3D Signing Avatar for Sinhala Sign Language

M. Punchimudiyanse[1], R.G.N. Meegama[2]

[1]*Department of Mathematics and Computer Science, The Open University of Sri Lanka,*
[1]malinda@ou.ac.lk

[2]*Department of Computer Science, University of Sri Jayawardenapura, Sri Lanka*
[2]rgn@sci.sjp.ac.lk

***Abstract*— Animating signs of a sign language requires on demand bone movement of a signing avatar based on single posture, multi posture signs for the words with a known sign, and fingerprinting signs for representing unknown words character by character. Ordering and sequencing of different types of signs corresponding to words/phrases of a typical spoken/written language are the most essential features of a 3D signing avatar. Typical techniques of signing avatar animation are video sequencing of signs of corresponding words of a sentence, or replaying movements of signs with motion-captured animation sequences. This paper presents a multi-facet 3D avatar and an animation framework that supports the definition and animation of sign gestures without motion capture hardware. The system is developed to accommodate any sign language, but a prototype has been built for the Sinhala sign language (SSL). The avatar system is initially built with 10 bones per arm. It is then extended to 29 bones per arm to improve flexibility in palm and fingers. The avatar animation framework is capable of signing 200 plus gestures of both single and multi posture SSL signs and 40 fingerprinting SSL signs. Speed and uniformity of the sign gesture animation is achieved by the automatic calculation of intermediate sequences of arm movements of signs within a given number of frames, or with a user defined increment value per bone. Animation system also supports the definition of fingerprinting signs, with a user defined tag set corresponding to the alphabet of any given sign language.**

***Keywords*— 3D Signing avatar, Animation framework, Sinhala Sign Language.**

I. INTRODUCTION

Express one's ideas to an aurally handicapped person requiring services of a trained sign interpreter who is a scarce resource in any country. Computer scientists throughout the world have tried to solve this problem using animated human models on demand, to perform a sign language gesture animation.

Instead of moving a 3D hand based on keyboard inputs, it is essential to initiate hand movement using an ordered set of commands when a sign language gesture is animated. There are two types of sign gestures in a sign language: static and varying gestures. For example, a word such as "you" (ඔබ) shows the right hand index finger pointing towards the observer in a static posture in Sinhala sign language (SSL). Therefore, irrespective of the initial position of the arm, the final position of the index finger is taken as the gesture. The latter considers the movement of the arm from one posture to another to represent a word in laymen language. For example, representing the word "ugly" (කැත) in Sinhala sign language requires moving the right hand little finger clockwise around the face while the other fingers are flexed [2]. One gesture per Sinhala word is more common in SSL, whereas a single gesture exists for entire phrase, such as "I love you" (ඔබ ඔබට ආදරෙයි) and "how are you" (ඔබට කොහොමද).

In a majority of techniques available to animate a sign language's gestures, a human model that moves the fingers in one hand or both hands simultaneously is required. This process is also known as manual signing. Moreover, Phonetic symbols that contain complex torso movement and facial expressions combined with mouth movements are referred to as non-manual signing or fingerprinting.

Graphical modelling of a human avatar requires a lot of time, creativity and specialist skills. Therefore good graphical human models, mostly commercial in nature, are usually built for computer games and the film industry. iCommunicator [1] for American Sign Language (ASL), which is a commercial signing application, utilizes such commercial graphical models coupled with expensive motion capture hardware for making video sequences to model human gestures using complex graphic cards.

II. RELATED WORK

A sequencing of motion-captured British sign language (BSL) words based on English sentences is the technique presented by Pezeshkpour et al. [3] to animate BSL. Gestures animated in a computer terminal using the TESSA framework supports conveying most common messages to a deaf person in British post offices with BSL. Vocabulary of the TESSA system is limited only to common words/phrases used in British post offices [4]. Adding new gestures to both these systems involves an expensive motion capture process.

