Humanoid Robot Interactions by Motion Description Language and Dialogue

Kankana Shukla and Ben Choi
Computer Science, Louisiana Tech University, USA
Email: kankanashukla.edu@gmail.com, pro@benchoi.org

Abstract—**This paper describes the development of a system for facilitating humanoid robots to acquire sequence of movements by motion description and dialogue. The unique feature of our system is that we define a descriptive language and framework that allows the specification of humanoid motions and the representation of motor skills. We subsequently developed and tested a virtual reality simulation system that can execute the motion descriptive language. We then extended the system to allow humanoid robots to acquire motion descriptions through conversing with human. We implemented and tested the proposed system on the NAO humanoid robot. Our implementation includes a dialogue module that continues to update the encoded motion descriptions as the conversation advances and an action module that retrieves the acquired motion descriptions and direct the robot to perform the specified tasks.**

Index Terms—**human-robot systems, humanoid robot, robotics, motion description language, virtual reality, machine learning**

I. Introduction

There has been intensive research to build robots that resemble the human form and imitate human behaviours. Humanoid robots are developed to use the infrastructures designed for humans, to ease the interactions with humans, and to help them integrate into human societies. This paper will focus on the interaction between human and humanoid robots.

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. Most of the existing humanoid simulation systems either require manual manipulations of body parts or capturing body movements enacted by a human.

To facilitate interaction between human and humanoid robots, we proposed a new motion description language and framework, which allows people to control or direct humanoid robots to perform complex motions. We first tested our proposed language and framework by developing a simulation system that can execute the humanoid motion description language and can automatically enact the motions of the humanoid robots.

We further extend our motion description language to allow dialogue between human and humanoid robots. Our proposed dialogue system allows people to instruct robots to perform specific tasks that consist of many steps of complex motions. However, the robot needs to be able to understand the human instructions that often do not provide sufficient details. In such case, the robot will query for more details if needed. The people and robot continue the dialogue until the motions of the specific tasks are fully specified.

We implemented and tested our proposed system on the NAO humanoid robot. Our test results show that the proposed system greatly enhances the interactions between people and humanoid robots.

The remaining sections of this paper are organized as follows. Section II outlines the related research on high-level language approaches to describe and to simulate humanoid motions. The humanoid motion description language and framework are described in Section III; while Section IV focuses on virtual reality simulation. Section V describes the proposed dialogue system and the implementation of the framework on NAO humanoid robots. Finally, Section VI provides the conclusion and outlines the future work.

II. Related Research

Research to describe humanoid motions begins with the work for describing human dances. Popular dance notation systems include Benesh [1], Labanotation [2], and EW [3]. Labanotation is more comprehensive than Benesh for describing human motion in general. EW can be applied on linkage systems other than human body. 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 [4]. Another approach such as Improv system uses natural language to script human behavior interacting in virtual environments [5].

This section outlines related work on the high-level language approaches to humanoid simulation [6], [7]. Poser Python [8], [9] is an implementation of the Python interpreter. VRML (Virtual Reality Modeling Language) is a scene-description language used widely on the internet. VRML uses TimeSensor to initiate and

Manuscript received May 1, 2013; revised June 5, 2013; accepted July 17, 2013.

synchronize all the motions in the scene. Improv [10] is a system for the creation of real-time behavior based on animated actors. Distributed logic programs such as STEP [11] are also being used to define and control the hand gestures of embodied agents in virtual worlds.

Several attempts have been made in the past to make a humanoid robot learn gestures by imitation [12]. Robot learning upper body motions by active imitation [13] and full body dynamic imitation [14] have also been performed in the past.

MIT's Cog project [15], [16] makes use of the principle that mental abilities can be improved by interacting with the environments. However, Cog robot currently lacks locomotion. On the contrary, Honda's humanoid robots [17] possess the state of the art locomotion system, despite lacking the autonomy and the learning abilities. We envision the union of these two types of robots, such as Albert HUBO [18], as the basis of our investigation.

