A Model-Based Approach to Calculating and Calibrating the Odometry for Quadruped Robots[⋆]

Haitao He and Xiaoping Chen

Department of Computer Science, The University of Science and Technology of China, HeFei, China

Abstract. This paper presents a model-based odometry calculation and calibration method (MBO) for quadruped robots. Instead of establishing the direct relation between target and actual speeds as previous methods did, MBO sets up a "parametric physical model" incorporating various properties of the robot and environment such as friction and inertia, through optimization with locomotor data. Based on this optimized model, one can compute the loci of robot legs' movement by forward kinematics and finally obtain odometric readings by analyzing the loci. Experiments on Sony AIBO ERS-7 robots demonstrate that the odometry error of MBO is generally 50% less than the existing methods. In addition, the calibration complexity is low.

1 Introduction

Odometry is very important for an autonomous robot[1,2,3,9], especially for the purpose of determining the robot's locomotory parameters (e.g., speed, position, orientation.) between fixes which are rare or cost demanding. It is relatively simple for a wheeled robot to calculate its odometry. The vehicle's offset from a known starting position can be computed with the data of encoders which monitor the wheels' revolutions and/or steering angles. Odometry calibration for wheeled robots often involves determining the values of kinematic parameters [2,9] or calibrating error model [3]. However, these methods cannot be implemented directly on legged robots, which have completely different mode of locomotion.

Two common motivations behind the current investigation into the odometry of legged robots are: (1) the odometry should be accurate and powerful enough with respect to the needs of real-world applications; (2) the odometry calibration should be as simple as possible. There is some work on meeting these two requirements jointly. Thomas Röfer [11] reports on a method for calculating odometry based on proprioception. German Team optimizes the parameters [5] of their walking engine [4] to make the actual walking speed as close as possible to that specified by the corresponding walk request. The rUNSWift team establishes the relation between raw walk command values and the corresponding actual speed values through polynomial curve fitting [6]. Lin and his colleagues propose calculating odometry for a hexapod robot from its body pose based on the kinematic configuration of its legs [7]. Stronger and Stone put forth a method for simultaneous calibration of action and sensor models autonomously [10].

⋆ This work is supported by the NSFC 60275024 and the 973 program 2003CB317000.

All the previous work can be divided into two categories: one is to directly establish a mapping between the target and actual speeds – we call this "direct calibration". The other is mainly based on proprioception, we call this "indirect calibration". With the "direct" method try to establish a mapping from one space (target speeds) to another space (actual speeds), one has to calibrate lots of points of the space, and another disadvantage is the mapping lack the information of the process of state changing from one point to a another point. The "indirect" method uses the information of sensors, and generates the continuous proprioceptive state, so as to avoid the negative aspects of "direct" methods.

Based on previous work, we propose in this paper a new odometry calculation and calibration method for quadruped robots, named MBO (model-based odometry), that is the first "indirect calibration" based on a "parametric physical model" for quadruped robots. Instead of establishing the direct relation, MBO sets up a "parametric physical model" incorporating both geometrical and physical properties of the robot such as friction and inertia through optimization of a parameterized version of the "triangle model" proposed in [8] with the locomotor data of the robot. Based on this model, MBO can figure out the loci of leg movement with forward kinematics and finally obtain the odometric readings by analyzing loci. Since it is straightforward to calculate the acceleration/deceleration of the robot's movements at runtime, MBO provides a "finer-granularity" odometry for quadruped robots. In addition, the calibration complexity is rather low—only twelve walking samples are needed for optimization of the parametric physical model, although MBO runs semi autonomously at the current stage. We implemented MBO and carried out experiments on AIBO ERS-7. The results show that the accuracy of the odometry calculated by MBO is markedly higher than that by previous methods.

The remainder of this paper is organized as follows: Section 2 describes briefly the principle of MBO. Section 3 explains calibration of the parametric physical model on the AIBO platform. Section 4 discusses how to calculate odometry from loci. Section 5 presents the experimental results and the paper is concluded in Section 6.

2 The Principle

Consider a walking engine that responds to any walking request by arranging the legs' movements according to some preset gait loci. If the robot is driven by motors, the robot's architecture usually allows each joint to receive a request every t_u time and returns feedback on its corresponding angles at the same frequency. According to a given leg model, the paw positions can be calculated by forward kinematics using the joint angles fed back by joint sensors. Moreover, a calculated locus can be simply acquired by connecting the consecutive paw positions generated from the angles.

