Reactive Free-Gait Generation to Follow Arbitrary Trajectories with a Hexapod Robot

Article *in* Robotics and Autonomous Systems · July 2004

DOI: 10.1016/j.robot.2004.04.001 · Source: DBLP

2 authors:

Josep M. Porta
Spanish National Research Council
138 PUBLICATIONS **2,469** CITATIONS
SEE PROFILE

Enric Celaya
Spanish National Research Council
55 PUBLICATIONS **465** CITATIONS
SEE PROFILE

Reactive Free-Gait Generation to Follow Arbitrary Trajectories with a Hexapod Robot

J.M. Porta [*] E. Celaya

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens i Artigas 4-6, E-08028, Barcelona, Spain

Abstract

We present a reactive controller that is able to displace a legged robot along an arbitrary trajectory with a high degree of accuracy. We designed the different modules of our controller so that they can deal with arbitrary leg configurations. In this way, any leg movement necessary to overcome unexpected terrain irregularities can be correctly compensated by the controller, while still following the trajectory commanded by the user. Since we move the robot as a reaction to leg movements while stepping, the speed of the robot is automatically adjusted to the terrain profile: the more obstacles in the terrain, the more leg movements necessary to overcome them, and the slower the movement of the robot. We prove that, as the terrain becomes simpler, so does the gait generated by our controller, automatically converging to the tripod gait when the terrain becomes flat. This is achieved without requiring a map of the terrain and, thus, our controller can be used by robots with minimum computational and sensing capabilities. The results we report using different legged robots and in different environments prove the adequacy of our approach.

Key words: Legged Robots, Gait Generation, Posture Control, Heading Control, Reactive Control.

1 Introduction

To be used in real applications, a legged robot must be controllable in a way similar to that of a wheeled robot, that is, just by providing the desired advance direction of the robot. Thus, a walking machine, even if it is human driven, has to show autonomous behavior for advancing along a given trajectory.

Our purpose is to use the legged robots in abrupt outdoor areas (forests, volcanoes, mountains, etc.) that are the kind of environments where legged robots result advantageous with respect to wheeled ones. Following a classical approach, the task of legged-robot walking is confronted as an optimization process. The gait of the robot is selected optimizing for instance, the number of footholds necessary to achieve a given position [1,2], the size of the stability margin [3,4], the desired trajectory and speed [5], the mobility of the legs [6] or the energy consumed [7].

The combinatorial complexity of evaluat-

[*] Corresponding author: J.M. Porta, *tel* +34 93 4015791, *fax* +34 93 4015750, *e-mail:* porta@iri.upc.es

ing all valid sequences of leg movements to find the optimal one may be so great as to prohibit its computational solution in an acceptable period of time. For this reason, traditional legged robot controllers try to alleviate this process using heuristics to evaluate only the most promising sequences [8,9]. An alternative is to only allow the robot to operate in a restricted type of terrains: flat, locally planar, etc [5]. In this way, only specialized planning processes for each type of terrain have to be developed. Finally, another possible approximation to reduce the complexity of the planning process consists in identifying sequences of leg movements that are optimal for a certain kind of obstacles (steps, ditches,. . .). These sequences constitute a kind of toolbox from which to select the adequate components once the elements that constitute the environment are identified [10]. The drawback of this approach is that the robot can only face environments composed by predefined types of obstacles.

Most of the above-mentioned approaches rely on the assumption that a map of the environment is either available or built by the robot. However, natural environments are highly dynamic and the cost of acquiring and maintaining an accurate map of the terrain becomes too high. The alternative is to use a reactive control paradigm [11]. In this control approach, the sequence of leg movements is not planned but reactively generated as a result of the interaction between the robot and the environment. Reactive controllers do not use any kind of terrain map: the robot perceives the terrain profile by the direct contact of the legs with its irregularities as they are confronted.

The reactive control approach has been successfully applied to legged robot control before, using relatively small robots with two Degrees of Freedom (DoF) per leg [12], or three DoF per leg [13,14,15] or following a biological-based inspiration to design the reflexes included in the controller [16,17]. However, if we analyze these works, we observe that they are focused on straight line locomotion on mainly flat terrain and, as the confronted terrain becomes more irregular, there is almost no control on the exact trajectory followed by the robot [18]. This lack of control is only acceptable when using small robots with reduced weight and inertial effects. These robots are only useful for inspection tasks since they can not carry significant loads. A controller for larger legged robots should combine the accurate control provided by planning-based control approaches with the efficiency provided by reactive controllers.

In this paper, we develop a reactive controller for legged robots that is able to accurately follow a user provided trajectory even when walking on very abrupt terrain. This makes the controller useful for applications on real environments.

The key issue of our controller is that no assumption about leg positions is made. The different modules of the controller are designed to deal with arbitrary leg configurations at any moment. The result is a controller able to adapt to any leg movement necessary to overcome unforeseen obstacles, while still following the trajectory designated by the user.

The paper is organized as follows. In section 2, we formalize the tasks we confront. Sections 3 and 4 describe the two subtasks of our controller: the gait pattern generation and the heading control. Section 5 presents results of applying the controller to different simulated robots and some preliminary results with real legged robots. Finally, section 6 summarizes the work and presents the conclusions that can be drawn from it.

2 Problem Formalization

Legged robot locomotion is achieved by making successive steps and, at the same time, by moving legs in contact with the ground in the appropriate way so that the robot's body is displaced along the desired trajectory. Thus, the problem of legged robot locomotion can be clearly decomposed in two subtasks, that of executing steps and that of displacing the robot's body to follow a given trajectory.

For step execution, we have to determine both the movements to execute a single step (including the negotiation with obstacles detected while executing this step) and the sequence in which steps are issued (the so-called *gait pattern*). In this paper, we assume the existence of pre-defined leg reflexes for step execution (see for instance those described in [19]) and we concentrate in the gait pattern generation problem.

The movements (and, thus, the time) to execute a single step can largely change due to terrain irregularities. Additionally, the lack of valid points where to place a leg can make a step execution to move the leg to a position far away from the initially intended one. The result of all these factors is that the gait pattern generator can not assume legs to be in pre-defined positions. In other words, the gait pattern generator must be able to decide which steps must be executed next from any arbitrary arrangement of legs.

As far as the advance of the robot along the given trajectory is concerned we assume that the robot's body should be kept parallel to the local ground profile (leaving enough clearance with it). In contrast, the advance of the robot will be controlled through driving commands specifying the heading direction of the robot. Here, we restrict our analysis to heading commands defining arcs of circumference (figure 1), from which the case of a straight-line trajectory can be readily generalized by taking the limit of the equations that describe our heading controller when the turning radio r goes to ∞.

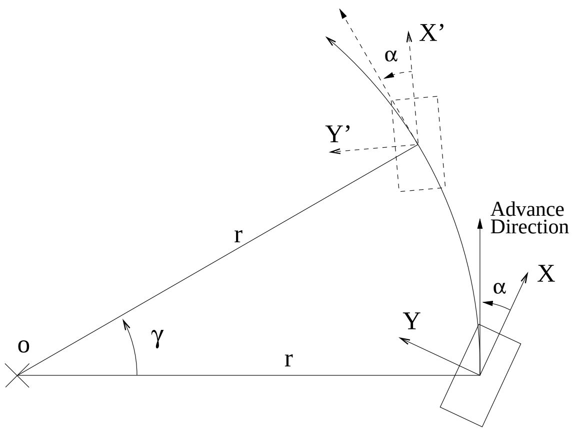


Fig. 1. The robot following a circular trajectory in its X-Y plane.

