doi: 10.21495/71-0-199

25th International Conference
ENGINEERING MECHANICS 2019
Svratka, Czech Republic, 13 – 16May 2019

STEP PLANNING ALGORITHM FOR N-LEGGED WALKING ROBOT WITH SMOOTH DIRECTION CONTROL

J. Králík[*], V. Venglář[**]

Abstract: *Precise step-planning algorithm is important when we want to drive an n-legged walking robot. Probably the most widely used precise control system is based on neural networks, but those require a lot of computing power. The main goal of this paper is to propose a computationally efficient method for precise and smooth movement of a robot. An algorithm capable of computing end-step position based on movement around a circle was developed. It can be driven by control vector or (with minor modification) by a radius and a tangent. It is simple to implement, adjust and debug, and needs only a small amount of computational power. The proposed algorithm only generates the end points of each step. It can be used as a basis for more complex solution and can be also used in a system with a large number of legs. The movement of body, the kinematics of mechanical parts and the manipulation of center of mass must be solved by other algorithms.*

Keywords: Walking robot, Planning Algorithm, Legged robot, Precise control, Quadruped

1. Introduction

Step-planning algorithm is fundamental when we want to drive a robot over a specific trajectory. Until now, if N-leg walking robot is to be driven, there are two possibilities to generate end-step points. The first is simple system which allows only straight walk and rotation on a current position. This solution is fast, robust and a simple to control, but when more precise control is needed, like going around an obstacle, it is insufficient. Second solution uses a neuron network to control the robot (Tran, 2014). This solution is precise, but it is demanding on computing power and every little change needs a new training of the neural network. The main idea is to make a simple, robust and precise solution to compute the end-step position for n-legged robot which allows us to drive the robot in any direction and with any radius of turning. This article examines firstly the basic idea of this algorithm, then shows how it works and finally compares advantages and disadvantages of this solution.

2. Method

Robot 2D trajectory control can be defined by translation vector (or by two orthogonal vectors) and rotation vector. Another possibility to control the robot is by path planning along a 2D curve. Main idea of this method is based on assumption that every movement along a curve or by vector can be split into system of movements along part of the circle. Every movement along a circle can be expressed as rotation around a specific point, where a translation is special case with infinite diameter. With this assumption movement along the circle can be computed if we know that the angular movement is same for all part of system. If radius is computed, coordinates of the circle and angular displacement can be computed and a precise end-step position can be determined. The algorithm is described on the system which is driven by orthogonal vector, alteration to system driven by curve is described later.

[*] Ing. Jan Králík: Institute of Solid Mechanics, Mechatronics and Biomechanics, Brno university of technology; Technická 2896/2; 616 69,Brno; CZ, Jan.Kralik1@vutbr.cz

[**] Ing. Vojtěch Venglář: Institute of Solid Mechanics, Mechatronics and Biomechanics, Brno university of technology; Technická 2896/2; 616 69,Brno; CZ, Vojtech.Venglar@vutbr.cz

The principle

This system is driven by two translation vector and one rotation vector, where vector k is in the direction of axis y, vector b is in the direction of axis x and vector z sets rotation around center. In this system it is not necessary for the center of robot (S) to be the center of rotation.

