Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances

Shin'ichiro Nakaoka
Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan
nakaoka@cvl.iis.u-tokyo.ac.jp

Atsushi Nakazawa
Cybermedia Center, Osaka University
1-32 Machikaneyama, Toyonaka, Osaka, 560-0043, Japan
nakazawa@cmc.osaka-u.ac.jp

Fumio Kanehiro
Kenji Kaneko
Mitsuharu Morisawa
Hirohisa Hirukawa
Intelligent Systems Research Institute
National Institute of Advanced Industrial Science and Technology
1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan
{f-kanehiro, k.kaneko, m.morisawa, hiro.hirukawa}@aist.go.jp

Katsushi Ikeuchi
Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan
ki@cvl.iis.u-tokyo.ac.jp

Abstract

This paper proposes a framework that achieves the Learning from Observation paradigm for learning dance motions. The framework enables a humanoid robot to imitate dance motions captured from human demonstrations. This study especially focuses on leg motions to achieve a novel attempt in which a biped-type robot imitates not only upper body motions but also leg motions including steps. Body differences between the robot and the original dancer make the problem difficult because the differences prevent the robot from straightforwardly following the original motions and they also change dynamic body balance. We propose leg task models, which play a key role in solving the problem. Low-level tasks in leg motion are modelled so that they clearly provide essential information required for keeping dynamic stability and important motion characteristics. The models divide the problem of adapting motions into the problem of recognizing a sequence of the tasks and the problem of executing the task sequence. We have developed a method for recognizing the tasks from captured motion data and a method for generating the motions of the tasks that can be executed by existing robots including HRP-2. HRP-2 successfully performed the generated motions, which imitated a traditional folk dance performed by human dancers.

KEY WORDS—learning from observation, imitation, biped humanoid robot, motion capture, entertainment robotics

1. Introduction

Learning from Observation (LFO) (Ikeuchi and Suehiro 1994) is a paradigm that enables a robot to acquire a way of doing a task just by observing demonstrations of a human instructor. This paper proposes a framework that achieves the LFO paradigm for learning dance motions including stepping leg motions. The framework enables a biped humanoid robot to use its own legs to support the body when the robot performs learned dance motions. This is a novel attempt at humanoid robots.

The performance of dances is a useful subject of learning whole-body motions. Since dances generally include various dynamic motions using the whole body, they can clarify

The International Journal of Robotics Research
Vol. 26, No. 8, August 2007, pp. 829–844
DOI: 10.1177/0278364907079430

basic problems of learning whole-body motions. In addition, the significance of humanoid robots can be enhanced when they can learn and replay dance motions that are significant from the viewpoint of art or entertainment (Nakazawa et al. 2002).

In order to achieve the LFO paradigm, it is important for a robot to understand "what to do" in a task. This means that the robot must recognize essential points for performing the task from demonstrations. Otherwise the robot cannot robustly perform the task under various disturbances. In hand operations including assembly (Ikeuchi and Suehiro 1994), pick and place (Kuniyoshi, Inaba and Inoue 1994), grasping (Kang and Ikeuchi 1997), and tying (Takamatsu et al. 2006), the central issue is how to represent a way of doing a task in order to achieve the robust operation because these tasks cannot be completed when a robot just replays demonstrated motion trajectories.

The same holds true for dances. First of all, human motion data such as joint trajectories cannot be directly mapped to a robot because of differences between their bodies. Their joints can have differences in mechanical structure, degrees of freedom (DOF), movable range, and/or maximum moving speed. Furthermore, even if the joints of a robot can track the original trajectories, the robot would not be able to keep its dynamic balance. The robot must move considering its own mass distribution and the physical property of the feet to let the feet contact stably to the floor and to support the body without falling down. Consequently, the original motions have to be modified to be feasible for a robot. At the same time, important characteristics of the original dance motions must be preserved as much as possible. Thus adapting motions into a robot must satisfy a number of constraints simultaneously. The constraints of leg motions are especially severe because of the physical interaction between the legs and the floor.

In order to solve the problem, we propose *leg task models*, which represent "what to do" in leg motions. A standing motion, stepping motion, and squatting motion are modelled as low-level *tasks*. Each task can represent its characteristics by its *skill parameters* without depending on a particular body. Leg motions in a dance are represented by a sequence of the low-level tasks, which is called a *task sequence*.

The most important objective of the leg tasks is to perform a sequence of various steps stably while keeping the original timings, positions, and orientations of the footfalls, which has not been achieved by the previous studies on biped humanoid robots that imitate human emotions. In order to achieve this, the task models provide the explicit contact states between soles and the floor.

We construct our LFO framework based on the leg task models. Its overview is shown in Figure 1. The framework divides the problem of learning into two subproblems: how to recognize a task sequence from observed motions and how to generate motions of a robot from the task sequence.

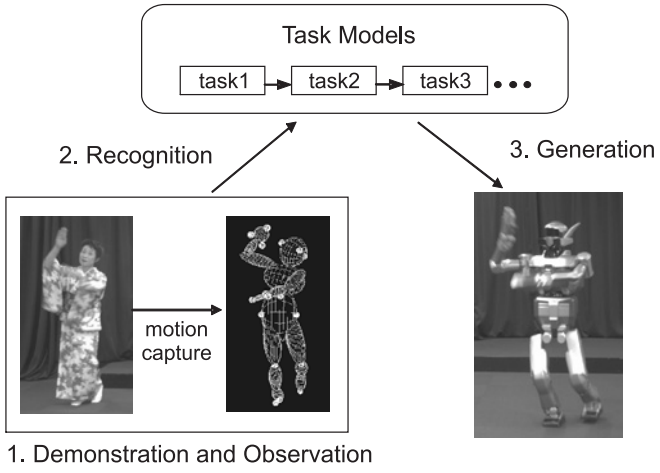

Fig. 1. Overview of our framework based on the LFO paradigm. A robot learns and performs a dance through the three steps shown. The recognition and generation processes are based on the task models.

In this paper, we propose a method for the recognition process using motion capture as the observation device. We also propose a generation method for existing robots including HRP-2 (Kaneko et al. 2004). These methods can automatically generate feasible robot motion data from captured human motions. With these methods, we verify the validity of our framework.

Note that upper body motions are processed by conventional methods because the main focus of this paper is on leg motions. The leg tasks are designed so that they can maintain dynamic balance for a given upper body motion.

In the next section, we review related studies. Section 3 clarifies the conditions for motions that the framework deals with and presents the details of the leg task models. Then, methods for the recognition process and the generation process are described in Section 4 and 5 respectively. Experimental results are shown in Section 6, and the results are discussed in Section 7. Finally, the contribution of this paper is summarized in Section 8.

2. Related Studies

The imitation of human motions by humanoids has been studied actively from various standpoints (Schaal 1999). Within such studies, we review those studies related to replaying various patterns of whole-body motion here.

Riley, Ude and Atkeson (2000) produced a dancing motion of a humanoid robot by converting a captured human motion into joint trajectories of the robot. For the same purpose, Pollard et al. (2002) proposed a method for constraining given joint trajectories within mechanical limitations of the joints.

Since these studies mainly focused on the constraints of joints, the methods are not sufficient to maintain the dynamic balance. In fact the pelvis of their robot was fixed in space. Yamane, Hodgins and Brown (2003) proposed a method for controlling a marionette so that it follows a captured human motion, but the mechanism of marionettes is quite different from that of biped humanoid robots.

For biped humanoid robots, Tamiya, Inaba and Inoue (1999) proposed a method that enables a robot to follow given motion trajectories while keeping its dynamic body balance, but the method can deal only with motions in which the robot is standing on one leg. Kagami et al. (2000) extended the method so that it allows changes of supporting leg. However the extended method directly inherited a constraint of the original method. That is, the projection of the center of mass to the floor (CM projection) must always be inside the *support area*, which is the convex hull of the sole planes on the floor. Hence the method cannot achieve so-called *dynamic walk* motions, where the CM projection can be outside the support area. Since humans generally perform dynamic-style steps, the limitation would significantly change the style of the original motions. On the other hand, our method allows dynamic-style steps.

Yamane and Nakamura (2003) proposed a *dynamics filter*, which is a general framework for converting a physically inconsistent motion for a given body into a consistent one. In their implementation, the filtering process is strictly local to each discrete time frame so that the filter can support online and interactive use. However, the strategy simultaneously loses the robustness of keeping global stability (e.g. not falling down) and key characteristics of a given motion because a result of the time-local tracking process may be far-from-the-original motion especially when the inconsistency in the motion is not small. Since the inconsistency caused by the difference between a human body and an existing robot like HRP-2 is significant, it would be difficult for the filter to convert a human dance motion into robot motion without losing the stability and other important characteristics.

