Rollover as a Gait in Legged Autonomous Robots: A Systems Analysis

Vadim Kyrylov[1], Mihai Catalina[2], and Henry Ng[2]

[1] Rogers State University; Claremore, OK 74017 USA
vkyrylov@sfu.ca
[2] Simon Fraser University – Surrey; Surrey, British Columbia V3T 0A3 Canada
mscatali@sfu.ca, henry_ng@alumni.sfu.ca

Abstract. Rollover has been normally regarded as an undesirable way of legged robot locomotion. In spite of this, we are looking for the improved robot movement by combining the rollover with regular gaits. By considering gaits on the macro level, we identify the conditions when adding rollover may result in a faster movement of a legged robot. Our method is generic because the number of legs does not matter; we are taking into account only the features and constraints shared by all legged robots. As the outcome of this study, we propose a mathematical model for estimating the efficiency of using rollover as additional gait. This model is illustrated by evaluating the speed gain achieved in 4-legged Sony Aibo ERS-7 robot if it is equipped with a rollover skill. While having in our implementation the same linear speed as walking, rollover in some situations eliminates the need to make turns, thus saving time for the robot to change its pose.

1 Introduction

Legged robots are designed to walk using different gaits, i.e. particular manners of moving on their feet. We want to relax this constraint by allowing the robot actively touching the floor with any part of its body, neck, head, and legs while moving by rolling over. The main hypothesis of this study is that, by combining the rollover with regular gaits, in some conditions a robot can move faster. We determine these conditions and propose a mathematical model for planning robot movements as a combination of walking and rollovers. We also use this model for evaluating speed gains using the 4-legged Sony Aibo robot as an example. Still our method is generic because the number of legs does not matter; we are taking into account only the features and constraints shared by all legged robots.

The term 'rollover of an artificial object' normally implies something negative; rollovers in most cases should be avoided. In particular, motor vehicle developers are doing their best to prevent rollovers. Same attitude appears to exist in the legged robot developer community; by design, rolling over in legged robots is regarded an unsuitable way to move [1, 2]. However, for the appropriately shaped robots rollovers may be deemed suitable. For example, in [3], this 'gait' was studied for the purpose of locomotion of a cylinder-shaped robot. One more recent study of rollover has been conducted for humanoid robots [4]. In this case, this mode of locomotion was used for the sole purpose of rising of the robot after it falls down. Still in general for the legged robots rollover as a gait has not been systematically studied as yet.

Unlike the works [1-4] that concentrate on the micro level design of the robot locomotion, in this paper we are taking a higher-level view of different gaits; hence '*A Systems Analysis*' in the title. Our main objective is to create a mathematical model that allows estimating the time needed to change the robot pose as the function of the ordered set of applied gaits. The sequence of gaits and their parameters are the controlled variables. Then we solve the discrete optimization problem by minimizing time with and without rollover and measure the time gain.

Although this study was started with 4-legged robots in mind, we make assumptions neither about the number of legs nor do we look into the details of leg motions. Rather, we consider the robot gaits on the macro level; they are dashing, pulling, sidestepping, turning, and optionally rolling over. To make our approach applicable to legged robots, we impose some constraints on the way how these gaits can be combined; we believe that these constraints are specific to most, if not all, legged robots. We view the robot motion planning task as deciding on what sequence of available macro level gaits to apply for making the desired change to the robot pose. For the purpose of planning, we are using generalized approximations of the gaits, each such sub model containing very few parameters that are easy to estimate in given robot.

The objective of this study is to show that using the rollover in a legged robot may result in moving faster. In doing so, we identify the main factors and their parameters affecting the time required robot to change its initial state to the desired final one. Then we derive equations that allow calculating this time and determine the best sequence of gaits and parameters thereof that result in the minimal time. Reducing the travelling time is especially critical in robotic soccer.

Since 1998, the RoboCup legged league was exclusively using Sony 4-legged robots; however, these robots have been recently phased out by the manufacturer. Because new types of legged robot may likely be developed for the RoboCup competitions in the future, we believe that it makes sense to investigate the locomotion capabilities of a generic legged robot with rolling over as one of possible gaits. We anticipate that the results of this study will be taken into account by the developers of the new generation RoboCup legged robots.