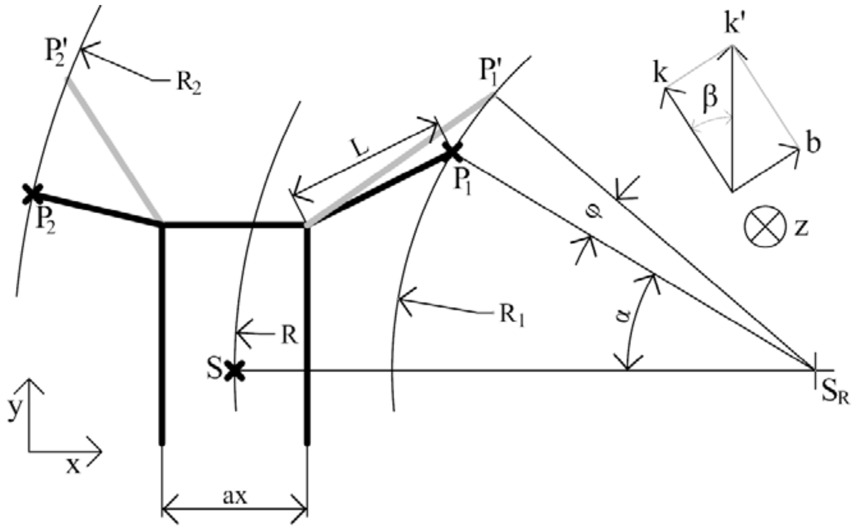

Fig. 1: computation of an end-step position

At the beginning, movement is transformed to translation in one direction. Vector k' is final translation vector and the angle β is angle between straight direction and final direction. There are special cases when only one vector is set, if the one set is k then $\beta = 0$ and $k' = k$, else if only vector b is set then $k'=|b|$ and $\beta = \frac{\pi}{2} \cdot \mathrm{sgn}(b)$

$$\beta = \tan^{-1}\left(\frac{b}{k}\right) \cdot \mathrm{sgn}(k) \tag{1}$$

$$k' = \sqrt{\mathrm{k}^2 + \mathrm{b}^2} \cdot \mathrm{sgn(k)} \tag{2}$$

In the next step, radius of the robot rotation (R) is computed, position of the center of rotation (R_x, R_y) and the angle of the robot rotation (φ). Computation of R is based on similarity of triangles

$$R = \frac{P_{nx} \cdot k'}{z \cdot \mathrm{sgn}(k)} \tag{3}$$

$$S_{rx} = |\cos(\beta) \cdot \mathrm{R}| \tag{4}$$

$$S_{ry} = |\sin(\beta) \cdot \mathrm{R}| \tag{5}$$

$$\varphi = \begin{cases} \frac{k\prime}{R} \text{ for } k' \neq 0 \\ \frac{z}{P_{nx}} \text{ for } k' = 0 \end{cases} \tag{6}$$

Final step included computation of the radius of the rotation for each leg and the angle of the leg with the robot axis (α). After that, the final position of leg can be computed and transformed to the original coordinates.

$$R_n = \sqrt{(\mathrm{P_{nx}} - S_{rx})^2 + \left(\mathrm{P_{ny}} + S_{ry}\right)^2} \tag{7}$$

$$\alpha_n = \sin^{-1}\left(\frac{P_{ny}}{R_n}\right) \cdot \mathrm{sgn}(z) \tag{8}$$

$$n_y = \sin[\alpha_n + \varphi \cdot \mathrm{sgn}(k)] \cdot R_n \cdot \mathrm{sgn}(z) \tag{9}$$

$$n_x = \{\cos[\alpha_n + \varphi \cdot \mathrm{sgn}(k)] - \cos(\alpha_n)\} \cdot R_n + P_{nx} \tag{10}$$

$$P'_{nx} = (n_x - P_{nx}) \cdot \cos(\beta) + (n_y - P_{ny}) \cdot \sin(\beta) + P_{nx} \tag{11}$$

$$P'_{ny} = \left(n_y - P_{ny}\right) \cdot \cos(\beta) - (n_x - P_{nx}) \cdot \sin(\beta) + P_{nx} \tag{12}$$

Method of application

This method is not only usable for a four-legged robot, but it is suitable for most legged robots. As we can see in fig. 2 bellow, it can be used for various types of robots and counts of legs. Furthermore, the center of robot's rotation (*S*) can be changed even during the movement. This can be helpful in situation when movement around an object without change in robot's position to that object (eg. to monitor, or turn around some object in the center of rotation) is wanted. For that situation, the center of rotation (*S*) and recalculate distance between the center of rotation and legs end-point (P_n) was changed. After that we count as the rotation on spot.