Thus, a heading command will specify the radius (r) of the arc of circumference to be followed at a given moment, in a way completely equivalent to how a car is driven by turning the steering wheel. If the wheel is turned counter-clockwise, the radius would be positive, and the turning center of the robot (o) would be at the left of the instantaneous velocity vector of the robot. Contrariwise, if the wheel is turned clockwise, the radius would be negative, and the turning center of the robot would be at the right of the velocity vector. Typically, the robot will move with its longitudinal axis (i.e., axis X) tangent to the trajectory and the turning center will be on the Y axis of the robot. However, we will consider also the general case in which there is an angle α (called the *crab angle*) from the positive X axis of the robot to its velocity vector.

Therefore, our controller will receive as input a heading command that is a pair (r, α) specifying both the turning radius and the crab angle, from which a turning center o in the $X - Y$ plane of the robot can be determined, which is given by:

$$o = \begin{pmatrix} o_x \\ o_y \end{pmatrix} = r \begin{pmatrix} -\sin\alpha \\ \cos\alpha \end{pmatrix}. \qquad (1)$$

The simple case where the turning center is located on the Y axis of the robot corresponds to $\alpha = 0$.

The heading command can be changed at any moment (and, thus, with legs in any possible position) and the controller must be able to properly react to these heading changes. Therefore, the controller must be able to move the robot along the desired trajectory even with legs in arbitrary positions.

So, to move on abrupt terrain following trajectories with arbitrary changes, both processes, that of gait pattern generation and that of heading control, have to be able to deal with arbitrary configurations of legs. In this way, these two subtasks become decoupled and they can be confronted separately. Next, we describe how we achieved these two subtasks in our controller.

3 Gait Pattern Generation

Periodic gaits can be obtained from very simple control rules. However, periodic gaits are only feasible for walking along regular trajectories (i.e., straight lines, fixed predefined arcs of circumference, etc). Since an arbitrary curve can be approximated by a sequence of arcs of circumference, a cyclic gait can be gradually modified to follow an irregular trajectory, provided direction changes are sufficiently smooth. However, when sharp direction changes are required (as, for example, when the heading of the robot is controlled by a human driver), the gradual adaptation of the gait cycle may become unfeasible, and a transition phase between one cyclic gait to the next may be necessary. In an extreme case in which sudden changes in the heading direction occur too often, the gait may never completely converge to a cyclic gait and it will become a free gait.

A similar problem appears with irregular terrain, where not all points of the ground are acceptable as foothold. In this case, a foot whose intended landing position turns out unreachable, or is not valid to support the load, will need to be landed in a different place, giving rise to gait perturbations and arbitrary leg configurations. So, for two of the most challenging problems in gait generation, omni-directional walking and adaptation to difficult terrain, the generation of a free gait is necessary.

The gait generation mechanism we next describe is reactive and, when required, it produces a free gait. A nice property of the mechanism is that, under certain conditions (i.e., regular terrain), it automatically generates a periodic gait. In other words, the gait generated by the robot emerges from the robot-terrain interaction instead of being specifically intended.

3.1 Requirements for Gait Generation

A gait generation system must ensure the stability of the robot and its ability to keep on advancing in next time slices.

To ensure stability, it must be granted that a sufficient set of legs stay on the ground supporting the body at any time. In the case of most six-legged robots, this requirement can be satisfied by observing the following rule:

Rule 1 *Never have two neighboring legs raised from the ground at the same time.*

Two legs are *neighbors* when they appear one next to the other in a closed circuit around the robot (figure 2).

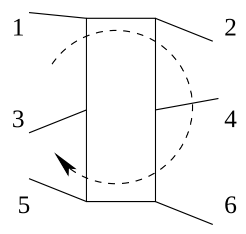


Fig. 2. Leg numbering and clockwise circuit.

In this work, we will assume rule 1 is a sufficient condition for stability. The fulfillment of this rule assures that, at any time, the robot will be supported by at least three non-neighboring legs, forming a triangle that we assume will always contain the vertical projection of the Center of Gravity, CoG This assumption might not hold in robots with very particular leg configurations or when the robot is too tilt. In any case, the violation of rule 1, however, will result in a situation in which two neighboring legs are out of the ground at the same time, most probably leaving the robot in a very unstable situation.

According to rule 1, a leg can be raised to make a step only while its two neighboring legs are in contact with the ground. However, rule 1 by itself is not enough to determine a gait. It leaves undetermined which one of a pair of neighboring legs should actually perform a step when rule 1 allows both of them to be raised. To deal with these situations, we have to assign priorities to the legs: the leg with a higher priority than its two neighboring legs would execute the step in the first place (provided that rule 1 is already fulfilled). Gait generation systems differ in the way in which they determine the priorities to step.

In some cases (specially in controllers based on central pattern generators [20,21,22]) priorities to step are based on time: legs that executed a step more recently have less priority to step again. In this way, the execution of steps of neighboring legs is alternated. This approach only works when walking on flat terrain and on a straight line since terrain irregularities or heading changes might force a given leg to execute a step twice in a row.

Other controllers [23,24] assign the priorities to step attending to the position of each foot: legs closer to the backward limit of the corresponding workspace have more priority. In this way, irregularities in the foothold positioning automatically result in alterations of the gait. The problem here is how to ensure that the alterations in the gait are properly compensated so that the robot is still able to produce a proper gait when the terrain irregularities are overcome.

In next sections, we analyze the gait that will be obtained from different priority assignments, independently of how these priorities are assigned. Using this general analysis, we introduce a gait pattern generation system with good properties such as the automatic convergence to tripod gait when walking on flat terrain.

3.2 General Analysis of Gait Pattern Generators

For two adjacent legs a and b, we will denote as

$$a < b$$

when leg b has more priority to step than leg a (i.e., it will execute a step earlier than leg a).

We define the *gait state* of a six-legged robot at a given time as the list of the six relationships as the ones above that can be established between neighboring legs. We will represent the gait state as a row of six symbols $<$ or $>$, corresponding to the relationships between priorities of neighbors taken in a clockwise circuit beginning in leg 1 taking the leg numbering of figure 2. The resulting order is: 1,2,4,6,5,3. Thus, for example, the state

$$<><><>$$

represents the following relationships between leg priorities

$$1 < 2 > 4 < 6 > 5 < 3 > 1.$$

The gait state is important because it determines the number of legs that can start a protraction at a given time: only legs that appear between a $<>$ pair are allowed to protract (note that, due to the cyclicity of the state list, leg 1 can protract when the state begins with $>$ and ends with $<$). According to this we can distinguish four *types of gait states*:

- **A**- Only one leg can protract.
- **B**- Two legs sharing a common neighbor can protract.
- **C**- Two legs not sharing a common neighbor can protract.
- **D**- Three legs can protract.

An important feature of the gait state is what we call the *clockwise circularity number* (CCN), defined as the number of $<$ relationships in the gait state.

The CCN can take any value between 1 and 5. Values 0 and 6, corresponding to the sequences $>>>>>>$ and $<<<<<<$, respectively, are inconsistent. The CCN defines a partition in the set of gait states. Table 1 shows all possible states, modulo cyclic permutations, classified by their CCN and gait state type.