To keep important characteristics on a different body, higher priority must be given to the factors that are more closely related with them. Shin et al. (2001) proposed a method that adapts a given motion to a CG character with higher priority on keeping the contact between the limbs and external objects. The priority is essential for motions including physical interactions with external objects, and our framework also puts importance on the contact between the feet and the floor. Although the dynamic stability of the contact does not have to be strict in CG animations, it must be strictly achieved for a robot, which makes our problem more difficult.

In the existing methods described above, the problem is basically solved by fitting low-level motion trajectories into a given body. On the other hand, our framework solves the problem by interpreting low-level motion trajectories as tasks, which are more abstract, structured motion representation, and motions of a robot are reconstructed from the representation. This paper shows that this approach is valid for achieving strictly feasible motions of a robot.

Kuroki et al. (2003) enabled a small biped humanoid to stably perform dance motions including dynamic-style steps. However, this achievement is not the same as our goal, because the motions of their robot were manually created using their motion editing tool (Kuroki et al. 2003).

3. Leg Task Models

3.1. Target Motion Class

Task models depend on a class of motions to learn. Our target class of motions is the dance under the following conditions.

1. The body contacts only with a level, flat floor.
2. The body contacts with the floor at a single sole or at double soles.
3. The soles do not slip on the floor.
4. At least one sole is always in contact with the floor.

Conditions (1) and (2) imply that the upper body is not directly constrained by external objects. It is therefore reasonable for the upper body to put higher priority on visual features than the lower body, which is directly constrained by the physical interaction with the floor. To this end, the task models of the upper body are separated from those of the lower body, and the lower body is moved to keep the dynamic balance for a given motion of the upper body. This paper focuses on the task models of the lower body called *leg task models*.

Conditions (3) and (4) exclude from this paper motions with a spin turn or a leap.

The scope of the paper is also limited to the class of dances in which a defined pattern of choreography is performed while keeping defined timings. Many dances are categorized into this class. A ballroom dance is an example not in the class, because motions depend on the interaction between a dancer and the partner (Kosuge et al. 2003).

3.2. Defined Tasks

Figure 2 shows the low-level tasks modelled for the target motion class described above.

The *R-STEP* represents one stepping motion by a right foot, and the *L-STEP* represents that by a left foot. These are collectively called the *STEP*. To be more accurate, a STEP task represents a motion in which one foot is lifted from the floor and is landed again while the other foot keeps contact with the floor (The former foot is called the *swing foot* and the latter foot is called the *support foot*). Using STEP tasks, various leg

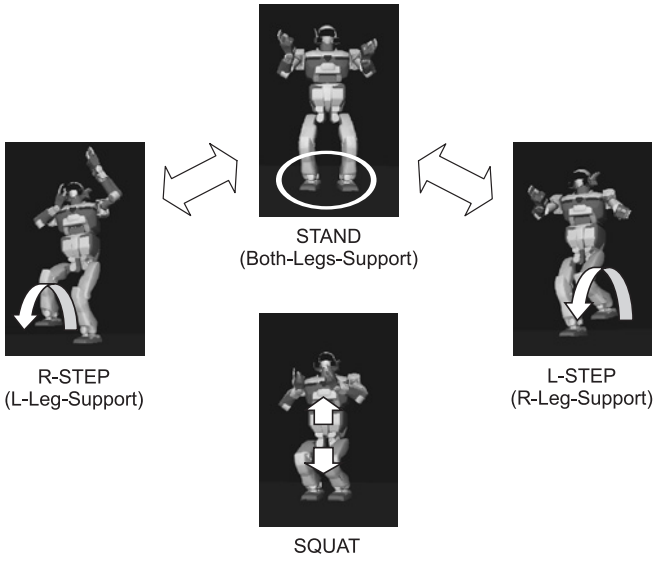

Fig. 2. The defined tasks. Arrowheads indicate transition relationship between tasks. Labels in parentheses show the support state.

motions including footfalls, side- or back-stepping, and kicks can be expressed. The *STAND* represents a motion in which the upper body is supported by both feet.

These tasks correspond to three *support states*, which indicate a set of legs that are supporting the body. The states include the left leg support, the right leg support and the both legs support. A state changes along with the transition between these tasks as shown by the arrowheads in Figure 2. Only one of these tasks is being performed at a time.

The *SQUAT* represents a motion in which the waist lowers once and rises again. Performing this task is independent of the other three tasks.

Although vertical movement of the waist is represented by SQUAT tasks, a task model that explicitly represents horizontal waist movement is not defined. Horizontal waist movement plays a main role in keeping the dynamic body balance, and the movement is indirectly controlled by STAND tasks. The details are described in Sections 5.4 and 5.5.

Each task has its own period to perform it. A continuous leg motion in a dance performance is represented by a *task sequence*, in which a number of tasks are arranged in a time sequence.

3.3. Skill Parameters

Each task has its own *skill parameters* that determine the timings and characteristics of the motion. Table 1 shows the sets of skill parameters defined for the tasks.

Table 1. Skill parameters

Common	t_0	Beginning time
	t_f	Finishing time
STAND	–	
STEP	Parameters of the swing foot	
	$\boldsymbol{r}_f$	Horizontal position on Σ_s at t_f
	$\boldsymbol{R}_f$	attitude on Σ_s at t_f
	Parameters of the waist	
	ψ_f	Yaw orientation on Σ_s at t_f
	Parameters of the mid-point (option)	
	t_1	Time of the mid-point
	$\boldsymbol{r}_1$,$\boldsymbol{R}_1$	Position and orientation of the swing foot on Σ_s at t_1
SQUAT	t_1	Time of the mid-point
	d_1	Waist height distance

(Σ_s is the relative coordinate based on the support foot.)

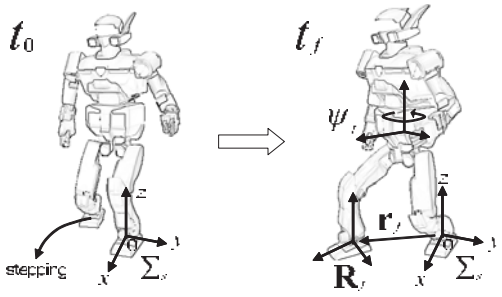

Fig. 3. Graphical representation of the STEP parameters.

All the tasks have the start time t_0 and the finish time t_f. These parameters enable tasks to be arranged in a time sequence, which is necessary to compose a choreography and to set a rhythm in a dance performance. To keep these values is especially important for the dances we deal with.

In the STEP, the values of the parameters with regard to position and orientation are based on Σ_s, which is the relative coordinate attached to the support foot. By the condition (3) in Section 3.1, Σ_s is fixed during a STEP task. In addition, z-axis of Σ_s is fixed at the upper direction of the global coordinate. Figure 3 shows the geometric correlation between the parameters.

Let $\boldsymbol{r}_f = (r_{f\cdot x} \;\; r_{f\cdot y})^T$ be the horizontal position of the swing sole when it lands on the floor at t_f. $\boldsymbol{R}_f$ is the rotation matrix that represents the attitude of the swing foot at t_f.

There is a case in which the swing foot takes a characteristic pose during a step. The pose is represented by a mid-point of

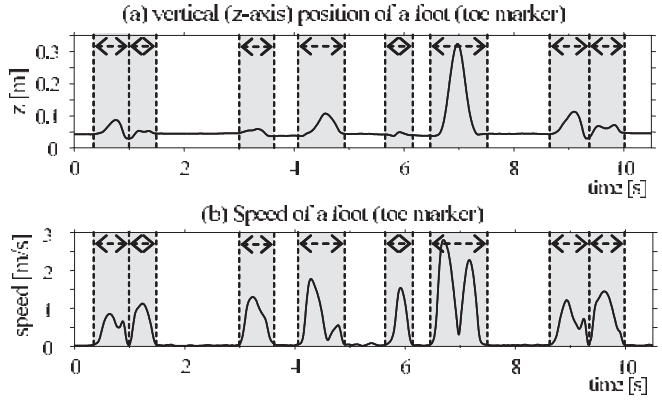

Fig. 4. Trajectories of a human foot motion, which are analyzed in the process of detecting STEP tasks. Each filled area corresponds to a STEP task detected with proper thresholds.

the swing foot trajectory. In this case, time of the mid-point, t_1, and position parameters $\boldsymbol{r}_1 = (r_{1\cdot x} \;\; r_{1\cdot y} \;\; r_{1\cdot z})^T$ and $\boldsymbol{R}_1$ at t_1 are added to the skill parameters. How to determine these values is described in Section 4.

Parameter ψ_f is the yaw angle of the waist at t_f. This parameter allows motions in which the global orientation of the body changes as a result of stepping.

The SQUAT has the parameters of the mid-point at which the waist reaches the lowest position in the motion. Skill parameter t_1 is the time at the mid-point, and d_1 is the vertical distance of the waist from the start point to the mid-point.

