14

A Fast and Smooth Walking Pattern Generator for Biped Robots

Han-Pang Huang and Jiu-Lou Yan
National Taiwan University
Taiwan

1. Introduction

In order to solve inverse kinematics of a multi-DOF (degree of freedom) mechanism, many methods have been proposed with the Jacobian linearization method. When solving inverse kinematics problems of the biped robot with this method, long computation time is required since the Jacobian matrix should be updated in order to solve the configuration for each different end-effector trajectory knot. In this chapter, two smooth trajectories are generated as target positions, one for swing leg's ankle, and the other for center of gravity (COG). These generated knot points in the task space with appropriate distance to each other are used to solve inverse kinematics by the proposed modified Jacobian method—Fixed Leg Jacobian. It can guarantee that only one iteration is required to solve the configuration when it is away from singularity with a small position error (0.0712% of leg length). The proposed algorithm can generate the gait in real-time including singularity avoidance and joint limit avoidance. Simulations have been carried out. The results showed that the proposed method can generate a smooth gait for robot walking on real-time implementation.

Compared with wheeled robots, legged robots have the advantage of being able to traverse uneven or sharp-height-changing environments. Nowadays, many vehicles, buildings and environments are designed for humans. Simple robots cannot enter and adapt to these places. Therefore, we must design complicated humanoid robots to do it. But when the designs become more complicated and with more DOFs, it is getting harder to control and generate the trajectories of them. The proposed algorithm can quickly generate smooth trajectories of the ankle and COG and solve inverse kinematics in order to achieve real-time control of biped robots. In this chapter, the focus is how to coordinate the swing leg, the fixed leg and the COG of the robot, and generate the gait in real-time. In the simulation, the robot has 24 DOFs, 6 in each leg (12 in two legs), 4 in each arm (8 in two arms), 2 in the torso and 2 in the head. The most important DOFs for balancing and walking are the twelve DOFs in the legs. DOFs in the fixed leg dominate the position of the COG, and the position of the ankle of the swing leg is given relative to the position of the ankle of the fixed leg in order to guarantee that the swing leg is in a proper position that it will not hit the fixed leg and touch the ground. The trajectories of the end-effectors planned with desired constraints are inputted to solve inverse kinematics, as shown in Fig. 1.

Many researchers have proposed the solutions to the problem while solving Jacobian linearized inverse kinematics. They include the damped least square method (DLS) (Wampler, 1986) and the robust damped least square method (RDLS) (Nakamura &

Fig. 1. Solving robot walking gait using inverse kinematics

Hanafusa, 1986), which are used for singularity avoidance. The weighted least-norm method (WLN) (Chan & Dubey, 1995) is used for joint limit avoidance. These two methods (RDLS and WLN) are used in this chapter to improve the performance of the algorithm. Many other methods are proposed, such as the selectively damped least squares methods (SDLS) (Buss & Kim, 2004), the gradient projection method (GPM) (Liegeois, 1997) and the extended Jacobian method EJM (Klein et al.,1995; Tevatia & Schaal, 2000).

On the other hand, the target positions that are one-by-one inputted to solve the inverse kinematics should also be planned appropriately. If the target positions are not planned close enough, many iterations will be wasted since the end-effectors are not in the desired positions and directions. Non-smooth target positions make the robot's joints oscillate. Regarding the generation of the trajectories, many methods were proposed, such as Lagrange interpolations, cubic spline, conventional tension spline (CTS) and modified tension spline (MTS) (Huang & Liu, 2005). Each method has its limitation. In this chapter, MTS is applied in order to generate smooth trajectories in position, velocity, acceleration, and jerk.

2. Robot kinematics

2.1 Inverse kinematics with pseudo inverse

The pseudo inverse method and two improving methods, RDLS, WLN, are used in this chapter to construct the inverse kinematics solver. By using forward kinematics method (DH method), the relationship between the position of the end-effectors and the joint angles can be found as Equation (1).

$$x = f(\theta) \tag{1}$$

The Jacobian linearized relationship between the velocities of the end-effectors and the joint angles are defined as Equation (2).

$$\dot{x} = J\dot{\theta} \tag{2}$$

where J denotes the Jacobian matrix; if there are redundant DOFs in the system, J is a rectangle matrix. Pseudo inverse method can be used to solve $\dot{\theta}$ with given $\dot{x}$:

$$\dot{\theta} = J^{+}\dot{x} \tag{3}$$

2.2 Robust damped least squares methods (RDLS)

If the determinant of JJ^T is zero or close to zero, singularity occurs. In order to avoid the singularity, robust damped least square method (RDLS) is applied. The idea of the damped least square method (DLS) is to minimize $||\dot{x} - J\dot{\theta}||^2 + \alpha||\dot{\theta}||^2$, the sum of the square of the residual error and the joint velocities. Here α is a positive damping factor. Thus, the pseudo inverse with DLS method is shown as Equation (4).

$$J^{+} = J^{T}(JJ^{T} + \alpha I_m)^{-1} \tag{4}$$

where I_m is an identity matrix with the same dimension as JJ^T matrix. The damping factor α helps to avoid singularity, but it also affects the solved $\dot{\theta}$. Thus, α should not be applied at nonsingular configurations. Nakamura et al. proposed a robust DLS method to solve this problem. A factor h is defined as Equation (5).

$$h(\boldsymbol{\theta}) = \sqrt{\det(JJ^{T})} \tag{5}$$

When h approaches to zero, it is getting closer to singularity. Then α is adjusted automatically with Equation (6).

$$\alpha = \begin{cases} \alpha_0(1 - h/h^s), & if\ h < h^s \\ 0, & otherwise \end{cases} \tag{6}$$

where h^s denotes the threshold value, α_0 is the value of damping factor at singular points. With the equation above, α is effective only when the configuration is near singular configuration.

