J Bionic Eng 15 (2018) 610–622
DOI: https://doi.org/10.1007/s42235-018-0050-8

Journal of Bionic Engineering
http://www.springer.com/journal/42235

Inverse Kinematics Analysis and COG Trajectory Planning Algorithms for Stable Walking of a Quadruped Robot with Redundant DOFs

Hyunkyoo Park[1], Bokeon Kwak[2], Joonbum Bae[2*]

1. *LG-electronics, Seoul* 08592, *Republic of Korea*
2. *Department of Mechanical Engineering*, *Ulsan National Institute of Science and Technology*, *Ulsan* 44919, *Republic of Korea*

Abstract

This paper presents a new Center of Gravity (COG) trajectory planning algorithm for a quadruped robot with redundant Degrees of Freedom (DOFs). Each leg has 7 DOFs, which allow the robot to exploit its kinematic redundancy for various locomotion and manipulation tasks. Also, the robot can suitably adapt to different environment (*e.g.*, passing through a narrow gap) by simply changing the body posture. However, the robot has significant COG movement during the leg swinging phase due to the heavy leg weights; the weight of all the four legs takes up 80% of the robot's total weight. To achieve stable walking in the presence of undesired COG movements, a new COG trajectory planning algorithm was proposed by using a combined Jacobian of COG and centroid of a support polygon including a foot contact constraint. Additionally, the inverse kinematics of each leg was solved by modified improved Jacobian pseudoinverse (mIJP) algorithm. The mIJP algorithm could generate desired trajectories for the joints even when the robot's leg is in a singular posture. Owing to these proposed methods, the robot was able to perform various modes of locomotion both in simulations and experiments with improved stability.

Keywords: legged robot, redundant degree-of-freedoms, stable walking, center-of-gravity planning

1 Introduction

A kinematically redundant manipulator is typically exploited to perform multiple tasks such as avoiding joint limit, obstacles, singularities, *etc*.[1–3]. To take advantage of a redundant manipulator's dexterity in locomotion, a robot that had four legs with 6 joints per each leg was developed in our previous work[4]. However, the robot tended to easily tumble during the walking, because the orientation of the end-effector (*i.e.*, foot) was not considered. Therefore, we have increased the number of joints per leg from 6 to 7 as shown in Fig. 1 to take the orientation into account as well as to utilize the kinematic redundancy. Compared with the similar size quadruped robot such as LittleDog[5], our work aims for high-degree of functionality and adaptation in various environments. For example, the proposed robot can walk on a plain terrain, cross over an obstacle, and pass through a narrow gap by changing its body posture. Also the robot can perform a simple task (*e.g.*, clearing a path) by using one of its leg as a manipulator. Another similar robot (but much larger than ours) named RoboSimian had dexterous four 7-DOF limbs for both locomotion and manipulation[6,7].

Many algorithmic approaches have been proposed to control redundant manipulators. In Ref. [8], the Weighted Least Norm (WLN) solution was proposed to ensure joint limit avoidance while avoiding unnecessary self-motion. Solving inverse kinematics under two constraints (*e.g.*, avoidance of obstacle and joint limit) was introduced in Ref. [9] using an extended Jacobian matrix. Also, a general scheme to manage multiple tasks using a task priority strategy was studied in Ref. [10]. However, they could not accurately follow the desired trajectory near the singularity. Here, we modified the Improved Jacobian Pseudoinverse (IJP) algorithm, which originally introduced in our previous work[4], to prevent sharp error peaks when a manipulator is close to the singularity. Detail discussion of the modified IJP algorithm will be given in section **2.2**.

Also, we consider an efficient Center of Gravity (COG) trajectory planning for stable walking. Of course,

***Corresponding author:** Joonbum Bae
E-mail: jbbae@unist.ac.kr
Hyunkyoo Park and Bokeon Kwak contributed equally to this work.

the COG based stability for a quadruped robot has been widely studied in other literatures; for example, Y-sway and E-sway motions[11], a sinusoidal sway[12], and a composite COG trajectory composed of quintic curves and straight lines[13]. However, a COG trajectory planning for a quadruped robot in particular one with heavy legs has received little attention. In this work, as the weight of all the four legs takes up 80% of the robot's total weight, a significant COG movement is inevitable during the leg swinging phase. Such COG movement easily jeopardizes the robot's walking stability. Thus, by considering a COG Jacobian and the centroid of a Support Polygon (SP) with foot contact constraints, a new COG trajectory planning is proposed for stable and efficient walking.

The remainder of this paper is organized as follows. After discussing the system configuration and the modified IJP algorithm in sections **2.1** and **2.2** respectively, we propose the COG trajectory planning method in section **2.3**. Simulation and experiment results and discussion are presented in section **3**. Finally, conclusion and future work are given in section **4**.

2 Materials and methods

2.1 Robot configuration

The robot presented in Fig. 1 has four identical legs each composed of 7 joints. The robot was tethered to an external power (12 V, 10 A) and a computer, which calculates a desired joint position and transmits back to the actuators (DYNAMIXEL: MX-106R, MX-64R, MX-24R[14]) through a serial communication (RS-485). In the case of the foot, a sphere shape was initially considered, but it caused the robot to easily be tumbled. Thus, we designed a wide but slightly curved foot by casting silicone material (Dragon skin 30[15]) to maintain enough stability for the supporting legs during the walking.

A detail configuration of the leg is given in Fig. 2. Although, a parallel leg structure has some advantages such as a centralization of mass and reduced moment of inertia[16], we chose a serial leg structure to perform multiple forms of locomotion and manipulation tasks. Also, the current leg structure may entail low speed and efficiency[17]. But, by considering torque output of the motors (3 N·m – 8 N·m), the leg structure was designed

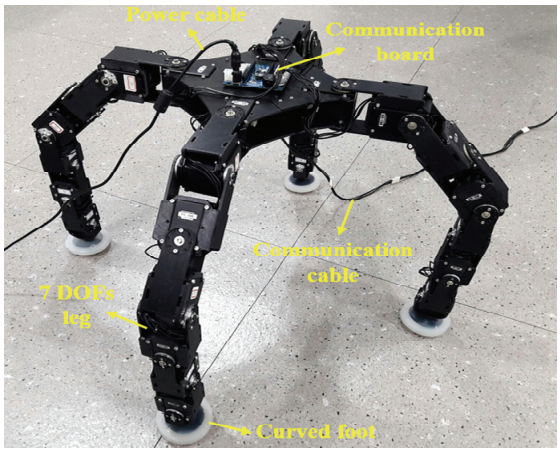


Fig. 1 The proposed quadruped robot. Each leg has 7 joints, and the whole system has 28 degrees of freedom.

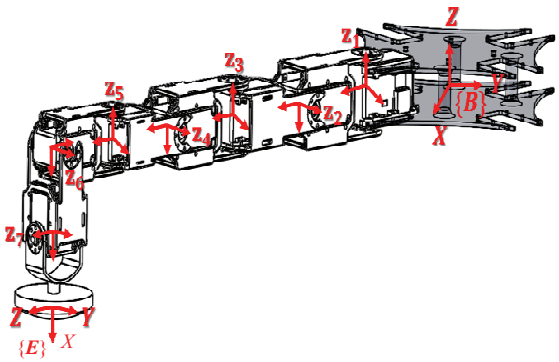


Fig. 2 Attached reference frames on the robot ({***B***}: body frame, {***E***}: end-effector frame) where *z*-axis of each reference frame coincides with the joint axis.

as simple as possible to reduce its own weight by serially connecting the seven joints. The kinematic relation between the robot's body frame {***B***} and the end-effector (*i.e.*, foot) frame {***E***} was established using Denavit-Hartenberg parameters. Note that all the four legs have exactly the same geometry and configurations. We took 3-axis positions and orientations into account to generate foot trajectory. Thus, each leg has one kinematic redundancy, which utilized to assure joint limit avoidance as discussed in section **2.2**. The total weight of the robot is 4.63 kg, the four legs take up 80% of it. The length of a fully extended leg is 48 cm, and the payload of the robot was about 500 g.

