# We are IntechOpen,
# the world's leading publisher of
# Open Access books
# Built by scientists, for scientists

## 6,900
Open access books available

## 186,000
International authors and editors

## 200M
Downloads

## 154
Countries delivered to

Our authors are among the

## TOP 1%
most cited scientists

## 12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
**BOOK CITATION INDEX**
INDEXED

**WEB OF SCIENCE**™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

# Interested in publishing with us?
# Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

# Humanoid Robotic Language and Virtual Reality Simulation

Ben Choi
*Louisiana Tech University*
*USA*

## 1. Introduction

This chapter describes the development of a humanoid robotic language and the creation of a virtual reality system for the simulation of humanoid robots. In this chapter we propose a description language for specifying motions for humanoid robots and for allowing humanoid robots to acquire motor skills. Locomotion greatly increases our ability to interact with our environments, which in turn increases our mental abilities. This principle also applies to humanoid robots. However, there are great difficulties to specify humanoid motions and to represent motor skills, which in most cases require four-dimensional space representations. We propose a representation framework that includes the following attributes: motion description layers, egocentric reference system, progressive quantized refinement, and automatic constraint satisfaction. We also outline strategies for acquiring new motor skills by learning from trial and error, macro approach, and programming.

Then, we use our new humanoid motion description language and framework as the base to build a virtual reality system to simulate humanoid robots. Currently most humanoid simulation systems require manual manipulations of body parts or require capturing movements enacted by a person. We aim to describe humanoid motions using our high-level humanoid motion description language. We defined new motion primitives and new syntax that allows programmers to easily describe complex humanoid motions. To make the motion specification possible, we defined a humanoid framework that models humanoid robots by specifying their capacities and limitations. Furthermore, we developed a simulation system that can execute the humanoid motion description language and can automatically handle conflicting motion statements and can ensure that the described motions are within the limitations of humanoid robots. Our simulation results show that the proposed system has great future potentials.

The remaining of this chapter is organized as follows. Section 2 outlines the related research on high-level language approaches to describe and to simulate humanoid motions. Section 3 describes the motives for defining a humanoid motion description framework, which includes methods for specifying humanoid motions and methods for acquiring new motor skills. Section 4 outlines the methods for specifying humanoid motions, which include the concepts of motion description layers, egocentric reference system, progressive quantized

refinement, and automatic constraint satisfaction. Section 5 outlines two methods for acquiring new motor skills: learning from trial and error and learning by macro approach. Section 6 describes the motives for developing a system to simulate humanoid robots in virtual reality environments. Section 7 defines a new humanoid motion description language called Cybele. It focuses on the syntactic aspects of the language, while Section 8 focuses on the semantic aspects of the language and defines the framework on which the language can be interpreted. Section 9 provides the implementation details of the humanoid simulation system. And, Section 10 gives the conclusion and outlines the future research.

## 2. Related Research

Research in describing humanoid motions begins with the works for describing human dances. Popular dance notation systems include Benesh (Causley, 1980), Labanotation (Hutchinson & Balanchine, 1987), and EW (Eshkol-Wachman, 2008). Benesh is the simplest one and is designed particularly for dance description. Labanotation is more comprehensive for describing human motion in general. EW can be applied on linkage systems other than human body. Computers are now used to aid the interpretation and visualization of these notations (Ryman et al., 1984; Adamson, 1987; Calvert et al., 1993; Schiphorst, 1992). Researchers used Labanotation as a basis to represent human motion, proposed to extract key motion primitives, and proposed architectures for digital representation of human movements (Badler et al., 1979). Another approach uses natural language; such as "Improv" system used natural language to script human behaviour interacting in virtual environments (Perlin & Gikdberg, 1996). Motion sequences can be generated by system that employs human biomechanical logic (Badler et al., 1994). This section outlines related work on the high-level language approaches to humanoid simulation (Nozawa et al., 2004; Nishimura et al., 2005). Several systems will be discussed, which including, Poser Python, VRML (Virtual Reality Modelling Language), Improv, STEP, and others. It focuses on high-level language approaches to humanoid simulation and omits other general concurrent languages such as OCCAM.

Poser Python (Python, 2008; Schrand, 2001) is an implementation of the Python interpreter that includes many commands that have been extended to recognize and execute commands not included in the standard Python language. Poser Python script language is a language combination that uses syntax and basic logic of Python and special commands tailored especially for Poser scene, manipulate them, and finally send them back to Poser. The language-controlled animation is a significant advantage of Poser-Python system.

VRML (Virtual Reality Modelling Language) is a scene-description language used widely on the internet. VRML uses TimeSensor to initiate and synchronize all the motions in the scene. It is possible that asynchronously generated events arrive at the identical time as one or more sensor-generated event. In such cases, all events generated are part of the same initial event cascade, and each event has the same timestamp. Based on this mechanism, VRML is quite suitable to visual presentations with user interactions. However, there is no direct way to describe complex motions with time overlapping.

Improv (Perlin & Goldberg 1996) is a system for the creation of real-time behaviour based on animated actors. Improv consists of two subsystems. The first subsystem is an animation engine that uses procedural techniques to enable authors to create layered, continuous, non-

repetitive motions and smooth transitions between them. The system uses an English-style scripting language so that creative experts who are not primarily programmers can create powerful interactive applications.

STEP (Huang et al., 2002) is a distributed logic program being used to define and control the hand gestures of embodied agents in virtual worlds. It works as an interface between the constructs of logic programming and the humanoid model defined in VRML. Using the STEP framework, different gesture dictionaries can be defined, and variants of a hand gesture, according to dynamically changing factors, can be generated on the fly. STEP also has two kinds of control logic definitions. One is parallel and the other is sequential. But different kinds of program blocks cannot be mixed together and must be executed one by one.

There are also other systems for animations such as Alice (Conway, 1997), Maya, Lightwave, and 3D Studio Max. They each have a beautiful user interface and have easy drag and drop functionality for the animation design and customization. However, most of them lack underlining programming languages that programmer can program to control various motions. Meanwhile, current motion description languages do not have motion synchronization at the language level. The details of the motion control make the simulation difficult to implement and debug at the language level. This also makes the motions non-reusable.

## 3. Humanoid Motion Description Framework

Locomotion greatly increases our ability to interact with our environments, which in turn increases our mental abilities. The principle that mental abilities can be improved by interacting with the environments is the basis for MIT Cog's project (Brooks et al., 1998; Arsenio, 2004). However, Cog robot currently lacks locomotion. On the other hand, Honda humanoid robots (Honda, 2008) possess the state of the art locomotion system, but lack the autonomy and the learning abilities. We envision the union of these two types of robots, such as Albert HUBO (Oh et al., 2006), as the basis of our investigation.

The humanoid robots of the near future will possess the abilities for locomotion, autonomy, and learning (Brooks, 2002; Arsenic, 2004; Burghart et al., 2005; Yokoi, 2007). Much research remains to be done on such autonomous humanoid robots (Brooks 1996; Scassellati 2001). In this chapter, we will focus on issues of developing a common framework for both specifying motions and for autonomously acquiring motor skills for such robots.

A unified framework to address both specifying motions and acquiring motor skills will facilitate the developments of autonomous humanoid robots. Neural Network, for example, may be a good medium for capturing and classifying motor skills. However, the resultant representation in terms of matrix of weights of connectivity is difficult to be interpreted and modified. Thus, in this investigation we choose to use symbolic approach by developing a description language.