There are also other systems for animations such as Alice [19], Maya, Lightwave, and 3D Studio Max. They each have a beautiful graphical user interface. However, most of them lack an underlying programming environment that can be used to control the motions. Subsequently, 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, at times even making the motions non-reusable.

Our language and our framework [20] are unique in many ways comparing to other related research [21]. 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 humanoid robots to interact with its environments using different level of granularity. Our Automatic Constraint Satisfaction system reduces the complexity of specifying humanoid motions.

III. Humanoid Motion Description Language and Framework

In this section, we defined a descriptive language and framework that allows the specification of humanoid motions and the representation of motor skills. Our humanoid motion description language, like any other language, 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.

The proposed language and framework for specifying humanoid motions include the following attributes: motion description layers, egocentric reference system, progressive quantized refinement, and automatic constraint satisfaction. Table I outlines the concept of motion description layers. Each description layer is a level of abstraction. Joint Angle 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. Motion primitive [22] 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 completed 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 with ease without having to specify the angle of each joint.

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 Fig. 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.

TABLE I. Motion Description Layers

Description Layer	Examples
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

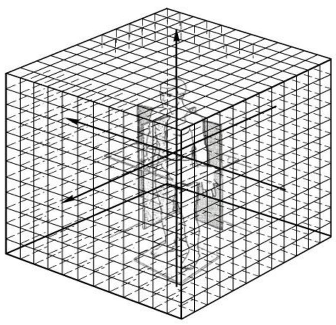


Figure 1. Egocentric space reference system

We proposed a concept called Progressive Quantized Refinement for a humanoid robot to interact with its environments using different level of granularity. Fig. 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 t.able, 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.

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. 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.

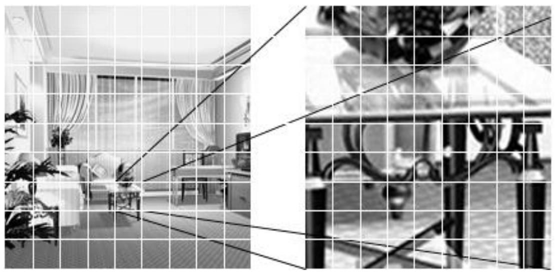


Figure 2. Concept of progressive quantized refinement

We developed a humanoid motion description language, called Cybele, based on the proposed framework. Cybele provides simple syntax for expressing parallel and sequential processing as well as the combination of them [20]. We also introduce syntax for expressing motions. 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 and extract a relatively small set of parameterized motions from infinite complex motion sequences and make them reusable as primitive motions.

Cybele is an 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. Sequential and parallel blocks can be combined or embedded with each other to describe complex motion logic.

Fig. 3 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 show a sample Cybele program in Fig. 4. In this program, Jack is the actor. He walks two steps backward. Then, he nods his head and simultaneously walks two steps forward. Then, he makes a right turn, a left turn, step right once, and step left once in a sequential way.

```
<motion statement>

::= <object_serial_option.> <motion>
'('<param_list>')'<description_part>';'<object_serial_o
ption>
<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
```

Figure 3. Motion statement BNF

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

Figure 4. Sample program

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. In Fig. 5 and Fig. 6, we provide the examples of syntax used for parallel and sequential blocks respectively. Fig. 7 shows an example of the partially overlapping motions, the time specification of which is shown in Fig. 8.

```
[  // 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
```

Figure 5. Parallel block

```
{  // begin sequential block
statement1;    // statement1 starts from time t0
                                                    (duration = d1)
statement2;    // statement2 starts from time 0
                                                    (duration = d2)
} // total duration is d1 + d2
```

Figure 6. Sequential block

```
[
{  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
]
```