It is well established [25] that the so-called *wave gaits* often observed in legged animals [26], constitute the most efficient and stable way to walk on a flat surface and, thus, our controller should be able to produce them. Wave gaits are characterized by

	Gait State Type			
CCN	**A**	**B**	**C**	**D**
1	<>>>>>			
2	<<>>>>	<><>>>	<>><>>	
3	<<<>>>	<<><>>	<<>><>	<><><>
4	<<<<>>	<<<><>	<<><<>	
5	<<<<<>			

Table 1
Gait states (modulo cyclic permutations) classified by type and CCN

a rear to front propagation of stepping actions forming two waves, one at each side of the body with the same frequency and in opposition of phase. Steps of adjacent legs are alternated and, thus, assuming that priorities to perform a step only change when a step is executed, after executing a step with leg a the following transition is performed in the gait state

$$\ldots < a > \ldots \Rightarrow \ldots > a < \ldots , \qquad (2)$$

The duty factor defined as the fraction of the gait period that a leg is in contact with the ground and generally denoted as β is used to classify the wave gaits. For each value of the duty factor between 1 and 1/2 we get a particular gait in the family of the wave gaits. Figure 3 shows the temporal representation for the case of $\beta = 5/6$, usually known as *slow wave gait.* In the figure, thick lines represent the protraction of each leg. Observe that the steps of rear legs are consistently delayed so that, as required by the slow wave gait, only one leg executes a step at a time. Again assuming that priorities to perform a step only change after a step is executed, we can compute the gait states for the wave gait. The right extreme of figure 3 shows the gait states resulting from this computation. The symbols on top of leg 2 correspond to its relationship with leg 1. From the figure, we see that, in this case, the CCN is 3. Figure 4 shows all possible transitions between gait states with $CCN = 3$ in all the cases. The labels in the arcs indicate the number of legs that have to execute a step simultaneously to pro-

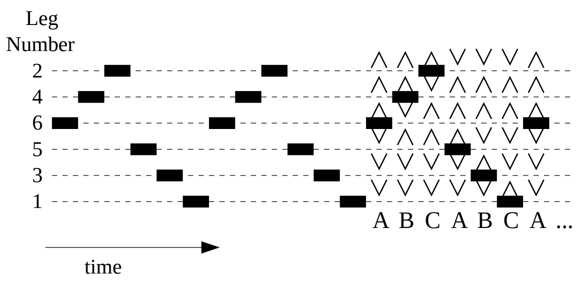


Fig. 3. Slow wave gait.

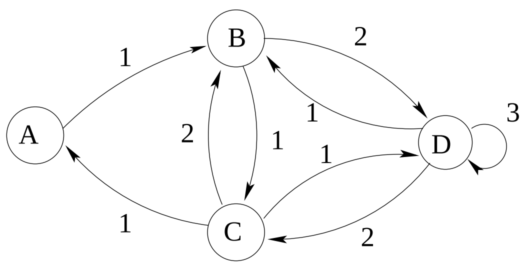


Fig. 4. Possible transitions between states with $CCN = 3$. The letters in the graph nodes represent the gait state type.

duce the corresponding transition. Analyzing the gait states at the right of figure 3, we can see that, as indicated in the bottom line of the figure, when the wave gait is generated, the robot follows a sequence of gait states with $CCN = 3$ and gait state types $ABC \ldots ABC$. The same sequence is obtained for all wave gaits with duty factors between 1 and 3/4.

For the special case of $\beta = 3/4$, in which legs 1,6 and 2,5 protract simultaneously and leg 3 and 4 step alone, the state sequence reduces to $BC \ldots BC$.

A similar analysis for β between 3/4 and $1/2$, shows that the corresponding sequence in this case is $BCD \ldots BCD$.

In the extreme case of the tripod gait with $\beta = 1/2$, in which three legs (1,4,5 and 2,3,6) protract at the same time, the sequence collapses to $D \ldots D$.

From this analysis, we conclude that all wave gaits have a $CCN = 3$. In other words, the complete family of wave gaits can be obtained from a proper initial gait state with $CCN = 3$ by alternating steps of adjacent legs, and with the appropriate delays in the steps of the rear legs. Since in wave gaits, neighboring legs always alternate their steps and since the alternation of steps of neighboring legs keep the CCN constant (see equation 2), only an initialization of legs with a $CCN = 3$ would allow the robot to execute a wave gait. With a CCN different from 3 a wave gait can only be achieved if the basic rules of step execution (alternation between steps of neighboring legs, rear to front step execution, etc) are somehow violated.

The transitions of any wave gait is a subgraph of that presented in figure 4. Thus, we can smoothly change from one wave gait to another by simply taking the desired branch from a node common to the two subgraphs of the corresponding wave gaits.

Even more, it is possible to show that, independently of the initial assignment of step priorities and whether or not legs are initially in contact with the ground, under the following conditions

(1) the robot is in a gait state with $CCN = 3$
(2) legs on the air try to recover contact with ground as fast as possible,
(3) all legs spend the same amount of time in the transfer phase,
(4) legs start their step as soon as possible,

the gait of the robot will soon converge to the tripod gait (an alternation of gait states $<><><>$ and $><><><$). Observe that all those conditions are likely to be hold when the robot walks on flat terrain. To prove this property, we only have to analyze the different gait state sequences from all the possible situations with $CCN = 3$. Table 2 shows this analysis for a gait state with $CCN = 3$ and gait type B and with one leg initially not in contact with the

T	Gait State Type	Type	CCN	Events
1	$a < b < c > d < e > f > a$ ● ● ● ● ● ○ ●	B	3	Initial Gait State
2	$a < b < c > d < e > f > a$ ● ● ○ ● ● ○ ●	B	3	c Starts a step
3	$a < b < c > d < e > f > a$ ● ● ○ ● ● ● ●	B	3	f Contacts ground
4	$a < b < c > d < e > f > a$ ● ● ○ ● ○ ● ●	B	3	e Starts a step
5	$a < b > c < d < e > f > a$ ● ● ● ● ○ ● ●	C	3	c Contacts ground
6	$a < b > c < d < e > f > a$ ● ○ ● ● ○ ● ●	C	3	b Starts a step
7	$a < b > c < d > e < f > a$ ● ○ ● ● ● ● ●	D	3	e Contacts ground
8	$a < b > c < d > e < f > a$ ● ○ ● ○ ● ○ ●	D	3	d, f Start step
9	$a > b < c < d > e < f > a$ ● ● ● ○ ● ○ ●	B	3	b Contacts ground
10	$a > b < c > d < e > f < a$ ● ● ● ● ● ● ●	D	3	d, f Contact ground
11	$a > b < c > d < e > f < a$ ○ ● ○ ● ○ ● ○	D	3	a, c, e Start step
12	$a < b > c < d > e < f > a$ ● ● ● ● ● ● ●	D	3	a, c, e Contact ground
13	$a < b > c < d > e < f > a$ ● ○ ● ○ ● ○ ●	D	3	b, d, f Start step
14	$a > b < c > d < e > f < a$ ● ● ● ● ● ● ●	D	3	b, d, f Contact ground
15	$a > b < c > d < e > f < a$ ○ ● ○ ● ○ ● ○	D	3	a, c, e Start step

Table 2
Gait evolution toward the tripod gait. We use the symbol ● for legs touching the ground and ○ for legs in the air.

ground. As it can be seen, the tripod gait is readily achieved. A similar reasoning can be done for all other initial situations showing that, in all the cases, the tripod gait is achieved.

So, it is clear that the preferred gait state at any moment is one with $CCN = 3$. In other words, independently of how we assign priorities to step, we must ensure that they conform a gait state with $CCN = 3$.