To reduce the motion capture process for each and every sign, the Signing Gesture Markup Language (SiGML) notation is used as presented by Elliott et al. [6], based on the HamNoSys version 2 by Prillwitz et al. [7] under the ViSiCAST project [5]. The SiGML notation supports defining signs of BSL, the Dutch and the German Sign languages. SiGML notation represents a sign gesture as a collection of movements of different parts of a human body in an XML-like notation. SiGML has a core set of commonly used hand postures while other hand poses are defined by the transition of hand location from these core sets of hand postures. Transitions can take the form of a straight line, rotation or zigzag motion [9]. A virtual avatar, vGuido, is developed with the contributions made by multiple institutions in the European Union under the e-SIGN project [8] to perform a synthesis of signs in SiGML notation from the content of government web sites and weather forecasts in Europe. Defining a sign in SiGML notation is a complex task even with a supporting editor because it requires a complete and comprehensive knowledge of existing core gestures.

Kaneko et al. has developed a technique of word-to-word translation of Japanese text to Japanese Sign Language (JSL) using TV program Making Language (TVML) [11]. TVML is a text-based markup language that generates graphically animated TV programs by simply writing a script in TVML

and played in a TVML player. TVML supports embedding motion data of bio vision hierarchy (BVH) format [12] into key frame positions and supports moving the scene camera to point a specific virtual character with different close up positions. Researchers have also developed a Microsoft DirectX based new TVML player to display sign gesture animations of the JSL [10]. This technique also requires expensive motion capture data of sign gestures.

M. Huenerfauth presented a classifier predicate based decoding technique to represent a sign gesture in ASL on different body/hand channels for the animation, and later improved it to embed motion captured facial expression data to signing virtual avatars[13]. R. Wolfe et al. developed a Finger spelling tutor for teaching finger spelling to school children [14]. Paula Signing tutor was developed by M. J. Davidson to train ASL for hearing adults [15]. Both software systems utilize 3D signing avatars to train ASL.

This paper presents an improved 3D signing avatar and an animation framework which has the following characteristics:

- Supports the animation of words, phrases and finger printed signs based on posture coordinates.
- Provides support for defining sign gestures as a group of combined hand/body poses
- Allows for the movement of avatar hands to a final position irrespective of initial position
- Manages movement of different skeleton components by means of an auto calculated or user-defined increment
- Provides the ability to add new gestures to the system without altering code.

The asynchronous nature of the avatar animation framework supports the accommodation of new sentences to the animation playlist while an animation is being played back. Seamless switchover to the character-to-character finger printing technique using a specific sign alphabet is supported when an unknown word is found in a sentence for which an animation does not exist in the database of known gestures.

III. Methodology

A. Human Avatar for Sign Animations

An animation avatar requires the modelling of a human figure in a graphical environment as a mesh and covering it with texture. A bone rig is then attached to the avatar with different animation channels attached to the bone rig. The Makehuman software, which provides facilities to build a human avatar based on gender, age and skin colour (African, Caucasian and Asian), is used in the present research. Makehuman does not support face rigs, hence the model is exported in Makehuman exchange (MHX) format without a skeleton attached to the mesh. The human model generated by Makehuman is imported to Blender animation software and the automatic rigging facility is then invoked.

The camera has to be positioned facing the human model, and proper lighting needs to be enforced on the model to minimize shadows and to display proper arm movements for all the signs. In this study, three light sources and one scene camera is focused on the human model, displaying the upper body of the avatar in the view port. Change of viewport resolution and camera zoom facility is facilitated through the avatar animation framework.

B. Animation Engine and Scripting Environment

The dynamic nature of the sign language animations require the real time rendering of each display frame. Therefore, the blender game engine (BGE) is selected as the animation engine. The BGE works in sensor, controller and actuator modes. When a defined sensor is triggered, the BGE initiates action/activity through an actuator. The "Always Sensor" in BGE, which runs in the idle frequency of 60Hz with a python controller, is used for the present research.

C. Human Avatar Rig Structure for the Sign Animation

To animate a given sign gesture, each bone/bone group of the model has to be animated independently. The arm is broken down into sub components named clavicle, deltoid, upper arm, fore arm, hand (palm) and five fingers as depicted in Fig. 1. Fingers have multiple bones of which the pose coordinates are measured, calculated and animated separately.