Figure 7. Complex block

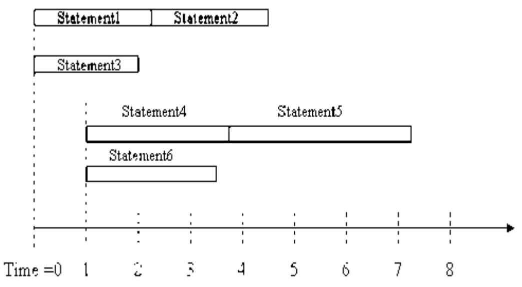
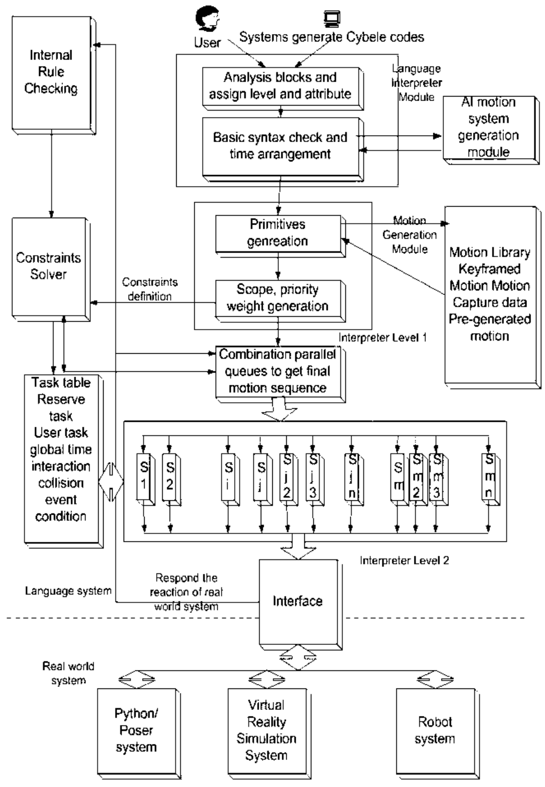


Figure 8. Complex block time slots

IV. Virtual Reality Simulation of Humanoid Robots

We developed a virtual reality system to simulate humanoid robots based on our 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 [23].

The overview of our humanoid simulation system is provided in Fig. 9. The simulator interprets the programs, breaks down motions into primitives with time, scope, and priority. Final motion sequences are then generated using a synchronization approach. We define the scope and priority for each joint in the humanoid and also incorporate system checks to achieve constraints satisfaction. 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.



User
Systems generate Cybele codes
Internal Rule Checking
Analysis blocks and assign level and attribute
Language Interpreter Module
AI motion system generation module
Basic syntax check and time arrangement
Primitives genreation
Motion Generation Module
Motion Library Keyframed Motion Motion Capture data Pre-generated motion
Constraints Solver
Constraints definition
Scope, priority weight generation
Interpreter Level 1
Combination parallel queues to get final motion sequence
Task table Reserve task User task global time interaction collision event condition
S 1
S 2
S i
S j
S j 2
S j 3
S j n
S m
S m 2
S m 3
S m n
Interpreter Level 2
Language system
Respond the reaction of real world system
Interface
Real world system
Python/ Poser system
Virtual Reality Simulation System
Robot system

Figure 9. Humanoid simulation system overview

The system first gets the Cybele program from some consoles or from other interfacing systems. It interprets the program by analysing 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.

After all the primitives are generated, the motion sequences (labelled S's in Fig. 9) 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 Fig. 9).

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 Fig. 9).

Our test results show that such a system is viable. One of our tests was to describe dance movements using Cybele. Fig. 10 shows one of our simulations of a humanoid dancing in a virtual environment.

Figure 10. Humanoid dancing in virtual environment

V. Implementation on NAO Humanoid Robot

After testing our proposed language and framework using virtual reality simulation, we extended them to create a dialogue system, which was implemented in a real humanoid robot, called NAO (made by Aldebaran Robotics). Our proposed dialogue system allows people to instruct robots to perform specific tasks that consist of many steps of complex motions. The robot needs to be able to understand the human instructions and will query for more details if needed. The people and robot continue the dialogue until the motions of the specific tasks are fully specified.

