# Walking Pattern Simulation based on Virtual Reality Toolbox

Bum-Joo Lee, Yong-Duk Kim, Jeong-Ki Yoo, In-Won Park and Jong-Hwan Kim

*Abstract*— **This paper presents the simulation of humanoid walking pattern using 3D simulator with Virtual Reality Toolbox. 3D simulator is essential for developing motion planning and navigation algorithm. By using the Virtual Reality Toolbox incorporated with MATLAB, it is easy and simple to handle 3D objects. Detailed program architecture is explained through a pseudo code. In addition, modifiable walking pattern generation method is introduced and tested using the developed simulator. In the simulator, 3D design data of humanoid designed using CAD program are used. Example walking pattern is also demonstrated.**

## I. Introduction

Although humanoid robots have inherent complexities in locomotion, many successful robots are demonstrated. These robots have abilities to walk, run and dance as humans do. However, despite of these abilities, it is still insufficient to work with human in outdoor environment. Since humanoid robots are commonly composed of very complex hardware and software systems in which various technologies are integrated, it is not only cumbersome but also risky in the aspect of experiment out of doors.

Walking pattern simulator helps to evaluate motion of a robot without real hardware systems. By simulating before examine with real robot, it is possible to expect reliable and safe movement of a humanoid robot. Also, it helps to find and prevent some possible problems during actual experiments.

The representative simulator is OpenHRP (Open Architecture Humanoid Robotics Platform) which is developed in AIST (National Institute of Advanced Industrial Science and Technology) [1]. The platform was built on CORBA (Common Object Request Broker Architecture) which is a standard middleware and ART-Linux (Ishiwata and Matsui, 1998) is used as the real-time controller of the robot. In addition, the simulator and the controllers are regarded as white boxes whose internal API is clearly described by IDL (Interface Definition Language). Theses total packages including the simulator and controllers are composed as OpenHRP, which can simulate the dynamics of structure varying kinematic chains between open chains and closed ones that compose a humanoid robot. Collision-detection between a robot and its environment can be accomplished and utilized to calculate forward dynamics. Also, it can treat several important sensory systems including vision sensing, F/T sensing, and attitude sensor management.

Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 373-1, Guseong-Dong, Yuseong-Gu, Daejeon, 305-701, Republic of Korea (e-mail: {bjlee, ydkim, jkyoo, iwpark, johkim}@rit.kaist.ac.kr).

In this paper, 3D walking pattern simulator is introduced, which is under development by the RIT Lab. at KAIST. Furthermore, novel walking pattern generation method used in this simulator is explained. Although the proposed simulator is the first version of development, it holds several advantages. First, the simulator was built using MATLAB with Virtual Reality Toolbox. MATLAB provides powerful engineering tool including frequently used mathematical functions. It is easy to implement control algorithm including visualization of data used in the algorithm. In addition, by using Virtual Reality Toolbox, it is convenient to treat 3D objects defined with VRML (Virtual Reality Modeling Language). Thus, it is possible to build a simulator within a relatively short period.

The first part of this paper introduces a simulator for the purpose of evaluation of proposed walking pattern generator. Since basic element of MATLAB is an array that does not require dimensioning, it is advantages to solve many technical computing problems, especially those with matrix and vector formulation. Note that it contains numerous merits to acquire and analysis simulation data.

The second part of this paper introduces a method to generate walking pattern to adjust from complex navigational command. This algorithm is an extended version of conventional 3D-LIPM method [2]. By manipulating the concept of ZMP (Zero Moment Point), a simple and direct means to control the humanoid dynamics is realized. Significantly, the proposed walking pattern generation enables control of CM velocity and step lengths independently. This was not achievable with a conventional 3D-LIPM control algorithm.

The paper is organized as follows: Section II presents fundamental elements of the simulator including 3D object handling. In Section III, architecture of the simulator is described. In Section IV, walking pattern generation which can handle complex navigational commands is introduced. Section V demonstrates experimental results and concluding remarks follow in Section VI.

## II. 3D Environment

### A. Program Environment

*1) Virtual Reality Toolbox:* Simulator developed in this paper is based on the Virtual Reality Toolbox. The Virtual Reality Toolbox allows to connect an virtual world, defined with VRML, to MATLAB or Simulink programs. VRML was first developed by artists and engineers who want to enhance the content of Web pages with advanced 3D graphics and interaction with those graphics. After that, some extensions are added and developed as VRML2 Standard.