Section 2 describes the problem addressed in this study and explains the main assumptions. In Section 3 we analyze the minimal-time optimization problem for the robot movement without rollovers. In Section 4 we introduce the rollover as a gait and provide the algorithm for optimizing robot movement with this optional gait. Section 5 describes the experimental results and Section 6 concludes this study.

2 The Optimized Robot Motion Problem: Main Assumptions

For the purpose of this study, we limit our consideration to 2D space. In doing so, we represent the state of the robot (pose) just by three parameters (x, y, δ); x, y being the coordinates of its center on the plane and δ the facing direction. We also assume that the robot has a set of gaits $\{G_0, G_1, \ldots, G_n\}$, each having a set of parameters. To reduce the dimension of the planning problem, we are using very limited set of parameters for each gait.

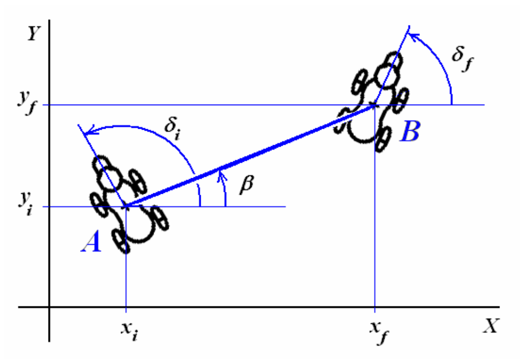

Fig. 1. Robot must change its initial state $\boldsymbol{A}(x_i, y_i, \delta_i)$ to $\boldsymbol{B}(x_f, y_f, \delta_f)$ in the minimal time

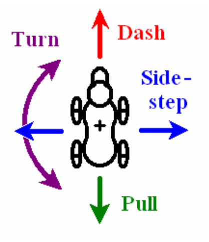

Fig. 2. The 'regular' gaits of a legged robot on the macro level

Consider the following optimization problem (Fig.1). Robot located in the initial state $\boldsymbol{A}(x_i, y_i, \delta_i)$ wants to get in the final state $\boldsymbol{B}(x_f, y_f, \delta_f)$ in the minimal time. Thus the robot must select the sequence of gaits and values of their parameters that minimize time to move.

Note that we consider planning ahead the desired robot orientation δ in the final point; in the robotic soccer the success of ball interception highly depends on this parameter.

To make this problem tractable, we make several simplifying assumptions.

1. There are no obstacles on the robot path.

2. We do not consider short-range moves, say, shorter than 1-2 robot length; rather, we are interested in optimizing longer-range movements when rollover may result in some gains. This allows neglecting the robot movement dynamics by assuming that starting/stopping are instant. (This is in particular reasonable for rather slow robots like Sony Aibos.)

3. Walking is split in four atomic macro level gaits: dashing, pulling, sidestepping, and turning while staying in the same place (Fig.2). These gaits can be applied one at a time; at any given instance, robot can either dash, or pull, or side-step, or turn, or roll.

4. Switching from one gait to another can be made instantly at any time (except the rollover in which each revolution must be fully completed before changing to different gait).

5. As time is the only criterion, each of these four gaits is fine tuned to achieve the maximal speed.

Assumptions 3-5 need more explanation. Generally, even with this very limited set of gaits, there is a continuum of walking options (from dashing/pulling/sidestepping straight to dashing-turning on a curve to just turning while staying on the same place). However, in legged mechanical systems, be they animals or robots, the movement along a non-straight trajectory can be achieved by interweaving of the four basic macro gaits shown in Fig.2. Because we assume that gaits could be applied in small discrete time intervals, the order of the three translation gates (dashing, pulling, and sidestepping) does not matter; Fig.3 illustrates this point. With dynamics neglected, the time to translate the robot thus does not depend on the trajectory if turns are not executed. This is similar to the well known *city block distance* [5]; the straight path in Fig.3 is as fasts as the other two. We believe that this assumption is true to most legged mechanical systems. If that was not the case, legged animals could be able to achieve higher speed while running in the diagonal directions; yet this does not happen. So in our model, this assumption implicitly sets the constraints inherent to legged robots.

