BIPedal walking: from gait design to experimental analysis

Christine Azevedo\textsuperscript{a} Nicolas Andreff\textsuperscript{b} Soraya Arias\textsuperscript{a}

\textsuperscript{a}INRIA, 655 avenue de l’Europe 38334 St.Ismier Cedex France
\textsuperscript{b}IFMA/LaRAMA, Campus de Clermont-Ferrand/Les Cezeaux BP 265, 63175 Aubière Cedex France

Abstract

This paper presents an experimental approach to the problem of designing and executing walking gaits on a dedicated 2-legged machine. We have oriented all our approach towards the experimental analysis of large pattern of walking gaits. This work has lead to the first experimental results obtained on the Bip anthropomorphic robot. The desired movements are designed off line using a model of the robot and tracked on the real system by means of a simple control law. The success of our approach is due both to an efficient mechatronic architecture and to the way it is used to achieve the goal of experimenting walking. The paper presents the system architecture, from mechanical to software issues and also describes the approach developed for designing and executing locomotion. Several results validate the accuracy of our modelling and exhibit the robustness and the efficiency of our controller architecture. We also present and evaluate one of the gaits realized with Bip, using both robotic and biomechanical criteria.

Key words: bipedal walking, gait design, control architecture, biomechanics, performance evaluation

1 Introduction

Legged robots belong to the class of mobile robots like those with wheels or caterpillars. Legs are adapted to cluttered environments allowing the machine to stride over obstacles and limiting the damages to the environment.
Fig. 1. The 8 DoF anthropomorphic walking robot: Bip and its kinematic model thanks to their small support surface. Moreover, the use of legs gives the capability of changing the configuration for striding over obstacles, which is not possible with other mobile vehicles using wheels for instance. Finally, bipeds are adapted from their design to human-oriented facilities (stairs, corridors). This potential efficiency has motivated the large amount of works on 2-legged robots [22,39,8,23,40,29,20]. The limitation of the number of support legs in bipeds raises specific needs such as the handling of impacts and contacts, and the preservation of the dynamic equilibrium in the sense of falling avoidance, despite possible disturbances.

Usually, the control of biped robots is addressed in two steps: i) the synthesis of walking patterns and ii) the control of the robot along a reference trajectory [11,21,10,12,30,33,34,15], with sometimes the on-line monitoring of variables related to the walking stability. A major research issue in the area of mobile robots is to increase their capacity of being autonomous, although their specific mode of locomotion is in itself a rich field of investigation. Despite the huge literature on the control of walking robots, optimization and stability of gaits remain an open question. Indeed, it appears that the resulting gaits are influenced by many parameters such as the step length, the average velocity or the foot clearance, the tuning method of which remains often a mystery. Moreover, many other relevant influence factors have to be considered when experimenting gaits: backlashes, frictions, flexibilities of the mechanical structure, motor and power characteristics, real-time control implementation or ground contacts, among others. Ignoring them or taking them into account through simplified models seem indeed unrealistic. Thus we claim that pertinent analysis of the gaits cannot be performed without conducting accurate experiments in view of tuning and evaluation. However, and to our knowledge, the literature does not exhibit such experimental comparisons between gaits. Another motivation of our approach comes from recent theoretical results [38] showing that walking control should not be restricted to the tracking of a single trajectory but over trajectory families. Therefore, a prerequisite is to gain hints of the biped behavior when performing gaits. We insist on the fact that these gait comparisons must be experimental. The main reasons why simulation does not suffice are linked to previous remarks concerning the in-
fluence parameters. First, the usual argument against simulation concerning the realism of the simulated model is all the stronger here that there does not exist any consensual biped model gathering all the physical phenomena. In particular, how should perturbations be realistically modelled ? Secondly, real-time issues are essential for biped control and strongly depend on the robot software and hardware architecture. Thirdly, this architecture and all the elements of the system strongly influence the energy expenditure, which is an item of high interest in view of autonomy.

Bip is an anthropomorphic biped robot with eight active joints [16,18,4] (fig.1). Dimensions and capabilities of Bip are close to human ones. Since robot configurations are not easy to handle using the joint values, we propose here a so-called output space where the configurations have a physical meaning. This output space is obtained using an output function from the joint space to the output space. Defining this output space, we bore in mind that each of its dimensions should allow for physical sensing. Thus, we open a way to sensor-based control of biped walking. However, to demonstrate the validity of such an output space, we restricted ourselves to use it in open-loop joint control as follows. First, we designed parameterized symmetrical statically stable walking gaits on flat ground. Then, a simple controller allows to track robustly the joint trajectories. Important factors in the efficiency of our controller are real-time performances and low sampling rates, these are handled through the software implementation. The ease of the trajectory design and the robustness of control architecture allows us to experiment several gaits [2]. It should be emphasized that a comparison between gaits requires to define evaluation criteria. Since we are concerned with an anthropomorphic robot, it looks natural to be inspired by biomechanical data.