Our humanoid motion description language, like any other languages, consists of syntactic and semantic aspects. Syntactic aspect specifies rules for combining words while semantic aspect specifies structures for interpretation and such provides the meaning. We propose different set of words and rules for different level of abstraction, such as using joint angles at

the low level and using "walk" and "jump" at the high level of abstraction. The interpretation and the meaning are based on our framework that includes egocentric reference system, progressive quantized refinement, and automatic constraint satisfaction.

Our language and our framework (Choi & Chen, 2002) are unique in many ways comparing to other related research (Kanehiro et al. 2004). Our reference system simplifies specification of locomotion and allows motions to be described by uniform and deterministic expressions. Our concept of Progressive Quantized Refinement allows a humanoid robot to interact with its environments using different level of granularity. Our Automatic Constraint Satisfaction system reduces the complexity of specifying humanoid motions. Moreover, our underlining model using non-deterministic finite state machines allows humanoid robots to learn new motor skills.

## 4. Specifying Humanoid Motions

The proposed language and framework for specifying humanoid motions includes the following attributes: motion description layers, egocentric reference system, progressive quantized refinement, and automatic constraint satisfaction, each of which is described as follows.

### 4.1 Motion Description Layers

Table 1 outlines the concept of motion description layers. Each description layer is a level of abstraction. Joint Angle layer describes a motion in terms of changes in the joint angles, such as knee joint rotate to 30 degree or elbow joint rotate to 45 degree. This layer provides detail and precise information that can readily be used to control various actuators of a robot. Path layer describes a motion in terms of a connected line that is specified by points. A simple path can be specified using two points, such as Hand (v1) that moves hand from current position to point v1. More points provide more detail specification of the path; for example, Foot (v1, v2) denoted that foot moves from current position through v1 to v2.

Motion primitive (Nakaoka et al., 2004) layer describes a motion in terms of a given set of essential motions that can be combined to form more complex motions. The set of essential motions must first be identified. It must be complete so that we can describe all possible motions of a humanoid robot. We must also provide a set of rules for specifying how one motion primitive can be combined with another. In effect, we are creating a formal language and insuring that the language is both complete and consistent. This is an axiomatic approach to describe humanoid motions.

Motion sequence layer describes a sequence of motions in terms of motion blocks such as walk, run, jump, and turn. Using this high-level description, we can describe a complex task

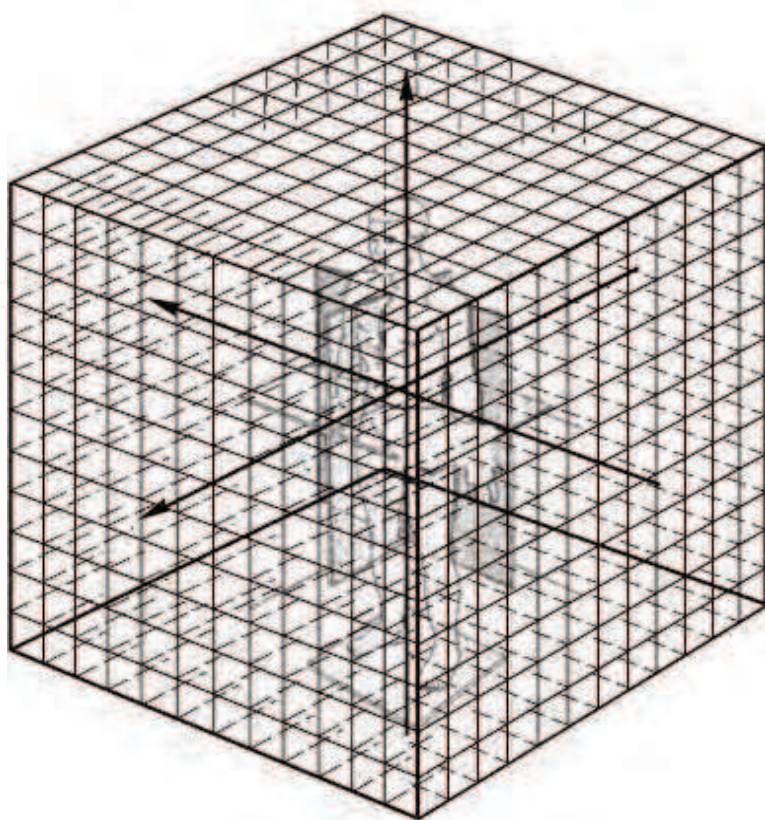| Description Layer | Example |
|---|---|
| Motion Sequence | Walk, Run, Jump, Turn |
| Motion Primitive | Raise, Lower, Forward, Backward |
| Path | Hand (v1), Foot (v1, v2) |
| Joint Angle | Knee Joint 30, Elbow Joint 45 |

Table 1. Motion Description Layers

Fig. 1. Egocentric Space Reference System

with ease without having to specify the angle of each joint. However, this high-level description is not as precise as low-level description and thus leaves must room for interpretation that is addressed in this investigation by using Progress Quantized Refinement discuss in Section 4.3.

### 4.2 Egocentric Reference System

We proposed an egocentric reference system for specifying space-time in discrete finite four-dimensional hyperspace. Each point in our reference system is represented by a quintuple (x, y, z, t). Each of the variables, x, y, z, and t, is an integer ranging from –128 to +127. The origin of the reference system locates at (0, 0, 0, 0). In short, each point in our reference system can be stored using four bytes or 32 bits.

Our reference system is egocentric in that the origin of space is located at the center of the torso of a humanoid robot, as denoted in Figure 1. The origin of time is located at the beginning of a state transition.

In our system, a motion is defined by a sequence of state transitions. Each state transition begins at time 0 and must be completed in 127 time units or less. Negative time units represent the time units used during the last state transition. Each state transition begins with the origin of space located at the center of the torso. In short, a state transition begins at (0, 0, 0, 0). All changes during a state transition are specified within the egocentric reference system.

Translation between the egocentric reference system and its world reference system is done at the end of each state transition. For example, beginning at a standing position as shown in
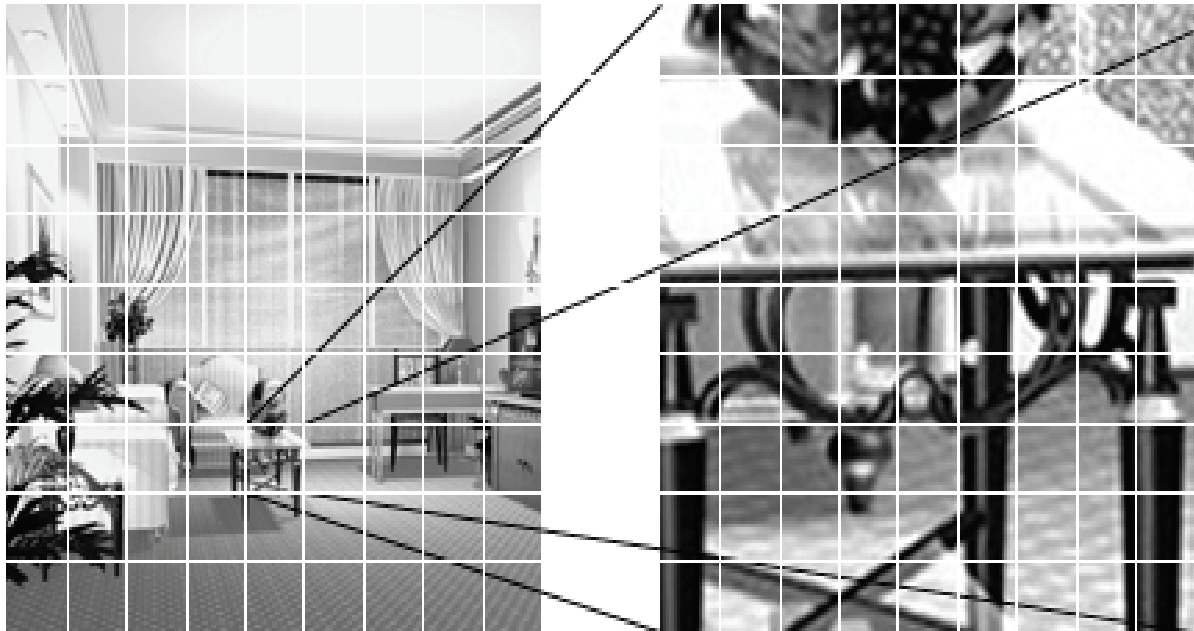