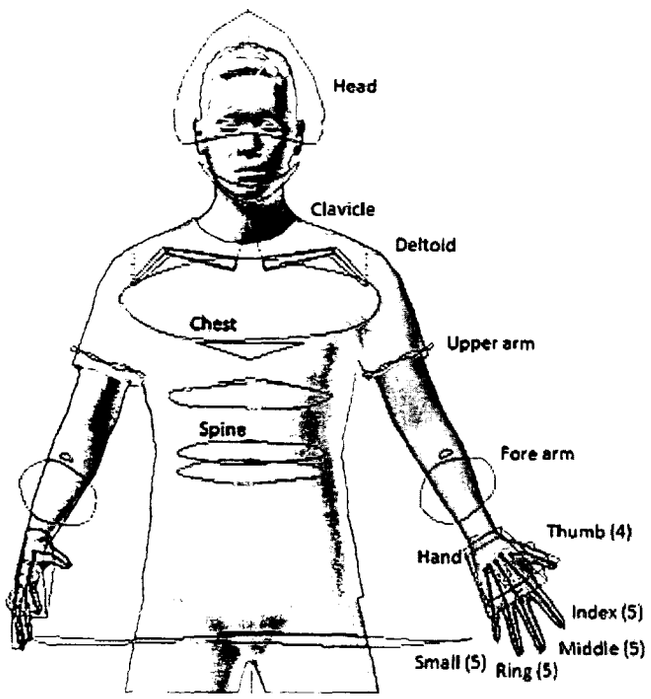

Fig. 1 Human model with different bones/bone groups that are animated. Number of channels in each bone is one unless specified in brackets

Human model used in the study also have limited head and upper body movement so as to animate gestures with head tilting as well as turning of the upper body. Abnormal movements such as the hand going across the body is limited.

Each bone in the animated human model has an idle position. Any bone that exhibits a movement out of its idle position gradually falls back to its idle position if the next gesture does not require a movement from the idle position. Table I illustrates how the rotations of each arm are defined.

TABLE I
ROTATION FUNCTIONS USED FOR COMPONENTS IN THE ARM AND BODY

Rotation function	Bone list utilizing the rotation function
Quaternion (W,X,Y,Z)	Clavicle, Deltoid, Upper arm , Each finger ,Head, Spine, Chest
Eular (X,Y,Z)	Forearm, Hand, Bone parts of each finger

D. Sign Gesture Classification

Sign gestures are classified into three categories. The first category requires arm/ fingers to end up in a static posture

irrespective of the initial arm position. The second category forms an animation sequence related to a gesture from a known starting position to a known final position. Animating such a gesture requires moving the arms from an unknown position to a known starting position, and subsequently, to a known final position. As this involves multiple hand poses, it is specified as a multi-posture sign gesture. The third category of sign gestures is the use of a group of gestures corresponding to alphabetical letters to create a sign. The best example is gesturing one's name using sign language. As a sign is not defined for a particular name, it has to be finger printed.

E. Animation Algorithm and the Sign Data Structure

Animating a human model according to a given sign gesture has to commence from the present position of each bone. Two techniques are used to determine the amount of movement of each bone towards its final position.

In the first technique, the animator specifies the maximum number of frames available for a sign to complete, from starting position to final position. An increment per bone is calculated using the formula given in Fig 2 based on the number of frames.

$$\forall_k \forall_i \quad Increment_{k,i} = \frac{|Start\ Value_{k,i} - End\ Value_{k,i}|}{Number\ of\ frames}$$

$$k = \begin{cases} W,X,Y,Z, & rotation\ type = Quaternion \\ X,Y,Z, & rotation\ type = Eular \end{cases}$$

$$i = bone\ 1 .. bone\ n$$

Fig. 2 Formula to obtain increment used in the calculation of next bone position

In the second technique, the animator has the flexibility to specify an increment value for each bone. A global adjustment for the value of each bone can be specified to accommodate the speed control of animation in slow machines. This technique gives total control to the animator to move specific bones at lower speeds than some of the other bones.

To animate a static posture it is required to have the set of coordinates of the final arm positions of the 29 bone channels in each of the left and the right arms. In addition, three channels are present for head, chest, and spine movements.