The remainder of this paper decomposes as follows. Section 2 describes shortly the whole mechatronic system Bip, focusing on software environment. Section 3 presents the output space we recommend to easily model the robot state and its use for trajectory planning. Section 4 shortly depicts the open-loop control used in the experiments and its actual real-time implementation. Finally, Section 5 presents the various gaits we experimented and their analysis upon biomechanical criteria.

2 Description of the Mechatronic System

Bip is a complete mechatronic system, its three main components are: the mechanical prototype, the electronic architecture and the software aspects. All these aspects are important for the global efficiency of the robot.
Fig. 2. Screw-nuts with satellite rollers transmitters (from left to right: single transmitter at the knee - two transmitters mounted in parallel at each ankle)

2.1 **Mechanical architecture**

**BIP** was aimed at having anthropomorphic characteristics (fig.1), but not at copying the human model [16]. Six joints (two hips $q_4$ and $q_8$; two knees $q_3$ and $q_7$; two ankles $q_2$ and $q_6$) compose the model in the sagittal plane (plane of forward progression). The rotations of these articulations control the forward progression. Two joints (two ankles $q_1$ and $q_5$) compose the model in the frontal plane. The rotations of these articulations are used to control the lateral balance.

The actuators are brushless DC motors. The joints are equipped with specific transmitters (fig. 2): screw-nuts with satellite rollers combined with rod-crank systems [35]. The nut with satellite rollers is inserted in a slider which is guided by four rollers that can move along a straight beam. The rotation of the screw produces the translation of the slider, which itself pushes or pulls on two rod-acting on an arm of the adjacent limb. These transmitters allow a high accuracy with low friction. They also provide high reversibility and variable mechanical advantage (e.g. the reduction ratio varies from 0.6 to 1 at the knee). High velocities and high torques can be transmitted by this mechanism. These characteristics allow the possibility of a dynamical control of the joints. These transmissions are arranged in a single form at hips and knees; they are mounted in parallel in association with Cardan universal joints at ankles (fig.2). The geometric parameters of the skeleton are close to the ones of a human of the same size. The mass distribution and capacities of joints velocities and torques are taken from human measured data:

<table>
<thead>
<tr>
<th>Segment</th>
<th>Length</th>
<th>Height</th>
<th>Weight</th>
<th>Center of mass (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thigh</td>
<td>0.41m</td>
<td></td>
<td>11kg</td>
<td>[0.258,0.028,0.00]</td>
</tr>
<tr>
<td>Leg</td>
<td>0.41m</td>
<td></td>
<td>6kg</td>
<td>[0.250,0.05,0.045]</td>
</tr>
<tr>
<td>Foot</td>
<td>0.29m</td>
<td>0.083m</td>
<td>2.5kg</td>
<td>[0.024,0.151,0.00]</td>
</tr>
<tr>
<td>Total</td>
<td>0.95m</td>
<td></td>
<td>46kg</td>
<td></td>
</tr>
</tbody>
</table>

In this paper, we present the results obtained with a simplified mechanical
system. The robot is maintained in the sagittal plane by a mechanical system composed of a torsion tube attached to a trolley with four wheels (fig.3). The tube ensures the lateral equilibrium, but in the sagittal plane the robot is totally free to move and fall. The system is composed of the two thighs, two legs and two feet, with six degrees of freedom. The wheels of the trolley are supposed to present no friction. The torsion tube is assumed to apply a pure vertical force on the pelvis.

2.2 Electronic architecture

The robot is equipped with several sensors, more detail can be found in [5]. Synchro-resolvers that can emulate digital encoders provide us with accurate relative angular positions of the motor axes. Potentiometers are directly mounted on every joint and allow for the recovery of the absolute positions at the system initialization. Switches are located near the extremities of the transmitters straight beams to indicate joint limits. Within each foot, three force sensors are located between the sole and the ankle cardan. They give the vertical component of the ground reaction force and the associated momentums, and the position \((l, L)\) of the center of pressure (COP) on the sole plane (fig.4).
2.3 Software architecture

2.3.1 General presentation

The drivers and programs for the Bip robot run on a MOTOROLA MVME162 board based on a 68040 processor at 32 MHz. This board uses the VxWORKS operating system (fig.5). Programs are first implemented and cross-compiled on a host workstation then downloaded through an Ethernet link via a VxWORKS 5.3 command to the target MVME162 card. Except the driver programs, the programs on the Bip robot were implemented through the ORCCAD programming environment, as presented in the following section.

