DEVELOPING DYNAMIC GAITS FOR FOUR LEGGED ROBOTS

Makoto Toyomasu and Ayumi Shinohara

Department of informatics, Kyushu University
Hakozaki 6-10-1, Fukuoka 812-8581, Japan

ABSTRACT

This paper presents our development of dynamic gaits for four legged robots. To make design and development of gaits easier, we developed an interactive tools. 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.

1. INTRODUCTION

We are challenging to the Sony 4-Legged Robot League of RoboCup Soccer. RoboCup is an international competition of robots held every year. 4-legged League is one of the events of this competition. As the name of the league shows, briefly speaking, the teams compete with each other in the soccer matches in this league. However, the players are not human beings but autonomous legged robots. It is provided that SONY/AIBO ERS-210 is used as players. The teams develop their programs that run the robots autonomously, that is, without remote control. Each ERS-210 has 3 degrees of freedom (3DOF) on its head and four legs. There is also 2DOF for the tail and 1DOF on its mouth. The body length (not including the head or tail) is approximately 18cm and the length of the leg (from shoulder to foot) is just under 12cm. On its head, there are a micro-camera, stereo microphone, infrared range sensor, and touch sensor. There is also a touch sensor at the bottom of each foot. In addition to embedded CPU, its body houses a gyroscope and two accelerometers.

Among many aspects of our development for RoboCup Soccer, this paper focuses on the dynamic gaits for four legged robots, which is the most fundamental factor to achieve good performance. Although some ideas were inspired by the reports and programs of advanced teams [1, 2, 3, 4], we have dealt almost everything from scratch.

2. DEVELOPING ENVIRONMENT

We use OPEN-R Software Development Kit (OPEN-R SDK) [5] for programming the robots. OPEN-R is the standard interface for the entertainment robot system that Sony is actively promoting, and the OPEN-R SDK is the cross development environment based on gcc (C++) where we can make software that works on AIBO. It provides us a low level Application Program Interfaces (API), such as making joints move, getting information from sensors, getting image from camera, and using wireless LAN via TCP/IP.

The development process typically consists of the following steps: (1) writing a program in C++ language on PC running Windows or Linux, (2) compile it using the cross-compiler to the binary for ARM processor, (3) copy it to a Memory Stick, and (4) slot it into the robot, and switch on the robot. Then the robot starts moving after a slow booting process (about 1 minutes). Some debugging output can be displayed at the console on PC through wireless LAN. However, the program easily crashes because of bugs before sending the output to PC, so that the debugging is a considerably heavy task, compared to debug a program running on PC itself. Moreover, the cross-compiling is also time consuming. For example, it takes 6 minutes 25 seconds to compile the whole program of JollyPochie [6], which we have developed for RoboCup2003, on a PC with Pentium III 866MHz, 504MB RAM, running WindowsXP. We note that the size of the program is rather smaller than those by advanced teams.

Therefore, we need to develop some software tools to support our programming and debugging tasks, as well as the programs themselves running on the robots.

3. INVERSE KINEMATICS

OPEN-R SDK provides us only low level functions. For example, we have to specify twelve angles (three joints for each leg) for every 8 micro-seconds movement. No higher level functions, such as standing up nor moving forward are given. Thus we have to compute everything, in order to make the robot moved meaningfully, such as walking around and kicking a ball.

In our program for RoboCup, we classified all motions into two groups. One is fixed motions, such as shooting a ball and dancing, and the other is parametric motions, typically used for walking for arbitrarily specified direction with an adjustable speed. The rest of the paper will deal with the second group of the motions.

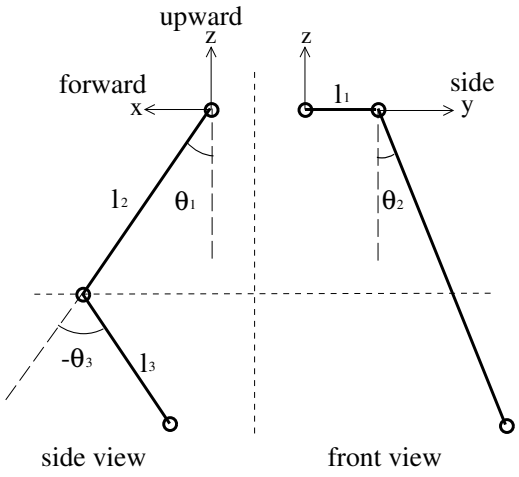


Fig. 1. Model and coordinate frame for leg kinematics

Let θ_1, θ_2, and θ_3 be angles for rotator, shoulder joint, and knee joint, and let l_1, l_2, and l_3 be the lengths of shoulder, upper limb, and lower limb (Fig. 1). Then the position (x, y, z) of the paw is represented by the following formulae:

$$\begin{aligned} x &= l_3 \sin\theta_3 \cos\theta_1 + (l_2 + l_3 \cos\theta_3) \cos\theta_2 \sin\theta_1, \\ y &= l_1 \qquad\qquad + (l_2 + l_3 \cos\theta_3) \sin\theta_2, \\ z &= l_3 \sin\theta_3 \sin\theta_1 - (l_2 + l_3 \cos\theta_3) \cos\theta_2 \cos\theta_1. \end{aligned}$$

On the other hand, we need their inverse, in order to allow a high-level description of the gait, in which we design the movement of each leg by its position of pow, instead of these angles. Since our robot has 3 degrees of freedom, we have the following closed-form inverse kinematics:

$$\begin{aligned} \theta_3 &= \pm\cos^{-1}\left(\frac{x^2 + z^2 + (y - l_1)^2 - (l_2^2 + l_3^2)}{2 l_2 l_3}\right), \\ \theta_2 &= \sin^{-1}\left(\frac{y - l_1}{l_2 + l_3 \cos\theta_3}\right), \\ \theta_1 &= \tan^{-1}\left(\frac{x}{-z}\right) \mp \cos^{-1}\left(\frac{(l_2 + l_3 \cos\theta_3)\cos\theta_2}{\sqrt{x^2 + z^2}}\right). \end{aligned}$$

Remark that we have two solutions for any reachable position (x, y, z), depending on the angle θ_3 of the knee is either positive or negative. Moreover, if $x = z = 0$ then θ_1 can be arbitrarily chosen.

4. LEG MOTION

In order to create wheel-like leg motions, we specify the following parameters.

landing position and *leaving position*: positions at which the leg reaches the ground and leaves from it.

Lift height: lift heights of the leg.

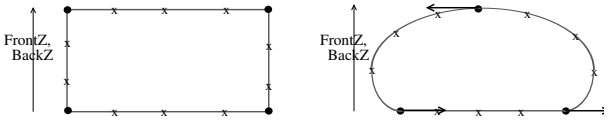
Stroke shape: We can choose either rectangle or Hermite curve interpolation of three points, shown in Fig. 2. Hermite curve smoothly connects between specified points.

These parameters determine the spatial trajectory of the paw. Moreover, additional two parameters determine the position (x, y, z) of the paw at each time step.

power ratio: time ratio of the paw touches the floor.

time period: total steps of one stroke.

While the paw touches the floor, it moves on the straight line at a fixed speed.



(a)rectangle (b)hermite curve

Fig. 2. stroke shapes

5. DESIGNING GAITS

Every gait in our locomotion system is specified by the following parameters.

foot positions: positions at which each leg reaches the ground and leaves from it.

FH, BH: heights of front and back body. They determines the position and posture of the body. $(50mm \leq FH, BH \leq 150mm)$.

β: power ratio, defined by the time ratio of each paw touches the floor. $(0 \leq \beta \leq 1.0)$.

ϕ: phase difference between the front left leg and the rear right legs. $(0 \leq \phi \leq 1.0)$.

α: phase difference between the left and right legs. $(0 \leq \alpha \leq 1.0)$.

Phase: period of one stroke.

Stroke type: shape of the stroke, either rectangle or Hermite curve.

FrontZ, BackZ: lift heights of front legs and rear legs. $(0mm \leq FrontZ, BackZ \leq 35mm)$.

By changing the parameters α, β, and ϕ, we can realize each gait such as walking and trotting (including pace) in Fig. 3 [7]. Galloping is unfortunately infeasible due to the lack of powers of motors relative to the weight of the body.

