Chapter 4
Biped Walking

What is the meaning of the word "walk"? Oxford Advanced Learner's Dictionary gives us a concise definition.

> **walk:**
> move along at a moderate pace by lifting up and putting down each foot in turn, so that one foot is on the ground while the other is being lifted
> (Oxford Advanced Learner's Dictionary, Oxford University Press)

Therefore, at least one foot must be in contact with the ground at any moment during walking. There exist two kind of walking, namely, static walking and dynamic walking. In "static walking", the projection of the center of mass never leaves the support polygon during the walking. In "dynamic walking", there exist periods when the projection of the center of mass leaves the support polygon.

$$\text{Walk} \begin{cases} \text{static walk} \\ \text{dynamic walk} \end{cases}$$

Most toy robots perform static walking using large feet. This is not interesting from the view point of control engineering since it is fairly easy. Nevertheless, human feet are too small with respect to the height of center of mass to perform static walking. Indeed we are performing dynamic walking in our daily life. The walking style with which we are so accustomed can be realized by dexterous control of the whole body balance which is essentially unstable. A biped walking machine is, therefore, beyond the scope of conventional mechanical engineering. This is the reason that so many researchers and engineers are attracted to biped walking machines as well as humanoid robots.

4.1 How to Realize Biped Walking?

Figure 4.1 shows the basic framework of biped walking control that will be used throughout this chapter. A set of time series of joint angles for desired walking is called a *walking pattern*, and to create it, we use a *walking pattern*

S. Kajita et al., *Introduction to Humanoid Robotics*,
Springer Tracts in Advanced Robotics 101,
DOI: 10.1007/978-3-642-54536-8_4,

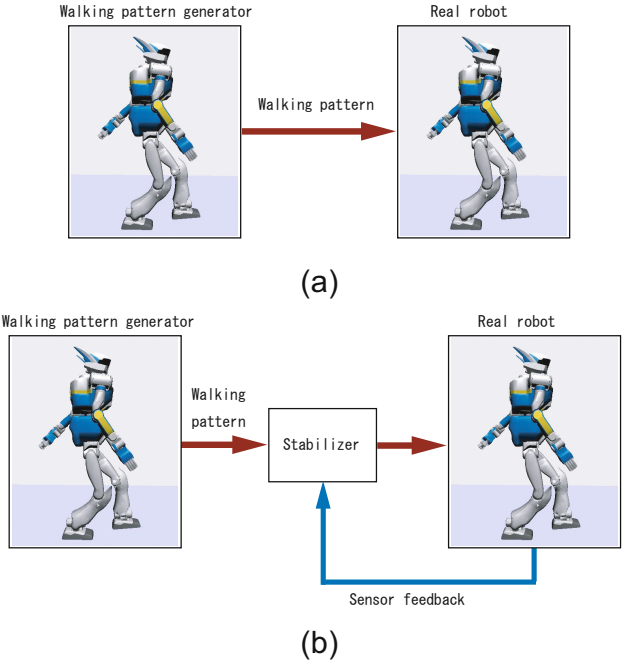


Fig. 4.1 Basic control framework in this book: (a)In an ideal condition, a biped robot can walk by following a walking pattern. (b)In a real environment, a biped robot need a stabilizer.

generator. In an ideal situation, biped walking can be realized just by giving a walking pattern to an actual robot (Fig. 4.1(a)). For this purpose, we must prepare an accurate model of the robot, a stiff mechanism which moves exactly as commanded and a perfect horizontal floor (a huge surface plate).

On the other hand, in a real situation, a life size humanoid robot can easily fall down by floor unevenness of only a few millimeters. A humanoid proportion and mass distribution tends to quickly amplify the posture error to an unstable level. To suppress this, we need the second software, which modifies the walking pattern by using gyros, accelerometers, force sensors and other devices. This is called a *stabilizer* (Fig. 4.1(b)).

The rest of this chapter is organized as follows. In section 4.2, 4.3 and 4.4, we explain walking pattern generators and explain stabilizers in section 4.5. In section 4.6, we spotlight the history of biped walking robot research. In section 4.7, we introduce a variety of biped control methods which are not restricted in the framework of Fig. 4.1.

4.2 Two Dimensional Walking Pattern Generation

In this section, we consider a basic principle of biped walking in 2D and derive an algorithm for walking pattern generation.

4.2.1 Two Dimensional Inverted Pendulum

Coarse-graining is a powerful method to handle a complex system. In celestial mechanics, researchers approximate the Sun and the planets as point masses while they have their own internal structure, and they can still calculate the orbits of the solar system with sufficient accuracy. In thermodynamics, vast states of molecules, number of the order of 10^{23} are coarse-grained as a few parameters like temperature or entropy, and it makes it possible for us to predict the thermodynamic phenomenon.

Likewise, to extract an *essence* of biped locomotion, we make three assumptions as coarse-graining on a humanoid robot with more than 30 DOF made of over thousands of mechanical and electrical parts. First, we assume that all the mass of the robot is concentrated at its center of mass (CoM). Second, we assume that the robot has massless legs, whose tips contact the ground at single rotating joints. At last, we only consider the forward/backward and the up/down motions of the robot, neglecting lateral motion. In other words, we assume the robot motion is constrained to the *sagittal plane* defined by the axis of walking direction and vertical axis. With these assumptions, we model a robot as a 2D inverted pendulum as shown in Fig. 4.2.

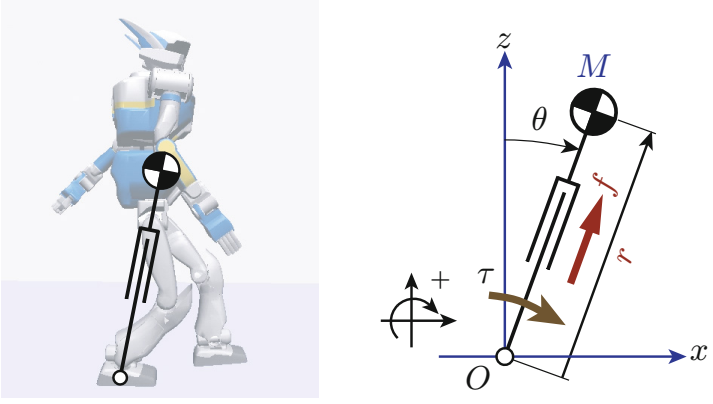


Fig. 4.2 2D inverted pendulum: The simplest model for a human or a walking robot. It consists of the center of mass (CoM) and massless telescopic leg. We measure the inclination of the pendulum θ from vertical, positive for counter clockwise rotation.

The inputs of the pendulum are the torque τ at the pivot and the kick force f at the prismatic joint along the leg. The dynamics of the pendulum are described as a couple of differential equations[1] as follows

$$r^2\ddot{\theta} + 2r\dot{r}\dot{\theta} - gr\sin\theta = \tau/M$$
$$\ddot{r} - r\dot{\theta}^2 + g\cos\theta = f/M.$$

We can simulate the behavior of the inverted pendulum by integrating them numerically with given input torque.

One of the important limitations is we cannot use big torque τ since the feet of biped robot is very small. If a walking robot has a point contact like a stilt, we must use

$$\tau = 0. \tag{4.1}$$

In this case, the pendulum will almost always fall down, unless the CoM is located precisely above the pivot. Even with such a simple pendulum, we have a variety of falling patterns corresponding to different kick forces, f, as illustrated in Fig. 4.3.

The most interesting case is Fig. 4.3(d) where the CoM moves horizontally under the kick force

$$f = \frac{Mg}{\cos\theta}. \tag{4.2}$$

Figure 4.4 illustrates the reason for the horizontal motion of the CoM.

Intuitively, we can say the pendulum is keeping the CoM height by extending its leg as fast as it is falling. We call this the Linear Inverted Pendulum [115][2].

4.2.2 Behavior of Linear Inverted Pendulum

The Linear Inverted Pendulum provides us a mathematically easy treatment of dynamics. Let us investigate the horizontal motion.

1 Horizontal Dynamics

By investigating Fig. 4.4 again, we see the horizontal component of the kick force f remains while the vertical component is canceled by gravity. The horizontal component accelerates the CoM horizontally, thus we have

$$M\ddot{x} = f\sin\theta. \tag{4.3}$$

[1] These equations can be derived by using Lagrange's method. Joseph-Louis Lagrange (1736-1813) was a French mathematician born in Italy. He is famous as a discoverer of the *Lagrange points*, the most suitable area to place space colonies.

[2] This is the simplest version of Linear Inverted Pendulum (LIP). The concept will be extended throughout of this chapter. Also, the LIP is an example of Zero-dynamics which is discussed in the textbook by Westervelt et al. [24]

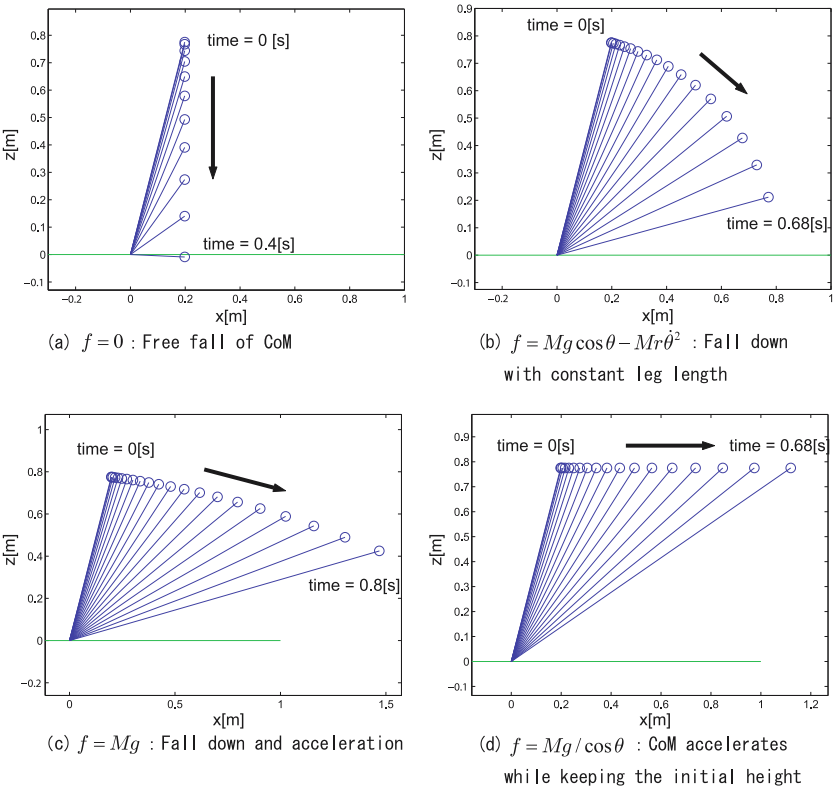


Fig. 4.3 Falling inverted pendulum under the various kick force f. The pivot torque is kept zero ($\tau = 0$) at all time.

By substituting (4.2), we get

$$M\ddot{x} = \frac{Mg}{\cos\theta}\sin\theta = Mg\tan\theta = Mg\frac{x}{z}$$

where, x, z gives the CoM of the inverted pendulum. By rewriting above equation, we obtain a differential equation for the horizontal dynamics of the CoM

$$\ddot{x} = \frac{g}{z}x. \tag{4.4}$$

Since we have constant z in a Linear Inverted Pendulum, we can easily solve this ordinary differential equation

$$x(t) = x(0)\cosh(t/T_c) + T_c\dot{x}(0)\sinh(t/T_c) \tag{4.5}$$

$$\dot{x}(t) = x(0)/T_c\sinh(t/T_c) + \dot{x}(0)\cosh(t/T_c) \tag{4.6}$$

$$T_c \equiv \sqrt{z/g}$$

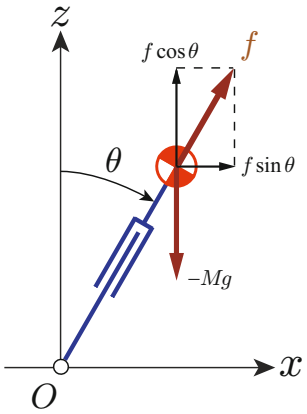


Fig. 4.4 The reason for the horizontal locus of the CoM. The kick force $f = Mg/\cos\theta$ always balance with the gravity acting on the point mass.

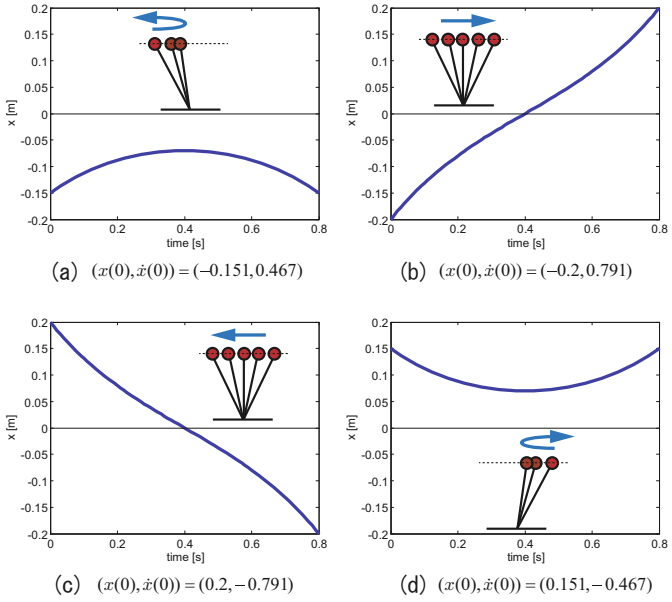


Fig. 4.5 Linear Inverted Pendulum under various initial conditions. CoM height: $z = 0.8$ m.

where T_c is the time constant depending the height of the CoM and gravity acceleration.

The initial position and velocity are given by $x(0), \dot{x}(0)$, which together are called the *initial conditions*. Figure 4.5 shows the motions under various initial conditions.

2 Time for Transfer

In many cases, we want to know the time that CoM takes to travel from one point to another. When an initial condition $(x_0, \dot{x}_0)$ and a condition $(x_1, \dot{x}_1)$ at certain moment is given, they are connected by following equations by using (4.5) and (4.6)

$$x_1 = x_0 \cosh(\tau/T_c) + T_c \dot{x}_0 \sinh(\tau/T_c) \tag{4.7}$$

$$\dot{x}_1 = x_0/T_c \sinh(\tau/T_c) + \dot{x}_0 \cosh(\tau/T_c) \tag{4.8}$$

where τ is the period that the CoM takes a trip from $(x_0, \dot{x}_0)$ to $(x_1, \dot{x}_1)$.

By using the relationship of $\cosh(x) = \frac{e^x + e^{-x}}{2}$, $\sinh(x) = \frac{e^x - e^{-x}}{2}$, these equations can be rewritten as

$$x_1 = \frac{x_0 + T_c \dot{x}_0}{2} e^{\tau/T_c} + \frac{x_0 - T_c \dot{x}_0}{2} e^{-\tau/T_c}, \tag{4.9}$$

$$\dot{x}_1 = \frac{x_0 + T_c \dot{x}_0}{2T_c} e^{\tau/T_c} - \frac{x_0 - T_c \dot{x}_0}{2T_c} e^{-\tau/T_c}. \tag{4.10}$$

From (4.9)$+T_c\times$(4.10), we get

$$x_1 + T_c \dot{x}_1 = (x_0 + T_c \dot{x}_0) e^{\tau/T_c}.$$

Therefore, τ can be calculated as

$$\tau = T_c \ln \frac{x_1 + T_c \dot{x}_1}{x_0 + T_c \dot{x}_0}. \tag{4.11}$$

Similarly, we can calculate τ from (4.9)$-T_c\times$(4.10) as

$$\tau = T_c \ln \frac{x_0 - T_c \dot{x}_0}{x_1 - T_c \dot{x}_1}. \tag{4.12}$$

Basically, (4.11) and (4.12) gives exactly the same result, except in the singular case where both of the numerator and denominator go close to zero in one of the equations.

