Rendering an Animated Avatar from SignWriting Notation

Kgatlhego Moemedi, James Connan,
Department of Computer Science, University of the Western Cape, Private Bag X 17, Bellville, 7535, South Africa, Telephone: +(27) 959 3010, Email: 2540901@uwc.ac.za, jconnan@uwc.ac.za

Abstract—This paper presents an approach for automatically generating signing animations from SignWriting notation. An avatar (virtual humanoid) with expressive gestures, as subtle as changes in facial expression, is used to render the sign language animations. SignWriting notation in an XML-based format called SWML (SignWriting Markup Language) is provided as input. Relevant features of sign language gestures are extracted from the SWML. These features are then converted to body animation parameters, which are used to animate the avatar. Our avatar renders realistic and aesthetically acceptable movements that can be recognized and understood by deaf people. The avatar can be used to render any sign language since SignWriting can be used to write any sign language.

Index Terms—Avatar, Animation parameters, Deaf, Signing animations.

I. INTRODUCTION

THE South African Sign Language (SASL) Project [1] at the University of the Western Cape is concerned with the translation of English to SASL and vice versa. In an attempt to facilitate access to information and communication services, a machine translation system that allows deaf people to access information in their native language is being developed. For visualization of sign language a machine translation system makes use of a three dimensional (3D) computer generated avatar. Furthermore the use of an avatar for sign language in a machine translation system requires a text or symbolic representation of sign language. In this work, which forms part of the SASL Project, our aim is to render an animated avatar endowed with expressive gestures for sign language communication. Moreover our goal is to generate an avatar that reproduces the exact gestures represented in the sign language notation, SignWriting. In addition we aim to render an animated avatar with realistic and aesthetically acceptable movements that can be recognized and understood by deaf people.

II. RELATED WORK

Currently, there is no international convention for writing sign languages. Writing systems that have been proposed include HamNoSys (Hamburg Notation System), Stokoe and SignWriting.

HamNoSys was developed at the University of Hamburg intended for researchers in sign language to record signs [2]. HamNoSys was not initially developed for computer animation, but the ViSiCAST project uses HamNosys to generate signing gestures [3].

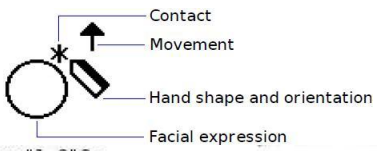
SignWriting is the most widely used system and is being used for the captioning of YouTube videos. VSigns generates animation sequences from SignWriting notation. The authors designed a system, which permits the user to enter words, which are searched in the dictionary. If a particular word is not found, the word is spelt using fingerspelling. The system allows the user to select an H-anim compliant avatar, which is used to present the animations. The dictionary of this system is based on about 3200 words [4]. They integrated a VRML TouchSensor Node with the VRML file describing the avatar. An advantage is that the viewer can zoom in and out and rotate the avatar within the 3-D space to recognize and understand the sign [4]. However this can only be achieved with a VRML compliant web browser.

The main contribution of our work is towards rendering animations automatically. Rather than having a lookup dictionary with a limited number of words, we have a system that can generate any sign language animation from the SWML input. This means that the avatar will be able to perform any sign written in SignWriting. Furthermore an advantage of generating sign automatically is to be able to annotate sign language videos.

III. INRODUCTION TO SIGNWRITING AND SWML

A. *SignWriting notation*

SignWriting was invented by a dancer, Valerie Sutton [5] and is composed of a set of graphical symbols that are used to represent signs. These symbols denote handshapes, body locations, facial expressions and hand and finger movements. The rules for combining symbols are easy so that people with no special training in sign language linguistics can use the notation [4]. SignWriting symbols are used in a pictograph called a signbox.




```
<?xml version="1.0"?>
<!DOCTYPE swml SYSTEM "http://www.signpuddle.com/swml/swml
  <swml dialect="S" version="1.1" lang="sgn" glosslang="">
    <sign lane="0">
      <gloss>sign-1</gloss>
      <symbol x="90" y="68">03-03-002-01-01-01</symbol>
      <symbol x="130" y="51">02-05-001-01-01-01</symbol>
      <symbol x="137" y="73">01-05-012-01-02-02</symbol>
      <symbol x="123" y="64">02-01-001-01-01-01</symbol>
    </sign>
  </swml>
```

Fig. 1. SASL transcription of “Hello” in SignWriting. The SWML format of the sign is shown.

