An Interactive Software Environment for Gait Generation and Control Design of Sony Legged Robots

Dragos Golubovic and Huosheng Hu

Department of Computer Science
University of Essex, Colchester C04 3SQ, UK
{dgolub,hhu}@essex.ac.uk

Abstract. This paper presents a modular approach to the development of an interactive software environment for gait generation and control design of Sony legged robots. A number of modules have been developed for monitoring robot states, gait generation, control design and image processing. A dynamic model of the leg and wheel-like motion are proposed to combine both wheeled and legged properties to produce smooth quadruped motion and high flexibility. Experimental results are presented to show the feasibility of the system

1   Introduction

Increased complexity and sophistication of advanced walking robots has led to continuing progress in building software environments to aid in the development of robust functionality. This is true not only because the physical construction of these robots is time consuming and expensive, but also because the evaluation and control of their gaits often requires prolonged training and frequent reconfiguration. To speed up the development cycle and decrease the design cost and time required for gaits generation, many software environments have been developed [2], [3]. The benefit of developing a suitable software environment includes the ability to record precise and voluminous data. Indeed, a flexible software environment plays an important role in many aspects of robotics research.

The main focus of this paper is the development of an interactive software environment for the design of a real-time control algorithm of AIBO football playing robots [4]. To make design and development of gaits easier, an interactive software environment has been developed at Essex. The software environment consists of three modules (state reflector, gait generation and vision) and gives a variety of useful features for the gait generation and development. By using a mouse or keyboard commands, an operator is able to record a sequence of movements that can be replayed in a sequence. It is also a very useful tool for debugging and evaluating quadruped gaits [5].

The rest of this paper is organized as follows. Section 2 describes the construction of an interactive software environment for the control design and gait generation of Sony AIBO robots. In section 3, the control design of Sony Legged

robots is presented, which includes the control system structure, a dynamic model of the leg and wheel-like motion. The experiment results are given in Section 4 to show the feasibility of the system. Finally, a brief conclusion and future work are presented in Section 5.

2   Building a Software Environment

The software architecture is a crucial aspect of Sony walking robots. The control software is ultimately responsible for managing the safe operation of the robot. In other words, the control software for a Sony walking robot must be carefully planned and constructed so that the derived gaits, however complex, conform to safety and efficiency specifications.

2.1   Modular Implementation

Since Sony AIBO robots have a large number of input and output parameters, their control design is complex. Therefore a modular approach is adopted here [1]. As shown in Figure 1, high-level control is conducted in a desktop PC (Pentium II 266) that is connected to the robot through a serial port (19200 baud). There are three main modules in it: a state reflector, a gait generator and an image interpreter. On the robot side a debug box has been mounted on the robot's back and connected to the PC via a cable. The next section describes these modules in more detail. Changes in any of these modules don't affect other modules, enabling users to split application development on several parts that can be carried out independently.

2.2   Module Description

Modular architecture gives provision for reconfiguration and extension, allowing the system to evolve with time, this is described in this section.

- □ **State reflector** – An internal state reflector has been incorporated to mirror the robot's state on the host computer, which is an abstract view of the actual robot's internal state, such as sensor information from the robot and control commands from the host computer. The state reflector is a set of data structures which allow the client to examine sensor information and control the robot by setting its values. In Table 1, CSensor holds basic sensor information sent from the robot and CPastReadings holds information about current and past sonar returns. The control commands for robot motions are listed in Table 2.
- □ **Communication routines** – The designed controller communicates with the robot using a handshake mechanism, and sends an appropriate command to the robot. The program executed on the robot waits for the command (Table 2) and executes it when the command has been received. After execution, the robot returns the result along with confirmation that data has been sent.
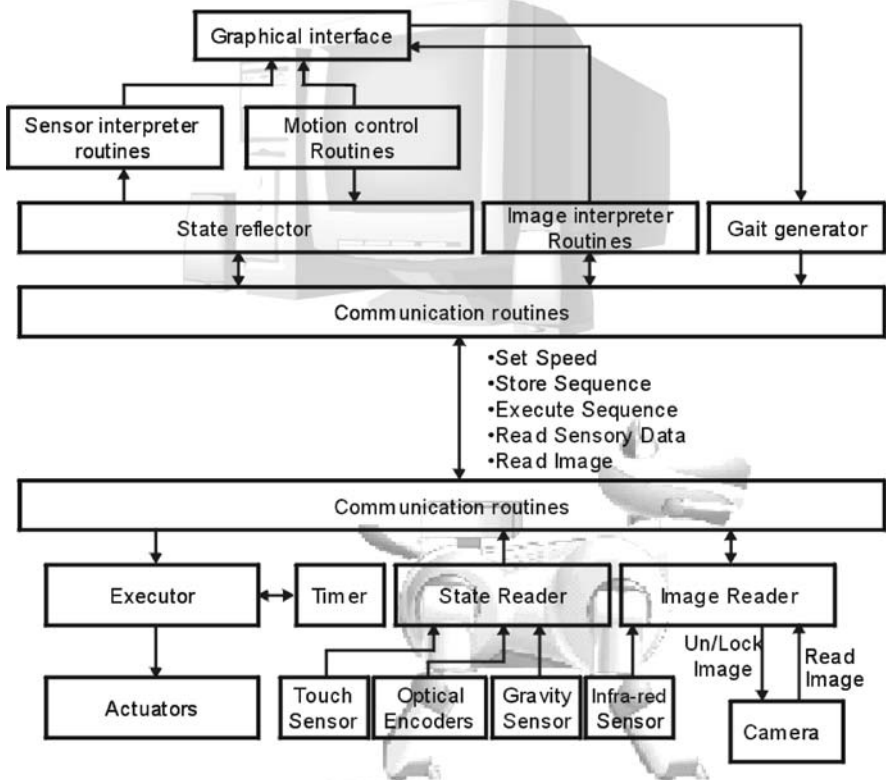
Fig. 1. Configuration of the proposed software environment

