Trot Gait Design Details for Quadrupeds

Vincent Hugel, Pierre Blazevic, Olivier Stasse, and Patrick Bonnin

Laboratoire de Robotique de Versailles (LRV)
10/12 avenue de l'Europe, 78140 Vélizy, France
hugel@robot.uvsq.fr

Abstract. This paper presents a full description of the design of a trot locomotion that has been implemented on AIBO quadrupeds in the Sony legged league. This work is inspired by the UNSW achievements in RoboCup 2000 and 2001 in Melbourne and Seattle. The French team rebuilt a complete trot locomotion from scratch, and introduced special features that differ from the Australian original design. Many papers have already been dedicated to the work on quadruped locomotion [2–4]. However they do not detail all the parts of the design.

1 Introduction

This paper is dedicated to the design from scratch of a trot gait that allows combining linear and angular body velocities, which can make the robot walk along a sideways direction or make turns round any point. From 1999 to 2001 the French team developments in locomotion [1] gave good results but motion turns were only possible round points located on the transverse axis of the body. BY the end of 2001 it was decided to design a new trot gait algorithm to allow better maneuverability of the robot and to incorporate the option of performing a crawl-like trot introduced with success by the New South Wales University team in RoboCup 2000. The algorithms developed by the French team are inspired by the details given by the UNSW team in its 2000 RoboCup report [2] but have been rewritten and adapted to include specific features. These features include exact support phase leg trajectories that are not approximated to straight line segments, leg home positions that vary with linear body velocity to improve dynamic balance, and times of changing velocity at leg landing.
Thanks to this locomotion design, the robot should be able to adopt a high-legged trot to cover large distances at high speed, and to adopt a crawl-like gait that is very efficient for approaching and covering the ball. This paper describes the parameters used for locomotion and the details of the design.

2 Parameters Used to Tune Locomotion

2.1 Leg Trajectory Placement Parameters

Leg trajectory placement parameters permit to place the leg trajectory with respect to the body reference frame. We distinguish the horizontal parameters, the α parameter for shifting home position, and the vertical parameters.

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 495–502, 2004.

Horizontal Distance Parameters. Figure 1 shows longitudinal and lateral spacings, and right and left shifts from geometric body centre (GBC) called ΔC_{left} and ΔC_{right}. Longitudinal spacings are taken from the limit defined by the GBC shifts.

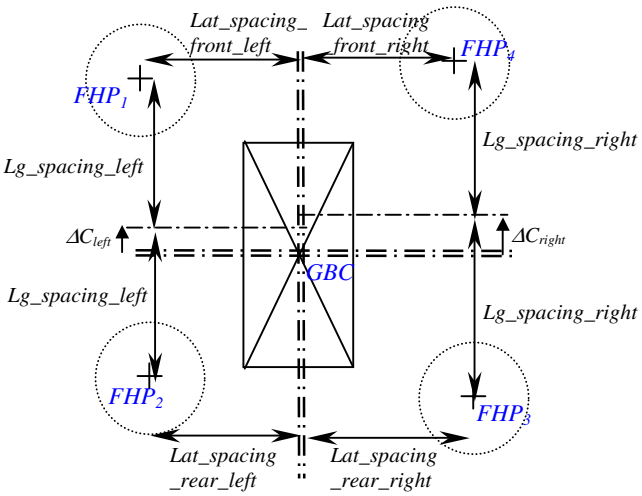

Fig. 1. Horizontal distance parameters that define fixed home positions for legs in stance phase.

Fixed home positions that serve as references are called $FHP1$, $FHP2$, $FHP3$, and $FHP4$. They are defined by the horizontal distance parameters.

Home Position Shifting Parameter. To implement a variable home position called CHP (see fig. 2) the parameter called α is introduced.

The home position shift from the fixed home position (FHP) is expressed as:

$$\overrightarrow{(FHP, CHP)} = \alpha \int \mathbf{v}.dt = \alpha.(\mathbf{v_i} + \mathbf{v_f})/2.T/2 \qquad (1)$$

where $\mathbf{v}$ is the linear velocity, T the cycle period, and $\mathbf{v_i}$ and $\mathbf{v_f}$ the linear velocities at the beginning and the end of the step.