Fig. 2. Concept of Progressive Quantized Refinement

Figure 1, the robot moved three units forward in positive y-axis direction and completed at a standing position, and the whole transition takes 9 units of time. Now, the center of the torso is located at (0, 3, 0, 9). Assuming at the beginning of the transition R(0, 0, 0, 0) in the robot's egocentric reference system is located at W(3, 2, 4, 2) in its world reference system. Also assume that y-axes of both systems are parallel and have the same direction, and each unit in the egocentric reference system represents 2 units in the world reference system. To reset R(0, 3, 0, 9) back to R(0, 0, 0, 0), we makes R(0, 0, 0, 0) now to corresponding to W(3, 2+3*2, 4, 2+9*2).

### 4.3 Progressive Quantized Refinement

We proposed a concept called Progressive Quantized Refinement for a humanoid robot to interact with its environments using different level of granularity. Figure 2 illustrates the concept; on the left picture a 9x9 unit squares is used to display a room while on the right picture the same sized 9x9 unit squares is used to display part of a table. For a robot to put an object on the table, the robot can first use the left picture to move toward the table. Then, it can use the right picture to put the object on the table.

At different states a robot can change its unit scale factor as needed. For example, a unit length in the robot's egocentric space reference system can be scaled to 1 cm, 1 inch, or 1 meter in its world reference system. A unit time can be scaled, for example, to 1 second, 1 minute, or 5 minutes.

### 4.4 Automatic Constraint Satisfaction

We proposed to use Automatic Constraint Satisfaction to reduce the complexity of specifying humanoid motions. There are many implicit requirements for locomotion, such as maintaining balance and structural integrity. Automatic constraint satisfaction system will provide additional changes to meet the implicit requirements.

A system for providing automatic constraint satisfaction for locomotion is very complex and much research is being done on areas such as motion planning and constraint satisfaction. For example, we can simply specify that the robot must move its right hand from current position (3, 6, 2, 0) to new position (3, 50, 2, 6). The simpler the specification, in most cases, requires the more complex constraint satisfaction. In our example, the hand must reach the new position using 6 units of time, so that speeds for various actuators must be adjusted to meet this requirement. If the hand cannot reach the new position by simply raising it and reaching out, then the robot must move the whole body toward the new position.

## 5. Acquiring New Motor Skills

The ability for acquiring new motor skills is essential for mental developments. The trivial approach is to simply program a humanoid robot for new required motor skills (Ude et al., 2004), which can easily be done by an experienced programmer using our proposed language and framework. Thus, in the following we will focus on strategies for acquiring motor skills through learning from trial and error and learning by macro approach.

Learning motor skills has not yet been a central focus of Machine Learning researchers. Thus, much research remains to be done on automatic acquiring new motor skills. We briefly outline strategies for creating such a system, which in part is based on the author's work on automata for learning sequential tasks (Choi, 1998; 2003).

### 5.1 Learning from Trial and Error

One way for acquiring new motor skills is by trial and error. This approach requires first identifying an objective and then selecting actions or motions to achieve the objective. In particularly, using our proposed framework, identifying an objective can be described as identifying a goal state, while selecting actions or motions can be described as selecting a sequence of transitions from the current state to the goal state.

Using our proposed framework, the underlining model is a non-deterministic finite state machine (Choi 2002). From one state, there may be several transitions to follow to other states. Choosing one transitions or the other is why we call this a trial and error approach and is why it is non-deterministic. To achieve the objective is to find a sequence of transitions from the current state to the goal state. As soon as a sequence of transitions is found, it can be stored for future use.

### 5.2 Learning by Macro Approach

Macro approach can be used in a supervised learning environment (Bentivegna & Atkeson, 2001; Arsenic, 2004; Calinon et al. 2007). After a humanoid robot is repeatedly instructed to perform certain task, the robot can store the sequence of motions to associate with a name of the task. After which, the robot can be commanded to perform the sequence by simply specifying the name of the task. This is a simple record and play back approach.

A more sophisticated approach is provided by the author (Choi, 2002; 2003), in which a robot can build a non-deterministic finite state machine based on the repeated instructions to perform certain task. The resulting non-deterministic finite state machine can then be used for other purposes such as for learning from trial and error as discussed above.

## 6. Virtual Reality Simulation of Humanoid Robots

We developed a humanoid motion description language, called Cybele, based on the above proposed framework. Our development process in turn enhances the strength of our framework. We also developed a virtual reality system to simulate humanoid robots (Zhou & Choi, 2007). Virtual reality simulation of humanoids has been an important subject due to its wide range of applications such as motion picture production and robotics (Kanehiro et al., 2001). To achieve realistic motions, programmers currently must be highly experienced to define all control factors for the movements of individual body parts on each time instance. Since it is far too difficulty to control the detail moments of each individual body parts, currently motion capture systems are widely used to record motions enacted by a person and then to use the captured motions to animate humanoids (Riley & Atkeson 2000; Safonova et al., 2002; Nakaoka et al., 2003; Ude et al., 2004).
We attempted a high-level and goal-oriented approach. As shown in Figure 3, there are two extremes on the approaches to humanoid simulations, the goal-oriented and the motion-oriented. The goal-oriented approach specifies the objectives using high-level languages, that are easy to write, and that use Artificial Intelligent methods to generated detailed motions, but that are much more difficulty in terms of building the system. On the other hand, the motion-oriented approach can specify detail motion using degree of freedom (DOF) of each join. However, it is more complicated to write such specifications and programmers need to provide all the details, but such a system is earlier to be built.

In this chapter we present a new high-level goal-oriented language named Cybele. The language is designed for humanoid motion description (Choi & Chen, 2002) and is independent of virtual reality simulation systems. Our new language provides simple syntax for expressing parallel and sequential processing as well as the combination of these two. We also introduce syntax for expressing motions. In particular, to solve the motion mixing problem in parallel and complex blocks, we present an approach to synthesizing motions with combination of partial states. We define key poses as states or partial states and create primitives between two key states. We also extract a relatively small set of parameterized motions from infinite complex motion sequences and make them reusable as primitive motions. Our simulator interprets the programs, breaks down motions into primitives with time, scope, and priority. Final motion sequences are then generated using a synchronization approach.