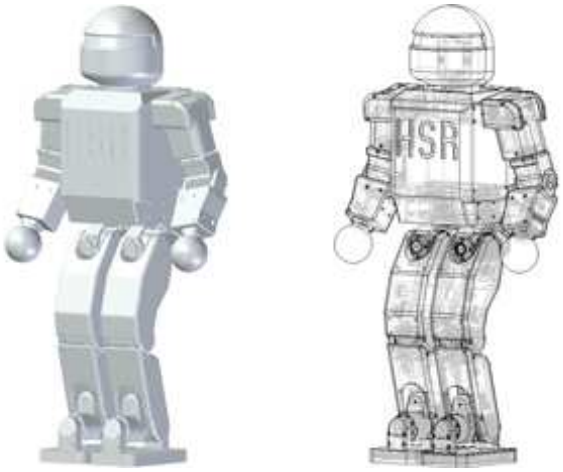
The Virtual Reality Toolbox provides two types of interfaces: MATLAB and Simulink. MATLAB interface is utilized in the proposed paper. The Virtual Reality Toolbox provides a flexible MATLAB interface to a virtual reality worlds. It is very simple to control the virtual world by using functions and methods by associating the MATLAB objects to a virtual world. Using the Virtual Reality Toolbox, it takes few times to make desire simulator and to test the proposed control scheme.

*2) MATLAB with VRML:* MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB features a family of add-on application-specific solutions called, *toolboxes*.

### B. 3D Data Processing

In order to use 3D-design files drawn in CAD tool, it should pass through converting process with following subsequences.

*1) Design necessary parts and assemble into one link:* Generally, robot is designed with 3D CAD tool and these data are also utilized in simulator for actuality. Fig. 1 shows the design of small sized humanoid robot, HSR-VIII, which is under production.




(a) (b)

Fig. 1. 3D design drawings (a) Shaded format (b) Visible and hidden edges format

*2) Alignment of the coordinate system into the assembled part:* Since the position and orientation of the links are represented in their reference frame, the assembled part should aligned exactly. Fig. 2 shows a shank part which is aligned along the reference frame. In this figure, center of the reference frame is coincided with center point between the knee joint and the ankle joint.

*3) Export the assembled into VRML file:* After finishing the design of each links, it should be saved or converted with VRML file extension (*.**wrl**).

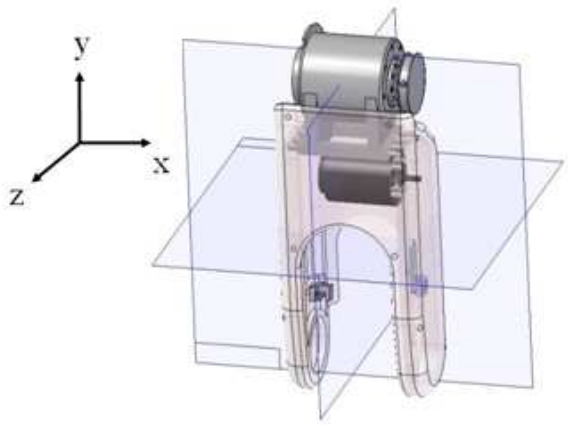



Fig. 2. Example of aligned link.

### C. Coordinate system

*1) Coordinate systems in each program:* Fig. 3 shows coordinate systems in each platform. VRML coordinate system is different compared to humanoid robot coordinate system. In the humanoid robotic, forward moving direction is defined as x-axis in Cartesian coordinate and the height of robot is defined in z-axis (Fig. 3 (a)). In contrast, y-axis of the VRML coordinate system points upward and its z-axis places objects either nearer or farther from front of the screen (Fig. 3 (a)). Thus, it is critical to apply a proper conversion relationship between these two different coordinate systems.


(a) (b)

Fig. 3. Coordinate systems. (a) Robot coordinate system. (b) VRML coordinate system.