3.3 The Gait Generation Mechanism

In our controller, we determine the gait state in the following way. First, if a leg a is not in contact with the ground the subsequence $< a >$ is included in the gait state. Next, if neighbouring legs a and b are both on the ground and leg a is more advanced than leg b, then the sequence $a < b$ is included in the gait state meaning that leg b must step first to avoid it to reach the backward limit of its workspace, blocking the robot. When the robot is moving along a straight line, the advance position for each feet is readily obtained. However, in a general case, we have to measure the advance position of each leg w.r.t. the current turning center or, equivalently, w.r.t. the arc of circumference that, at a given moment, defines the trajectory. To evaluate this advance position, we compute the angle δ between the projection in the $X-Y$ plane of the current feet positions and the center of the corresponding workspace. This angle is measured from the current turning center and its sign is set attending to the direction along the trajectory we want the robot to move. If for legs a and b we have that $\delta_a < \delta_b$, then $a < b$ is introduced in the gait state. In the case of ties, we favor legs 1, 4, and 5.

The priority assignment using the above simple rules can lead to a gait state with CCN different from 3. Consequently, we have to adjust the CCN. We achieve this with the following *CCN adjustment* procedure: we increase the CCN (if it is below 3) by applying the conversion rule

$$\ldots >>< \ldots \Rightarrow \ldots ><< \ldots$$

and, if the initial CCN is above 3, we use the rule

$$\ldots ><< \ldots \Rightarrow \ldots >>< \ldots$$

Using table 1, we can see that these adjustment rules can always be applied as many times as necessary transforming any gait state with a CCN different from 3 into one with CCN equal to 3, but without modifying the gait state type (the columns in table 1).

Observe that, when we set the gait state, we favor the steps of those legs that are in a more critical situation (i.e., near its workspace limits) and, since the CCN adjustment never affects these more critical legs (it never modifies subsequences $\ldots <> \ldots$ in the gait state), the most urgent steps are always issued first.

The gait state determination should be applied at every time slice. In normal circumstances, we will observe an alternation between steps of neighboring legs, but if a leg has to be landed in a very unusual place (in a more backward position that one of its neighbors) then we will observe an automatic alteration in the gait. In the same way, if a new heading command is issued the gait controller automatically changes the gait state (and, thus, the sequence of issued steps) since changes in the turning center result in changes in the δ's used to determine the priorities to step. In any case, our controller retains the property that, if flat terrain is eventually achieved, the tripod gait will automatically emerge.

4 Heading Control

Up to this point we have described the way to determine when legs start a step. The problem now is how to coordinate the movements performed by legs in support and legs in return phase so that the robot is moved along the desired trajectory. This is essentially the problem of body and leg movement coordination, which has been recognized as a challenging problem in legged robot control [27], especially when terrain irregularities can force legs in return phase to be moved to unforeseen positions. Next, we propose a general, though simple, solution to this problem.

To drive the robot along the commanded trajectory, all legs in support phase must move along arcs of circumference centered at the turning center o. The landing position after a step (or Anterior Extreme Position, AEP) allows to choose the position of this arc for each leg.

The AEP could be determined according to multiple criteria. We adopt the one in [5] which consists in making each leg reach the central position of its workspace in the middle of its expected travel between consecutive steps.

Formally, if $c^i = (c^i_x, c^i_y, c^i_z)$ is the central position of the workspace of leg i, the intended AEP for this leg is

$$AEP^i = M_{r,\alpha}(\gamma)\, c^i, \qquad (3)$$

with

$$M_{r,\alpha}(\gamma) = R_z(\alpha) T_y(r) R_z(\gamma) \\ T_y(-r) R_z(-\alpha), \qquad (4)$$

and where $T_y(\cdot)$ is a translation along the Y axis of the reference frame of the body and $R_z(\cdot)$ a rotation about its Z axis, and γ is half the step size. The matrix $M_{r,\alpha}$ corresponds to a rotation of angle γ around the current turning center o. Note that the Z component of the AEP is given only as a reference, since in any case, it is completely determined by the ground elevation at this point.

This procedure provides only an intended AEP since actually placing the foot at it may be forbidden by terrain conditions. In this case, a different point should be searched for in the vicinity of the intended AEP by the leg reflexes we assume to be present in the controller.

No matter whether a foot is placed in the desired AEP or not, all feet in contact with the ground have to be moved in a coordinated way: without modifying the distances between them. Thus, positions of feet $P = \{p^i \mid i \in [1, n]\}$ (with $p^i = (p^i_x, p^i_y, p^i_z)$ and n the number of legs of the robot) can be moved to positions q^i using a spatial rigid transformation

$$q^i(\Omega) = q^i(x, y, z, \phi, \theta, \psi) = T_z(z)T_y(y)T_x(x)R_z(\phi)R_y(\theta)R_x(\psi)\, p^i, \tag{5}$$

where $T_x(\cdot)$, $T_y(\cdot)$, $T_z(\cdot)$, $R_x(\cdot)$, $R_y(\cdot)$ and $R_z(\cdot)$ are translations along, and rotations about, the corresponding axes of the reference frame of the body. The robot's body can be moved in six DoF just using the appropriate set of parameters Ω. Translations along Z and rotations about X and Y have to be used to keep the robot parallel to the ground profile. Movements along the trajectory defined in the $X - Y$ plane are performed using translations along X and Y and rotations about Z. The movements in these three dimensions are linked into a single DoF: the displacement of the robot along the arc of circumference traced from the current turning center. The question is to find the best position for the robot's body along that arc for an arbitrary set of leg positions. We define a criterion to evaluate all possible body positions on the trajectory taking into account leg positions. Then, the robot has to be moved to the point that optimizes the criterion at every moment.

In [19], we introduced a way to evaluate the position of the robot's body according to the distance between feet and the central positions of leg workspaces (that are at a fixed position w.r.t. the robot's body). The minimization of this criterion increases the (average) mobility of the robot since legs have maximum mobility when they are as far away as possible from the limits of their workspaces or, what is the same, in the center of their workspaces.

Formally, if $Q = \{q^i \mid i \in [1, n]\}$, is the set of legs positions w.r.t. the body and $C = \{c^i \mid i \in [1, n]\}$ represents the center of the leg workspaces, then

$$E_{Q,C} = \sum_{i=1}^{n} \|q^i - c^i\|^2, \tag{6}$$

is the *quadratic error* between Q and C.

Using equation 5, we can find the components of the gradient vector of the quadratic error function in posture P (i.e., Q with

$\Omega = 0$):

$$
\begin{aligned}
\left.\frac{\partial E_{Q,C}}{\partial x}\right|_{\Omega=0} &= 2\sum_{i=1}^{n}(p_x^i - c_x^i)\\
\left.\frac{\partial E_{Q,C}}{\partial y}\right|_{\Omega=0} &= 2\sum_{i=1}^{n}(p_y^i - c_y^i)\\
\left.\frac{\partial E_{Q,C}}{\partial z}\right|_{\Omega=0} &= 2\sum_{i=1}^{n}(p_z^i - c_z^i)\\
\left.\frac{\partial E_{Q,C}}{\partial \psi}\right|_{\Omega=0} &= 2\sum_{i=1}^{n}(c_y^i p_z^i - c_z^i p_y^i)\\
\left.\frac{\partial E_{Q,C}}{\partial \theta}\right|_{\Omega=0} &= 2\sum_{i=1}^{n}(c_z^i p_x^i - c_x^i p_z^i)\\
\left.\frac{\partial E_{Q,C}}{\partial \phi}\right|_{\Omega=0} &= 2\sum_{i=1}^{n}(c_x^i p_y^i - c_y^i p_x^i)
\end{aligned}
\tag{7}
$$

For the optimal posture, the six gradient components of equation 7 must vanish. An iterative gradient-descent process can be used to reach this optimal posture (i.e., a relative minimum of the quadratic error function).