Figure 1 gives an example of how symbols are put together to represent signs. The graphical nature of SignWriting is impractical for parsing by a computer. Thus, SignWriting has been encoded in a mark-up language called SWML (SignWriting Markup Language) by authors in [6]. An example of SWML is shown in Figure 1. SWML is a practical format to parse with a computer and is briefly discussed in the next subsection.

B. SWML

SWML is an XML-based encoding of SignWriting. It is application and computer platform independent. It can be used in language and document processing such as translation, animation, etc [4]. Each symbol in SWML corresponds to some aspect of sign language. These symbols are structured in such a way that they identify the aspect of sign language that the symbol corresponds to. We give a brief overview of the symbols structure in the subsection that follows.

a. SWML Symbol structure

The symbols are divided into 7 categories. These are: hands, movements, facial expressions, trunk and limbs, dynamics, punctuation and location sorting [7]. Each symbol has a unique symbol id (identity), which we use to identify the different aspects of sign language. The 6-tuple symbol id is defined as:

$$s_{id} = (c, g, b, v, f, r)$$

1) c – category
2) g – group
3) b – base
4) v – variation
5) f –fill
6) r – rotation

The *x* and *y* coordinates of a symbol in a signbox are denoted as s_x and s_y respectively. The length and width of a symbol are denoted by s_{length} and s_{width} respectively. Thus the symbol *s* is defined as:

$$s = (s_{id}, s_x, s_y, s_{width}, s_{length})$$

A sign is composed of a set of symbols and is denoted as:

$$Sign = \{s_1, s_2, .. s_n\}$$

We use the symbol id to generate animation parameters, which are used to animate our avatar. In the following section, we explain how we use these fields to generate animation parameters.

IV. Conversion of SWML into Animation Parameters

This section explains how we use the symbol id to generate animation parameters. We implemented a parser, which is divided into two classes. The first class of the parser, *SignSpelling Sequence*, is responsible for spell checking and determining the order of reading symbols in a sign. The second class, *ActionStrip*, is responsible for generating animation parameters. The two classes of the parser are discussed in detail in the next subsections. The parser extracts significant information from the symbol id of each symbol and generates animation parameters.

A. SignSpelling Sequence Class

The *SignSpelling Sequence* firstly checks if all symbols are used correctly in a sign and if the sign is valid. For example, a finger movement symbol requires a hand symbol. If a hand symbol is not included, then the sign is incorrect. Secondly, if the sign is correct, *SignSpelling Sequence* determines the type of a sign. We have defined three types of signs as finger-spelling, basic signs and compound signs. A finger-spelling sign contains hand symbols only, with possibly a hand movement symbol. A basic sign contains only one sign. A compound sign consists of multiple basic signs. Lastly, *SignSpelling Sequence* determines the order of reading the symbols in a sign. This is very important since signs are not written linearly from left to right [8]. In SignWriting, a sign can be read from any direction depending on the symbols used to compose the sign. Thus we use a centering symbol to determine the order of reading the symbols. A centering symbol is used as a reference to determine the symbols on the right, left, top or bottom. Generally the symbols at the stem of the arrow are read first, followed by the symbols at the arrowhead. We show an example of a compound sign in Figure 2, and illustrate the order of reading the symbols. When the symbols have been ordered, they are read from top to bottom.

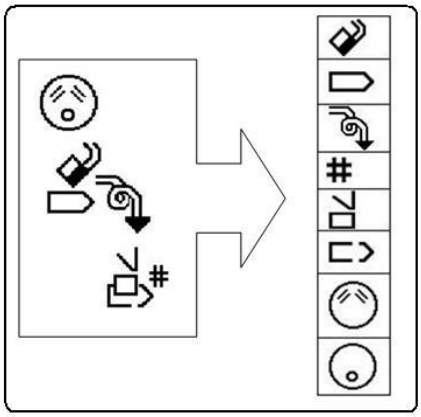


Fig. 2. A graphical representation of the SignSpelling class. The symbols on the right are organized in the order of reading the sign from top to bottom

Following that, the *SignSpelling Sequence* class sends the sorted list of symbols to the *ActionStrip* class where the symbol ids are interpreted and animation parameters are generated.