The loop that executes the BGE script through the "Always Sensor" is executed once every 1/60 seconds as per the game engine default setting. Because the values in the variables are initialized in every loop, text files are used to store the states and animation playlist that needs to be persistent across script rounds. In addition, the initial and the current XYZ Euler coordinates of a bone or WXYZ quaternion coordinates of each bone is stored as a property variable. The playlist and other necessary temporary files that are dynamically generated by the BGE script are saved as text files. Inverse kinematic features available in the human model is turned off to prevent automatic movement of other bones when one bone is moved by the animation script. The algorithm given below is used to change the properties/coordinates of each bone.

```
ARM ANIMATION ALGORITHM:
//relevant rotation function line is invoked for each bone channel
// rotation_euler function is for a bone with Euler rotation
// rotation_quaternion function is for a bone with quaternion rotation
// position refers to W,X,Y,Z coordinates
// increment is user defined or calculated based on number of frames
 For each bone in bone list
 For Each position in bone of given bone
  IF current position not equal to new position
    IF current position > new position
      IF (current position – new position) <= increment
          current position = new position
      else
          current position = current position – increment
      end if
    end if
    IF current position  < new position
      IF (new position – current position) <= increment
          current position = new position
      else
          current position = current position + increment
      end if
    end if
  end if
end for
end for
 rotation_eular (bone, (currentposition XYZ /radian conversion))
 rotation_quaternion (bone, current position WXYZ)
If Wait for required frames till given pause of posture expires
Signal bone animation complete
end if
```

F. Data structure of Gesture Vocabulary

Five lines of expressions define a gesture with static posture. Multi-posture-based signs contain multiples of five lines to animate the hands/head/upper-body to different positions. The composition of data in these five lines are as follows:

Line1: laymenword, single/multiple, bothhand, number_of_lines_in_gueture, coordinates of CVL separated by commas, increments of CVL separated by commas, coordinates of CVR separated by commas, increments of CVR separated by commas,.., coordinates of HAL separated by commas, increments of HAL separated by commas, coordinates of HAR separated by commas, increments of HAR separated by commas, waittime, needtowait Y/N.

Line2/3: laymenword, single/multiple, fingerleft / fingerright, no_of_lines_in_gueture, coordinates of TFL separated by commas, increments of TFL separated by commas, coordinates of TFR separated by commas, increments of TFR separated by commas,.., coordinates of SFL separated by commas, increments of SFL separated by commas, coordinates of SFR separated by commas, increments of SFR separated by commas, waittime, needtowait Y/N.

Line4: laymenword, single/multiple, head, no_of_lines_in_gueture, coordinates of Head separated by commas, increments of Head separated by commas, waittime, needtowait Y/N.

Line5: laymenword, single/multiple, chestspine, no_of_lines_in_gueture, coordinates of Chest separated by commas, increments of Chest separated by commas, coordinates of Spine separated by commas, increments of Spine separated by commas, waittime, needtowait Y/N.

G. Modular Structure of the Sign Gesture Animation Plan

Although typical sign language does not contain gestures for every word of a natural language, gesture may represent a phrase of a natural language. The important part of an

animation framework, which controls the 3D signing avatar, is to build the animation playlist of sign gestures based on a given list of sentences in a natural language. This framework supports dynamically appending the sentences to the sentence list. The modular structure of the animation plan and the associated files are given in Fig. 3.