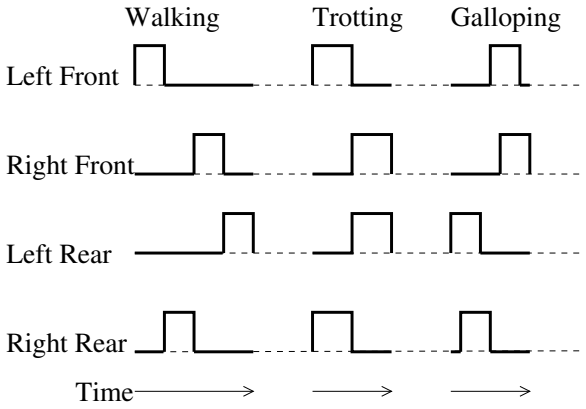


Fig. 3. A timing diagram of various gaits. The function is zero when the foot is on the ground and nonzero when it is in the air.

From these parameters, we compute all angles of the joints in four legs by inverse kinematics, described in the Section 3. It is a quite time consuming task to select the best set of the parameters, which requires a lot of trial and error. As described in Section 2, even one trial might be cumbersome, if we do it in a naive way. In order to reduce these tasks, we developed an interactive software tool (Fig. 4), which can change values for all parameters in the GUI window executed in PC. These values will be transmitted to the robot via TCP connection, and change the gaits immediately. Moreover, we have developed a visualizer of the gaits, which shows the movement of the four legs in 3D-animation at the screen of PC (Fig. 5). We implemented it in python programming language [8], with using the visual module [9]. These were very useful tools for designing, debugging and evaluating gaits. For the game, we chose a stable, fast and low stance gaits.

6. COMPOSITE WALKING

For the direction of walking the four legs are more or less treated as wheels. Turning is done by moving each leg in a different diagonal direction. Walking and turning can also be combined resulting in curved walking. This is done by simple vector addition of three components (forward, sideways, turning) for each leg. By vector addition we can drive

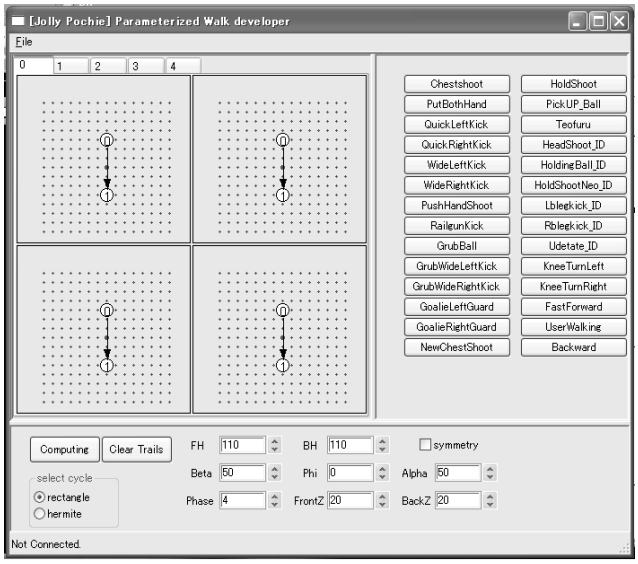


Fig. 4. Gaits developing tool

the robot in any direction with different orientations. Forward walk, right sideways and right turning are defined positive vector. Backward walk, left sideways and left turning are defined negative vector. Composition is made within step size. Max step size is shown in the figure 7.

7. PERFORMANCE

Using this dynamic gaits developed by our tools, we succeeded to get a semi-dynamic trotting gait with a maximum walking speed of 224 mm/sec forward or backward, 200 mm/sec sideways, or 2.2rad/sec turning. As a result, we took second place (first place was ASURA team) in the AIBO race at the OPEN-R techno-forum held in Fukuoka in August 2003.

8. CONCLUDING REMARKS

We have developed these tools and programs, which are the base for playing soccer by the robots. As a initial step of our challenge, we are rather satisfied with the performance. However, apparently we need more efforts to compete with the advanced teams. We describe our future plan on the gaits in the sequel.