The tasks do not have parameters that represent positions at the start point. Those values inherit the result of the previous tasks. In addition, all the geometric parameters are described by relative coordinates. These features allow local modification of tasks in a task sequence, which is necessary for the process of skill refinement described in Section 5.7.

4. Task Recognition

This section describes a method for recognizing a task sequence from marker trajectories obtained by motion capture. For each task model, first, temporal segments corresponding to a task are detected. Then, for each detected task, the values of its skill parameters are extracted from correlations between several markers.

A STEP task is detected by analyzing the trajectory of a foot. Let $\boldsymbol{p}(t) = (p_x(t) \;\; p_y(t) \;\; p_z(t))^T$ be the position of a foot marker at time t. Speed of the foot marker is represented as $v_p(t) = |\dot{\boldsymbol{p}}(t)|$. In Figure 4, graph (a) shows an example of $p_z(t)$, and graph (b) shows its $v_p(t)$. In a trajectory of $v_p(t)$, a segment that continuously has positive values can be regarded as a movement of a foot in stepping.

Therefore, a segment that satisfies the following conditions is detected as one STEP task:

$$v_p(t) \geq v_{step} \quad (t_0 \leq t \leq t_f), \qquad \int_{t_0}^{t_f} v_p(t)dt \geq l_{step}, \tag{1}$$

where v_{step} and l_{step} are threshold values in terms of velocity and moving distance respectively. These thresholds eliminate noise-like slight motions when the foot is a support foot. In this way, a right foot and left foot are analyzed to detect R-STEPs and L-STEPs respectively.

Ikemoto, Arikan and Forsyth (2006) proposed a method for detecting when the feet should be planted on the floor, and the method is available for detecting the segments of STEPs. The advantage of their method is the reliableness achieved by a trained classifier, but the classifier requires additional training procedures by a user for each different body model. Our simple method can be sufficiently reliable for our purpose as we show in Section 6.2.

In a STEP task, whether the mid-point is added or not must be determined. First, a model trajectory of the swing foot is generated by interpolation from the start point to the finish point. If the distance between the model trajectory and the actual trajectory is far, the mid-point is added to express that trajectory.

Here we define a spline-like interpolating function which passes $n(\geq 2)$ points where the time, value and velocity are t_i, $\boldsymbol{y}_i$ and $\dot{\boldsymbol{y}}_i$ respectively. One segment between the two adjacent points is expressed by a third polynomial equation. The function is expressed as

$$\boldsymbol{f}_n\langle(t_1, \boldsymbol{y}_1, \dot{\boldsymbol{y}}_1), \cdots, (t_n, \boldsymbol{y}_n, \dot{\boldsymbol{y}}_n)\rangle(t). \tag{2}$$

When $\dot{\boldsymbol{y}}_i$ is omitted, $\dot{\boldsymbol{y}}_i$ is supposed to be $\boldsymbol{0}$. (The expression is also used in Section 5.)

By using the function, the interpolated trajectory is generated as $\boldsymbol{p}'(t) = f_2\langle\,(t_0,\boldsymbol{p}(t_0)),\,(t_f,\boldsymbol{p}(t_f))\,\rangle(t)$. The difference between two trajectories is defined as $d(t) = |\boldsymbol{p}'(t) - \boldsymbol{p}(t)|$. The mid-point is added at time t_1 if

$$d(t_1) = \max_{t_0 < t < t_f} d(t), \quad d(t_1) > d_{step}, \tag{3}$$

where d_{step} is a threshold distance.

After R-STEPs and L-STEPs are detected, STAND tasks are detected as the segments in which neither R-STEPs nor L-STEPs appear.

SQUAT tasks are detected by analyzing a vertical trajectory of the waist. In Figure 5, graph (a) shows the vertical position of the waist $h(t)$ at time t, and graph (b) shows its velocity $v_h(t) = \dot{h}(t)$. A SQUAT task corresponds to a motion in which the waist lowers and rises again. This kind of motion is detected as a segment from t_0 to t_f that satisfies

$$\begin{cases} v_h(t) < 0 & (t_0 \leq t < t_1) \\ v_h(t) > 0 & (t_1 < t \leq t_f) \end{cases}, \qquad \int_{t_0}^{t_f} |v_h(t)|dt \geq l_{squat}, \tag{4}$$

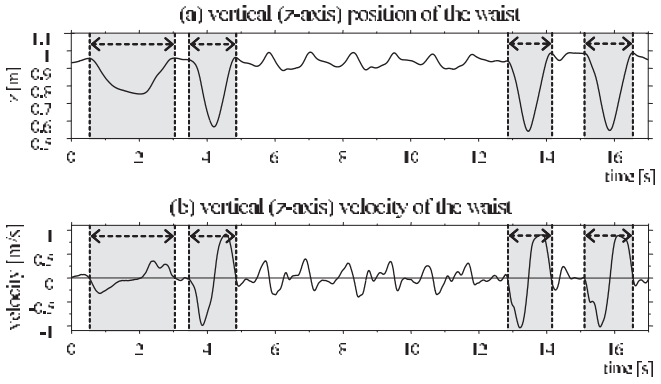

Fig. 5. Vertical motion trajectories of the waist. Each filled area corresponds to a SQUAT task detected with a proper threshold.

where t_1 corresponds to the timing of the lowest waist position and l_{squat} is a threshold for the vertical moving distance, which eliminates slight vertical motions that are not regarded as a SQUAT.

For each task, the values of its skill parameters are extracted after the segment is detected. Time values t_0, t_f and t_1 (if required) in the conditions of the detection are directly set to the time parameters. The values of the other parameters with regard to the positions and orientations are extracted from the positional correlations between the corresponding markers at the moments of the time parameters.

5. Task Generation

5.1. Overview

This section describes a method for generating motions of a robot from a task sequence. As shown in Figure 6, the method consists of several components and processes, which are integrated into the *task generation system*.

In addition to a task sequence, the system requires joint angle trajectories of the upper body. Leg motions are generated so that they can keep the dynamic balance for the given upper body motions. The system can accept any upper body motions as long as they satisfy the mechanical constraints of a robot. An example of generating upper body motions is presented in Section 6.3.

From the input data, the system generates reference motion data that is executable for the robot. The data consists of joint angle trajectories of the whole body and a reference Zero Moment Point (ZMP) trajectory. The latter is used for on-line stabilization control.

5.2. Components of the Task Generation System

As shown in Figure 6(b), the system has six units of body state variables. They are desired ZMP, support state, position and attitude of the swing foot, waist orientation, vertical waist position, horizontal waist position, and torso orientation. Leg motions are basically determined by these states. Arrowheads towards each variable indicate which process controls it.

Figure 6(a) shows *task processors*. Each task processor works for a particular task model. For each corresponding task in the task sequence, a processor controls state variables indicated by the arrowheads according to the skill parameters of the task.

The state variables are also controlled by filters of the whole-body dynamics shown in Figure 6(d). Two filters work in the system; one is the *ZMP compensation filter* for preventing the body from falling down, and the other is the *yaw moment compensation filter* for preventing spinning of the soles.

The processors and the filters work in *generation loops* of a proper time step Δt. In every loop, the state variables are updated, and the joint angles of the legs are calculated by the inverse kinematics between the foot and the waist. The generation loop is iterated until the final task in a task sequence is finished. As a result, joint angle trajectories of the robot are generated.

In a generation loop, the system checks a fault that makes the generated motion infeasible for the robot (Figure 6(e)). Such faults include collisions between body parts and overruns of maximum joint angles/velocities. If a fault is detected, the system eliminates the fault by modifying the skill parameters related with the fault. This process is called *skill refinement*.

The system has the *link model* of the robot, which is used for calculating the forward and inverse kinematics and the dynamic properties required by the filters. The model is also used for the collision detection in the skill refinement.

The system has *execution parameters* shown in Table 2. These parameters correspond with motion factors that depend on the paricular robot. They are needed to generate feasible and stable motions for various robots. Details of each parameter are described in the following sections.

