3D Animation framework for sign language

M. Punchimudiyanse
Department of Mathematics and Computer Science
Faculty of Natural Sciences
The Open university of Sri Lanka
Nawala, Nugegoda, Sri Lanka

R.G.N. Meegama (rgn@sci.sjp.ac.lk)
Department of Computer Science
Faculty of Applied Sciences
University of Sri Jayewardenepura
Nugegoda, Sri Lanka

Abstract

A critical part of animating a sign language using virtual avatar is to display a sign gesture having multiple rotational arm poses to identify a word instead of a single static arm pose. Sequencing a group of gestures related to a sentence requires each gesture in the middle of a sentence to be animated using different initial arm positions. Sequencing pre-captured arm videos, ordering preset animations compiled by 3D animations, and ordering motion capture data are the widely used techniques used by sign language animators presently. The transition from one word to another is not smooth as the initial and the terminating positions of each animation is not the same. This paper presents a technique with smooth transitions between gestures to animate a sign language. A sequencing technique is also presented to animate known words using gestures that are already defined and also to animate unknown words using character-to-character sign animation. New sentences are dynamically added in real-time and the system will adjust the animation list automatically by appending the required animations of the words in the new sentence to the end of the playlist. Results indicate an average distance of 3.81 pixels for 27 static pose finger spelling characters.

Keywords: Animation framework, Sign gesture animation, Virtual avatar, Gesture sequence model.

Introduction

Animating a human model on demand is a key part of Sign Language gesture animation. Commercial laymen language to sign language applications such as iCommunicator (PPR Direct, 2006) for American Sign Language (ASL) uses commercially available graphical human models built for computer games or film industry for making video sequences using complex graphic cards and sophisticated and expensive motion capture hardware to model human gestures.

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 laymen 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. 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 (ed. Stone, 2007).

In a majority of techniques available to animate a sign language gestures, a human model that moves the fingers of either 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 finger printing.

Literature Survey

A technique developed by Pezeshkpour et al.(Pezeshkpour et al., 1999) makes use of a sequencing mechanism of motion captured British sign language (BSL) words based on English sentences. The TESSA framework developed by Cox et al. (Cox et al., 2002) uses the BSL gestures to ease communication barriers between a healthy and a deaf person in British post offices. Adding new gestures to both these systems involve expensive motion capture process.

The Signing Gesture Markup Language (SiGML) notation breaks down a sign gesture into a collection of movements of different parts of a human body in an XML-like application language developed by Elliott et al. (Elliott et al., 2001) under the ViSiCAST project (ViSiCAST project, 2000) based on the HamNoSys version 2 by Prillwitz et al.(Prillwitz et al., 1989). This markup language has a core set of commonly used hand postures such as flat hand and fist while other hand poses are defined by the transition of hand location from these core set of hand postures. Transitions can take the form of a straight line, rotation or zigzag motion (Glauert et al., 2004). The SiGML notation supports defining signs of BSL, Dutch and German Sign languages. A virtual avatar, vGuido, is developed under the e-SIGN project (eSIGN project, 2002) to perform a synthesis of signs in SiGML notations from the content of government web sites and weather forecasts in the European Union. 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 to word-to-word translation of Japanese text to Japanese Sign Language (JSL) using TV program Making Language (TVML) (TVML, 2014). 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 (BVH, 2000) 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 (Kaneko et al., 2010). This technique too requires expensive motion capture data of sign gestures.

In this paper, we propose a novel animation framework having the following characteristics:

a) complete control to the sign animator
b) support for defining sign gestures in non hierarchical manner
c) animate hand gestures based on posture coordinates

d) move hands to a final position irrespective of initial position
e) manage movement of different skeleton component by an external increment
f) ability to add new gestures to the system without altering code .

The proposed animation framework is not bound to a specific human model and as such, the animator can change the human model at any time, add new skeleton bending points and improve the script on new movements. The asynchronous nature of the framework supports accommodating new sentences in real time to while an animation is being played back. Seamless switchover to character-to-character finger printing technique using 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.