4.2.3 Orbital Energy

To understand motions of a linear inverted pendulum intuitively, it is useful to imagine a potential as shown in Fig. 4.6. Figure 4.6(a) shows the case which the CoM changes its moving direction when the initial velocity is not enough. In Fig. 4.6(b), the CoM keeps original moving direction due to sufficient initial speed. In this case, the CoM undergoes minimum speed at the top of the potential where the CoM is just above the ankle pivot.

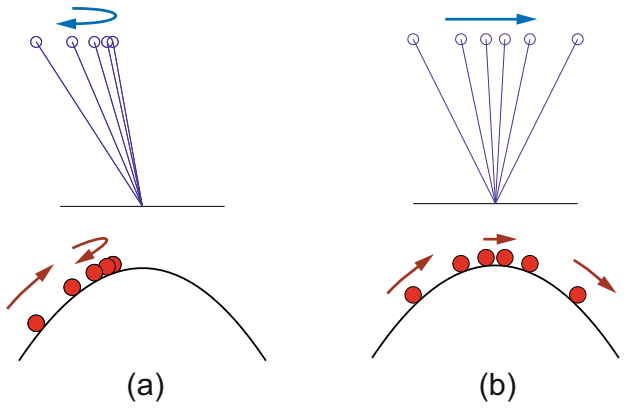


Fig. 4.6 Linear Inverted Pendulum and imaginary potential

Let us calculate the relationship between this imaginary potential and the motion of the CoM. We multiply $\dot{x}$ on both side of the equation of motion (4.4), then integrate it

$$\dot{x}(\ddot{x} - \frac{g}{z}x) = 0$$
$$\int \{\ddot{x}\dot{x} - \frac{g}{z}x\dot{x}\}dt = constant.$$

The result is

$$\frac{1}{2}\dot{x}^2 - \frac{g}{2z}x^2 = constant \equiv E. \tag{4.13}$$

The first term of the left hand side is kinetic energy and the second term is the imaginary potential energy which is illustrated by Fig. 4.6. In this calculation, we assume energies per unit mass so that we can omit mass of the CoM.

Let us call the sum of the kinetic energy and the imaginary potential energy as, E, the *orbital energy*[3] [36]. Equation (4.13) shows that **the orbital energy is conserved in the motion of Linear Inverted Pendulum**.

[3] In mechanics, this value is called constant of motion [36].

In the case of Fig. 4.6(a), where the CoM reverts without getting to the top of the potential, the orbital energy is given by

$$E = -\frac{g}{2z}x_{rev}^2, \tag{4.14}$$

where x_{rev} is the horizontal position of the CoM at the moment the reversion. In this case the orbital energy is negative or zero.

In the case of Fig. 4.6(b), where the CoM passes the top of the potential, the orbital energy is given by

$$E = \frac{1}{2}\dot{x}_{top}^2, \tag{4.15}$$

where $\dot{x}_{top}(> 0)$ is the speed at the instant that the CoM passes above of the ankle pivot. In this case, the orbital energy is positive.

Suppose we have obtained the CoM position and the speed of an inverted pendulum at certain instant. By checking the sign of the orbital energy E calculated by (4.13), we can immediately predict whether the CoM will pass the potential or not. If $E > 0$, we can predict the CoM speed at the top of the potential as

$$|\dot{x}_{top}| = \sqrt{2E}.$$

If $E < 0$, we can predict the position where the CoM will revert as

$$|x_{rev}| = \sqrt{-\frac{2zE}{g}}.$$

4.2.4 Support Leg Exchange

Although the motion of a linear inverted pendulum is determined only by its initial condition, we can control its speed because an initial condition can be modified by the touchdown timing. As illustrated in Fig. 4.7, a support exchange of quickened touchdown will decelerate the walking speed, and a support exchange of delayed touchdown will accelerate the walking. This corresponds to our experience, for example, we put down our leg quickly on the ground for sudden stops.

Let us figure out the relationship between a support exchange and the pendulum motions. Figure 4.8 shows the moment of a support exchange. The step length is s, the position of the CoM with respect to the former contact point is x_f, and the speed of the CoM at the instant of the exchange is v_f. For simplicity, we assume that the leg support is exchanged instantly, therefore, v_f is the final speed of the previous support phase as well as the initial speed of the new support phase.

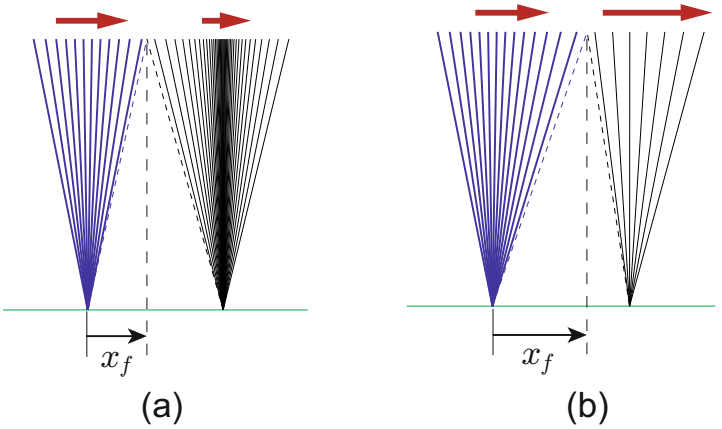


Fig. 4.7 Control of walking speed (with fixed step length) (a) Earlier touchdown of the next step results slow down (b) Later touchdown of the next step results speed up

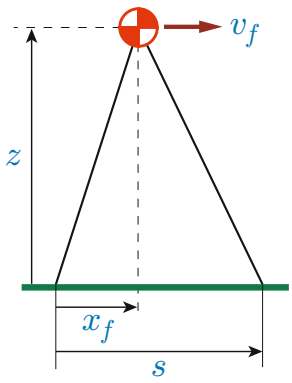


Fig. 4.8 State of a support leg exchange

By defining the orbital energies before and after the exchange as E_1 and E_2 respectively, we have

$$E_1 = -\frac{g}{2z}x_f^2 + \frac{1}{2}v_f^2 \tag{4.16}$$

$$E_2 = -\frac{g}{2z}(x_f - s)^2 + \frac{1}{2}v_f^2. \tag{4.17}$$

When the orbital energies E_1, E_2 were given, we can calculate the necessary exchange condition by eliminating v_f from (4.16), (4.17) to obtain

$$x_f = \frac{z}{gs}(E_2 - E_1) + \frac{s}{2}. \tag{4.18}$$

The speed at the moment of the exchange can be calculated from (4.16)

$$v_f = \sqrt{2E_1 + \frac{g}{z}x_f^2}. \tag{4.19}$$

4.2.5 Planning a Simple Biped Gait

Let us design a simple walking pattern using the result of former sections. We assume an ideal biped robot on a level plane walks just one step and stops (Fig. 4.9). The robot exchanges its support twice and three orbital energies must be specified.

For the walk start(a→b), the orbital energy is specified by the initial position of the CoM,

$$E_0 = -\frac{g}{2z}x_s^2.$$

For the step (b→c→d), the orbital energy is specified by the speed v_1 at the moment that CoM passes over the supporting point

$$E_1 = \frac{1}{2}v_1^2.$$

For the walk finish (d→e), the orbital energy is specified by the final position of the CoM.

$$E_2 = -\frac{g}{2z}x_e^2.$$

Using (4.18), the first support exchange condition x_{f0} is obtained from E_0, E_1, and the second support exchange condition x_{f1} is obtained from E_1, E_2. Desired walking motion is realized by controlling the swing leg so that those exchanges occur at the right time.

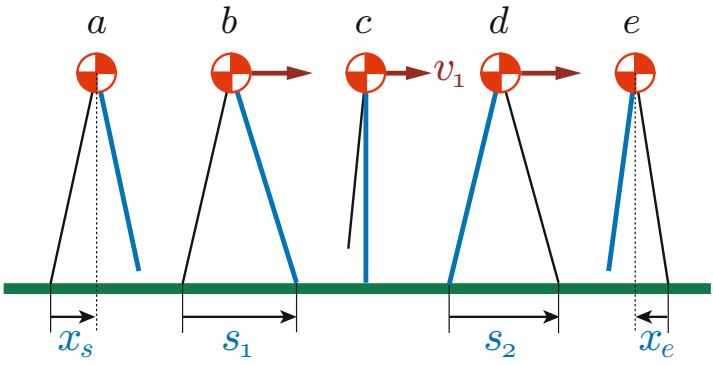


Fig. 4.9 Specification for a walk of one step forward. We need support leg exchanges twice.

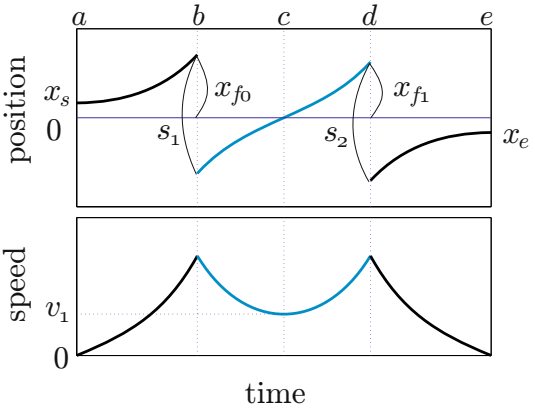


Fig. 4.10 Planned trajectory of the center of mass, position and velocity

Since speed at each exchange are calculated by (4.19), the complete trajectory of each step can be obtained. Figure 4.10 shows the time profile of the position and the velocity of the CoM. The position graph takes origin at each support point, therefore, the plot jumps at the moment of support exchange (b, d) and the amount of the jump indicates the step length. The velocity graph depicts the dynamic change of CoM speed and the peaks at support exchange.

4.2.6 Extension to a Walk on Uneven Terrain

Although the method explained so far was limited to a walk on level ground, we can use the same method for walking on uneven terrain with a small modification. Let us explain this.

Returning to the inverted pendulum model of Fig. 4.2, we consider the case that the center of mass moves on a sloped line as illustrated in Fig. 4.11 described by

$$z = kx + z_c \tag{4.20}$$

where k is the inclination of the line, z_c is the z intersection of the line. We call the line which the CoM moves along as *constraint line*.

Let us calculate the kick force f to realize such motion. First, we decompose the kick force f into the horizontal part f_x and the vertical part f_z

$$f_x = f \sin\theta = (x/r)f \tag{4.21}$$

$$f_z = f \cos\theta = (z/r)f. \tag{4.22}$$

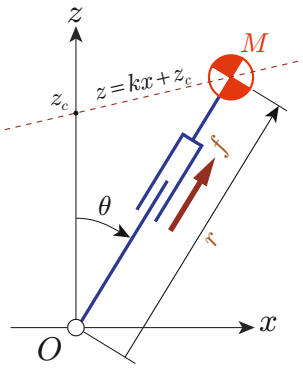


Fig. 4.11 The CoM is controlled to move on the constraint line (broken line) by the kick force f, while the ankle rotates freely($\tau = 0$).

To let the CoM move along the constraint line, the sum of the kick force and the gravity force must be parallel with the constraint line. That is explained as

$$f_x : f_z - Mg = 1 : k. \tag{4.23}$$

Substituting (4.21) and (4.22) into (4.23) and solving for f, we obtain

$$f = \frac{Mgr}{z - kx}. \tag{4.24}$$

A simpler equation can be obtained by using the constraint equation (4.20)

$$f = \frac{Mgr}{z_c}. \tag{4.25}$$

Therefore, the CoM moves on a constraint line by applying the kick force f in proportional with the leg length r (in addition, we need the initial condition matching the constraint). Such pendulum motions are illustrated in Fig. 4.12. In this graph, the arrows are indicating the magnitude and direction of the kick force.

Let us see the dynamics of the CoM under this control. The horizontal dynamics of the CoM can be obtained by substituting (4.25) into (4.21), and applying $f_x = M\ddot{x}$ to obtain

$$\ddot{x} = \frac{g}{z_c} x. \tag{4.26}$$

Surprisingly, this is identical to (4.4), which was derived for the horizontal motion of CoM! To make clear the meaning of this result, we display in Fig. 4.13 the simulation of two inverted pendula moving on different constraint lines. The two constraint lines have different slopes but the same intersection z_c, and the pendula start from the same horizontal position x_0. One might expect that the pendulum on an ascending line ($k > 0$) will decelerate

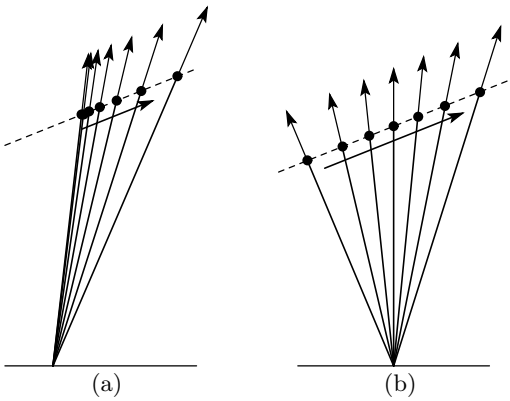


Fig. 4.12 Kick force vector and pendulum motion

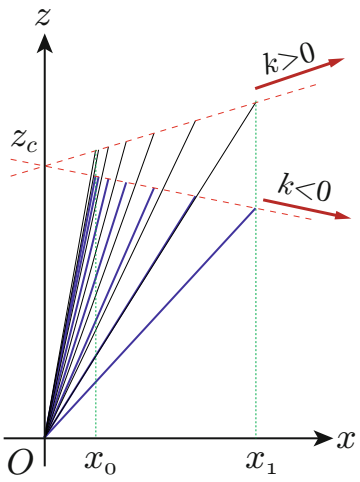


Fig. 4.13 Horizontal motion of a linear inverted pendulum is independent from the slope of the constraint line.

and the pendulum on an descending line ($k < 0$) will accelerate. Nevertheless, the horizontal motions of two pendula are exactly same and they arrive at x_1 simultaneously as shown in Fig. 4.13. This is because the gravity effect is canceled by the kick force and the pendulum motion is governed only by the horizontal location of the CoM.

By using this fact, we can easily design a walking pattern on uneven terrain. Fig. 4.14 shows an example planning of stair climbing. First, we set proper landing points (triangles), then specify constraint lines by connecting the points which are located at the height of z_c from the landing points. By controlling the CoM to move along these constraint lines, we get the horizontal dynamics of CoM for each step as

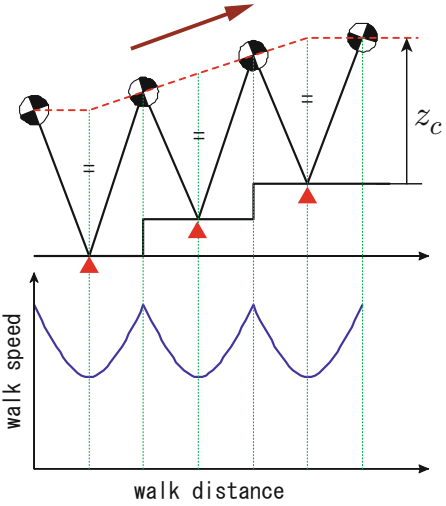


Fig. 4.14 Walking on the stairs by using linear inverted pendulum. (above) Setting of constraint lines (lower) horizontal velocity of the center of mass.

$$\ddot{x} = \frac{g}{z_c} x.$$

Thus, we can apply the method of the previous section. The lower graph of Fig. 4.14 indicates that the horizontal speed of the CoM is not affected by the stair climbing.