If a robot is held in the air while walking, its paw locus looks like the curve shown in Fig.1. The solid line from s to e in Fig.1 presents the trajectory of support phase of a leg. If the leg keeps in contact with the ground and has no paw slippage during the support phase, the movement of the robot is fully determined by the locus of the support phase. Therefore, the problem of odometry calculating is reduced into that of calculating the actual loci of legs.

Specifically, the displacement of one walk cycle is determined by the corresponding trajectory on the $x - y$ plane. Suppose a leg makes contact with the ground at point s and is raised in the air at point e,their projected projective points on the $x - y$ plane is marked as P_s and P_e respectively (Fig.1). The displacement S of this walk cycle can be computed by setting $S = P_s - P_e$. It follows under the assumption of no obstruction that the displacement of one walk cycle only concerns the point where the leg is placed onto the ground and the point where the leg is raised. Therefore,the error caused by discrete feedback of joint sensors is limited and can occur only within a few cycles of feedback when the leg is placed on or raised from the ground.

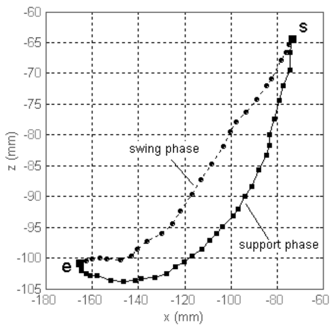

Fig. 1. One walk cycle of sampled leg locus. The sample interval t_u=8ms. The leg is in contact with the ground at point s, and raised in the air at point e.

The assumption of no slippage is problematic for real-world applications. Generally, odometry for legged robots is error-prone due to the noises and/or uncertainty inherent in the robots, their motion and even their environments. In order to simplify the modeling, one usually neglects physical properties such as friction, weight of the robot and its components, inertia, motor strengths, causing impaired performance in real-world applications [4]. As a remedy to this problem, we try to model these factors including both geometrical and physical properties as nuch as possible. We take a parameterized version of “triangle model” [8] as the starting point for our “parametric physical model” (see Fig. 2) and optimize it with the locomotor data of the robot’s movements in the application environment by using a genetic algorithm. The expressive power and the adaptability of this model primarily stems from the variance of the parameters. For example, the length of some parts of the leg would be reduced by the genetic algorithm when the robot walks on a slipperier carpet. Due to this adaptability of the parametric physical model, MBO can provide more accurate odometric readings, as our experiments demonstrate.

3 Model Setup

There are two steps in MBO to set up a parametric physical model: (1) setting an temporary profile of the parameterized triangle model; (2) obtaining the final model through optimization. This section describes the detailed method on the sony AIBO robotics platform.

3.1 Parameterized Triangle Model

The AIBO robot has four legs. Each leg has three joints known as the rotator, abductor, and knee. The most widely used model is the triangle model described in [8], in which the lengths and widths of the legs are both considered. The model is simple and convenient to use. MBO uses the triangle model as the basis,but parameterizes it (Fig.2).

The following parameters describe in a related coordinate system whose origin is the position of joint rotator: 1). position correction of joint rotators six parameters.

2). length of fore and hind upper legs, L1 in Fig.2 two parameters. 3). width of fore and hind upper legs, L3 in Fig.2 two parameters. 4). length of fore and hind upper legs, L1 in Fig.2 two parameters. 5). the rotation angle correction of joint knee,θ in Fig.2. 6). the zero correction of joint rotator, abductor and knee, three parameters.

The paw position $P(x, y, z)$ can be determined using following transformations, written in matrix:

$$\begin{pmatrix} A1' \\ A2' \\ A3' \end{pmatrix} = \begin{pmatrix} A1 \\ A2 \\ A3 \end{pmatrix} + \begin{pmatrix} A1Correction \\ A2Correction \\ A3Correction \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(A1') & 0 & -\sin(A1') & 0 \\ 0 & 1 & 0 & 0 \\ \sin(A1') & 0 & \cos(A1') & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos(A2') & 0 & -\sin(A2') & 0 \\ 0 & 1 & 0 & 0 \\ \sin(A2') & 0 & \cos(A2') & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L1 \\ 0 & 0 & 0 & 1 \end{pmatrix} *$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & L3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos(A3' + \Delta R) & 0 & -\sin(A3' + \Delta R) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(A3' + \Delta R) & 0 & \cos(A3' + \Delta R) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L2 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ 1 \end{pmatrix} \tag{2}$$

where $(\Delta x, \Delta y, \Delta z, \Delta 1)^T$ denotes the position correction of leg joint rotator,ΔR the rotation angle correction of the joint knee.