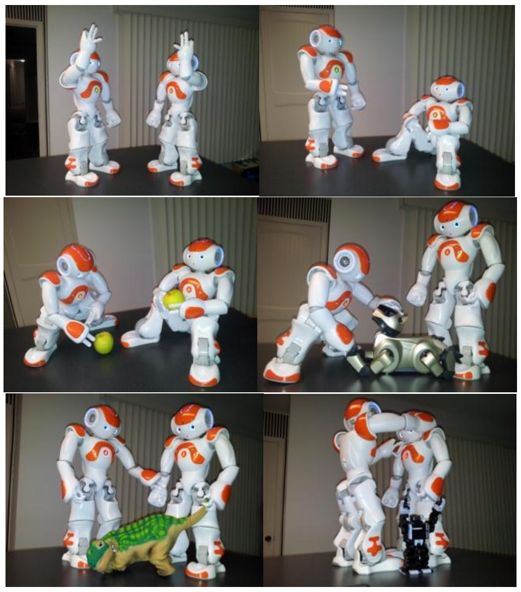


Figure 11. NAO robot interactions

We partition our system into a dialogue module and an action module. The dialogue module is involved for conversing with the human and interpreting the requirements with the help of NAO's speech synthesis and recognition system. The dialogue consists of commands given to NAO by the human and NAO converting the dialogue into motion descriptions. If the dialogue is not interpreted by NAO, it asks the human to reiterate. NAO continues to refine and update the motion description as the conversation progresses. This process allows NAO robots to acquire the needed motion descriptions for performing specific task. Once the motion description is fully specified, the action module assists NAO to convert these motion descriptions into actions and perform them in sequence.

The process for allowing NAO robots to acquire motion descriptions by conversing with human is outlined in the following steps: (1) a person directs the robot to start the learning mode; (2) the robot queries for commands by using text to speech synthesis; (3) the person provides verbal instructions; (4) the robot uses speech recognition module to convert the instructions into text; (5) the robot processes the text and convert them into motion descriptions; (6) in case the instructions do not provide enough details, the robot will ask more specific questions, (7) the steps from steps 2 to 6 are repeated until the required sequence of motions are fully conversed.

During the testing of the dialogue system, we instructed two NAO robots to interact with each other and with other robots and objects. Fig. 11 shows six snapshots of the interaction. First, we simply instruct two NAO robots to greet each other as seen in the top left snapshot, while the top right photo was the result when we instructed one of the robots to sit down. In the middle left snapshot, we can see the trained NAO robot recognizing an apple and then performing sequences of motions associated with the apple. We then instruct NAO robot to pet on the head of a Sony AIBO robot dog, which can be viewed in the middle right snapshot. On the bottom left photo shows an instructed NAO robot reaching the tail of a PLEO robotic dinosaur, while on the bottom right snapshot was taken when we instructed two NAO robots to reach another by extending arms but avoiding and not interacting with the I-SOBOT robot in front of them.

VI. Conclusion and Future Work

In order to address the problems associated with specifying humanoid motions and to facilitating interactions between human and humanoid robots, we propose a new descriptive language and framework. Our techniques are unique for specifying humanoid motion and representing motor skills. We then developed a virtual reality simulation system to test the proposed language and framework. After successful simulations, we extend our system to allow humanoid robots to acquire motion descriptions by conversing with human, and implemented and tested the system on NAO humanoid robots.

The future work involves extending the pattern recognition system of NAO robots to recognize human motions, and thus enable the robot to acquire motion descriptions by observations and to continue enhance the description based on the continuous observations.

REFERENCES

[1] M. Causley, *An introduction to Benesh Movement Notation*, 1980.

[2] A. Hutchinson and G. Balanchine, *Labanotation: The System of Analyzing and Recording Movement*, 1987.

[4] N. I. Badler and S. W.Smoliar, "Digital representation of human movement," *Computer Surveys*, vol. 11, no 1, March 1979.

[3] E. Wachman. (2008). *Eshkol-Wachman Movement Notation*, Available: http: www.movementnotation.com

[5] K. Perlin and A. Gikdberg, "Improv: A system for scripting interactive actors in virtual worlds," *Computer Graphics Proceeding*, pp. 205-216, 1996.