By these method, we can easily produce walking patterns for various ground surface geometries. The authors have developed a simple biped robot (Fig. 4.15) which could walk over stairs and obstacles by real-time sensing [116].

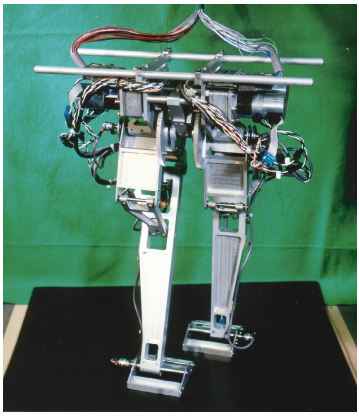


Fig. 4.15 Meltran II, a biped robot with bird-like legs. The light weight legs allows us to treat this robot as a linear inverted pendulum.

4.3 3D Walking Pattern Generation

In this section, we extend the linear inverted pendulum to 3D and examine its nature. Using a 3D linear inverted pendulum, a 3D walking pattern generation is described.

4.3.1 3D Linear Inverted Pendulum

Let us approximate a biped walking robot in 3D space as an imaginary inverted pendulum of Fig. 4.16, which consists of the CoM of the robot and a massless leg connecting the CoM and the supporting point. We assume the pendulum can freely rotate about the supporting point and the leg can change its length by using a kick force f. We can decompose the kick force f into three components, x,y and z

$$f_x = (x/r)f \tag{4.27}$$
$$f_y = (y/r)f \tag{4.28}$$
$$f_z = (z/r)f \tag{4.29}$$

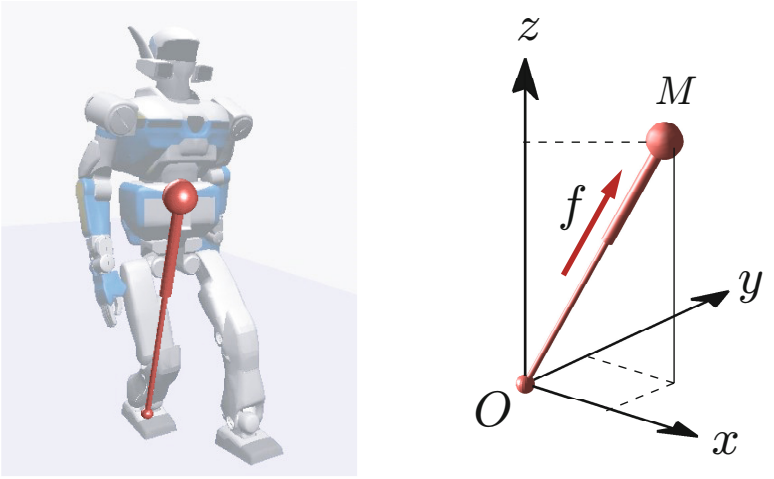


Fig. 4.16 3D inverted pendulum as an approximated walking robot. The supporting point is a spherical joint which allows free rotation. The leg can change its length by generating a kick force f.

where r is the distance between the supporting point and the CoM. Only the kick force and gravity act on the CoM, thus, the motion equation of the CoM is given

$$M\ddot{x} = (x/r)f \tag{4.30}$$
$$M\ddot{y} = (y/r)f \tag{4.31}$$
$$M\ddot{z} = (z/r)f - Mg. \tag{4.32}$$

As we did for 2D inverted pendulum, let us consider a constraint for the CoM. For a 3D inverted pendulum, we introduce a *constraint plane* defined as

$$z = k_x x + k_y y + z_c, \tag{4.33}$$

where k_x, k_y determine the slope and z_c determines the height of the constraint plane.

To let the CoM move along this plane, we need its acceleration be orthogonal to the normal vector of the constraint. Therefore, we need

$$[f(\frac{x}{r}) \quad f(\frac{y}{r}) \quad f(\frac{z}{r}) - Mg] \begin{bmatrix} -k_x \\ -k_y \\ 1 \end{bmatrix} = 0. \tag{4.34}$$

By solving this equation for f, and substituting into (4.33), we obtain

$$f = \frac{Mgr}{z_c}. \tag{4.35}$$

The center of mass moves on the constraint plane by applying the kick force f in proportion to the leg length r. Figure 4.17 shows an image of this motion.

The horizontal dynamics of the CoM can be derived from (4.30) and (4.31) by substituting the kick force of (4.35) to obtain

$$\ddot{x} = \frac{g}{z_c}x, \tag{4.36}$$
$$\ddot{y} = \frac{g}{z_c}y. \tag{4.37}$$

These are linear equations having z_c, the intersection of the constraint plane as the only parameter. The inclination parameters k_x, k_y of the constraint plane do not affect the horizontal motion of the CoM since they are not part of (4.36) and (4.37). We call such pendulum as *3D linear inverted pendulum*[4].

[4] The 3D linear inverted pendulum was originally discovered by Hara, Yokogawa and Sadao [59]. It was expanded to take into account of the ankle torque by Kajita, Matsumoto and Saigo [118].

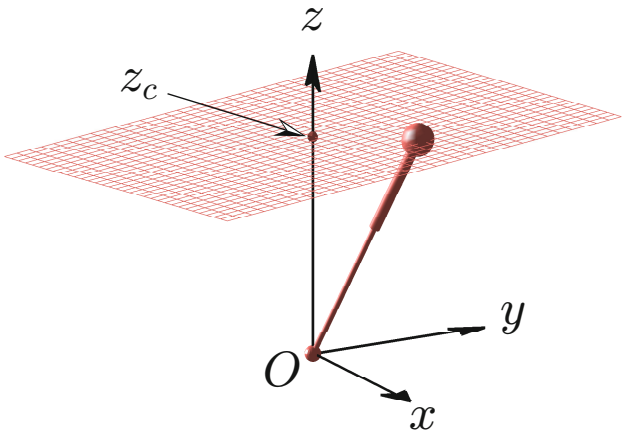
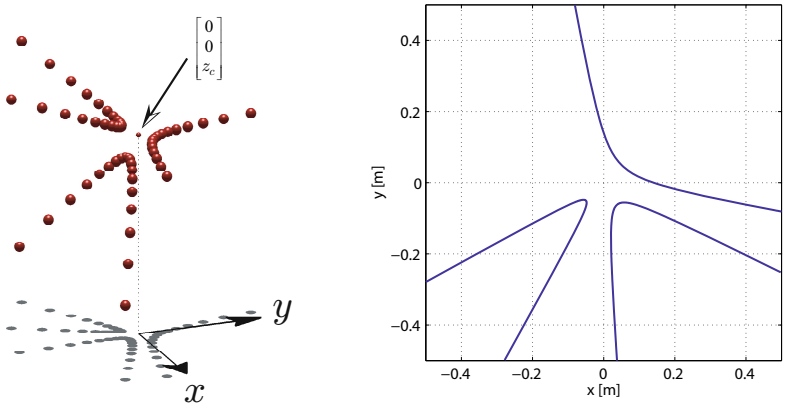


Fig. 4.17 3D linear inverted pendulum. The center of mass moves on the constraint plane by properly controlling the kick force. Inclination of the constraint plane does not affect to the horizontal motion of the CoM.

4.3.2 Natures of the 3D Linear Inverted Pendulum

The nature of 3D linear inverted pendulum is much more interesting although it is a concatenation of two 2D linear inverted pendula. Figure 4.18 shows three trajectories of a 3D linear inverted pendulum whose constraint plane



(a) (b)

Fig. 4.18 Motions of 3D linear inverted pendulum. (a)CoM trajectories in 3D space (b)Horizontal projections of trajectories.

has the same z intersection. We can observe an impressive comet-like trajectory having the z intersection $[0\ 0\ z_c]$ as the focal point[5]. We can treat the dynamics of these three-dimensional trajectories as the projection onto the xy-plane (Fig. 4.18(b)) by neglecting the inclination of the constraint plane.

1 Kepler's Second Law

From an analogy from celestial mechanics, let us check Kepler's second law: A planet has constant areal speed. Areal velocity v_{area} is area swept by the line connecting origin and the CoM per unit time and is given as

$$v_{area} = \frac{1}{2}(x\dot{y} - \dot{x}y). \tag{4.38}$$

We can check the time derivative of areal speed for 3D linear inverted pendulum as follows

$$\begin{aligned}\frac{d}{dt}(v_{area}) &= \frac{1}{2}(\dot{x}\dot{y} + x\ddot{y} - \ddot{x}y - \dot{x}\dot{y}) \\ &= \frac{1}{2}(x\ddot{y} - \ddot{x}y) \quad \leftarrow (\text{substitute eqs.(4.36) and (4.37)}) \\ &= \frac{1}{2}(x\frac{g}{z_h}y - \frac{g}{z_h}xy) \\ &= 0.\end{aligned}$$

Thus, the areal speed of 3D linear inverted pendulum is constant[6].

2 Rotation of the Reference Frame

Next we consider coordinate transformation from the original reference frame xy to a new reference frame $x'y'$ rotated θ from it as in Fig. 4.19. The transformation is

$$x = cx' - sy' \tag{4.39}$$

$$y = sx' + cy' \tag{4.40}$$

$$c \equiv \cos\theta, s \equiv \sin\theta.$$

[5] LIP trajectories are different from comet trajectories which travel around the Sun. More proper analogy is a trajectory of particles in Rutherford scattering experiment which proved the existence of the nucleus of an atom. As shown subsequently, this is indeed a very good analogy.

[6] Generally, areal speed is conserved by any motion in a central force field. It represents the conservation of angular momentum in a different way.

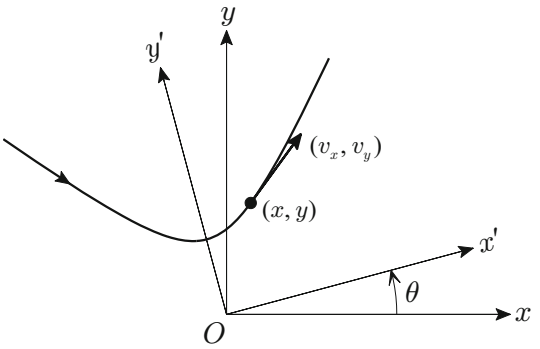


Fig. 4.19 Horizontal projection of a 3D-LIPM trajectory and the new reference frame $x'y'$ rotated θ from the original frame

Let us confirm that the equations of the 3D linear inverted pendulum (4.36) and (4.37) are still valid in the new frame $x'y'$. By substituting the coordinate transformation given by above equations we get

$$c\ddot{x}' - s\ddot{y}' = \frac{g}{z_h}(cx' - sy') \tag{4.41}$$

$$s\ddot{x}' + c\ddot{y}' = \frac{g}{z_h}(sx' + cy'). \tag{4.42}$$

From $c\times(4.41)$ $+s\times(4.42)$ and $-s\times(4.41)$ $+c\times(4.42)$ we get the dynamics represented in $x'y'$ frame given by

$$\ddot{x}' = \frac{g}{z_c}x'$$
$$\ddot{y}' = \frac{g}{z_c}y'.$$

This transformation is possible for any given rotation angle θ. Therefore, we have confirmed that the 3D linear inverted pendulum dynamics can be always separated into two orthogonal components independently from the orientation of the reference frame[7].

3 Geometry of 3D Linear Inverted Pendulum Trajectory

We can calculate the geometry of the trajectory using the transformation of Fig. 4.19 and orbital energies. The orbital energy along the x' axis of a new frame is given as

[7] Motions under gravity force or electro-static force, which is proportional to inverse square of the distance cannot be treated like this.

$$E'_x = -\frac{g}{2z_c}(cx + sy)^2 + \frac{1}{2}(c\dot{x} + s\dot{y})^2. \tag{4.43}$$

This orbital energy E'_x changes by the rotation of the frame θ and takes extreme value when x' or y' correspond to the axis of symmetry of the trajectory. From

$$\begin{aligned} \frac{\partial E'_x}{\partial \theta} &= -A\cos 2\theta + \frac{B}{2}\sin 2\theta = 0 \\ A &\equiv (g/z_c)xy - \dot{x}\dot{y} \\ B &\equiv (g/z_c)(x^2 - y^2) - (\dot{x}^2 - \dot{y}^2) \end{aligned} \tag{4.44}$$

we can calculate θ which indicates the symmetry axis of the trajectory,

$$\theta = \begin{cases} (1/2)\tan^{-1}(2A/B) & (\text{if } B \neq 0) \\ \pi/4 & (\text{if } A \neq 0, B = 0). \end{cases} \tag{4.45}$$

In the case of $A = 0, B = 0$, the trajectory becomes a straight line toward origin or a straight line from origin, and the line itself is a symmetry axis.

Suppose we have already chosen the frame x, y to fit the axis of symmetry. In this case, the orbital energy E'_x must take extreme value with $\theta = 0$. By substituting $\theta = 0$ into (4.44), we get

$$(g/z_c)xy - \dot{x}\dot{y} = 0.$$

After transferring the second term to the right side, we square both sides of the equation to obtain

$$(g/z_c)^2 x^2 y^2 = \dot{x}^2\dot{y}^2.$$

We substitute the following two equations which were derived from the definition of orbital energy

$$\dot{x}^2 = 2E_x + \frac{g}{z_c}x^2 \quad \dot{y}^2 = 2E_y + \frac{g}{z_c}y^2$$

to obtain an equation which represents geometric form of 3D linear inverted pendulum trajectory given by

$$\frac{g}{2z_c E_x}x^2 + \frac{g}{2z_c E_y}y^2 + 1 = 0. \tag{4.46}$$

This is an equation of hyperbola since one of E_x, E_y is negative and the other is positive[8].

[8] Hyperbolic trajectories are observed in Rutherford scattering or swing-by trajectories of spacecraft like Voyager I and II. I feel excitement finding similar trajectories in elementary particles, in planetary space, and in biped locomotion. Don't you?!

4.3.3 3D Walking Pattern Generation

Figure 4.20 shows an example of walking pattern based on the 3D linear inverted pendulum. By setting a proper constraint plane, we can apply the same pattern to stairs or an uneven floor.

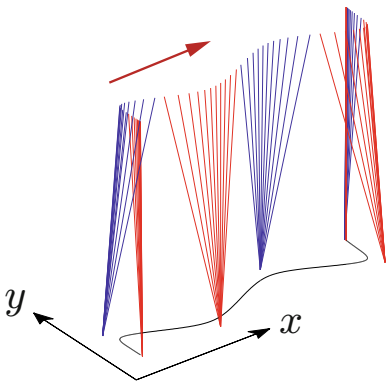


Fig. 4.20 Walking pattern on flat floor based on 3D linear inverted pendulum. Three forward steps from a standstill to a stop.

For three dimensional walking, we need a simultaneous support exchange for the x and y directions, thus we cannot use the walking pattern generation explained in Section 4.2.5, which requires an arbitrary time of support exchange. In the following discussion, we consider walking with constant pace of support exchange and denote T_{sup} for support period of each step.

1 Walk Primitive

Let us define a piece of a 3D linear inverted pendulum trajectory as Fig. 4.21, which is symmetric about y axis and defined in a period of $[0 \quad T_{sup}]$. As previously described, Fig. 4.21(a) is a part of hyperbola. We will call this piece of trajectory as a *walk primitive*.