We believe that further simplification of this model by merging the three translation gaits in just one is unreasonable because most legged robots are asymmetrical. Thus we assume that dashing, pulling, and sidestepping can be executed with different maximal speeds. This is one more macro level feature of legged robots that is reflected in the proposed model.

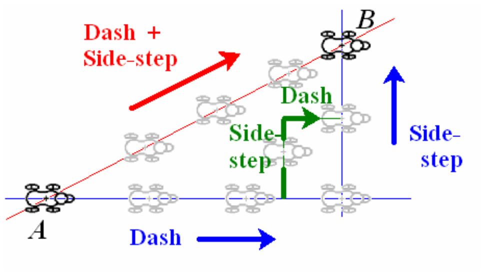

Fig. 3. Three alternative paths for translating the robot. All are taking same time to travel.

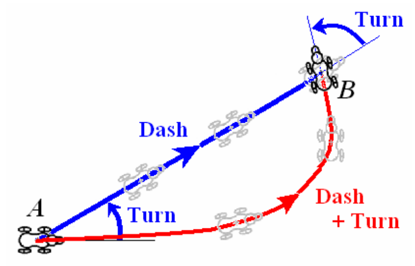

Fig. 4. Alternative paths: (top) turn-dash-turn and (bottom) dashes interweaved with turns

With turns included, applying same gaits with different timing may result in the trajectories having different end points. In Fig.4, the two trajectories have same end points and robot direction in it. However, for the bottom trajectory this is achieved by special selection of the time intervals when the robot is making turns. Finding the optimal sequence of gaits thus becomes a non-trivial problem, especially if dynamics is a factor. To simplify the analysis, we are making one more assumption: turns, if any, are only made in the start and end points like in the top trajectory in Fig. 4. With neglected dynamics, time necessary to move the robot in the required end pose by first making a turn followed by a translation followed by a second turn thus cannot be improved by applying turns and translations differently.

Rollovers provide extra choice to the robot how to move. With rollovers, there is an additional constraint: the robot must be always making integer number of revolutions.

The above assumptions imply that:

1. Robot has five alternative actions associated with macro level gaits:
   - G_t= “turn in place by angle α”,
   - G_d= “dash distance d”,
   - G_p= “pull distance d”,

- G_s= "sidestep distance *d*", and
- G_r= "make *N* rollovers".

where α, d, and N are robot decision parameters.

2. Robot can execute any one of these actions at any time except the rollover which, once started, must be completed.
3. As the maximal force is always assumed, force is not a gait parameter.
4. Because legged robots are typically asymmetrical, each translation gait has its own speed.
5. The set of the robot actions and their timing is limited to an optional turn in the start point ***A***, translation to the end point ***B*** and a second optional turn in this point.
6. Translation may be any combination of gates {G_d, G_p, G_s, G_r}; using G_r being the factor whose impact on the time required for reaching the end point we investigate.

With these assumptions in mind, in what follows, we derive equations for computing the robot trajectories on a plane and the traveling times with and without rollovers. Then we solve the optimization problem and compare the results for these two options.

3 Robot Motion without Rollover

3.1 Robot 2D Kinematics

First we assume the robot that is incapable of rolling over. Even with the limited set of possible gaits the robot has infinite number of options. Indeed, it can make any combination of turns, dashes or pulls and/or sidesteps to make its way to the final pose. Finding a rigorous method for minimizing the time could be subject of a stand-alone study; here we are using heuristic approach.

As it was explained in Section 2, if robot is limited to making turns in points ***A*** and ***B*** only, this would not affect the time given the assumptions we have made. This substantially simplifies the problem by leaving finite set of just seven options which all can be explicitly evaluated before making a choice (Fig.5). The first three are either *turn-dash-turn*, or *turn-pull-turn*, or *turn-sidestep-turn*. It is also possible for the robot to move with only one turn combined with two translations. There are total four choices of such trajectories; two shown on the right-hand pane in Fig.5. In each, sidestepping is always applied as the second translation.