The amount of data transferred from the user's application to the robot and the other way around varies from one command to the other. A 19200-baud channel has been used.

☐ **Gait generator** – The gait generator communicates with the robot by passing a sequence of arrays that are transformed into a sequence of robot movements. It creates different gaits in a form of a sequence of arrays. In this software environment, users can move all robots' joints at the same time and record its movements in an array sequence, which can then be repeated and tested. Therefore users can create gaits and test various motions necessary for robot's mobility.

☐ **Image reader** – Gathering image snapshots and processing images can be done completely independently from the rest of the application. The size of a captured image is 144×76 pixels and each pixel has three bytes for colour information. Since transferring whole image through a 19200baud connection takes 45 second, an adjustable scale factor is added to reduce transfer time if necessary. A locking mechanism has been adopted to allow the transfer of the current image to be safely completed before the new snapshot image can be grabbed.

Table 1. State reflector data structure

CSensor				CPastReadings	
struct Leg	FRLeg			struct Leg *	FRLeg
struct Leg	FLLeg			struct Leg *	FLLeg
struct Leg	BRLeg			struct Leg *	BRLeg
struct Leg	BLLeg			struct Leg *	BLLeg
struct Head	head			struct Head*	head
struct Leg	tail			struct Leg*	tail
struct Gravity	gravity			struct Gravity *	gravity

Struct Leg	Struct Head	Struct Tail	Struct Gravity
Theta1 [degrees]	Pan [degrees]	Pan	X
Theta2 [degrees]	Tilt [degrees]	Tilt	Y
Theta3 [degrees]	Roll [degrees]	[degrees]	Z
TouchSen [true/false]	Mouth [degrees]		[degrees]

Table 2. Control commands

Command	Communication time	Size of transferred data
Motor on	10ms	1 byte
Read Data	100ms	74 byte
Send motion sequence	Sequence length $\times$ 100ms	Sequence length $\times$ 74 byte
Execute	10ms	1 byte
Set speed	50ms	8 byte
Read image	45sec/{scale factor}	76$\times$ 144 (byte/{scale factor}

3   Control Design

The key concept in this paper is based on movements of a four-wheeled car. Each leg tip is moving in a rectangle trajectory. Front and rear legs from the opposite sides are in the same phase and the other two legs are opposite. Projection of the mass of the robot to the ground is on the line that connects to legs, which are in touch with the ground. The centre of paw rotation is initially at the same inverse kinematic coordinate for both front legs and both rear legs. This prevents the robot from falling to the side and stabilizes the camera. Gravity sensors have been used to obtain information on body position.