2.3.2 Programming with ORCCAD

ORCCAD (Open Robot Controller Computer Aided Design) [36] is both a methodology and a software environment. It is dedicated to the implementation of complex robotic control laws running on advanced robotics platforms. ORCCAD is particularly concerned with systems which strongly interact with their environment, through several actuators and sensors. Its prime goal is to enforce software reusability and to take care of all the system and real-time issues while the automatician focuses on the control algorithm. This software environment developed at Inria has been used for several years to conduct various experiments [31,24,14,25]. In this section we recall the ORCCAD’s main concepts, a more comprehensive description can be found in [36]. The key entity in ORCCAD methodology is the Robot Task, which represents an elementary robotic action, like a sensor-aided grasping of an object. Hence a robotic application is defined in ORCCAD as the logical composition of several Robot Tasks running in sequence and/or in parallel. This composition is called a Robot Procedure. Once the Robot Tasks and Robot Procedures have been specified, ORCCAD translates the main Robot Procedure (corresponding to the application) into a real-time C++ code: VxWORKS for hard real-time applications and LINUX or SOLARIS for soft real-time applications. Thus, reusability is considered in ORCCAD both on specification level
modules designed through the Orccad GUI can be used on others Robot Tasks) and on execution level (i.e. portability on different operating systems).

2.3.2.1 The Robot Task

An Orccad Robot Task is defined as the specification of:

1. a Physical Resource, which represents the mechanical system to be controlled, i.e., actuators and sensors of the robotic system.
2. a Control Law, considered as an invariant computation chain between sensors and actuators, running periodically until a goal is achieved or a given condition is met.
3. a Logical Behavior, represented by several logical states, the switching from a state to another occurs when given conditions are met and then the associated discrete events are emitted.

For example, tracking a pre-computed trajectory using a PD control law as defined in section 4 is a good candidate for a Robot Task. Thus, a Robot Task merges the continuous aspects of a control law and the discrete aspects of a logical behavior:

- the continuous aspects are specified through a block diagram scheme whose building blocks are reusable software modules communicating through data ports. This module chain usually begins by reading the physical resource state represented by physical resource modules then algorithmic modules are used to compute the control law, and the output is sent to the robot system via a physical resource module again. For example, the calculation of a Jacobian matrix or a gravity vector can be handled in an algorithmic module. The modules are linked with one another via the data ports. These data can be information about the robotic state, the result of an intermediate calculation, a calculation state data, or the final control to be given to the robotic system.
- The computation chain implementing the continuous aspects described above may require several computational modes: an initialization mode, a nominal execution mode and an ending mode. The transitions between a mode (or state) to another define the discrete behavior associated to a Robot Task. This discrete behavior consists in the sequence of three predefined states (respectively: initialization, nominal execution, end). For each of these states, the corresponding computation function must be defined for each module of the block diagram scheme. The switching between one state to another is fired by a special unique module, called the automaton module. This module handles the set of signals conditioning the control law execution, i.e., the signals that can be produced by the physical resource and algorithmic modules during the computation chain execution. Each link between a module and the automaton is typed according to the transition of state
to be reached when an event is received by the automaton module through the link. The types considered are: preconditions, postconditions, and exceptions. When all preconditions are met, then the automaton switches all modules specified for a Robot Task from initialization mode to nominal computation mode.

At last, the execution time for each algorithmic module is to be provided as part of the Robot Task specification.

2.3.2.2 Robot Procedure

In order to build a complete robotic application, ORCCAD provides the concept of Robots Procedures. These Robot-Procedures are used to logically and hierarchically compose Robot-Tasks and Robot-Procedures in structures of increasing complexity from a single action to a full mission. For example, a Robot Procedure can be:

```
start (RobotTask_1 then RobotTask_2) until EVENT_Y
then
start (RobotProcedure_2 in parallel to RobotTask_3) until EVENT_Z
```

With such an approach, the logical behavior is independent of the implementation details of each elementary action. Hence, the robotic application specification portability is ensured: a Robot Procedure can be used on a different robotic platform if the associated Robot Tasks are available for the new platform.

Two different languages can be used to implement Robot Procedures in ORCCAD: 1) Esterel language [7,6] (better suited for simple applications); 2) MaestRo language [13] (suited for more complex robotic missions).

3 Gait trajectory generation

We propose an approach to generate intuitively parameterized walking gaits for the robot.

![Fig. 6. Support phases in walking gait](image-url)
3.1 Presentation

Normal human gait is symmetric, cyclic and three-dimensional [37,19]. Most of the movements occur in the sagittal plane. The cyclic nature is provided by the periodic leg movement moving each foot from one position of support to the next (fig.6). A complete gait cycle begins when one foot strikes the ground and ends when it strikes the ground again. A step is referred to one given foot, it begins when this foot strikes the ground and ends when it takes-off. Two major phases divide the cycle: stance (approximately 60% of the cycle) and swing (approximately 40% of the cycle). We can distinguish single support and double support periods depending on the number of feet in contact with the ground.

In statically stable walking, the body is always in a stable position, i.e. if at any time the motion is stopped, the robot will not fall. This type of walk is adopted in particular situations such as walking on unstable ground or descending stairs. Walking speed must be slow and the steps length short. Dynamically stable walking is the normal human gait, i.e. the body is continually falling towards its next step, but never actually loses control. The motion cannot be stopped at any time without falling over.