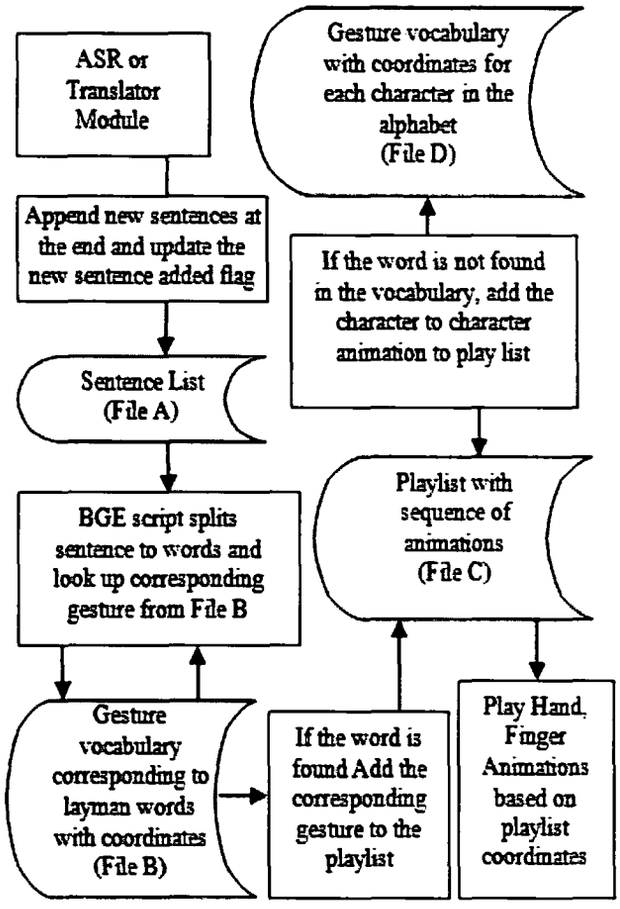

Fig. 3 Sign gesture animation plan

File A: Contains the sentences of a natural language populated either by ASR or any other mechanism.

File B: Contains the list of sign gestures corresponding to each known natural language word or a phrase. Both multi and static posture sign gestures are defined in this file.

File C: Contains the playlist of the sign gesture definitions sorted in the playing order according to the sentences added to File A. This will be dynamically generated and expanded by the animation framework in every sign gesture play session.

File D: Contains the list of gesture definitions for the phonetic alphabet of the natural language to construct laymen language words not found in the sign language.

Three more files are used in addition to the above to store the number of sentences read from file A, to store global parameters related to the animation framework and to specify the set of phonetic symbols of the alphabet of the laymen language.

SEQUENCE ALGORITHM:

1. On application start clear word count and sentence count.
2. Play Idle Animation
3. If sentence count is less than sentences present in sentence file(File A) remember sentence count in file A and generate playlist (File C) otherwise go to step 4
4. If word count is less than word count in playlist file Read playlist (File C) and get next word to play from current position otherwise go to step 2.
5. Adjust the default increments of each bone position based on the parameters specified in global settings file. Use auto increment calculation module or user defined increments based on user preference.
6. Make the change to arm position based on the increment in each frame and play the frame in view port
7. Check the end of gesture play back comparing the current position with required position
8. Wait for the number of frames as specified in File C.
9. Update the word count (i.e. keep number of words that already played from the playlist).
10. If end of playlist is true go to step 3.

Based on the sequencing algorithm given above, the proposed system executes the steps 2,3,4,2 sequence until a new sentence is added to File A by the external component of the system (such as a speech recognizer or a web crawler to signal the animator framework in step 3 that there are new sentences to be played). The word count and the sentence count is kept within the application to continue building the playlist from the position where it has most recently stopped.

H. Playlist Format

The playlist file (File C) contains a list of sign data structures to be played sequentially. When a play list is built, a header line is introduced to each segment of sign gesture that is being played. The header line format given below excludes the header and colon sign. The static posture based sign in the playlist contains a header line, animation details of bones in both hands in line 2, animation details of left hand and right hand fingers in lines 3 and 4, animation details of the head in line 5 and the animation details of the chest and spine in line 6 respectively.

Header : Linenumber# NP/P (play/notplay)# laymen word#single/multiple/fingerspell#no_of_lines_in_symbol#ext raparameters

I. Definition of a Sign in Animation Database

The human model must be loaded into the Blender software to define a new gesture in the animation database of the signing avatar. The arms of the model are moved to the final position of the gesture using the mouse or keyboard shortcuts. At this point, the Blender software displays the coordinates of each bone when a bone is selected. Table II is populated using the coordinates of the bone positions and subsequently, the sign vocabulary (file B) is defined using the posture coordinates.