To achieve constraints satisfaction, we define the scope and priority for each joint in the humanoid. This system checks constraints on the joints affected by the motion and determines which motion is in conflict with some others. In addition, to create a multi-platform system, we implement a prototype system with two parts following the same paradigm as the Java virtual machine. The first part interprets the program and generates motion sequences. The second part translates motion primitives to interface with systems such as virtual reality simulation systems, animation systems, or robotic systems (Kanayama & Wu, 2000; Hirai et al., 1998; Sony, 2008).
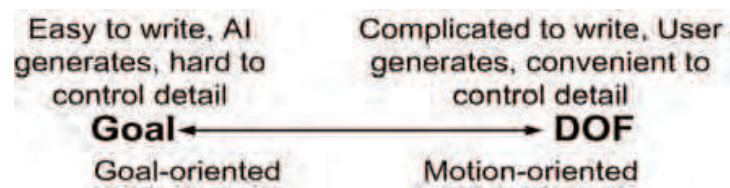
Fig. 3. Two extremes of motion languages

# 7. Humanoid Robotic Language Specification

Our new humanoid motion description language is called Cybele (Choi & Chen 2002). The features of Cybele language are object-oriented and scripting motion description language. Besides the conventional variable declaration, functions, and control flow, we create a new syntax for describing parallel actions. We also create a new complex motion statement and expand the use of the conventional motion control concepts using new parallel blocks.

## 7.1 Motion Statements

To describe humanoid motions, we abstracted a collection of predefined motions for the whole body and body parts. We decompose the humanoid in a standard way as specified in humanoid animation specification (Harris et al., 1999). Figure 4 shows the BNF of motion statements. In general, a motion statement is composed of a main part, parameter part, followed by a colon ':', a description part, and ends with a semicolon. A main part includes the object name and the motion name. An object name can be optional. If there is no object name, the system will take the current default object.

A parameter part takes motion parameters that provide additional detail to specify the motion. Each expression is separated by comma ','. The description part can be provided right after the motion function or at the end of a block. It provides a set of attributes to change the scale of the motion. We currently define two attributes namely starttime and speed. We can adapt approaches (Pollard, 1999) to retarget and change the scale of motion.

## 7.2 Specification of Complex Motions

Complex motions can be specified as combinations of sequential and/or parallel sub-motions. Sequential motions will execute one after another. Parallel motions will execute in parallel. The start time for parallel motions may not be the same and thus producing partially overlapping motions.

For ease of specification, we defined our language such that programmers can write overlapping sequential and/or parallel motions in compounded program blocks. Each block can be sub-block of other blocks. A compound block can be sequential, parallel, or mixed of the two.

```
<motion_statement>
      ::= <object_serial_option.><motion>'('<param_list>')'<description_part>';'
<object_serial_option>
      ::= <object> '.'| <object> '.' <object_serial_option> | NULL
<param_list>
      ::= <parameter>| <param_list> ',' <parameter>
```

```
<description_part>
        ::= ':' <description_list>|NULL
<description_list>
        ::= <attribute>|<description_list>','<attribute>
<attribute>
        ::= | speed '=' <float>| starttime '=' <float>        //unit second
```
Fig. 4. Motion statement BNF

```
[  // begin parallel block
  statement1;      // statement1 starts from time 0
  statement2;      // statement2 starts from time 0
  statement3 : starttime = t1;  // statement3 starts from time t1
]  // parallel block end
```
Fig. 5. Parallel block

```
{  // begin sequential block
  statement1; // statement1 starts from time t0 (duration=d1)
  statement2; // statement2 starts from (t0+d1) (duration=d2)
}  // total duration is d1+d2
```
Fig. 6. Sequential block

We define the syntax of parallel block as statements enclosed within "[ ]". Figure 5 shows an example. All statements inside parallel block are considered being issued concurrently. They all start at the same time unless otherwise specified by the "starttime" parameter. In Figure 5 for example, the syntax "statement3 : starttime = t1" specified that statement 3 starts t1 units of time later than the other two statements.

We define the syntax of sequential block as statements enclosed within "{ }". Figure 6 shows an example, in which statement2 starts after statement1 completed. A delay can be used such that statement 2 does not need to starts right after the completion of statement 1.

Sequential and parallel blocks can be combined or embedded with each other to describe complex motion logic. Figure 7 shows an example of the structure. The code in Figure 7 illustrates the time specification as shown in Figure 8.

### 7.3 Sample Program
We show a sample Cybele program in Figure 9. In this program, Jack is the actor. He walks backward two steps. Then, he nods his head, while at the same time, he walks forward two steps. Then, he makes a right turn, a left turn, step right once, and step left once in a sequential way.

## 8. Humanoid Motion Framework in Cybele

To specify humanoid motions, we need to define humanoid framework. Meanwhile, complex motions require checking the constraints and the limitations of humanoid robots. We use H-Anim (H-Anim 2008), an international standard, as our humanoid model. We adopt the hierarchy tree structure of humanoids (Lee & Shin 1999), use degree of freedom (DOF) (Badler et al., 1993) to describe each joint, define states of humanoid, assign priorities to motions, and check constraints and limitations.

```
[
   {  statement1; statement2;  }
    // a sequence begin from time 0
   statement3;    // statement3 begins at time 0
   [  {  statement4; statement5;  }
       // a sequence begin from time 1
     statement6;   // statement6 begins at time 1
   ] : starttime = 1
]
```
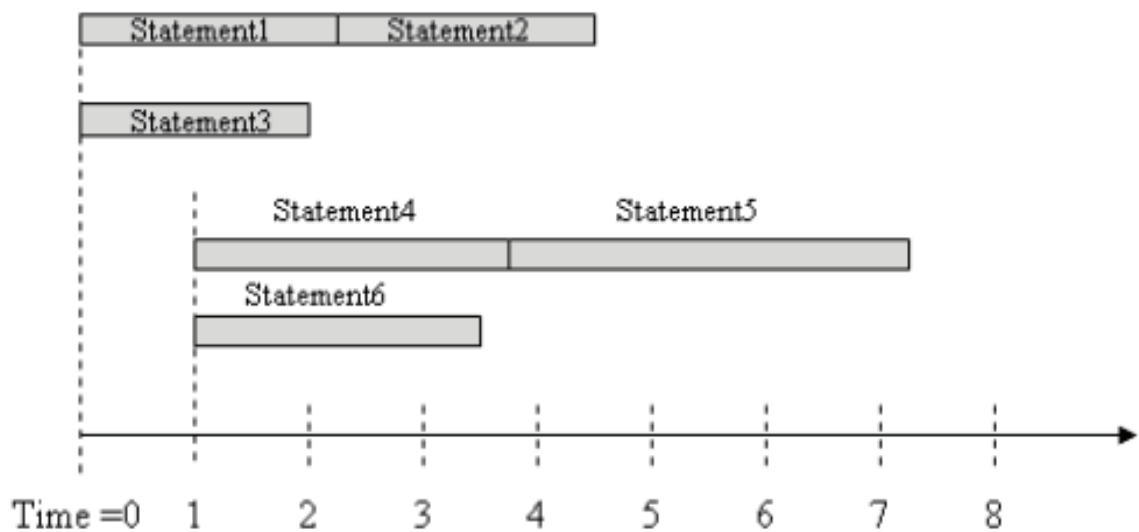Fig. 7. Complex block