B. Action Strip Class

The *ActionStrip* class is responsible for generating animation parameters. It receives as input a sorted list of symbols from the *SignSpelling Sequence*. In order to generate animation parameters, the *ActionStrip* class needs

to interpret the input. Thus the *ActionStrip* class contains information about each field of the symbol id. It uses information provided by the fields of the symbol id to specify the BAP (Body Animation Parameters) of the corresponding joints. Currently, our system can interpret symbols from three categories namely; hands, facial expressions and movements. We look at how *ActionStrip* class generates animation parameters for each of the three categories.

a) Hand Category

In SignWriting one basic symbol can undergo numerous transformations such as rotation, orientation or mirroring. All basic symbols are right handed with a rotation angle of 0°, with the hand orientated parallel to the wall with the palm facing towards the signer. Figure 3 shows how one basic symbol can be rotated and mirrored.

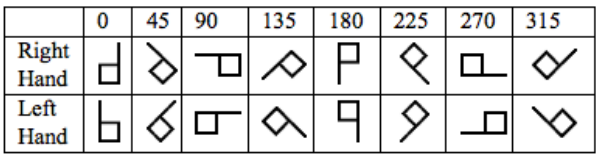


Fig. 3. The angles of rotation of the left and right hand. The second row indicates the mirroring of the first row.

Since the basic symbol is right handed, the mirrored symbol refers to the left hand. The *rotation* (*r*) field specifies whether the symbol is mirrored. The right hand is indicated by $1 \leq r \leq 8$. The left hand is indicated by $9 \leq r \leq 16$. Furthermore the *rotation* field indicates the rotation of the hand, which occurs at a multiple angle of 45°. The left hand is rotated clockwise, where $r=9$ represents no rotation. The right hand is rotated anti-clockwise where $r=1$ indicates no rotation as indicated in Figure 3.

Once the left or the right hand has been identified, we determine the animation parameter of the finger joints to produce the corresponding handshape. We use a unique combination of "*c-g-b-v*" of the symbol id to identify the basic symbol. The "*c-g-b-v*" combination of the symbol id specifies animation parameters of the finger joints of a basic symbol. We show the skeleton of the avatar's hand in Figure 4. The hands are very complex articulated structures with multiple degrees of freedom (DOFs) [9].

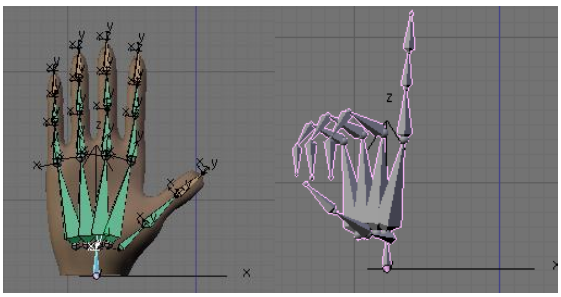


Fig. 4. The skeleton of the hand of our avatar.

Thus we have created a predefined lookup table, which defines animation parameters of the finger joints. The authors in [9] used forward kinematics and inverse kinematics which allow bones to be rotated to create poses. They also applied rotational constraints to the bones to achieve poses that are possible to humans.

After determining the animation parameters of the finger joints we determine the hand orientation. Hand orientation can be divided into two parts. The first part of the orientation refers to the direction in which the palm is facing. This orientation is viewed from the signer's point of view. In Figure 5 we show how the hands are orientated in SignWriting. The hand orientation is identified by the *fill* field in the symbol id. If $f = 1$ or $f = 4$ the palm is facing towards the signer. If $f = 2$ or $f = 5$ the signer sees the side of his hand. If $f = 3$ or $f = 6$ the palm is facing away from the signer. The second part of the hand orientation specifies the plane which the hand is parallel to. The hand can be parallel to the wall or to the floor plane and is identified by the *fill* field, $1 \leq f \leq 3$ and $3 \leq f \leq 6$ respectively. In Figure 5, we show the different hand orientations along the wall and floor plane.