The posture control mechanism just described keeps the robot body parallel to the terrain below the robot (as desired) but provides a too coarse way to move the body as a reaction to leg movements: there is no explicit attempt to follow a specific trajectory with the body, it just follows the legs wherever they move. In that framework, heading control can only be done through the selection of landing positions for the different steps, and the trajectory becomes subject to disturbances caused by modifications in the actual footholds imposed by terrain conditions and other perturbing effects such as the exact order at which steps are performed. When following a given trajectory, the movement of the body on its $X - Y$ plane is restricted by the current heading command and, so, what we have to use in this case is a restricted form of the posture optimization mechanism just described.

As mentioned, to propel the body along a given trajectory, legs in support phase must be moved along arcs of circumference centered at the current turning center. The relation between the positions of leg i before (p_i) and after (q_i) a rotation around the turning center can be expressed as

$$q^i(\gamma) = M_{r,\alpha}(\gamma)p^i, \tag{8}$$

where $M_{r,\alpha}(\gamma)$ is the rigid transformation defined in equation 4.

Using equations 6 and 8, we can find the derivative of the quadratic error function in posture P (i.e., Q with $\gamma = 0$):

$$
\begin{aligned}
\left.\frac{\partial E_{Q,C}}{\partial \gamma}\right|_{\gamma=0} =& 2r\cos\alpha\sum_{i=1}^{n}(p_x^i - c_x^i)+\\
&+ 2r\sin\alpha\sum_{i=1}^{n}(p_y^i - c_y^i)+\\
&+ 2\sum_{i=1}^{n}(c_x^i p_y^i - c_y^i p_x^i).
\end{aligned}
\tag{9}
$$

This gradient factor replaces the gradient components w.r.t. x, y, and ϕ of equation 7. Therefore, the minimum $E_{Q,C}$ value on the trajectory is found by making the derivatives w.r.t. γ, z, ψ, and θ decrease to zero. For any arbitrary disposition of legs, we can compute the value of these derivatives and, attending to their sign, apply a small rigid transformation ($M_{r,\alpha}$, T_z, R_x, and R_y respectively) to all legs in support phase until the corresponding derivative vanishes. At this point the robot is at the best position attainable along the trajectory for a given set of leg positions. As mentioned, translations along Z and rotations around X and Y keep the robot parallel to the local ground profile and movements along γ smoothly displace the robot along the trajectory.

The fact that the proposed posture control

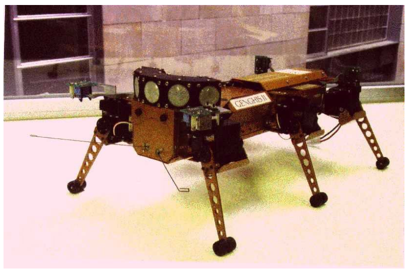


Fig. 5. The Genghis II robot.

mechanism is a particularization of a more general one can be used to get out of possible deadlock situations. A deadlock can appear when terrain conditions force an excessive displacement of footholds from the intended AEP's, so that some legs reach their workspace limit when in support phase. In this case, the only way to keep on advancing is by altering the trajectory in some way. The application of the trajectory-free posture optimization mechanism summarized in equation 7 provides a good choice to produce such a trajectory modification, since its effect will be an increase in the global mobility of the robot.

5 Results

To obtain results with a real robot, we used a Genghis II robot [28] (see figure 5). Genghis II's body is about 40 cm long and 15 cm wide. Each leg is approximately 10 cm long and has two DOF: a rotation around a vertical axis fixed to the body and another around a non-fixed horizontal axis. When legs are completely vertical, body clearance is about 8 cm. The robot is provided with force sensors at each joint (actually, it is the current used by each motor that is measured), contact/force sensors all along the lower part of the body, two frontal whiskers to detect contacts, one pitch inclinometer, four infrared sensors, and a set of five pyro sensors. Genghis is adequate for testing the performance of the gait pattern generation module of the controller (described in section 3) since this module is just related with the sequence at which step are execute and not with the exact movements of each leg to perform those steps nor with the particular structure of the legs. We will later see that this is not the case for the heading control module (described in section 4).

The controller is programmed in PCBL [29] a modification of the BL programming language [30] that was developed to facilitate the implementation of behavior-based controllers according to the main guidelines of the subsumption architecture.

5.1 Gait Generation Test

Tests of the controller including the movement generation module showed that the average speed of Genghis on smooth ground is about $5cm/s$, progressively decreasing as the difficulty of the terrain is increased. The limiting factor for speed is the slow movement imposed on leg descent in order to reliably detect ground contacts with the noisy force sensors of the robot's motors.

Using our controller, Genghis is able to climb up and down vertical steps of more than 10 cm, which is about the leg length and 2 cm more than the maximum body-ground clearance.

Tests in general terrain with all kinds of irregularities showed the ability of the robot to negotiate virtually all kind of difficulties, getting stuck only on some rare occasions in which a foot gets trapped in a narrow cavity.

Figure 6-A shows the achieved advance due to the first steps of the robot when using a simple controller that generates the tripod gait. As mentioned, this is the optimal gait when walking on flat terrain. This

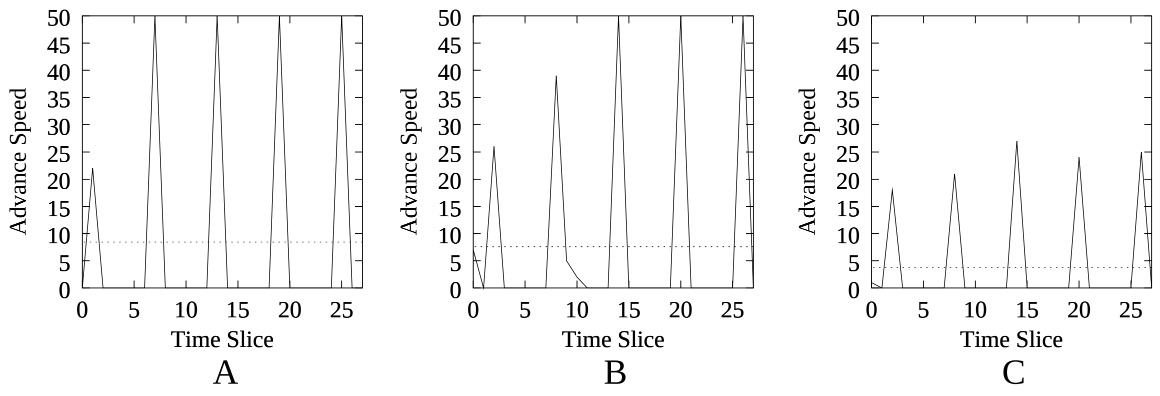


Fig. 6. Advance speed at the first steps using the tripod gait (A), using our gait generation mechanism (B), and using the same controller but without the CNN adjustment mechanism (C). The dashed lines represent the average speed.

controller waits until all legs are in contact with the ground and then it issues alternately steps with legs $(1, 4, 5)$ and with legs $(2, 3, 6)$. The result shown in the figure corresponds to a single execution departing from a random initial posture. As it can be seen, every time a tripod is issued, the robot advances 50 units and, in average (the dashed line in the figure), the robot advances approximately 8 units per time slice.

