FORCE-BASED CONTROL OF A SIX-LEGGED ROBOT ON ABRUPT TERRAIN USING THE SUBSUMPTION ARCHITECTURE

Enric Celaya and Josep M. Porta
Institut de Cibernètica (UPC-CSIC),
Diagonal 647, 08028 Barcelona, SPAIN.
Phone: +34 3 401 6657; Fax: +34 3 401 6605;
e-mails: celaya@ic.upc.es, porta@ic.upc.es

October 16, 2000

Abstract

We have developed a controller for a six-legged robot that allows it to walk on rough or even abrupt terrain using only force sensors as feedback. In an initial approach, which at first seemed natural, we took a subsumption-based controller tailored to walk on flat surfaces and tried to add a force-compliance layer on top of it, thus maintaining the modularity of the design and building the controller in a purely incremental way, as proposed by Brooks in [1]. However, this approach could not be followed without making substantial modifications in the existing layers, in contradiction with our requirement of incrementality. Modularity was also lost, since lower levels of competence depended on the upper ones to work. A solution to these problems was found by redesigning the layer structure into a new one in which (compliant) walking corresponds to the uppermost level of competence, and in which a level for non-compliant walking has been ruled out.

1 INTRODUCTION

Locomotion control in legged robots is much more complex than in wheeled robots. While in the latter it is usually enough to control the velocity of two powered wheels to get a reasonable mobility, in legged locomotion, even for the simplest case, the issues of stability, gait generation, turning strategies and coordination of many degrees of freedom have to be addressed. Certainly, the use of legged locomotion in robots can only be justified by its superior capabilities to traverse rough terrain. Therefore, it is primordial for a legged robot to be able to walk with a certain degree of confidence not only on flat ground but also on uneven surfaces, otherwise there would be no reason for not using wheels.

When terrain conditions are not too bad, an acceptable walking performance may be obtained using a cyclic pattern of fixed movements of the legs, without any use of sensory feedback. Usually, this relatively simple approach already provides better mobility than wheels. However, when ground irregularities are important, a rigid

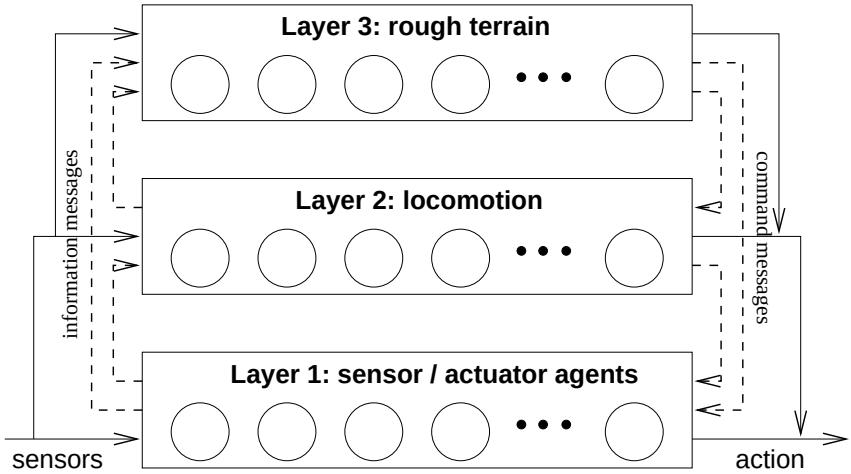

Figure 1: Layer scheme of the adaptive walk controller of C. Ferrell.

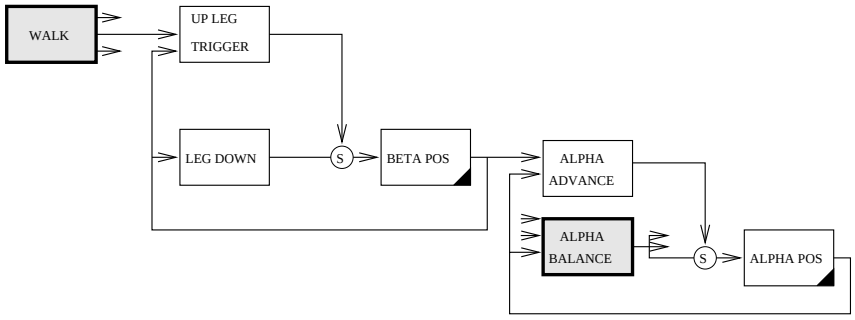

Figure 2: *Brooks' network for simple walk. All behaviors except the shaded ones are repeated six times, one for each leg. Motor interfaces are marked with a black triangle in its bottom right corner.*