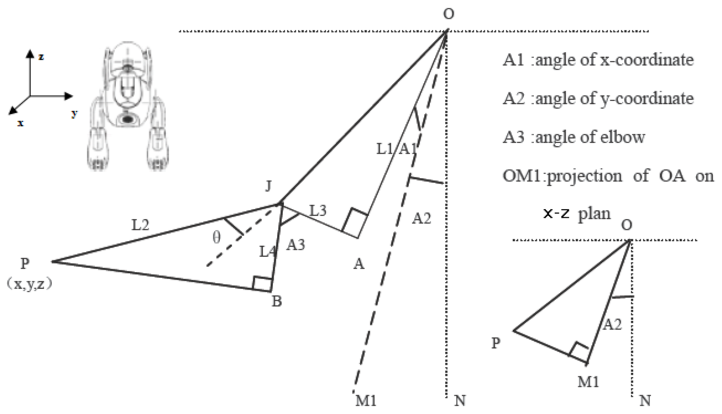
O
z
y
x
A1 :angle of x-coordinate
A2 :angle of y-coordinate
A3 :angle of elbow
OM1:projection of OA on x-z plan
L1/A1
J
L3
A2
L2
θ
L4
A3
A
P
(x,y,z)
B
M1
N
O
A2
P
M1
N

Fig. 2. The triangle leg model of AIBOs. Point O is the joint rotator position. Point P is the paw position.

3.2 Model Optimization

A position can be determined by a given sequence of all leg joints data and a given leg model. However, the calculated position is not close sufficiently to the actual position if the calculation is based on a pure leg model, because there are too many factors affecting the accuracy, (as described in Sect.2). Therefore, MBO optimizes the initial model described above with the locomotor data of the robot's movements in the application environment. For this purpose, the model is re-described by a set of parameters, denoted $M_i(p_{i1}, p_{i2}, \ldots, p_{in})$. Thus model optimization is reduced to a search of the best set of parameters in an n-dimensional space. A basic genetic algorithm is implemented here for the optimization. It first computes the displacement S_i with a given model M_i and a given sequence of consecutive joints data D according to formula (3) and determines the fitness of a model M_i according to formula (4):

$$S_i = F(M_i, D) \tag{3}$$

$$f_i = \{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{pmatrix} * (S_i^o - S_i)\}^T * \{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{pmatrix} * (S_i^o - S_i)\} + \omega \tag{4}$$

where ω denotes the variance between the displacements of fore legs and hind legs. The method of analyzing odometry readings from loci is used here and will be described in Sect. 4.

4 Odometry Calculation

The purpose of MBO is to obtain the odometric readings for the robot through calculating its paw loci by forward kinematics based on the optimized physical model. The robot's movement is completely determined by the movement of its support legs. From the loci of all legs, one can deduce the support legs and swing legs and calculate the odometric readings for the robot.

The detailed method of odometry calculation depends on the robot and its walking type. In this paper, we assume that the quadruped robot uses the trot gait which lifts the two diagonally opposite legs alternately [8].

Support Legs Calculation. A pair of diagonal legs are in contact with the ground as support legs and the other two legs swing in the air at any given time. Even though the robot may not have static balance and a third leg may fall on the ground, the effect of the third leg is ignored here for sake of simplicity. The two diagonal legs whose positions are lower are taken as the support ones.One plane can be determined by three arbitrarily chosen legs, the remaining leg is below or above the plane. If the remaining is under the plane, the supports legs are the diagonal legs which are composed by it and otherwise are the other pair.

Translation and Rotation Calculation. A robot walks by lifting the two pairs of diagonally opposite legs alternately. Each pair of diagonal legs swing in the air for a moment and then stay touching ground . The translation of the robot is calculated according to the movement of the legs that are touching the ground. Let R_{trans} and R_{rot} denote the translation and rotation of the robot, respectively; f_{trans} and h_{trans} denote the displacement of fore and hind support leg, respectively; $P_f(t)$ and $P_h(t)$ projective points on $x-y$ plane of the fore and hind support leg's paw position at time t.

The translation is computed by following rules:i. if support legs do not change from time $t-1$ to time t, then, $f_{trans} = P_f(t-1) - P_f(t)$, $h_{trans} = P_h(t-1) - P_h(t)$, $R_{trans} = (f_{trans} + h_{trans})/2$. ii. if support legs have changed form time $t-1$ to time t,then $R_{trans} = 0$.

The rotation is computed by following rules: i. if support legs do not change from time $t-1$ to time t, then set vector $l = P_f(t-1) - P_h(t-1)$,vector $c = P_f(t) - P_h(t)$. The rotation of robot is equal to the angle between vector l and vector c. ii. if support legs have changed form time $t-1$ to time t, then $R_{rot} = 0$.