Fig. 8. Complex block time slot

### 8.1 Degree of Freedom and Joints
In humanoid robots, joints connect segments through attachment frames that are called sites. Each segment can have several sites. Joints connect sites on different segments within the same figure. Each joint can have at most six degree of freedoms (DOF), including three for rotational and three for translational. In our system, figures have only three rotational DOF, which is defined by the corresponding angle around the axis. Some constraints include rotation axis, joint angle, and upper and lower limits of the joint angle.

```
{
    Jack.backwalk(2) : speed = 1;
    [
      Jack.nod(1);
      Jack.walk(2) : speed= 1;
    ]
    Jack.turnright();
    Jack.turnleft();
    Jack.sidewalkright(1);
    Jack.sidewalkleft(1);
}
```
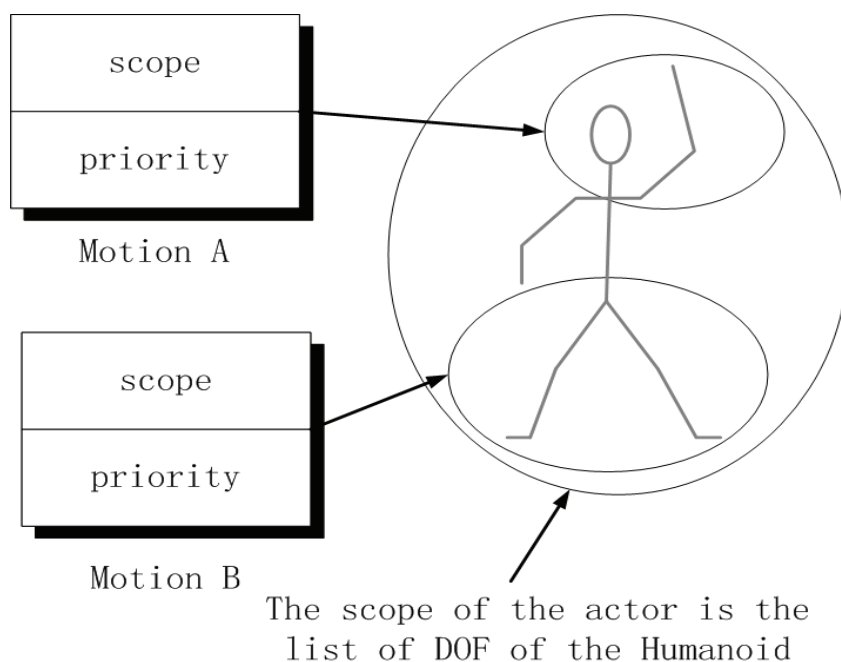
Fig. 9. Sample program



Fig. 10. Scope and priority

## 8.2 States of Humanoid

States are very important aspects of the humanoid model. Static and recognizable states such as stand, sit, crawl, prone, or supine is defined by the relative positioning of each body part, joint and DOF in our system. Programmers can create their customized states by creating new states, which can be a set of DOF for each affected joint. The same state can be applied on different humanoid models, which makes different humanoids move into certain posture.

We describe a partial state of a humanoid as a body part or a set of body parts associated with certain predefined DOF. Partial states are assigned scope and priority (Figure 10). Scope defines the subset of joints that the motion affected. Priority defines the strength and relative importance of sub-motion with respect to all other motions that are in current or

different blocks. The scope and priority allows us to combine partial states to form whole body states or postures, and allows new motions to be generated and synchronized.

### 8.3 State Transition Approach

After we defined states of humanoid, we can define transitions between two states. Transitions between states (Badler et al, 1994) imply movement from one posture to a new posture. Some transitions are possible and some are not, based on the limitation of humanoid robots. Certain transitions are selected and defined as primitive motions. Some of the motion primitives we defined are walk, run, jump, and turn. Our system takes the motion primitives and automatically generates the complete motion sequence from the initial state to the final state.

| Weight | Descriptions |
|---|---|
| $\omega = 1$ | This weighted motion take highest priority over other motions within the same scope. All other existent motions within the scope are transitioned to finish then this weighted motion starts. |
| $0 < \omega < 1$ | This weighted motion is combined with other weighted motions within the same scope, based on their relative weights. |
| $\omega = 0$ | This weighted motion cannot be combined with any other motion within the same scope. The motion is executed after existing motions finished. |

Table 2. Combination of Motions Based on Weights

### 8.4 Motion Combination with Constraints Checking

When two motions intersect with each other in terms of time and/or scope, we need to check the constraints imposed by the limitations of humanoid robots. For example, we cannot require an arm to move forward at the same time require the arm to move backward. When motions are organized in a sequential way, one after anther, and we do not need to check this type of conflicts.

We automated the process of combining parallel motions and resolving possible conflicts between motions. The process uses the predefined scope and priority of each motion. Table 2 shows some examples of priorities that is described here as weights. The weight ranges from 0 to 1 and 1 being the highest. For example, a motion with weight 0 cannot be combined with any other motion within the same scope, while a motion with weight 1 will be executed prior to any other motion within its scope. If two motions have the same predefined weight, then the relative position in the program block is used to determine their priorities. In this case, the first statement has higher priority than the second one. To reduce the complexity of assigned detailed weight, we predefined weight categories and grouped motions into the categories.

### 8.5 Three Dimensions of Complex Humanoid Motions

We defined three dimensions for complex humanoid motions. There are time, scope, and level. Figure 11 shows an example of their relationships. The time axis shows the beginning and ending time for each motion. The scope axis indicates which joints or body parts are affected by the motion. The level axis shows the hierarchical structure among various motions nested within complex sequential and parallel blocks. The level is used during

motion combination process, in which lowest level motions are created first then process to the highest level.