*2) Transformation from Rotation matrix to Rotation axis:* Normally, there are two kinds of notational methodology to apply an object orientation relative to the inertial frame. One is a rotation matrix well used in classical robotics and the other is axis-orientation, which is well known as quaternion and used in aerodynamics. In the Virtual Reality Toolbox, the axis-orientation method is used. The relationships between the rotational matrix and the axis orientation are given by:

$$\begin{aligned} \cos\phi &= (r_{11} + r_{22} + r_{33} - 1)/2 \\ \boldsymbol{a} &= \frac{1}{2\sin\phi}\begin{bmatrix} r_{32} - r_{23} & r_{13} - r_{31} & r_{21} - r_{12} \end{bmatrix}^T \end{aligned} \tag{1}$$

where $\boldsymbol{a}$ is a axis of orientation and $\phi$ is a orientation angle, respectively. $r_{ij}$ is a direction cosine of general rotation matrix, which is given by:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

## III. Architecture of the Simulator

### A. Simulator layers

Generally, dynamic simulator is composed of following three components:

1) Controller part
2) Dynamic engine part
3) Display part

Each component is developed as independent modules for portability. The controller part takes charge of control algorithm, which is expected to examine. In this simulator, *Modifiable Walking Pattern Generator* is mounted as a gait planner. Detailed algorithm is explained in section IV. In the dynamic engine part, physical information of a robot is considered. Main purpose of this part is to give a decision whether the walking pattern is stable or not. Even though conventional method is to use ZMP criteria, this in not yet implemented in this simulator and left as a future work. Note that main concern of this paper is to introduce how to make simulation using VRML associated with MATLAB. Display part draws a robot in 3D space.

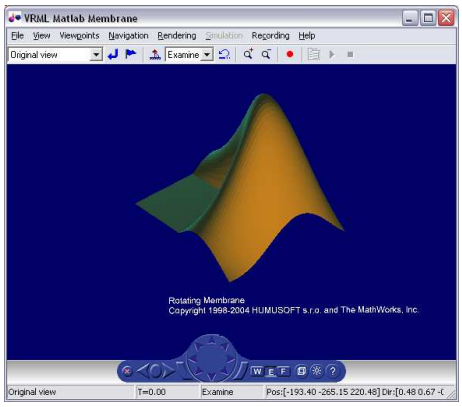



Fig. 4. Virtual Reality Toolbox viewer.

Fig. 4 illustrates default viewer of the Virtual Reality Toolbox for viewing virtual worlds. It contains navigational panel, which is commonly used for navigation operations. Note that simulation result can be saved as animation file (*.avi). Using this viewer, walking patterns can be tested with reality.

### B. Program Components

As commented above, program is designed with several dependent components. Block diagram of these components is illustrated in Fig. 5.

*1) Main code:* This is a main routine where sub-routines are executed. In this part, some initial and final operations are carried out: 3D object shaped variable definition, flag setting, simulation data management, etc.

*2) Walking pattern code:* This is a core algorithm part to generate walking patterns. At every sample time, this code is executed. To evaluate the proposed walking pattern algorithm, complex navigational commands are requested. After translating these commands into the desired walking

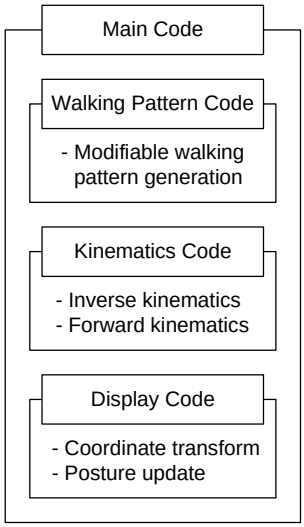



Fig. 5. Code structure.

state, the CM motions are generated under the 3D-LIPM dynamics. Then, the position and orientation of the CM is transferred into kinematics code.

*3) Kinematics code:* In this part, the CM motion is translated as joint angles using inverse kinematics. These data are utilized as servo control inputs. It is assumed that in the simulator, the joint angles are controlled exactly according to the control inputs. Note that in order to obtain posture of a robot with respect to the global frame, the position and orientation of each link are calculated through the forward kinematics.

*4) Display code:* Based on the VRML, robot is drawn in 3D space. After conversion of the coordinate systems from the robot to the VRML, the position and orientation of each link are updated at every sample time.

## IV. Walking Pattern Generation

In order to take complex navigational commands, a humanoid robot requires a walking pattern generator that is able to modify the pattern at any point during walking [3], [4]. Walking pattern generator embedded in the simulator allows this by manipulating the ZMP trajectory in real time. In the controller, a humanoid robot is modeled as 3D-LIPM and its walking state, that is CM position and velocity, is controlled independently.