Table 2. Execution parameters

h_h	Reference sole height for the horizontal impact reduction
h_v	Reference sole height for the vertical impact reduction
v_v	Velocity threshold for the vertical impact reduction
h_w	Default waist height
h_s	Default stepping height
t_z	Time margin of the ZMP transition
t_p	Time threshold of centering the ZMP
θ_s	Angle margin of the knee joints
Δt	Discrete time step

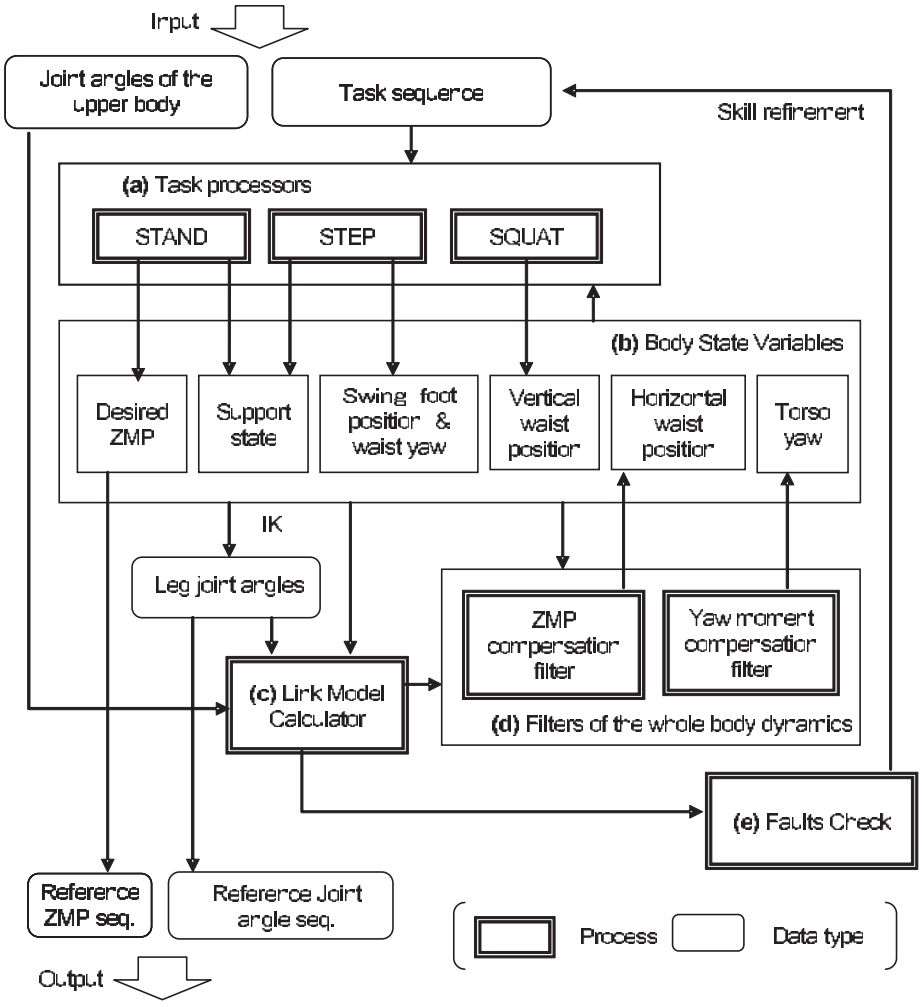

Fig. 6. The components and processes of the task generation system.

5.3. Task Processors

The SQUAT processor controls the vertical position of the waist. Let $p_{wt\cdot z}(t)$ be the vertical position of the waist at time t. This value is usually the value of execution parameter h_w. The SQUAT processor changes this value according to the skill parameters of SQUAT during its execution. By using the interpolation function (2), the processor set the following trajectory generated from skill parameters t_0, t_1, t_f, d_1:

$$p_{wt\cdot z}(t) = f_3\langle(t_0, h_w), (t_1, h_w - d_1), (t_f, h_w)\rangle(t). \quad (5)$$

The STEP processor first updates the support state into the right leg support (in the L-STEP) or the left leg support (in the R-STEP). Then the processor controls the position and attitude of the swing foot and the yaw orientation of the waist during the task execution. The support foot is not controlled; it remains in the same position.

Let $\boldsymbol{p}_{sw}(t) = (p_{sw\cdot x}(t), p_{sw\cdot y}(t), p_{sw\cdot z}(t))^T$ be the position of the swing foot at time t during the task execution. This trajectory is generated from the skill parameters of the STEP: t_0, t_1, t_f, $\boldsymbol{r}_1$, $\boldsymbol{r}_f$. In the following description, values with regard to positions and attitudes are based on Σ_s, which is the relative coordinate of the support foot.

We first describe the case in which a STEP does not have the mid-point. In this case, let $t_1 = (t_0 + t_f)/2$, and the vertical (z-axis) trajectory of the swing foot is generated by using execution parameter h_s:

$$p_{sw\cdot z}(t) = f_3\langle(t_0, 0), (t_1, h_s), (t_f, 0)\rangle(t). \quad (6)$$

This trajectory might cause a large impact when the foot touches the floor because the vertical velocity of the sole is not adequately reduced before that time. The impact would make the performance of the robot unstable. In order to resolve the problem, if value t_v that satisfies

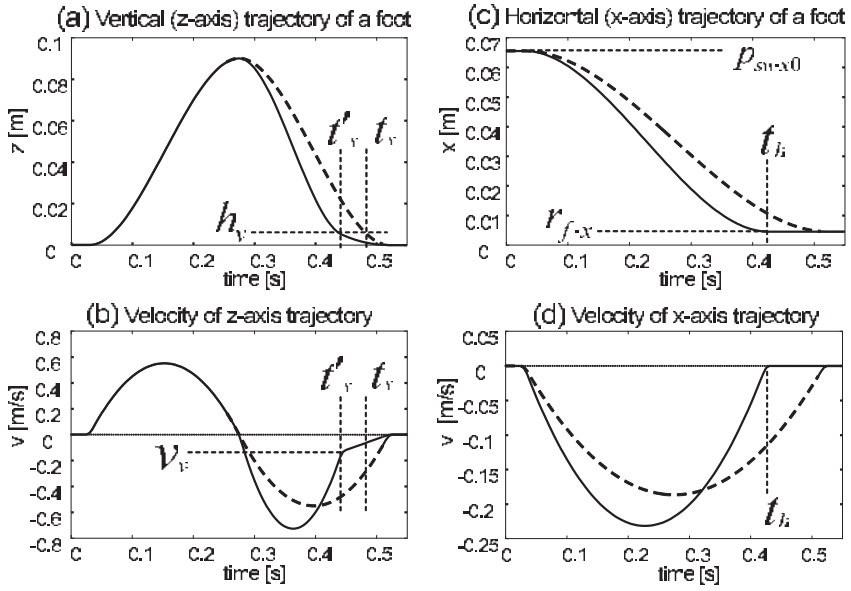

Fig. 7. Graphs of generated foot trajectories. The dashed lines show plain trajectories and the solid lines show trajectories to which the smooth factors are applied.

$$\begin{aligned} p_{sw\cdot z}(t_v) &= h_v, \quad (t_1 < t_v < t_f) \\ \dot{p}_{sw\cdot z}(t_v) &< -v_v, \end{aligned} \tag{7}$$

for execution parameters h_v and v_v exists, the following trajectory is used:

$$t_v{}' = t_f - \frac{2h_v}{v_v},$$

$$p_{sw\cdot z}(t) = \begin{cases} f_3\langle (t_0, 0), (t_1, h_s), (t_v{}', h_v, -v_v)\rangle(t), \\ (t_0 \le t \le t_v{}') \\ \dfrac{v_v^2}{4h_v}(t - t_f)^2. \quad (t_v{}' < t \le t_f). \end{cases} \tag{8}$$

In the trajectory, the velocity becomes v_v at the height of h_v, and in the final segment $(t_v{}' < t \le t_f)$, the velocity is decreasing with a constant acceleration. Graph (a) of Figure 7 shows a trajectory example generated by equation (6) (dashed-line) and that by equation (8) (solid-line). Graph (b) shows their velocities.

Horizontal elements of the foot trajectory are generated as the trajectory from the beginning point to the finishing point by interpolation:

$$p_{sw\cdot x}(t) = f_3\langle (t_0, p_{sw\cdot x}(t_0)), (t_h, r_{f\cdot x}), (t_f, r_{f\cdot x})\rangle(t), \tag{9}$$

where t_h is defined by $p_{sw\cdot z}(t_h) = h_h$ (h_h is an execution parameter). y-element is also generated by a similar function.

Point $(t_h, r_{f\cdot x})$ in equation (9) is inserted to reduce the horizontal impact on landing. In an actual performance, the sole might touch the floor before the horizontal movement of the sole stops. The point resolves the impact by setting a margin between the end of the horizontal movement and the landing. Graphs (c) and (d) of Figure 7 show the difference caused by this point.

When the STEP has the mid-point, (t_1, h_s) in equation (6) or (8) is replaced with $(t_1, r_{1\cdot z})$, and $(t_1, r_{1\cdot x})$ is inserted in equation (9).

The orientation trajectories of the swing foot and the waist are also generated. The trajectory of the swing foot are generated by using an interpolation similar to equation (9) so that the trajectory passes $(t_1,\boldsymbol{R}_1)$, $(t_h,\boldsymbol{R}_f)$ and $(t_f,\boldsymbol{R}_f)$. Our implementation uses the roll-pitch-yaw expression for the interpolation. Note that the roll and pitch elements of $\boldsymbol{R}_f$ are set to zero because the sole plane must be level with the floor when the foot is a support foot. The yaw-axis orientation of the waist, $\psi_{wt}(t)$, is also generated from ψ_f.

