, moving it forward to a predetermined position and moving it down to the ground position.

It is remarkable that our gait generation mechanism is completely local since the decision of making a leg to step depends only on the state of the two neighbor legs and not on a global evaluation of all leg states. This completely local mechanism contrasts with other existing gait generators [2], [6]. In our controller there is no restriction between the two middle legs (as it exist in other gait generation mechanisms [1]) so they can step together as it is observed in some insects [10].

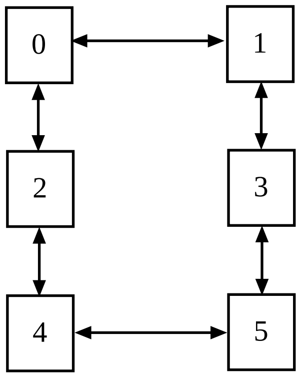

Figure 4: *Neighbor relation between legs used for the token passing and the support relation between legs.*

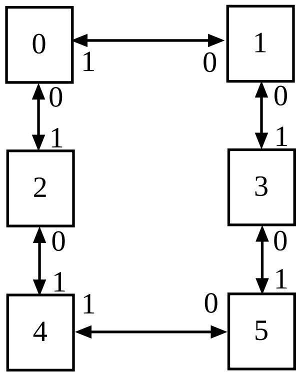

Figure 5: *Initial token distribution with 1 meaning that the leg has the corresponding token. With this distribution the only leg enabled to step is number 4.*

This gait generation mechanism produces the tripod gait, but other gaits inside the metachronal family can be obtained by introducing a delay between the satisfaction of all conditions to execute a step with the rear legs and the actual step execution. For instance, with a delay of three times the time to perform a step, the emerging gait is the slow gait (see Fig. 3).

4.2 Leg retraction

The retraction velocity of the legs has to be automatically adapted to the current gait.

Brooks (in [2]) uses a simple mechanism that produces that effect so we adopt it. This mechanism is completely equivalent to the advance balance described in section 3.2 except that its action has to be limited to those legs that are not stepping.

4.3 Speed and Direction control

There are, at least, three ways to control the advance speed of our robot:

1. Decrementing the step length of all the legs.
2. Changing the time between consecutive steps.
3. Changing the gait. This is easy to do just by adjusting one parameter (the step delay between rear legs).

The first two ways to control the advance speed are independent of the gait generation mechanism so they are applicable even if we change it.

To make the robot turn it is necessary to set different advance speed to each side of the robot. The last two mentioned methods to change the speed affect both sides of the robot because the token passing protocol extends their effects to all legs, but the first method can be independently applied for each leg. If we set different step lengths in both sides of the robot, then a difference in the average advance position of the right legs with respect to the left ones will appear. To correct this difference we use the Yaw balance (introduced in section 3.2). This balance, when compensating the difference, makes the robot turn. The largest the difference between the step lengths of both sides of the robot, the strongest the turn effect is. When the two sides use opposite step lengths then the robot turns in place.

These control strategies are useful if we want to continuously modify the advance speed and direction of the robot, but they are inappropriate when we want to brusquely change them. Our gait generation mechanism, however, gives us the chance of inverting the advance speed: if we flip the tokens assigned to all legs at the same time that we invert their step length then the robot will immediately undo the last executed steps. It also permits to start a turn-in-place movement just flipping the tokens shared between legs of one side of the robot while changing their step length.

5 Integration process

Once we have solved each one of the two problems posed in the introduction we have to join the partial solutions to get a controller that solves our primary objective: walking on rough terrain. The process of combining the two solutions that we have presented can be seen as enlarging the terrain adaptation module with a dynamic component that continuously changes the position at which the robot is trying to adapt. Conversely, it can be seen as expanding the movement generation module with the capacity to operate in any environment (not just in flat terrain).

When we join the two presented controllers we detect some interactions that must be solved:

- There are two Advance and Yaw balances: As we have explained in section 4.2, the advance an yaw balances appear in both controllers. When they are working together we have to disconnect one of them to avoid duplication.
- There are two different behaviors that try to descend a leg to the ground: the Land reflex and the descending movement of the step behavior. We remove the second and keep the former because it does not

suppose that the ground is in a pre-fixed position.