Methodology

Building human avatar for sign animations

Building an animation avatar require modeling a human figure in a graphical environment using a mesh and covering it with appropriate texture. The Makehuman software, which provides facilities to build a human avatar based on gender, age and skin color (African, Caucasian and Asian) is used in the present research. Because Makehuman does not support face rigs, 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 which automatically initiates the rigging facility (adding a bone skeleton to the mesh) to the model to be animated with different animation channels attached to skeleton bones.

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, 3 light sources and one scene camera is focused to the human model displaying the upper body of the avatar in the view port.

Animation engine and scripting environment

As the sign language animations, it is required to have real time rendering of each display frame. Therefore, the blender game engine (BGE) has to be 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.

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 Figure. 1.

Animators are free to include more channels for the animation because the data structure that defines gesture contains sufficient flexibility to add new channels. As each bone in the animated human model has an idle position, any gesture that does not exhibit a movement in a specific channel automatically sets in an idle position.

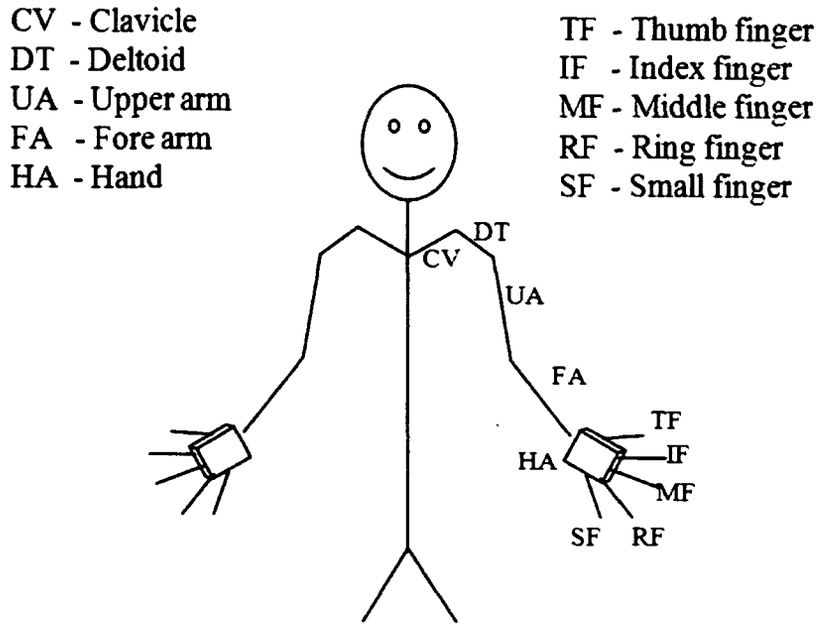

Figure 1. *Rig structure and naming of bones.*

Table 1 illustrates how the rotations of each arm are defined. Two types of rotation functions are used. The quaternion function avoids gimbal lock in 3D rotation.

Table 1. *Rotations functions used for components in the arm.*

Rotation function	**Bone list utilizing the rotation function**
Quaternion	Clavicle, Deltoid, Upper arm, Fingers
Eular	Forearm, Hand

Sign guesture 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 postures, 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.

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 the each bone structure. In the proposed technique, each bone is moved to the required position by incrementing the distance by $1/10^{th}$ of the final location at each frame.

To animate a static posture it is required to have the set of coordinates of the final arm position ten bone channels (CV, DT, UA, FA, HA, TF, IF, MF, RF, SF) for the left and the right arms.

The animator also has the flexibility to specify an increment value for each bone. A global adjustment to this value for each bone can be specified to accommodate control of the speed of animation in slow machines.

The loop that executes the BGE script through the "Always Sensor" is executed once every 1/60 seconds as 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 need 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. The algorithm given below is used to change the properties of each bone.