5 Experiments

The experimental platform is the sony AIBO-ERS7.The parametric physical model is optimized using 12 samples in Table 1.

Table 1. Training samples for setting up the parametric physical model in the experiments

walk type	$C_i(x, y, \theta)$ of samples,$T_i = 5s$	number of sample
forward	$\{(200, 0, 0), (300, 0, 0)\}$	4
backward	$\{(-250, 0, 0)\}$	2
sidewalk	$\{(0, 200, 0)\}$	2
rotation	$\{(0, 0, 130), (0, 0, 180)\}$	4

These experiments are to compare the accuracy of MBO with that of German Team 2004.Both odometric readings returned by German Team 2004 and MBO are recorded and the actual displacement of the robot is measured manually. Equation (6) in Sect.3 is adopted here to evaluate the errors.We use a external camera to capture the position and orientation of the robot. The average error of position and orientation are $\pm$2cm and $\pm$5 degree.

Experiment 1. First, we test the errors caused by executing a single instruction each time, with each of these instructions sampled twice or thrice. The experiment results are shown in Table 2, which show that the odometrical readings returned by MBO are always closer to the measurements and is robust to the different actions.

Table 2. Results of experiment 1

instruction (C,T)	measured displacement	displacement of GT04	displacement of MBO	error rate of MBO	rate proportion (MBO/GT04)
{(300,0,0),6000}	(2050,90,5)	(1850,0,0)	(2046,-58,-2.7)	0.08	0.74
	(2060,200,10)	(1850,0,0)	(2040,67,0.4)	0.08	0.53
{(300,0,60),6000}	(110,80,415)	(21,0,364)	(129,72,414.2)	0.005	0.04
	(-80,100,420)	(21,0,364)	(91,92,412.6)	0.04	0.32
{(-250,0,0),6000}	(-1050,-40,0)	(-1500,0,0)	(-1198,-45,-0.7)	0.14	0.32
	(-1080,0,-10)	(-1500,0,0)	(-1180,-52,-5.9)	0.11	0.27
{(0,200,0),4000}	(0,870,10)	(0,811,0)	(-45,850,3.5)	0.09	0.70
	(-50,860,7)	(0,811,0)	(-62,833,12.5)	0.07	0.63
{(0,0,130),4000}	(-40,80,470)	(0,0,524)	(2,12,475.2)	0.02	0.18
	(10,10,475)	(0,0,524)	(-37,-1,441)	0.07	0.76
	(-10,15,445)	(0,0,511)	(-30,3,443)	0.007	0.05

Experiment 2. Let the robot start at point (0,0,0), perform one of the following sequences of instructions, and then stop: Ψ_1={{(150,0,0),500}, {(200,0,0),500}, {(250,0,0), 500}, {(300,0,0),500}}; Ψ_2={{(300,0,0),1000}, {(250,0,0),500}, {(200,0,0),500}, {(150,0,

0),500}}; Ψ_3={{(0,0,100),500}, {(0,0,130),500}, {(0,0,160),500}, {(0,0,190),500}}; Ψ_4= {{(0,0,200),1000}, {(0,0,170),500}, {(0,0,140),500}, {(0,0,110),500}}; Ψ_5={{(200,0,0), 2000}, {(200,0,60),2000}, {(200,0,0),2000}, {(200,0,-60),2000}}.

$\Psi_1$is an accelerating process in x-axis direction and Ψ_2 a decelerating process after a sudden start . Ψ_3 and Ψ_4 are similar to Ψ_1 and Ψ_2 , respectively, but in φ direction. The results (Table 3) show that MBO is more accurate than German Team 2004 except in the case Ψ_4, where both methods are equally good. It is worth noting that odometrical readings returned by MBO for rotation is much better. Ψ_5 is used to test the odometry accuracy when the robot moves along a curve and MBO is also better in this case. Moreover, the actual trajectory caused by Ψ_5 is approximated by fitting a smooth curve over several manually measured points that the robot passes, as shown in Fig. 3. The shape of the trajectory generated by MBO is much closer to the actual one than that generates by German Team 2004. To sum up, the accuracy of MBO improves by at least 50% in most cases.