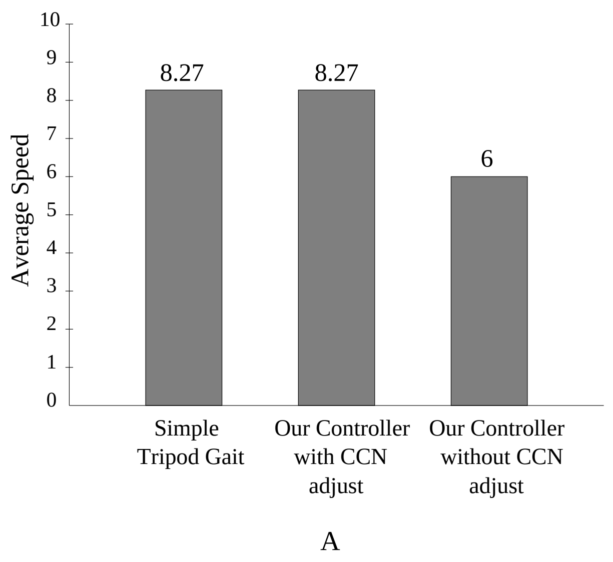
Figure 6-B, shows the results obtained in a similar experiment but using our gait generation mechanism. We can see that, after a few time slices, the result is that of the tripod gait controller.

To appreciate the effect of the CCN adjustment mechanism, we show the first steps issued by our gait generation controller but without the CCN adjustment. Figure 6-C, shows the results obtained by departing from an initial gait state with $CCN = 2$. In this case, the robot can not issue more than 2 steps simultaneously and, consequently, the average speed decreases.

To perform long series of experiments in order to get statistically significant results, we implemented different gait generation mechanisms using a simple 2D simulation of the Genghis robot. In this simulation, the execution of a step consists of two phases: one to raise and advance the leg and another to descend the leg until it touches the ground. The first phase of the step takes one time slice and the second one takes five time slices. The simulated robot is initialized with all legs in contact with the ground but in a random advance position.

Figure 7-A, shows the comparison of the average speed of the three controllers (the one performing the tripod gait, our controller, and our controller but without the CCN adjustment mechanism) departing from 100 initial postures generated at random and running for 500 time slices. Our gait generation mechanism performs at the same level as the tripod one but only when the CCN adjustment is performed.

The advantage of our gait generation mechanism over the one that implements the tripod gait is clear when walking over irregular surfaces. In this case, each leg takes a variable time to find a foothold and the tripod gait is not always the optimal one. The tripod gait controller must wait until all legs are in contact with the ground before starting a new tripod. In contrast, our controller issues a step with a given leg as soon as its two neighboring legs reach the ground. Consequently, there are fewer delays in the beginning of the steps and the average speed is increased. The magnitude of this increment depends on the particular



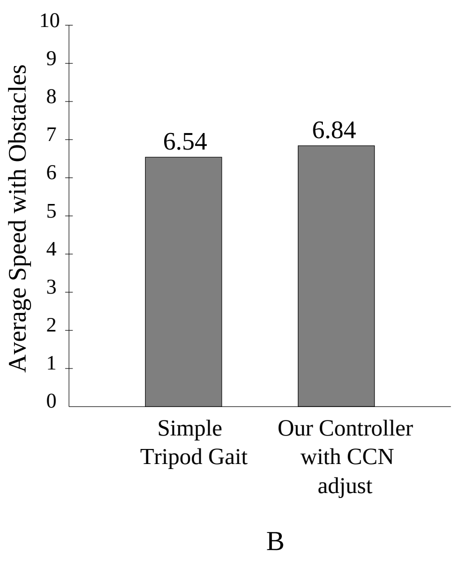


Fig. 7. Average speed when walking on flat terrain (A) and walking on abrupt terrain (B), with different gait generation mechanisms.

terrain configuration but, in any case, the performance of our gait generation mechanism is never below that of the tripod gait controller. Figure 7-B, shows the average speed using the two controllers departing form 100 randomly generated initial postures and executing each controller during 500 time slices. The terrain profile for each experiment is generated at random.

5.2 Heading Control Test

In the Genghis robot, the heading can be approximately controlled by setting opposite strokes on legs of both sides. In our experiments, the robot turns around at a speed of about $8°/s$. Trajectories with frequent heading changes could be executed without any problem. The controller also provided successful results when we use it in a simple navigation task (to avoid collisions with walls and to scape from cliffs).

The Genghis robot only has two DoF per leg and, to correctly follow an arbitrary trajectory, the robot must have 3 independent DoF per leg. To test the full controller as presented in this paper, we used a simulator of the Argos robot, currently in de-

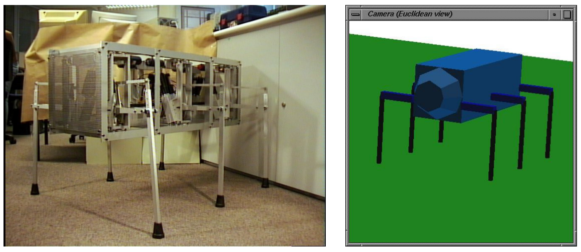


Fig. 8. The Argos robot and its simulation.

velopment in our institute (see figure 8). Argos will have six legs, each with three independent DoF. Its total length will be about 1 m and its weight about 50 Kg including batteries. It will be provided with force and contact sensors in each leg. In our simulator [31], we can define different ground shapes and robots with 3 DoF per leg, with variable number of legs, and even with different leg arrangements. The simulation of the Argos robot can be seen in figure 8. In all the cases, the robot has no information about the terrain configuration and can only sense it via the contact sensors of the legs (as it would occur in a real application).

Figure 9 shows Argos executing different heading commands. First (snapshot *A*) it follows a straight trajectory, then (snapshots *B* and *C*) it executes a turn in place, next (snapshot *D*) it walks straight ahead

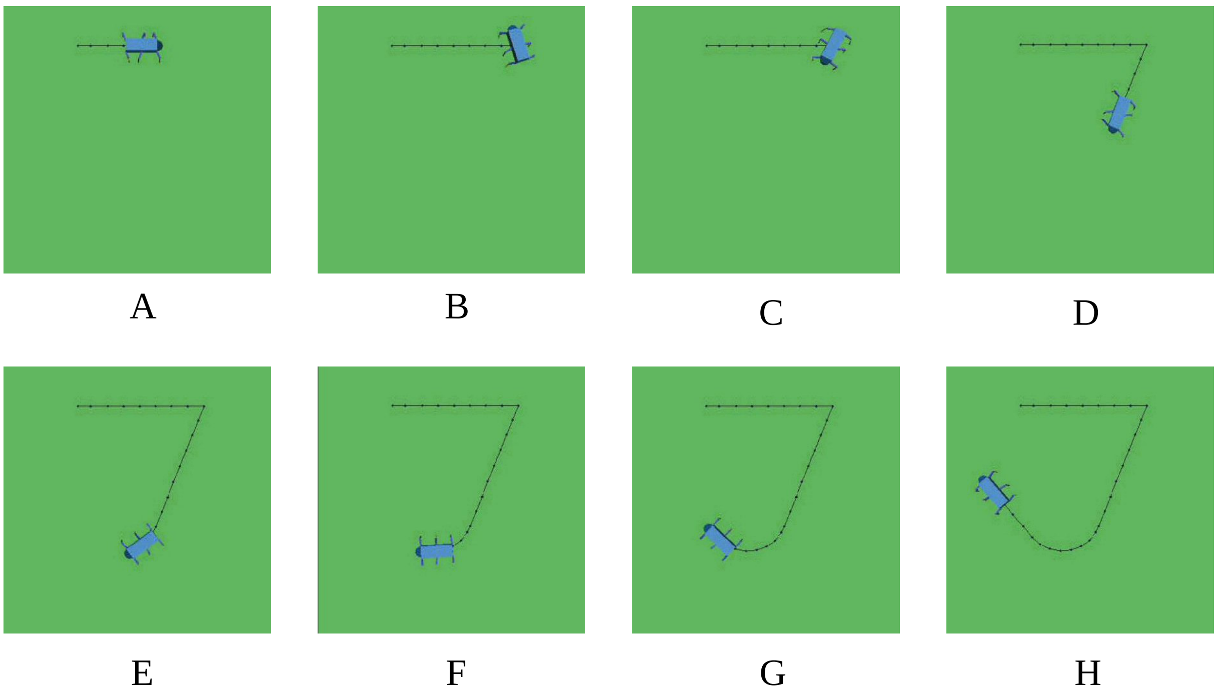


Fig. 9. A sequence of snapshots of the simulated Argos robot executing different heading commands.