3.1   Control System Structure

The control system structure consists of both kinematics and dynamics levels (Figure 2). The kinematics level involves two sub-levels: a pattern generator and a leg trajectory generator. Each leg has its own trajectory generator that determines the course of the leg endpoint. When a timing signal has been received, the leg must begin its swing/stance cycles. In order to emulate the accurate foot placement the trajectory generator plans a trajectory in foot position coordinates and then converts them to joint positions using inverse kinematics.
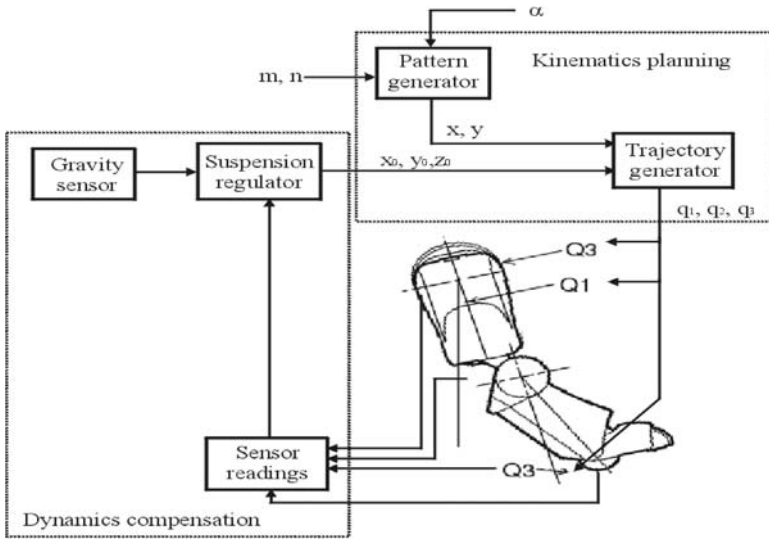
Fig. 2. Gait generation mechanism

The pattern generator provides repetitive motion of a leg and synchronization of movements with the other three legs. Gait planning depends on the velocity and heading of the robot. The time and space coordination of the motion involves a decision regarding which leg should be lifted or placed. It must be made in terms of the condition of terrain, stability requirements, speed requirements, mobility requirements and power consumption.

3.2   Dynamic Model of the Leg

Figure 3 shows the single robot's joint with its three motorized rotational axes. Forces applied to the leg differ whether the leg is on the ground or not. The set of 3 generalized coordinates $q[q1, q2, q3]$ is used to determine the mechanism position.

Each coordinate corresponds to one degree of freedom (DOF). There are two rotational segments upper limb (S1) and lower limb (S2). Angle θi is the relative rotation of the $i-th$ segment with respect to the $(i-1)-th$ segment around the axis. The dynamics equations are derived on the basis of D'Ambler's principle [6]. For the $k-th$ segment, we assume that $\vec{G_k}$ is its gravity force vector; $\vec{F_{kE}}$ is the resultant of other external forces acting on it; $\vec{M_{kE}}$ is the resultant of other external moments acting on it; $\vec{F_{k1}}$ is the resultant of the internal forces on it; $\vec{M_{k1}}$ is the resultant of the internal moments on it; $\vec{P_i}$ is the vector of the drive in the joint $Si-1$; $\vec{r}_{ij}=\vec{S_iC_j}$. In contrast, Cj is the centre of gravity of the j-th segment; if D'Ambler's principal of inertial forces is applied, we have

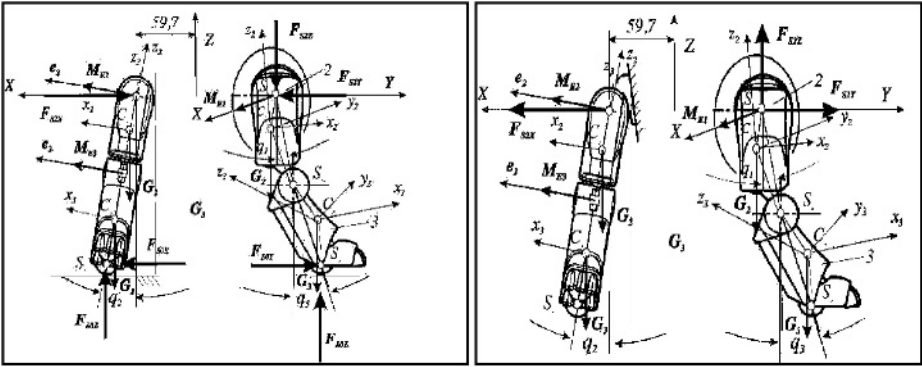**Fig. 3.** Dynamic model of a front-right leg on the ground (left) and in the air (right)

$$\vec{F}_{Sj} + \vec{P}_i + \sum_{k=1}^{3} \left(\vec{G}_k + \vec{F}_{k1} + \vec{F}_{kE} \right) = 0 \qquad (1)$$

If D'Ambler's principal is applied to the inertial moment relative to the $Si - 1$, then

$$\vec{M}_{Sj} + \vec{P}_i + \sum_{k=1}^{3} \left[\vec{M}_{k1} + \vec{r}_{i-1,k} \times \left(\vec{G}_k + \vec{F}_{k1} + \vec{F}_{kE} \right) + \vec{M}_{kE} \right] = 0 \qquad (2)$$

The system equations (1) and (2) can be transformed into the matrix form (3)

$$W(q)\ddot{q} = P + U(q, \dot{q}) \qquad (3)$$

P presents the column vector of driving forces and torques in the mechanism joints. The matrix W depends on the generalized coordinates q, and U depends on q and generalized velocity. The algorithm for computing W and U is derived from general theorems of dynamics and these matrices also depend on the configuration.