ARM ANIMATION ALGORITHM:

```
//relevant update position function line should be present for each bone channel
// Euler function will not work for a bone having quaternion rotation
// Quaternion function will not work for a bone having Euler rotation

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
      endif
    else
    IF current position  < new position
      IF (new position – current position) < increment
          current position = new position
      else
          current position = current position + increment
      endif
  endif
 end for
end for
Update positions rotation eular (bone, current position XYZ /radian conversion)
Update positions rotation quaternion (bone, current position WXYZ)
If Wait for required frames till given pause of posture expires
    Signal bone animation complete
endif
```

In order to simplify the scripting process, a four character position variable convention where the first 2 letters denote the boneid, the 3rd letter for left or right arm (L / R) and the last for the axis (X,Y,Z,W) is used. (eg. UALX – left upper arm x-axis position)

Sign data structure of gesture vocabulary

Three lines of expressions define a static gesture. The expression of the first line contains rotational coordinates of the both hands from clavicle up to the hand bone. The expression of the second and third line contain rotational coordinates of the fingers of the left hand and right hand respectively. Multi-posture-based signs contain

multiples of 3 lines to animate the hands to different positions. The composition of data in these 3 lines are as follows:

Line1: laymenword, single/multiple, bothhand, no_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.

The bone order of the expression in line 1 is CV,DT,UA,FA,HA. Line 2 and 3 has the bone order of TF,IF,MF,RF,SF.

Modular structure of the sign guesture animation plan

Although typical sign language does not contain gestures for every word of laymen language, gesture may represent a multiple words of a laymen language. The important part of an animation framework is to build the animation playlist of sign gestures based on a given list of sentences in laymen language. This framework supports dynamically appending the sentences to the sentence list. The animation framework detects the presence of a new sentences and the playlist file is appended accordingly. The modular structure of the animation plan and the associated files are given in Figure. 2.

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

File B: Contains the list of sign gestures corresponding to each known laymen word or 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 laymen 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.