## 9. Cybele Humanoid Simulation System

We developed a virtual reality simulator for humanoid robots based on our new motion description language and humanoid framework. The simulation system interprets the motions specified by the language and checks the constraints based on the humanoid framework. The system is designed to interact with user inputs and control humanoid robots in real time.

```
{     //Block 1: level 1, attribute 0
...   //statements: BlockID 1, level 1, attribute 0
    [    //Block 2: level 2, attribute 1
    ...   //statements: BlockID 2, level 2, attribute 1
    ]    //Block 2: end
...   //statements: BlockID 1, level 1, attribute 0
    [    //Block 3: level 2, attribute 1
    ...   //statements: BlockID 3, level 2, attribute 1
        {    //Block 4: level 3, attribute 0
        ...   //statements: BlockID 4, level 3, attribute 0
            [    //Block 5: level 4, attribute 1
            ...   //statements: BlockID 5, level 4, attribute 1
            ]    //Block 5: end
        }    //Block 4: end
    ...   //statements: BlockID 3, level 2, attribute 1
        [    //Block 6: level 3, attribute 1
        ...   //statements: BlockID 6, level 3, attribute 1
        ]    //Block 6: end
    ...   //statements: BlockID 3, level 2, attribute 1
    ]    //Block 3: end
...   //statement: BlockID 1, level 1, attribute 0
}    //Block 1: end
```
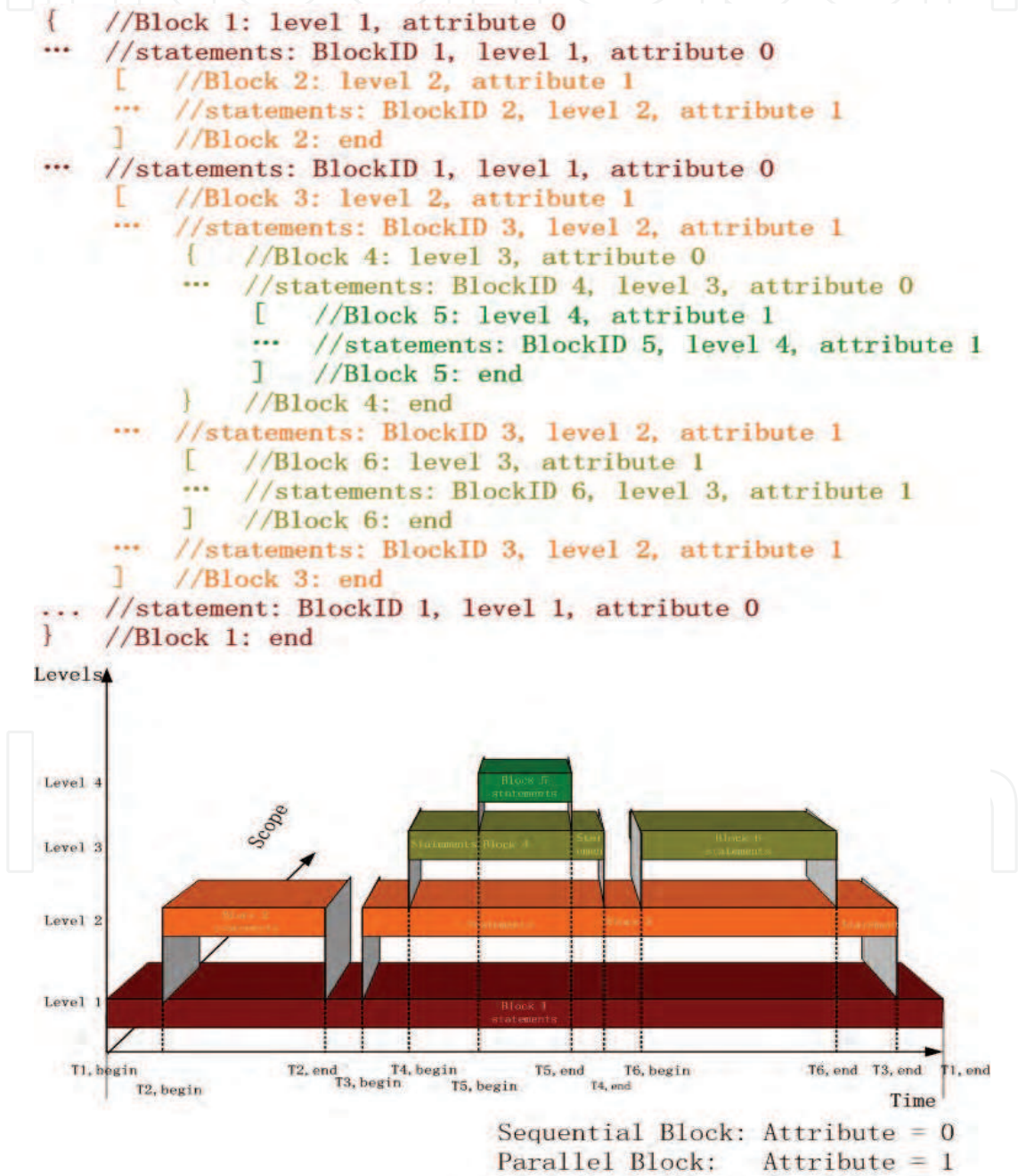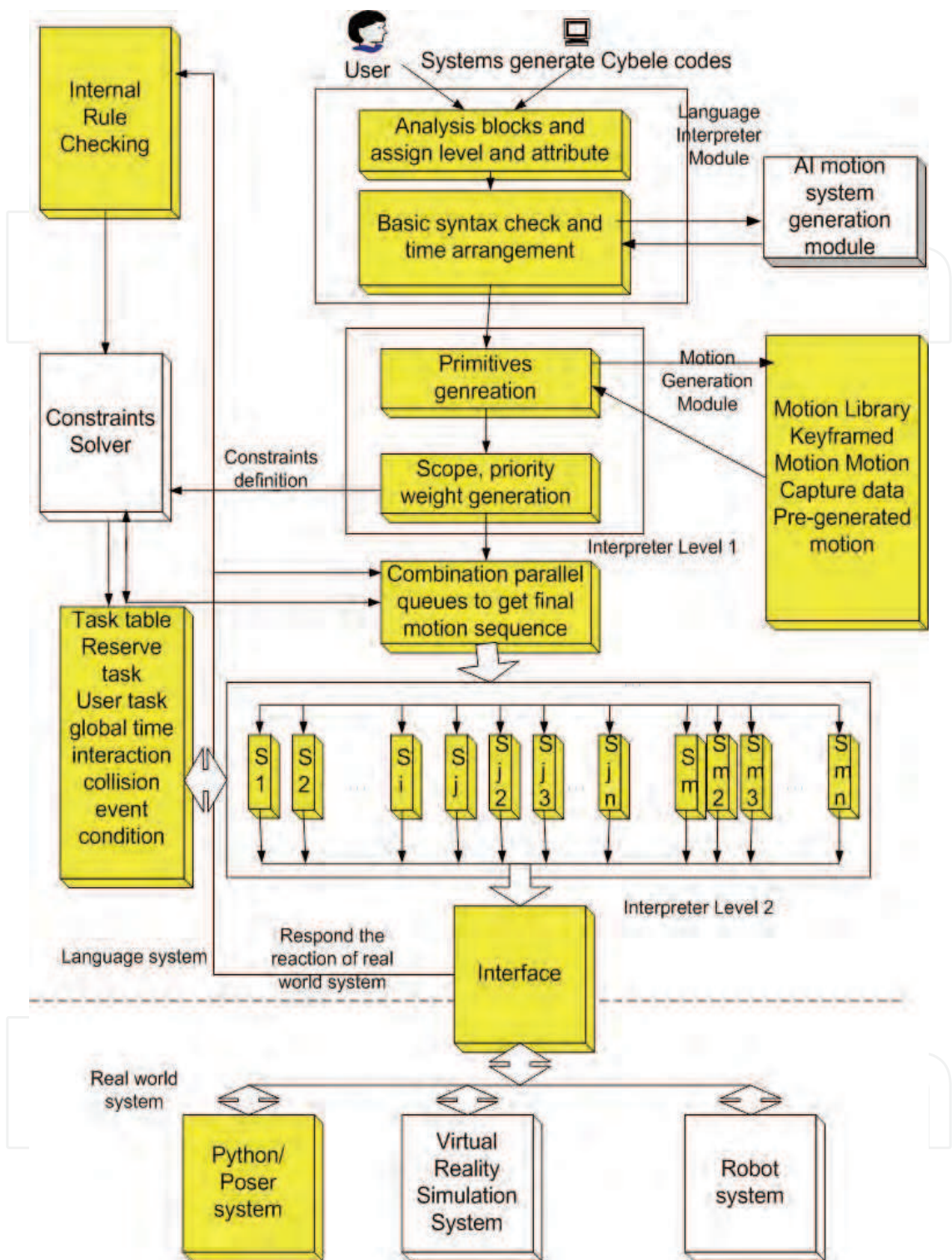


Fig. 11. Three Dimensions of Humanoid Motions

Fig. 12. Humanoid Simulation System Overview

Figure 12 shows an overview of our humanoid simulation system. The system first gets the Cybele program from some consoles or from other interfacing systems. It interprets the program by analyzing various syntax and structures. Then the language interpreter module passes the results to the motion generation module, which creates motion sequences. In this

module, all instructions will be broken down into primitives and tagged with level and block information. Every primitive is generated based on parameters and attributes such as duration, speed, and so on. It performs the following functions:

1.  Sub-motion generation. The interpreter checks the library to find the proper primitive sequences.

2.  Time schedule. A time scheduler schedules the duration for each sub-motion.
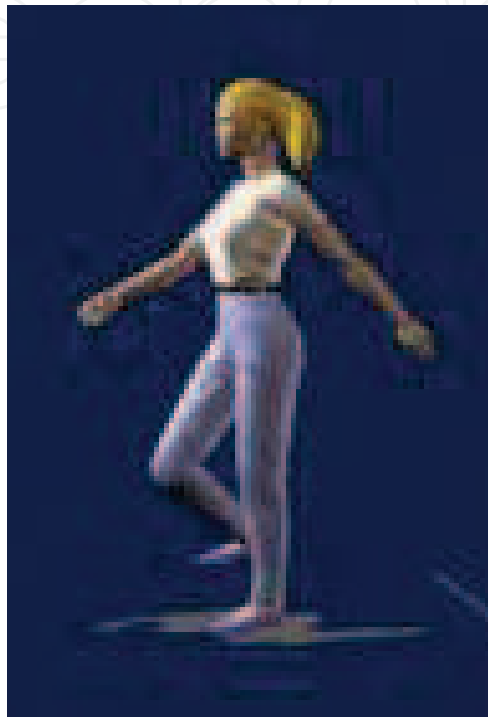


Fig. 13. Humanoid Dancing in Virtual Environment

After all the primitives are generated, the motion sequences (labelled S's in the figure) are transferred to the interface. The interface has two functions. The first function is to transmit the elementary motion sequences to various real world systems, which consist of Python/Poser system, Virtual Reality Simulation system, and Robot system (as shown in Figure 12).