3.3   Generation of Wheel-Like Motion

A trajectory refers to both the path of the tip's movement of a limb (paw), and the velocity along the path. Thus, a trajectory has both spatial and temporal aspects. The spatial aspect is the sequence of the locations of the endpoint from the start of the movement to the goal, and the temporal aspect is the time dependence along the path.

Six posture parameters used for designing the gait, m, n, Xo, Yo, Zo and α, are common parameters for each leg; but they can differ between front and rear legs. If posture parameters for the front and rear legs are not identical, the top plane of the body will make δ angle with the ground. X, Y and Z coordinates of the paw are determined by the angular targets of a leg.

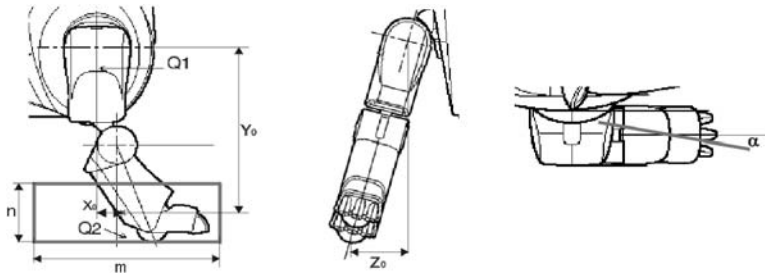**Fig. 4.** Pawn trajectory

$$\Theta_2 = \arcsin\left(Z/\sqrt{x^2 + y^2 + Z^2}\right) \tag{4}$$

$$\Theta_3 = 2\arccos\left(\sqrt{x^2 + y^2 + Z^2}/2l\cos\Theta_2\right) \tag{5}$$

$$\Theta_1 = arctg(y/x) - \arccos\left(\sqrt{x^2 + y^2 + Z^2}/2l\cos\Theta_2\right) \tag{6}$$

The body velocity depends upon the width of the elliptical paw trajectory (m), the duty factor χ and the cycle period.

$$V = m/x^T \tag{7}$$

An ideal duty factor can reach the value of 0.5 (when only two legs are on the ground at the same time). The maximum value of duty factor is 1. Equation 5 shows that the increase of the body velocity can be achieved by either increasing m or decreasing T. However, the excessive increase of m factor can lead to the increase of χ, because the duty factor depends on many parameters, which lead to poor performance.

A new suspension mechanism is adopted to adjust the height of a foot to terrain with vertical elevation. Suspension mechanism can prevent a leg's up and down motion during walking. That is, when a leg is up for swing forward, a suspension mechanism should stretch to its limit, and make the vertical stroke of a leg shorter.

4   Experimental Results

To evaluate gaits with different parameters $(m, n, X0, Y0, Z0, \alpha)$ and usefulness of the developed software environment, we tested a Sony AIBO robot on both a rough and a flat terrain. During its walk the robot has been connected to the PC running applications through a debug box via a PC's serial port. After several test runs, the following values of gait parameters achieved the best stability and the fastest speed: $M = 4$; $n = 3$; $\alpha = 15$; $X_0 = 0$; $Y_0 = 110$, $Z_0 = 20$. Each walking gait is tested separately to check the validity of parameters used.
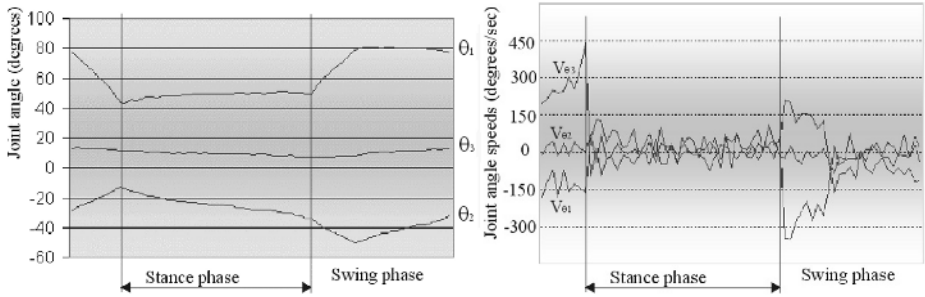
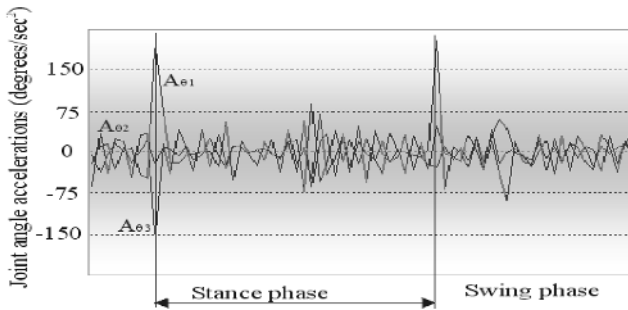**Fig. 5.** Joint angles (left) and speeds (right) during a sequence of step cycles