2.3 Weighted least-norm method

The weighted least-norm method is designed from the idea of null space. The general solution of $\dot{\theta}$ for solving inverse kinematics can be written as Equation (7).

$$\dot{\boldsymbol{\theta}} = J^{+}\dot{\mathbf{x}} + (I - J^{+}J)\boldsymbol{\varphi} \tag{7}$$

where φ is an arbitrary vector. $J^{+}\dot{x}$ is the particular solution, and $(I\text{-}J^{+}J)\varphi$ is the homogeneous solution. Joint limit avoidance is important for humanoid robots in order to act like human beings. A weighted least-norm (WLN) solution based scheme for avoiding joint limits is proposed by Chan & Dubey. In this method, a performance criterion $H(\theta)$ is defined as Equation (8).

$$H(\boldsymbol{\theta}) = \sum_{i=1}^{n} \frac{1}{4} \frac{(\theta_{i,\max} - \theta_{i,\min})^2}{(\theta_{i,\max} - \theta_i)(\theta_i - \theta_{i,\min})} \tag{8}$$

When any joint approaches its limit, the value of $H(\theta)$ grows very fast, and so is its partial differentiation $\partial H(\theta)/\partial\theta_i$. Thus, the weighting matrix is defined as Equations (9) and (10).

$$W = \begin{bmatrix} w_1 & 0 & \cdots & \cdots & 0 \\ 0 & w_2 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & w_n \end{bmatrix} \tag{9}$$

$$w_i = 1 + \left| \frac{\partial H(\boldsymbol{\theta})}{\partial \theta_i} \right| \tag{10}$$

The WLN method can be expressed as Equations (11) and (12).

$$J_W = JW^{-1/2} \tag{11}$$

$$\dot{\boldsymbol{\theta}} = W^{-1/2} J_W{}^{+} \dot{\mathbf{x}} + (W^{-1/2} J_W{}^{+} - J^{+}) \dot{\mathbf{x}} \tag{12}$$

2.4 RWLN method

The RWLN method (Yu, 2006) is the combination of the RDLS method and the WLN method. The equation used to solve inverse kinematics with the RWLN method is shown as Equations (13) and (14).

$$\dot{\boldsymbol{\theta}} = W^{-1/2} J_{W,\alpha}{}^{+} \dot{\mathbf{x}} + (W^{-1/2} J_{W,\alpha}{}^{+} - J_{\alpha}{}^{+}) \dot{\mathbf{x}} \tag{13}$$

$$J_{W,\alpha}{}^{+} = J_W^T (J_W J_W^T + \alpha_W I_m)^{-1} \tag{14}$$

It can avoid the singularity of J^{+}, and can also avoid the singularity of J_W^{+} with auto-adjusting α and α_W.

3. Gait generation algorithm

The solving process of inverse kinematics is described in the following.

1. The D-H forward kinematics of each limb and head is constructed independently. All limbs have the same base point which is at the middle point of the two hips.
2. Determine the trajectory of the end-effector of the swing leg using MTS method and the trajectory of the COG using preview control method (Kajita et al., 2006). The relationship between the swing leg and fixed leg is discussed in the following discussion "Relative Input".
3. Construct the conventional Jacobian matrix and the COG Jacobian matrix.
4. Construct the proposed F-Jacobian matrix and then combine it into the conventional Jacobian matrix.
5. Solve inverse kinematics using the proposed F-Jacobian method with the inputs, the trajectories of the swing leg and COG.

3.1 Relative input

If the trajectories of the end-effectors of the swing leg and the fixed leg are assigned independently, the swing leg may touch the fixed leg or even be lower than the fixed leg to break the balance of the robot if there is no any other good checking method to check it. So the position of the ankle of the swing leg should be dependent upon the position of the swing leg, as shown in Fig. 2 and Equation (15).

$$\vec{r}_{swing} = \vec{r}_{fixed} + R_n \vec{r}_{bl} + \vec{r}_{traj,i} \tag{15}$$

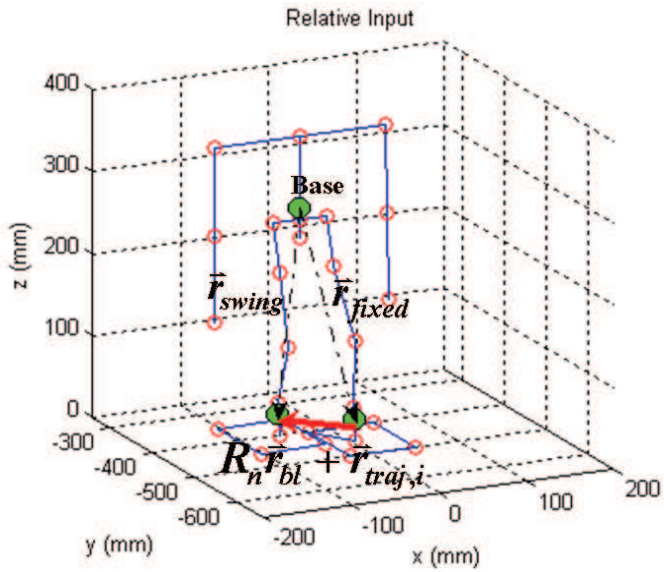

Fig. 2. Relative input trajectory of the swing leg

where $\vec{r}_{swing}$ denotes the position vector from "Base" to the ankle of the swing leg, $\vec{r}_{fixed}$ denotes the position vector from "Base" to the ankle of the fixed leg, $\vec{r}_{bl}$ denotes the position vector from the ankle of the fixed leg to the ankle of the swing leg, R_n denotes the rotation matrix that expresses the rotation between the base coordinate frame and the ground. The magnitude of $\vec{r}_{bl}$ is a constant. And the $\vec{r}_{traj,i}$ denotes the *i*-th planned trajectory point of the ankle of the swing leg. It can control how the swing leg moves in each stride. The R_n is not an identity matrix when the orientation of the robot is not the same as the world coordinates. In this chapter, in order to simplify the system, the orientation of the robot is the same as the world coordinates and the R_n is an identity matrix. With dependent position input of the ankle of the swing leg, we can give a trajectory easily that the swing leg will not lower than the fixed leg to avoid breaking the balance of the robot and touching the fixed leg.