This parameter allows to better distribute the load of the body between the two thrusting legs whatever the motion direction. Hence we get a CHP point for every leg. The CHP point is the middle point of the leg straight-line trajectory in support phase. This straight line joins landing and take-off points of the leg.

Vertical Parameters. They are the height of the front legs and the height of the rear legs with respect to the related shoulder joint.

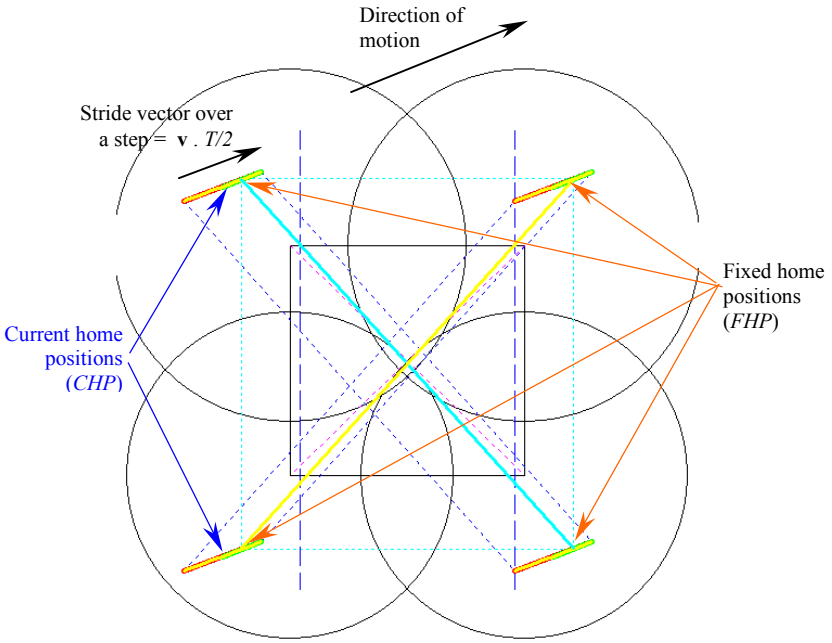

Fig. 2. Top view of the robot achieving a linear sideways motion. Velocity of the robot has two components v_x and v_y. The dotted rectangle joins the fixed home positions (FHP) as they are defined by the trajectory parameters (section 2.1). When a linear motion is required the home position varies as a function of the stride length (that is the velocity magnitude times the period of the stance phase ($T/2$)) and the direction of motion. The variable home positions are called CHP (current home positions). Here they are the middles of the linear stride segments.

2.2 Kinematic Parameters

They describe the shape of the leg trajectory over a cycle period. The kinematic parameters are the following:

- step period = $T/2$, where T is the total time of the leg cycle,
- upward phase height,
- downward phase height,
- time ratio of upward phase with respect to total air phase,
- time ratio of downward phase with respect to total air phase,
- maximal accelerations for v_x,v_y and w.

The upward phase starts when the leg leaves the ground. It terminates when the leg starts its swing phase. The downward phase starts when the leg ends the swing phase. It terminates when the leg touches ground. By varying maximal accelerations, we can vary the time of acceleration/deceleration of the robot.

2.3 Additional Parameters for Oscillation Moves

The oscillation moves that can be introduced are the following:

- up and down moves of the body,
- lateral moves of the body (right and left),
- angular moves round the transverse axis of the body.

The relating parameters are the amplitude and the phase lag of each oscillation move. The phase lag is defined with respect to the time cycle of a leg trajectory.

3 Design Details

3.1 Velocity or Position Control

The decision making module of the robot can run velocity or position control, and switch between both when needed.
Velocity control allows to request for a set of combined velocities, v_y, v_x, and w. Position control allows to request for a set of combined displacements (Dx, Dy, $D\theta$). This means that after a series of steps, the body centre should have moved by the displacement vector (Dx, Dy), and the body should have turned an angle equal to $D\theta$. In position control mode, it is possible to specify maximal velocities during the required displacement.