When a support time T_{sup} and an intersection of constraint plane z_c are given, a walk primitive is uniquely determined by its terminal position $(\bar{x}, \bar{y})$ since it is symmetric. The terminal speed $(\bar{v}_x, \bar{v}_y)$ can be calculated as follows.

Using the symmetric nature of the walk primitive, the initial condition along the x axis is $(-\bar{x},\ \bar{v}_x)$ and the terminal position is $\bar{x}$. From the analytic solution of linear inverted pendulum (4.5), we have

$$\bar{x} = -\bar{x}C + T_c \bar{v}_x S \tag{4.47}$$

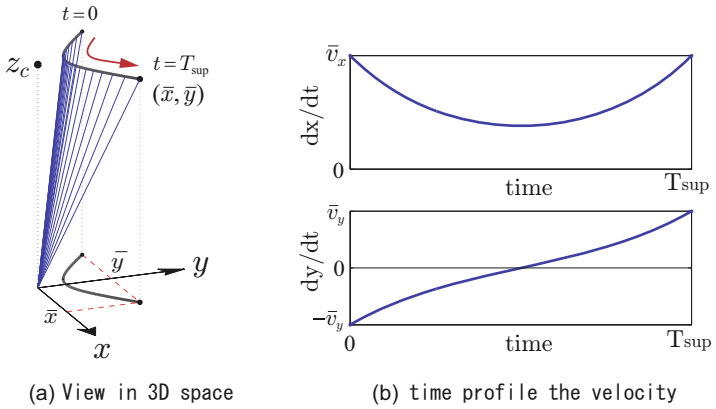


Fig. 4.21 Walk primitive: basic 3D walking pattern

where

$$T_c \equiv \sqrt{\frac{z_c}{g}},\ C \equiv \cosh \frac{T_{sup}}{T_c},\ S \equiv \sinh \frac{T_{sup}}{T_c}.$$

Solving (4.47) for the terminal velocity $\bar{v}_x$ gives

$$\bar{v}_x = \bar{x}(C+1)/(T_c S). \tag{4.48}$$

Likewise, for the y component of the walk primitive, the initial condition is $(\bar{y},\ -\bar{v}_y)$ and the terminal position is $\bar{y}$, thus we get the terminal velocity as,

$$\begin{aligned} \bar{y} &= \bar{y}C + T_c(-\bar{v}_y)S, \\ \bar{v}_y &= \bar{y}(C-1)/(T_c S). \end{aligned} \tag{4.49}$$

Walking primitives allows us to easily produce a walking trajectory. For example, a straight walk with step length of $2\bar{x}$ can be made by connecting identical walk primitives, reversing the sign of the y-component after each contact.

2 Walk Parameters

In practical situations like stair climbing or obstacle avoidance, it is frequently required to directly specify the foot placements. Figure 4.22 illustrates an example of simple foot placements, which can be represented by using step length and step width as the following data.

n	1	2	3	4	5
$s_x^{(n)}$	0.0	0.3	0.3	0.3	0
$s_y^{(n)}$	0.2	0.2	0.2	0.2	0.2

where s_x are the step length along the walking direction, and s_y are the step width for lateral direction. We call this data the *walk parameters* since it is

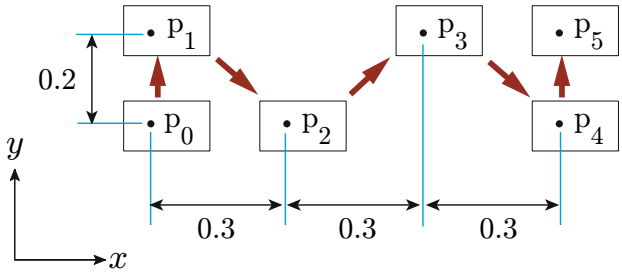


Fig. 4.22 Points of foot placement $p_0 \ldots p_N$. Step length and width are determined from these points. The rectangles are footprints of a robot.

encoding the foot placements in Fig. 4.22. By putting superscript (n) to indicate the data of n-th step, the n-th footplace $(p_x^{(n)}, p_y^{(n)})$ can be represented as

$$\begin{bmatrix} p_x^{(n)} \\ p_y^{(n)} \end{bmatrix} = \begin{bmatrix} p_x^{(n-1)} + s_x^{(n)} \\ p_y^{(n-1)} - (-1)^n s_y^{(n)} \end{bmatrix}, \tag{4.50}$$

where $(p_x^{(0)}, p_y^{(0)})$ is the place of the first support foot, in this case, the right foot. If the first support foot was the left foot, we must replace $-(-1)^n$ by $+(-1)^n$ in the equation.

The walk primitive for the n-th step can be determined as

$$\begin{bmatrix} \bar{x}^{(n)} \\ \bar{y}^{(n)} \end{bmatrix} = \begin{bmatrix} s_x^{(n+1)}/2 \\ (-1)^n s_y^{(n+1)}/2 \end{bmatrix}. \tag{4.51}$$

Note that the walk primitive of the n-th step is determined by the $n+1$-th step length and width. This is necessary for the proper coordination between foot placements and walking motion.

The terminal velocity of a walk primitive is calculated by (4.48) and (4.49),

$$\begin{bmatrix} \bar{v}_x^{(n)} \\ \bar{v}_y^{(n)} \end{bmatrix} = \begin{bmatrix} (C+1)/(T_c S)\bar{x}^{(n)} \\ (C-1)/(T_c S)\bar{y}^{(n)} \end{bmatrix}. \tag{4.52}$$

The series of walk primitives determined in this way are discontinuous at the beginning and the end of a set of steps. The method to obtain a continuous and realizable walking pattern is explained in the next section.

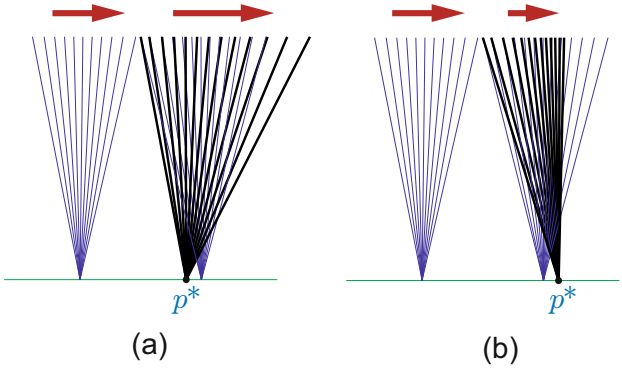


Fig. 4.23 Walking speed adjustment for fixed step cycle [41, 97] (a) Speed up by taking a shorter step. (b) Slow down by taking a longer step.

3 Modification of Foot Placements

For a robot walking with fixed step cycle, we can control its speed by adjusting foot placements[9]. Its intuitive explanation is depicted in Fig. 4.23.

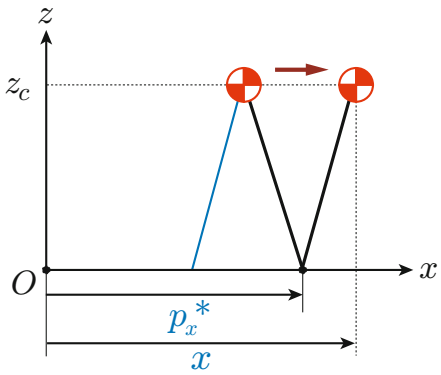


Fig. 4.24 A Linear Inverted Pendulum represented in the ground fixed frame

Let us denote the modified foot placement as p_x^* and calculate how it affects the walking motion. Figure 4.24 illustrates a linear inverted pendulum with respect to the ground fixed frame[10],

[9] In the control method of Fig. 4.7 (page.114), the step length was fixed, and the walking speed was controlled by modifying touch down timing.

[10] We will only discuss the motion along x-axis, but the same result is obtained for y-axis motion.

$$\ddot{x} = \frac{g}{z_c}(x - p_x^*). \tag{4.53}$$

Its analytical solution is,

$$x(t) = (x_i^{(n)} - p_x^*)\cosh(t/T_c) + T_c\dot{x}_i^{(n)}\sinh(t/T_c) + p_x^* \tag{4.54}$$

$$\dot{x}(t) = \frac{x_i^{(n)} - p_x^*}{T_c}\sinh(t/T_c) + \dot{x}_i^{(n)}\cosh(t/T_c), \tag{4.55}$$

where $x_i^{(n)}, \dot{x}_i^{(n)}$ are initial conditions at the beginning of the n-th step.

Therefore, the relationship between the foot placement p_x^* and the final state of n-th step is

$$\begin{bmatrix} x_f^{(n)} \\ \dot{x}_f^{(n)} \end{bmatrix} = \begin{bmatrix} C & T_c S \\ S/T_c & C \end{bmatrix} \begin{bmatrix} x_i^{(n)} \\ \dot{x}_i^{(n)} \end{bmatrix} + \begin{bmatrix} 1 - C \\ -S/T_c \end{bmatrix} p_x^*. \tag{4.56}$$

Step 1 Set support period T_{sup} and walk parameters s_x, s_y. Set initial position of CoM (x, y) and initial foot placement $(p_x^*, p_y^*) = (p_x^{(0)}, p_y^{(0)})$.
Step 2 $T := 0, n := 0$.
Step 3 Integrate equation of linear inverted pendulum (4.53) (and the equation for y-axis) from T to $T + T_{sup}$.
Step 4 $T := T + T_{sup}, \quad n := n + 1$
Step 5 Calculate the next foot place $(p_x^{(n)}, p_y^{(n)})$ using (4.50).
Step 6 Set the next walk primitive $(\bar{x}^{(n)}, \bar{y}^{(n)})$ using (4.51) and (4.52).
Step 7 Calculate target state $(x^d, \dot{x}^d)$ by (4.57). Calculate target state $(y^d, \dot{y}^d)$ by corresponding equation.
Step 8 Calculate modified foot placement (p_x^*, p_y^*) by (4.59) (as well as y component).
Step 9 Goto Step 3.

Fig. 4.25 Algorithm of walking pattern generation based on 3D-LIP

As the target state we use the terminal state of the walk primitive presented in the ground frame

$$\begin{bmatrix} x^d \\ \dot{x}^d \end{bmatrix} = \begin{bmatrix} p_x^{(n)} + \bar{x}^{(n)} \\ \bar{v}_x^{(n)} \end{bmatrix}. \tag{4.57}$$

Let us calculate the foot placement which ends up the final state closest to the target $(x^d, \dot{x}^d)$. The evaluation function can be defined as

$$N \equiv a(x^d - x_f^{(n)})^2 + b(\dot{x}^d - \dot{x}_f^{(n)})^2 \tag{4.58}$$

where a, b are positive weights. Substituting (4.56) into the evaluation function and by using $\partial N/\partial p_x^* = 0$, we can obtain the foot placement which will minimize N

$$\begin{aligned} p_x^* = &-\frac{a(C-1)}{D}(x^d - Cx_i^{(n)} - T_c S\dot{x}_i^{(n)}) \\ &-\frac{bS}{T_c D}(\dot{x}^d - \frac{S}{T_c}x_i^{(n)} - C\dot{x}_i^{(n)}) \\ D \equiv &\, a(C-1)^2 + b(S/T_c)^2. \end{aligned} \tag{4.59}$$

The method of walking pattern generation can be organized as an algorithm shown as Fig. 4.25.

Figure 4.26 shows the generated walking pattern based on the walk parameters of Fig. 4.22. We can observe a slight back step at the walk start to obtain acceleration, and a slightly wider step at the walk end to obtain deceleration. Since such modifications of foot placement are necessary, this method cannot realize the exact foot placement specified by the walk parameters. Nevertheless, since (4.59) guarantees the error converges to zero, the robot can take a specified foot placement in steady walking.

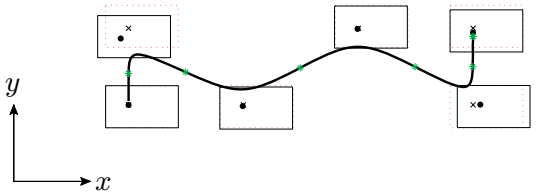


Fig. 4.26 Walking pattern generated by the proposed algorithm. Bold line is the trajectory of the CoM, black circles are modified foot placements. The desired foot placements are shown by ×. For walk start and walk end, foot placements are modified largely to obtain proper acceleration and deceleration. $z_c = 0.8, T_{sup} = 0.8$,$a = 10, b = 1$.

For diagonal walking, we change s_y for each step.

n	1	2	3	4	5
$s_x^{(n)}$	0.0	0.2	0.2	0.2	0
$s_y^{(n)}$	0.2	0.3	0.1	0.3	0.2

Figure 4.27 shows the pattern generated by this set of walk parameters. If we set all s_x to zero in this set, pure side walking is realized.

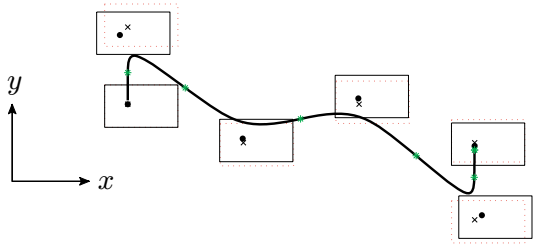


Fig. 4.27 Example of diagonal walk. To obtain a side step, s_y is changed for each step. $z_c = 0.8, T_{sup} = 0.8,\ a = 10, b = 1$.

4 Changing Walk Direction

For changing walk direction, we need to add heading information to our walk parameters. The direction change taking a step is denoted by s_θ as illustrated in Fig. 4.28[11].

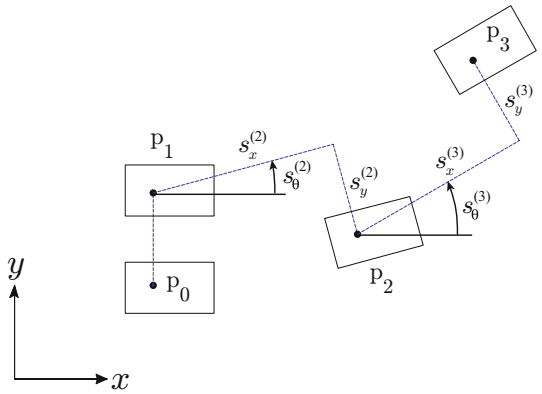


Fig. 4.28 Foot placement definition including change of direction. The heading direction s_θ is measured from x-axis in counter clock-wise direction.

The foot placement of the n-th step $(p_x^{(n)}, p_y^{(n)})$ is determined by

$$\begin{bmatrix} p_x^{(n)} \\ p_y^{(n)} \end{bmatrix} = \begin{bmatrix} p_x^{(n-1)} \\ p_y^{(n-1)} \end{bmatrix} + \begin{bmatrix} \cos s_\theta^{(n)} & -\sin s_\theta^{(n)} \\ \sin s_\theta^{(n)} & \cos s_\theta^{(n)} \end{bmatrix} \begin{bmatrix} s_x^{(n)} \\ -(-1)^n s_y^{(n)} \end{bmatrix}. \tag{4.60}$$

[11] This is a little bit simpler definition than the method we are using now. For example, this method cannot generate a stepping turn in place. Finding a better definition is an exercise for the readers.