3.2 Modified Tension Splines (MTS)

It is also important that we should input smooth and well-defined trajectories to our system. MTS method is based on conventional tension splines (CTS), and the tension factor can be

arbitrarily assigned in this method. MTS method also generates trajectories with smooth position, velocity, acceleration and jerk. Trajectory planned by MTS is defined as Equation (16).

$$q_{j,i}(t) = \frac{1}{\sigma_{j,i}^2}\frac{\sinh\left[\sigma_{j,i}\left(t_{i+1}-t\right)\right]}{\sinh\left(\sigma_{j,i}h_i\right)}\ddot{q}_{j,i} + \frac{1}{\sigma_{j,i}^2}\frac{\sinh\left[\sigma_{j,i}\left(t-t_i\right)\right]}{\sinh\left(\sigma_{j,i}h_i\right)}\ddot{q}_{j,i+1} + \left(q_{j,i} - \frac{1}{\sigma_{j,i}^2}\ddot{q}_{j,i}\right)\frac{t_{i+1}-t}{h_i} + \left(q_{j,i+1} - \frac{1}{\sigma_{j,i}^2}\ddot{q}_{j,i+1}\right)\frac{t-t_i}{h_i} \tag{16}$$

where $h_i = t_{i+1}-t_i$, the time interval of the i-th CTS segment, α_{ij} denotes the tension factor of the ith CTS segment. The acceleration $\ddot{q}_{ij}$ can be found as Equation (17). After solving the $\ddot{q}_{ij}$, the knots q can be found by Equation (16).

$$\ddot{q}_{j,i}(t) - \sigma_{j,i}^2 q_{j,i}(t) = \frac{t_{i+1}-t}{h_i}\left(\ddot{q}_{j,i} - \sigma_{j,i}^2 q_{j,i}\right) + \frac{t-t_i}{h_i}\left(\ddot{q}_{j,i+1} - \sigma_{j,i}^2 q_{j,i+1}\right), \quad t_i \le t \le t_{i+1} \tag{17}$$

3.3 Fixed Leg Jacobian (F-Jacobian)

After constructing the DH parameters, the Jacobian matrix can be found by the cross product method, and then the limbs and the head can be controlled independently with the Jacobian matrix. The ankles, the fingertips and the head are chosen as end-effectors. But if we solve inverse kinematics of the limbs and the head of the robot independently, it is very difficult to decide where the positions of the end-effectors should be because DH forward kinematics method constructs the joint positions and orientations in its own coordinates instead of the world coordinates, and all positions of the end-effectors in the world coordinates are influenced by the movements of the fixed leg. Equation (18) describes the conventional Jacobian matrix that is used while solving inverse kinematics independently.

$$\begin{bmatrix} \dot{d}_{LL} \\ \dot{d}_{RL} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \end{bmatrix} = \begin{bmatrix} J_{LL} & 0 & 0 & 0 & 0 \\ 0 & J_{RL} & 0 & 0 & 0 \\ 0 & 0 & J_{LA} & 0 & 0 \\ 0 & 0 & 0 & J_{RA} & 0 \\ 0 & 0 & 0 & 0 & J_{H} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{LL} \\ \dot{\theta}_{RL} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \tag{18}$$

where $\dot{d}$ denotes the vector from the current end-effector position to the desired end-effector position in its own coordinates; LL, RL, LA, RA, and H denotes left leg, right leg, left arm, right arm, and head, respectively. If we just want to control the limbs and the head of the robot independently, it is enough to solve inverse kinematics with the equation above in one iteration if $\dot{d}$ is given appropriately (not too large).

In fact, it is not enough to control the end-effectors independently. Relative input should be used to prevent the swing leg from hitting the fixed leg or touching the ground. The velocity of the ankle of the fixed leg in the world coordinates is zero because it is fixed on the

ground. The velocities of all other end-effectors in the world coordinates can be found by subtracting the velocity of the fixed leg from them. In world coordinates, the velocities of the end-effectors are shown in Equation (19) – (21).

$$V_{fixed,world} = 0, \quad \omega_{fixed,world} = 0 \tag{19}$$

$$V_{swing,world} = V_{swing,local} - V_{fixed,local}, \quad \omega_{swing,world} = \omega_{swing,local} - \omega_{fixed,local} \tag{20}$$

$$V_{other,world} = V_{other,local} - V_{fixed,local}, \quad \omega_{other,world} = \omega_{other,local} - \omega_{fixed,local} \tag{21}$$

where V_{fixed} and ω_{fixed} denote the velocity and the angular velocity of the fixed leg; the subscript "*world*" denotes the variable is in the world coordinates, and the subscript "*local*" denotes the variable is in its own body-fixed coordinates.
Since the "Base" of the legs and arms are the same, the Jacobian matrix expressing the influence of the fixed leg to all other end-effectors can be calculated as follows. It is calculated using the cross product method, as shown in Equation (22) – (24).