Although the trajectory of the horizontal waist position is determined by the ZMP compensation filter, the STEP processor temporarily sets a trajectory that is used as the initial trajectory for the filter. Our implementation uses the following trajectory as the initial one:

$$p_{wt\cdot x}(t) = f_2\left\langle \left(t_0, \frac{p_{sw\cdot x}(t_0)}{2}\right), \left(t_f, \frac{p_{sw\cdot x}(t_f)}{2}\right)\right\rangle(t). \tag{10}$$

The trajectory of the y-axis is also generated in the same way.

The STAND processor changes the support state into the both legs support, and controls the *desired ZMP*, details of which are described in the following subsections.

5.4. ZMP Compensation Filter

The dynamic body balance must be considered in order that a robot should not fall down. In the task generation system, horizontal waist movement is used for this purpose. The movement is determined by the ZMP compensation filter.

The system generates motions under the condition that the whole face of a sole keeps contact with the level, flat floor while supporting the body. Let us call the condition *the contact condition*. If the condition is satisfied in dynamics as well as geometry, the robot will not fall down. The ZMP (Vukobratovic 1990) is useful when considering the condition in dynamics.

It is possible to calculate the ZMP under the assumption that the sole of a support foot extends over the floor. This ZMP is called the *Imaginary ZMP (IZMP)*. The contact condition in dynamics corresponds to the requirement that the IZMP is inside the *support area*, a convex hull of actual sole planes on the floor. Let us call the condition the *ZMP condition*. The problem is how to make the IZMP correspond to a desired ZMP that satisfies the ZMP condition, and has been studied by Nishiwaki et al. (2002) and Kajita et al. (2003).

We implemented the ZMP compensation filter based on the method proposed by Nishiwaki et al. The inputs of the method are an arbitrary desired ZMP trajectory and the IZMP trajectory calculated from a given whole-body motion, and the output of the method is the difference in the horizontal waist position. The difference can be applied by calculating the inverse kinematics between a foot and the waist. As a result, the IZMP is approximately close to the desired ZMP. The method can be iteratively applied until the result converges sufficiently.

Since the filter does not modify the joint angle trajectories of the upper body and the values of the skill parameters, it can preserve the characteristics of dancing motions.

5.5. Desired ZMP Trajectory

The ZMP condition allows various trajectories of the desired ZMP inside the support area. However, actual stability of the robot depends not only on satisfying the condition but also on the ZMP trajectory itself because of physical model errors and disturbances in control. An irregular trajectory causes irregular changes in the whole-body momentum, which is not desirable for stable control. In addition, the ZMP should be the center of the support area as far as possible because this increases the margin for errors and disturbances. Considering these factors, the STAND processor determines desired ZMP as follows.

Let $\boldsymbol{p}_{zmp}(t)$ be the trajectory of the desired ZMP on the floor plane. First, when the support foot of the STEP task executed after a STAND task is different from that executed before it, the STAND processor sets the following trajectory, which moves along the line between the centers of both the support feet:

$$\begin{aligned}\boldsymbol{p}_{zmp}(t) &= \boldsymbol{f}_4\langle(t_0,\boldsymbol{p}_{zmp\cdot 0}),(t_0+t_z,\boldsymbol{p}_{zmp\cdot 0}),\\ &\quad (t_f-t_z,\boldsymbol{p}_{zmp\cdot f}),(t_f,\boldsymbol{p}_{zmp\cdot f})\rangle(t),\end{aligned} \tag{11}$$

where t_0 and t_f are the beginning time and the finishing time of the STAND respectively, $\boldsymbol{p}_{zmp\cdot 0}$ is the center of the support foot in the preceding STEP, and $\boldsymbol{p}_{zmp\cdot f}$ is that in the next STEP. In this trajectory, the ZMP remains still for a moment at the beginning and ending according to execution parameter t_z; this behaviour can increase the stability at the moments when errors in control tend to increase due to the change of support state.

When the condition

$$(t_f - t_0) - 4t_z \geq t_p \tag{12}$$

is true for the execution parameter t_p, the following points are inserted into equation (11):

$$(t_0 + 2t_z, \boldsymbol{p}_c), \quad (t_f - 2t_z, \boldsymbol{p}_c), \tag{13}$$

where $\boldsymbol{p}_c$ is the center of both feet. Using this procedure, the legs make a stable standing pose for a while when a STAND has a certain length of execution time.

When the support foot of the STEP task executed after a STAND task is the same as that executed before it, the STAND processor sets the following trajectory:

$$\begin{aligned}\boldsymbol{p}_{zmp}(t) &= \boldsymbol{f}_5\left\langle(t_0,\boldsymbol{p}_{zmp\cdot 0}),(t_0+t_z,\boldsymbol{p}_{zmp\cdot 0}),\left(\tfrac{t_0+t_f}{2},\boldsymbol{p}_c\right),\right.\\ &\quad \left.(t_f-t_z,\boldsymbol{p}_{zmp\cdot 0}),(t_f,\boldsymbol{p}_{zmp\cdot 0})\right\rangle(t).\end{aligned} \tag{14}$$

In this case, the ZMP moves to the center of the support area and returns although the destination of the desired ZMP is the same as the starting position. Regardless of the desired ZMP, the actual ZMP tends to move towards the swing foot when it lands on the floor because of the reaction force from the floor around the landing position. The desired ZMP trajectory that is close to this behaviour can make the actual control more stable. If the condition (12) is true, a pausing motion is inserted by replacing point $\left((t_0 + t_f)/2,\ \boldsymbol{p}_c\right)$ in equation (14) with the two points in (13), as well as the first case.

In this way, the desired ZMP is determined by the STAND processor. On the other hand, for the period of single-leg support, the desired ZMP remains under the center of the support foot. Since transition to that position is processed by the STAND processor, the STEP processor need not control the desired ZMP.

5.6. Yaw Moment Compensation Filter

When the yaw-axis moment that a robot exerts on the floor exceeds that exerted by the friction between the soles and the floor, the robot spins on the floor. Even when the original human performance does not include spins, a robot might spin.

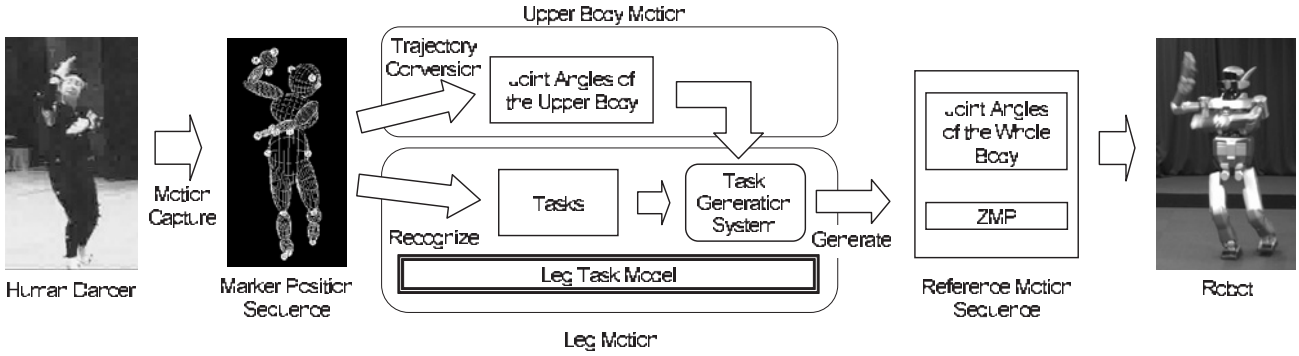

Fig. 8. Process outline. First, a human motion is obtained using motion capture. Then a sequence of leg tasks is recognized from the captured marker trajectories of the lower body. For the upper body, the marker trajectories are converted into robot joint angle trajectories. The task sequence and the joint angle trajectories of the upper body are processed by the task generation system, and the system outputs the robot reference motion data. Finally, the reference data is input into a robot control system, and the robot performs the dance.

Tamiya, Inaba and Inoue (1999) proposed a method for constraining the moment exerted by the whole body within a constant value by a compensation technique. We use the yaw-moment compensation part of the method as a filter in the task generation system in order to generate motions that do not cause spins in the actual robot. The method can use any set of joints for compensation with arbitrary weights, and the filter uses only a yaw-axis joint between the chest and the waist, which is called 'torso yaw' in Figure 6, to compensate the yaw-moment of the whole body. Although the method can deal with only single-leg support, this is not a serious problem because the friction moment is sufficient to prevent spins in most cases when the robot is supported by both legs. Thus the compensation is applied only for the period of the single-leg support.