The second function of the interface module is to collect the feedbacks and other system parameters from the real world system. These feedbacks are used be the language system to adapt to the current constraints imposed be the environment.

There are two types of constraints in our system. One is an internal rules constraint which is defined by the system and cannot be changed after the environment is created. The other is a rules constraint created by programmers after the environment is created and that can be changed by programmers. All these constraints are resolved by the Constraints Solver module (shown on the right side of Figure 12).

As can be seen from the system overview (Figure 12), this system is quite complex. Currently, we have completed the implementation and testing of the highlighted modules. Our test results show that such a system is viable. One of our tests was to describe dance

movements using Cybele. Figure 13 shows one of our simulations of a humanoid dancing in a virtual environment.

## 10. Conclusion and Future Research

In this chapter we described a description language and framework for specifying humanoid motions and for allowing humanoid robots to learn motor skills through interacting with the environments. The language and framework are unique and encompassing many areas of research interesting to researchers in epigenetic robots. The framework can also serve as an outline of strategies for future research programs in humanoid motion description and motor skill acquisition.

Based on our motion description framework, we developed a new humanoid motion description language called Cybele. New features of the language include complex parallel and sequential blocks, new syntax for motion statements, and built-in constraints for motion combination. The motion description links tightly to the underlying models. We defined a humanoid motion framework to make specification of humanoid motion possible. Then we created a simulation system to implement the framework and to execute the motion descriptions. For our system, we also created a solution to generate complex parallel and sequential motion sequences based on the limitations and constraints of the humanoid body. To solve the motion mixing problem in complex blocks, we presented an approach for synthesizing motions with partial state combinations. In the solution, scope and priority are defined, and primitives are combined to produce final motions with constraints checking. Test results show that Cybele is an effective motion description language, and also show that such virtual reality simulation of humanoid robots is viable.

The proposed simulation system is quite complex, which includes using artificial intelligent techniques to automatically generate detailed motion sequences based on given goal statements. Further work is required to develop such AI motion generation system and to further develop AI methods for solving various constraints imposed by the limitation of the humanoid robots and by the environments. Although our system is designed to interact directly with humanoid robots, we have yet to test the system using real robot. Although our virtual simulation shows such system is feasible, a real humanoid robot will encounter much more constraints when interacting directly with real human environments including interacting with human. Thus, future research would also focus on multi-agent systems in which robots, human, and their environments interacting with each others in real time.

For humanoid robots to be able to effectively function in real environments and interacting with people, they must be able to adapt and able to learn. Most researchers realize this requirement and many are working on various learning methods for humanoids. However, much research remains to be done for humanoid robots to learn motor skills. Although this chapter outlined some strategies including, creating a motion description language, using non-deterministic finite state machine to model sequence of motion states, and using automatic constraint satisfaction to generation motions plausible for the underlying humanoid bodies, much work remains to be done in this exciting area of research.

## 11. References

Adamson A. (1987). "Calaban", Demonstration at Birmingham University, England, June 1987, http://www.bham.ac.uk/calaban/.

Arsenic, A.M. (2004). "Developmental learning on a humanoid robot," *2004 IEEE International Joint Conference on Neural Networks*, vol.4, pp. 3167- 3172, 25-29 July 2004.

Arsenio, A.M. (2004). *Cognitive-Developmental Learning for a Humanoid Robot: A Caregiver's Gift*, Doctoral thesis, Massachusetts Inst. of Tech., Cambridge, Dept. of Electrical Engineering and Computer Science.

Badler, N. I., & Smoliar, S. W. (1979). "Digital Representation of Human Movement", *Computer Surveys*, Vol. 11, No 1, March 1979.

Badler, N. I., Bindiganavale, R., Granieri, J. P., Wei, S., and Zhao, X. (1994). "Posture Interpolation with Collision Avoidance," *In Proc. Computer Animation*, pp.13-20.

Badler, N., Phillips, C., and Webber, B. (1993). *Simulating Humans: Computer Graphics, Animation, and Control*, Oxford University Press.

Bentivegna, D.C. & Atkeson, C.G. (2001). "Learning from observation using primitives," *IEEE International Conference on Robotics and Automation*, vol.2, pp. 1988-1993.

Brooks, R.A. (2002). "Humanoid robots," *Communications of the ACM*, Special Issue: Robots: intelligence, versatility, adaptivity, Vol. 45, Issue 3, pp.   33-38, March 2002.

Brooks, R.A. (1996). "Prospects for human level intelligence for humanoid robots," Proceedings of the First International Symposium on Humanoid Robots (HURO-96), pp. 17-24.

Brooks, R.A., Breazeal, C.; Marjanovic, M.; Scassellati, B. & Williamson, M.M. (1998). "The Cog Project: Building a Humanoid Robot," *IARP First International Workshop on Humanoid and Human Friendly Robotics*, (Tsukuba, Japan), pp. I-1, October 26-27.

Burghart, C., Mikut, R., Stiefelhagen, R., Asfour, T., Holzapfel, H., Steinhaus, P., & Dillmann, R., (2005). "A cognitive architecture for a humanoid robot: a first approach," *2005 5th IEEE-RAS International Conference on Humanoid Robots*, pp. 357- 362, 5-7 Dec. 2005.

Calinon, S.,  Guenter, F., & Billard, A. (2007). "On Learning, Representing and Generalizing a Task in a Humanoid Robot," *IEEE transactions on systems, man and cybernetics*, Part B. Special issue on robot learning by observation, demonstration and imitation, vol. 37, num. 2, 2007, p. 286-298.

Calvert, T.W., Bruderlin, A., Mah, S., Schiphorst, T., & Welman, C. (1993). "The Evolution of an Interface for Choreographers", *Interchi*, pp. 24-29.

Causley, M. (1980). *An introduction to Benesh Movement Notation*, ISBN: 0836992806.

Choi, B. (1998). "Automata for Learning Sequential Tasks," *New Generation Computing: Computing Paradigms and Computational Intelligence*, Vol. 16, No. 1, pp. 23-54.

Choi, B. (2002). "Applying Learning by Example for Digital Design Automation," *Applied Intelligence*, Vol. 16, No. 3, pp. 205-221.

Choi, B. (2003). "Inductive Inference by Using Information Compression," *Computational Intelligence* 19 (2), 164-185.

Choi, B., and Chen, Y. (2002). "Humanoid Motion Description Language," *Second International Workshop on Epigenetic Robotics*, pp. 21-24.