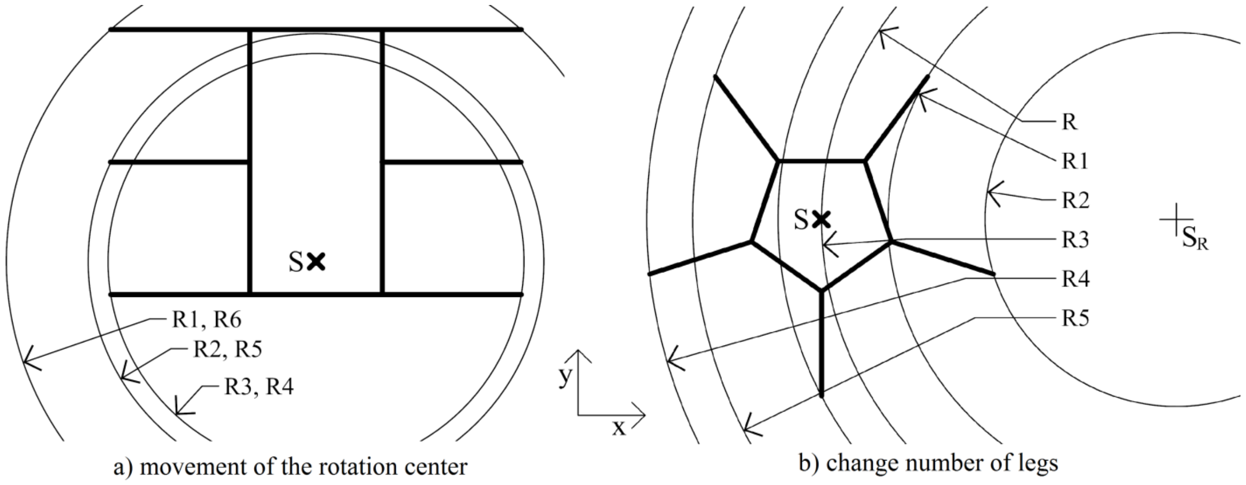

Fig. 2: possibilities of use

There is one more possibility to use this algorithm. Namely, it can be used to drive the robot along the circle, but not as in paragraph above, but in the normal mode. For that situation eq. 1-6 wasn't used and input parameters are the center (S_R) and the angle (φ) of the rotation, or the angle (φ), the rotation (β) and the radius (R). The radius (R), or the center (S_R) can be calculated from distance between S_R and S, or from β and R. Afterwards, from the outside view same result as if controlled by vector *k, b, z* is achieved, but in reality travel around a specific point, or turn with an accurate radius is made possible.

In every control system these days it is important to consider how complicated it is to implement control algorithm to a microcontroller and how much computing power it needs. This system is very simple and requires only little computing capacity. Another advantage is the possibility of processing computation in cycle, or more precisely computing eq. 1–6 before cycle and eq. 7–12 in cycle, or as matrix calculation. Algorithm is easy to implement to the microcontroller which drives the robot and every change can be easily implemented and debugged. Furthermore, if rotation (z) and translation in axis y (k) is used, equations which transform the coordinates (eq. 1, 2, 11 and 12) will be unnecessary and computation of the rotation center (eq. 3 and 4) will be simplified to $S_{rx} = R$ and $S_{ry} = 0$. Simple condition like this can simplify computation and can decrease demands on computation power.

3. Results

Algorithm results in simulated situation (*Fig. 3*) are accurate. For demonstration a system with straight movement and rotation was used, because if all three vectors had been used the final plot would have been far more confusing than the one which was used (*Fig. 3*). In comparison to simulation, the real results are incorrect (Tab. 1), but this was caused by the inaccurate servomotors and the incomplete kinematic model.

Tab. 1: comparison of simulation and the real results

Forward walk (without algorithm)			**Walk around (with algorithm)**	
Desired step size	Error (x axis)	Error (y axis)	Desired radius	Error
small (3 cm)	8 % ± 3 %	12 % ± 5 %	small (R = 20 cm)	13 % ± 4 %
big (7 cm)	6 % ± 2 %	9 % ± 5 %	big (R = 85 cm)	11 % ± 3 %