2.2 Modified IJP algorithm

First, we used differential kinematics to generate foot trajectory owing to its simple relation between the

joint space and task space velocity[18,19]. To utilize the kinematic redundancy of the leg for its joint limit avoidance, we adopted the Weighted Least Norm (WLN) solution[8]. Let $\boldsymbol{J} \in \mathbb{R}^{6\times7}$ as a single-leg Jacobian, and $\dot{\boldsymbol{x}}_d \in \mathbb{R}^6$ as a desired velocity in the task space whose 3 DOFs are for position and the remaining 3 DOFs for orientation. Then, the joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^7$ was found to be[19]:

$$\dot{\boldsymbol{q}} = \boldsymbol{W}^{-1}\boldsymbol{J}^T[\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T]^{-1}(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}), \tag{1}$$

where $\boldsymbol{K} \in \mathbb{R}^{6\times6}$ is a positive definite diagonal matrix, $\boldsymbol{e} \in \mathbb{R}^6$ is a tracking error in the task space, and $\boldsymbol{W} \in \mathbb{R}^{7\times7}$ is a diagonal matrix. Note that $\boldsymbol{e}$ was calculated by the difference between the desired joint angle and the measured angle using an encoder. Also, the diagonal elements w_i (i = 1, 2, ⋯ , 7) of $\boldsymbol{W}$ was defined as below[8]:

$$\begin{aligned} &w_i = 1 + \left|\partial \boldsymbol{H}(\boldsymbol{q}) / \partial q_i\right| \quad \text{if } \Delta\left|\partial \boldsymbol{H}(\boldsymbol{q}) / \partial q_i\right| \geq 0, \\ &\text{otherwise } w_i = 1, \end{aligned} \tag{2}$$

where, $\boldsymbol{q} \in \mathbb{R}^7$ is a joint position vector, and $\partial \boldsymbol{H}(\boldsymbol{q}) / \partial q_i$ is a performance criterion to assure joint limit avoidance. To penalize a joint that moves toward its rotational limit, the performance criterion proposed by Zghal *et al.*[20] was used:

$$\frac{\partial \boldsymbol{H}(\boldsymbol{q})}{\partial q_i} = \frac{(q_{i,\max} - q_{i,\min})^2 (2q_i - q_{i,\max} - q_{i,\min})}{4(q_{i,\max} - q_i)^2 (q_i - q_{i,\min})^2}, \tag{3}$$

where q_i is current position of joint i, $q_{i,\max}$ and $q_{i,\min}$ are the maximum and minimum rotational limit (constants), respectively. After that, the joint position vector $\boldsymbol{q}$ was obtained by Euler integration of $\dot{\boldsymbol{q}}$.

The basic idea of the IJP algorithm was calculating a given trajectory in reversed order[4]. During the reversed-order calculation, IJP algorithm requires the end-effector to be converged to the end point of a desired trajectory. In our previous work[4], this requirement was fulfilled owing to the kinematic redundancy of 3. However, as we are considering both 3-axis orientation as well as 3-axis position, the kinematic redundancy is reduced down to 1. Although IJP algorithm mostly worked well, its performance was significantly degraded when the initial posture of the leg was close to a singularity (*i.e.*, higher Condition Number (CN)). For example, Fig. 3 shows simulation results when the robot's leg

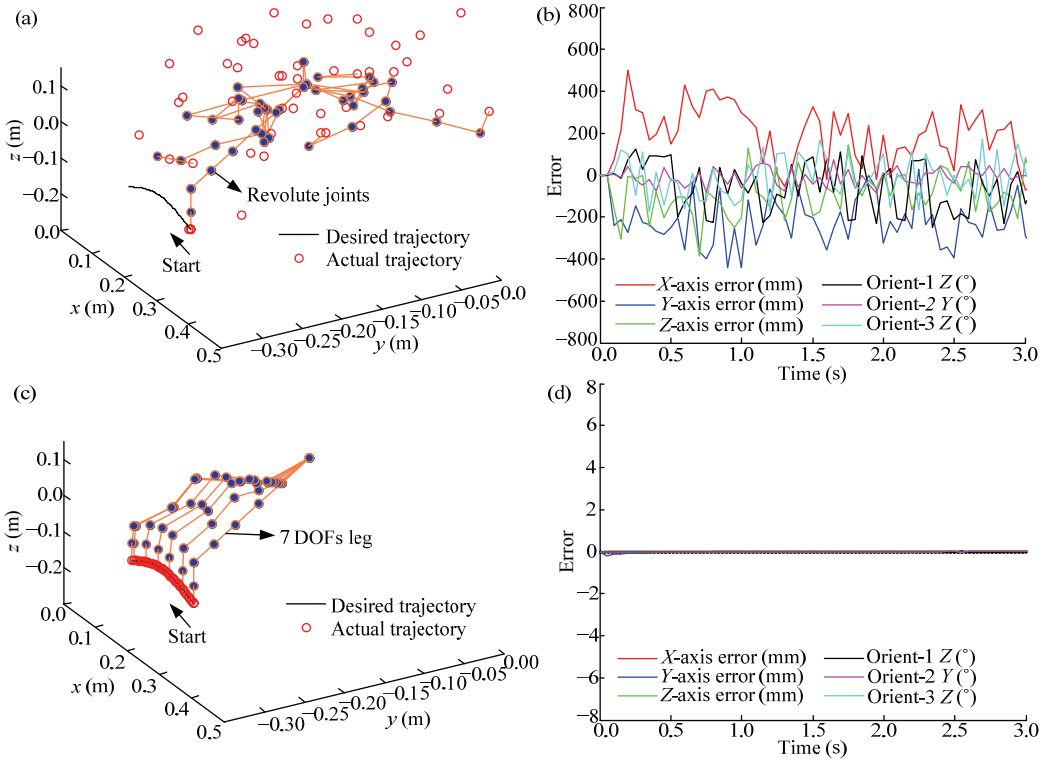


Fig. 3 Simulation results when the robot's leg is commanded to follow a desired trajectory (black line). Note that the blue circles express the revolute joints of the leg. Also, the initial posture of the leg is the same as Fig. 2, which is close to a singularity. (a, b) tracking performance and error using IJP algorithm; (c, d) using the modified IJP algorithm.

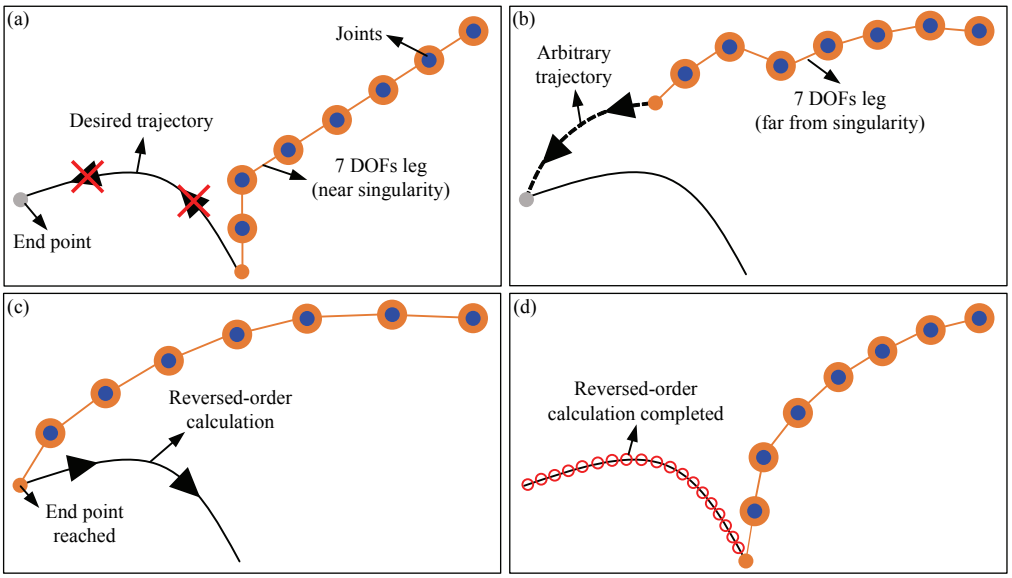


Fig. 4 Steps of the modified IJP (mIJP) algorithm in section **2.2**. (a) The leg could not follow the desired trajectory from its initial posture when $CN|_{q0} > \varepsilon$; (b) the leg follows an arbitrary trajectory generated in cubic spline manner to reach the end point $\boldsymbol{x}_{\mathrm{d,end}}$; (c) now the leg follows the reversed trajectory using Eq. (1); (d) reversed-order calculation is completed.

is commanded to follow a desired trajectory. Note that the initial posture is the same as Fig. 2, and it is close to a singularity. While using IJP algorithm, the leg could not follow the desired trajectory and exhibited significant error as shown in Figs. 3a and 3b. Here, the red circles in Figs. 3a and 3c express the actual trajectory followed by the robot.