In the present work only static stability is considered. Indeed, since possible applications of biped robots are exploration of unknown regions or rescue operations in damaged areas, requiring cautious walking, it is necessary to master statically stable walking. Static stability is obtained if the ground projection of its center of mass (COM) falls within the convex hull of the foot support area (support polygon). For detailed analysis of human walking, the reader is referred to [27,32,41]. One of the questions in biomechanics research is to determine performance criteria of a successful gait, considering both aesthetics of walking and reliability of the locomotor act. The physical quantities used for evaluation should be able to describe the important aspects of walking [9,28]:

- symmetry and simplicity of the movements
- maintenance of balance
- mechanical load on the body
- energy expenditure.

The trajectories presented in this document correspond to statically stable, symmetric walking gaits in the sagittal plane. They avoid the double support phase and are neither based on anthropomorphic data nor optimization criteria. Nevertheless, we defined them to have some aesthetic similarities with human walking.
3.2 Output function

Walking trajectories are hard to design in the joint space. Indeed, it is not intuitive to find the joint motions that both fulfill the locomotion requirements and the stability constraints. To overcome this difficulty, the two immediate ideas would be to record the joint motion on a human being (with the trouble of converting the data into the robot model) or to place manually the robot in key positions (with the associated practical complexity). Alternately, we define in this section an output space where the robot six DOF are more intuitively handled. Thus, there exists a so-called output function, parameterized by the robot geometry, from the joint space to the output space. Inverting it, we can transform output space trajectories into joint trajectories, that can be followed using the control technique presented next (§ 4). The output function which seems the most physically interpretable is given by:

\[ f = \begin{pmatrix} x_{\text{com}} & y_h & \theta_t & x_a & y_a & \theta_s \end{pmatrix}^T \]  

(1)

where \( x_{\text{com}} \) is the center of mass projection onto the walking direction, \( y_h \) the hip height, \( \theta_t \) the trunk angle, \( x_a \) the free ankle projection onto the walking direction, \( y_a \) the free ankle height, \( \theta_s \) the free sole angle and the reference frame is attached to the toe end of the supporting foot, with orientation given in Figure 7.

3.3 Statically stable, symmetric gait

To define a trajectory, we need to define the desired time-varying values of the output function:

\[ f(t) = \begin{pmatrix} x_{\text{com}}(t) & y_h(t) & \theta_t(t) & x_a(t) & y_a(t) & \theta_s(t) \end{pmatrix}^T, \quad \forall \ t_0 \leq t \leq t_f \]  

(2)
where \( t_0 \) (resp. \( t_f \)) is the time when the step begins (resp. ends). These values must of course be at least time continuous and avoid ground collision. We require also that they preserve the robot static stability.

**Static stability**  To preserve the robot static stability, one must ensure that the COM ground projection always remains in the supporting polygon. In our case, this means that \( x_{com} \) must remain between the heel and the toe of the supporting foot (in the single support phase) or the rear foot heel and the front foot toe (in the double support phase).

**Double support handling**  During double support phase, robot legs form an over-actuated closed kinematic chain, which can be seen as a hyperstaticity-like problem. Therefore, any joint motion must be compensated for by motion of the other joints. Furthermore, the force sensors placed on the feet do not allow us to measure tangential forces. Since this yields a tough theoretical, practical and numerical problem, out of the present scope of our study, we choose to simply solve it by reducing the double support phase duration. Therefore, the right single support phase is immediately followed by the left single support phase and reciprocally. Each step consists then only of a single support phase.

**Step length and feet overlapping**  Consequently to the discussion above, the COM projection must lie, at the end of each step, simultaneously between the heel and the toe of both feet. To increase robustness, we defined a security margin at the heel \( x_1 \) and another one at the toe \( x_2 \), thus reducing the interval of allowed COM projection positions. These two security margins could hence be considered as design parameters of the gait. However, their definition implies that the two feet must “overlap” in the walking direction, with a minimal overlapping length equal to \( x_1 + x_2 \). Choosing this minimal overlapping length, we immediately deduce the following relation between \( x_1 \), \( x_2 \), the step length \( l_s \) and the foot length \( d \):

\[
 l_s = 2(d - x_1 - x_2) \tag{3}
\]

We can thus replace, in the parameter set, \( x_2 \) by \( l_s \). The remaining parameter \( x_1 \) can be seen as defining whether the robot walks on its heels or on its toes, which is equivalent to defining a median position for the COM projection:

\[
 x_{med} = \frac{-d + x_1 - x_2}{2} = x_2 - \frac{l_s}{4} \tag{4}
\]
Consequently, $x_1$ and $x_2$ can favorably be obtained by the more physically interpretable design parameters $x_{med}$ and $l_s$:

\begin{align}
  x_1 &= d - x_{med} - \frac{3l_s}{4} \\
  x_2 &= x_{med} + \frac{l_s}{4}
\end{align}