### A. Equations of Motion

Fig. 6 illustrates the model, which is used in the paper for walking pattern generation. This model is introduced by Kajita et al. [2] and holds some practical advantages. In this model, it is assumed that the leg is a weightless telescopic limb and the height of CM is constant, namely $z = Z_C$. With these assumption, the dynamic equations are given in (2).

$$\begin{bmatrix} \ddot{y} - \frac{g}{Z_C} y \\ \ddot{x} - \frac{g}{Z_C} x \end{bmatrix} = -\frac{g}{Z_C} \begin{bmatrix} y_{zmp} \\ x_{zmp} \end{bmatrix}. \quad (2)$$

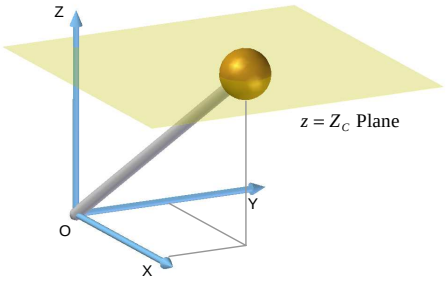



Fig. 6. 3D Linear Inverted Pendulum Model.

From (2), the state space equation of motion in the sagittal plane with arbitrary ZMP function $p(t)$ is derived with the convolution integral form, which is shown as follows:

$$\begin{bmatrix} x_f \\ T_C\dot{x}_f \end{bmatrix} = \begin{bmatrix} C_T & S_T \\ S_T & C_T \end{bmatrix} \begin{bmatrix} x_i \\ T_C\dot{x}_i \end{bmatrix} - \frac{1}{T_C} \begin{bmatrix} \int_0^T S_t\overline{p}(t)dt \\ \int_0^T C_t\overline{p}(t)dt \end{bmatrix} \quad (3)$$

where $(x_i, v_i)/(x_f, v_f)$ represents initial/final position and velocity of the CM in the sagittal, respectively. $S_t$ and $C_t$ are defined as $\cosh(t/T_C)$ and $\sinh(t/T_C)$ with time constant $T_C = \sqrt{Z_C/g}$. The functions $p(t)$ is ZMP trajectory for the sagittal. Lastly, $\overline{p}(t) = p(T-t)$. Solutions for motion in the lateral plane can be similarly obtained.

The latter term of (3) represents additional state by ZMP manipulation. Variations of ZMP trajectory means additional acceleration or deceleration of the CM. This has the effect that the CM motion can be flexibly adjusted in response to dynamically changing navigational commands.

### B. Modifiable Walking Pattern

Since the robot is modeled as the inverted pendulum with a point mass, its state in 3D space can be fully represented in terms of position and velocity. The CM position and velocity are defined as two dimensional vector in the following.

***Definition** 1:* WS (Walking State) is defined as follows:

$$\boldsymbol{x} = \begin{bmatrix} x & T_C v \end{bmatrix}^T \text{ for sagittal motion}$$
$$\boldsymbol{y} = \begin{bmatrix} y & T_C w \end{bmatrix}^T \text{ for lateral motion,}$$

The primary factors constraining the walking pattern in single support phase include the single support time and the ZMP trajectory functions. By selecting suitable ZMP trajectory and the single support time, it is possible to move the current WS to the desired one. Among infinitely many possible ZMP functions, it can be shown that any arbitrary bounded ZMP function can be equivalently represented by a simple step function with same bound constraints. In other words, the walking state obtained at the end of a single support phase given an arbitrary function can be obtained from an appropriately selected step function.

To completely specify the ZMP trajectories in single support phase would require determination of five parameters (for step amplitudes and switching times for sagittal and lateral plane and for state transition time). In terms of implementation, it is desirable to reduce this by one so that the parameters can be exactly determined from the equations of motion (2). Subsequently, the use of a constant function was selected for the sagittal ZMP function. Fig. 7 illustrates two proposed ZMP functions.

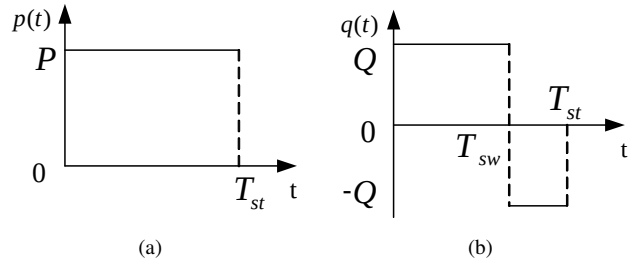



Fig. 7. Proposed ZMP functions. (a) Constant function (sagittal plane). (b) Step function (lateral plane).