When we drive the robot, some parameters that control the walk are hidden to simplify the interface to the locomotion routines. The three parameters that are visible are: Forward, Sideways and Turn components for the walk. By manipulating these parameters we can drive the robot in any direction with different orientations. Our odometry estimates the displacement and orientation based on these values. If

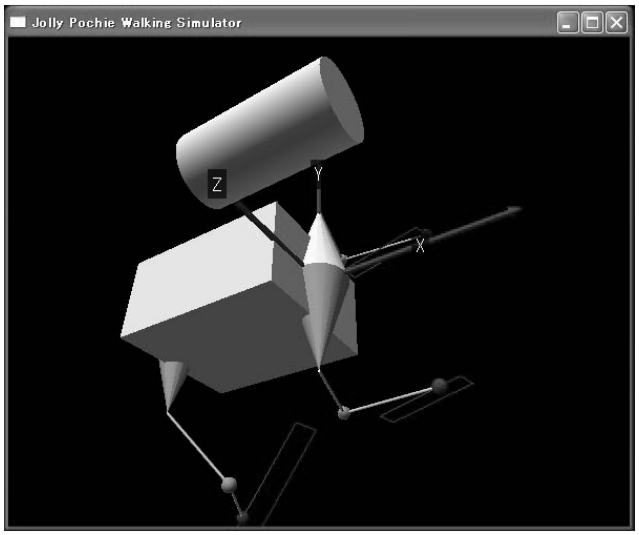


Fig. 5. Visualizer of the gaits

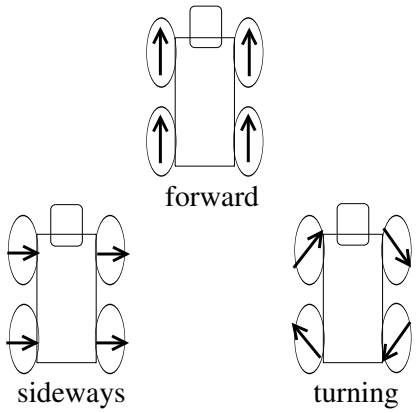


Fig. 6. Three basic components of walking

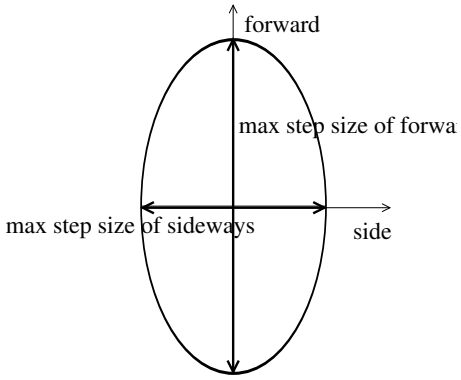


Fig. 7. Max step size of a leg

we change any parameters that affects the walking style, the odometry may not be able to provide an accurate estimation for the displacement and orientation. Therefore, odometry must be recalibrated for different sets of parameters.

When we request the robot to walk straight forward, it may turn a bit after a few steps, this effect introduces a slight error in the odometry. The direction of turn varies with different robots, we suspect that it is due to the differences in weight distribution, motor strength, etc., that make it hard to correct by adjusting the requested mapping. The error becomes serious when a motion contains all of these three components. It is quite troublesome to adjust odometry manually, because it also depends on the surface of the floor. In order to reduce the burden to adjust odometry, we are now developing software tools which support semi-automatic learning of these parameters from many trials.

9. ACKNOWLEDGEMENTS

The authors thank to our teammates of JollyPochie, for fruitful discussions and cooperation to develop the system.

10. REFERENCES

[1] Manuela Veloso et al., "CMPack-02: CMU's Legged Robot Soccer Team," Tech. Rep., 2002.

[2] Andre Olave et al., "The UNSW RoboCup 2002 Legged League Team," Tech. Rep., 2002.

[3] Takeshi Kato et al., "The Kyushu United Team 2002 in the Four Legged Robot League," Tech. Rep., 2002, <http://www.asura.ac/>.

[4] Burkhard et al., "German Team 2002," Tech. Rep., 2002, <http://www.robocup.de/germanteam/>.

[5] Sony Corporation Entertainment Robot Company, "Open-R," <http://openr.aibo.com>.

[6] Ayumi Shinohara et al., "JollyPochie —team for robocup soccer 4-legged robot league—," <http://www.i.kyushu-u.ac.jp/JollyPochie/>.

[7] Sandro Boccuzzo and Daniel Steiner, "Artificial and natural walking machines: Rapid Locomotion," Research note, Seminar Artificial Intelligence, 2002.

[8] Guido van Rossum, "Python," <http://www.python.org>.

[9] David Scherer, "Visual python," <http://www.vpython.org>.