Fig. 6. Joint accelerations during a sequence of step cycles

The walking motion was obtained on flat ground and the readings from the optical encoders of the robot's joints at each step cycle. As can been seen in figures 5 and 6, targeted and achieved angles differ by an average value of 2.743 degrees. Difference is greater during a stance phase, which is understandable because during this phase the joints have to cope with the body weight of the robot and the reaction force from the ground. The average difference during stance phase is 3.56 degrees.

The effect of the suspension control was checked under dynamic walking. The robot runs over irregular terrain and the shoulder's height was sampled from gravity sensors, as shown in Fig. 7. Without the suspension mechanism the angle of the robot's top plane toward the ground becomes larger in accordance with terrain irregularity and duty factor while the suspension-controlled case is still small.

5   Conclusion and Future Work

A modular approach is adopted for the development of a useful software tool for gait generation and control design of Sony AIBO quadruped robots, which makes future improvement easy. A model of quadruped robot's gait is presented. Implementing wheel-like motions for legs reduces the mechanical complexity
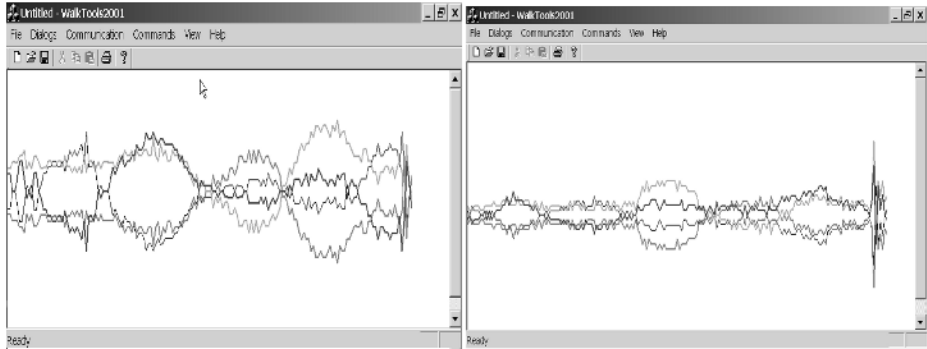
Fig. 7. Variations in shoulder's height without (left) and with (right) suspension respectively

intrinsic legged systems whist maintaining attractive performance. The designed gaits showed good results in speed, maneuverability and stability. Special care was given to maintaining stability. This allows the successful implementation of behaviours that rely on camera readings and is therefore important.

Current efforts are focused on the replacement of the analytical part of the suspension mechanism with neural networks. This allows shoulder height for each leg to be modified via trained neural networks in order to make AIBO robots more adaptive to ground roughness and improve the robot's stability.

Acknowledgements

References

1. P. Israel Doerchuk, W. Simon, V. Nguyen, "A Modular Approach to Intelligent Control of Simulated Joint Leg", IEEE Robotics & Automation Magazine, June 1998, pp. 12-20.
2. J. Reichler, F. Delcomyn, "Dynamics Simulation and Controller Interfacing for Legged Robots?", Int. J. of Robotics Research, Vol. 19, No. 1, 2000, page 42-58.
3. M. Maza, J. Fontaine, M. Armada, P. Gonzalez, "Wheel+Legs- A New Solution For Traction Enhancement Without Additive Soil Compaction", IEEE Robotics and Automation Magazine, June 1998, pages 26-32.
4. M. Fujita and K. Kageyama, "Development of an Autonomous Quadruped Robot for Robot Entertainment", Journal of Autonomous Robots, Vol. 5, pages 1-14, 1999.
5. R. Reeve and J. Hallam, "Control of Walking by Central Pattern Generators", Journal of Intelligent Autonomous Systems, 1995, pages 695-701.
6. M. Vukobratovic and V. Potkonjak, "Dynamics of Manipulation Robots. Theory and Applicaton", Communication and Control Engineering Series, Springer-Verlag, 1982.