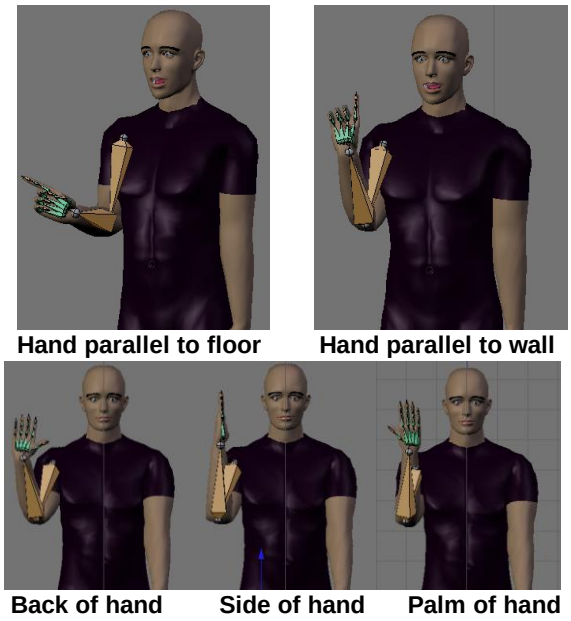


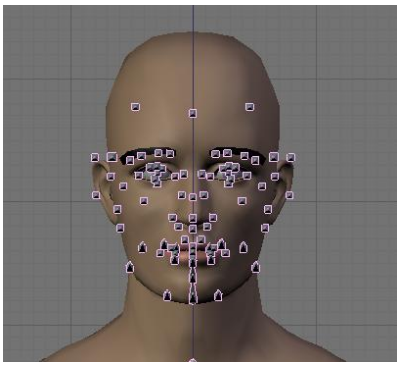
Fig. 5. The different orientations of the hand along the floor and wall plane.

We use the *fill* and the *rotation* fields to determine the animation parameters of the elbow and the shoulder to produce the corresponding hand orientation and rotation. All the hand symbols are interpreted using this approach. After processing the hand symbols all the body animation parameters are stored.

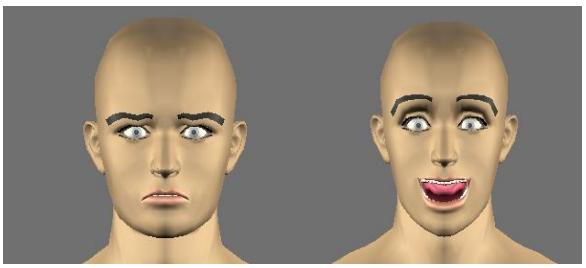
b) Facial expression Category

Facial expressions in sign language are very important because they express the grammar, and convey feeling and interest [5]. Facial expressions include among others raising eyebrows, wrinkled forehead and eyes blinking. Facial expressions used for yes/no questions are different from facial expressions used for wh-questions e.g. who,

where, etc. [10]. Yes-no questions require raised brows. Wh- questions require furrowed brows. A change in facial expression could express a completely different message. In SignWriting the facial expression symbols encode information about which part of the face must be moved and how it should move. An approach that we used to determine the animation parameters of the finger joints is also used to determine the FAPs (facial animation parameters). We use the unique combination of "*c-g-b-v*" of the symbol id to identify the part of the face that should move. This unique combination also specifies if for example, the cheeks are puffed or sucked in, or eyes are half opened or wide opened. The *rotation* field indicates the direction in which the facial part must move. For the facial expressions that include eyes, cheeks, eyebrows and ears, we use the *fill* field to determine if the left, right or both facial parts are used. If $f = 1$ the facial expression includes the left and right side of the facial part. If $f = 2$ the facial expression includes the right side of the facial part. If $f = 3$ the facial expression include the left side of facial part.



The skeleton of the face of the avatar



Eyebrows frowned **Eyes wide opened**

Fig. 6. The subtle changes in facial expression. Moving the bones on the points of interest such as the lips create a frown. The bones on the eyebrows creates furrowed or raised eyebrows.