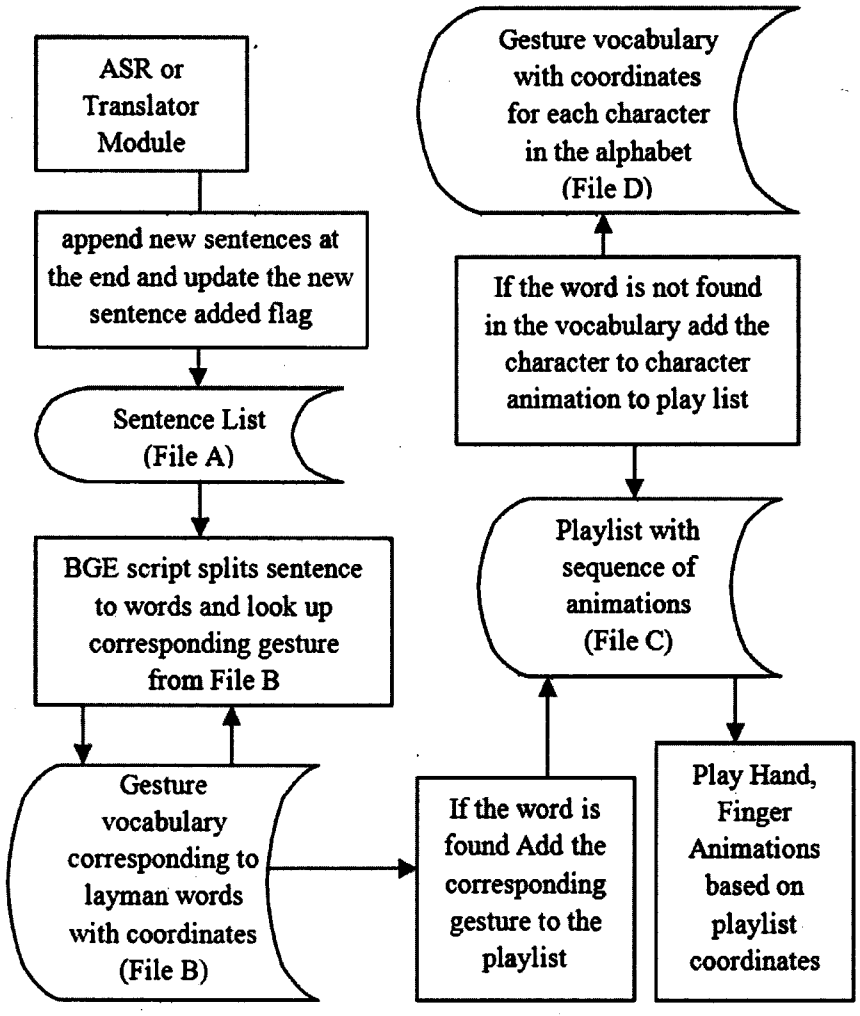

Figure 2. *Sign Guesture Animation plan.*

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 and the settings in File C related to each gesture using increment adjustment module.
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 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.

Increment adjustment

The increments specified for each bone position is adjusted according to global settings. For instance, if the speed of the animation is slow due to the speed of processor and the capability of graphics hardware, the global increment is increased to move the arm to the correct position in lesser number of frames.

Specific sign gestures require arms to be moved faster than movement of hand rotation or finger orientations. In such cases, the increments of the arm bones could be increased without changing the increments of the finger bones to make the movement customized based on the requirement of the sign gesture.

The other feature of this increment module is to have a rotation function of a bone around a given axis by ceasing the increment of the other axis while maintaining the original increment of the given axis based on the gesture header information.

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 header and the colon sign. The static posture based sign in the playlist contains a header line, animation details of bones in both hands in line 2 and animation details of both left hand and right hand fingers in lines 3 and 4, respectively.

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

Data Analysis

To test the animation framework, a prototype is built using BGE and a sentence file containing five sentences having static posture words, multi-posture words and unknown words is used.

A sign vocabulary consists of ten static posture signs and five multi-posture signs. Twenty seven static posture and five multi-posture finger spelling alphabet is used to animate unknown words.

The screenshots of animating two word sentences in Sinhala language "ඔබට ආයුබෝවන්" (obata aayuboovan', meaning "long life for you") is depicted in Figure. 3. Both words belong to the category of static postures of which the final position of the arms is used to express the meaning of the word.

The animation framework provided the expected functionality of decoding the sentence to two known gestures, looking up the sign database, populated the playlist and then animated each static posture sign according to the sign posture defined in the sign vocabulary.

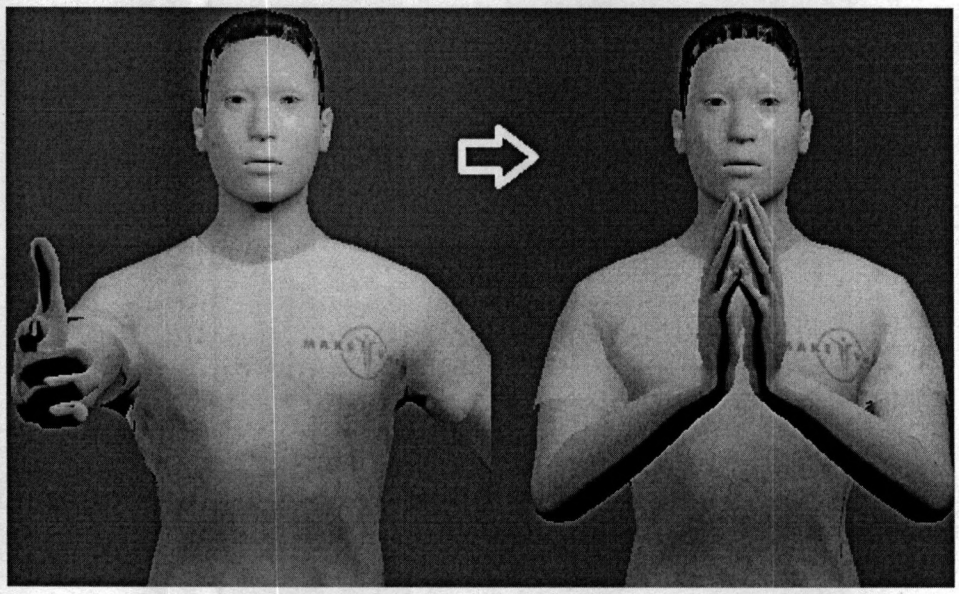

Figure 3. *Animating two static posture words in a sentence.*

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 Figure. 4. Animation framework properly displayed the multi posture sign gesture according to the definition listed in sign vocabulary.