While in single support phase, the robot can vary the walking state by manipulating the ZMP function. However, this does not mean that the robot has ability to change the walking state to any desired one because the robot is constrained by ZMP boundary. The ZMP functions must be bounded to some region which is defined by physical foot size. Thus, the desired state is also bounded by some permissable region. This region is defined as the Feasible Region, FR.

***Definition** 2:* Given the initial WS $\mathbf{w}_i$, $Z_{min}, Z_{max}$ and some $T_{max} \geq 0$, a set is defined as the *feasible region*, *FR*, if $\forall\, \mathbf{w}_f \in FR$, there exists a ZMP function $z(t)$ with the following properties:

$$z : [0, T_{max}] \rightarrow \mathbb{R}^2,$$
$$Z_{min} \leq z(t) \leq Z_{max}$$

such that

$$f_z(\mathbf{w}_i) = \mathbf{w}_f$$

where $f$ is the state transition function for either the sagittal or lateral plane (3). $FR^c$ is defined as the *infeasible region*.

Fig. 8 illustrates this concept. It can be determined whether the desired walking state is feasible or not by observing the walking state is in FR or not. That is, FR does act as criteria to judge that the navigational command is possible or not.

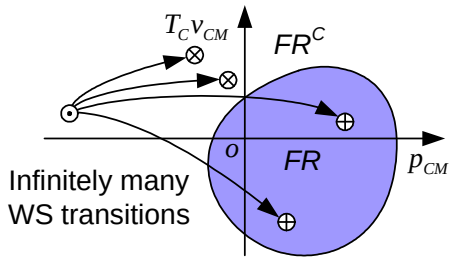



Fig. 8. Feasible and infeasible regions on WS plane ($\odot$: initial WS, $\oplus$: final WS $\in FR$, $\otimes$: final WS $\in FR^c$).

Fig.9 shows the control block diagram used for HSR-VII, which is most recently developed in RIT Lab. For each sample time, the navigational command is converted

to the correspondent waking state, desired walking state. Depending on the feasibility, the target walking state is decided. Subsequently, motion of the CM is updated in real-time.

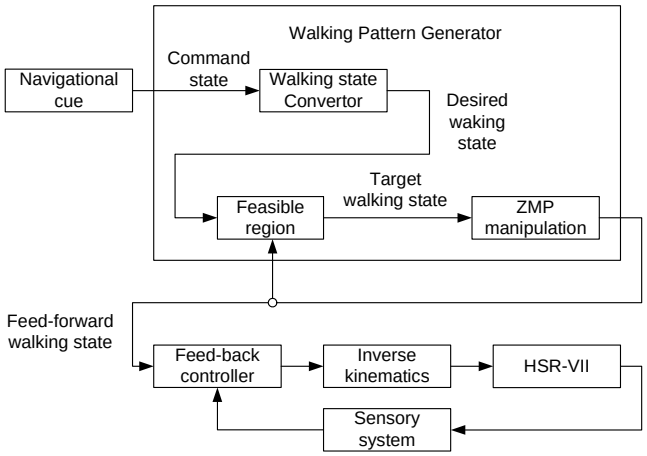



Fig. 9. Control block diagram.

## V. Simulation

Example walking patterns are generated based on the Modifiable Walking Pattern Generator for the small-sized humanoid robot, HanSaRam-VIII [5]–[7].

### A. Simulation Condition

Walking patterns for simulations were generated with navigational commands in Fig. 10.

| Time | $WF_L$ | $WF_R$ | $WS_L$ | $WS_R$ | $\Theta_L$ | $\Theta_R$ | $T_{ssL}$ | $T_{ssR}$ | $T_{dsL}$ | $T_{dsR}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Initial | 6 | 6 | 6 | −6 | 0 | 0 | 0.4 | 0.4 | 0.2 | 0.2 |
| After 1st step | 6 | 6 | 6 | −6 | 0 | 0 | 0.6 | 0.6 | 0.3 | 0.3 |
| After 2nd step | 5 | 6 | 7 | −8 | −10 | −30 | 0.6 | 0.8 | 0.2 | 0.3 |
| After 4th step | 6 | 5 | 8 | −7 | 30 | 10 | 0.8 | 0.6 | 0.3 | 0.2 |
| After 6th step | 2 | −5 | 8 | −7 | 60 | 10 | 0.8 | 0.6 | 0.3 | 0.2 |
| After 9th step | −6 | −5 | 8 | −8 | 60 | 0 | 0.8 | 0.6 | 0.3 | 0.2 |
| After 11th step | −6 | −5 | 8 | −8 | 0 | −60 | 0.8 | 0.6 | 0.3 | 0.2 |
| After 14th step | 0 | 0 | 6 | −6 | 0 | 0 | 0.6 | 0.6 | 0.2 | 0.2 |

Fig. 10. An example of navigational commands.

