Implementation of a hierarchical walk controller for the LAURON III hexapod robot

Enric Celaya and José Luis Albarral
Institut de Robòtica i Informàtica Industrial (IRI), UPC-CSIC,c/ Llorens i Artigas 4-6, Barcelona, Spain.

SYNOPSIS

A control structure for a hexapod walking robot was previously developed at the IRI and implemented in simulation, as well as in a small robot with two degree of freedom legs. This same structure is now being implemented in a robot with three d.o.f. legs (LAURON III, from FZI). The problems found during the implementation in the real robot are summarized here. The main aspects that required special attention were the implementation of subsumption-like modules in MCA2, the native control architecture of the robot, and the difficulties to achieve the coordination of the movements of legs with the available actuators.

1 INTRODUCTION

The development of a generic interface for robot navigation, largely independent of the robot used, is the long-term goal of the project ARGOS currently in progress at the IRI (1). The navigation interface will allow the user to specify the navigation target by selecting it directly on images taken by a camera transported by the robot. The expected typical scenarios will be unstructured, previously unknown, outdoor environments, where the robot will need to avoid obstacles, negotiate terrain difficulties, and recover from navigation errors autonomously. When moving in natural, unprepared environments, the use of legged robots is preferable to wheeled ones due to their superior capability to overcome obstacles and deal with irregular terrain. Recently, we have acquired a six-legged robot LAURON III for this purpose (Fig. 1). LAURON III has been designed at FZI in Karlsruhe, and is the result of about ten years of progressive improvement on the previous designs, LAURON I and II. Each leg of the robot has three degrees of freedom, and its head holds a pant/tilt mechanism that allows orienting the camera in any direction. Each foot is provided with a three-axis force sensor, and each motor has a current sensor that can be used to detect forces opposing to its movement. The body has two inclinometers and two infrared proximity sensors.

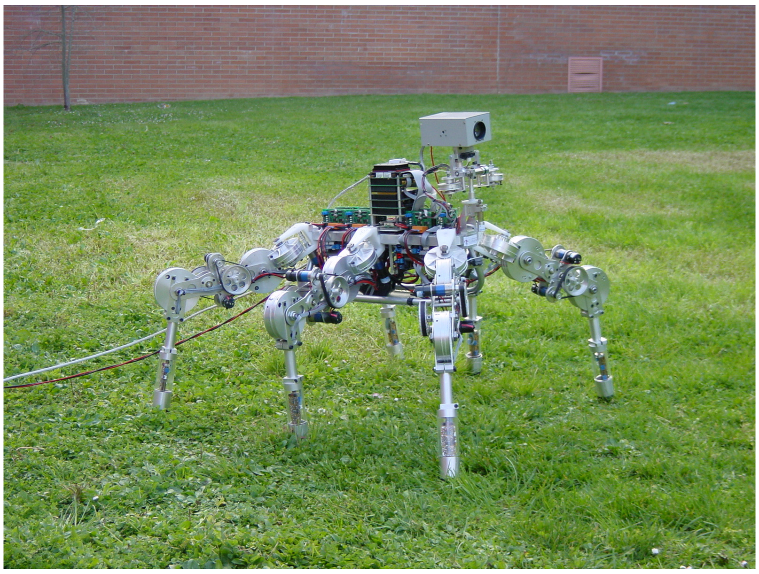

Fig. 1 The LAURON III robot.

Our first objective is to develop a “pilot” for the robot, i.e., a walk controller that adapts to the terrain profile and negotiates obstacles, able to respond to commands for direction changes in real time (instead of following a predefined trajectory computed off-line).

Algorithms for LAURON III to walk on unstructured terrain were developed at FZI, and tested on natural terrain with good performance (2). The general approach followed in that implementation was based on a centralized control: a cycle phase is centrally defined for the coordination of the movement of all legs, and individual leg trajectories are computed for each part of the leg cycle (transfer and support) and modified when a problem occurs, as for example, when a leg hits an obstacle.