walking pattern often produces false steps, sometimes leaving a leg on the air when it should be sustaining the robot and pushing it forwards, sometimes lifting a leg that is supporting much of the robot weight, which makes the robot fall down with undesired effects on the task being performed or on the robot itself. It is also frequent that the robot gets trapped when confronted with an obstacle, even if a way out would be easy to find.

In order to improve the performance of legged locomotion by adapting leg movements to different terrain conditions and situations, some sensing is necessary. Our purpose is to build a force-based adaptive walking controller for a legged robot. The platform for our experiments is a six-legged robot Genghis II (IS Robotics, Somerville, MA). Each leg has two degrees of freedom (d.o.f.), advance and lift, actuated by two motors that we denote as α_i and β_i, respectively, each one provided with its own force sensor. The robot has a number of other sensors of diverse types, but for the purposes of this work, only the force sensors are used.

2 THE SUBSUMPTION ARCHITECTURE

The subsumption architecture was introduced by Brooks [1] as an alternative approach to more classical robot control schemes. An important aspect of the subsumption architecture is that it favors a tight coupling between sensors and motors giving rise to reactive behaviors, instead of elaborated planning based strategies. Given that locomotion is a basic capability of most animals, it is natural trying to achieve it in a reactive way, rather that depending on too high level processes involving planning.

Perhaps the main promise of subsumption is modularity; that is, the possibility of building a robot controller bottom-up, by decomposing the task in levels of competence. Modularity means that each level can be completely programmed and tested on the robot, and has not to be modified when new levels are built on top of it, in a process often called "layering". In order to test the modularity of the subsumption architecture, we took an existing non-adaptive walk controller for the robot, and tried to reach a higher level of competence by adding an adaptive layer on top of it. A similar task was already done by Brooks [2], but with rather preliminary results. Subsequent work done by the same group [5] pursued more sophisticated adaptive walking with Hannibal, a rather more complex robot with 3-d.o.f. legs provided with force and position sensors at each joint, one global spine d.o.f., contact sensors in its feet, and a number of other sensors.

The approach followed by both Brooks and Ferrell was basically the same we follow: augmenting a non-adaptive walk controller with an adaptive layer (Fig.1). Since they reported a good modularity of the architecture, we expected our task to take place without too many complications.

3 EXTENDING A WALK CONTROLLER

Our departing non-adaptive walk controller was basically the same presented in [2]. It is written in Behavior Language (BL) [3], a language specially designed to implement the subsumption architecture, whose functional units are modules called *behaviors* able to share information through message passing.

In this controller (Fig. 2), the *walk* behavior sends a series of messages in a pre-specified sequence to each *up-leg trigger* behavior that forces the leg to be raised. The leg is then advanced by an *α-advance* behavior, whose task consists in moving the leg forwards whenever it has been raised above some threshold. Finally, the *leg-down* behavior, which continuously tries to keep the leg in its low position, returns the leg down completing the stepping movement. The advance movement of the robot is achieved by moving backwards the legs that are in contact with the ground. This task is accomplished by the *α-balance* behavior, which tries to maintain a balanced position on the horizontal plane for all legs. Thus, every time a leg steps forwards, the remaining ones compensate by moving backwards a short distance. Speed and gait may be both varied by just changing the pace of message sending and the sequence of legs established in *walk*.

The basic idea in our force-compliant walking approach consists in making each leg descend until its foot reaches the ground, and trying to keep it in contact all along the power stroke (the phase in which a leg sustains the robot and moves backwards pushing the robot forwards). This effect can be obtained according to the philosophy of subsumption by updating the position that *leg-down* tries to reach at any time: instead of having a fixed low position as the target, it is made to descend a little bit

lower whenever the force sensor detects that it is not touching the ground. In our implementation we replaced *leg-down* by the *land* behavior that sends relative, instead of absolute, motor commands.