Navigational commands are composed of ten elements. $WF$ and $WS$ represent forward and side step lengths, respectively. Here, subscripts, L and R, mean left and right leg, respectively. $\theta$ is rotation angle of the footstep. Lastly, $T_{ss}$ nd $T_{ds}$ represent single and double support time, respectively. With these objectives, navigational commands are informed to the robot with listed times as specified.

### B. Results

Fig. 11 shows the generated walking pattern using *modifiable walking pattern generator*. As illustrated, the footsteps are generated, even when the commands varied suddenly, that is, the sudden side and backward walking motion with some rotations during the whole path.

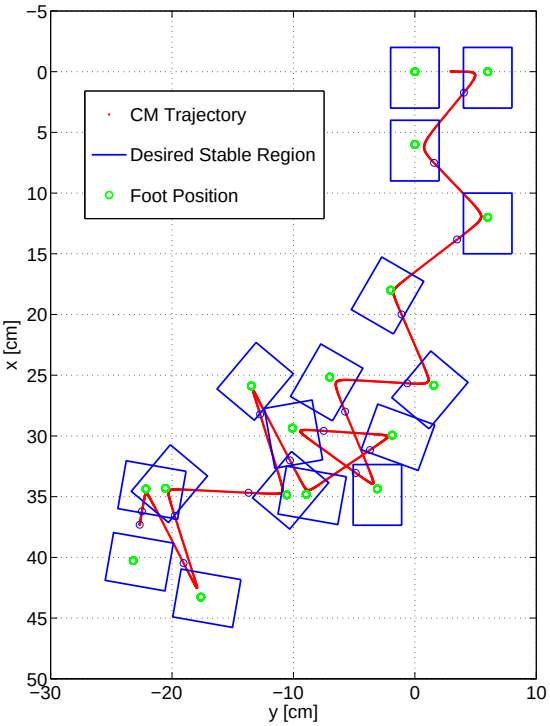



Fig. 11. An example walking pattern.

Fig. 12 shows a sequence of snapshots, which was captured sequentially in two second intervals from the simulator. As illustrated in Fig. 11, the robot achieved all that motions including rotatively backward and side walking motions. Refer to an animation clip [8], [9].

## VI. Conclusions

In this paper, walking pattern simulator was introduced. This simulator was built on Virtual Reality Toolbox with MATLAB interface. The simulator was composed of three modules, namely, waking pattern code, kinematics code and display code. In the process of waking pattern generation, modifiable walking pattern algorithm was implemented, which could handle complex navigational commands. In kinematics code, inverse and forward kinematics were calculated. Finally, display code took charge of 3D graphics.

Significantly, modifiable walking pattern algorithm was detailed. This algorithm is based on the conventional 3D-LIPM approach, but extended to manipulate the ZMP over the convex hull of the foot polygon. By define a feasible region, it was possible to check whether the command was practicable or not. This algorithm were implemented on the developed simulator and evaluated its usefulness. Dynamic engine, which consider the interaction between a robot and its environment, is left as the further work.