Therefore, we modified IJP algorithm by calculating the joint position $\boldsymbol{q}_{\mathrm{end}} \in \mathbb{R}^7$ at the end point of a desired trajectory $\boldsymbol{x}_{\mathrm{d,end}} \in \mathbb{R}^6$ in different manner. If the initial posture of the leg $\boldsymbol{q}_0 \in \mathbb{R}^7$ is close to a singularity, then it could not follow a desired trajectory as shown in Fig. 4a. Instead, we let the leg follow an arbitrary trajectory as shown in Fig. 4b to reach $\boldsymbol{x}_{\mathrm{d,end}}$. Note that this step (Fig. 4b) was the only difference between IJP and modified IJP (mIJP). In previous IJP algorithm, the joint position at the end point (*i.e.*, $\boldsymbol{q}_{\mathrm{end}}$) was calculated by following the desired trajectory no matter how the tracking error was large. In other words, the way how to reach $\boldsymbol{x}_{\mathrm{d,end}}$ was changed in mIJP algorithm. Formally speaking, such arbitrary trajectory was considered when CN at $\boldsymbol{q}_0$ (*i.e.*, $CN|_{q0}$) was greater than a threshold ε; if $CN|_{q0} \le \varepsilon$, on the other hand, conventional IJP algorithm was worked well. Given that $CN|_{q0} > \varepsilon$ in Fig. 4b, $\boldsymbol{q}_0$ was firstly changed to a non-singular posture $\boldsymbol{q}_{0,n}$. A typical example of $\boldsymbol{q}_{0,n}$ is the leg position shown in Fig. 1. After that, cscvn function in Matlab[21] was used to generate an arbitrary trajectory in cubic spline manner, which connects $\boldsymbol{x}_{\mathrm{d,end}}, \boldsymbol{x}_{\mathrm{c}} \in \mathbb{R}^6$, and $f(\boldsymbol{q}_{0,n}) \in \mathbb{R}^6$ where $f(\ )$ is the forward kinematics, and $\boldsymbol{x}_{\mathrm{c}}$ is a central point between $\boldsymbol{x}_{\mathrm{d,end}}$ and $f(\boldsymbol{q}_{0,n})$. The trajectory generated by this method successfully found $\boldsymbol{q}_{\mathrm{end}}$ at the last stage of Fig. 4b.

As the leg reached to $\boldsymbol{x}_{\mathrm{d,end}}$, it started follow the reversed trajectory in Fig. 4c using Eq. (1) and completed the calculation of all the joint position in Fig. 4d. Lastly, inverting all the joint positions calculated in previous step gives the inverse kinematics solution of the original trajectory in Fig. 4a. By applying mIJP algorithm, we could significantly reduce the error with improved tracking performance as shown in Fig. 3c and Fig. 3d. Note that in both simulations (Figs. 3a and 3c) the desired trajectory was the same. However, all the joints are within their rotational limits only in Fig. 3c owing to the WLN method and mIJP algorithm.

2.3 COG trajectory planning algorithm

2.3.1 Motivation

One way to study the stability of static walking is to examine the Support Polygon (SP) formed by supporting legs (*i.e.*, stance legs)[22]. As long as the projection of the COG lies inside the SP, the robot is statically stable. Therefore, a gait sequence that can maximize the

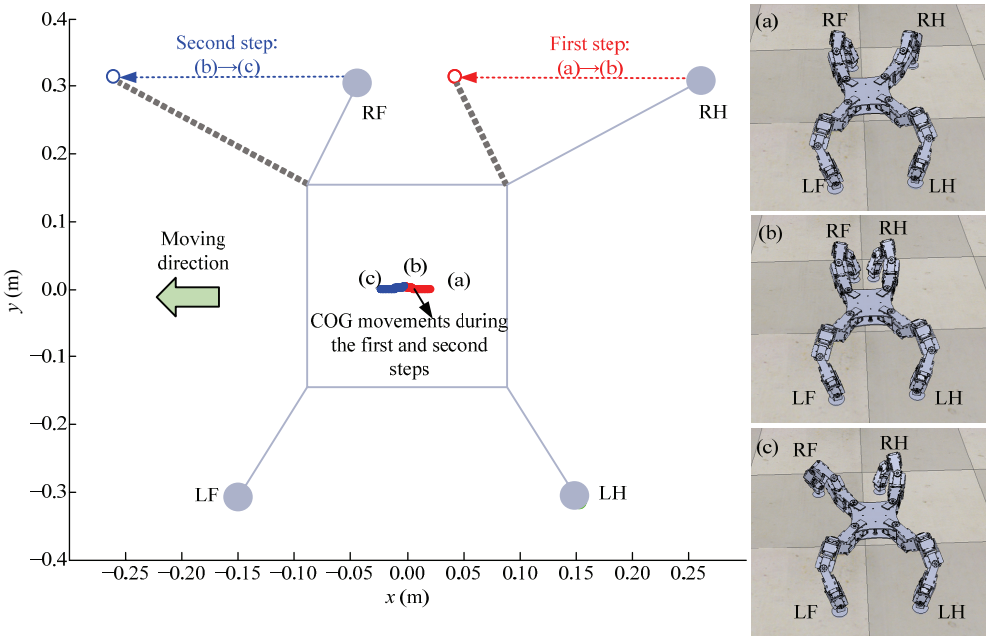


Fig. 5 The undesired COG movement during the sequential swinging of RH and RF legs. (a) Before the leg swinging; (b) after RH leg swinging (first step); (c) after RF leg swinging (second step). The red and blue traces show the COG movement during the first and second leg swinging, respectively.

stability during walking should be considered. In static walking, six leg sequences are plausible, but only one sequence can make the COG stay inside the SP at all times with maximum stability[22]; namely, Right-Hind (RH) leg, Right-Front (RF) leg, Left-Hind (LH) leg, and Left-Front (LF) leg. In this work, we chose this walking pattern during the simulation and experiment for statically stable walking.

From the gait sequence (RH, RF, LH, LF), we considered the intersection of two subsequent support polygons, which is referred to as Double Support Polygon (DSP). Obviously, the COG also should remain inside the DSP for stable walking. However, undesired COG movement during the leg swinging phase happened due to its heavy weight. Fig. 5 shows the simulation result obtained from V-REP (Coppelia Robotics[23]) when RH and RF legs are swinging sequentially. Fig. 5a shows the initial posture before the leg swinging, while Figs. 5b and 5c show the robot after swinging its RH leg and RF leg, respectively. Note that red and blue lines express the trace of COG movement during the sequential leg swinging. Due to that COG movement the robot often tumbled during the walking, which implies that the COG indeed crossed the DSP. Although we cannot avoid such undesired COG movement, we can at least prevent the COG from crossing the DSP. For this reason, we propose a new COG trajectory planning method to shift the COG toward a critical point where the COG is guaranteed to remain inside the DSP while minimizing unnecessary motions.