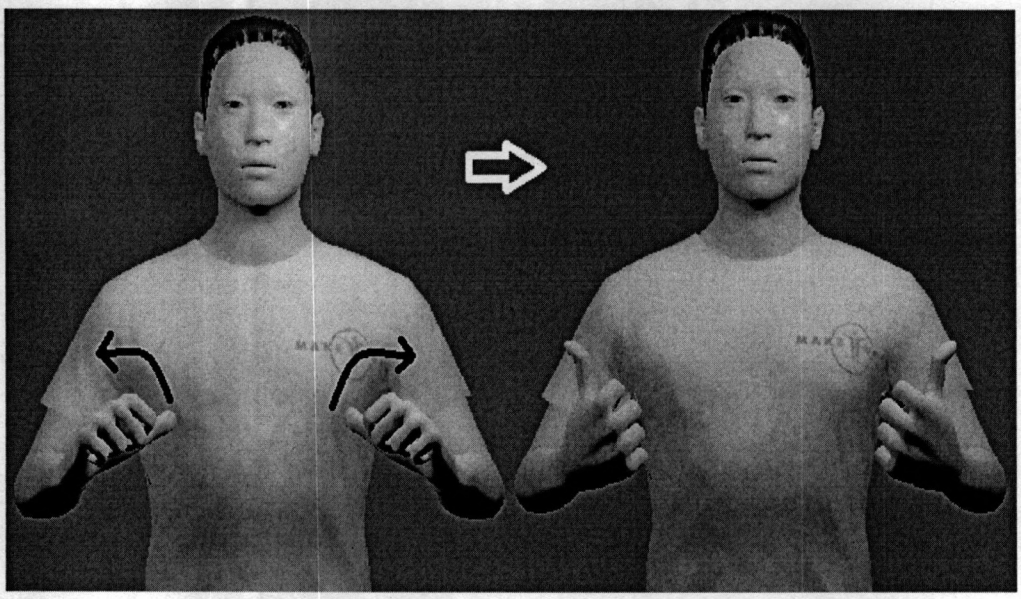

Figure 4. *Animating multi-posture gesture*

In the third example, a finger-spelled word that does not have a sign gesture is animated. A person's name such as "gayan" is pronounced as "gayaan" (ගයාන්) is animated as given in the Figure. 5.

The proposed animation framework first identifies it as an unknown word then decodes it to phonetic sounds g + a + y + aa + n (ග් + අ + ය් + ආ + න්). 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.