TABLE II
POSTURE COORDINATES OF A STATIC POSTURE SIGN

Bone	Left Hand				Right Hand			
	X	Y	Z	W	X	Y	Z	W
CV	0	0	0	1	0	0	0	1
DT	0	0	0	1	0	0	0	1
UA	0.25	-0.2	1.32	0.04	-0.5	-0.8	2.2	0.07
FA	100	59.5	21.5	---	15.4	2.1	24.5	---
HA	87.5	96.1	-3.8	---	9.8	64.3	128	---

Bone	Left Hand				Right Hand			
TF-SF	0	0	0	1	0	0	0	1
TF1-SF3	0	0	0	---	0	0	0	---
IF4-SF4	0	0	0	1	0	0	0	1
HD, CHT, SPN	0	0	0	1	0	0	0	1

As seen in Table II, a value of 0 for the XYZ and a value of 1 for the W is given as the idle positions of a particular bone. A multi-posture sign gesture contains two or more sets of posture coordinates for different middle positions of each sign gesture. Also, every position of a bone consists of a corresponding user-defined increment value as defined by the animator to have different speeds of the arm movement across a series of frames adhering to technique 1 given in section E.

IV. Result and Discussion

3D avatar and its animation framework is first tested with a small vocabulary of 15 SSL signs and 32 fingerprinting signs of Sinhala sign alphabet. Definition of a sign with 10 bone channels and only user defined increments are supported [16].

The system presented in this paper includes automatic calculations of bone increment based on the number of frames to improve the speed of the animation. The number of bone channels increased to 32 to improve flexibility of hand of the model and to add support for animating head and upper body of the model.

A larger vocabulary of 200 SSL signs and 40 finger printing signs are defined in the animation database and is used to test the 3D signing avatar with a manually typed sentence file of nine sentences. Test sentences include static posture (single posture), multi posture, finger spelling and different combinations of categories of signs as given in table III.

TABLE III
Sentences used in Testing the 3D Avatar for SSL Animation

Sentence	English meaning	Posture categories
ඔබට ආයුබෝවන්	Long life for you	Two Single posture words
ඔබට කොහොමද	How are you	Multi posture one gesture for phrase
වතුර අපිරිසිදු	Water is dirty	Two multi posture
මම පාඩම් කරනවා	I am studying	Single posture first word, next 2 words Multi posture phrase
මල්ලි බඩේ කැක්කුම	Younger brother has stomach-ache	Multi posture first word, next 2 words Multi posture phrase
තාත්තා බත් කනවා	Father is eating rice	Three multiple posture words
ඔබ කවුද	Who are you (question)	First word single posture, 2nd word multi posture
මම උදාර	I am udara	First word single posture, 2nd word finger spelled name
ශාන්ත වාඩිවෙන්න	Shantha sit	Finger spelled first word including multi posture character, 2nd word multi posture

The sentences file is a phonetic English text file that can be populated using ASR, Web page content or any other information extracting system. The 3D animated avatar has animated all the tested sentences correctly by looking up signs from the animation database and populating the playlist. Body/arm position increments are calculated according to parameters as specified in the global settings file. It has successfully switched to finger spelled animation mode whenever an unknown word occurs in a sentence.

The screenshots of animating two word sentences in Sinhala language "ඔබට ආයුබෝවන්' (obata aayuboovan', meaning "long life for you") is depicted in Fig. 4. Both words belong to the category of static postures for which the final position of the arms is used to express the meaning of the word.

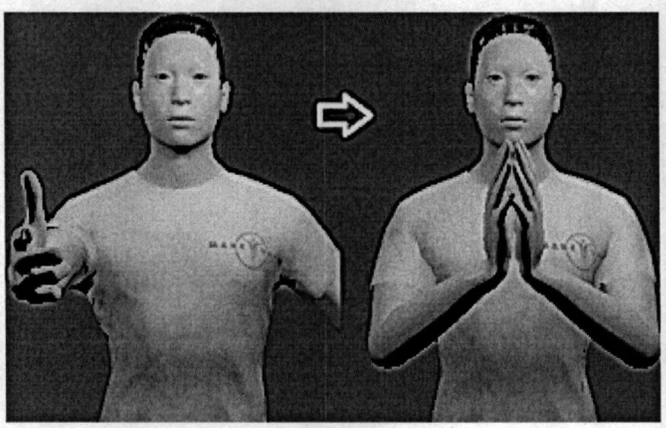

Fig. 4 Animating two static posture words in a sentence

The second example, having a multi-posture sign gesture, is used to animate the phrase "ඔබට කොහොමද" ("obata kohomadha", meaning "how are you") as depicted in Fig. 5.