$$J_F = \begin{bmatrix} J_{Translational} \\ J_{Rotational} \end{bmatrix} \tag{22}$$

It is called "F-Jacobian" (Fixed Leg Jacobian) method. Note that $J_{Rotational}$ is given as Equation (23).

$$J_{Rotational} = -\begin{bmatrix} \omega_{e,1} & \omega_{e,2} & \dots & \omega_{e,j} & \dots & \omega_{e,n} \end{bmatrix} \tag{23}$$

where n denotes the total number of joints of the fixed leg, $\omega_{e,j}$ are unit normal vectors of the joints of the fixed leg. The minus sign is multiplied since when a joint in the fixed leg rotates clockwise in its coordinates, the body rotates counterclockwise in the world coordinates. On the other hand, the $J_{Translational}$ is calculated as Equation (24).

$$J_{Translational}{}^{T} = -\begin{bmatrix} \omega_{e,1} \times \vec{r}_{1,F \to end\_eff} \\ \omega_{e,2} \times \vec{r}_{2,F \to end\_eff} \\ \dots \\ \omega_{e,j} \times \vec{r}_{j,F \to end\_eff} \\ \dots \\ \omega_{e,n} \times \vec{r}_{n,F \to end\_eff} \end{bmatrix} \tag{24}$$

where $\vec{r}_{j,F \to end\_eff}$ denotes the vector from the j-th joint of the fixed leg to the end-effector affected by the movement of the motion of the fixed leg.

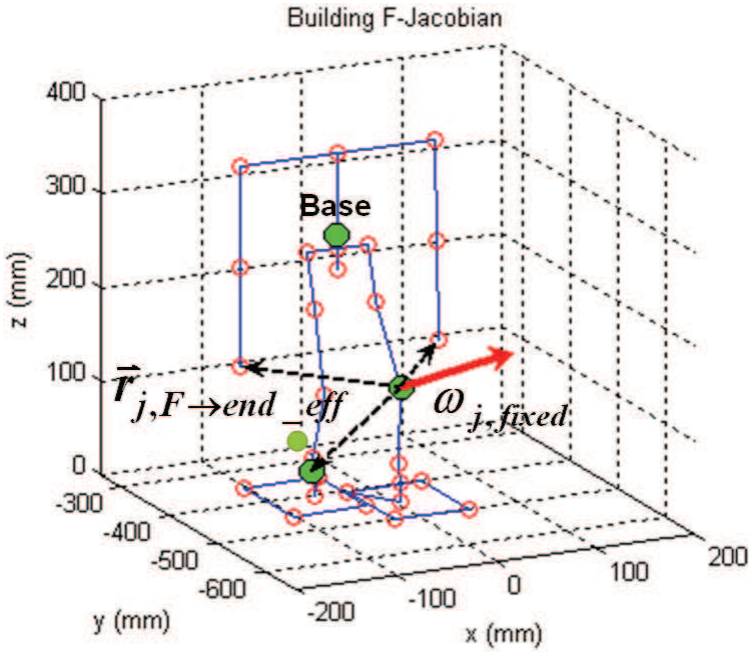

Fig. 3. Build the F-Jacobian

The F-Jacobian matrix is written as Equation (25).

$$\begin{bmatrix} \dot{d}_{fixed} \\ \dot{d}_{swing} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \end{bmatrix} = \begin{bmatrix} J_{fixed} & 0 & 0 & 0 & 0 \\ J_{f\to s} & J_{swing} & 0 & 0 & 0 \\ J_{f\to LA} & 0 & J_{LA} & 0 & 0 \\ J_{f\to RA} & 0 & 0 & J_{RA} & 0 \\ J_{f\to H} & 0 & 0 & 0 & J_{H} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{fixed} \\ \dot{\theta}_{swing} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \tag{25}$$

where $J_{f\to X}$ denotes the F-Jacobian matrices; the subscript "X" denotes the end-effector affected by the movement of the joints of the fixed leg, such as the swing leg and the fingertips.

Recall the Equation (15).

$$\vec{r}_{swing} = \vec{r}_{fixed} + R_n \vec{r}_{bl} + \vec{r}_{traj,i} \tag{15}$$

The 3-by-1 column vector $\dot{d}_{swing}$ is given by:

$$\dot{d}_{swing} = \vec{r}_{swing,next} - \vec{r}_{swing,now} \tag{26}$$

To achieve the desired position of the fixed leg, the $\vec{r}_{fixed}$ term converges to the target position after iterations of inverse kinematics algorithm. The $\dot{d}_{swing}$ term also changes with

the $\bar{r}_{fixed}$ term. Here the F-Jacobian method can be used to describe and compensate the effect of the motion of the fixed leg in the Jacobian matrix in order to reduce the total iterations while solving inverse kinematics. Equation (27) can be obtained by multiplying the second row and the $\dot{\theta}$ column of the matrices in Equation (25) as

$$\dot{d}_{swing} = J_{f\to s}\dot{\theta}_{fixed} + J_{swing}\dot{\theta}_{swing} \tag{27}$$

The $J_{f\to s}\dot{\theta}_{fixed}$ term means the effect of the fixed leg in the world coordinates, and serves as the compensation term. The $J_{swing}\dot{\theta}_{swing}$ term means the affection of the joints of the swing leg itself. Without the compensation of F-Jacobian term, the end-effectors will oscillate and then converge to the desired position slower.