2.3.2 Longitudinal Stability Margin (LSM)

Before going through the planning algorithm, we briefly discuss the stability criterion that we used in this section. To evaluate the stability of the walking, we employed Longitudinal Stability Margin (LSM)[24,25] owing to its ease of implementation and intuition. Note that an inertia effect (*e.g.*, ZMP[26]) was not considered in static stability of walking. Consider a Support Polygon (SP) formed by three supporting legs (*i.e.*, in stance phase) as shown in Fig. 6, while only one leg is in the middle of swinging phase. Given that the vertical projection of COG, we considered two longitudinal distances d_1 and d_2. The distance between the front edge and the COG is d_1, while d_2 measures the distance between the rear edge and the COG. Note that both d_1 and d_2 are parallel with the moving direction. Here, the *LSM* is found to be $LSM = \min(d_1, d_2)$[27]. As long as the *LSM* is greater than or equal to a Minimum Stability Margin (MSM), the robot is stable during the walking. After

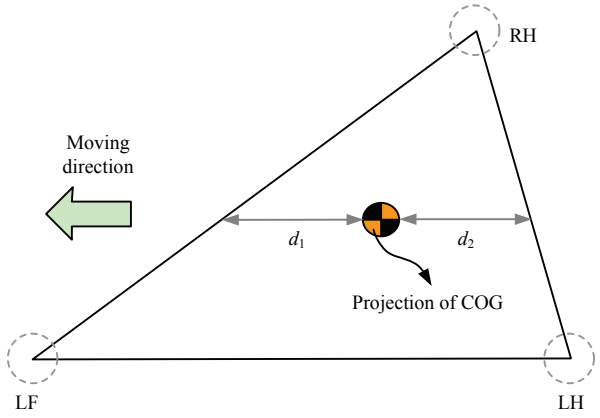


Fig. 6 A support polygon formed by three supporting legs, while one leg (in this case, RF leg) is in the middle of swinging phase. The Longitudinal Stability Margin (LSM) is found to be: $LSM = \min(d_1, d_2)$.

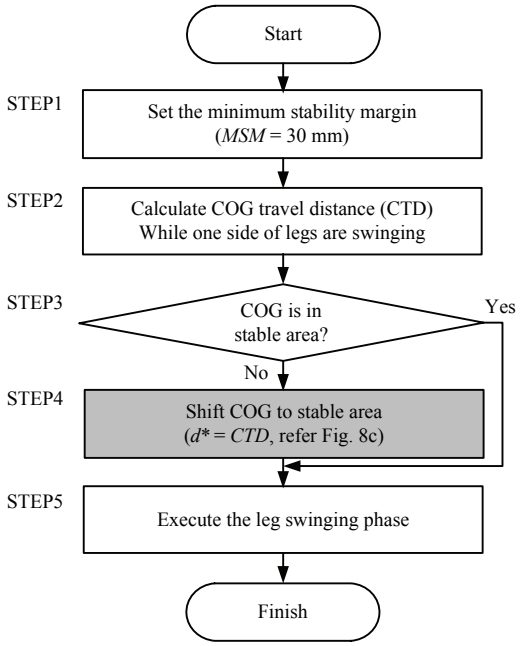


Fig. 7 The flowchart of COG trajectory planning algorithm.

consideration of *LSM*, now we present the COG trajectory planning algorithm in next section.

2.3.3 COG trajectory planning algorithm

Overall, the proposed algorithm consists of five steps as shown in Fig. 7. Among them, STEP 4 is the key stage where COG shifting is performed if the *LSM* criterion is expected to be violated in next walking. The detailed methods are described as follows:

STEP1: As discussed earlier, the robot is stable when the *LSM* is greater than or equal to *MSM*. Ideally, *MSM* may be considered to be 0; however, by considering the heavy weight of the legs we set the *MSM* as 30 mm in this work.

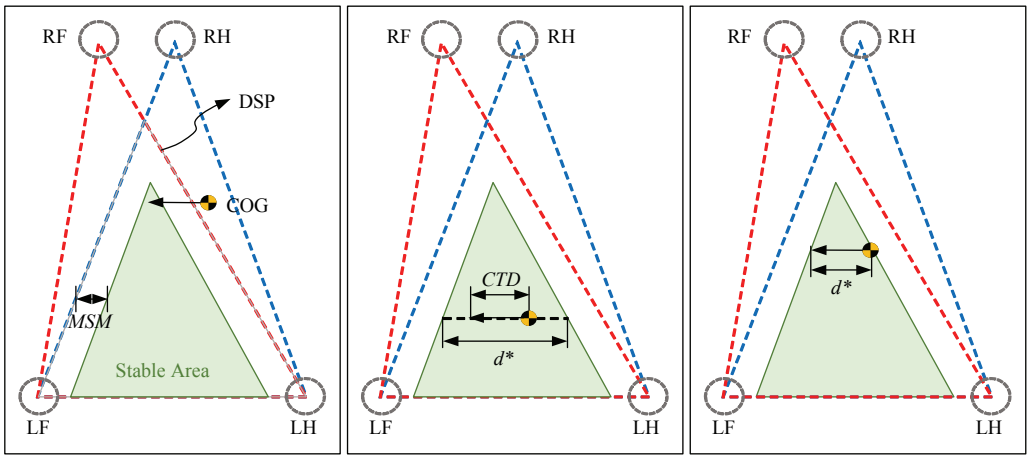
STEP2: The COG Traveled Distance (*CTD*) is calculated using the kinematic model of the robot while the legs of one side are sequentially swinging in forward. Note that *CTD* refers the longitudinal distance of red and blue traces of COG in Fig. 5. Also, the robot has not yet actuated its joints at this stage.

STEP3: Check whether the COG trajectory is within the stable area or not. If it is (*i.e.*, the *LSM* criterion holds), then move to STEP5; otherwise, move to STEP4.

STEP4: The goal of this step is shifting the COG to a stable area where the whole COG trajectory satisfies ($LSM \geq MSM$). In doing so, we tried to minimize unnecessary shifting motion. Suppose that the *CTD* was calculated in STEP2 while right side legs are sequentially swinging (again, without actual movements), and then projected in Fig. 8a. Note that the red and blue triangles express the SP before RH and RF leg swinging, respectively. Also, the stable area (green triangle) is smaller than the DSP since the *MSM* was set to 30 mm in STEP1. Clearly, the whole COG trajectory (black arrow) is deviated from the stable area, which causes the robot to be tumbled while actually performing the RH and RF leg swinging. Therefore, it is required to shift the COG trajectory inside the stable area before executing the leg swing phase.

After arbitrary shifting of the COG trajectory, one may result in Fig. 8b; although the whole COG trajectory is inside the stable area, it involves excessive transversal movement and therefore inefficient. By letting d^* as a distance, which coincides with the COG trajectory, between the front and rear edges of the stable area, the *CTD* is not equal to d^* in Fig. 8b. On the other hand, if we shift the COG trajectory just until $CTD = d^*$ satisfied as shown in Fig. 8c, any unnecessary transversal movement is minimized. Note that *CTD* is constant in Fig. 8.

To perform COG shifting, utilizing a fixed global frame would be useful; however, such global frame was not readily available in our experiment environment. Instead, we employed a movable body frame $\{\boldsymbol{B}\}$ in Fig. 2. By letting $\boldsymbol{P}_{\mathrm{COG}} \in \mathbb{R}^3$ and $\boldsymbol{P}_{\mathrm{Cent}} \in \mathbb{R}^3$ as the position vector of the robot's COG and the centroid of the SP



(a) Before COG shifting (unstable) (b) Stable, but inefficient ($d^* \neq CTD$) (c) Stable, and efficient ($d^* = CTD$)

Fig. 8 COG trajectory shifting plan. The red and blue triangles express SP before RH and RF leg swinging, respectively. While the green SP is a stable area where $LSM \geq MSM$ holds. (a) Unstable COG trajectory before shifting; (b) stable but inefficient COG trajectory shifting; (c) stable and efficient COG trajectory shifting.