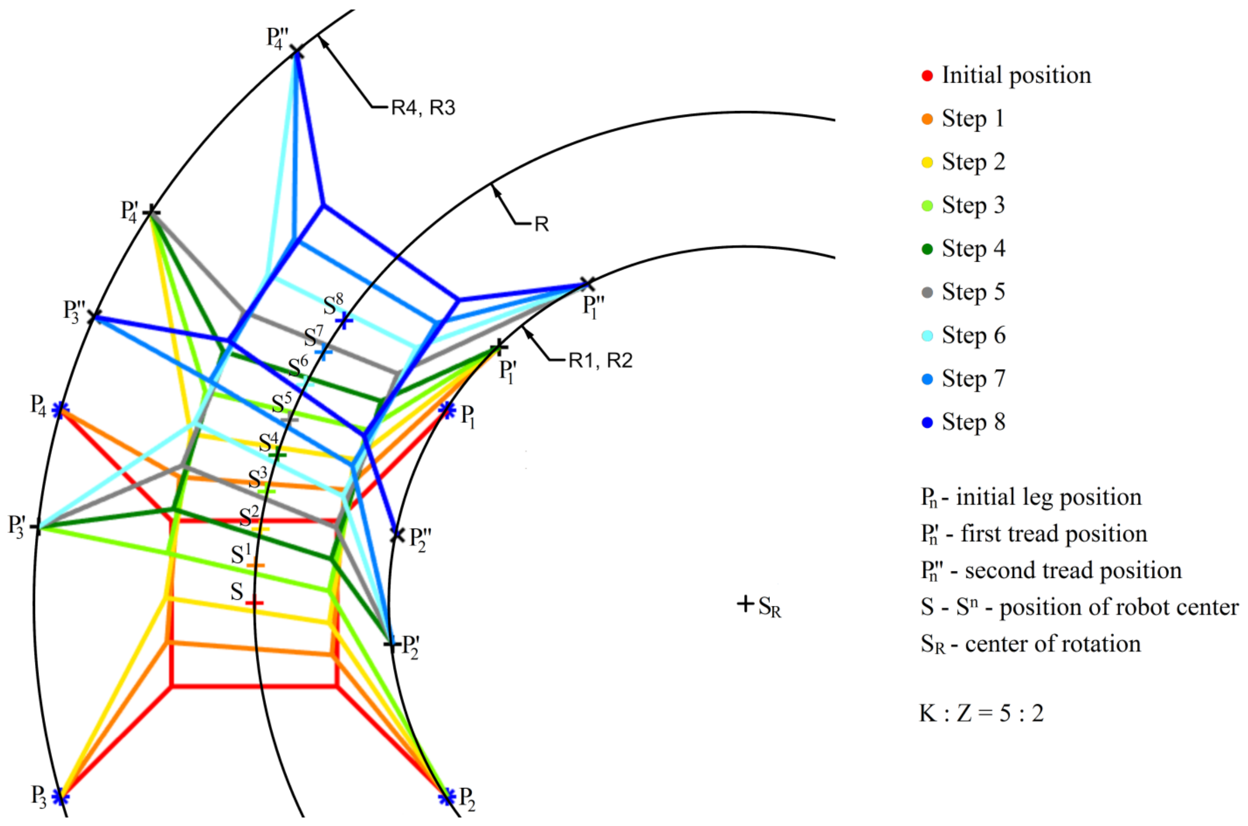

Fig. 3: test of Algorithm, ratio K : Z = 5 : 2

This became the worst disadvantage in comparison with the neural network driven system. Algorithm can only generate the end-points, which is why the kinematic of the robot and manipulation with the center of mass must be solved separately. There are three main advantages of this algorithm. First, it can be driven by vector, along a curve or by combination of both. Second, there are more possibilities of going around an obstacle (from facing the obstacle to passing by side). Third, it is easy to implement, adjust and debug (compared to neural network solution).

4. Conclusions

New step-planning algorithm for computation of the end-step position of leg for walking robot has been described in this article. The simulation was accurate but real results are different. This is caused by the tested robot. Algorithm can only generate the end-step position; everything else must be solved separately. Precise step-planning algorithm can be used as a basis for a more complex solution and can also be used in a system with a large number of legs. The system is also usable with low obstacles, but for higher obstacles 3D step-planning algorithm and further research is needed. The system is accurate enough to be used for indoor movement with simple localization, such as localization using bearing only beacons (Krejsa, 2010). This can be used in places where items need to be transported, but wheeled robot cannot be used.

Acknowledgement

This research was supported by project FSI-S-17-4334 Application of modern methods in modeling, identification and rapid prototyping of control systems for research and development of mechatronic devices.

References

Tran DT, Koo IM, Lee YH, et al. (2014) Central pattern generator based reflexive control of quadruped walking robots using a recurrent neural network. *Robotics and autonomous systems.* 62(10) pp. 1497-1516.

Krejsa, J. and Věchet S. (2010) Odometry-free mobile robot localization using bearing only beacons. *Proceedings of 14th International Power Electronics and Motion Control Conference EPE-PEMC 2010*, Ohrid, pp. T5-40-T5-45.