3.4 COG Jacobian (Center of Gravity Jacobian)

The position, velocity and acceleration of COG are highly related with whether the robot falls or not. The position of COG can be computed by averaging the sum of the product of the linkage masses and their position vectors, as shown in Equation (28). Note that the $\bar{r}_{COG}$ is a 3-by-1 vector described in Cartesian coordinates.

$$\bar{r}_{COG} = \frac{1}{n}\sum_{i=1}^{n} m_i \cdot \bar{r}_{m,i} \Big/ \sum_{i=1}^{n} m_i \tag{28}$$

where "Base" denotes the start point of forward kinematics, $\bar{r}_{m,i}$ denotes the vector from base to the COG of linkage i , and m_i denotes the mass of linkage i.

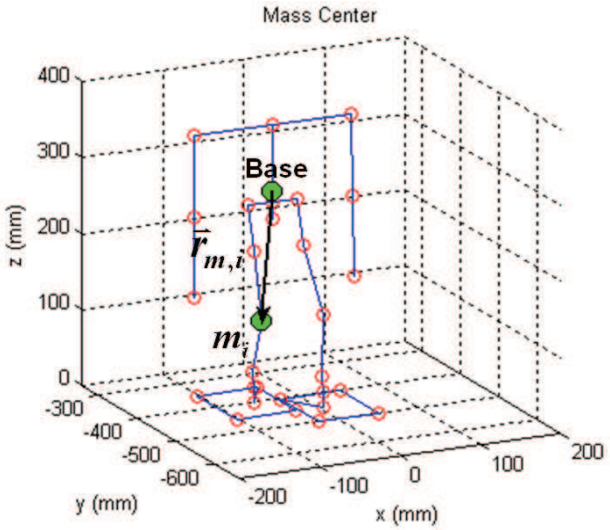

Fig. 4. Position of the COG of a linkage

When a joint rotates, only parts of the whole body are rotated, while the others are not rotated. Separating the rotated parts and the fixed parts (without rotating), we obtain

$$M \cdot \vec{r}_{COG} = \left(\sum_h m_{a,h} \cdot \vec{r}_{a,h} + \sum_k m_{ua,k} \cdot \vec{r}_{ua,k} \right) \Big|_{\text{joint}=j} \tag{29}$$

where M denotes the total mass of the robot, subscript "a" denotes the parts that are affected by the rotation, subscript "ua" denotes the parts that are unaffected by the rotation, the $m_{a,h}$ and $m_{ua,k}$ denote the mass that are affected and unaffected by the joint j, and the vectors $\vec{r}_{a,h}$ and $\vec{r}_{ua,k}$ denote the position of the COG of each affected part and each unaffected part. The equation can also be written as

$$M \cdot \vec{r}_{COG} = \left(M_a \cdot \vec{r}_a + M_{ua} \cdot \vec{r}_{ua} \right) \Big|_{\text{joint}=j} = M_{a,j} \cdot \vec{r}_{a,j} + M_{ua,j} \cdot \vec{r}_{ua,j} \tag{30}$$

where $M_{a,j}$ denotes the total mass of the parts affected by joint j, and $M_{ua,j}$ denotes the total mass of the parts unaffected by joint j, $\vec{r}_{a,j}$ and $\vec{r}_{ua,j}$ denote the position vector of the COG of the affected and unaffected parts. Note that the members of affected and unaffected parts change with different joint j, and they also depend upon each different control system. The position change of $\vec{r}_{COG}$ caused by the rotation of joint j can be approximated as

$$\Delta\vec{r}_{COG,j} = \frac{M_{a,j}}{M} \cdot \Delta\vec{r}_{a,j} + \frac{M_{ua,j}}{M} \cdot \Delta\vec{r}_{ua,j} \tag{31}$$

Since $\vec{r}_{ua}$ is unaffected by the rotation, $\Delta\vec{r}_{ua}$ is always equal to zero. $\Delta\vec{r}_{COG,j}$ denotes the displacement of the whole robot's COG caused by the rotation of the j-th joint. Thus, the COG Jacobian can be obtained as

$$\Delta\vec{r}_{COG,j} = \frac{M_{a,j}}{M} \Delta\vec{r}_{a,j} = J_{COG,j} \cdot \dot{\theta}_j \tag{32}$$

where the $J_{COG,j}$ denotes the COG Jacobian of joint j, $\dot{\theta}_j$ denotes the angular speed of joint j. The COG Jacobian matrix of the joints on the limbs except the joints on the fixed leg can be found as Fig. 5 and Equation (33).

$$J_{COG,j} = \frac{M_a}{M} \vec{\omega}_{e,j} \times \vec{r}_{c \to j} \tag{33}$$

where $\vec{\omega}_{a,j}$ denotes the unit vector along the z-direction of the j-th axis, $\vec{r}_{c \to j}$ denotes the vector from the j-th joint to the COG of the affected parts. We can also use this concept to get the COG Jacobian of the fixed leg, as shown in Fig.6 and Equation (34).

$$\Delta\vec{r}_{COG,j} = J_{COG,j} \cdot \dot{\theta}_j = -\frac{M_a}{M} \vec{\omega}_{e,j} \times \vec{r}_{c \to j} \cdot \dot{\theta}_j \tag{34}$$

In the coordinates of the fixed leg, the rotation of a joint rotates the parts that are lower than it. But in the world coordinates, the rotation causes the parts higher than the joint to rotate with the same angular velocity in negative direction.

$$J_{COG,j} = -\frac{M_a}{M} \vec{\omega}_{a,j} \times \vec{r}_{c \to j} \tag{35}$$