Because we are interested in minimizing time, of the six parameters of the robot motion optimization problem, (x_i, y_i, δ_i, x_f, y_f, δ_f), only three are independent. In particular, for the *turn-dash-turn* combination (Fig.6), these three independent parameters could be:

$$displacement = \sqrt{(x_i - x_f)^2 + (y_i - y_f)^2}\,, \tag{1}$$

$$angle1 = \beta - \delta_i\,, \tag{2}$$

$$angle2 = \delta_f - \beta\,, \tag{3}$$

where β is the orientation angle of the displacement vector, which is determined by the two points, (x_i, y_i) and (x_f, y_f) (Fig. 1).

For the *turn-pull-turn* movement, the two angles are:

$$angle1 = (\beta + \pi) - \delta_i, \tag{4}$$

$$angle2 = \delta_f - (\beta + \pi), \tag{5}$$

For *turn-sidestep-turn*, these angles can be calculated as:

$$angle1 = (\beta + \pi/2) - \delta_i, \tag{6}$$

$$angle2 = \delta_f - (\beta + \pi/2), \tag{7}$$

It is assumed that the turning angles returned by (2)-(7) are normalized to the range [-π,-π). Note that these angles differ for different robot translation modes.

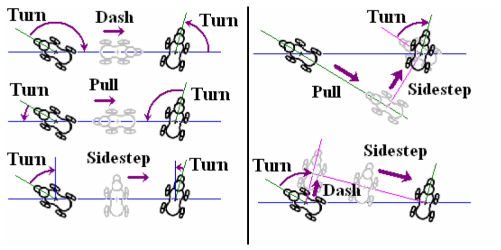

Fig. 5. Five of the total of seven robot movement without rollovers

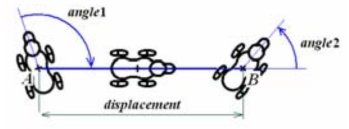

Fig. 6. Movement parameters

For the two motions shown on the right-hand pane in Fig5, there is just one turning angle:

$$angle1 = \delta_f - \delta_i \tag{8}$$

and two displacements for the first and second translation:

$$displacement1 = \begin{cases} displacement \cdot |\sin(\delta_f - \beta)|, & \text{if turn at } \boldsymbol{A} \\ displacement \cdot |\cos(\delta_i - \beta)|, & \text{if turn at } \boldsymbol{B} \end{cases}, \tag{9}$$

$$displacement2 = \begin{cases} displacement \cdot |\cos(\delta_f - \beta)|, & \text{if turn at } \boldsymbol{A} \\ displacement \cdot |\sin(\delta_i - \beta)|, & \text{if turn at } \boldsymbol{B} \end{cases}, \tag{10}$$

There are also two similar motions with the robot turning in the direction opposite to the orientation required in the final pose.

3.2 Minimal-Time Optimal Robot Motion Planning without the Rollover

Thus we have just a set of seven options to evaluate and to choose one having minimal time for the legged robot to move from ***A*** to ***B***. Because of the inborn asymmetry, this time with different gaits may substantially differ. Normally, dash is designed as the fastest of the three translation gaits. Also, depending on translation gait chosen, the total amount of time spent on turns also differs. In the example in Fig.5, the sum of two turns is minimal if the robot chooses to sidestep. If sidestepping is not too slow indeed, in this example it could be best choice in this particular case. So we can expect that, in general, of the seven options that robot has to choose from, some may require noticeably shorter time than others.

With robot dynamics neglected, we assume that the angular speed of the robot is ω and its translation speeds for dashing, pulling, and sidestepping are v_d, v_p, v_s, respectively.

So the time needed to change the robot state from (x_i, y_i, δ_i) to (x_f, y_f, δ_f) using the translation gait G_k is,

$$time_k = (|angle1_k| + |angle2_k|) / \omega + displacement / v_k, \tag{11}$$

where **k** is one of the identifiers **d**, **p**, **s**. of the translation gaits.