The walk primitive for the n-th step is given by

$$\begin{bmatrix} \bar{x}^{(n)} \\ \bar{y}^{(n)} \end{bmatrix} = \begin{bmatrix} \cos s_\theta^{(n+1)} & -\sin s_\theta^{(n+1)} \\ \sin s_\theta^{(n+1)} & \cos s_\theta^{(n+1)} \end{bmatrix} \begin{bmatrix} s_x^{(n+1)}/2 \\ (-1)^n s_y^{(n+1)}/2 \end{bmatrix}. \tag{4.61}$$

The speed of walk primitive is also calculated as

$$\begin{bmatrix} \bar{v}_x^{(n)} \\ \bar{v}_y^{(n)} \end{bmatrix} = \begin{bmatrix} \cos s_\theta^{(n+1)} & -\sin s_\theta^{(n+1)} \\ \sin s_\theta^{(n+1)} & \cos s_\theta^{(n+1)} \end{bmatrix} \begin{bmatrix} (1+C)/(T_c S)\bar{x}^{(n)} \\ (C-1)/(T_c S)\bar{y}^{(n)} \end{bmatrix}. \tag{4.62}$$

One can generate a walking pattern with arbitrary heading control by using (4.60) instead of (4.50) in walking pattern generation algorithm Step 5, and by using (4.61) and (4.62) instead of (4.51) and (4.51) in the Step 6.

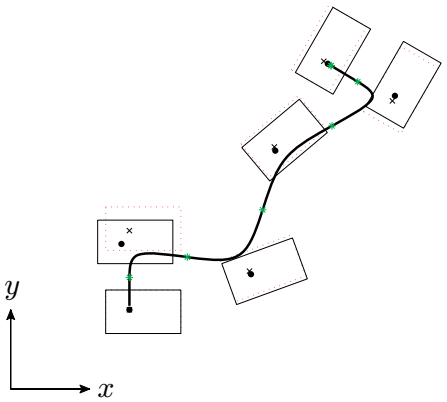


Fig. 4.29 Walk on an arc. 20 [deg] rotation per step. $z_c = 0.8, T_{sup} = 0.8$,weights $a = 10, b = 1$.

For example, to create a walk on an arc we can use the following parameters which specifies 20 [deg] rotation per one step.

n	1	2	3	4	5
s_x	0.0	0.25	0.25	0.25	0
s_y	0.2	0.2	0.2	0.2	0.2
s_θ	0	20	40	60	60

Figure 4.29 shows the walking trajectory generated from this walk parameter.

4.3.4 Introducing Double Support Phase

So far, we have been assuming that the support leg of the inverted pendulum model is exchanged instantaneously. However, such abrupt support

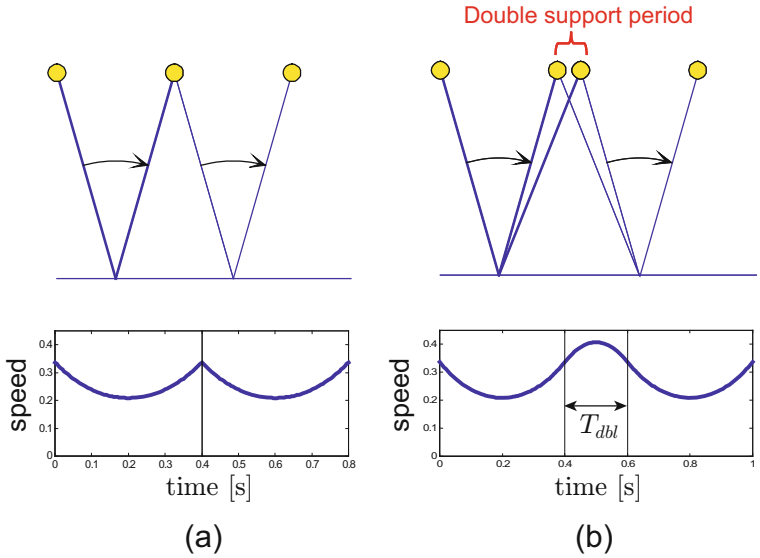


Fig. 4.30 Improvement of support exchange. (a) Instantaneous support exchange results bending point of CoM speed graph. CoM acceleration jumps from maximum to minimum. (b) By introducing double support phase, we can obtain continuous profiles of CoM speed and acceleration.

exchange results the horizontal acceleration jumps from maximum to minimum Fig. 4.30(a). As the result, the robot may suffer huge impacts and may possibly be damaged.

To obtain a smoother walking pattern which is suitable for real robots, a double support phase with a predetermined period T_{dbl} is inserted in the moment of support leg exchange.

What we need is smooth velocity profiles without sudden changes of slope. By such motion, we can avoid discontinuous changes of acceleration, thus the ZMP smoothly transfers from the former support foot to the new support. For this purpose, we generate velocity profiles using third order polynomials so that we can specify the speeds and accelerations at the beginning and the end of a double support period. As the result, the position of CoM is described by forth order polynomials (Fig. 4.30(b)),

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4. \tag{4.63}$$

The coefficients $a_0 \ldots a_4$ are determined by position, velocity and acceleration of the CoM at the instant of support exchange.

By inserting double supports, a robot takes larger step than planned. This can be compensated by a proportional reduction of walk primitives in advance. Figure 4.31 shows an example of a walking trajectory with a double support phase.

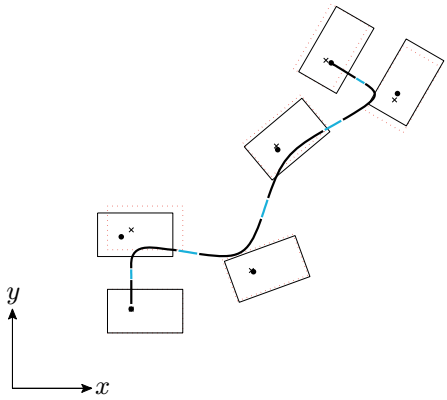


Fig. 4.31 A walking trajectory with double support periods. The CoM trajectories while double support phase are indicated by gray lines. The same parameters as were used in Fig. 4.29 are used. $z_c = 0.8, T_{sup} = 0.7, T_{dbl} = 0.1$,weights $a = 10, b = 1$.

It should be noted that while longer period of double support results smoother support exchange, it also requires undesirable quicker swing leg motion. Therefore we have a trade-off in determining T_{dbl}.

4.3.5 From Linear Inverted Pendulum to Multi-body Model

The easiest way to generate a walking pattern by using the linear inverted pendulum is to let the pelvis link follow the CoM motion of LIP. First, the real position of the CoM is calculated using a multi-body model and its position with respect to the pelvis frame is determined. After that, the position of the pelvis link is directly determined from the liner inverted pendulum assuming that the relative position of the CoM is kept constant with respect to the pelvis. In addition, we must calculate the swing foot trajectory so that it arrives the desired foot place at the specified time of touchdown.

Once we determine the trajectories for the pelvis and the both feet, the leg joint angles can be obtained by inverse kinematics as explained in Chapter 2.

This method is based on an assumption that the multi-body dynamics of the robot can be approximated by a simple inverted pendulum and its validity can be confirmed by using ZMP described in the former chapter. By calculating ZMP using multi-body model, we can evaluate the effects of swing leg reaction and errors in CoM position which were neglected in a linear inverted pendulum. Figure 4.32 shows two ZMPs, one based on the linear inverted pendulum, and one based on multi-body dynamics and the proposed

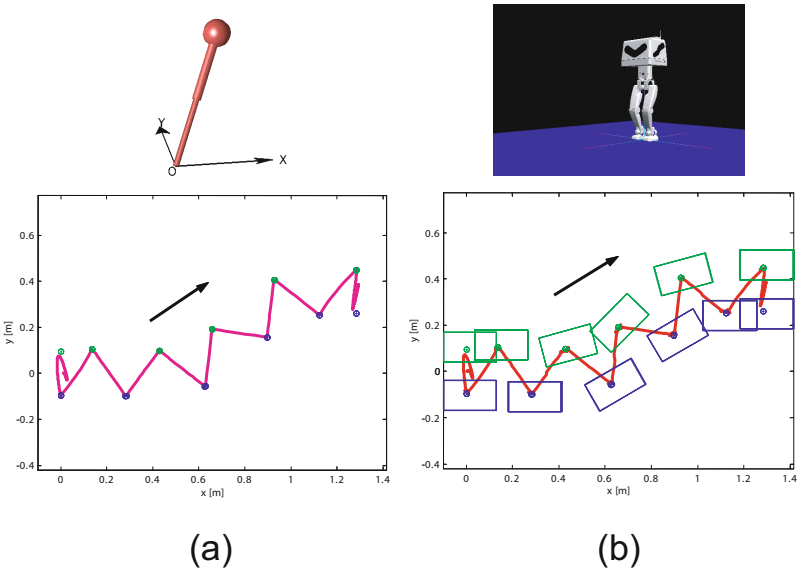


Fig. 4.32 Comparison of ZMP trajectory (a) ZMP calculated from 3D linear inverted pendulum model (b) ZMP calculated from multi-body dynamics whose pelvis link moves as 3D-LIP

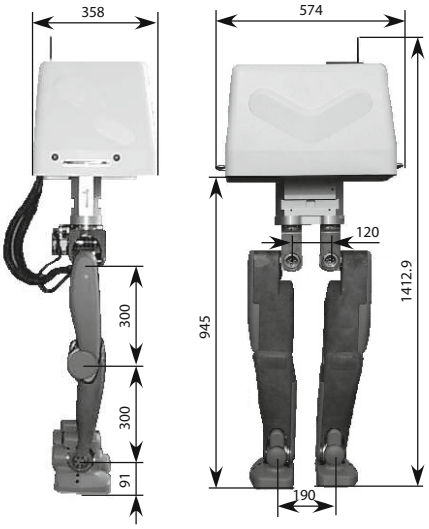


Fig. 4.33 Biped robot HRP-2L

pattern generation. Both of ZMPs are sufficiently close, hence we can conclude a multi-body dynamics can be simplified as a simple inverted pendulum in this case.

4.3.6 Implementation Example

Let us see an implementation of the proposed walking pattern generation. Figure 4.33 shows a biped robot HRP-2L which was developed in "Humanoid Robotics Project"(HRP). This robot was built to evaluate the leg part of HRP-2, the humanoid robot which was the final goal of the project. Each leg has six degrees of freedom and the robot is equipped with a Pentium II

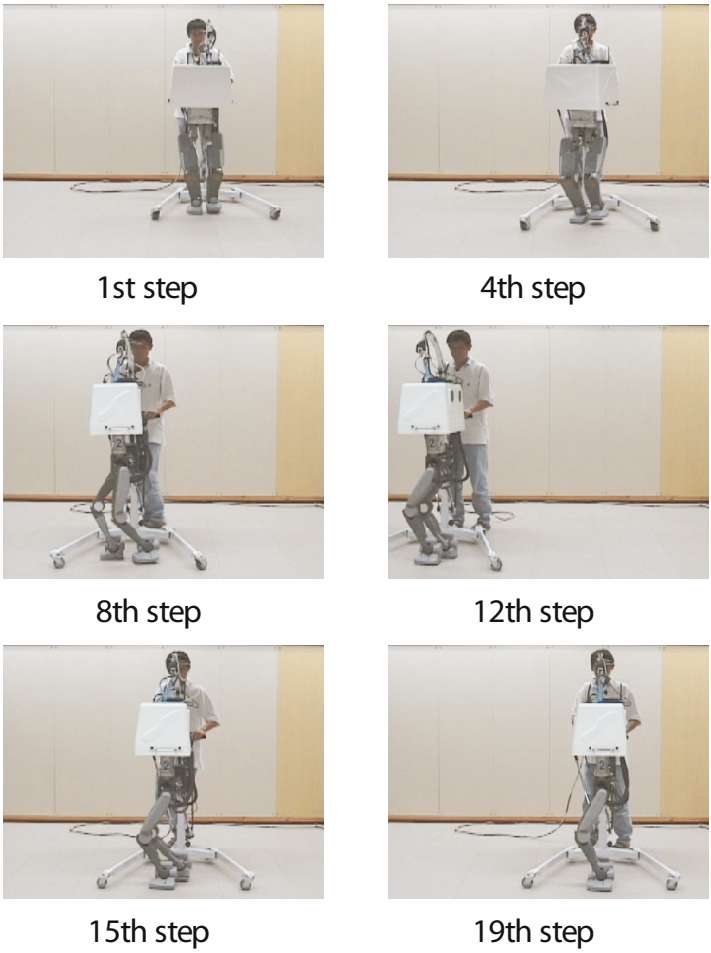


Fig. 4.34 Snapshots of real-time walking control

933MHz based on-board computer on its body part. The total weight is 58.2 [kg] including batteries of 11.4 [kg] and dummy weights of 22.6 [kg] which emulates upper body.

The algorithm of Fig. 4.25 can generate a walking pattern where at least two future steps were given. So we could build a walking control system which allows real-time step modification by specifying the walk parameter of two steps in future (s_x, s_y, s_θ) with a joystick. Figure 4.34 shows snapshots of our experiment of real-time walking control.

4.4 ZMP Based Walking Pattern Generation

4.4.1 Cart-Table Model

Let us think about a new model illustrated in Fig. 4.35. Here, a cart with mass M runs on a table whose mass is negligibly small. Although the table foot is too small to keep balance having a cart on the edge of the table, it can still keep an instantaneous balance if the cart runs with certain acceleration. We call this a *cart-table model*.

Since a cart-table model corresponds the case of a single mass at constant height in section 3.5.2, the ZMP is given as

$$p = x - \frac{z_c}{g}\ddot{x}. \tag{4.64}$$

We call this equation a *ZMP equation*.

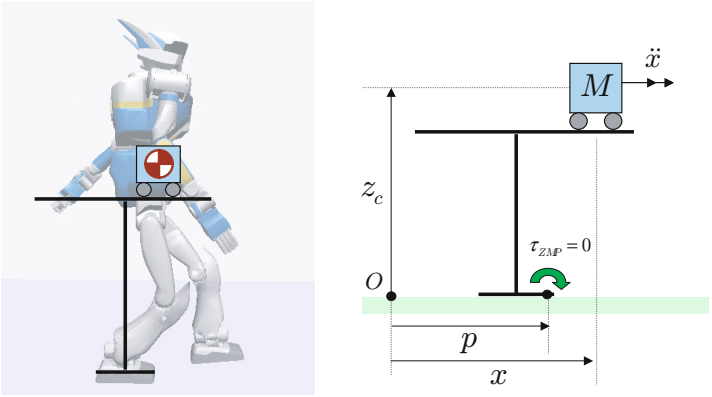


Fig. 4.35 Cart-table model: Dynamics of walking robot is approximated as a cart running on a massless table. The state of the running cart determines the center of pressure which acts from the floor, in other words, the cart changes ZMP.

On the other hand, the equation of linear inverted pendulum was given as following (Fig. 4.24).

$$\ddot{x} = \frac{g}{z_c}(x - p). \tag{4.65}$$

By regarding p as ZMP and not a foot place point as we did previously, we can treat a robot applying ankle torque and a robot in double support phase in a unified manner [134]. Moreover, we can see that (4.64) and (4.65) are the same equations with different outlooks.

A linear inverted pendulum model and a cart-table model are compared in Fig. 4.36. In a linear inverted pendulum model, the CoM motion is generated by the ZMP (Fig. 4.36(a)), and in a cart-table model, the ZMP is generated by the CoM motion (Fig. 4.36(b)). Therefore, these two models have opposite input-output causality.