In a previous work at the IRI, we developed a decentralized control structure for six-legged walking robots (3), which was actually implemented in a simpler robot (Genghis II), whose legs have only two d.o.f., so that not every aspect of the devised controller could be implemented on it. A further implementation was made in simulation for a hexapod with three-d.o.f. legs, but, until now, the controller has not been fully implemented on a real robot. With LAURON III, we have now the opportunity of implementing our control structure and compare it with a centralized one using the same robot—a comparison that is rarely found in the literature. To do this, we kept the low-level routines for sensor acquisition and motor control, and substituted the higher-level processes for the pilot according to our control structure. At this time, the implementation is only partially complete, so that a full comparison between both implementations is not possible yet. In this paper we just point at some specific aspects that needed to be addressed in the implementation on the real robot that had not been foreseen in simulation.

2 OUR CONTROL STRUCTURE FOR WALKING ON ROUGH TERRAIN

Our purpose is to implement a controller that allows the robot to walk reliably in rough terrain, following the direction commands given on-line by a human driver or by a higher level navigation process. We made a hierarchical decomposition of the task in several levels of competence as proposed by the Brooks' Subsumption Architecture (4). For the decomposition of this task, we have devised five different levels, as shown in Fig. 2, that are described next:

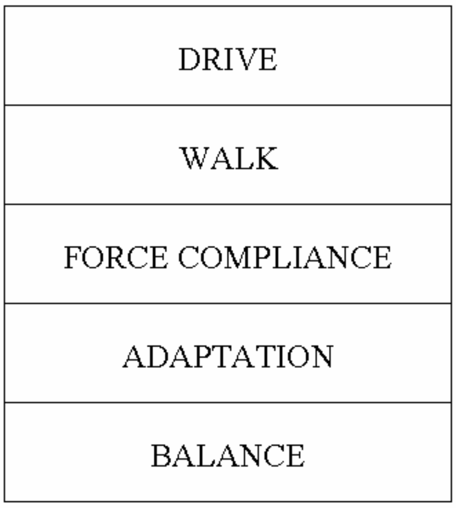

Fig. 2 Hierarchy of control layers.

The *balance level* keeps control of the body posture. Its purpose is to centre and orient the body according to the current feet positions, in order to keep the robot stability and improve leg mobility. This level plays a central role in the control structure, since it makes the body "follow" the legs when they move responding to commands issued by higher level behaviours. Once the *balance level* is active, only individual leg movements need to be considered by higher level behaviours, since appropriate body movements will be automatically issued. At this level, leg positions are never individually modified: all of them are moved in coordination in such a way that the distances between them stay constant, so that their positions with respect to the supporting ground remains fixed and only the body is moved.

The *balance level* is composed by six *balance behaviours*, one for each degree of freedom of the body, and each of them acts independently of the other five. Since individual leg movements produced by higher level behaviours may give rise to arbitrary feet configurations, the problem is to decide what should be the target position for the body in any conceivable feet configuration. For this, a reference position for each foot is defined, and a distance of a feet configuration to the reference position is defined as the sum of distances of the individual feet to their respective reference positions. The target position will be that for which this distance is minimized.

The *adaptation level* modifies the target position for the *balance level* according to the current conditions as detected by sensors. Note that the only information used at the *balance level* is proprioceptive, i.e., the position of the feet with respect to the body, so that external conditions do not influence it. The *adaptation level* allows to take into account external information in order to improve robot stability and manoeuvrability, as for example, raising the body when an obstacle is detected below, or displacing it forwards when the robot climbs up a ramp.

The *compliance level* grants that all legs in support phase keep ground contact, simply making a foot to descend when no vertical force is detected on it. This mechanism combined with the balance in the vertical direction makes the legs adapt to any ground profile. This level is also responsible for the usual reaction of “stepping higher” when a leg that is performing a long horizontal movement detects a collision.