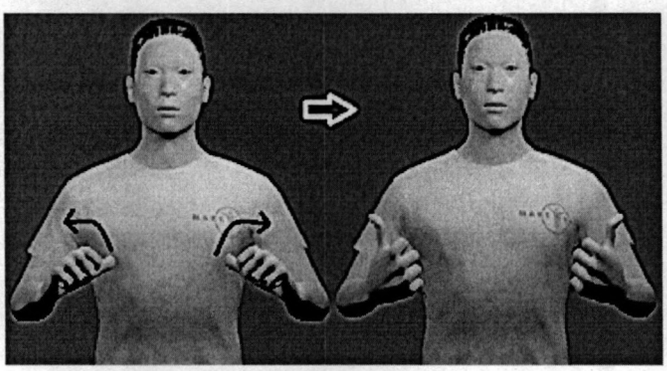

Fig. 5 Animating multi-posture gesture

It must be noted that although the general word based component of the avatar animation system is universal to any sign language, the phonetic symbols of the finger spelling component has to be defined according to a specific natural language that matches the corresponding sign language. Finger spelling phonetic symbols of the Sinhala language is defined using a four-line format in an external file that is loaded to the avatar animation system as given below.

cnssingle = " a,i,u,e,o,k,g,t,d,n,p,b,m,y,r,l,v,s,h,f "
cnsdual = " zk,zg,ch,jh,cn,zt,zd,zn,dh,zp,zb,sh,zl "
cnstriple = " axa,ixi,uxu,eze,oxo,xau,zri,zon,zch,zjh,zth,zdh, zsh,xon,zng,zau,zru,zkf,xmb,xae,txh,qxr "
cnslarge = " aeae,zrii,ziluu,zilu,jhcn,qndh,zruu,zndx,zaik,xjhx, qxya"

In the third example, a finger-spelled word that does not have a sign gesture is animated. A person's name such as "udara" which is pronounced as "udaara" (උදාර) is animated as given in Fig. 6.

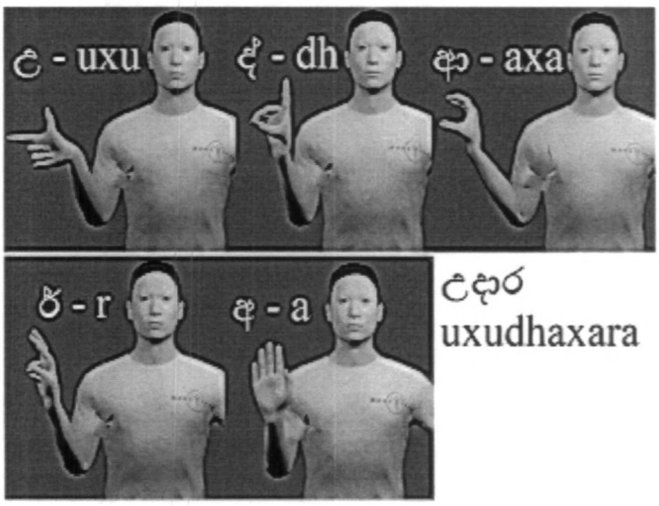

Fig. 6 Animating Finger spelled word udara

3D avatar animation system first looks for known phrases in a given sentence, then looks up for known words. All the remaining words are treated as unknown words. The animation script identifies the name udara as an unknown word, then decodes it to phonetic tags uxu + dh + axa + r + a (උ + ද් + ආ + ර් + අ). It then looks up the relevant phonetic sign gestures through a file containing the phonetic vocabulary (file d) and appends the gesture coordinates to the playlist file. Finally, the finger-spelled word is played letter-by-letter as a regular word.

In comparison to the techniques presented in [3], [4], [11], and [13], the proposed system provides a means of animating sign gestures without motion captured data.

The system developed by Elliott et al. [6] contains animations recorded as video clips and requires training the SiGML to define a new gesture. Moreover, it is limited to specific European sign languages with their own avatars. The technique proposed by Kaneko et al. [11] requires training the TVML to define a sign gesture. In contrast, the 3D signing avatar developed for SSL is capable of defining any sign with the sole requirement of identifying gesture coordinates of the final bone position. The animation script developed with the avatar, writes the bone coordinates of current pose to a text file so that the animator can easily populate the sign database with the current pose of the human model when a new sign is added.