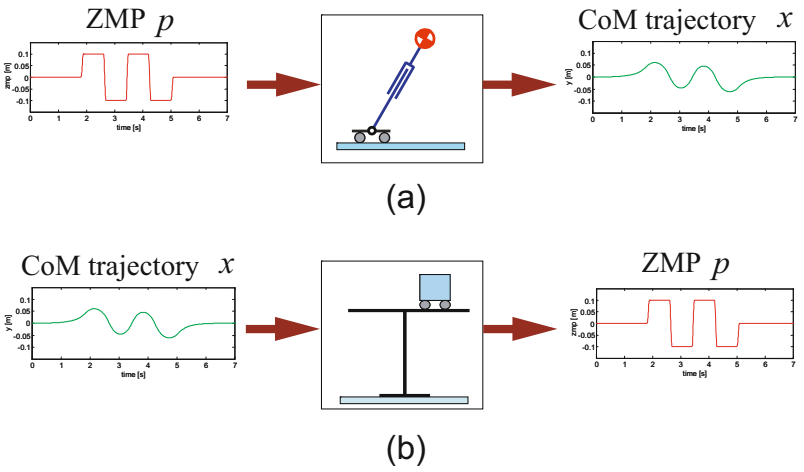


Fig. 4.36 Comparison of two models for relationship between ZMP and CoM (a) A linear inverted pendulum model inputs ZMP and outputs CoM motion. (b) A cart-table model inputs CoM motion and outputs ZMP.

As we described in the former section, a method based on a linear inverted pendulum assumes input-output relationship of Fig. 4.36(a) and the walking pattern is calculated in the following process.

(Specify target CoM motion) ⇒ (Calculate appropriate ZMP)

In this case, it is difficult to plan ZMP as expected. Indeed, we have modified the ZMP (support foot placement) in the method of the previous section.

Now, let us consider a walking pattern generation based on a cart-table model. In this case, assuming the causal relationship of Fig. 4.36(b), we calculate a walking pattern by the following manner[12].

(Specify target ZMP trajectory) ⇒ (Calculate appropriate CoM motion)

As the result, we can obtain a walking pattern which realizes the specified ZMP trajectory. Let us call such a method *ZMP based walking pattern generation.*

4.4.2 Off-Line Walking Pattern Generation

ZMP based walking pattern generation was first proposed by Vukobratović and Stepanenko in their paper published in 1972 [90], but their algorithm takes considerable computation time. Then, Takanishi et al. proposed a practical method which transforms the target ZMP pattern into a Fourier series by using FFT, solve the ZMP equation (4.64) in the frequency domain and obtains the CoM trajectory by using inverse FFT [11][13]. A pattern generator based on this method played a particularly important role in the early stage of the Humanoid Robotics Project.

In this section, we introduce a fast and efficient algorithm that was recently proposed by Nishiwaki et al. [114][14] Let us discretize the ZMP equation with a sampling time Δt. For this purpose, the acceleration $\ddot{x}$ is approximated as

$$\ddot{x}_i = \frac{x_{i-1} - 2x_i + x_{i+1}}{\Delta t^2}, \tag{4.66}$$

where $x_i \equiv x(i\Delta t)$. Using this approximation, the discretized ZMP equation is

$$\begin{aligned} p_i &= ax_{i-1} + bx_i + cx_{i+1}, \\ a_i &\equiv -z_c/(g\Delta t^2), \\ b_i &\equiv 2z_c/(g\Delta t^2) + 1, \\ c_i &\equiv -z_c/(g\Delta t^2). \end{aligned} \tag{4.67}$$

Putting the equations (4.67) in a column for the period of the specified $(1 \ldots N)$, and representing them as a single matrix equation gives

[12] There exist an infinite numbers of possible CoM motions which realize the given ZMP trajectory, however, almost all of them suffer divergence. Fig. 4.36(b) can be regarded as a mechanism which guarantees an executable solution.

[13] Later, Takanishi's method was extended to handle real-time pattern generation [40].

[14] Another fast and efficient method was proposed by Nagasaka [93].

$$\begin{bmatrix} p'_1 \\ p_2 \\ \vdots \\ p_{N-1} \\ p'_N \end{bmatrix} = \begin{bmatrix} a_1+b_1 & c_1 & 0 & & \\ a_2 & b_2 & c_2 & \ddots & \\ & & \ddots & & \\ & & & \ddots & a_{N-1} \; b_{N-1} \; c_{N-1} \\ & & & & 0 \quad a_N \quad b_N+c_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}, \tag{4.68}$$

where p'_1, p'_N are specified by using initial and terminal velocities v_1, v_N as

$$p'_1 = p_1 + a_1 v_1 \Delta t, \quad p'_N = p_N - c_N v_N \Delta t.$$

Rewriting (4.68) as

$$\boldsymbol{p} = \boldsymbol{A}\boldsymbol{x}$$

gives the representation of the solution by

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{p}. \tag{4.69}$$

Although $\boldsymbol{A}$ is a huge square matrix with several thousands columns and several thousands lows, there exists an efficient algorithm to compute the inverse [136] since it is a tridiagonal matrix whose elements are all zeros except its main diagonal, the adjacent diagonals above and below it.

From this CoM trajectory, we can generate a walking pattern for a multi-body model by using the method of Section 4.3.5. Then we can calculate the ZMP trajectory for the multi-body model.

$$\boldsymbol{p}^* = RealZMP(\boldsymbol{x}). \tag{4.70}$$

The function $RealZMP()$ calculates ZMP based on a multi-body model and $\boldsymbol{p}^*$ is the obtained ZMP. The ZMP error $\boldsymbol{p}^* - \boldsymbol{p}^d$ contains information about the difference between the cart-table model and the multi-body model. Again using (4.69), we can calculate the CoM variation to compensate the ZMP error

$$\Delta\boldsymbol{x} = \boldsymbol{A}^{-1}(\boldsymbol{p}^* - \boldsymbol{p}^d).$$

The CoM trajectory is updated by

$$\boldsymbol{x} := \boldsymbol{x} - \Delta\boldsymbol{x}.$$

Going back to (4.70), we can repeat the same process until the ZMP error becomes sufficiently small.

This is a very efficient algorithm. According to Nishiwaki et al. [73] it takes only 140 [ms] in generating a walking pattern for three steps (3.2 s) of the humanoid robot H7 [113] which have 32 DOF using dual Pentium II 750MHz. They have realized joystick controlled real-time walking by generating three future steps at every step cycle and by properly connecting them.

4.4.3 On-Line Walking Pattern Generation

In this section, we introduce the pattern generation method which is currently used for our humanoid robot HRP-2.

1 ZMP Tracking Control

Regarding a cart-table model as a dynamical system, one can imagine a servo system which realize a target ZMP tracking by feedback control as shown in Fig. 4.37.

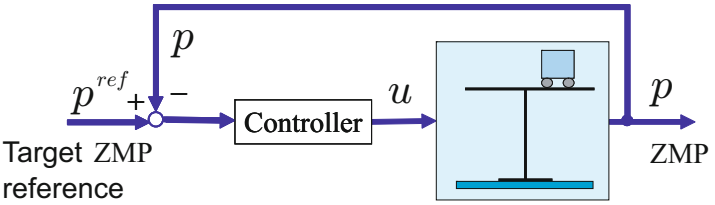


Fig. 4.37 Servo controller to track the target ZMP

We define a derivative of cart acceleration (jerk) as a system input to treat a cart-table model in a framework of the standard modern control theory,

$$u = \dddot{x}.$$

By using this input, we can rewrite the ZMP equation (4.64) into the following *state space representation*

$$\begin{aligned} \tfrac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \\ p &= \begin{bmatrix} 1 & 0 & -\frac{z_c}{g} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}. \end{aligned} \tag{4.71}$$

Starting from this equation, the modern control theory gives us a systematic way to design a controller [55]. However, we cannot obtain an appropriate walking pattern by using such controllers. For an example, suppose the walking motion of Fig. 4.38(a) which specifies a robot to walk forward one step of 30 cm. The target ZMP have step-like change at 1.5 s, but keeps constant before and after it. Notice that the CoM motion starts before the change of the ZMP. This means that the cart must move before the change of input in the system of Fig. 4.37. On the other hand, in an ordinary servo system, we have the output motion with a certain delay after a change of reference as in

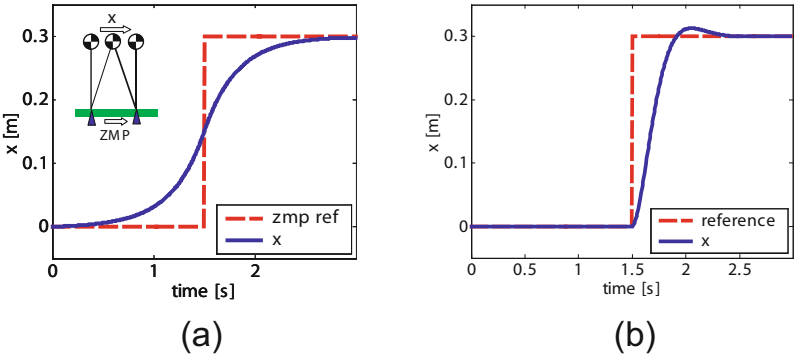


Fig. 4.38 Input-output causality (a) One step walking. ZMP (broken line), CoM (bold line). The CoM starts moving before the step change of the ZMP. (b) In an ordinal servo system, the output (bold line) moves after the change of reference input (broken line).

Fig. 4.37(b) and this is normal causality. In biped walking pattern generation, a future information must go back to and affect the past!

2 Preview Control System

Although we need future information, we do not have to develop a time-machine[15]. For example, when driving a car, we are always monitoring the road ahead and using information about the future location of the car for steering. This can be regarded as using future information for smooth driving. To appreciate the importance of using such future information, imagine driving at a high speed on a freeway with the top half of the windshield obscured, so you can only see a few meters ahead.

A method which utilizes future information is called *preview control* [128, 82, 86, 130]. Let us introduce a controller design based on this theory.

At the beginning we discretize a continuous-time system of (4.71) using a sample time of Δt to design a digital controller

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{b}u_k \\ p_k = \boldsymbol{c}\boldsymbol{x}_k \end{cases} \tag{4.72}$$

where

$$\begin{aligned} \boldsymbol{x}_k &\equiv [\, x(k\Delta t)\ \dot{x}(k\Delta t)\ \ddot{x}(k\Delta t)\,]^T, \\ u_k &\equiv u(k\Delta t), \\ p_k &\equiv p(k\Delta t), \end{aligned}$$

[15] By the way, a realistic time-machine technology which can send a message to the past using particle physics is depicted in James P. Hogans's *Thrice upon a Time* (Baen Books). This is a master-piece Sci-Fi novel.

and

$$\boldsymbol{A} \equiv \begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{b} \equiv \begin{bmatrix} \Delta t^3/6 \\ \Delta t^2/2 \\ \Delta t \end{bmatrix},$$
$$\boldsymbol{c} \equiv [\,1\ 0\ -z_c/g\,].$$

To let the system output p_k follow the target ZMP p_k^{ref} as closely as possible, we consider the problem to minimize the following performance

$$J = \sum_{j=1}^{\infty} \{Q(p_j^{ref} - p_j)^2 + Ru_j^2\}, \tag{4.73}$$

where Q, R are positive weights. This is called a *tracking control problem*. According to the preview control theory, this performance index J can be minimized by the following input which uses the future target references up to N steps [54].

$$u_k = -\boldsymbol{K}x_k + [f_1, f_2, \cdots f_N] \begin{bmatrix} p_{k+1}^{ref} \\ \vdots \\ p_{k+N}^{ref} \end{bmatrix}, \tag{4.74}$$

where

$$\begin{aligned} \boldsymbol{K} &\equiv (R + \boldsymbol{b}^T\boldsymbol{P}\boldsymbol{b})^{-1}\boldsymbol{b}^T\boldsymbol{P}\boldsymbol{A} \\ f_i &\equiv (R + \boldsymbol{b}^T\boldsymbol{P}\boldsymbol{b})^{-1}\boldsymbol{b}^T(\boldsymbol{A} - \boldsymbol{b}\boldsymbol{K})^{T*(i-1)}\boldsymbol{c}^T Q. \end{aligned} \tag{4.75}$$

The matrix $\boldsymbol{P}$ is a solution of

$$\boldsymbol{P} = \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{A} + \boldsymbol{c}^T Q\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{b}(R + \boldsymbol{b}^T\boldsymbol{P}\boldsymbol{b})^{-1}\boldsymbol{b}^T\boldsymbol{P}\boldsymbol{A} \tag{4.76}$$

which is called a Riccati Equation[16].

By observing (4.74), we can see that a preview controller consists of a state feedback (the first term of right hand side) and a feed-forward of a inner product between the future target reference up to N steps and the weights $[f_1, \ldots, f_N]$ (the second term).

3 Improvement of Preview Controller

We observed an offset tracking error of ZMP in a long distance walking pattern generated by (4.74). To solve this problem, we rewrote (4.72) into the following expanded form:

[16] You do not have to worry about this complicated matrix equation. By using a command **dlqr** of Matlab Control System Toolbox or GNU Octave, you will immediately obtain the numerical solution of $\boldsymbol{P}$ and $\boldsymbol{K}$.

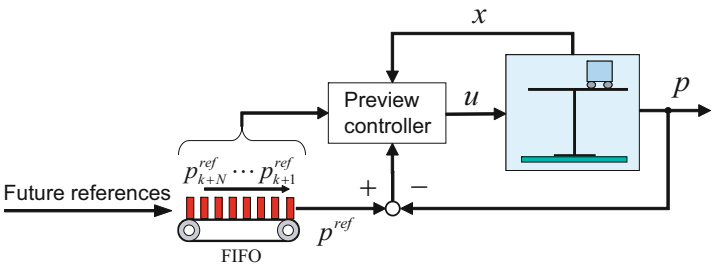


Fig. 4.39 Walking pattern generation by preview control

$$\begin{cases} \boldsymbol{x}^*_{k+1} = \tilde{\boldsymbol{A}}\boldsymbol{x}^*_k + \tilde{\boldsymbol{b}}\Delta u_k \\ p_k = \tilde{\boldsymbol{c}}\boldsymbol{x}^*_k \end{cases}, \tag{4.77}$$

where the new input and state vector were taken as

$$\Delta u_k \equiv u_k - u_{k-1}, \quad \Delta \boldsymbol{x}_k \equiv \boldsymbol{x}_k - \boldsymbol{x}_{k-1},$$
$$\boldsymbol{x}^*_k \equiv \begin{bmatrix} p_k \\ \Delta \boldsymbol{x}_k \end{bmatrix}.$$

The matrices are

$$\tilde{\boldsymbol{A}} \equiv \begin{bmatrix} 1 & \boldsymbol{cA} \\ \boldsymbol{0} & \boldsymbol{A} \end{bmatrix}, \quad \tilde{\boldsymbol{b}} \equiv \begin{bmatrix} \boldsymbol{cb} \\ \boldsymbol{b} \end{bmatrix},$$
$$\tilde{\boldsymbol{c}} \equiv [1\ 0\ 0\ 0].$$

Let us design a controller for the system of (4.77) to minimize the following performance

$$J = \sum_{j=k}^{\infty} \{Q(p^{ref}_j - p_j)^2 + R\Delta u_j^2\}. \tag{4.78}$$

The preview controller is

$$\Delta u_k = -\tilde{\boldsymbol{K}}\boldsymbol{x}^*_k + \sum_{j=1}^{N} \tilde{f}_j p^{ref}_{k+j} \tag{4.79}$$

where $\tilde{\boldsymbol{K}}, \tilde{f}_j$ are gains obtained by substituting $\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}, \tilde{\boldsymbol{c}}, Q$ and R into (4.75) and (4.76). By summing up equations (4.79) for $k = 1, \ldots, N$, we can obtain a preview controller for the original system of (4.72):

$$u_k = -K_s \sum_{i=0}^{k} (p_j^{ref} - p_j) - K_x \boldsymbol{x}_k + \sum_{j=1}^{N} g_j p_{k+j}^{ref} \tag{4.80}$$

$$\begin{bmatrix} K_s \\ \boldsymbol{K}_x \end{bmatrix} \equiv \tilde{\boldsymbol{K}}, \quad g_j := \sum_{i=j}^{N+j} \tilde{f}_i.$$

A block diagram for a pattern generation based on preview control is illustrated in Fig. 4.39. The future target ZMP reference is stored in a FIFO (First-In-First-Out) buffer visualized as a belt conveyer, and its output value is regarded as the *current* reference. The preview controller calculates the control input using the ZMP reference on the FIFO buffer and the state of the cart. The cart state $x, \dot{x}$ is the result of the pattern generation, the CoM motion which satisfies the target ZMP.