The first thing we note is that, with this approach, the α-*advance* behavior cannot be unconditionally triggered whenever the leg is above some predefined position, since α-*advance* would move the leg forwards whenever its foot were placed on a high enough obstacle. The easiest way to solve this is to eliminate the α-*advance* and replace the *up-leg trigger* behavior by a *step* behavior that first lifts and then advances the leg.

A second problem comes from the fact that, since now legs have no predefined down position, the overall elevation of the robot may fluctuate out of control: consider for example the case in which the robot crosses a plateau (Fig. 3). As soon as a leg finds the elevation it will remain relatively raised during its power stroke, keeping the body in an invariant level position, but resulting in an inappropriate posture for walking. To correct this effect, we introduced a β-*balance* behavior, completely analog to α-*balance* except in that it works in the vertical direction. Thus, when legs are, on average, too raised or too lowered, β-*balance* will correct the situation by sending the same increment or decrement to all legs, with the net effect of moving the body vertically while preserving the same relative height differences between legs, which are necessary for the adaptation to the shape of the ground. An emergent effect of β-balance is that when a leg is raised to reach a higher position, the other legs will compensate by descending, thus raising the body of the robot, and helping the first leg reach even highly.

There is a third problem with the *walk* behavior in the adaptive case. Since the robot has no contact sensors on its feet, the only way we can detect ground contact is by slowly lowering each leg and checking the force in the motor, and stopping when the reaction force of the ground is detected. This makes the descent time of legs variable, and using a fixed stepping pace may result in raising one leg while the previous one is still on the air, thus causing the fall of the robot on its feet. A cautious walker that waits enough time to ensure that all other legs have reached the ground before raising a leg would be inefficient, since it would be uselessly waiting for nothing most of the times.

Apart from these considerations, observations of walking insects show that they do not follow a fixed gait pattern [7]. Instead, they use a family of wave gaits, or metacronal gaits [8, 6], in which rear legs step in opposition of phase, and legs at each side follow a back-to-front stepping wave. When controlling a single parameter (the stepping frequence of the rear legs), a wave gait continuously varies from a very stable slow gait, in which only one leg is stepping at any time, to the fastest tripod gait, in which three legs step simultaneously while the other three stay on the ground. The intermediate case in which exactly two legs are stepping at any time constitutes the parallelogram gait or ripple gait. Wave gaits provide a good balance between efficiency and stability, and allow the adaptation of speed to terrain conditions.

We implemented wave gate into our force-compliant control in the following way:

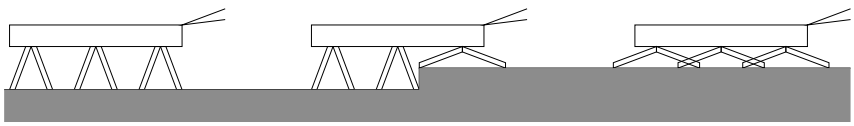

Figure 3: Effect of crossing a plateau when leg descent is controlled by ground contact.

Each leg has a *ready* behavior that, when active, waits for the neighbor legs to be on the ground (Fig. 4). When this happens, it sends a triggering signal to its corresponding *step* behavior as well as to the *ready* behavior of the next leg to put it into the active state. Rear legs send also the signal to its opposite leg (Fig. 5). Walking is initiated by activating the *ready* behavior of one of the rear legs, after which it is self-maintained. This completely distributed scheme is simpler than those proposed by R. Beer et col. [4] or C. Ferrell [5], and automatically adapts the walking speed to the maximum permitted by the accommodation of legs to ground irregularities.

As a last improvement, when a force is detected during the forward movement of a leg, a reflex of retreat and stepping higher is issued by the *skip* behavior in an attempt to get rid of the obstacle that caused the collision. This action is performed as many times as necessary, every time at a higher position, until the obstacle is cleared. The final version of the complete controller is schematically shown in Fig. 7.