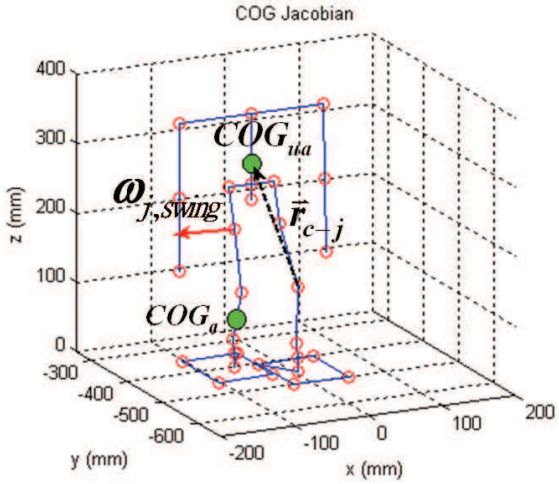

Fig. 5. Construction of COG Jacobian—swing leg

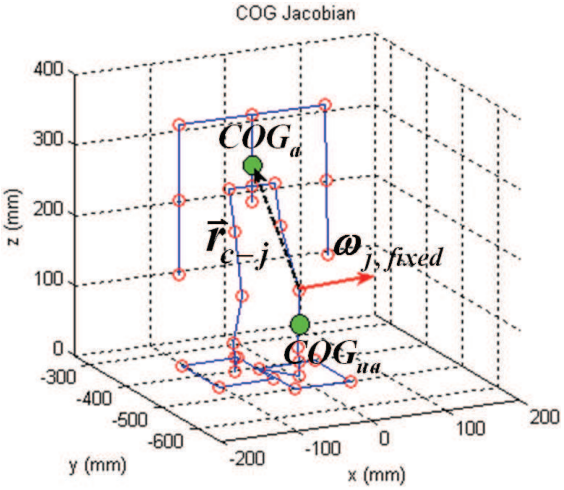

Fig. 6. Construction of COG Jacobian—fixed leg

The Jacobian matrix with COG Jacobian can be written as

$$\begin{bmatrix} \dot{d}_{fixed} \\ \dot{d}_{swing} \\ \dot{d}_{LA} \\ \dot{d}_{RA} \\ \dot{d}_{H} \\ \dot{d}_{COG} \end{bmatrix} = \begin{bmatrix} J_{fixed} & 0 & 0 & 0 & 0 \\ J_{f\to s} & J_{swing} & 0 & 0 & 0 \\ J_{f\to LA} & 0 & J_{LA} & 0 & 0 \\ J_{f\to RA} & 0 & 0 & J_{RA} & 0 \\ J_{f\to H} & 0 & 0 & 0 & J_{H} \\ J_{f\to C} & J_{s\to C} & J_{LA\to C} & J_{RA\to C} & J_{H\to C} \end{bmatrix} \begin{bmatrix} \dot{\theta}_{fixed} \\ \dot{\theta}_{swing} \\ \dot{\theta}_{LA} \\ \dot{\theta}_{RA} \\ \dot{\theta}_{H} \end{bmatrix} \tag{36}$$

where $\dot{d}_{fixed}$ is a 3-by-1 matrix, it controls the three orientations of the fixed leg. The other three DOFs of the fixed leg are used to control the position of the COG, $\dot{d}_{COG}$ is a 3-by-1 matrix that controls the position of COG, $J_{X\to C}$ denotes the COG Jacobian, the "X" denotes the limbs that their movements affect the position of the COM.

4. Simulation

All the input trajectories of the swing leg in the simulations below are generated with "Relative input" method, and the position of the swing leg is given relative to the fixed leg in order to guarantee the swing leg is in a proper position, as shown in Fig. 7. The COG trajectory is generated smoothly and fits the COG/ZMP (zero moment point) inverted pendulum constraint with the preview control method, as shown in Fig. 8.

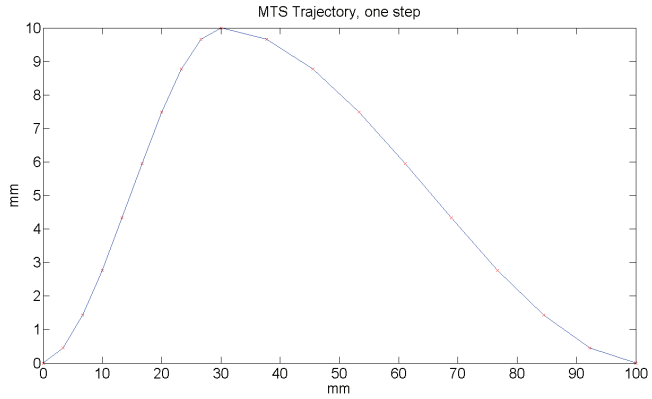

Fig. 7. Planned trajectory of the ankle of the swing leg

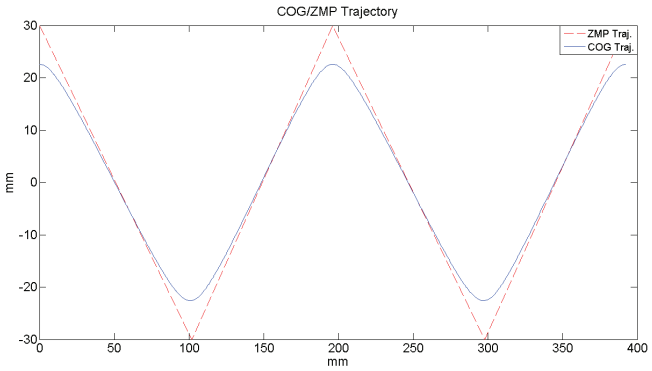

Fig. 8. Planned trajectory of the COG

The inputs above are inputted to the robot walking system in the simulation. The computation times of the same robot model with conventional Jacobian matrix and proposed F-Jacobian matrix will be compared. Each gait configuration should be solved as soon as possible in order to gain more CPU resources in each time period, and then each gait

configuration will be sent to the robot within the desired time interval to control the robot walking speed. Human walks about 90 to 110 steps in a minute. It means 0.55 to 0.67 second per step. The robot should walk faster than at least 1.0sec/step to simulate human walking. The simulation results are shown below. All the simulations below are done with a personal computer equipped with Intel Core2Duo E6300 1.83GHz and 2GB RAM.