(5)  (6)

A second consequence of the minimal overlapping length choice is that the COM projection must lie, when the step begins ($t = t_0$), at distance $x_1$ from the support foot heel, and, when the step ends ($t = t_f$) at distance $x_2$ from the support foot toe:

\begin{align}
  x_{com}(t_0) &= -d + x_1 \\
  x_{com}(t_f) &= -x_2
\end{align}

(7)  (8)

The minimal overlapping length also uniquely defines the initial and final free ankle positions:

\begin{align}
  x_a(t_0) &= -d + x_1 + x_2 - d_2 \\
  x_a(t_f) &= d_1 - x_1 - x_2
\end{align}

(9)  (10)

where $d_1$ (resp. $d_2$) is the length of the front (resp. rear) part of the foot.

Since we study statically stable gaits, we are not in the conditions of “normal” human walking. Combining the reduction of the duration of the double support phase and the minimal overlapping, allows us to provide a flowing and “natural” aspect to our gaits.

**Free ankle height**  To prevent, in a simple manner, the free foot from hitting the ground before the step ends, we arbitrarily chose to keep it horizontal ($\theta_s(t) = 0, \forall t$). Thus, the non-contact constraint becomes equivalent to the
free ankle height $y_a$ being greater than the support ankle height ($h$):

$$y_a(t_0) = y_a(t_f) = h \quad \text{(11)}$$

$$y_a(t) > h, \forall t_0 < t < t_f \quad \text{(12)}$$

**Trunk angle and hip height** Finally, the last two output function components $(\theta_t, y_h)$ are not subject to any specific constraint. Nevertheless, we require arbitrarily that the trunk remains vertical ($\theta_s(t) = 0, \forall t$). The hip height is left completely free during the step, defining implicitly the knee bending. To fulfill the symmetry condition, it should nevertheless be equal at times $t = t_0$ and $t = t_f$.

**Summary** Taking into account all the constraints above, the statically stable, symmetric gait trajectories are of the form:

$$\forall t_0 < t < t_f$$

$$f(t) = (x_{\text{com}}(t), y_h(t), 0, x_a(t), y_a(t), 0)^T \quad (> h) \quad \text{(13)}$$

$$f(t_0) = (-d + x_1 \ y_h(t_0), 0, -d + x_1 + x_2 - d_2, h, 0)^T$$

$$f(t_f) = (-x_2 \ y_h(t_f), 0, d_1 - x_1 - x_2, h, 0)^T$$

### 3.4 Polynomial trajectories

Having expressed the constraints on the desired output function, we can now address the step trajectory generation itself. To do so, the easiest is to interpolate polynomials between intermediate key positions in the output space:

$$f(t) = \sum_{i=0}^{n} A_i t^i \quad \text{where} \quad A_i = (a_{x_{\text{com}}}^i, a_y^i, a_{\theta_t}^i, a_{x_a}^i, a_{y_a}^i, a_{\theta_s}^i)^T \in \mathbb{R}^6 \quad \text{(14)}$$

For a walking trajectory on flat ground such that it is time-continuously differentiable and that the foot lands smoothly (i.e. without impact), polynomials of the 4th degree are enough. Indeed, this requires 5 constraints on each output space component, two of which were already exhibited and the remaining three constraints may be given in the following way.

To avoid impact between the free foot and the ground, we impose that the velocity in the output space must be 0 at step end:

$$\frac{df}{dt}(t_f) = 0 \quad \text{(15)}$$

To ensure velocity continuity when switching support, the initial output space velocity is also 0:

$$\frac{df}{dt}(t_0) = 0 \quad \text{(16)}$$
Finally, the last constraint is arbitrarily given at intermediate time $t_m = (t_0 + t_f)/2$ in order to actually take the free foot off the ground. It is such that

$$f(t_m) = \begin{pmatrix} \frac{x_{com}(t_0) + x_{com}(t_f)}{2} y_h(t_m) 0 \frac{x_a(t_0) + x_a(t_f)}{2} y_a(t_m) 0 \end{pmatrix}^T$$

where $y_a(t_m) > h$ and $y_h(t_m)$ are free parameters of the trajectory, the other terms being already defined.

To conclude, the output space trajectories are obtained in this work through 4th degree polynomial interpolation between initial, intermediate and final positions. They ensure robot static stability, without motion in the double support phase nor impact between the free foot and the ground. They are parameterized by:

- initial hip height ($y_h(t_0)$),
- intermediate hip height ($y_h(t_m)$),
- intermediate free ankle height ($y_a(t_m)$),
- step length ($l_s$),
- median position of the center of mass ground projection ($x_{med}$);
- step duration ($t_f - t_0$).