- There are two kinds of logical sensors for ground detection. Ones are based on the force readings of the lift motors and the other are based on the current lift position of the leg. Since in irregular terrain the ground can be at any height, the second logical sensors must be disconnected and the modules that received theirs signals should receive information from the other sensors. This change alters the inputs to the gait generation mechanism (but not the mechanism itself) and extends the inhibition on the leg retraction (now it affects to all legs that are not in contact with the ground and not just to those legs that are stepping).
- There are some behaviors that can try to move a leg simultaneously. For instance, the step behavior of a leg can be triggered at the same time that a leg reflex is controlling that leg. We have to establish a priority system between them. We choose to assign the higher priority to the step behavior because it is not sensible to keep on looking for a supporting surface (that is the job of the leg reflexes) if a leg is in conditions to step (and look for this surface on a more advanced position). With this decision we have three levels of priority: the step behavior is the most prioritary movement and the leg reflexes have more priority than the balances. This three level priority system was difficult to construct in Brooks BL [5] but its easy to implement in our new language [11].

After all these modifications some parameters must be tuned. For instance, the thresholds allowed by the balances and the size of their correction movements have to be adjusted to produce smooth walking motion.

With a good design of the terrain adaptation and the movement generation modules, these are minor changes that can be limited to some parameter modifications or message suppressing and these constructions are really easy to implement within the subsumption architecture.

We are free to enlarge the set of reflexes in the terrain adaptation module extending the capabilities of our robot to tackle with new obstacles in a completely transparent way for the movement generation module. Conversely, we can change the movement generation module (including new strategies to choose the gait, to turn, ...) without changing any reflex of the terrain adaptation module.

Once the whole controller is working, we can enlarge its capabilities. We have implemented some behaviors that are not useful when one of the two modules is working alone but that can improve the global performance when they are working together:

- A *Steps anticipation* behavior: When Genghis detects a step with its front whiskers, it raises its body on the side of the detected obstacle to facilitate climbing on it.
- A *Wall avoidance* behavior: Contact with the front whiskers slightly decrements the step length of legs on the opposite side of the robot. In this way, a prolongated contact that reveals the presence of an untraversable step, makes the robot turn to avoid it. If the contact is detected with both whiskers then the robot will move backwards.
- A *Cliff avoidance* behavior: If a front leg (in the advance direction of the robot) is unable to find a supporting surface after looking for it for a while, then the robot can assume that it is in front of a

cliff. Then it can use the direction control strategies to avoid falling down.

6 Results

In the next sections we give some data that can serve as an evaluation of the terrain adaptation module, the movement generation one and the whole controller.

6.1 Terrain Adaptation evaluation

We have confronted our robot to some isolated obstacles as a controlled test to measure its capacity to adapt to general terrain conditions. All robot's legs can be individually placed over an obstacle up to 8 cm and the robot is able to compensate for it maintain ground contact with all legs. Front and rear legs can find a supporting surface in 8 cm deep holes, while the middle legs can adapt to holes of 3 cm.

6.2 Movement Generation evaluation

The movement generation module described in section 4 can make Genghis advance at a top speed of 6.6 cm/s, which is really fast taking into account the robot's body length. This top speed is achieved using tripod gait. If we use slow gait, then the speed reduces to the third part (2.2 cm/s). These speeds are the same when the robot walks backwards.

The robot can make a 360 degrees turn-in-place in approximately 20 seconds.

6.3 Global evaluation

When we join the two modules as described in section 5 then the resulting gait is not as perfect as the gait generated by the movement generation module alone. This is due to the imperfect force sensors of the robot. For instance, Fig. 6 shows the resulting gait when the robot walks on flat terrain. The average

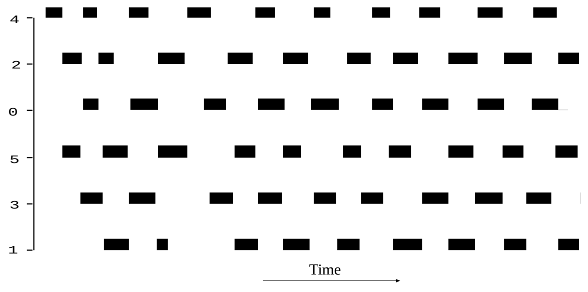

Figure 6: *A typical gait when walking on flat terrain. The shadowed areas indicate when the corresponding leg is stepping.*

number of legs that are stepping simultaneously and can be used as a quantitative measure of how far is the resulting gait from the tripod one. Using a perfect tripod gait this value should be 3 and with our controller, it is around 2.2.