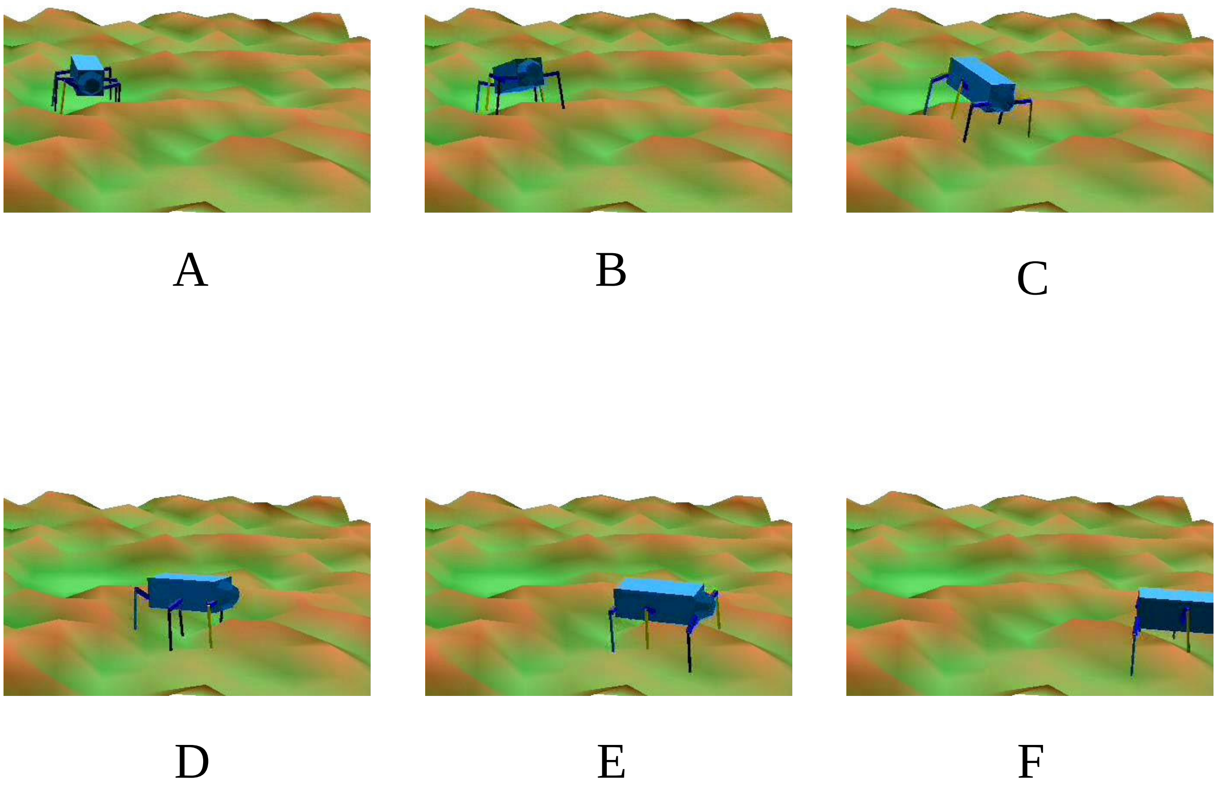


Fig. 10. Argos performing a turn to the left while walking on abrupt terrain.

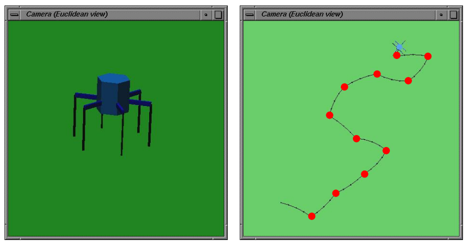


Fig. 11. Right picture show the trace followed by the spider-like robot shown on the left receiving heading commands generated at random.

again, next (snapshots E to F) it makes a turn with a radius different from 0, finally (snapshot H) the robot returns to the straight ahead advance. Our controller is able to execute any heading change without any problem and with smooth transitions when new turning centers are given.

The overall performance of the controller can be seen in the sequence of snapshots of figure 10 where the simulated Argos robot is walking over irregular terrain while executing a turn to the left: despite the terrain is very abrupt the robot traverses it without any problem.

In our development, we have taken few assumptions about the structure of the robot and, consequently, the resulting controller can be applied to different robots with minor parameter adjustment. To show this point, we simulated a spider-like six-legged robot using the same controller we applied to the Argos robot. Figure 11 shows the trace followed by this spider-like robot receiving heading commands generated at random (the circles mark the moment new heading commands are issued). The figure illustrates that the robot is able to perform very sharp direction changes, as well as smooth ones, without problems.

To test the generality of our controller, we are currently implementing it on a Lauron III six-legged robot we recently acquired (see figure 12). We had to re-implement the controller using the MCA2 framework [32]

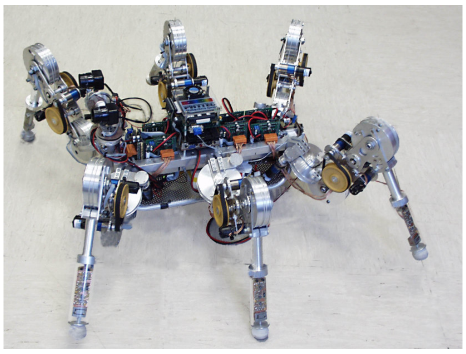


Fig. 12. The Lauron III six-legged robot designed and built by the FZI group at the University of Karlsruhe.

but the implementation only required of some parameter adjustment leaving the entire controller structure unchanged. The main issue was to implement a discrete version of the posture control introduced in section 4. In a real robot the smooth displacements of legs in contact with the ground have to be approximated using discrete displacements, trying to avoid leg slip as much as possible. The performance of the heading control mechanism introduced in this paper relies on this approximation to be accurate enough. The results we obtained so far [33] show that using Lauron, the leg slip is less than 1 cm, which is small enough to be compensated by the mechanical structure of the legs. Thanks to this accuracy the results with Lauron do not significantly differ from those we obtained with the simulation.

6 Conclusions

In this paper, we have presented a reactive controller for legged robots. The idea we followed is to decouple the control of legs when in return phase from that of legs when in support phase and to confront both subproblems assuming no special leg configurations.

For the first subproblem, we have introduced a gait generation mechanism that guarantees the stability of the robot at any time and that produces an efficient gait pattern (always in accordance with the terrain conditions). For instance, we have proved that our gait generation mechanism spontaneously produces the tripod gait when walking on flat terrain.

For the second subproblem we have used a particularization of a posture control introduced in [19]. This new mechanism tries to minimize at any time the distance between legs and the centers of their respective workspaces, but taking into account the restrictions in the robot's movements imposed by the trajectory to be followed.

Since the two subtasks are solved for any possible configuration of legs, the resulting controller is able to efficiently react to any leg movement necessary to adapt to terrain irregularities.