5.7. Skill Refinement

The task processors and the filters of the whole-body dynamics mainly consider the contact condition described in Section 5.4. However, for other kinds of problem, a generated motion can include faults that make executing the motion on the robot impossible. Possible faults include stepping beyond the movable distance, self-collisions, overruns in joint angle range, and over-speed in angular velocity limit. These faults are due to the fact that the skill parameters obtained from human motions cannot necessarily be executed on the robot because of the mechanical constraints and the difference in body shapes. These faults must be eliminated by adjusting skill parameters on the robot body.

The occurrence of these faults is difficult to determine without executing a motion, because a humanoid robot has a high degree of freedom and complex body shape, and because the motion is generated through many processes with many factors. The task generation system simulates the result of executing a motion by kinematics calculation in Figure 6(c). If faults are detected from the simulation (Figure 6(e)), the system modifies the value of the skill parameters of the task associated with the faults according to given rules defined for each kind of fault. Then the system runs the time frame back to the time before executing the task, and repeats the generation process to eliminate the faults.

In most of the possible faults, solutions of the skill parameters that do not cause the fault can be close to the original values because the value was actually executed by a human dancer. In addition, since the number of skill parameters is not great, the number of candidates for modification is limited. These features make it possible to resolve most faults by simple rules for modifying skill parameters. The rules are constructed so that faults are resolved using the smallest modification possible.

Observing case examples of faults is first required in order to construct concrete rules of skill refinement. In Section 6.4, we present the rules constructed from faults that actually occurred in the dance motions we tested.

6. Experimental Results

In this section, we presented the result of an experiment to verify our framework. The framework produced dance performances of a robot from human dance performances. Figure 8 outlines the generation process.

We used humanoid robot HRP-2 (Kaneko et al. 2004). HRP-2 is a biped humanoid robot consisting of 30-DOF joints. Its size (1.54 m in height) and weight (58 kg) are similar to those of humans.

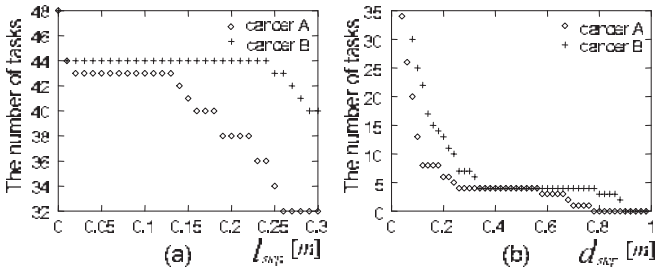

Fig. 9. Correlation between thresholds and detection results. The left graph shows the correlation between l_{step} and the number of STEPs detected. The right graph shows that between d_{step} and the number of STEPs with mid-point.

6.1. Obtaining Human Dance Motions

In the experiment, we tested a traditional Japanese folk dance called "Aizu-Bandaisan". The motions of the dance satisfy the conditions described in Section 3.1. The dance includes many dynamic-style steps with various characteristics.

We used an optical-type motion capture called Vicon for obtaining human motions. The system we used consisted of eight infra-red video cameras and 34 body markers. Three-dimensional positions of the markers were captured at a rate of 120 frames/s.

We captured motions of the dance performed by two dancers: a female dancer A and a male dancer B. They performed in time to the same music. From this data, we extracted the initial 35 seconds for experimental data. The data include four repetitions of a choreography pattern.

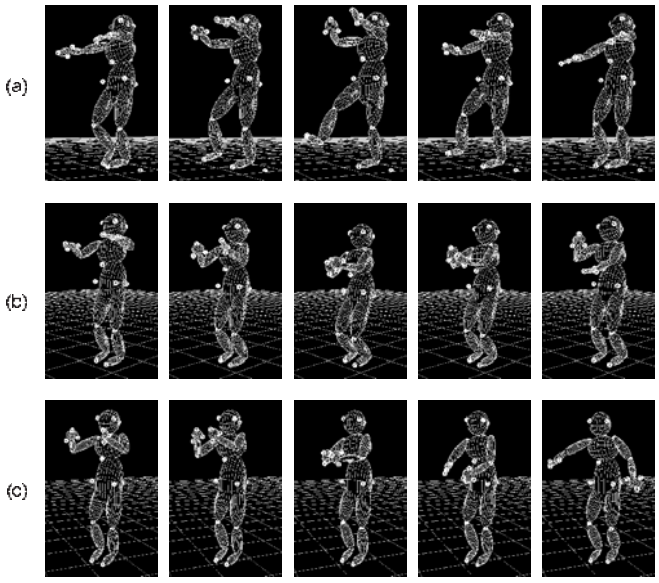

Fig. 10. Motion examples: (a) is a kick-up motion, which was detected as a STEP with mid-point; (b) was detected as a SQUAT; (c) is the motion of another dancer at the same timing as (b), which was not detected as a SQUAT.

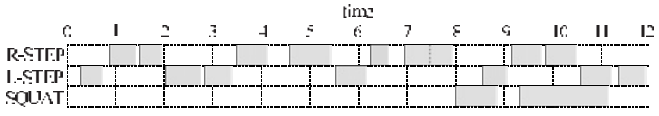

Fig. 11. Part of the task sequence recognized from dancer A. $(0 - 12$ s)

6.2. Results of the Task Recognition

A sequence of leg tasks is recognized from the captured marker trajectories of the lower body, by using the method described in Section 4.

The method requires determining the threshold values described in Section 4. Graph (a) in Figure 9 shows the correlation between the threshold l_{step} and the number of STEP tasks detected. This result was obtained by setting v_{step} to 0.04 m/s. The result indicates that 43 STEPs were detected in dancer A in the range $0.02 \text{ m} \leq l_{step} \leq 0.13$ m, which is a stable result. When l_{step} was less than that range, the result rapidly increased because even noise-like slight motions of a support foot were recognized as tasks. Conversely, if l_{step} was larger, some of the true STEPs were eliminated. Likewise, 44 STEPs were stably detected in dancer B in the range $0.02 \text{ m} \leq l_{step} \leq 0.24$ m. We determined l_{step} to be 0.04 m to correctly detect STEP tasks.

d_{step} is the threshold for judging whether the swing foot makes a characteristic pose in the middle of a stepping motion or not. Graph (b) in Figure 9 shows the correlation between d_{step} and the number of STEPs with mid-point. This indicates that four is the stable result for both dancer A and dancer B. We determined d_{step} to be 0.4 m from this result. With this value, the number of STEP tasks with mid-point corresponded with the number of motions like a kick, an example of which is shown in Figure 10(a).

The threshold l_{squat} determines resolution of vertical waist motion. We set l_{squat} at 0.12 m. For this value, the motion of dancer A shown in Figure 10(b) was recognized as a SQUAT. At the same timing, dancer B performed the motion shown in Figure 10(c), which was not recognized as a SQUAT. This would be a proper result.

Figure 11 shows a part (from the beginning to 12 seconds) of the task sequence recognized from dancer A.

Note that the difference between dancer A and B in the number of detected tasks is due to the difference in choreography details.

6.3. Converting Upper Body Motions

For upper body motions, we used a conventional way of adapting motion trajectories, because it was sufficient for the purpose of verifying our framework, which focuses on leg motions. The outline of the adaptation process is as follows.

First, each joint angle of the robot is determined from positions of the several markers related to the joint, and trajectories of the joints are acquired. Then the trajectories are transformed so that they are under the joint limits with respect to angle range and angular velocity.

Angle range is constrained by a scaling method. The scaling is partially applied to the temporal blocks where the value is over the limit. Angular velocity is constrained by a method proposed by Pollard et al. (2002). This method preserves timings of global oscillations well.

In this way, the captured marker trajectories of the upper body were automatically converted into the joint angle trajectories that can be executed on HRP-2.

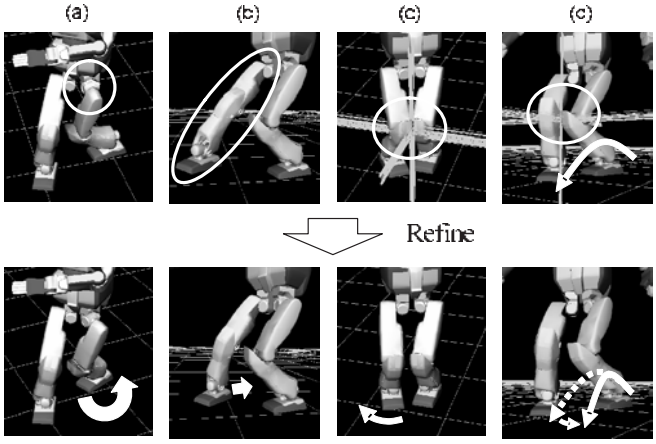

Fig. 12. Fault examples: (a) is an overrunning of the angle limit of a coxa yaw joint; (b) is an overrunning of the possible step distance; (c) is a self-collision between the knee joints; (d) is a self-collision during stepping.