The CoM trajectory calculated by the proposed method and the resulted ZMP are shown in Fig. 4.40. The upper graph is the motion along the walking

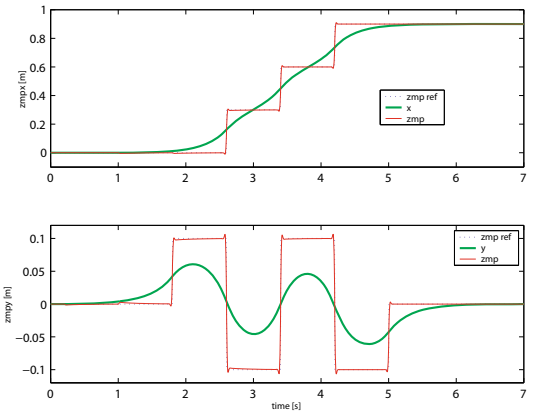


Fig. 4.40 CoM trajectory obtained by preview control

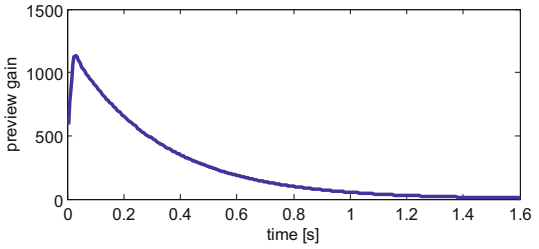


Fig. 4.41 Preview control gain g_j

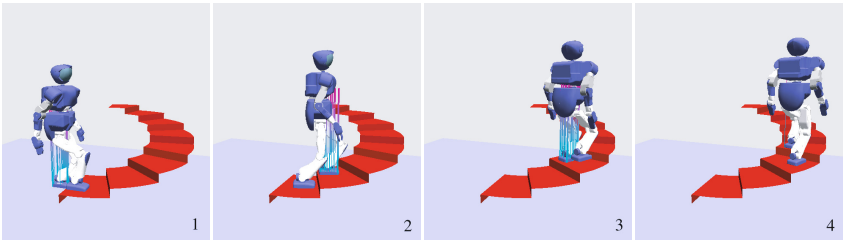


Fig. 4.42 Walk simulation on spiral stairs

direction and the lower graph is for the lateral direction[17], and we can see the proper CoM motions are generated for step-like target ZMP and pulse like target ZMP, respectively. The preview gain used for this calculation is shown in Fig. 4.41. Since the preview gain becomes negligibly small at 1.6 s in this graph, we can conclude that the information more than 1.6 s future ahead will not contribute the control performance.

Figure 4.42 shows a walking pattern on spiral stairs using the above explained method.

4.4.4 Dynamics Filter Based on Preview Control

1 Structure of a Dynamics Filter

Since a preview control based pattern generation depends on a cart-table model, it does not guarantee the stability of the motion which cannot be represented by the simplified model, for example, significantly changing upper body posture while walking. In such a case, however, we can use a cart-table model as an error system around the target motion, and can calculate the modification of the CoM trajectory to compensate for the ZMP error by using preview control again.

Figure 4.43 shows the entire structure of the proposed system. The inputs are the target ZMP (ZMP^{ref}) and the whole state of the robot*Robot state*, which consists of the angles and the speeds of all the joints, the pelvis configuration, its speed and angular velocity. We can calculate the ZMP error (ΔZMP) from these inputs, and put it into the FIFO buffer. Also put *Robot state* into another FIFO. By retrieving Δ ZMP from the FIFO output after a certain time delay, we have the future ZMP error with respect to the delayed moment on the FIFO buffer. Thus, we can calculate a proper compensation of CoM by using preview control. By applying the CoM compensation to the delayed *Robot state*, we obtain improved joint trajectories which better satisfies the target ZMP reference. Generally, a system of Fig. 4.43 is called

[17] A plane made of the heading axis and the vertical axis is called sagittal plane and a plane made of the lateral axis and the vertical axis is called lateral plane.

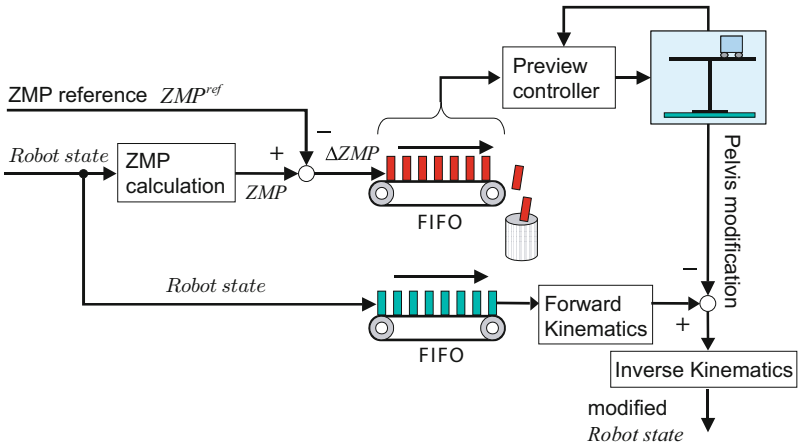


Fig. 4.43 Structure of a dynamics filter based on preview control

a *dynamics filter*. A dynamics filter converts a given motion pattern into an improved one which satisfies the desired properties [93].

2 Evaluation of the Dynamics Filter

As an example, let us make a pattern where HRP-2 performs a squat in the middle of walking and goes back to normal walking after that, as shown in Fig. 4.44. First, we simply add a squatting motion to a walking pattern assuming constant height of CoM and the resulted ZMP trajectory is shown in Fig. 4.45(a). It is observed that the ZMP (bold line) approaches the boundary of the support polygon (dotted lines) and the walking pattern has very small stability margin. Therefore, the robot falls down immediately after the squatting motion in the dynamic simulation.

This walking pattern was modified by the dynamics filter of Fig. 4.43 and we obtained the pattern of Fig. 4.45(b). We used the delay time of 0.8 s for the FIFO buffers and a preview controller without integrator (see (4.74)). The maximum absolute ZMP error which was originally 0.11 m decreased to 0.05 m by applying the dynamics filter. Since the new walking pattern has

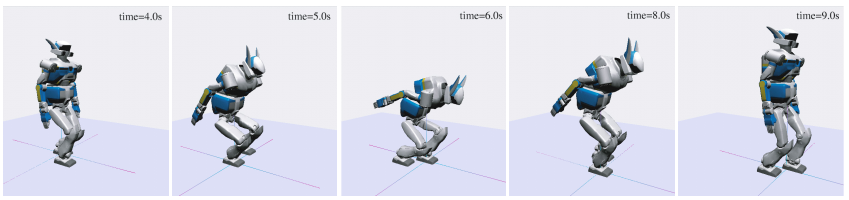
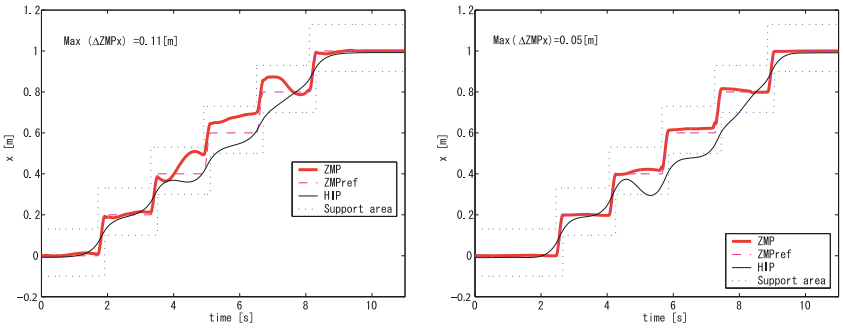


Fig. 4.44 Example walking pattern obtained preview control based dynamics filter



(a) Without dynamics filter (b) With dynamics filter

Fig. 4.45 Effects of the dynamics filter: ZMP(bold lines) is modified to be inside of support polygon boundary(dotted line) thus sufficient stability margin is acquired

enough stability margin, the robot could successfully walk in the dynamic simulation.

The modified walking pattern is shown in Fig. 4.44. By applying the proposed dynamics filter, we can obtain a realizable walking pattern even though the original pattern largely deviated from a simple cart-table model dynamics.

4.4.5 Advanced Pattern Generators

The method of preview control is not the only way for online walking pattern generation. As one of the practical methods, Harada et al. proposed to use an analytical solution of the ZMP equation [60]. Later, this was improved by Morisawa et al. for more efficient and responsive pattern generation [85]. These methods were used to realize HRP-2's reactive walking.

Preview control belongs the general scheme called Model Predictive Control (MPC), which calculates the control input by optimizing the future trajectory. Based on MPC, Wieber proposed a walking pattern generation which does not require a prescribed ZMP [137, 9]. By this method, ZMP and CoM trajectories can be simultaneously generated from the given profiles of the support polygon.

4.5 Stabilizer

Figure 4.46 shows snapshots of HRP-2 walking on an uneven floor. As explained in the beginning of this chapter, in a real environment which contains unmodeled errors and floor unevenness, a biped robot following a prepared pattern suffers a rapid evolution of errors between the reference and the actual state and falls down in a few steps. To suppress the error development and guarantee the walking motion along the specified walking pattern we use a stabilizer.

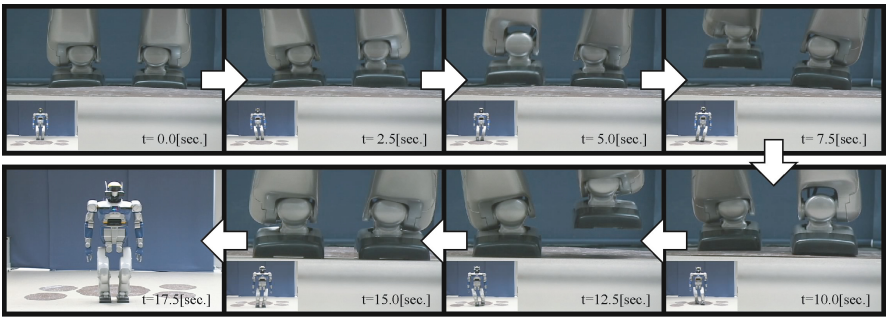


Fig. 4.46 Feet of HRP-2 walking on an uneven floor

4.5.1 Principles of Stabilizing Control

In this section, we introduce basic principles which are used to construct a stabilizer. To simplify the descriptions, we explain a method to stabilize around a state of standstill. Since a stabilizer works to absorb a small errors around the reference walking pattern, we should not lose generality with this approach.

1 Control by an Ankle Torque

We can stabilize whole body balance or posture by using the ankle torque of the support leg by modeling the whole robot as a simple inverted pendulum. An ankle torque, τ and a linear inverted pendulum have a relationship of

$$\ddot{x} = \frac{g}{z_c}x + \frac{1}{Mz_c}\tau. \tag{4.81}$$

The simplest feedback law to stabilize this pendulum is as following[18]:

$$\tau = -k_p x - k_d \dot{x}, \tag{4.82}$$

where k_p, k_d are the feedback gains determined for given response frequency ω and damping coefficient ζ,

$$k_p \equiv M(z_c\omega_n^2 + g), \quad k_d \equiv 2Mz_c\zeta\omega_n.$$

Although this control law looks easy, its implementation is difficult. First, accurate control of ankle torque is an extremely difficult problem for most walking robots equipped with high-reduction gears. Moreover, to realize

[18] For a use of more advanced control law, one must consider the problem of integrator windup, an unexpected accumulation of the integrator while the ankle torque saturates.

stable walking, we need fine tuning of feedback gains and torque limiters to suppress the toe/heel lift-off caused by excessive ankle torque.

This method was used by many biped robots developed in 1980s and 1990s, for example, WL-10RD by Takanishi et al. [10], Idaten II by Miyazaki and Arimoto [109, 32], Kenkyaku-2 by Sano and Furuhso [46] and Meltran II by Kajita and Tani [116].

2 Control by Modifying Foot Placements

The second method based on a inverted pendulum model is to modify foot placements for the stabilization. For this control, we can apply the same principle of the walking speed control explained in section 4.3.3(see Fig.4.23) on page 129). There are few robots which were stabilized by foot place modification, such as BIPER-3, a stilt walker developed by Shimoyama and Miura [41], and the series of hopping robots by Raibert and his colleagues [97].

3 ZMP Control by CoM Acceleration

Let think about a stabilization method based on a cart-table model. In this case we must measure the ZMP to design a feedback controller, and by assuming the time constant of the ZMP sensor as T, the ZMP equation is

$$p = \frac{1}{1+sT}(x - \frac{z_c}{g}\ddot{x}). \tag{4.83}$$

The state space representation having the cart acceleration $\ddot{x}$ as the input is

$$\frac{d}{dt}\begin{bmatrix} p \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -1/T & 1/T & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} p \\ x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} -z_c/(gT) \\ 0 \\ 1 \end{bmatrix}\ddot{x}. \tag{4.84}$$

We can design a state feedback law to stabilize this system as

$$\ddot{x} = -k_1 p - k_2 x - k_3 \dot{x}. \tag{4.85}$$

The feedback gains k_1, k_2, k_3 can be determined by a standard control theory like a pole placement or the LQ optimal control. This control law was introduced as *torso position compliance control* by Nagasaka, Inaba and Inoue [70][19]. This control law is effective for a walking robot with feet of high stiffness and was used to stabilize humanoid robots H5 and H7[113]. A simpler but effective controller is proposed by Choi, Kim and You [139].

As an alternative method, Okada, Furuta and Tomiyama proposed a stabilization method based on the CoM acceleration control via dynamic change of

[19] In the original work, they used the different procedure to derive the equivalent control law. Napoleon et al. shows another interpretation for their control law in terms of the zero-dynamics control theory [94].

sampling time and implemented to control the biped robot MK.3 and the humanoid robot morph3 [143].

4 Body Posture Control by Crotch Joints

For most walking robots, we desire that the body maintain an upright posture during walking. The easiest way is to rotate the crotch(hip) joints so that the body keeps the desired state based on a posture sensor readout. This is possible even for point contact feet since the torque around the crotch joint is generated by the friction force on the ground. This control is implemented in the Raibert's hopping robots [97], and a humanoid robot developed by Kumagai et al. [84].

5 Model ZMP Control

As a novel method of body posture control, Hirose, Takenaka et al. proposed a *model ZMP control*. According to their explanation, it works as

> *When the body of the real robot was inclined more forward than the model, the model body is more strongly accelerated than the planned trajectory. This changes the target inertial force and the target ZMP then goes more backward than the original ZMP, hence producing posture recovery in the real robot* [83, 62].

Let us consider the physical meaning of this model ZMP control using a version of cart-table model of Fig. 4.47. We assume the table foot has a free rotating joint which inclines the table at an angle of θ relative to the vertical position. The cart acceleration $\ddot{x}$ corresponds to the body acceleration.