The *walk level* is in charge of deciding when each leg must execute a step and start the transfer phase. The execution of a step at this level consists in just raising the leg and advancing it to the target position: the descent to the ground will be performed by the *compliance level*, and the advance of the body will be performed by the *balance level*. We do not make the robot follow a specific gait pattern: the decisions about when legs should make a step are taken locally at each leg, using information of the state of other legs. Thus, for each pair of neighbouring legs, the more advanced in its support phase will be the first to start its transfer phase. As a general rule to grant robot stability, a leg will only enter the transfer phase after their two neighbouring legs have finished their descent to the ground and are supporting the robot. This decentralized strategy produces a free gait that varies according to terrain conditions, and spontaneously generates a tripod gait when walking on flat terrain.

To improve the reliability of walking on difficult terrain, we avoid raising a leg when the robot is trying to solve a conflictive situation, as for example when a leg encountered an obstacle and is trying to find its way with the “step higher” reflex, which is an indication that terrain conditions are getting bad.

The *drive level* changes the advance direction according to the direction commands received from a higher navigation process or from the user. A direction command is summarized as the specification of the desired instantaneous turning centre in the X-Y plane of the robot body. Target positions for feet in transfer phase (usually known as AEP: the Anterior Extreme Position) are obtained applying to their respective reference positions, rotations of a given angle around the instantaneous turning centre. No body trajectory is computed, since the *balance level* will automatically drive the robot in the direction followed by individual legs.

3 SUBSUMPTION AND MCA2

The native control architecture of the robot LAURON III is MCA2 (5), in which most low-level tasks for sensor data acquisition and motor control are already implemented for the robot. MCA2 follows a classical control structure that implements a global “sense-think-act” cycle iterated at a given frequency. At each iteration, sensor data are gathered by the lower level modules (“sense” part of the cycle) and passed up to the higher levels for further processing until the top-most level is reached and action decisions are taken (“think” part). The control commands generated are then passed down towards the actuators (“act” part).

Levels in this architecture are composed of a number of basic processing units called *modules.* Each module includes a *sense( )* and a *controle( )* function, and communicates with other modules via four ports: *sensor in, sensor out*, *control in,* and *control out* (see Fig. 3), which can be connected to each other by means of *edges*.

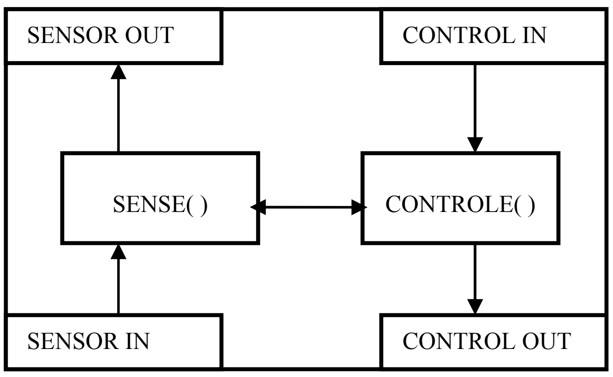

Fig. 3: Schematics of an MCA2 module

The "sense" part of the cycle iteration executes the *sense( )* function of each module, starting from the lowest level, up to the top-level. Data generated at *sensor out* ports are passed to the input ports connected to them by edges. Once the top level is reached, the "act" part of the cycle executes the *controle( )* function of modules in the reverse order, descending to the bottom level, and passing data through the edges connecting *control out* with *control in* ports.

The Subsumption Architecture resembles that of MCA2 in that modules (there called *behaviours*) are also organized in different levels, and data are passed between behaviours through communication lines connecting them. The difference is that, in the case of the Subsumption Architecture, each level executes its own complete sense-think-act cycle connecting sensors to actuators. This means that, at a given time, several behaviours can be sending conflicting commands to the same actuators, so that an arbitration mechanism is necessary. The way the Subsumption Architecture implements such arbitration is by means of suppression and inhibition connections, with pre-established priorities for their messages.