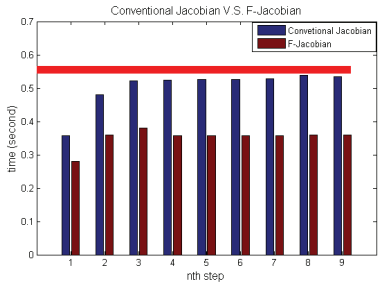

Fig. 9. Simulation results of computation time

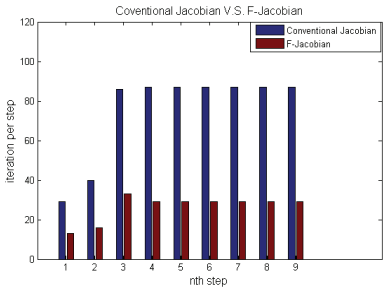

Fig. 10. Simulation results of iterations

The graphics above show the computation time and the iterations per step of the pseudo inverse method with and without F-Jacobian method. Step 1 and step 2 are initial steps, so they are with different computation time and iterations per step. The red bar is the 0.55sec/step reference line of human walking speed. Inverse kinematics with F-Jacobian solves the planned 9 steps in 3.07 second and 232 iterations (average 0.341 second and 25.8 iterations), and inverse kinematics without F-Jacobian solves the planned 9 steps in 4.62 second and 677 iterations (average 0.513 second and 75.2 iterations). Clearly, the proposed method, F-Jacobian, saved 33.55% computation time and 65.73% iterations per step. Except initial steps, each step contains 27 configurations. In the 27 configurations, the first three and last three points need no iteration because they are at the same position. The proposed method, F-Jacobian, can solve each configuration in only one computation when the acceptable error is 0.2mm (0.0712% of the length of legs). "Acceptable error" means the acceptable position error value when solving inverse kinematics. If the position error is smaller than the acceptable error, the next trajectory knot will be inputted to the inverse kinematics solver. If the position error is still larger than the acceptable error, the same trajectory knot will be inputted to the solver again. Since the input to the inverse kinematics are close and smooth enough, the proposed method can get smooth trajectories and solve each configuration in one iteration. But for the same input, inverse kinematics without F-

Jacobian makes the joints oscillate and needs about 3 iterations to solve one configuration. Fig. 11 and Fig. 12 show the solved trajectories of the left ankle (end-effector) with and without F-Jacobian.

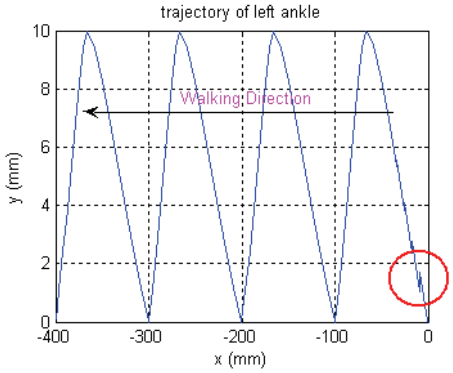

Fig. 11. Solved trajectory with F-Jacobian

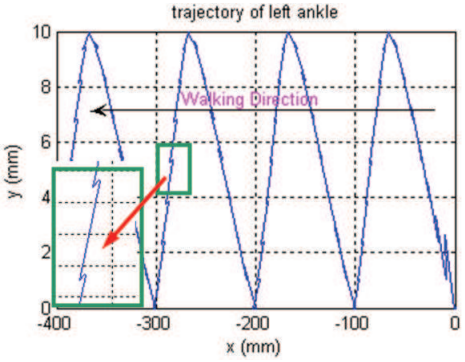

Fig. 12. Solved trajectory without F-Jacobian

Without F-Jacobian, the solved trajectories are not all useable. Only points that are in the acceptable error range are useable. We can reduce the number of useless points with F-Jacobian method. The robot walks from 0 to -400mm in the simulation. Only some configurations in the initial steps are not solved in one computation since the home configuration of the robot is near singularity. After initial steps, the robot has bent its knees, and hence keeps the robot away from the singular configurations. All the configurations after initial steps are solved in one computation. Fig.13 and Fig.14 show the position error after each inverse kinematics computation (conventional Jacobian vs. the proposed F- Jacobian).
In the figures, the max error of conventional Jacobian method is about fifteen times larger than F-Jacobian method. Fig.15 shows the acceptable error versus the total iterations of the inverse kinematics computation (for 232 trajectory knots) with and without F-Jacobian method.
From the figure, we can see the number of total iterations for conventional Jacobian method grows much faster than the F-Jacobian method when we choose the acceptable error from 2mm to 0.0002mm.