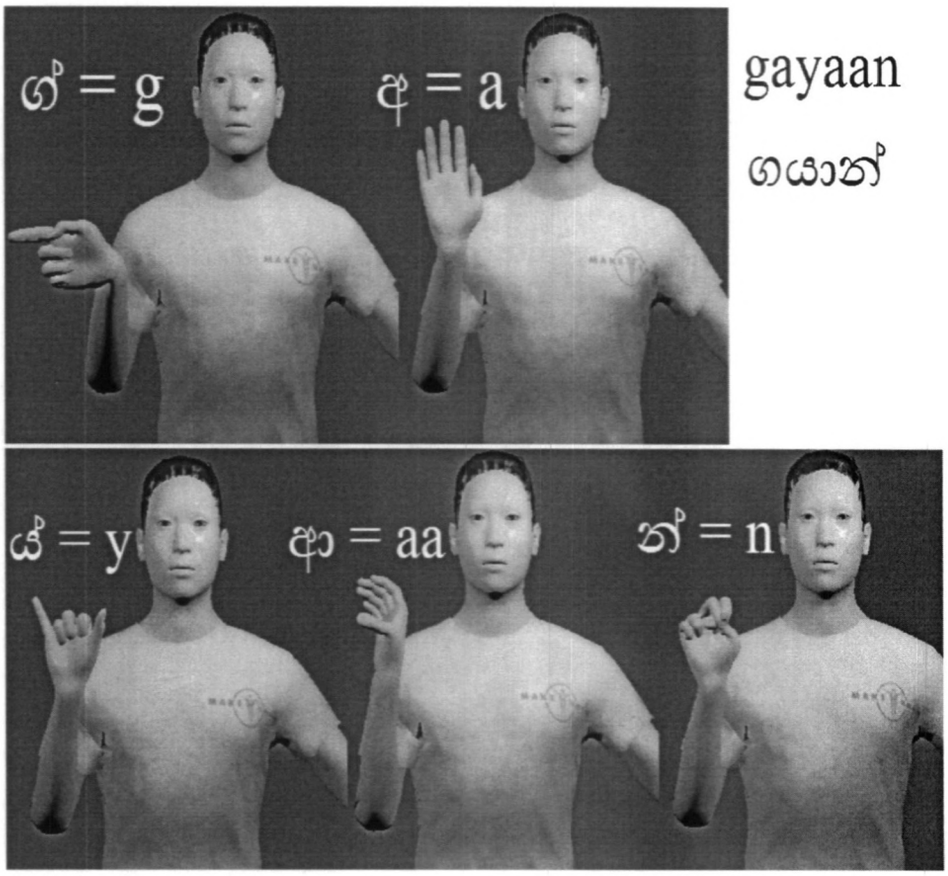

Figure 5. *Animating Finger spelled word gayan.*

The Sobel operator (Sobel and Feldman, 1968) is used to extract the contours of the hand in both the original and the output images. Then, the hand contour of a finger pose of a character in the sign alphabet is compared with the corresponding hand contour output of the finger pose generated by the animation framework.

The average distance error (Staib and Duncan, 1996) between the actual and the generated position of the hand gesture is used to compare the accuracy of the proposed technique as given in Table 2.

Overall, an average distance error of 3.81 pixels is observed for 27 static pose finger spelling characters. Although visual comparison between images does not show a significant deviation, original images of the sign alphabet drawn by hand using different scales contributed to higher pixel distances in some character signs.

Table 2. *Average distance error between output and original images of finger spelled signs (In Pixels)*

Char	Err	Char	Err	Char	Err	Char	Err
අ a	3.86	එ e	3.81	ල් l	2.19	ට් t	3.75
ආ aa	5.13	ඒ ee	6.98	ම් m	5.45	ත් th	3.51
ඇ ae	4.72	ෆ් f	2.57	න් n	2.19	උ u	5.80
බ් b	1.70	ග් g	3.05	ංo on	2.72	ඌ uu	3.63
ච් ch	7.54	හ් h	2.29	ප් p	3.02	ව් v	2.15
ඩ් d	6.18	ඉ i	2.64	ර් r	2.87	ය් y	5.43
ද් dh	5.81	ක් k	2.03	ස් s	1.96	Overall	3.81

Discussion and Conclusion

The definition of the sign is based on the posture coordinates. A typical static posture sign coordinates are collected as in the format defined in Table 3. These data are then defined in the sign vocabulary according to the format specified in section sign data structure of gesture vocabulary.

As seen in Table 3, 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 increment value defined by the animator to have different speeds of the arm movement across a series of frames.

Table 3. *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	---
TF	0	0	0	1	0	0	0	1
IF	0	0	0	1	0	0	0	1
MF	0	0	0	1	0	0	0	1
RF	0	0	0	1	0	0	0	1
SF	0	0	0	1	0	0	0	1

It must be noted that although the general word based component of the animation framework is universal to any sign language, the phonetic symbols of the finger spelling component has to be defined according to8 a specific layman language that matches the corresponding sign language. Finger spelling phonetic symbols of the Sinhala language is defined using a three-line format in an external file that is loaded to the animation framework as given below.

cnssingle#a,i,u,e,o,k,g,t,d,n,p,b,m,y,r,l,v,s,h,l,f
cnsdual#aa,ae,ii,uu,ee,ai,oo,au,on,ng,ch,jh,cn,nd,th,dh,mb,sh
cnslg#aeae,jhcn,njh,ndh

The human model must be loaded into the Blender software to define a new gesture for this animation framework. 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 3 is populated using the coordinates of the bone positions and subsequently, the sign vocabulary (File B) is defined using the posture coordinates.