4 REDEFINING LAYERS

As noted before, the initial approach we followed was to add a compliance layer on top of the walking layer. This approach seems at first natural, and is the same as the one followed by Brooks and by Ferrell in their works. However, by doing this we obtained a controller (that we will not describe in detail here) that, while performing in a more or less acceptable way, was difficult to understand and modify. In the course of adding the new layer we noted two things:

1. It was not possible to include adaptation in a purely incremental way. The original layers for non-adaptive walk needed to be changed in some important aspects. As explained above, some behaviors such as *walk* or α-*advance* had to be eliminated or redesigned.
2. In order to keep the simple walk level fully functional in the absence of the force-compliance layer, we were forced to design some specific modules for particular levels that had to be completely activated or deactivated depending on whether the upper layer was present or not. Thus, for example, the workings of the *land* and the *walk* behaviors were different for the compliant and non-compliant cases. Though it is possible, in principle, to design these behaviors within the subsumption architecture, the result is not at all natural. In other

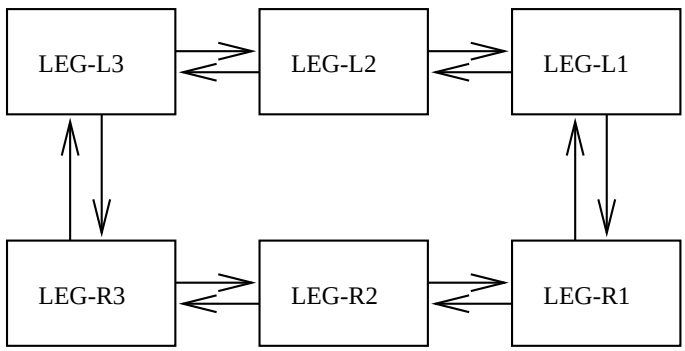

Figure 4: *Network of neighbor relationships inhibiting simultaneous stepping of legs.*

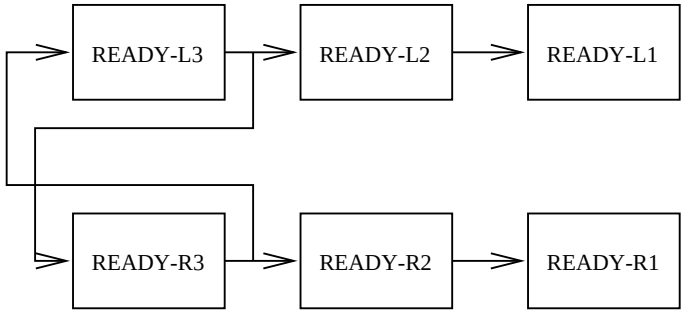

Figure 5: *Triggering scheme between* ready *behaviors.*

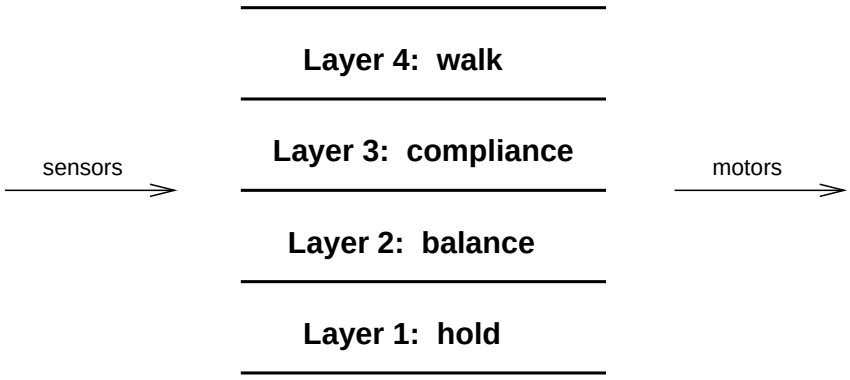

Figure 6: *Our new decomposition in layers of the force-compliant walk.*

words, the layered organization was not obtained as a feature of the architecture, but became a completely artificial construction enforced on this architecture.

A possible conclusion after these observations could be that the properties of modularity and incrementality of the subsumption architecture cannot always be maintained. However, a different conclusion is also possible: We could have begun with a wrong decomposition of the task in layers. In fact, we have been assuming without discussion that compliance corresponds to a higher level than walking, but this is not necessarily true. From this perspective we reconsidered the decomposition of the whole task in levels from scratch, and the result was that shown in Fig. 6, in which balance and compliance are both of a lower level than walk. This organization appears to be completely natural: For a robot to walk on rough terrain, the capability of keeping its feet on the ground and maintaining its stability is prior to, and more fundamental than that of advancing.