For the implementation of our Subsumption-based controller in the MCA2 architecture, the natural choice is to use MCA2 modules as behaviours. The only problem is that there is no mechanism allowing the definition of multiple connections to the same port with different priorities, which would be necessary to reproduce the suppression and inhibition mechanisms. To solve this lack we devised a standard procedure for message priority management, what allows the isolation of the priority management tasks from the pure behavioural control tasks. A priority management module is simply a regular module with many control (resp., sensor) input slots and a single control (resp., sensor) output (Fig. 4). Input port slots have a predetermined priority, so that if a message arrives to the first slot, it passes through the output and messages received in all other slots are ignored. If no message is received at the first slot, the same strategy is applied for second slot, and so on until the last slot.

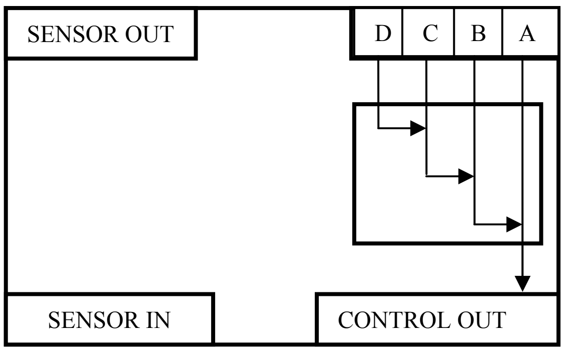

Fig. 4: Priority management module

An example will serve to explain its use (Fig. 5). We have three behaviours that may change a leg position at different levels in the hierarchy: A *balance behaviour,* the *ground contact behaviour* (which is part of the *force compliance level*), and a *step behaviour* that may be triggered by the *walk level*. The *step behaviour* must suppress the action of both *balance* and *ground contact* behaviours to move the leg to the new intended position. On the other side, *ground contact* must suppress the *balance behaviour* for this leg while it stays on the air. With the connections shown in Fig. 5, leg positions commanded by *ground contact* suppress the positions commanded by the *balance behaviour*, while the issue of a step suppresses both *ground contact* and *balance*, as required.

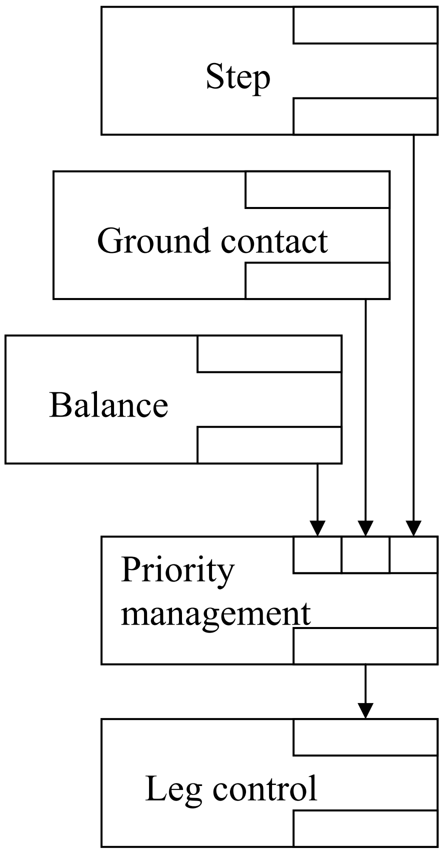

Fig. 5 Example with three priority levels.

4 LEG COORDINATION

Many legged robot designs find inspiration in insects. In the case of the LAURON III, its leg structure mimics that of the stick insect. However, the large scaling factor existing between the insect and the robot puts in evidence the well known law according to which the strength of a support scales with area, while the weight scales with volume: in consequence, whereas the insect is able to carry several times its own weight, the robot is only able to hold a fraction of it, and similarly, while the insect can climb easily in extremely difficult situations, the robot often experiences problems just to keep the body raised in non-optimal conditions.