In comparison to the techniques presented by Pezeshkpour et al., Cox et al. and Kaneko et al., (Pezeshkpour et al., 1999; Cox et al., 2002; Kaneko et al., 2010) the proposed system provides a means of animating sign gestures without motion captured data.

The system developed by Elliott et al. (Elliott et al., 2001) 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. (Kaneko et al., 2010) requires training the TVML (TVML, 2014) to define a sign gesture. In contrast, the proposed framework is capable of defining any sign with the sole requirement of identifying gesture coordinates of the final bone position. .

Further experiments must be carried out to animate facial expressions and also to obtain a smooth animation that prevents arm movements bisecting the body.

Nevertheless, the proposed technique provides a complete framework to an animator of any sign language in both word/phrase-based animation as well as pronunciation-based finger printing. It is implemented with open source software and does not require expensive motion capture hardware. Moreover, it provides complete flexibility to the animator to move specific bones at different speeds.

Acknowledgements

This research is funded by National Science Foundation Technology grant number: TG/2014/Tech-D/02

References

PPR Direct, Inc. (2006). *iCommunicator - Speech to Text to Sign-Language Conversion... in real time!.* [ONLINE] Available at: http://www.icommunicator.com/downloads/iCommunicator. [Accessed 08 February 14].

Stone, A. (Ed.)(2007) An introduction to Sri Lankan Sign Language. Karunaratne & Sons, Sri Lanka.

Pezeshkpour, F., Marshall, I., Elliott, R. and Bangham J.A. (1999) Development of a legible deaf signing virtual human. *In - Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Florence, Italy, vol. 1, pp. 333-338.

Cox, S.J. et al. (2002) TESSA, a system to aid communication with deaf people. *In - Proceedings of Fifth International ACM Conference on Assistive Technologies (ASSETS)*, Edinburgh, Scotland, UK, pp. 205-212

ViSiCAST project (2000) ViSiCAST project website. [Online]. Available : http://www.visicast.co.uk/. [Accessed 30 April 14].

Elliott, R., Glauert, J., Kennaway, R. and Parsons, K. J. (2001) D5-2: SiGML definition. working document, University of East Anglia, UK, ViSiCAST Project

Prillwitz, S. et al. (1989) HamNoSys Version 2.0 : 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

eSIGN project (2002) eSIGN project web site. [Online]. Available at : http://www.sign-lang.uni-hamburg.de/esign/. [Accessed 17 May 14].

Glauert, J., Kennaway, R., Elliott R. and Theobald, B. J. (2004) Virtual human signing as expressive animation, *In - Proceedings of Symposium on Language, Speech and Guesture for Expressive Characters, The Society for the study of artificial intelligence and simulation of behaviour (AISB)*, UK , pp. 98-106.

Kaneko, H., Hamaguchi, N., Doke, M. and Inoue, S. (2010) Sign language animation using TVML, *In - Proceedings of 9th International Conference on Virtual-Reality Continuum and Its Applications in Industry (VRCAI)*, Seoul, Korea, pp. 289-292.

TVML (2014) TVML Language Specification Version 2.0 [Online]. Available : http://www.nhk.or.jp/strl/tvml/english/onlinemanual/spec/index.html. [Accessed 15 September 14].

BVH (2000) BVH File Specification. [Online]., Available : http://www.character-studio.net/bvh_file_specification.htm. [Accessed 07 August 14].

Sobel, I. and Feldman, G. (1968) A 3x3 isotropic gradient operator for image processing, *presented at the Stanford Artificial Intelligence Project (SAIL)*

Staib, L. H. and Duncan, J. S. (1996) Model-based deformable surface finding for medical images, *In - IEEE Transactions on Medical Imaging*, vol. 15, no. 5, pp. 720-731.