V. Conclusions

Further experiments must be carried out with the animation system of the 3D avatar to animate facial expressions, synthesis of numbers, and dates.

A human avatar, with the same bone dimensions developed with Makehuman software, could be swapped with the current human model of the system. Textures are dynamically loaded from external files, hence body color and clothes can be enhanced or changed without making any modification to coding. Further work concerning proportioning and scaling is required if the bone dimensions of a new model is different to the current human avatar.

Nevertheless, 3D human avatar and the animation system with the new phonetic tag set is an improvement on the initial work [16] by the same authors. It provides a complete system to a sign animator of any sign language in both word/phrase-based animation as well as pronunciation-based finger printing even though the prototype and sign database is built for the Sinhala sign language. It is implemented with open source software and does not require expensive motion capture hardware to define new signs in the sign database.

Acknowledgment

This research is funded by National Science Foundation of Sri Lanka.

References

[1] PPR Direct, Inc, Brooklyn, New York, (2006) "iCommunicator - Speech to Text to Sign-Language Conversion... in real time!", [Online] available:http://www.icommunicator.com/downloads/iCommunicator-infosheet.pdf

[2] A. Stone, "An Introduction to Sri Lankan Sign Language", N. D. Abeygunawardana et al., Ed., Rohana Special School, Matara, Sri Lanka, 2007.

[3] F. Pezeshkpour, I. Marshall, R. Elliott, and J. A. Bangham, "Development of a Legible Deaf Signing Virtual Human", *in Proc. IEEE ICMCS'99*, Florence, Italy, vol. 1, pp. 333-338, 1999.

[4] S.J. Cox et al., "TESSA, a system to aid communication with deaf people.", *in Proc. ASSETS'2002*, New York, USA, pp. 205-212, 2002.

[5] (2000) ViSiCAST project website. [Online]. Available : http://www.visicast.co.uk/

[6] R. Elliott, J. Glauert, R. Kennaway and K. J. Parsons, "D5-2: sigml definition. working document", University of East Anglia, UK, ViSiCAST Project, 2001.

[7] S. Prillwitz, et al. "Hamburg notation system for sign languages - an introductory guide." *International Studies on Sign Language and the Communication of the Deaf*, University of Hamburg, Germany, vol. 5, 1989.

[8] (2002) eSIGN project web site. [Online]. Available : http://www.sign-lang.uni-hamburg.de/esign/

[9] J. Glauert, R. Kennaway, R. Elliott and B. J. Theobald, "Virtual human signing as expressive animation", *in Proc. AISB'2004*, UK , pp. 98-106, 2004.

[10] H. Kaneko, N. Hamaguchi, M. Doke and S. Inoue, "Sign language animation using tvml", *in Proc. VRCAI '10*, Seoul, Korea, pp. 289-292, 2010.

[11] (2014) TVML Language Specification Version 2.0 [Online]. Available : http://www.nhk.or.jp/strl/tvml/english/onlinemanual/spec/index.html

[12] (2000) BVH File Specification. [Online]., Available : http://www.character-studio.net/bvh_file_specification.htm

[13] M. Huenerfauth, "Learning to Generate Understandable Animations of American Sign Language", *In Proc, Effective Access Technologies Conference*, Rochester, New York, USA. June 17-18, 2014.

[14] R. Wolfe, M. J. Davidson, J. McDonald and F. Carrie, "Using an Animation-based Technology to Support Reading Curricula for Deaf Elementary Schoolchildren", *In Proc, The 22nd Annual International Technology & Persons with Disabilities Conference*, Los Angeles, CA March 21, 2007.

[15] M. J. Davidson, "PAULA: A Computer-Based Sign Language Tutor for Hearing Adults", *In Proc, The Eighth International Conference on Intelligent Tutoring Systems (ITS 2006): Teaching with Agents, Robots and NLP*, Jhongli, Taiwan, 2006.

[16] M. Punchimudiyanse and R. G. N. Meegama, "3D Animation framework for sign language", *In Proc, International Conference on Engineering and Technology*, March 17-18, 2015, Colombo, Sri Lanka.