Due to the relatively little powerful actuators of the robot, it is difficult, or impossible, following a commanded leg trajectory with a certain precision. While the precise execution of a leg movement in the transfer phase is relatively unimportant, movements of legs in the support phase must be carefully coordinated to avoid feet slippage or the generation of internal forces in the robot structure. Thus, to ensure leg coordination, precise computation of leg trajectories does not suffice; the progress of each leg must be monitored in order to avoid too large deviations. When a leg does not reach its intended position in due time, the other legs should slow down to wait for the delayed one.

In our case, the *balance level* is responsible of all leg movements that need coordination, and so, we devised a mechanism for leg coordination at this level. The way we designed the balance behaviours is that, at each time slice, only a small movement is done by each leg in the appropriate direction, in such a way that the inter-feet distances are kept constant. In a first approach, it seemed sensible to take the actual current position of each leg, and execute the desired small displacement from there. The problem with this approach is that actuators are not always able to reach the commanded position in due time, resulting in a delay that accumulates with successive iterations, with the result that legs tend to desynchronize very fast. To solve this problem, we computed the new position of each leg from the last commanded position, instead of from the actual one. In this way, when a leg is delayed for any reason, the new commanded position is always correct.

With this approach, the coordination mechanism must monitor the difference between the actual and the commanded position of each leg and try to keep this difference into an acceptable range. To do this, a convenient difference threshold is defined and, if some leg goes beyond this threshold, all other legs will wait for this leg before continuing its movement. In practice we have found that this threshold cannot be arbitrarily small, since the low level control does not react strongly enough to too small position differences, and the robot gets stuck when a leg stays near its intended position but is not able to reach it. One way to get out of this situation is to set the target position for the conflicting leg further away from its current position, what causes a stronger response of the low-level motor control that usually is enough to move the leg. However, this action is unsatisfactory because it may be the source of desynchronization with other legs. We are looking for alternative strategies to recover from such situation.

5 RESULTS

We have completed the implementation of the four lower levels of our control structure. With them, the robot is able to reliably walk on irregular terrain. A free gait is generated, which depends on the difficulties found by legs, but a fast convergence towards a tripod gait is observed when the robot advances on flat ground.

A comparison was made with the FZI implementation to test the effect of the control on leg coordination during posture changes. Little difference was observed in the execution of body translations along the three spatial directions. Where differences are significant is in rotations: with legs at the reference position, a rotation of the body around the Z axis from -15° to $+15^{\circ}$, produces a slippage of some feet on ground of 3 to 4 centimetres in the FZI implementation. The same action using our implementation with leg coordination produced feet displacements below 1 cm.

We expect to complete the implementation of the full controller in the near future and be able to further compare its performance in different navigation tasks.

ACKNOWLEDGEMENT

This work has been supported by the Dirección General de Investigación del Ministerio de Ciencia y Tecnología under the project "Autonomous robot navigation guided by visual targets" (DPI2000-1352-C02-01).

REFERENCES

[1] E. Celaya and C. Torras (2002): "Visual Navigation Outdoors: the ARGOS Project", Proc. of the 7th. *International Conference on Intelligent Autonomous Systems (IAS-7)*, Marina del Rey, USA, March 2002, pp. 63-67.

[2] Gaßmann B., Scholl K.-U., Berns K. (2001): "Behavior Control of LAURON III for Walking in Unstructured Terrain" CLAWAR 2001, *Int. Conference on Climbing and Walking Robots*, September 2001, Karlsruhe, Germany, pp. 651-658.

[3] E. Celaya and J.M. Porta (1998): "A Control Structure for the Locomotion of a Legged Robot on Difficult Terrain", *IEEE Robotics and Automation Magazine*, Vol. 5, No. 2, June 1998, pp. 43-51.

[4] 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 1986.

[5] Scholl K.-U., Albiez J., Gassmann B. (2001): "MCA – An Expandable Modular Controller Architecture", *3rd Real-Time Linux Workshop*, 2001, Milano, Italy.