Once the output space trajectories obtained, they are translated into joint space trajectories, i.e. joints desired positions and velocities ($q_d$ and $\dot{q}_d$). This is done by numerical inversion of the output function, involving the inversion of the forward kinematic model. This is not much of a trouble in the present state of the controller since this translation is done off-line and its result is checked before on-line tracking of the joint trajectory (as described below). Notice that experimentally, we never encountered any incoherent joint trajectory.

However, in a future evolution of the controller, where measures will be made in the output space through adequate sensors, this inversion will have to be done on-line with guaranteed results, which is similar to the use of an inverse kinematic model in Cartesian control of a serial manipulator.

The next step is therefore to define a tracking controller for the joint trajectories.

4 Control

4.1 Modelling

The dynamics of the robot can be written using Lagrangian form as follows:

$$M(q)\ddot{q} + N(q, \dot{q})\dot{q} + G(q) = \Gamma + \Gamma_e$$

(18)
where \( q \) is the configuration vector containing the 6 joint positions (fig.7). \( M \) is the inertia matrix, \( N \) is the matrix containing the centrifugal forces and Coriolis effects, \( G \) is the gravity vector. \( \Gamma \) is the vector of the 6 torques applied to the joints. \( \Gamma_e \) is the vector of the torques induced by external forces applied to the robot, such as ground reaction forces. The support foot of the robot is assumed to be fixed on the ground. We can therefore write two dynamic equations for each single support phase, depending on the support foot:

\[
\begin{align*}
M_{RSS}(q)\ddot{q} + N_{RSS}(q, \dot{q}).\dot{q} + G_{RSS}(q) &= \Gamma \\
M_{LSS}(q)\ddot{q} + N_{LSS}(q, \dot{q}).\dot{q} + G_{LSS}(q) &= \Gamma 
\end{align*}
\] (19)

where index \( RSS \) stands for right single support and \( LSS \) stands for left single support.

4.2 Trajectory tracking

The goal of the control law is to robustly track joint trajectories computed off-line. A simple control law: proportional derivative with gravity and friction compensation allows to satisfy low sampling rates, despite of a limited computation power [4]. The torques are hence computed as follows:

\[
\Gamma = K_p(q_d - q) + K_v(\dot{q}_d - \dot{q}) + \hat{G}(q)
\] (20)

where \( \hat{G} \) is the estimated gravity vector, computed from the robot physical model, depending on the support phase. The control parameters \( K_p \) and \( K_v \) have been experimentally selected using trial and error approach. Even though the trajectories are defined to avoid any double-support phase, the control must deal with unpredicted cases where both feet remain on the ground (model and measurement errors). To do so, we use a linear combination of the single-support phases models, weighted by the weight supported by each foot:

\[
\hat{G}(q) = \begin{cases} 
G_{RSS}(q) \\
G_{LSS}(q) \\
\lambda G_{RSS} + (1 - \lambda) G_{LSS}
\end{cases}
\] (21)

with \( \lambda \) the rate between the weight supported by the right foot given by the force sensors and the total weight of the robot. The actual control is the vector \( U = (U_i)_{i=1:6} \) of the currents sent to each motor. Using Armstrong [1] model, we estimated the friction constants \( F \) and introduced them at the motor level. Therefore, the current sent to the motor is:

\[
U = U' + F \cdot \text{sign}(\dot{\Theta}_m)
\] (22)
Fig. 9. Different gaits experimented with the robot

\[ \dot{\Theta}_m = \dot{\Theta} \]

\[ \dot{U}' = \dot{U'} \]

with \( \dot{\Theta}_m \) the motor velocity and \( U' \) the current converted from \( \Gamma \). The relation between the motor torques \( \Gamma_m \) and the joint torques \( \Gamma \) is:

\[ \Gamma_m = J^T \Gamma \quad (23) \]

where \( J \) is the matrix of the reduction ratios which are variable and depend on the robot configuration according to the transmitters design (§2.1). Finally, \( U' \) is obtained by:

\[ U' = K \Gamma_m \quad (24) \]

with \( K \) is the diagonal matrix containing the motor electrical constants.

4.3 Orccad Specification of the Bip Walk

In order to implement our algorithms on the robotic platform, we used the Orccad software environment. Details of implementation are given in Annex, page 26.

5 Experimental Results

We experimented several walking gaits on the robot. In figure 9 we give some intermediary postures of these tested gaits. Table 1 gives a short description of the gaits characteristics. In this paper we develop essentially the results obtained for one of these gaits (i.e. Gait0), referred to as the reference gait.
Fig. 10. Execution times

Fig. 11. **Left:** Output function over 1 step - **Right:** Joint positions evolution over 2 steps

in the sequel. For more detail about the other gaits see [2]. Experiments were realized with a sampling time $T_e = 8ms$ (fig.10).