[6] Y. Nozawa, H. Dohi, H. Iba, and M. Ishizuka, "Humanoid robot presentation controlled by multimodal presentation markup language MPML," in *Proc. 13th IEEE International Workshop on Robot and Human Interactive Communication*, 20-22 Sept. 2004, pp. 153- 158.

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

[8] Python. (2013). Python programming language – official website. Available: http:www.python.org

[9] R. Schrand, Poser 4 Pro Pack fx & Design, Coriolis Group, 2001

[10] K. Perlin and A. Goldberg, "Improv: A system for scripting interactive actors in virtual worlds," *Computer Graphics*, vol. 29, no. 3, 1996.

[11] Z. Huang, A. Eliëns, and C. Visser, "STEP: A scripting language for embodied agents," in *Proc. of the Workshop of Lifelike Animated Agents*, Tokyo, 2002.

[12] S. Calinon and A. Billard, "Learning of gestures by imitation in a humanoid robot," *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*, pp. 153-177, 2007.

[13] J. P. Bandera, R. Marfil, L. M. Tanco, J. A. Rodriguez, A. Bandera, and F. Sandoval, "Robot learning of upper-body human motion by active imitation," *IEEE-RAS International Conference on Humanoid Robots - Humanoids,* 2006..

[14] J. B. Cole, D. B. Grimes, and R. P. N. Rao, "Learning full-body motions from monocular vision: dynamic imitation in a humanoid robot," In *Proc. 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007.

[15] R. A. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, and M. M. Williamson, "The cog project: Building a humanoid robot," in *Proc. IARP First International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba, Japan, October 26-27, 1998, pp. I-1,

[16] A. M. Arsenio, "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, 2004.

[17] Honda (2013). Honda Humanoid robot: http:world.honda.comASIMO

[18] J. H. Oh, D. Hanson, W. S. Kim, Y. Han, J. Y. Kim, and I. W. Park, "Design of Android type humanoid robot albert HUBO," in *Proc. 2006 IEEERSJ International Conference on Intelligent Robots and Systems*, Beijing, Oct. 2006, pp. 1428-1433.

[19] M. J. Conway, Alice, "Easy-to-Learn 3D scripting language for novices," Ph. D thesis, University of Virginia, 1997.

[20] B. Choi and Y. Chen, "Humanoid motion description language," in *Proc. Second International Workshop on Epigenetic Robotics*, 2002, pp. 21-24.

[21] F. Kanehiro, H. Hirukawa, and S. Kajita, "OpenHRP: Open architecture humanoid robotics platform," *The International Journal of Robotics Research*, vol. 23, no. 2, pp.155-165, 2004.

[22] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi, "Leg motion primitives for a dancing humanoid robot," in *Proc. 2004 IEEE International Conference on Robotics and Automation*, vol.1, 26 April-1 May 2004, pp. 610- 615.

[23] Y. Zhou and B. Choi, "Virtual reality simulation of humanoid robots," in *Proc. IEEE Industrial Electronics Society 33rd Annual Conference,* 2007, pp. 2772-2777.

Kankana Shukla is a Masters student in Computer Science and Biomedical Engineering at Louisiana Tech University. She completed her bachelors in Electronics and Instrumentation. Her research interest includes Data Mining, Web Mining, Big Data Analysis, Machine Learning, Bioinformatics, Biostatistics, Robotics and Artificial Intelligence. Her future work includes pursuing a Ph.D. degree in Data Mining and Biostatistics.

Dr. Ben Choi has a Ph.D. degree in Electrical and Computer Engineering and also has a Pilot certificate for flying airplanes and helicopters. He is an Associate Professor in Computer Science at Louisiana Tech University. He received his Ph.D., M.S., and B.S. degrees from The Ohio State University, studied Computer Science, Computer Engineering, and Electrical Engineering. His areas of research include Humanoid Robots, Artificial Intelligence, Machine Learning, Intelligent Agents, Semantic Web, Data Mining, Fuzzy Systems, and Parallel Computing. His future research includes developing advanced software and hardware methods for building intelligent machines and theorizing the Universe as a Computer.