6.4. Results of the Task Generation

The task generation system requires determining the appropriate values of the execution parameters for a robot.

Parameter h_w, which represents the normal height of the waist, is closely related to the problem of a singular point in the leg joint structure. The generation system does not suppose the singular point in which the knee is fully extended. Parameter h_w must be lower than the waist height when extending the legs, which is 0.71 m in HRP-2. A value close to the height of the extended pose makes the possible stepping distance shorter, and the scale of original steps cannot be sufficiently realized. Conversely, an excessive low height makes leg poses too different from those in human performances. We determined h_w to be 0.61 m considering these respects.

The other parameters were determined as follows: h_s was 0.05 m, t_p was 0.4 s, and Δt was 0.005 s. The values of the remaining parameters were determined through executing motions on the robot, and is described in Section 6.5.

For the skill refinement described in Section 5.7, the faults shown in Figure 12 were observed in the original skill values.

(a) An overrunning of the angle limit of a coxa yaw joint. This fault can be avoided by modifying the parameter ψ_f of a STEP task within the joint limit.

(b) An overrunning of the possible step distance. In this case, the swing foot cannot reach the goal position $\boldsymbol{r}_f$, and the knee joint gets into a singular point. This fault can be avoided by modifying $\boldsymbol{r}_f$ towards the start position of the swing foot in a STEP. When this fault occurs on the way to the mid-point, the fault is avoided by moving $\boldsymbol{r}_1$, the position of the mid-point, in the direction that shortens the distances to both the start and finish positions of the swing foot. This fault is detected by checking whether the angles of the knee joints are close to the singular point; angles less than θ_s are recognized as a fault. We determined θ_s to be 6 deg.

(c) A self-collision between the knee joints. This fault can be avoided by modifying the yaw-element of $\boldsymbol{R}_f$, which is the finishing attitude of the swing foot in the last STEP.

(d) A self-collision during stepping. This fault can be avoided by rotating $\boldsymbol{r}_f$ and $\boldsymbol{R}_f$ away from the support foot on the coordinate of the swing foot at the start of the STEP.

In addition to these faults, overruns of the angular velocities in the knee joints were observed. This fault is avoided by applying the same modification as in (b).

In dancer A, the following numbers of faults were observed: (a) 3, (b) 8, (c) 1, and (d) 2. The results for dancer B were, (a) 0, (b) 20, (c) 2, and (d) 0. A number of (b) faults were observed in dancer B, because dancer B tended to step with long strides. Note that the frequency of fault (b) is a trade-off against the normal waist height h_w; a lower h_w reduces the frequency of this fault. After resolving these faults, a number of overruns of the angular velocities in the knee joints were still observed; the number was 14 in all 43 STEPs in dancer A, and 11 in all 44 STEPs in dancer B.

The ways of avoiding the faults described above have been implemented as rules of the skill refinement. The implementation automatically resolved all the faults observed in the experimental data.

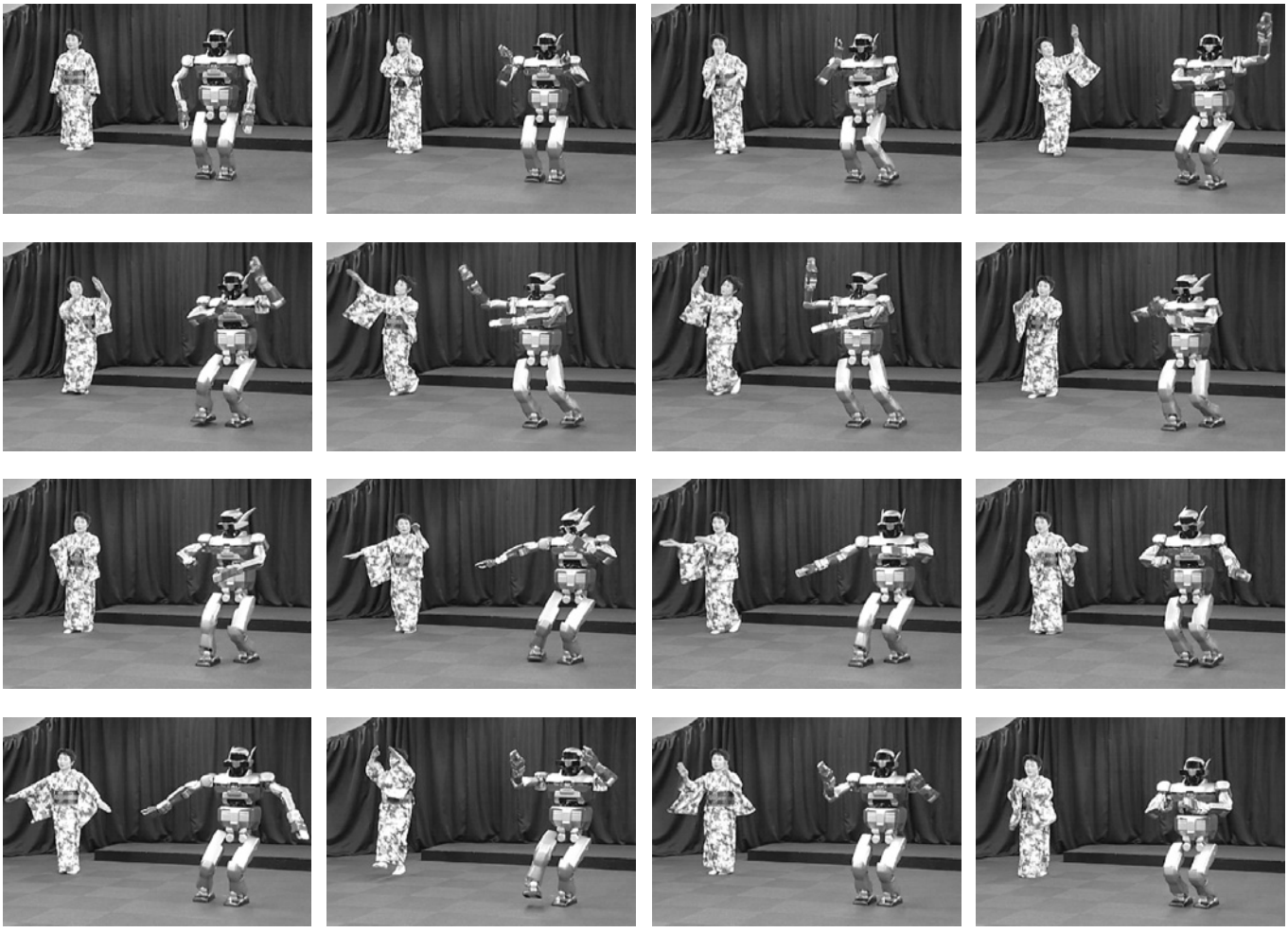

Fig. 13. Dance performance of Aizu-Bandaisan by HRP-2 and a human master (dancer A). The motion of HRP-2 imitated the motion captured from the dancer, and the dancer replayed the dance. HRP-2 successfully performed the imitative dance motion while keeping the original music tempo. (A video file of the demonstration is available at http://www.cvi.iis.u-tokyo.ac.jp/video.html.).

For dynamic balance, the output of the ZMP compensation filter successfully converged. The resulting ZMP trajectories corresponded to the desired ZMP trajectories.

As a result, the task generation system generated the reference motion data that satisfies both the mechanical constraints and the dynamic balance of HRP-2.

6.5. Dance Performances by HRP-2

We used a control system of *OpenHRP* (Kanehiro, Hirukawa and Kajita 2004; Yokoi et al. 2001) to apply the generated motion data to HRP-2. The system basically consists of a sequence controller and a stabilizer. For each joint, the sequence controller tracks a given joint angle trajectory with the PD control. In addition, the stabilizer slightly modifies the horizontal waist position in order to correct errors between a given reference ZMP and an actual ZMP obtained by force sensors. This kind of feedback stabilization control is necessary due to disturbances and model errors.

The execution parameter h_h,h_v,v_v and t_z directly affect the stability of a robot. Their values were determined from the actual behaviour. We obtained the following values to produce stable performances on HRP-2: $h_h = 0.006$ m, $h_v = 0.005$ m, $v_v = 0.13$ m/s, $t_z = 0.025$ s respectively.

Finally, HRP-2 successfully performed the generated motions. The performances were sufficiently stable because the sole of a support foot was kept flat on the floor. Figure 13 shows the performance by HRP-2 and dancer A. In this performance, the motion of HRP-2 imitated the motion captured from the dancer, and the dancer replayed the dance.