Table 3. Results of experiment 2

Order	Measured	GT04	MBO	MBO /GT04
Ψ_1	(2050,90,5)	(450,0,0)	(612,-17,-1.3)	0.43
Ψ_2	(675,30,3)	(600,0,0)	(670,47,2.2)	0.48
Ψ_3	(-10,-5,-6)	(0,0,290)	(-10,-2,-13.1)	0.02
Ψ_4	(-15,20,60)	(0,0,50)	(-12,8,50.1)	0.99
Ψ_5	(130,760,0)	(529,919,0)	(141,959,8.0)	0.46

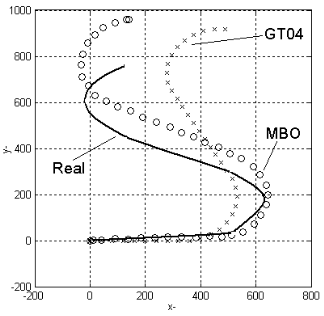

Fig. 3. The trajectories of Ψ_5

6 Conclusions

The original notion of odometry calculation for quadruped robots basically concerns the relation between the output (the odometric readings) and the input (the actual motion data) of a movement. As far as we know, all previous methods of the odometry calculation for quadruped robots are technically based on this notion in the sense that some direct relations between the target and actual speed are established and employed to tell the odometric readings. An alternative approach is proposed in this paper. Instead of establishing the direct relation, MBO sets up a "parametric physical model" incorporating various properties of the robot and even the environment such as friction and inertia. Based on this model, MBO deduces the loci of leg movements by forward kinematics and obtains the odometric readings by analyzing loci in execution time.

We described the major steps and tested the performance of this method on the AIBO platform with a generally applicable methodology. MBO fits the quadruped robot whose swing phase occupies not greater than 50% of its whole gait trajectory. The experiments showed the calibration complexity is as low as only twelve samples, covering 4 basic motions—straight forward, straight backward, pure sideways walking, and pure rotation. The error is 50% less than existing methods and the error rate is below 8% for most motion types. MBO does not demand additional sensors, and it only employs the feedback of the leg joint sensors while returning odometric readings online. Another feature of MBO is that the instantaneous speed of the robot offered by MBO is very sensitive with steadily lower error. This would provide a new opportunity for more precise motion control of legged robots[7].

For that purpose, we need to work further on the prediction based on the odometry. In addition, achieving the fully autonomous odometry calibration is a most important future work. This implies that we need some on-line and on-board optimization methods. Another interesting problem concerns the choice of types and number of training samples for building an optimal "parametric physical model", especially when "mixed motions" (e.g., moving forward and sideways at the same time [5]) are considered more thoroughly.

Acknowledgment

We would like to thank Kai Xu, Fei Liu and Benjamin Johnston for their valuable advice and helpful discussions. We also would like to thank the members of the WrightEagle Team for their efforts in developing the software used as a basis for the work reported in this paper.

References

1. Kelly, A.: Fast and easy systematic and stochastic odometry calibration. In: International Conference on Intelligent Robots and Systems (October 2004)
2. Borenstein, J., Feng, L.: Measurement and correction of systematic odometry errors in mobile robots. IEEE Transactions on Robotics and Automation (December 1996)
3. Chong, K.S., Kleeman, L.: Accurate Odomctry and Error Modelling for a Mobile Robot. In: International Conference on Robotics and Automation, April 1997, pp. 2783–2788 (1997)
4. Düffert, U., Hoffmann, J.: Reliable and Precise Gait Modeling for a Quadruped Robot. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 49–58. Springer, Heidelberg (2006)
5. Hengst, B., Ibbotson, D., Pham, S.B., Sammut, C.: Omnidirectional locomotion for quadruped robots. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 368–373. Springer, Heidelberg (2002)
6. Chen, W.: Odometry Calibration and Gait Optimisation. The University of New Wales School of Computer Science and Engineering, Technical Report (2005)
7. Lin, P.-C., Komsuoḡlu, H., Koditschek, D.E.: Legged Odometry from Body Pose in Hexapod Robot. Experimental Robotics IX, STAR 21, pp. 439-448 (2006)
8. Hengst, B., Ibbotson, D., Pham, S.B., Sammut, C.: The UNSW United 2000 Sony Legged Robot Software System, School of Computer Science and Engineering University of New South Wales, Technial Report (2000)
9. Antonelli, G., Chiaverini, S., Fusco, G.: An Odometry Calibration Method for Mobile Robots Based on the Least-Squares Technique. In: Proceedings of the American Control Conference (June 2003)
10. Stronger, D., Stone, P.: Simultaneous Calibration of Action and Sensor Models on a Mobile Robot. In: IEEE International Conference on Robotics and Automation (April 2005)
11. Röfer, T.: Evolutionary Gait-Optimization Using a Fitness Function Based on Proprioception. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 310–322. Springer, Heidelberg (2005)