seen from the body frame, respectively, we defined $\boldsymbol{P}_{\mathrm{S}} = \boldsymbol{P}_{\mathrm{COG}} - \boldsymbol{P}_{\mathrm{Cent}}$. After that a combined Jacobian $\boldsymbol{J}_{\mathrm{S}} \in \mathbb{R}^{3\times 28}$ of the COG and centroid was considered as:

$$\dot{\boldsymbol{P}}_{\mathrm{S}} = \boldsymbol{J}_{\mathrm{S}}\dot{\boldsymbol{q}}_{\mathrm{all}}, \tag{4}$$

where $\dot{\boldsymbol{q}}_{\mathrm{all}} \in \mathbb{R}^{28}$ is a joint velocity of the whole body. In addition, we added foot contact constraints to fix the shape of SP while shifting the COG as:

$$\boldsymbol{P}_{\mathrm{leg},i} - \boldsymbol{P}_{\mathrm{leg},j} = \text{Constant}, \tag{5}$$

$$\dot{\boldsymbol{P}}_{\mathrm{leg},i} - \dot{\boldsymbol{P}}_{\mathrm{leg},j} = 0, \tag{6}$$

where $\boldsymbol{P}_{\mathrm{leg},i} \in \mathbb{R}^3$ and $\boldsymbol{P}_{\mathrm{leg},j} \in \mathbb{R}^3$ are the position of the foot that connected to leg i and j, respectively, and $\dot{\boldsymbol{P}}_{\mathrm{leg},i} \in \mathbb{R}^3$ and $\dot{\boldsymbol{P}}_{\mathrm{leg},j} \in \mathbb{R}^3$ are their time derivatives. In joint space expression, Eq. (6) can be reformulated as:

$$\begin{bmatrix} \boldsymbol{J}_{\mathrm{leg},1} & -\boldsymbol{J}_{\mathrm{leg},2} & 0 & 0 \\ 0 & \boldsymbol{J}_{\mathrm{leg},2} & -\boldsymbol{J}_{\mathrm{leg},3} & 0 \\ 0 & 0 & \boldsymbol{J}_{\mathrm{leg},3} & -\boldsymbol{J}_{\mathrm{leg},4} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\mathrm{leg},1} \\ \dot{\boldsymbol{q}}_{\mathrm{leg},2} \\ \dot{\boldsymbol{q}}_{\mathrm{leg},3} \\ \dot{\boldsymbol{q}}_{\mathrm{leg},4} \end{bmatrix} = 0, \tag{7}$$

where $\boldsymbol{q}_{\mathrm{leg},i} \in \mathbb{R}^7$ is a joint angle vector of leg i and $\boldsymbol{J}_{\mathrm{leg},i} \in \mathbb{R}^{3\times 7}$ is corresponding Jacobian matrix for $i = 1, 2, 3, 4$. Note that the column vector $[\dot{\boldsymbol{q}}_{\mathrm{leg},1}\ \dot{\boldsymbol{q}}_{\mathrm{leg},2}\ \dot{\boldsymbol{q}}_{\mathrm{leg},3}\ \dot{\boldsymbol{q}}_{\mathrm{leg},4}]^{\mathrm{T}}$ is equivalent to $\dot{\boldsymbol{q}}_{\mathrm{all}}$. To incorporate the foot contact constraints Eq. (7) into Eq. (4), we rearranged the whole 28 joints into two groups; namely, alpha joints $\boldsymbol{q}_\alpha \in \mathbb{R}^{12}$ and beta joints $\boldsymbol{q}_\beta \in \mathbb{R}^{16}$. Then, Eq. (7) is rewritten as:

$$\begin{bmatrix} \boldsymbol{H}_\alpha & \boldsymbol{H}_\beta \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_\alpha \\ \dot{\boldsymbol{q}}_\beta \end{bmatrix} = 0, \tag{8}$$

where $\dot{\boldsymbol{q}}_\alpha$ and $\dot{\boldsymbol{q}}_\beta$ are time derivatives of $\boldsymbol{q}_\alpha$ and $\boldsymbol{q}_\beta$, and $\boldsymbol{H}_\alpha \in \mathbb{R}^{9\times 12}$ and $\boldsymbol{H}_\beta \in \mathbb{R}^{9\times 16}$ are collection of columns that correspond to $\dot{\boldsymbol{q}}_\alpha$ and $\dot{\boldsymbol{q}}_\beta$, respectively. The alpha joint position $\boldsymbol{q}_\alpha$ was designated to have a direct relationship with a desired $\boldsymbol{P}_{\mathrm{S}}$, which the robot has to follow. On the other hand, the beta joint position $\boldsymbol{q}_\beta$ was determined by $\boldsymbol{q}_\alpha$; in other words, $\boldsymbol{q}_\beta$ is indirectly related with $\boldsymbol{P}_{\mathrm{S}}$. To ensure the joint limit avoidance of both $\boldsymbol{q}_\alpha$ and $\boldsymbol{q}_\beta$, we again utilized their kinematic redundancy. Since the dimension of $\boldsymbol{P}_{\mathrm{S}}$ was only 3, the dimension of $\boldsymbol{q}_\alpha$ should be greater than 3 to retain the redundancy. Similarly, the dimension of $\boldsymbol{q}_\beta$ is required to be greater than the dimension of $\boldsymbol{q}_\alpha$. Therefore, we equally, assigned three joints per each leg to the alpha joint group, while the rest of 28 joints (*i.e.*, the remaining 16 joints) belong to the beta joint group. Then, Eq. (8) is rewritten as:

$$\dot{\boldsymbol{q}}_\beta = -\boldsymbol{H}_\beta^{+}\boldsymbol{H}_\alpha\dot{\boldsymbol{q}}_\alpha, \tag{9}$$

where $\boldsymbol{H}_\beta^{+} \in \mathbb{R}^{16\times 9}$ is a pseudo-inverse of $\boldsymbol{H}_\beta$. By incorporating the above foot contact constraint, Eq. (4) is now rewritten as:

$$\dot{\boldsymbol{P}}_{\mathrm{S}} = \boldsymbol{J}_{\mathrm{S}}\dot{\boldsymbol{q}}_{\mathrm{all}} = \boldsymbol{J}_{\mathrm{S}}\begin{bmatrix} \boldsymbol{I}_{12} \\ -\boldsymbol{H}_\beta^{+}\boldsymbol{H}_\alpha \end{bmatrix}\dot{\boldsymbol{q}}_\alpha = \boldsymbol{J}_{\mathrm{C}}\dot{\boldsymbol{q}}_\alpha, \tag{10}$$

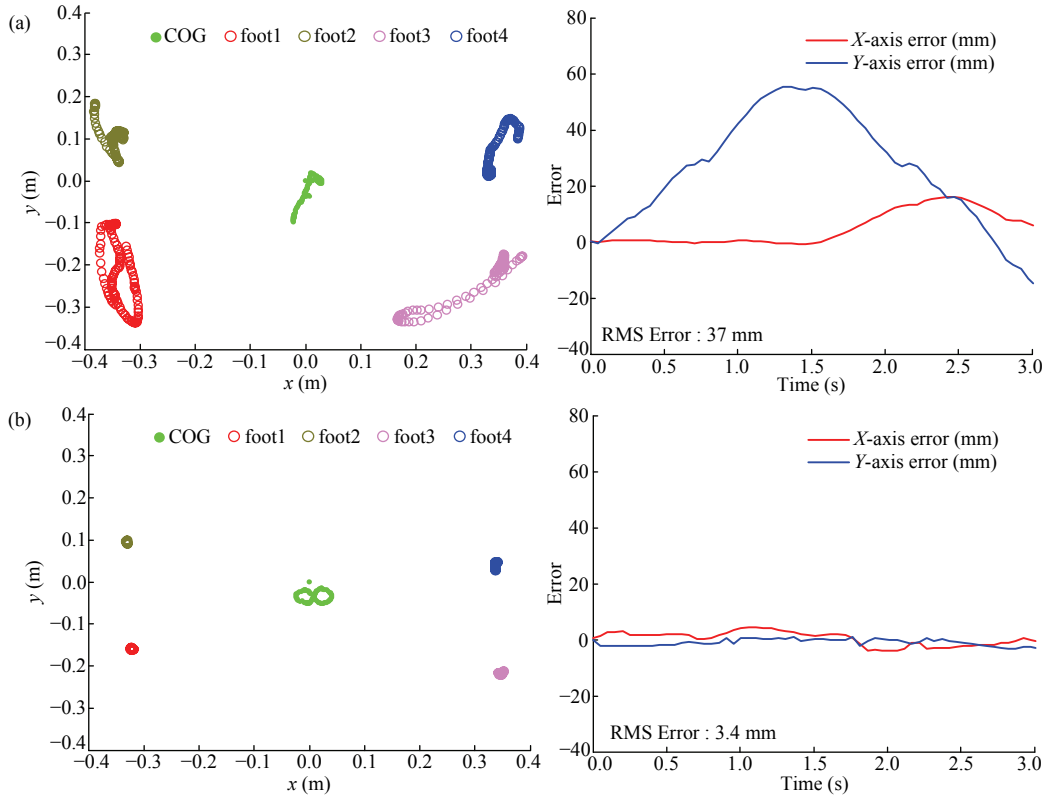


Fig. 9 The robot was commanded to follow its COG to a figure. 8 (*i.e.*, numeric number eight) trajectory in V-REP simulator. (a) Without using the foot contact constraints; (b) with using the foot contact constraints.