Calculating time with just one turn and two translation gaits can be done in the similar way:

$$time_{jk} = |angle1| / \omega + displacement1_{jk} / v_k + displacement2_{jk} / v_s, \tag{12}$$

where **j** indicates whether the turn was made in ***A*** or ***B*** and **k** is the identifier of the translation gait applied first (either **d** or **p**); the rest parameters are given by (8)-(10).

Thus to find the set of actions for changing its state from $\boldsymbol{A}(x_i, y_i, \delta_i)$ to $\boldsymbol{B}(x_f, y_f, \delta_f)$ in minimal time without rollover, the robot should determine the time for each of the seven options. Then the movement mode delivering the minimal time is selected.

4 Robot Motion with Rollover

4.1 Robot Rollover Kinematics

The rollover kinematics is determined by three parameters, the first being the number of full revolutions *N*. In what follows, we identify the rest two.

For the purpose of modeling rolls, a legged robot is regarded a truncated cone. For example, because the Sony Aibo robot has a head, its effective diameter at shoulders is greater that that at the hips. For this reason, in the general case, the rolling path is an arc whose radius is *R*.

Let $chord_1$ be the linear displacement of the robot center, and θ is the angle by which the robot direction changes per full revolution (Fig.7).

Thus we have

$$chord_1 = 2 \cdot R \sin\frac{\theta}{2} \cdot \tag{13}$$

In what follows, we do not call this parameter 'displacement' because the latter designates the distance between points ***A*** and ***B***. (In the particular case when the robot could be approximated by a cylinder, R is infinite and θ is zero; $chord_1$ is just the circumference of the cylinder.)

For any number N of revolutions, the chord connecting the initial and final points is:

$$chord_N = 2 \cdot R \sin \frac{N\theta}{2} \cdot \tag{14}$$

It is important to make sure that the following constraint is satisfied:

$$N \le \frac{\pi}{\theta} \cdot \tag{15}$$

Indeed, if $N = \pi/\theta$, robot rolls exactly half circumference whose radius is R. Further increments of N would result in that the robot would be approaching back to the initial point $\boldsymbol{A}$.

For the purpose of analysis, we have found it more convenient not using R as a parameter of the robot kinematics. Instead, we introduce θ as the second and $chord_1$ as the third independent parameters; both can be easily measured experimentally. By solving (13) and (14), we eliminate R. So the final formula for the straight distance covered by N rollovers is:

$$chord_N = chord_1 \cdot \frac{\sin \frac{N\theta}{2}}{\sin \frac{\theta}{2}} \tag{16}$$

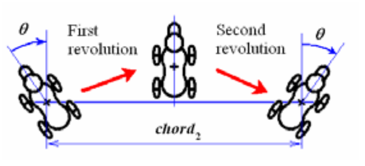

Fig. 7. Robot direction changes by θ after each revolution

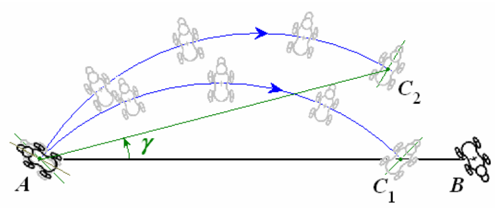

Fig. 8. Alternative rollover paths leading close to the final point $\boldsymbol{B}$

4.2 Minimal-Time Optimal Robot Motion Planning: The General Case

Rollover adds two degrees of freedom in the planning task. They are the number of revolutions N and the direction γ from the initial point to the end point of the rollover (Fig. 8).

These action parameters are not all uniquely determined by the initial and final states of the robot. We simplify the problem by assuming that the rollover trajectory is always selected by setting γ =0; i.e. the end point $\boldsymbol{C}$ of the last rollover must be on line $\boldsymbol{AB}$. However, this is just a near-optimal solution; by carefully choosing γ, in some cases it is indeed possible to reach the final pose faster. However, this gain in speed, if any, would be negligibly small.