3.2 Adjusting Velocities to Robot's Leg Workspaces

In velocity control the velocities to be adjusted are the velocities the robot must reach. In position control the velocities to be adjusted are:

- the maximal velocities the robot should not exceed along the displacement required. They are given by the decision module.
- the successive velocities the robot must use to achieve the required displacements. These velocities must converge towards zero.

• **Use of a "workspace" calibration table.** For adjusting velocities a calibration table taking leg workspace boundaries into account is used. This calibration table can be illustrated by a 3D space where point coordinates represent body displacements $(\Delta x, \Delta y, \Delta\theta)$ over a step. A triplet of constant velocities (v_x, v_y, w) is linked to the displacements $(\Delta x, \Delta y, \Delta\theta)$ by

$$(v_x, v_y, w) = \Delta T_{step}.(\Delta x, \Delta y, \Delta\theta), \tag{2}$$

where $\Delta T_{step} = T/2$.

In the $(\Delta x, \Delta y, \Delta\theta)$ 3D space, we can calculate a closed 3D surface whose points represent the optimum set of triplet $(\Delta x, \Delta y, \Delta\theta)_{opt}$ the leg can achieve while remaining inside its workspace.

In other terms, components of points located inside the closed surface are displacements $(\Delta x, \Delta y, \Delta\theta)$ that can be achieved by the robot over one step.

Hence if the set of velocities required is located out of the 3D surface, the robot cannot achieve them. They must be reduced. Hence the decision module commands the robot to reach the optimal triplet of velocities given by

$$(\Delta x, \Delta y, \Delta\theta)_{opt}/\Delta T_{step}, \tag{3}$$

where $(\Delta x, \Delta y, \Delta\theta)_{opt}$ is defined by

$$(\Delta x, \Delta y, \Delta\theta)_{opt} = k.\Delta T_{step}.(v_x, v_y, w). \tag{4}$$

where k is a reduction factor.

Directions of vectors $(\Delta x, \Delta y, \Delta\theta)$ and (v_x, v_y, w) remain the same.

Of course the triplet of velocities remains unchanged if inside the 3D surface. The 3D surface can be computed at boot time using the trot parameters set.

• **How the workspace calibration table is built.** The calibration table contains 3D points of the 3D surface. The table is built using 3D discrete spherical coordinates. Starting with $(0, 0, 0)$ the number of points explored grows radially until the surface is reached. Beyond this surface the set of displacements cannot be achieved over a step.

Given a triplet of displacements $(\Delta x, \Delta y, \Delta\theta)$ over a step, it is marked as being inside the 3D surface if landing and take-off points belong to the workspace of the leg in support. Using this technique $(\Delta x, \Delta y, \Delta\theta)$ is adjusted automatically, and the decision module does not have to set a stride parameter for the gait.

In summary, the 3D surface ensures that leg trajectories will not go beyond the leg working space. And adjusting ΔT_{step} to a minimal value permits to set maximal forward velocity.

3.3 Update Velocity at Next Step or After Next Step

Once a pair of legs is on the ground, the velocity profile that has been planned cannot be changed all along this leg pair support phase.

When the request comes in the first half of the current step, new velocities are taken into account at starting the next support phase (i.e. the next step). When the request comes in the second half of the current step, new velocities are taken into account at starting the after next support phase (i.e. after the next step). This is because we consider that we do not have enough time to re-plan the stance phase trajectory of the current legs in the air.

• **Defining velocities at the end of a step.** A step is defined by an initial and a final triplet of velocities (see section 3.5 for generating support stance phase), and by the landing point. Knowing the initial velocity of the next step and before starting it, the problem is to get the final velocity to define the step completely. This final velocity must be known in advance ($T/4$ before starting

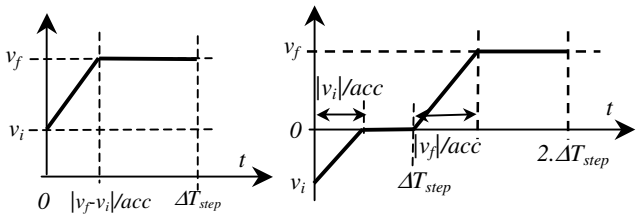

Fig. 3. Linear velocity profiles used to deal with velocity control. v_i is the velocity at the beginning of the step. v_f is the velocity at the end of the step.

the next step at least). All three velocities (v_x, v_y, w) follow a linear profile (see figure 3).