where $\boldsymbol{J}_{\mathrm{C}} \in \mathbb{R}^{3\times12}$ is the combined Jacobian with foot contact constraints. Finally, by applying the WLN solution in Eq. (1) to $\dot{\boldsymbol{P}}_{\mathrm{S}} = \boldsymbol{J}_{\mathrm{C}}\dot{\boldsymbol{q}}_{\alpha}$, the alpha joint velocity $\dot{\boldsymbol{q}}_{\alpha}$ is found to be:

$$\dot{\boldsymbol{q}}_{\alpha} = \boldsymbol{W}_{\alpha}^{-1}\boldsymbol{J}_{\mathrm{C}}^{\mathrm{T}}[\boldsymbol{J}_{\mathrm{C}}\boldsymbol{W}_{\alpha}^{-1}\boldsymbol{J}_{\mathrm{C}}^{\mathrm{T}}]^{-1}\dot{\boldsymbol{P}}_{\mathrm{S}}, \tag{11}$$

where $\boldsymbol{W}_{\alpha} \in \mathbb{R}^{12\times12}$ is a diagonal matrix whose diagonal elements are defined to be Eq. (2). By multiplying $\boldsymbol{H}_{\beta}$ on the both side of Eq. (9), $\boldsymbol{H}_{\alpha}\dot{\boldsymbol{q}}_{\alpha} = \boldsymbol{H}_{\beta}\dot{\boldsymbol{q}}_{\beta}$ is obtained. By also taking the WLN solution in Eq. (1) to $\boldsymbol{H}_{\alpha}\dot{\boldsymbol{q}}_{\alpha} = \boldsymbol{H}_{\beta}\dot{\boldsymbol{q}}_{\beta}$, the beta joint velocity $\dot{\boldsymbol{q}}_{\beta}$ is found to be:

$$\dot{\boldsymbol{q}}_{\beta} = -\boldsymbol{W}_{\beta}^{-1}\boldsymbol{H}_{\beta}^{\mathrm{T}}[\boldsymbol{H}_{\beta}\boldsymbol{W}_{\beta}^{-1}\boldsymbol{H}_{\beta}^{\mathrm{T}}]^{-1}\boldsymbol{H}_{\alpha}\dot{\boldsymbol{q}}_{\alpha}, \tag{12}$$

where $\boldsymbol{W}_{\beta} \in \mathbb{R}^{16\times16}$ is again a diagonal matrix and its diagonal elements are found from Eq. (2). Once the desired $\boldsymbol{P}_{\mathrm{S}}$ is given, the robot can shift its COG to a specific point or can follow a desired trajectory by Eqs. (11) and (12). To demonstrate their applicability, a simulation was performed and the robot's COG was commanded to follow a figure. 8 (*i.e.*, numeric number eight like) trajectory. Fig. 9a shows the COG trajectory followed by the robot (green trace) and its error without using the constraint Eq. (7); more specifically, our previous method[4] was used to generate a trajectory without foot constraints by directly applying the pseudo-inverse to Eq. (4). Note that the foot contact points are also depicted, but they are not remained constant as shown in Fig. 9a. On the other hand, applying the foot contact constraints yield better performance with much reduced error as presented in Fig. 9b. Although there was a minute foot slippage during the simulation, each foot holds almost the same position.

STEP5: Finally, the robot executes the leg swinging phase and repeats the algorithm for the next phase.

3 Results and discussion

3.1 Simulation results

To test the locomotion capability of the robot and the proposed algorithms, V-REP simulator was used. First, we simulated the robot to walk forward using the gait sequence (RH, RF, LH, LF) as shown in Fig. 10. The mIJP algorithm was used to generate all the foot trajectories, while the planning algorithm in Fig. 7 was used to shift the COG. In Fig. 10a, the robot shifted its COG

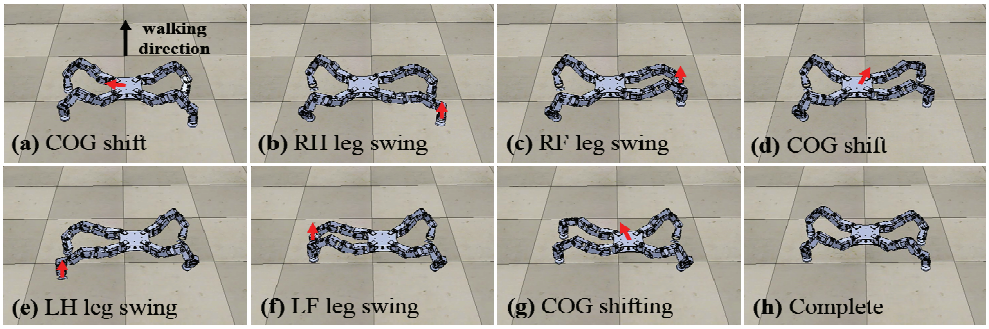


Fig. 10 Simulation of one cycle of walking in V-REP simulator. The red arrows imply the movement of either the robot's body or its legs.

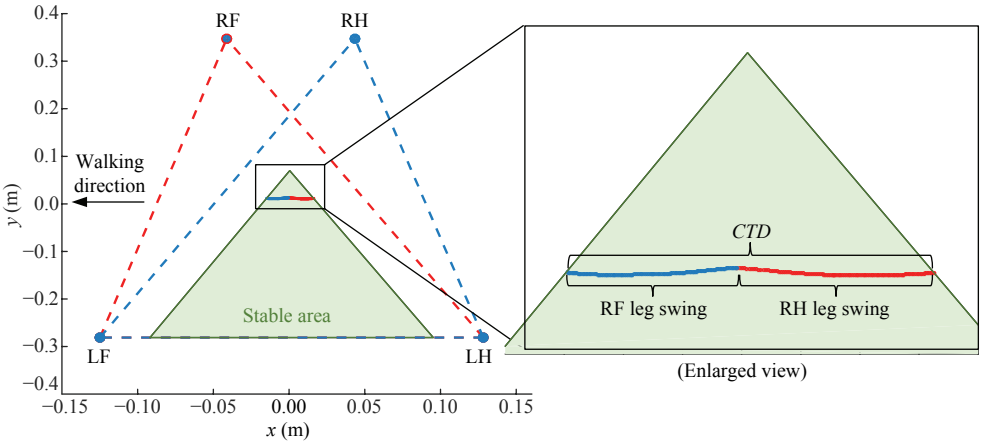


Fig. 11 COG shifting in Fig. 10a before RH and RF leg swinging. The red and blue trace shows the undesired COG movement during RH and RF leg swinging, respectively. Note that COG trajectory is shifted inside the stable area as in Fig. 8c.