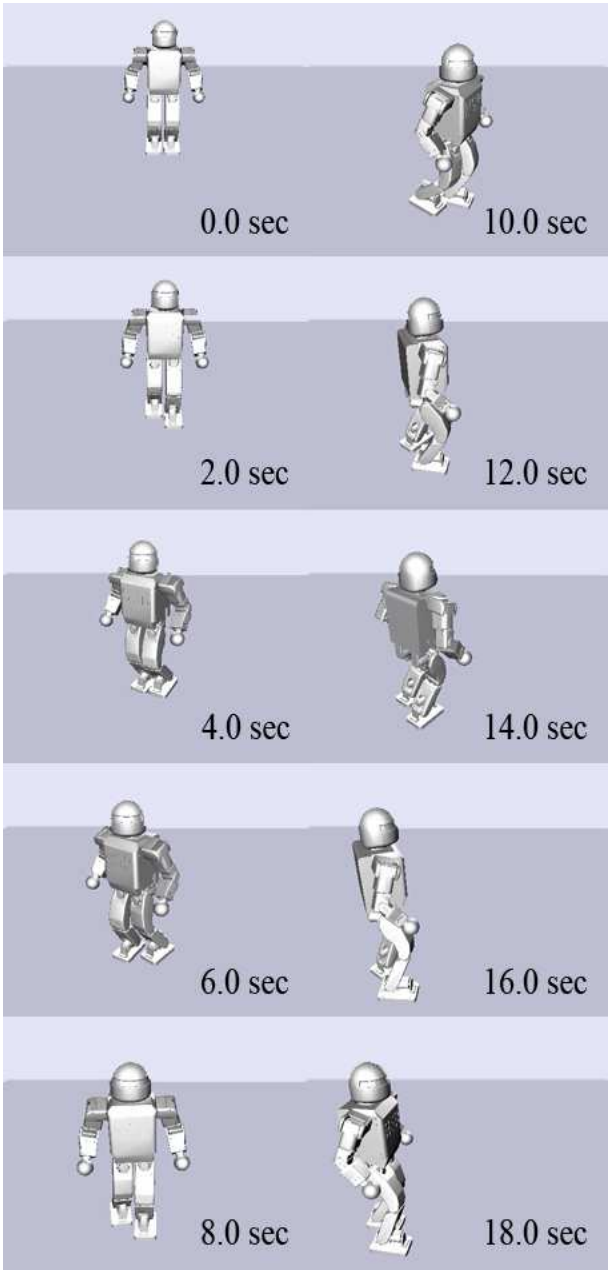



Fig. 12. A snapshot of the example walking pattern generated in the simulator.

## REFERENCES

[1] F. Kanehiro, H. Hirukawa, and S. Kajita, "OpenHRP: Open Architecture Humanoid Robitics Plarform" in the Int. Journal of Robotics Research, vol. 23, no. 2, Feb. 2004, pp. 155-165.

[2] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa, "A Realtime Pattern Generator for Biped Walking" in Proc. of IEEE Int. Conf. on Robotics and Automation, Washington, DC, May 2002, pp. 31-37.

[3] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade, "Online environment reconstruction for biped navigation" in Proc. of IEEE Int. Conf. on Robotics and Automations, Orlando, Florida, May 2006, pp. 3089-3094.

[4] J. Chestnutt, P. Michel, K. Nishiwaki, J. Kuffner, and S. Kagami, "An intelligent joystick for biped control" in Proc. of IEEE Int. Conf. on Robotics and Automations, Orlando, Florida, May 2006, pp. 860-865.

[5] J.-H. Kim, D.-H. Kim, Y.-J. Kim, K.-H. Park, J.-H. Park, C.-K. Moon, K. T. Seow, and K.-C. Koh, "Humanoid robot hansaram: Recent progress and development" in J. of Advanced Computational Intelligence & Intelligent Informatics, vol. 8, no. 1, Jan. 2004, pp. 45-55,

[6] Y.-D. Kim, B.-J. Lee, J.-K. Yoo, and J.-H. Kim, "Compensation for the Landing Impact Force of a Humanoid Robot by Time Domain Passivity Approach" in Proc. of IEEE Int. Conf. on Robotics and Automations, Orlando, Florida, May 2006, pp. 1225-1230.

[7] J.-H. Kim, D.-H. Kim, Y.-J. Kim, K.-T. Seow, "Soccer Robotics (Springer Tracts in Advanced Robotics)," Springer Verlag, 3540218599, 326, Sep. 2004

[8] B.-J. Lee, D. Stonier, Y.-D. Kim, J.-K. Yoo, and J.-H. Kim, "Modifiable Walking Pattern with HanSaRam-VII" at http://rit.kaist.ac.kr/∼ritlab/research/HanSaRam/ MWPG_simulation .wmv , Apr. 2007.

[9] B.-J. Lee, D. Stonier, Y.-D. Kim, J.-K. Yoo, and J.-H. Kim, "Modifiable Walking Pattern with HanSaRam-VII" at http://rit.kaist.ac.kr/∼ritlab/research/HanSaRam/ Modifiable_Walking _Pattern.wmv , Nov. 2006.