The top speed of the global controller is about 3 cm/s. The reduction of the speed (compared with that of the movement generation module working alone) is due to an increase of the time to execute each step caused for the use by the Land reflex as a part of the step movement. The increment of the step time increases the time between consecutive steps and this reduces the speed (as we have explained in section 4.3).

If we confront the robot with an environment with small obstacles (up to 1.5 cm) then the average speed is slightly reduced to 2.5 cm/s. The same experiment repeated with 5 cm obstacles shows that the speed is reduced to 1.75 cm/s and that the average number of legs that steps simultaneously becomes 1.8.

As we can see the gait generation mechanism adapts to the terrain conditions increasing the stability by reducing the number of legs that step simultaneously. Nevertheless, even with large obstacles the resulting gait is efficient and the resulting advance speed is acceptable.

The robot is able to climb steps and to descent cliffs up to 8 cm. With some parameter modification this value could be increased but this would be at the cost of making less efficient other aspects of the task.

7 Conclusions and Future Work

We have presented a complete behavior-based controller for a six-legged robot that allows it to walk on irregular terrain and negotiate insurmountable obstacles as walls or cliffs. The main point of our work is the way in which the two components of the task of walking on rough terrain (the terrain adaptation and the movement generation) are separately solved an flexibly integrated. The resulting controller is both robust (it shows a good performance in several environment with really large obstacles) and extensible (either the terrain adaptation module or the movement generation one can be extended without affecting the other module).

As a future work, it seem reasonable to add a module to our controller that uses the simple mechanisms to modify the advance direction of the robot to drive it following a specific target.

Another task that seems interesting is to use learning techniques to find the set of parameters that characterize each motor of the robot. This will permit to develop better logical contact sensors based on the force readings of the motors and will improve the global performance of the controller. Additionally, self-adaptation mechanisms can be incorporated to the controller to change its behavior making it more suited for climbing steps or for walking on flat surfaces according to the terrain conditions or to the mission requirements.

Acknowledgments

This work has been partially supported by a grant from the *Comissionat per a Universitats i Recerca de la Generalitat the Catalunya* to the first author and by the Comisión Interministerial de Ciencia y Tecnología (CICYT), under the project "Subsymbolic techniques for constraint satisfaction, vision and robotics" (TAP93-0451). We also thank the assistance received from the HEROS EC HCM Network members to the development of this work.

References

[1] Beer R. D., Chiel H. J., Quinn R. D., Larsson P. (1992): "A Distributed Neural Network Architecture for Hexapod Robot Locomotion", *Neural Computation*, No. 4, pp. 356–365.

[2] Brooks, R.A. (1989): "A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network", *Neural Computation*, No. 1, pp. 253-262.

[3] Brooks, R.A. (1990): "The Behavior Language; User's Guide", MIT A.I. Memo 1227.

[4] Brooks, R. A. (1991): "Intelligence without representation", *Artificial Intelligence*, 47, pp. 139-159

[5] Celaya, E. and Porta, J.M. (1995): "Force-Based Control of a Six-Legged Robot on Abrupt Terrain Using the Subsumption Architecture", *Proc. of the 7th. Int. Conf. on Advanced Robotics* (ICAR'95), September 1995, pp. 413-419.

[6] Ferrell C. (1993): "Robust agent control of an autonomous robot with many sensors and actuators". Technical Report 1443, MIT AI Lab

[7] Gat, E. (1991): "ALFA: A Language for Programming Reactive Robotic Control Systems", In Proceedings of the

1991 IEEE *International Conference on Robotics and Automation*, Sacramento, California, pp. 1116-1120

[8] I. S. Robotics (1994): "The Genghis II Legged Robot Manual" February, 1994, v1.5.1

[9] Espenschied K. S., Quinn R. D., Beer R. D., Chiel H. J. (1996): "Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot" In *Robotics and Autonomous Systems*, 18, pp. 59-64

[10] Pearson, K. G., Franklin, R. (1984): "Characteristics of Leg Movements and Patterns of Coordination in Locusts Walking on Rough Terrain", *Int. Journal of Robotics Research*, Vol. 3, No. 2, pp. 101-112

[11] Porta, J.M. and Celaya, E. (1996): "The Behavior Language for IBM PC: User's guide", IRI Tech. Report., September-96

[12] Song S. M., Waldron K. J. (1987) "An analytical approach for Gait Study and its application on Wave Gait", *Int. Journal of Robotics Research*, Vol. 6, No. 2, pp. 60-71

[13] Wilson, D. M. (1966): "Insect Walking", Annual Rev. of Entomology,11, pp. 103-122