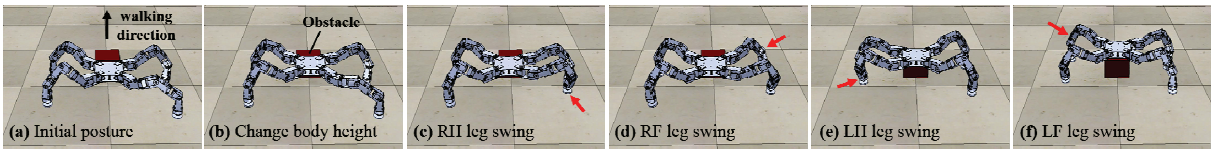


Fig. 12 The robot overcame a 15 cm height obstacle by elevating its body height in (b) while walking forward.

trajectory before starting the RH and RF legs swing phase. As a result, the COG trajectory was shifted inside the stable area as shown in Fig. 11 and it satisfied $d^* = CTD$ criterion as discussed in Fig. 8c. Note that the red and blue trace shows the undesired COG movement during the RH and RF leg swinging, respectively. However, the robot could walk stably because the COG trajectory lied inside the stable area. The robot successfully finished the rest of the leg swing phase and COG shifting to complete one cycle of walking.

Owing to the robot's long legs and redundant DOFs, the robot could elevate its body height up to 180% from the initial posture. We put 15 cm height of obstacle in front of the robot and applied the same algorithms as before. By simply changing the body height as shown in Fig. 12b, the robot was easily able to overcome the obstacle and continued to walk forward.

Next, we tested a scenario when the robot has to use one of its legs as a manipulator to clear the path. We put a small cylinder in front of the robot as shown in Fig. 13. In this case, LF leg was used as a manipulator while the robot was walking forward. After completing the RH, RF, and LH leg swing phases, the robot shifted its COG inside the stable area formed by the three supporting feet (*i.e.*, RH, RF, LH legs) using the algorithm in Fig. 7. This ensured the robot to stand stable while clearing the cylinder using the LF leg as shown in Fig. 13e.

Additionally, we demonstrated a different form of

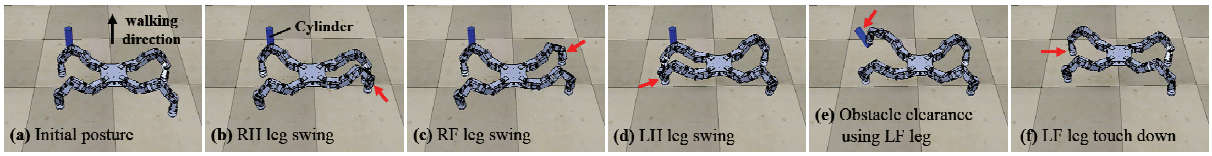


Fig. 13 Obstacle clearance by using one of its legs as a manipulator.

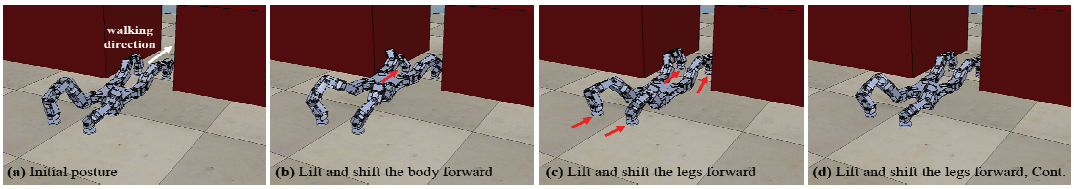


Fig. 14 The robot passed through a narrow gap (width: 40 cm) by using crawling gait.

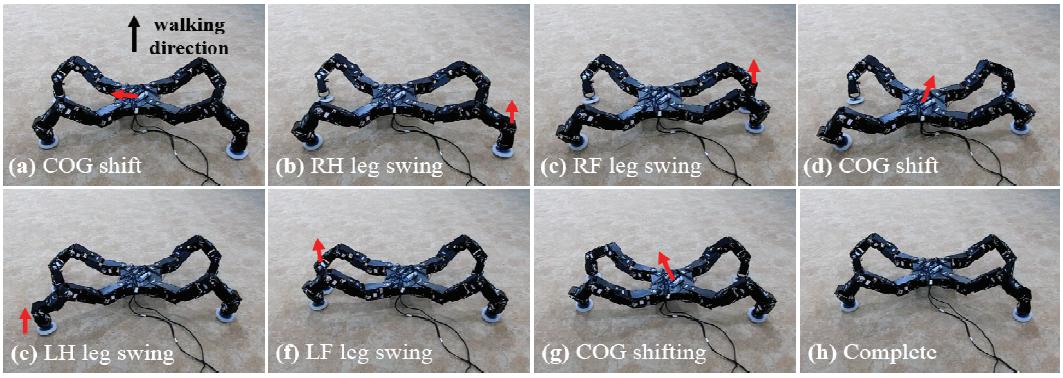


Fig. 15 The walking experiment of the robot as in Fig. 10. The red arrows imply the movement of either the robot's body or its legs.

locomotion other than the nominal walking gait (RH, RF, LH, LF) to test the locomotory ability. Here, we simulated the robot to pass through a narrow gap (width: 40 cm) using the crawling gait as shown in Fig. 14. First, the robot lifted and then shifted its body forward in Fig. 14b. While in Figs. 14c and 14d, the robot moves all of the legs in forward to start the next crawling. Since the body was contacted to the ground except in Fig. 14b, the stability needed not to be considered; however, both Eqs. (11) and (12) are still useful at shifting the body forward.

3.2 Experimental verification

Same as the simulation, walking experiments were performed as shown in Fig. 15 using the walking gait (RH, RF, LH, LF). During the experiment, the robot was tethered to a power source to actuate the motors. Also, all the computations were performed by an external computer. The mIJP algorithm was used to generate the foot trajectories, while the walking stability was maintained using the COG trajectory algorithm. The robot shifted the COG trajectory in Fig. 15a before swinging the RH and RF legs in Figs. 15b and 15c. Thereafter, the robot repeated the COG shifting and leg swinging to complete one step of walking. The robot could traverse 13 cm forward in single walking cycle and achieved a statically stable gait owing to the proposed methods. It took 9 seconds to complete all the sequence of walking shown in Fig. 15, and therefore average moving speed was 1.4 $cm \cdot s^{-1}$.

The body-height change during the walking was also demonstrated. A 15 cm height of obstacle was set in front of the robot whose initial body height was only 8.5 cm as shown in Fig. 16a. However, it was sufficient to overcome the obstacle as long as the robot elevated its body height more than 15 cm. After changing the body height in Fig. 16b, the robot continued to walk forward using the same (RH, RF, LH, LF) waking gait with the help of the modified IJP algorithm and COG trajectory planning.

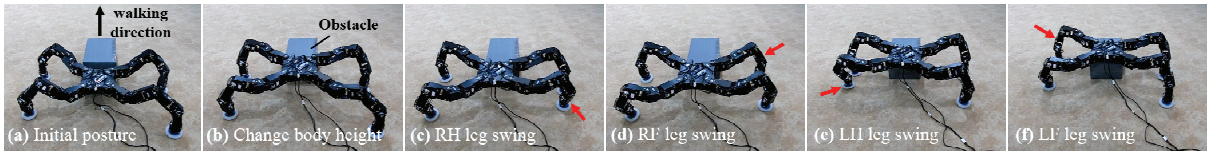


Fig. 16 The robot elevated its body height to overcome the obstacle (height: 15 cm) and continued to walk as simulated in Fig. 12.

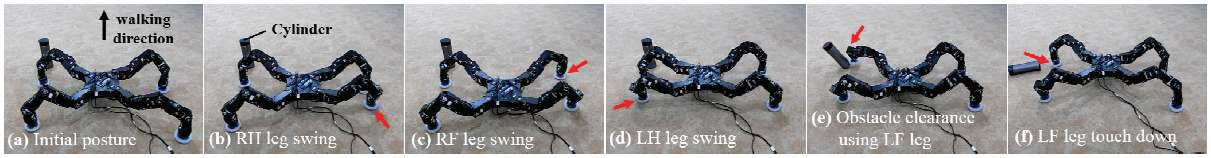


Fig. 17 Clearance of the cylindrical obstacle using the LF leg during the walking as simulated in Fig. 13.