The rate of increase in velocity is defined by a maximal acceleration that is fixed. There is a different acceleration for every velocity v_x, v_y, and w. They have been set so that the robot can reach maximal velocity from still motion in a step time period. This implies: $\Delta T_{step} > |v|/acc$, where v represents whatever velocity, and acc the related acceleration (positive value).

• **Defining velocities in position control.** In position control, the goal is different. Here we do not want the robot to reach a certain set of velocities but we want it to halt motion after trotting the requested distance and turning the requested angle. And to allow re-entrance of velocity control with position control and conversely, position control is designed by updating velocities for every leg switch, like in velocity control. Velocities must converge towards 0 since the robot must halt motion once distances and angle have been achieved.

Let be $(Dx, Dy, D\theta)$ the displacements to achieve in position control. Velocities are updated 2 times a cycle period in the middle of two consecutive steps. At these times of update, the remaining displacement $(Dx, Dy, D\theta)_{remain}$ to perform is calculated The displacements already achieved are simply subtracted from the previous displacements.

Velocities are initialized to $(Dx, Dy, D\theta)_{remain}/\Delta T_{step}$ and filtered using workspace 3D surface (see section 3.2). Then velocities are optimized in order to finish the displacements as soon as possible. In case the initial velocity is too high and distance requested to be covered too small, it can appear that the robot oversteps the point to reach. But even so, the position control algorithm makes the robot joint the point by making it trot in the reverse direction.

3.4 Updating the Landing Point for the Next Step or the after Next Step

Knowing velocities at the beginning and the end of the step considered (i.e. velocities at leg landing and take-off) and by intergrating the velocity profile described in the previous section, we compute the displacements $(\Delta x, \Delta y, \Delta\theta)$ over this step. Then landing point coordinates are computed the following way: if $(\Delta x, \Delta y)$ stands for the linear displacement over a step (legs numbered 1 and 3 for instance) and $\Delta\theta$ for the angle to rotate, we have A and A' respectively

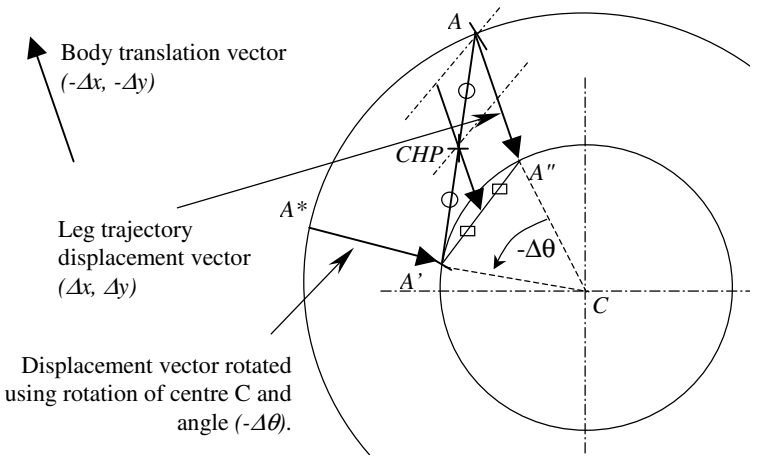

Fig. 4. A are $A^{'}$ are the landing and take-off points of the step. The body turns round C (GBC shifted) with an angle $\Delta\theta$. Hence the leg tip turns (-$\Delta\theta$). If we start with the translation, we get to $A^{'}$ by passing through $A^{"}$. If we start with the rotation, $A^{"}$ is the image point obtained by rotation of leg landing point A. Then the translation of vector $A^{*}A^{'}$ must be performed.).