Our model contains additional bones added on facial feature points such as eyes, eyebrows, etc [11] as indicated in Figure 6. These bones were given no limitation in order to make it easier to perform facial expressions. When we have identified the facial parts and their movements we generate the FAPs. This we do by using a predefined lookup table, which defines the FAPs of the corresponding facial parts. We show some of the facial expressions that the avatar produced in Figure 6. The subtle changes in facial expressed can be noticed. For example the eyebrows in Figure 6 are frowned. When all the facial processing is done the FAPs are generated and stored.

c) Movement Category

SignWriting uses arrows to indicate path movement. There are different arrows to indicate movements associated with the left hand, the right hand and both hands moving together simultaneously. The different arrows are shown in Figure 7. We use the *fill* field to identify the hand to which the movement is associated. If $f = 1$ then the movement is associated with the right hand. If $f = 2$ then the movement is associated with the left hand. If $f = 3$ then the movement is associated with the left and right hand.

	Left	Right	Both Hands	Curves	Circles
Wall					
Floor					

Fig. 7. The angles of rotation of the left and right hand. The second row indicates the mirroring of the first row.

Movement can be along the wall or the floor plane. The unique combination of "*c-g*" of the symbol id indicates whether the movement is along the floor or wall plane. Additionally the unique combination of "*c-g-b-v*" of the symbol id indicates the type of movement. It indicates whether the movement is a straight wall plane, a curve, or a circle (see Figure 7). To distinguish between shorter and longer movements, we use the *variation* field, which indicates the length of the movement. We determine the direction of the movement using the *rotation* field. Similar to rotation of the hands, the movement rotation occurs at a multiple angles of 45°.

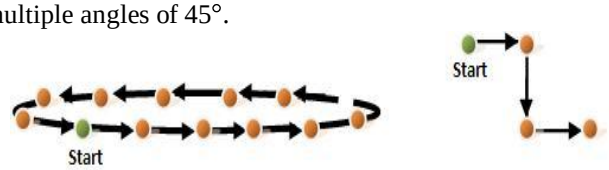


Fig.6. Using more keyframes to produce circular and other complex movements. The orange dots indicate the keyframes. The green dots indicate where the movements start.

To produce the type of movement, in the specified direction, we use the keyframe technique. The keyframe technique is used in animations to define the starting points and ending points of a smooth transition [12]. Interpolation is then used to produce "in between" frames to obtain a smooth transition. Different types of movement will require varying numbers of keyframes. A straight wall plane movement will require two keyframes, the starting point and the end point. However a circular movement will require more keyframes to produce a circular motion as shown in Figure 6. A keyframe is inserted at a certain position in time, where the position defines the timing of the movement. Let the starting point be defined as point $\mathbf{x}(t_0)$ at time t_0. The position $\mathbf{x}(t_i)$ at time Δt_i is defined as:

$$x(t_i) = x(t_{i-1}) + s(\Delta t_i)$$

where Δt_i=25 and s is the speed. In SignWriting the

dynamic symbols indicate if the movement is slower or faster than normal. We defined speed as $s = 1$, $s = 2$, $s = 3$ for very fast, neutral and very slow movements respectively. Thus if $\mathbf{x}(t_0) = 0$, then the next keyframe for a fast ($s = 1$) speed will be at $\mathbf{x}(t_1) = 25$. For a normal speed $\mathbf{x}(t_1) = 50$ and for a slower movement $\mathbf{x}(t_1) = 50$. This means it will take 25, 50 and 75 seconds for fast, normal and slow movement respectively. When all the movement symbols have been processed the keyframes are generated and stored.

V. Animation

All the hand, facial and movement symbols that were provided as input in SWML format are now converted to animation parameters. To produce an animation, animation parameters are selected following the order of reading the symbols in a sign, as was determined by the *SignSpelling Sequence* class of the parser. Animation is now represented as a sequence of keyframe poses, facial expressions and handshapes. Animation is then achieved by blending these poses together. This is done by using quaternion interpolation to acquire a smooth transition between frames. The avatar is then displayed on the screen performing the signing animation corresponding to the input.