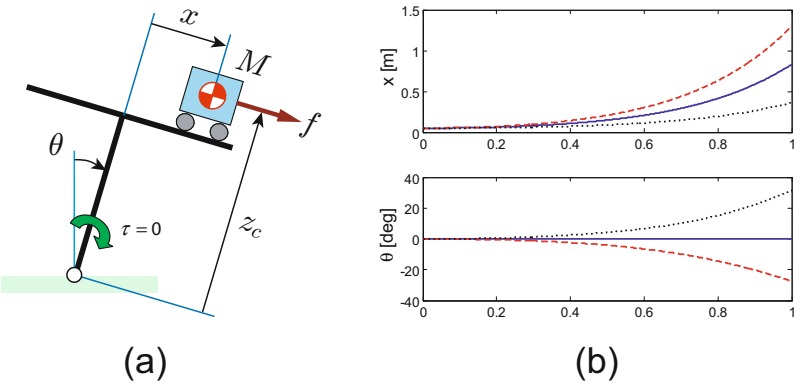


Fig. 4.47 (a) A cart-table model with free rotating joint. θ is positive in the clockwise direction. (b) Cart acceleration and the table inclination. Bold lines: with proper acceleration the table keeps upright ($\ddot{\theta} = 0$), Broken lines: with excessive acceleration the table rises ($\ddot{\theta} < 0$), Dotted lines: with insufficient acceleration the table sinks ($\ddot{\theta} > 0$). The model ZMP control uses this phenomenon.

The equation of motion obtained by Lagrange's method is

$$\begin{cases} (x^2 + z_c^2)\ddot{\theta} + \ddot{x}z_c - g(z_c \sin\theta + x\cos\theta) + 2x\dot{x}\dot{\theta} = \tau/M \\ \ddot{x} + \ddot{z}_c - \dot{\theta}^2 x + g\sin\theta = f/M \end{cases}, \tag{4.86}$$

where τ is the torque acting at the table foot and f is the force to accelerate the cart on the table. By linearizing the first equation around $\theta, \dot{\theta} = 0$ and substituting $\tau = 0$, we obtain

$$(x^2 + z_c^2)\ddot{\theta} = gx + gz_c\theta - z_c\ddot{x}. \tag{4.87}$$

Next, assume the target position of the cart x_d is generated by the dynamics of

$$\ddot{x}_d = \frac{g}{z_c}(x_d - p_d), \tag{4.88}$$

where p_d is the target ZMP. Intuitively speaking, when the target ZMP is placed backword ($p_d < 0$) the cart accelerate strongly and when it is placed forward ($p_d > 0$) the cart acceleration becomes small. Since (4.88) is just a model to generate the acceleration, the target ZMP can be assigned at the outside of the support polygon.

Now, suppose we can control the cart acceleration as we want. By substituting $x = x_d$ and (4.88) into (4.87) we get

$$\ddot{\theta} = \frac{gz_c}{x_d^2 + z_c^2}\theta + \frac{g}{x_d^2 + z_c^2}\, p_d. \tag{4.89}$$

From this result, we can see that the table inclination θ can be controlled by the target ZMP p_d, and its dynamics is determined by the cart position and gravity acceleration.

6 Impact Absorption by Joint Backdrive

Embedding a joint torque sensor and applying the reference position as a function of the torque measurement, we can realize a virtual spring-damper system. Takanishi et al. utilized such system to absorb the vibration at touchdown impact [10]. Also when a joint has a small reduction ratio (1/1 to 1/50 approximately), it backdrives by the external forces from the environment thus a position control system behaves as a spring-damper by itself. Kenkyaku 1, a biped robot developed by Furusho et al. utilized this for its walking control [47]. Sorao, Murakami and Ohnishi reported an impact absorption at support exchange using an impedance controller which depends on a sensor-less torque measurement by *disturbance observer* [75].

7 Stabilization by LQ Control

A walking robot can be modeled as a multi-input-multi-output (MIMO) system having joint torques $\boldsymbol{u}$ as its input vector and output as the state of all

links. Let us define a state vector $\boldsymbol{x}$ consisting of all the posture and speed variables of all the links, and linearize the motion equation from $\boldsymbol{u}$ to $\boldsymbol{x}$. By a straightforward application of LQ control theory, we get a state feedback law of

$$\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x},$$

where $\boldsymbol{K}$ is a feedback gain which is a huge $N \times 2N$ matrix for the number of joint N. This approach was applied to a 6 DOF biped, CW-2 by Mita et al. [133] and 12 DOF biped developed by Yoshino [145]. Especially, the latter work reports that the robot could successfully walk at 3 [km/h] on the floor with unevenness of 6 [mm].

4.5.2 Stabilizing Control of Honda Humanoid Robot

A practical stabilizing control system is constructed by combining multiple control principles described above. Let review one such successful implementation.

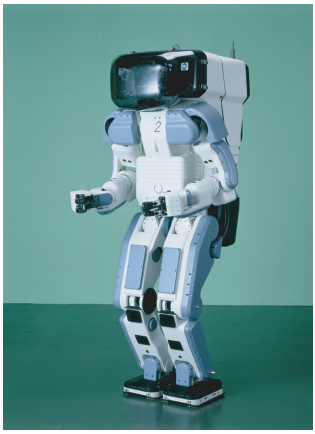


Fig. 4.48 Humanoid robot P2 (1996) (By courtesy of Honda Motor Co., Ltd.)

Figure 4.48 shows a humanoid robot P2 developed by Honda Motor Co., Ltd. and officially announced in 1996. Its control system has a state-of-art quality even at the time of this printing.[20] and its technology is well disclosed [83, 62]. The walking control system of P2 is illustrated in Fig. 4.49. The feedback path from the body inclination to the ground reaction force corresponds the ankle torque control described in **1**. It should be noted that the passive compliant elements inserted in the feet of P2 makes this control easier

[20] It is assumed that the same type of controllers are also used for another Honda humanoids P3 and ASIMO which are the successors of P2.

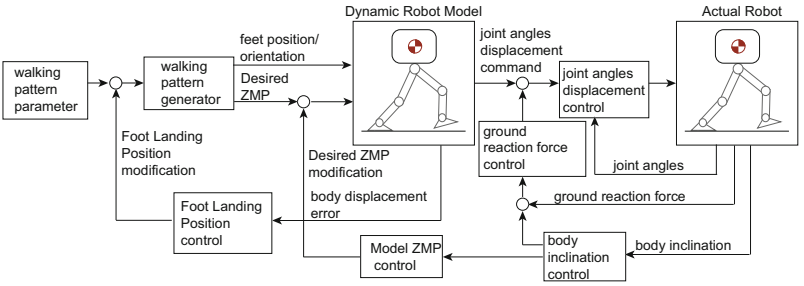


Fig. 4.49 The walking control system of Honda P2

to implement. When the robot body gets an excessive inclination the model ZMP control **5** works to resolve it. The horizontal body displacement error caused by this is corrected by the foot landing position control **2**. As we see, a robust walking control of P2 is supported by an ingenious combination of multiple control schemes.

4.5.3 Advanced Stabilizers

Recently, we proposed a new stabilizer based on a state space model of LIPM with ZMP control lag [117]. This stabilizer allows our new humanoid robot HRP-4C to walk on a certain level of uneven ground as well as to perform human-like walking with toe supporting [69]. One of the key issues of this stabilizer is a proper setting of the state feedback gain which was discussed by Sugihara [125]. Another important issue is distribution of the desired ZMP. A related concept of stabilization based on a total ground force distribution was discussed by Hyon [42].

A huge disturbance during walking, for example a kick on the body, can immediately cause the robot to deviate from a preplanned trajectory and consequently fall down. The robot can avoid falling even in this case, if it can re-plan the walking pattern by letting the deviated state be the initial condition. By properly implementing this concept, very robust walking can be realized [72, 85, 51].

As we discussed in Section 4.5.1, adaptive change of foot placement is a basic strategy for stabilization. Pratt et al. defined the **capture point** which is a point on the ground where the robot can step to achieve its complete stop [50]. Based on the capture point concept, Englsberger et al. proposed a simple framework for real-time walking pattern generation and stabilization [44].

4.6 Pioneers of Dynamic Biped Walking Technology

Let us look back in the research history of dynamic biped walking control. Since the dawn of the robotics research, biped walk was widely recognized as

a challenge of the highest difficulty. The earliest research to develop hardware were started by Ichiro Kato in Waseda University in 1966, and by D.C. Witt in Oxford University in 1968 [68, 19]. As we explained in chapter 1, in 1973, Kato and his colleagues built the world's first humanoid robot WABOT-1 which had two arms and two legs and was controlled by a computer. Although it was a remarkable achievement, WABOT-1 could only perform static walking. About 1980, there was a big trend of research to realize dynamic biped walking and many Japanese researchers actively developed theories and robot hardware. We can see those activities in *Special Issue: Biped Walking Robot*, the journal of Robotics Society of Japan, Vol.1, No.3. In 1986, there already exists many biped robots which could perform dynamic walking as shown in Fig. 4.50 [27].

In the same year, a group of researchers in Honda Motor Co.,Ltd. started a secret project of biped walking robot. After complete silence for 10 years[21], their efforts have borne fruit as the humanoid robot P2. The humanoid robot P2 suddenly appeared with an excellent hardware and walk control technology, and it depressed researches who have been working on biped walking control until that time. At the same time, it also inspired many researchers by demonstrating the great possibility of humanoid robots. This yielded the power to start the *Humanoid Robotics Project* (1998-2002), the national R&D project of Japan.

People who get used to watch ASIMO or QRIO might think the robots in Fig. 4.50 are primitive. However, the researchers who developed them by struggling with poor computers and weak actuators of these days are the real pioneers of biped walking technology. Indeed, most of the modeling and control technique for biped walking were developed until those days. Moreover, I can hardly believe that Honda's research project was independent from those many research works already published in 1986. The progress of science and technology is not done by a genius in one night. There is no brilliant breakthrough independent from the heritage. Therefore, the true progress can be done only by the open exchange of ideas in a collaboration of many researchers, beyond institutes, beyond countries.

4.7 Additional Methods for Biped Control

In this section, we introduce biped robots and controls based on methods totally different from the concepts so far explained.

[21] Compared with universities, it is estimated that more than 100 times the financial budget as well as manpower was invested by Honda during this time. Since the robotics community held a mostly negative view on biped robotics in those days, the long-range vision of the Honda people should be highly respected!

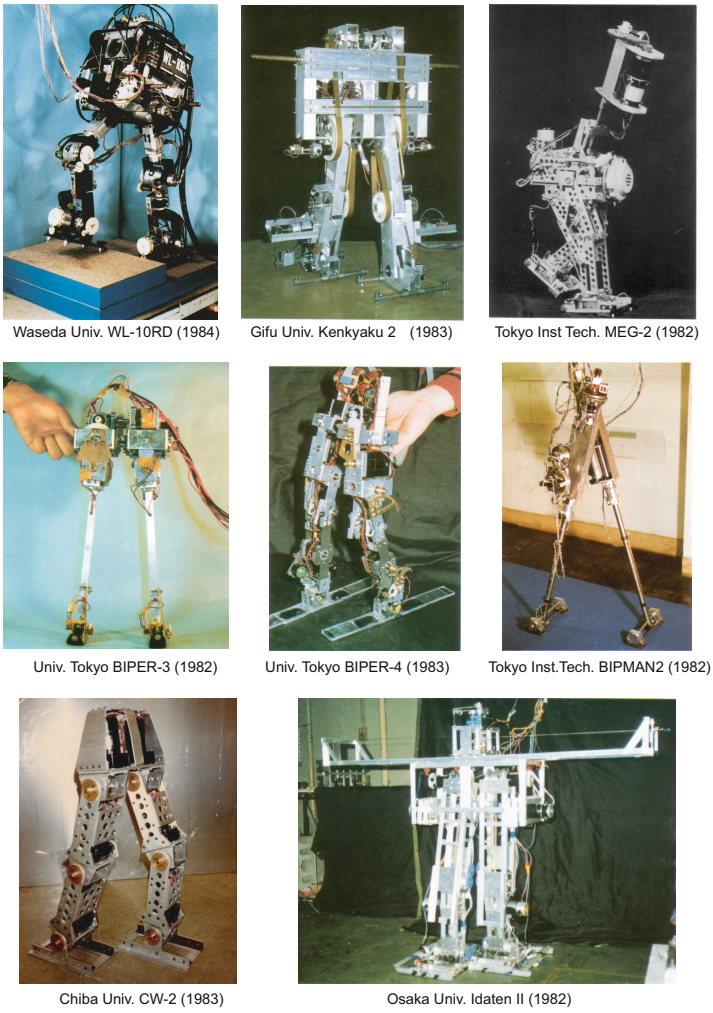


Fig. 4.50 Biped robots which could walk dynamically before 1986 [27]

4.7.1 Passive Dynamic Walk

Passive dynamic walkers are mechanisms which can walk down a shallow slope only using potential energy. Tad McGeer intensively analyzed its dynamics and designed a 2D walker with free knee joints which can perform stunningly human like walking without any actuation or control [131, 132]. Following and improving this concept, a 3D passive walker was also realized by Collins, Wisse and Ruina [111].

Passive dynamic walking has attracted many researchers who are seeking for the fundamental nature of biped locomotion. There are works analyzing its chaotic behavior [14, 74] and its robust stability [141].

In the original paper of McGeer [131], he suggested a "powered" passive walker, which can walk on a level ground or uphill slopes by adding small actuators. It is called a **semi-passive dynamic walker**. There exist theoretical works [7, 29, 91] and successful implementations were developed by Hobbelen and Wisse [21] and by Narioka, Tsugawa and Hosoda [71]. Collins, Ruina, Tedrake and Wisse have evaluated the power consumption of their successful semi-passive walkers and estimated that one of their robots is approximately ten times more efficient in comparison with Honda's ASIMO [110].

4.7.2 Nonlinear Oscillator and Central Pattern Generators

A group of researchers are considering that biped walking should not be analytically planned, but must be the result of nonlinear oscillations emerging from feedback and dynamic interaction between the system and the environment.

Katoh and Mori built a stilt type biped robot, BIPMAN2, which used a stable limit cycle generated by a nonlinear oscillator based on a coupled van der Pol equation. The robot could take one dynamic step forward [100].

Taga et al. simulated a human's muscle-born system with distributed nonlinear oscillators (Central Pattern Generators: CPGs) and found the simulated robot can naturally generate walking or running motions which are robust against disturbances [126].

Inspired from Taga's work, Hase et al. simulated detailed 3D human model controlled by a hierarchical CPG system and demonstrated stable 3D walking and running[61]. They also used a hill-climbing algorithm to optimize the CPG parameters for various performance indices and reported that walking patterns with human-like properties were successfully obtained.

Recently, Hyon, Morimoto and Kawato have demonstrated CPG-based dynamic walking by their human-sized humanoid robot [107].

4.7.3 Learning and Evolutionary Computing

The most radical approach might be to build a robot which can learn or evolve walking by itself. Doya built a simple biped robot consists of three links, and a self learning system of hardware in the loop. The system generated a walking pattern randomly and learned by using a hill-climbing algorithm using actual traveled distance as its performance index [57]. As the result, the robot could acquire a variety of walking patterns including a jumping gait and a tumbling gait which were not expected. de Garis designed a neural network based walking control system whose weights are tuned by a genetic algorithm, and simulated a evolution of biped walking [20].

Tedrake et al. designed a 3D semi-passive walker equipped with four actuators whose motion can be acquired by online reinforcement learning. It is reported that the robot could learn an adequate walking pattern for the various floor conditions within about twenty minutes [106].