landing and take-off points of the first leg (1) of the diagonal pair (figure 4), and B and B' referring to the second leg (3), expressed as:

$$\begin{aligned} CHP_1 &= FHP_1 + \alpha(\Delta x, \Delta y) \\ CHP_3 &= FHP_3 + \alpha(\Delta x, \Delta y) \end{aligned}$$

$$\begin{aligned} A_x &= \Delta x/2 + CHP_{1x} - (-\Delta y/2 + CHP_{1y}).tan(\Delta\theta/2) \\ A_y &= \Delta y/2 + CHP_{1y} + (-\Delta x/2 + CHP_{1x}).tan(\Delta\theta/2) \\ A_x^{'} &= -\Delta x/2 + CHP_{1x} - (-\Delta y/2 + CHP_{1y}).tan(\Delta\theta/2) \\ A_y^{'} &= -\Delta y/2 + CHP_{1y} + (-\Delta x/2 + CHP_{1x}).tan(\Delta\theta/2) \end{aligned} \tag{5}$$

$$\begin{aligned} B_x &= \Delta x/2 + CHP_{3x} - (-\Delta y/2 + CHP_{3y}).tan(\Delta\theta/2) \\ B_y &= \Delta y/2 + CHP_{3y} + (-\Delta x/2 + CHP_{3x}).tan(\Delta\theta/2) \\ B_x^{'} &= -\Delta x/2 + CHP_{3x} - (-\Delta y/2 + CHP_{3y}).tan(\Delta\theta/2) \\ B_y^{'} &= -\Delta y/2 + CHP_{3y} + (-\Delta x/2 + CHP_{3x}).tan(\Delta\theta/2) \end{aligned}$$

These equations refer to figure 4 that describes how landing and take-off points of the first leg of the diagonal support are defined knowing translation and turn-in-place motions of the body $(\Delta x, \Delta y, \Delta\theta)$ over the step.

3.5 Computing Current Cartesian Point of Leg Trajectory

The reference frame considered is the body reference frame whose plane xy is horizontal and whose centre is C (C is the point shifted from the geometric body centre GBC by ΔC along the longitudinal axis).

• **Stance phase.** At every time interval of the stance phase displacements $(dx, dy, d\theta)$ performed from time of landing until present time are calculated by integrating the velocity profile, knowing velocities at the beginning and the end of the step.

Then the stance point is calculated using these displacements by:

$$\begin{aligned} cpx &= LandingPoint.x - dx \\ cpy &= LandingPoint.y - dy \\ current\_stance\_point.x &= cos(d\theta).cpx + sin(d\theta).cpy \\ current\_stance\_point.y &= -sin(d\theta).cpx + cos(d\theta).cpy \\ current\_stance\_point.z &= front\_height\_or\_rear\_height \end{aligned} \tag{6}$$

• **Air phases.** They are implemented using the heights to move the leg up and down, and the next landing point.

4 Conclusion and Perspectives

Trot parameters can be adjusted easily by updating the calibration table. Hence it is possible to implement high-legged trot or the trot introduced by the UNSW team where the fore part of the body is lowered. However the whole locomotion is based on the trot gait. And it would be great to be able to switch between trot gait and other gaits such as crawl gaits that feature always three legs on the ground. Crawl gaits can be useful for approching maneuvers when slow and precise moves are required. In addition maybe gaits faster than trot can be introduced. These gaits involve the control of ballistic phases. New studies will focus on how to switch between trot, crawl and faster gaits continuously.

References

1. Hugel, V.: On the Control of Legged Locomotion Applied to Hexapod and Quadruped Prototypes, PhD Thesis (in French), Université de Paris VI, Nov. 1999.
2. Hensgt, B., Ibbotson, D., Bao Pham, S., Sammut, C.:The UNSW United 2000 Sony Legged Robot Software System, School of Computer Science and Engineering, University of New South Wales, 2000 RoboCup SONY legged league report.
3. Veloso M,Lenser S.,Vail D.,Roth M.,Stroupe A. and Chernova S.: CMPack-02: CMU's Legged Robot Soccer Team, Carnegie Mellon University, 2002 RoboCup SONY legged league report.
4. Upenn, IROS 2001 Hawai. Chitta, S., and OStrowski, J.P.: New Insights into Quasi-static and Dynamic Omnidirectional Quadrupedal Walking, IROS 2001.