When we rewrote our code according to this new decomposition, the result was enlightening. In this case, layers could be built bottom up, and lower layers never had to be modified as a result of adding an upper layer, in contrast with our experience with the original decomposition. Furthermore, the connection of each new layer with the lower ones was clear and natural. The diagram shown in Fig. 7 clearly reflects this organization in layers. All four levels can be tested by themselves and all are entirely present as a necessary component of the higher ones. It is apparent that all the problems with modularity and incrementality that we found were due to the

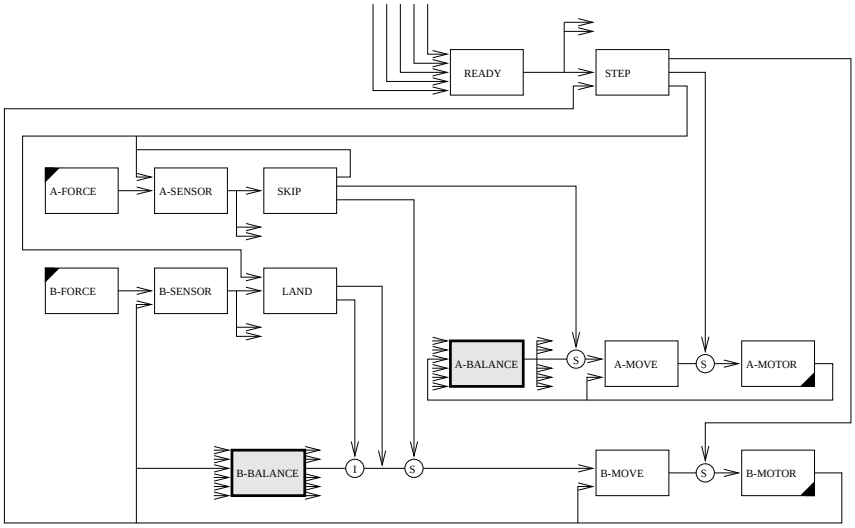

Figure 7: *Network of the complete adaptive controller. All behaviors, except A- and B-*balance *are repeated 6 times, one for each leg. Sensor and motor interfaces are indicated by a black triangle in their top left and right bottom corner, respectively.*

inappropriateness of the original decomposition of the task in levels.

If a moral is to be drawn from this experience, it should be that, while a robot controller can be built bottom up following the principles of the subsumption architecture according to an adequate task decomposition, the task decomposition itself cannot. It is necessary to consider the final highest level task in order to end up with a workable decomposition. This makes sense if we recognize that any given task could be decomposed in several different ways. Perhaps some decompositions of a lower level task are adequate for certain extensions but not for others. We just cannot take a particular task decomposition and expect it to work for whatever extensions we can imagine. In some way, we could say that level decomposition has to be done top down in order to allow a bottom up construction of complete robot controllers.

5 IMPLEMENTATION ISSUES AND RESULTS

Communication between behaviors in BL is performed by connecting an output port of one behavior to an input port of another behavior. Three mechanisms to modify these connections are provided:

1. Inhibition: Cancels any signal from an output port during a short period of time. In our controller a signal from the *land* behavior inhibits the corresponding output of the *β-balance* behavior to avoid any interferences while a leg is descending.
2. Suppression: Sends a message to an input port blocking any messages from other ports. It is used to establish priority between two incoming signals.
3. Default: Is similar to suppression, except that the priorities of the messages are interchanged.

What happens when two or more suppressing signals are connected to the same input port is ambiguous. For instance, in our controller there are three behaviors that send control messages to α-motor (*step*, *skip* and α-*balance*). *Step* has the highest priority and α-*balance* the lowest one. To establish three priority levels we use an implementation artifice consisting in splitting the motor interface into two behaviors, α-*move* and α-*motor*, which provides intermediate levels of input and output ports. The α-*move* behavior just forwards any incoming message to the α-*motor* behavior (in fact, in our implementation the inputs to α-*move* are increments that are converted into absolute positions before being sent to α-*motor*). The priority that *skip* has over α-*balance* is guaranteed by the suppression mechanism used in the input to α-*move* (Fig. 7). The highest priority of *step* is assured by suppressing the input to α-*motor*. A similar construction is used for the β-*motors*.