**Walking gait generation**  The reference gait corresponds to steps of 38cm length. The robot moves at a speed of 0.095 m.s$^{-1}$. As we said static walking is slow. In comparison, walking velocity of an adult is of 1.3 m.s$^{-1}$. The characteristics of the step, described in the proposed output space are (fig.11-left):

- the pelvis height is the same at the beginning and at the end of the step, but it increases by 1cm during the step,
- the center of mass projection moves from the heel of the support foot towards the toe,
- the flying foot remains horizontal,
- the trunk is vertical during the whole step,
- and the height of the swing leg ankle increases by 5cm to avoid the ground and lands 38cm ahead.

Translating this trajectory in the joint space and replicating the joint trajectory symmetrically for each step yields a complex joint trajectory (fig.11-right) which could hardly be defined intuitively.

**Trajectory tracking**  Using the control law presented in section 4, we were able to experiment the gait on the real robot (fig.12). The tracking errors on the joints positions are always under 0.02 rad (1.3 degrees) (fig.13). It is possible to compute the normal support forces (fig.14) and the center
### Table 1

<table>
<thead>
<tr>
<th>Gait</th>
<th>Step length</th>
<th>Velocity</th>
<th>Energy expenditure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.38 m</td>
<td>0.095 m.s(^{-1})</td>
<td>14.5 J.s(^{-1})/153 J.m(^{-1})</td>
<td>reference gait</td>
</tr>
<tr>
<td>1</td>
<td>0.38 m</td>
<td>0.190 m.s(^{-1})</td>
<td>32 J.s(^{-1})/168 J.m(^{-1})</td>
<td>twice faster</td>
</tr>
<tr>
<td>2</td>
<td>0.50 m</td>
<td>0.125 m.s(^{-1})</td>
<td>16 J.s(^{-1})/128 J.m(^{-1})</td>
<td>larger steps</td>
</tr>
<tr>
<td>3</td>
<td>0.38 m</td>
<td>0.095 m.s(^{-1})</td>
<td>12 J.s(^{-1})/130 J.m(^{-1})</td>
<td>knee bended</td>
</tr>
<tr>
<td>4</td>
<td>0.38 m</td>
<td>0.095 m.s(^{-1})</td>
<td>12 J.s(^{-1})/129 J.m(^{-1})</td>
<td>knee slightly bended</td>
</tr>
<tr>
<td>5</td>
<td>0.38 m</td>
<td>0.095 m.s(^{-1})</td>
<td>15 J.s(^{-1})/166 J.m(^{-1})</td>
<td>pelvis height fixed</td>
</tr>
<tr>
<td>6</td>
<td>0.38 m</td>
<td>0.095 m.s(^{-1})</td>
<td>22 J.s(^{-1})/235 J.m(^{-1})</td>
<td>higher foot height</td>
</tr>
<tr>
<td>7</td>
<td>0.38 m</td>
<td>0.095 m.s(^{-1})</td>
<td>14 J.s(^{-1})/146 J.m(^{-1})</td>
<td>walking backwards</td>
</tr>
</tbody>
</table>

Description of the different gaits experimented on BIP

---

**Fig. 12. Snapshots from a movie of the experiments**

**Fig. 13. Gait0 experimental data - Left:** Evolution of the tracking errors for 6 steps  
**Right:** Joint torques measured for 6 steps

of pressure location (fig.15) from the force sensor data. Since our control is a regulation on pre-computed trajectories, the real support changing do not always occur as expected, explaining most of the errors. Nevertheless there is no abnormal shock at the feet impacts. It is easy to see that the normal support forces vary linearly between two steps, which confirms the hypothesis made for the computation of the gravity vector (§ 4.1). Theoretically, in the case of static walking, the center of mass projection (COM) and the center of pressure (COP) are supposed to be superposed. In Figure 15 we can see it is not exactly the case. However, the static stability is ensured since both the
Biomechanical interpretation  
To evaluate the gait, we applied biomechanical criteria. The first one is related to the aesthetics of walking, the others to the reliability of the locomotor act:

- symmetry and simplicity of the movements: this point has been considered when defining the trajectories and is therefore fulfilled,
- maintenance of balance: the static stability is ensured by maintaining the center of mass within the support polygon. Figure 15 shows that despite a simple open-loop control, this criteria is satisfied.
- mechanical load on body: the torques (fig.13) are repeated from one step to another without considerable increase, even at foot impact with the ground. The support forces (fig.14) also show a constant load (450N) and a good shock absorption.
- energy expenditure: we computed the total energy expenditure from the
measured motors torques and velocities data:

\[ E = \sum_{i=1}^{N_e} \sum_{j=1}^{6} |\Gamma_j(i) \dot{q}_j(i) T_e| \]