This simplification means that of the continuum of potential trajectories we will be selecting one of just two possible pairs with the end points $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ shown in Fig.9. They require N or (N+1) revolutions, respectively. After the robot reaches any of these end points, it faces the task to reach the final state $\boldsymbol{B}$ from $\boldsymbol{C}_1$ or $\boldsymbol{C}_2$ in minimal time. To accomplish this, robot applies the optimization method without rollovers described in section 3.2.

In Fig.9 all four paths have different time to reach the final point. The paths differ in the initial and final turning angles; also different is the number of revolutions and the residual paths after rolling.

In the nutshell, to calculate time to move from $\boldsymbol{A}$ to $\boldsymbol{B}$ in the general case, robot first determines the required time for the following options:

- without rollovers (seven options),
- two options with N rollovers and seven options to move from $\boldsymbol{C}_1$ to $\boldsymbol{B}$, and
- two options with N+1 rollovers and seven options to move from $\boldsymbol{C}_2$ to $\boldsymbol{B}$.

Then of the total of 7·3+2+2=25, the option having the minimal time is selected.

The time to reach the end point $\boldsymbol{C}_1$ ($\boldsymbol{C}_2$) with N rollovers along k-th trajectory is:

$$timeR_k = |angle1_k| / \omega + N \cdot revtime, \tag{17}$$

where $angle1_k$ is the first turn angle and *revtime* is time to execute one full revolution.

The number of revolutions N is determined from the inequality

$$chord_N \leq displacement \leq chord_{N+1}, \tag{18}$$

where $chord_N$ is given by (14).

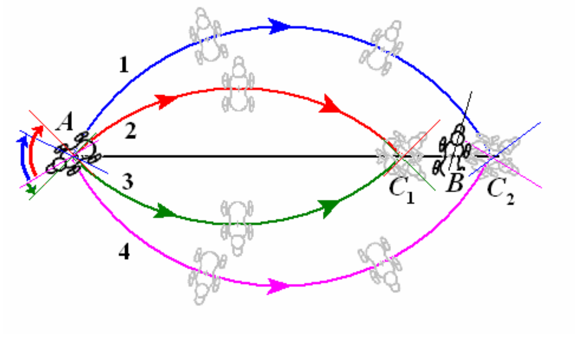

Fig. 9. The two pairs of alternative paths with rollovers. Note different turning angles before and after rolling.

For each trajectory in Fig.9, the turning angle $angle1_k$ can be found as follows:

$$angle1_1 = (\delta_i - \beta) - (\pi + (N+1) \cdot \theta) / 2, \tag{19}$$

$$angle1_2 = (\delta_i - \beta) - (\pi + N \cdot \theta) / 2, \tag{20}$$

$$angle1_3 = (\delta_i - \beta) - (3\pi - N \cdot \theta)/2, \tag{21}$$

$$angle1_4 = (\delta_i - \beta) - (3\pi - (N+1) \cdot \theta)/2. \tag{22}$$

Upon arrival in point $\boldsymbol{C}_1$ ($\boldsymbol{C}_2$), the robot coordinates and body direction are

$$x_C = x_i + chord_N \cdot \cos\beta, \tag{23}$$

$$y_C = y_i + chord_N \cdot \sin\beta, \tag{24}$$

$$\delta_C = \delta_i + angle1_k \pm N \cdot \theta, \tag{25}$$

where in (25) the minus sign applies to trajectories 1, 2 and the plus sign to trajectories 3, 4.

Then using the new robot initial state $\boldsymbol{C}(x_C, y_C, \delta_C)$ we can find the time needed to cover the remaining path to $\boldsymbol{B}(x_f, y_f, \delta_f)$, by applying the algorithm described in Section 3.2.

This provides the full description of the algorithm for calculating time to change the robot pose $\boldsymbol{A}(x_i, y_i, \delta_i)$ to pose $\boldsymbol{B}(x_f, y_f, \delta_f)$ for any given set of gaits that satisfy our assumptions.