An interesting aspect of the controller is that there is no explicit behavior to back off a leg when it is overloaded (the basic force compliance mechanism used in [2]). An equivalent effect emerges from the combined action of *land* and β-*balance*. Since the weight of the robot is distributed between all legs, when the weight supported by one of them is too high, the weight supported by the other legs should decrease, eventually causing the *land* behavior to become active for some leg. This descending movement alters the average position of the legs, which is immediately corrected by β-*balance* by ordering all legs to ascend. Due to the inhibition mechanism, only the legs that are not being lowered by *land*, as is the case for the overloaded one, are actually affected by these ascending orders.

A problem observed in the actual performance of the robot, already reported in [2], is the following: when for any reason the front legs happen to be more raised than the rear ones, they become overloaded and, by the effect explained before, they tend to raise even more. Brooks tried to solve this problem using new sensorial information from a pitch inclinometer and inhibiting the relaxation of the front or rear legs in the appropriate circumstances. In our approach, we make no use of new sensorial information; instead, we just add a new component to β-*balance* that tries to keep the average height of the front legs equal to that of the rear legs. The effect of this action is to put the robot parallel to its supporting surface. We also included a third component into β-*balance* that does the same for legs on both sides of the robot. Note that the three components of β-*balance* are orthogonal; that is, the action of one of them has no influence on the others.

One of the most challenging implementation aspects is the interpretation of the force sensors output. This is so because what sensors measure is not really force but the difference between actual and commanded motor positions, and only when the leg is not moving can this reading be interpreted as force. On the other hand, since there are no position nor velocity sensors for the motors, we have no means to know whether the leg is moving or not. The problem then is to detect real forces without being confused by the fictitious force readings generated by the movement of a leg. There are two situations in which we need to detect forces while a leg is potentially moving:

- When *land* is descending a leg step by step, in order to detect ground contact.
- During the advance movements of a leg, to inform the *skip* behavior when a collision has happened.

Empirically we have found that short movements, as those generated in the first case, produce a burst of sensor readings a short time after the motor command is issued. For this situation we have implemented a β-sensor behavior that takes this into account in

order to correctly interpret forces. For long movements, as those corresponding to the second case, two separate bursts of sensor readings are detected that are presumably associated with the beginning and the end of the movement. The α-*sensor* has been built to deal with these situations.

With these sensors, the performance achieved by the adaptive walk controller is very robust, thus allowing the robot to walk easily over obstacles of a height of about 3/4 the leg length.

6 CONCLUSIONS

From the origins of the subsumption architecture, one of the most controversial questions raised around it concerns its scalability, that is, its potential to allow complex robot behavior, including such tasks as planning, map building, or manipulation of internal representations. In the example presented here, we have seen that even the most basic layers may become inappropriate for further improvements relatively soon. We think that this is not a limitation of the subsumption architecture but reflects the wide range of possibilities from which the designer should choose according to the final high level task to be performed.

If high level functionalities have to be reached some day, this will be only possible on the basis of principled and well-grounded lower level behaviors. We have presented a new layer structure for one of the first problems a mobile robot has to solve: locomotion using sensorial feedback. We hope that this work will serve to gain better understanding of this problem and that it will help to find the way to reach higher goals in the future.

ACKNOWLEDGEMENTS

This work has been partially supported by the Comisión Interministerial de Ciencia y Tecnología (CICYT), under the project "Subsymbolic techniques for constraint satisfaction, vision and robotics" (TAP93-0451) and by a grant from the Comissionat per a Universitats i Recerca de la Generalitat de Catalunya.

References

[1] Brooks, R.A. (1986):"A Robust Layered Control System for a Mobile Robot",*IEEE Journal of Robotics and Automation*, vol. RA-2, No. 1, pp. 14-23, March.

[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] Chiel, H.J., Beer, R.D., Quinn, R.D. and Espenschield, K.S. (1992): "Robustness of a Distributed Neural Network Controller for Locomotion in a Hexapod Robot", *IEEE Trans. on Robotics and Automation*, vol. 8, No. 3, June 1992, pp. 293-303.

[5] Ferrell C. (1993): "Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators", M.S. Thesis, MIT.

[6] Orin, D.E. (1982): "Supervisory Control of a Multilegged Robot", *Int. Journal of Robotics Research*, Vol. 1, No. 1, pp. 79-91.

[7] Pearson, K.G. and 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.

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