where \( N_e \) is the number of samples and \( T_e \) the sampling period. It is interesting to compute the energy consumption in function of the forward progression of the body. For the reference gait \textbf{Gait0}, the average consumption by length unit \( \text{65.41} \text{J.m}^{-1} \) and by time unit \( \text{6.21} \text{J.s}^{-1} \). In normal human walking analysis, the energy consumption (metabolic + mechanic) is normalized with respect to the weight and its average value is \( \text{3.5} \text{J.kg}^{-1}.\text{m}^{-1} \) (\( \text{0.8} \text{cal.kg}^{-1}.\text{m}^{-1} \)) \cite{17}. In comparison, dividing the average consumption by length unit of the robot by its mass (46kg) yields a normalized energy consumption very similar since it is equal to \( \text{1.42} \text{J.kg}^{-1}.\text{m}^{-1} \) for a walking speed 10 times lower than the normal human walking speed. This seems coherent with the fact that human walking is dynamically stable while the experimented gaits are statically stable. Nevertheless, this result remains very interesting, keeping in mind that dynamic walking is more economic. The energy expenditure for the other gaits are gathered in Table 1. The repartition of the robot consumption is: 30\% for the ankles, 30\% for the knees and 40\% for hips.

With respect to these criteria, the first gait in the history of \textbf{Bip} is hence fully satisfactory, even though these criteria need now to be optimized.

6 Conclusion

This paper presented an experimental approach of locomotion gait analysis. We proposed a method for generating quickly and easily parameterized trajectories. This lead to the computation of various walking movements for the robot \textbf{Bip} ensuring static stability and aesthetic aspects. A controller was defined to track the pre-defined joint trajectories with a low sampling period. The programs were developed through the dedicated real-time environment \textbf{ORCCAD}. In particular, the use of \textbf{ORCCAD} simplified widely the control law implementation. As many as 8 gaits could thus be experimented on the robot. In any case the robot could walk without falling, despite rather strong perturbations. We proposed a comparative analysis of these gaits, showing the influence of the parameters on the energy consumption. A movie presenting these results was realized and presented at Hanover Expo2000. Our approach could be easily transferred to any biped robot, as long as the number of degrees of freedom is compatible with the output function dimension. Only the model and the physical characteristics have to be adjusted. Future work will extend our approach to a 3-dimensional version of the \textbf{Bip}.
robot, some simulation results have already been achieved [26].
The research team Bip is now working on new approaches of the gait con-
trol. The objective is to realize 3-dimensional dynamic gaits for a 15 active
joints prototype. Two main axes of research are explored. The first idea is
to generate a complete set of gaits, the robot will chose in this “library” the
best solution to deal at each instant with its actual situation [38]. An other
direction, inspired by human behavior, is the use of optimal control together
with model predictive techniques to avoid the use of reference gaits [3].

Acknowledgement - This work is the result of the collaboration of sev-
eral researchers and engineers. Hence, the authors would like to warmly thank
Bernard Espiau, Roger Pissard-Gibollet, Pierre-Brice Wieber, Gérard Baille
and Pascal Di Giacomo from INRIA as well as Philippe Sardain from LMS
for their precious help.
Annex: **ORCCAD Specification of the Bip Walk**

In order to implement our algorithms on the robotic platform, we used the ORCCAD software environment. In this section, we explain how we use ORCCAD to implement the walking control law presented in section 4.2.

### 6.1 The Robot Task Level

We decomposed the control law in modules as we would have decomposed it in regular C functions: each module has its inputs and outputs and performs a calculation that might be used again in another control law. Figure 16 shows the Robot Task corresponding to the walking gait control law based on trajectory tracking presented above.

- The Bip robot is represented through the controlled physical resource BIP depicted in figure 17. Input and output ports are configured by specifying data type and driver functions. Exceptions can also be easily defined through the GUI. Similarly, the sensors are handled by a non-controlled physical resource Sensors for which no input ports are required.
Joints/Actuators conversions in (24) are performed as presented in figure 18. Conversions converts joints torques into motors currents. Transmitters converts motors positions given by the encoders into joints positions using the variable reduction ratios. Limits raises an exception if one of the joint positions is over the joint limitation.

Support phase detection is performed through Sensors and support modules as presented in figure 19. The latter determines the robot support phase using the information given by the feet sensors, i.e. port force_sensor of the non controlled physical resource Sensors, and the rate $\lambda$ of (21).

Trajectory generation is performed by the module trajectories presented in figure 20. This module is linked to a file containing an array of joints desired positions and velocities. The module raises an event when the end of the trajectory is reached.

Trajectory tracking is specified as presented in figure 21. The control module computes the joint torques from (20). The model module computes the
6.2 The Robot Procedure Level

The main application corresponds to the sequencing of a Robot Task performing the initialization of the robotic system, and the gait control law...
Fig. 22. ORCCAD Robot Procedure for the BIP Walking Application

ROBOT TASK presented above. This initialization ROBOT TASK aims to start the robotic system and to identify the absolute start position of the BIP robot. Hence the ROBOT PROCEDURE corresponding to the application can be simply expressed as:

```
start RobotTask_Init
then
start RobotTask_Gait
```

This application is specified through the ROBOT Procedure written in ESTEREL language, given in figure 22.
References


26