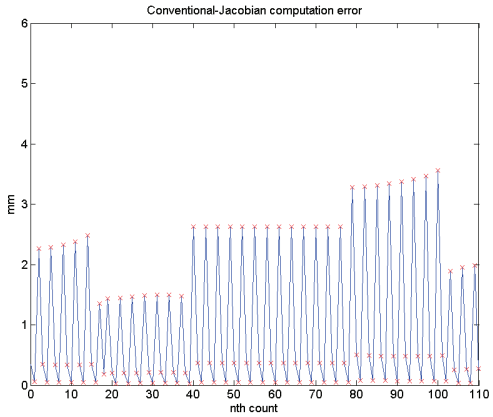

Fig. 13. Position error--conventional Jacobian

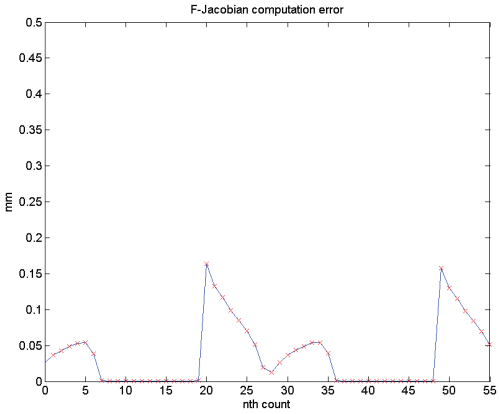

Fig. 14. Position error F-Jacobian

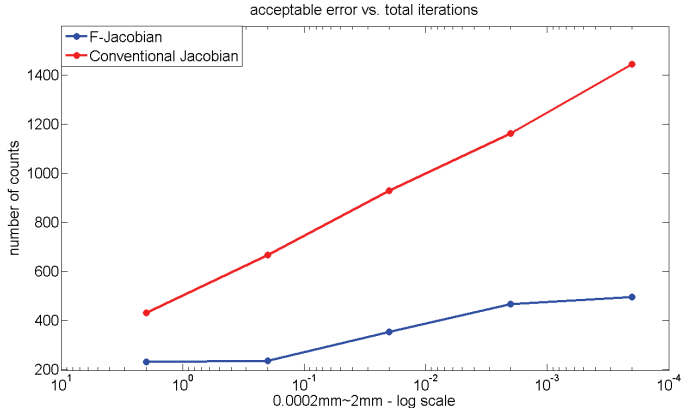

Fig. 15. Acceptable error vs. total iterations

5. Conclusions

Cooperation of the swing leg and the fixed leg is important for a humanoid robot. In this chapter, the position of the swing leg is dependent upon the position of the fixed leg. In order to solve inverse kinematics faster, F-Jacobian method is applied. The F-Jacobian method can compensate the displacements of the other end-effectors that are caused by the rotation of each joint in the fixed leg. F-Jacobian method can save computation time and generate knots more smoothly and effectively than the conventional Jacobian method. F-Jacobian method can be extended to construct COG Jacobian and other Jacobian matrices in order to describe and compensate the effect of the fixed leg. Simulations are done and justify that the F-Jacobian has better performance than conventional Jacobian method.

6. References

Buss, S. R. & Kim, J. S. (2004). Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, Vol. 10, pp. 37-49

Chan, T. F. & Dubey, R. V. (1995). A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators. *IEEE Trans. On Robotics and Automation*, Vol. 11, No. 2, pp. 286-292

Huang, H. P. & Liu, C. P. (2005). A Novel Trajectory Optimization and Workspace Boundary Singularity Solution for Industrial Robots. *Proceedings of Automation the Eighth International Conference on Automation Technology Conference*, pp. 1-6

Klein, C. A., Caroline, C. J. & Ahmed, S. (1995). A New Formulation of the Extended Jacobian Method and its Use in Mapping Algorithmic Singularities for Kinematically Redundant Manipulators. *IEEE Trans. on Robotics and Automation*, Vol. 11, No. 1, pp. 50-55

Kajita S., Morisawa M., Harada K., Kaneko K., Kanehiro F., Fujiwara K. & Hirukawa H. (2006). Biped Walking Pattern Generator allowing Auxiliary ZMP Control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2993-2999

Liegeois, A. (1997). Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms. *IEEE Trans. systems, Man, and Cybernetics*, Vol. 7, No. 12

Nakamura, Y. & Hanafusa H. (1986). Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control. *ASME Journal of Dynamic Systems, Measurement and Control*, Vol. 108, pp. 163-171

Tevatia, G. & Schaal S. (2000). Inverse Kinematics for Humanoid Robots. *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 294-299

Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, No. 1, pp 93-101

Yu, S. W. (2006). Walking Pattern Analysis and Control of a Humanoid Robot. Master Thesis, Department of Mechanical Engineering, National Taiwan University, pp. 28-30

Biped Robots

Edited by Prof. Armando Carlos Pina Filho

ISBN 978-953-307-216-6
Hard cover, 322 pages
Publisher InTech
Published online 04, February, 2011
Published in print edition February, 2011

Biped robots represent a very interesting research subject, with several particularities and scope topics, such as: mechanical design, gait simulation, patterns generation, kinematics, dynamics, equilibrium, stability, kinds of control, adaptability, biomechanics, cybernetics, and rehabilitation technologies. We have diverse problems related to these topics, making the study of biped robots a very complex subject, and many times the results of researches are not totally satisfactory. However, with scientific and technological advances, based on theoretical and experimental works, many researchers have collaborated in the evolution of the biped robots design, looking for to develop autonomous systems, as well as to help in rehabilitation technologies of human beings. Thus, this book intends to present some works related to the study of biped robots, developed by researchers worldwide.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Han-Pang Huang and Jiu-Lou Yan (2011). A Fast and Smooth Walking Pattern Generator for Biped Robots, Biped Robots, Prof. Armando Carlos Pina Filho (Ed.), ISBN: 978-953-307-216-6, InTech, Available from: http://www.intechopen.com/books/biped-robots/a-fast-and-smooth-walking-pattern-generator-for-biped-robots

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821