The presented controller has been designed in accordance with the main ideas of the reactive control paradigm and, thus, it does not use any map of the environment. The fact that our controller has good performance even if unforeseen obstacles are encountered or if the environment is changed on-line supports the hypothesis that processes for which high level planning-based methods were supposed to be needed can actually be solved using much simpler tools.

However, a remarkable difference w.r.t. other reactive controllers is that ours is able to follow the desired trajectory with a high degree of accuracy, even when walking on very abrupt terrain. In this way, we achieve the good performance of deliberative controllers but with the low computational cost of reactive ones. This makes our controller adequate for real applications where the robot has to move in unknown dynamic environments.

Of course, if the robot is expected to move in a static environment and a detailed map of that environment is available (which is not the case for natural environments), a classical planning-based controller would result better than any possible reactive controller, including ours.

Finally, our controller has been developed under very general assumptions about the robot and, thus, it can be applied to different legged robots. To validate this, we have performed tests with the Genghis robot and with other simulated and real robot structures, with good results in all cases, as reported in this paper. Hopefully, others could use our results to take advantage of the superior capabilities of legged robots with respect to wheeled ones, but without having to deal with the complexity of legged robot locomotion.

Acknowledgments

We would like to express our gratitude to Dr. Lluís Ros and Prof. Carme Torras for their cooperation and comments during the preparation of this paper.

This paper has been partially supported by the Spanish *Ministerio de Ciencia y Tecnología* and FEDER funds, under the projects DPI2000-1352-C02-01 and DPI2003-05193-C02-01 of the *Plan Nacional de I+D+I.*

References

[1] J. Bares, M. Herbert, T. Kanade, E. Krotkov, T. Mitchell, W. L. Whittaker, Ambler: An Autonomous Rover for Planetary Exploration, IEEE Computer 22 (6) (1989) 18–26.

[2] E. Krotkov, R. Simmons, W. L. Whittaker, Autonomous Walking Results with the Ambler Hexapod Planetary Rover, in: Proceedings of the International Conference on Intelligent Autonomous Systems, 1993, pp. 46–53.

[3] S. Hirose, A Study of Design and Control of a Quadruped Walking Vehicle, The International Journal of Robotics Research 3 (2) (1984) 113–133.

[4] D. Kumar, K. Deb, A. Ghosh, Optimal Path and Gait Generations Simultaneously of a Six-Legged Robot using a GA-fuzzy Approach, Robotics and Autonomous Systems 41(1) (2002) 1–20.

[5] D. E. Orin, Supervisory Control of a Multilegged Robot, The International Journal of Robotics Research 1 (1) (1982) 79–91.

[6] R. B. McGhee, G. I. Iswandhi, Adaptive Locomotion of a Multilegged Robot over Rough Terrain, IEEE Transactions on Systems, man and Cybernetics SMC-9 (4) (1979) 176–182.

[7] D. W. Marhefka, D. E. Orin, Gait Planning for Energy Efficiency in Walking Machines, in: Proceedings of the 1997 IEEE International Conference on Robotics and Automation, 1997, pp. 474–480.

[8] P. K. Pal, V. Mahadev, K. Jayarajan, Gait Generation for a Six-Legged Walking Machine through Graph Search, in: Proceedings of the 1994 IEEE Conference on Robotics and Automation, 1994, pp. 1332–1337.

[9] S. Salmi, A. Halme, Implementing and Testing a Reasoning-Based Free Gait Algorithm in the Six-Llegged Walking Machine,MECANT, in: Proceedings of the IFAC International Conference on Intelligent Autonomous Vehicles, Helsinki, Finland, 1995, pp. 127–132.

[10] B. S. Choi, S. M. Song, Fully Automated Obstacle-Crossing Gaits for Walking Machines, IEEE Transactions on Systems, Man and Cybernetics 18 (6) (1988) 952–964.

[11] R. C. Arkin, Behavior-Based Robotics, Intelligent Robotics and Autonomous Agents, The MIT Press, 1998.

[12] R. A. Brooks, A Robot that Walks: Emergent Behavior form a Carefully Evolved Network, Neural Computation 1 (2) (1989) 253–262.

[13] M. B. Binnard, Leg Design for a Small Walking Robot, Master's thesis, B. Sc. MIT (1992).

[14] C. Ferrell, Robust Agent Control of and Autonomous Robot with Many Sensors and Actuators, Master's thesis, M. S. Thesis MIT (1993).

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

[16] F. Delcomyn, M. E. Nelson, Architectures for a Biomimetic Hexapod Robot, Robotics and Autonomous Systems 30 (2000) 5–15.

[17] B. Klaassen, R. Linnemann, D. Spenneberg, F. Kirchner, Biomimetic Walking Robot SCORPION: Control and Modeling, Robotics and Autonomous Systems 41(2–3) (2002) 69–76.

[18] U. Saranli, M. Buehler, D. E. Koditschek, RHex: A Simple and Highly Mobile Hexapod Robot, The International Journal of Robotics Research 7 (20) (2001) 616–631.

[19] E. Celaya, J. M. Porta, A Control Structure for the Locomotion of a Legged Robot on Difficult Terrain, IEEE Robotics and Automation Magazine, Special Issue on Walking Robots 5 (2) (1998) 43–51.

[20] S. T. Venkataraman, A Simple Legged Locomotion Gait Model, Robotics and

Autonomous Systems 22 (1) (1997) 75–85.

[21] H. J. Chiel, R. D. Beer, J. C. Gallagher, Evolution and Analysis of Model CPGs for Walking I. Dynamical modules., J. Computational Neuroscience 7 (2) (1999) 99–118.

[22] R. D. Beer, H. J. Chiel, J. C. Gallagher, Evolution and Analysis of Model CPGs for Walking II. General Principles and Individual Variability., J. Computational Neuroscience 7 (2) (1999) 119–147.

[23] R. D. Beer, H. J. Chiel, P. Larsson, A Distributed Neural Network Architecture for Hexapod Robot Locomotion, Neural Computation 4 (1992) 356–365.

[24] H. J. Chiel, R. D. Beer, R. D. Quinn, Robustness of a Distributed Neural Network Controller for Locomotion in Hexapod Robots, IEEE Transactions on Robotics and Automation 8 (3) (1992) 293–303.

[25] S. M. Song, K. J. Waldron, An Analytical Approach for Gait Study and its Application on Wave Gaits, The International Journal of Robotics Research 6 (2) (1987) 60–71.

[26] D. M. Wilson, Insect Walking, Annual Rev. of entomology 11 (1966) 103–122.

[27] D. Wettergreen, C. E. Thorpe, Gait Generation for Legged Robots, in: Proceedings of the *IEEE* International Conference on Intelligent Robots and Systems, 1992, pp. 1413–1420.

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

[29] J. M. Porta, E. Celaya, The Behavior Language for PC: User's Guide, Tech. Rep. IC-DT-9602, Institut de Cibernètica (1996).

[30] R. A. Brooks, The Behavior Language; User's Guide, Tech. Rep. AI Memo 1227, MIT (1990).

[31] J. M. Porta, The Legged Robot 3D Simulator: Description and Programming Guide, Tech. Rep. IRI-DT-2000/4, Institut de Robòtica i Informàtica Industrial (2000).

[32] K.-U. Scholl, J. Albiez, B. Gassmann, MCA – an expandable modular controller architecture, in: Poceedings of the 4th Linux Real Time Workshop, Milano, Italy, 2001, pp. 409–416.

[33] E. Celaya, J. L. Albarral, Implementation of a Hierarchical Walk Controller for the LAURON III Hexapod Robot, in: International Conference on Climbing and Walking robots (Clawar 2003), 2003, pp. 409–416.