7. Discussion

As described in Section 1, it is more difficult for a robot to learn motions when the physical features of the robot are different from those of a human instructor. Though HRP-2 does have different features from the instructor, the learning of motions should satisfy the following requirements.

- The motions should be stably performed by the robot.
- The motions should keep the important characteristics of the original motions.

From these viewpoints, this section discusses the validity of our framework based on the experimental results.

7.1. Stability

The experimental results verified that the framework correctly worked to produce dynamically stable motions in spite that the configuration and mass distribution of HRP-2 are significantly different from those of the human instructors and that the rigid feet of HRP-2 cannot fit the soles to the floor in the same way as human feet do. The results were realized by the leg task models that can provide the sequence of the desired contact states between the feet and the floor by which dynamically stable motions of the robot can be generated and the approaching trajectories of the feet can be planned to fit the soles to the floor smoothly. Dynamic balance was kept by controlling the horizontal position of the waist.

In the current implementation, several execution parameters have to be determined heuristically to generate the motions as described in Section 6.4 and 6.5, which is additional work other than the demonstrations, and may reduce the advantage of LFO. Since an identical set of parameters was used to generate the valid motions from the motions of different dancers A and B, we expect that other motions would be successfully processed by a set of parameters tuned to the generic motion. In order to verify this assumption, more tests on various kinds of dance motions should be carried out.

The stability discussed above assumes that the joints can follow the generated trajectories, but the assumption does not hold when a desired joint angle or velocity is outside the admissible range, or a set of joint angles causes a self-collision. In fact, HRP-2 caused a number of range overruns and self-collisions as shown in Section 6.4. In order to reduce such faults, the hardware of the robot should certainly be improved. At the same time, the framework should be able to generate feasible motions for a given robot whose physical ability is more limited than that of humans in order that the framework can be used for various humanoid robots including existing ones.

The problem was successfully solved by skill refinement based on leg task models. An admissible set of joint trajectories can be found by searching the feature space of the skill parameters. However, the rules for skill refinement were implemented heuristically after observing experimental data, and it is not guaranteed that the current implementation would be valid for arbitrary motions. A more rigorous method should be constructed in future work.

7.2. Keeping Important Characteristics

It is not possible to reproduce human motions completely using a robot whose physical constraints are different from those of the original performer. Better reproduction can be achieved by giving higher priority to factors that vary the important characteristics of a dance. The task models were designed so that they can clearly represent those factors with high priority.

In general, keeping the rhythm of the motion is one of the most important factors for judging whether a dance performance is skillful or not. Our framework puts the highest priority on the temporal characteristics of each task, which are represented by skill parameters t_0, t_1 and t_f. The original values of these parameters were maintained throughout the generation process. This invariance enabled HRP-2 to achieve high fidelity performance in rhythm.

High priority is also put on the position and orientation of a support foot and vertical motion of the waist. They are basic factors for making a pose of the lower body. The task generation system tries to keep them as close as possible to the original ones while preserving the temporal characteristics. On the other hand, low priority is set on the details of foot trajectories.

The resulting performances by HRP-2 were highly appreciated by dance specialists. A grand master, dancer A, commented that the performances were skillful and that she was clearly able to recognize the characteristics of the dance depending on the individual dancers, from the reproduced leg motions of the robot. However, it is a subjective evaluation for a specific case, and it is desirable to establish an objective method to give a quantitative evaluation of the quality of the reproduction in future work.

8. Summary

The contribution of the paper is summarized as follows.

A framework that achieves the LFO paradigm for the dance tasks of biped humanoid robots was proposed. The framework is based on leg task models, which provide the information necessary to perform leg motions.

The implementation of the framework was presented. A method for recognizing the leg tasks from human motion data and a method for generating robot motion data from the recognized tasks were developed.

The framework was verified by experiments on HRP-2 using a traditional folk dance. The results illustrated that the

framework is valid and maintains stability and retains important characteristics of leg motions.

The result that the robot stably imitated human dance motions including dynamic-style steps while keeping the original motion rhythm is a novel achievement for biped humanoid robots.

Acknowledgement

This work is supported in part by the Japan Science and Technology Agency (JST) under the Ikeuchi CREST project, and in part by the Grant-in-Aid for Scientific Research on Priority Areas (C) 16016218 of the Ministry of Education, Culture, Sports, Science and Technology.

We are very grateful to the dance group "Aizu Gyokusui-kai" for presenting their dance performances.

References

Ikemoto, L., Arikan, O. and Forsyth, D. (2006). Knowing when to put your foot down. *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, New York, pp. 49–53, ACM Press.

Ikeuchi, K. and Suehiro, T. (1994). Toward an assembly plan from observation. Part I: Task recognition with polyhedral objects. *IEEE Transaction on Robotics and Automation*, **10**(3): 368–385.

Kagami, S., Kanehiro, F., Tamiya, Y., Inaba, M. and Inoue, H. (2000). AutoBalancer: an online dynamic balance compensation scheme for humanoid robots. *Proceedings of Fourth International Workshop on Algorithmic Foundations of Robotics*, pp. SA-79–SA-89.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., et al. (2003). Biped walking pattern generation by using preview control of zero-moment point. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 1620–1626.

Kanehiro, F., Hirukawa, H., Kajita, S. (2004). OpenHRP: open architecture humanoid robotics platform. *International Journal of Robotics Research*, **23**(2): 155–165.

Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., et al. (2004). Humanoid robot HRP-2. *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, pp. 1083–1090.

Kang, S. B. and Ikeuchi, K. (1997). Toward automatic robot instruction from perception–mapping human grasps to manipulator grasps. *IEEE Transactions on Robotics and Automation*, **13**(1): 81–95.

Kosuge, K., Hayashi, T., Hirata, Y. and Tobiyama, R. (2003). Dance partner robot -Ms DancerR-. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, pp. 3459–3464.

Kuniyoshi, Y., Inaba, M. and Inoue, H. (1994) Learning by watching: extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, **10**(6): 799–822.

Kuroki, Y., Blank, B., Mikami, T., Mayeux, P., Miyamoto, A., Playter, R., et al. (2003). Motion creating system for a small biped entertainment robot. *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, pp. 1394–1399.

Kuroki, Y., Fujita, M., Ishida, T., Nagasaka, K. and Yamaguchi, J. (2003). A small biped entertainment robot exploring attractive applications. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 471–476.

Nakazawa, A., Nakaoka, S., Kudoh, S. and Ikeuchi, K. (2002). Digital archive of human dance motion. *Proceedings of the Eighth International Conference on Virtual Systems and MultiMedia*, Gyeongju, Korea, pp. 952–960.

Nishiwaki, K., Kagami, S., Kuniyoshi, Y., Inaba, M. and Inoue, H. (2002). Online generation of gumanoid walking motion based on a fast generation method of motion pattern that follows desired ZMP. *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Swizerland, pp. 2684–2689, EPFL.

Pollard, N. S., K.Hodgins, J., Riley, M. J. and Atkeson, C. G. (2002). Adapting human motion for the control of a humanoid robot. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC, pp. 1390–1397.

Riley, M., Ude, A. and Atkeson, C. G. (2000). Methods for motion generation and interaction with a humanoid robot: case studies of dancing and catching. *Proceedings of AAAI and CMU Workshop on Interactive Robotics and Entertainment 2000*, Pittsburgh, PA, pp. 35–42.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Science*, **3**: 233–242.

Shin, H. J., Lee, J., Shin, S. Y. and Gleicher, M. (2001). Computer puppetry: an importance-based approach. *ACM Transactions on Graphics*, **20**(2): 67–94.

Takamatsu, J., Morita, T., Ogawara, K., Kimura, H. and Ikeuchi, K. (2006). Representation for nnot-tying task. *IEEE Transactions on Robotics*, **22**(1): pp. 65–78.

Tamiya, Y., Inaba, M. and Inoue, H. (1999). Realtime balance compensation for dynamic motion of full-body humanoid standing on one leg. *Journal of the Robotics Society of Japan*, **17**(2): pp. 268–274 [in Japanese].

Vukobratovic, M., B.Borovac, , D.Surla, and Stokic, D. (1990). *Biped Locomotion: Dynamics, Stability, Control and Application*, Vol. 7 of *Scientific Fundamentals of Robotics*, Springer-Verlag.

Yamane, K., Hodgins, J. K. and Brown, H. B. (2003). Controlling a marionette with human motion capture data. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, pp. 3834–3841.

Yamane, K. and Nakamura, Y. (2003). Dynamics filter – concept and implementation of online motion generator for human figures. *IEEE Transaction on Robotics and Automation*, **19**(3): 421–432.

Yokoi, K., Kanehiro, F., Kaneko, K., Fujiwara, K., Kajita, S. and Hirukawa, H. (2001). A Honda humanoid robot controlled by AIST software. *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, Tokyo, pp. 259–264.