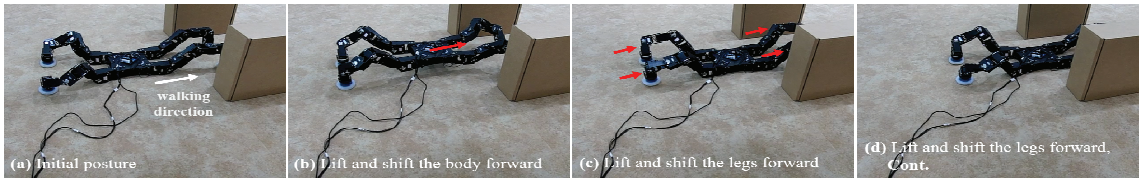


Fig. 18 The robot passed through a narrow gap (width: 45 cm) by using a crawling gait as simulated in Fig. 14.

Also, the robot was tested to clear a cylindrical obstacle while walking, and the result is shown in Fig. 17. After sequential completion of RH, RF, and LH leg swinging, the robot relocated its COG inside the stable area formed by the three supporting legs using both Eqs. (11) and (12). After that, the robot used its LF leg as a manipulator to clear the obstacle as shown in Fig. 17e. Without the prior COG relocation, the robot was tumbled during the obstacle clearance. This experiment demonstrated that the robot is able to perform a simple manipulation task by using one of its legs.

Lastly, the robot passed through a narrow gap (width: 45 cm) by changing its nominal walking gait to crawling gait in Fig. 18. The robot lifted and then shifted its body forward in Fig. 18b. Note that the body shifting motion was generated using the both Eqs. (11) and (12). After that, the robot moved all of its legs forward in Figs. 18c and 18d to initiate the next crawling gait using the modified IJP algorithm and average moving speed was 2.6 cm·sec^{-1}. The robot is expected to perform various modes of locomotion in future work by utilizing its redundant DOFs. By comparing some related works[5–7,12,13], this paper has addressed that how a quadruped robot with redundant DOFs can efficiently shift its COG and walk in statically stable manner when the total weight of the legs is much heavier than the body; in this work, all the four legs takes up 80% of the robot's total weight.

4 Conclusion

In this paper, a quadruped robot with 28 DOFs was proposed to achieve various modes of locomotion and manipulation task. The inverse kinematics of each leg was handled by the modified Improved Jacobian Pseudoinverse (mIJP) algorithm, and it could generate an accurate trajectory even when the initial posture of the leg is close to a singularity. This algorithm was used to generate all the desired foot trajectories for walking or a simple manipulation task. Especially, a new COG trajectory planning algorithm was proposed by combining the Jacobian of COG and centroid of the support polygon including foot contact constraints. This algorithm enabled the robot to shift its COG trajectory inside the stable area while minimizing unnecessary motion. As a result, the robot was able to walk stably in both simulation and experiment. Additional tests were also performed to demonstrate various modes of locomotion such as varying the body height, obstacle clearance, and passing through a narrow gap.

As a future work, we are planning to use sensor feedback (*e.g.*, IMU sensor, force sensor, *etc.*) to detect obstacles or different terrain. This will enable the robot

to overcome more complex environment autonomously. In addition, it is necessary to improve the computation speed to enhance the reflex against environmental disturbance.

Acknowledgment

This work was supported by the 2018 Research Fund (1.180015.01) of UNIST (Ulsan National Institute of Science and Technology), the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2015R1C1A1A01053763), and the NRF Grant funded by the Korean Government (MSIT) (No. NRF-2016R1A5A1938472).

References

[1] Chiaverini S, Oriolo G, Walker I D. Kinematically redundant manipulators. In: Siciliano B, Khatib O, eds., *Handbook of Robotics*, Springer, Berlin-Heidelberg, Germany, 2008, 245–268.

[2] Omrčen D, Žlajpah L, Nemec B. Compensation of velocity and/or acceleration joint saturation applied to redundant manipulator. *Robotics and Autonomous Systems*, 2007, **55**, 337–344.

[3] Zhao Y J, Jin L, Zhang P, Li J Y. Inverse displacement analysis of a hyper-redundant elephant's trunk robot. *Journal of Bionic Engineering*, 2018, **15**, 397–407.

[4] Kwak B, Park H, Bae J. Development of a quadruped robot with redundant DOFs for high-degree of functionality and adaptation. *Proceedings of IEEE International Conference on Advanced Intelligent Mechatronics* (*AIM*), Banff, Canada, 2016, 608–613.

[5] Shkolnik A, Tedrake R. Inverse kinematics for a point-foot quadruped robot with dynamic redundancy resolution. *Proceedings of IEEE International Conference on Robotics and Automation* (*ICRA*), Rome, Italy, 2007, 4331–4336.

[6] Byl K, Byl M, Satzinger B. Algorithmic optimization of inverse kinematics tables for high degree-of-freedom limbs. *Proceedings of ASME Dynamic Systems and Control Conference* (*DSCC*), San Antonio, USA, 2014.

[7] Byl K, Byl M. Design of fast walking with one-versus two-at-a-time swing leg motions for RoboSimian. *Proceedings of IEEE International Conference on Technologies for Practical Robot Applications* (*TePRA*), Woburn, USA, 2015, 1–7.

[8] Chan T F, Dubey R V. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, 1995, **11**, 286–292.

[9] Sciavicco L, Siciliano B. A solution algorithm to the inverse kinematic problem for redundant manipulators. *IEEE Journal on Robotics and Automation*, 1988, **4**, 403–410.

[10] Siciliano B, Slotine J-J E. A general framework for managing multiple tasks in highly redundant robotic systems. *Proceedings of International Conference on Advanced Robotics*, Pisa, Italy, 1991, 1211–1216.

[11] Cheng F T, Lee H L, Orin D E. Increasing the locomotive stability margin of multilegged vehicles. *Proceedings of IEEE International Conference on Robotics and Automation* (*ICRA*), Detroit, USA, 1999, 1708–1714.

[12] Pongas D, Mistry M, Schaal S. A robust quadruped walking gait for traversing rough terrain. *Proceedings of IEEE International Conference on Robotics and Automation* (*ICRA*), Roma, Italy, 2007, 1474–1479.

[13] Zhang S S, Rong X W, Li Y B, Li B. A composite cog trajectory planning method for the quadruped robot walking on rough terrain. *International Journal of Control and Automation*, 2015, **8**, 101–118.

[14] ROBOTIS Dynamixel, [2018-03-13], http://www.robotis.com/.

[15] SMOOTH-ON Dragon skin, [2018-03-13], https://www.smooth-on.com/product-line/dragon-skin/.

[16] He J, Gao F. Type synthesis for bionic quadruped walking robots. *Journal of Bionic Engineering*, 2015, **12**, 527–538.

[17] Gonzalez-Rodriguez A G, Gonzalez-Rodriguez A, Castillo-Garcia F. Improving the energy efficiency and speed of walking robots. *Mechatronics*, 2014, **24**, 476–488.

[18] Whitney D E. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-machine Systems*, 1969, **10**, 47–53.

[19] Siciliano B, Sciavicco L, Villani L, Oriolo G. *Robotics*: *Modeling*, *Planning and Control*, Springer, London, UK, 2009.

[20] Zghal H, Dubey R V, Euler J A. Efficient gradient projection optimization for manipulators with multiple degrees of redundancy. *Proceedings of IEEE International Conference on Robotics and Automation* (*ICRA*), Cincinnati, USA, 1990, 1006–1011.

[21] MathWorks MATLAB, [2018-03-01], https://www.mathworks.com/.

[22] McGhee R B, Frank A A. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 1968, **3**, 331–351.

[23] Coppelia Robotics V-REP, [2018-03-01], http://www.coppeliarobotics.com/.

[24] Song S M, Waldron K J. An analytical approach for gait study and its applications on wave gaits. *The International Journal of Robotics Research*, 1987, **6**, 60–71.

[25] Hirose S, Tsukagoshi H, Yoneda K. Normalized energy stability margin and its contour of walking vehicles on rough terrain. *Proceedings of IEEE International Conference on Robotics and Automation* (*ICRA*), Seoul, Korea, 2001, 181–186.

[26] Vukobratović M, Borovac B. Zero-moment point – thirty five years of its life. *International Journal of Humanoid Robotics*, 2004, **1**, 157–173.

[27] Messuri D A, Klein C A. Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. *IEEE Journal on Robotics and Automation*, 1985, **1**, 132–141.