Conway, M.J. (1997). *Alice: Easy-to-Learn 3D scripting language for novices*, Ph. D thesis, University of Virginia.

Eshkol-Wachman (2008). Eshkol-Wachman Movement Notation, http://www.movementnotation.com/

H-anim (2008). Humanoid animation work group. http://www.h-anim.org.

Harris, J.A., Pittman, A.M., Waller, M.S., and Dark, C.L. (1999). *Dance a while Handbook for Folk, Square, Contra and Social Dance*, Benjamin Cummings.

Hirai, K, Hirose, M. Haikawa, Y. Takenaka, T. (1998). "The Development of Honda Humanoid Robot", *Proceeding of IEEE Intl. Conference on Robotics and Automation (ICRA)*, pp. 1321-1326.

Honda (2008). Honda Humanoid robot: http://world.honda.com/ASIMO/

Huang, Z., Eliëns, A., and Visser, C. (2002). "STEP: a Scripting Language for Embodied Agents," *Proceedings of the Workshop of Lifelike Animated Agents*, Tokyo.

Hutchinson A. & Balanchine, G. (1987). *Labanotation: The System of Analyzing and Recording Movement*, ISBN: 0878305270.

Kanayama, Y., and Wu, C. (2000). "It's Time to Make Mobile Robots Programmable," *Proceedings of Intl. Conference on Robotic and Automation (ICRA)*, San Francisco.

Kanehiro, F., Hirukawa, H. & Kajita, S. (2004). "OpenHRP: Open Architecture Humanoid Robotics Platform," *The International Journal of Robotics Research*, Vol. 23, No. 2, 155-165.

Kanehiro, F., Miyata, N., Kajita, S., Fujiwara, K., Hirukawa, H., Nakamura, Y., Yamane, K., Kohara, I., Kawamura, Y., & Sankai, Y. "Virtual Humanoid Robot Platform to Develop Controllers of Real Humanoid Robots without Porting," *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hawaii, Oct. 29 - Nov. 03, 2001, pp. 1093-1099.

Lee. J and Shin. S. (1999). "A hierarchical approach to interactive motion editing for human-like figures," *Proceedings of SIGGRAPH 1999*, pages 39-48.

Nakaoka, S., Nakazawa, A., Yokoi, K., & Ikeuchi, K. (2004). "Leg motion primitives for a dancing humanoid robot," *2004 IEEE International Conference on Robotics and Automation*, Vol.1, pp. 610- 615, 26 April-1 May 2004.

Nakaoka, S., Nakazawa, A., Yokoi, K., Hirukawa, H., & Ikeuchi, K. (2003). "Generating whole body motions for a biped humanoid robot from captured human dances," *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, vol.3, pp. 3905- 3910, 14-19 Sept. 2003.

Nishimura, Y., Kushida, K., Dohi, H., Ishizuka, M., Takeuchi, J., & Tsujino, H. (2005). "Development and psychological evaluation of multimodal presentation markup language for humanoid robots," *2005 5th IEEE-RAS International Conference on Humanoid Robots*, pp. 393- 398, 5-7 Dec. 2005.

Nozawa, Y., Dohi, H., Iba, H., & Ishizuka, M. (2004). "Humanoid robot presentation controlled by multimodal presentation markup language MPML," *13th IEEE*

*International Workshop on Robot and Human Interactive Communication*, pp. 153- 158, 20-22 Sept. 2004.

Oh, J.H., Hanson, D., Kim, W.S., Han, Y., Kim, J.Y., & Park, I.W. (2006). "Design of Android type Humanoid Robot Albert HUBO," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1428-1433, Beijing, Oct. 2006.

Perlin, K, Gikdberg, A. (1996). "Improv: A System for Scripting Interactive Actors in Virtual Worlds", *Computer Graphics Proceeding*, pp.205-216.

Perlin, K., and Goldberg, A. (1996). "Improv: A System for Scripting Interactive Actors in Virtual Worlds," *Computer Graphics*; Vol. 29 No. 3.

Pollard, N. S. (1999). "Simple Machines for Scaling Human Motion," *Eurographics Workshop on Animation and Simulation*, Milan, Italy.

Python (2008). Python. http://www.python.org

Riley, M. & Atkeson, C.G. (2000). "Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching," *Proc. 2000 Workshop on Interactive Robotics and Entertainment*, pp. 35-42, Robotics Inst., Carnegie Mellon Univ.

Ryman, R., Singh, B., Beatty, J., & Booth, K. (1984). "A Computerized Editor of Benesh Movement Notation," *Dance Research Journal*, 16(1): 27-34.

Safonova A., Nancy S.P., & Hodgins, J.K. (2002). "Adapting human motion for the control of a humanoid robot," *Proceedings of International Conference on Robotics and Automation*, pp. 1390-1397.

Scassellati, B.M. (2001). *Foundations for a Theory of Mind for a Humanoid Robot*, Doctoral thesis, Massachusetts Inst. of Tech., Cambridge, Dept. of Electrical Engineering and Computer Science.

Schiphorst, T. (1992). "LifeForms: Design Tools for Choreography", *Dance and Technology I: Moving Toward the Future*, pp. 46-52.

Schrand, R. (2001). *Poser 4 Pro Pack f/x & Design*, Coriolis Group.

Sony (2008). Sony QRIO humanoid robot, http://www.sony.net/SonyInfo/ CorporateInfo/History/sonyhistory-j.html.

Ude, A., Atkesona, C.G., & Rileyd, M. (2004). "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, Vol. 47, Issues 2-3, 30 June 2004, pp. 93-108.

Yokoi, K. (2007). "Humanoid robotics," *International Conference on Control, Automation and Systems*, pp. lxxiv-lxxix, Seoul, 17-20 Oct. 2007.

Zhou, Y. & Choi, B. (2007). "Virtual Reality Simulation of Humanoid Robots," *IEEE Industrial Electronics Society (IECON) 33rd Annual Conference*, pp. 2772-2777.
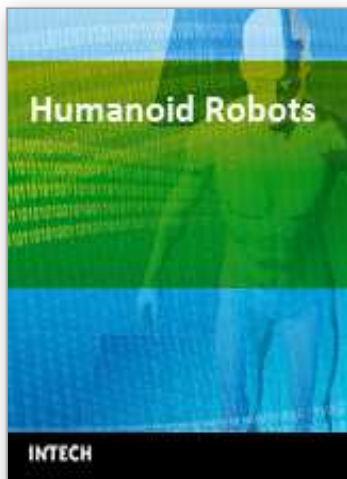
**Humanoid Robots**

Edited by Ben Choi

Humanoid robots are developed to use the infrastructures designed for humans, to ease the interactions with humans, and to help the integrations into human societies. The developments of humanoid robots proceed from building individual robots to establishing societies of robots working alongside with humans. This book addresses the problems of constructing a humanoid body and mind from generating walk patterns and balance maintenance to encoding and specifying humanoid motions and the control of eye and head movements for focusing attention on moving objects. It provides methods for learning motor skills and for language acquisition and describes how to generate facial movements for expressing various emotions and provides methods for decision making and planning. This book discusses the leading researches and challenges in building humanoid robots in order to prepare for the near future when human societies will be advanced by using humanoid robots.

**How to reference**

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

Ben Choi (2009). Humanoid Robotic Language and Virtual Reality Simulation, Humanoid Robots, Ben Choi (Ed.), ISBN: 978-953-7619-44-2, InTech, Available from:
http://www.intechopen.com/books/humanoid_robots/humanoid_robotic_language_and_virtual_reality_simulation