5 Experiments and Performance

The authors came to the idea to use rollovers in late 2005 soon after the first experience with Sony ERS-7 robots equipped with the well-known *Tekkotsu* software package [6]. The walking gaits that come with this software appeared to be too slow. On the other hand, the rounded body of the robot and its ability to place its head in rather low position rendered themselves for rolling possibly faster than walking. So the authors implemented the rollover in ERS-7 and conducted a set of experiments to measure its parameters (Fig.10). After completing this study, the authors discovered that the rollover indeed had been earlier implemented as part of a student project an undergraduate course in robotics at Carnegie-Mellon. However, in that case was used different robot ERS-210. Presumably because of its boxy shape rolling appeared to be clumsy and noticeably slower than our ERS-7.

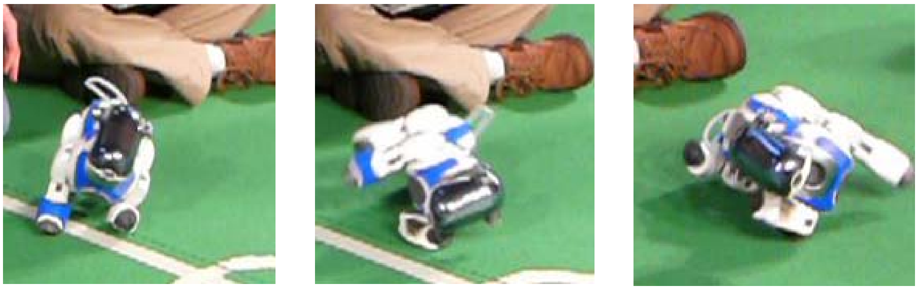

Fig. 10. A Sony Aibo ERS-7 robot making a rollover

The rollover 'gait' in our experiments was designed using only common sense and thus cannot be regarded fully optimized. Yet we have gained same translation speed (31.0 cm/s in our rollover vs. 30.9 cm/s in *Tekkotsu* dashing). Table 1 provides the experimental values of the main parameters of the ERS-7 robot gaits as they were found in *Tekkotsu*; the rollover parameters are also included.

Table 1. Sony Aibo ERS-7 robot parameter estimates

Gait	Parameter	Mean value	Standard deviation
Straight line dashing	speed, cm/s	30.9	4.5
Side stepping	speed, cm/s	20.4	1.65
Pulling	speed, cm/s	14.0	1.42
Turning	angular speed, degrees/s	205	19.1
Rollover:	Chord, cm	36.6	1.3
	Revolution time, s	1.18	0.06
	Angle increment, degrees	22.5	1.8
	Calculated translation speed, cm/s	31.0	N/A

Using the averages of these parameters, we have implemented a software model to estimate the robot motion performance with and without rollovers. Based on expressions (1)-(25), this model calculates the anticipated time for the robot to change its pose ***A*** to pose ***B***. Table 2 presents the summary of the calculation results.

The calculations have been made for 100 values of *alpha*1 in the range $(-\pi, \pi)$, combined with 100 values of *displacement* in the range (30, 180) cm, and 100 values *alpha*2 in the range $(0, \pi)$, thus making the total of 10^6 points uniformly distributed in the 3D parameter space. Distances greater than 180 cm were not considered because from (15) we get the maximal number of revolutions N=8; thus from (14) $chord_8$=187.6 cm, which is the maximal reach by rolling over for this robot.

The central column (highlighted in boldface) shows the results for the robot parameters taken from Table 1. It demonstrates that in 42.6 per cent of cases the option to rollover would reduce time by 0.285 seconds on the average. In a soccer game this time gain means that the robot intercepting the ball would be likely outplaying a non-rolling opponent if all the rest conditions are same. Yet this is a somewhat pessimistic estimate; if for the same values of the model the displacement range is limited to (30, 120) cm, the rollover gives noticeably better average time advantage 0.316 seconds in 56.7 per cent of cases. This is because the contribution of the curvature to the rollover path on shorter distances is less significant.

The two left-hand columns give the idea how the advantage gained with optional rollovers decays with the reduced linear speed of rollovers. At 80 per cent of the dashing speed the gain is achieved just 9.0 per cent time; at 60 per cent the contribution of rollover completely diminishes.

Table 2. Rollover vs. non-rollover motion comparison

Performance indicator	**(Rollover linear speed)/(dashing speed)**				
	0.60	0.80	**1.00**	1.20	1.40
Average time without rollovers, s	4.24	4.24	**4.24**	4.24	4.24
Average time with rollovers, s	4.24	4.22	**4.12**	3.84	3.50
Per cent of times when rollover is faster	0.001	9.02	**42.6**	74.2	93.2
Average gain when rollover is faster, s	0.001	0.18	**0.285**	0.540	0.791
Relative gain when rollover is faster, per cent	0.01	4.27	**6.71**	12.7	18.6

We believe that in Aibos rolling over is more energy efficient than walking and can be further improved. This is supported by the experimental observation of that the robot dashing speed tended to decrease with distance; hence the relatively high standard deviation of the dashing speed in Table 1. We attribute this to chemical processes in the battery under high load. After some rest, robot could be able to dash with the original speed again for a short time. With rollover, however, we did not observe this negative effect of the battery drain.

Because no attempts have been made to optimize the rollover motion sequence, it would be nice to know what might happen if one had managed to improve the coordinated action of the 15 joints of the robot's neck and legs that are making it rolling over. The two right-hand columns in Table 2 show these projections. If we improve the rollover linear speed by 20 per cent, robot would be faster by 0.54 seconds in 74.2 per cent cases. With 40 per cent speed increase, rolling over would be advantageous 93.2 per cent of the time.

6 Conclusion

The rollover 'gait' adds more flexibility in the legged robot movement. Even when the rollover has about same linear speed as dashing, if introduced in a robot, this extra gait allows in some situations saving time on turns that are thus becoming unnecessary due to the possibility to move in the lateral direction. Because the proposed model is taking into account only high-level features of robot motion, we believe that it is applicable to almost any legged robots. By plugging into our model different parameter values, a robot designer may determine if adding the rollover capability would or would not give any advantage in speed.

Evaluating the potential gains for Sony Aibo robot from rolling over was just an illustration of the proposed method. Our study demonstrates that this gain is tangible; this gives the reason for attempting to implement the rollover it in the legged soccer robots.

Yet we are looking forward to the new developments in the legged RoboCup league; thus we hope that this study would be helpful for making high-level design decisions in the development of new robots.

Acknowledgements

The authors thank Prof. Manuela Veloso of CMU and her Ph.D. student Sonya Chernova, who provided the materials on the student projects in the robotics course taught at their school.

References

1. Alexandre, P.: An Autonomous Micro Walking Machine with Rollover Recovery Capability. In: Workshop II: New Approaches on Dynamic Walking and Climbing Machines of the 8th International Conference on Advanced Robotics, Monterey, CA, USA, pp. 48–55 (1997)
2. Son, Y., Kamano, T., Yasuno, T., Suzuki, T., Harada, F.: Generation of Adaptive Gait Patterns for Quadruped Robot with CPG Network Including Motor Dynamic Model. Electrical Engineering in Japan 155(1), 2148–2154 (2006); Translated from Denki Gakkai Ronbunshi 124-C(10) (2004)
3. Stoeter, S.A., Papanikolopoulos, N.: Kinematic Motion Model for Jumping Scout Robots. IEEE Transactions on Robotics 22(2), 398–403 (2006)
4. Kuniyoshia, Y., Ohmuraa, Y., Teradaa, K., Nagakuboc, A., Eitokua, S., Yamamotob, T.: Embodied basis of invariant features in execution and perception of whole-body dynamic actions - knacks and focuses of Roll-and-Rise motion. Robotics and Autonomous Systems 48, 189–201 (2004)
5. Krause, E.F.: Taxicab Geometry. Dover Publications, Mineola (1987)
6. Touretzky, D.S., Ethan, J., Tira-Thompson, E.J.: Tekkotsu: a Sony AIBO application development framework. The Neuromorphic Engineer 1(2), 12 (2004)