VI. Experiments and Discussion

The goal of the experiment was to measure the accuracy and recognition of the signing animations produced by our avatar model. Recognition was measured by recognition (recognition = 1) or non-recognition (recognition = 0) of signing animations. Recognition was tested by asking the participants to identify the sign that was performed by the avatar as shown in Figure 7(a).

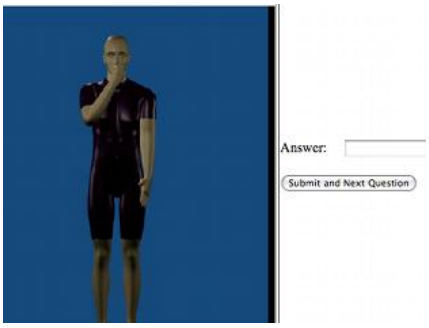


Fig.7 (a). The test subjects identify the sign performed by the avatar. In this test recognition rate is measured.

Fig.7 (b). The test subjects compare the sign performed by the avatar to the sign written in SignWriting. In this test accuracy rate is measured.

Accuracy was measured by identifying a match (accuracy = 1) or non-match (accuracy = 0). Accuracy was tested by showing the participants the sign written in SignWriting and the avatar performing the same sign as shown in Figure 7 (b). The participants were then asked to state if the avatar is signing what is represented in SignWriting notation. Furthermore the participants were asked to give comments indicating such as what made the sign unrecognizable.

In this experiment we used the SignWriting signs from the SignPuddle Database [13]. For effective results we asked the founder of SignWriting and other SignWriting experts to evaluate our system. All animations were rendered using Blender [14] software. The final signing animations were output in *avi* video format. 11 subjects with SignWriting skills from the SignWriting mailing list participated in the test. Subjects were invited to participate in the study and were given a brief summary of the test description. The test was run online. The test consisted of 8 signs.

A. Test results and discussion

The signing animations performed by the avatar need to be as natural as possible in order to be recognized and understood by deaf people. The experimental findings indicated a recognition rate of 82%. For further analysis we evaluated the comments provided by the test subjects about the animations performed by the avatar. The comments revealed some discrepancies that made the signing animations difficult to identify. Some of the discrepancies were "*The left hand seems to be too angled downward"* and "The *movement is too large*". Furthermore the additional comments indicated some variations in dialect, which contributed to the signing animation being unrecognized. The test results indicate that the signing animations can be understood which implies that the avatar performs realistic and recognizable animations.

TABLE I
THE ACCURACY RATE OF THE SIGNING ANIMATIONS.

Signs	Recognition in %	Accuracy in %
Thank you	91	91
Hello	91	91
Now	72	36
Same	45	18
Food	91	91
House	81	81
Understand?	91	0
Time	91	91
Average	82	62

With respect to the avatar performing the exact gestures represented in SignWriting notation, the results indicated that the signing animations were performed appropriately with an accuracy of 62%. An in-depth analysis indicated that the signing animations were performed accurately with a rate of 41% and 21% as "Excellent" and "Good" respectively. Some of the comments that were added by the test subjects were: "Hands

should stop when elbow is at a 90 degree angle.”, “*The avatar is signing the SW as it is written.*”, “*It’s pretty good but id like to see a larger arcing movement.”*, “*This is correct. Both arms have to move to show time*”. When the avatar performs a signing animation it starts and ends at the default position, which may look somewhat unnatural. There appears to be little correlation between the recognition rate of signs and the accuracy with which they were judged to have been performed. Table 1 shows that the word ‘*understand?’* had recognition of 91% with an accuracy of 0%. The sign ‘*Thank you*’ had recognition of 91% and accuracy of 91%. This shows that even though a sign may have been rendered inaccurately, it can still be recognizable. Some technicalities affecting the accuracy of the system are explained below.

1) Insufficient accuracy of interpreting the input. If the fingertip points to an eyebrow, the sign means eyebrows and not eyes. In the test case we had a sign for ‘*same*’. This sign requires two parallel index fingers to lie on the floor plane and make contact with each other twice. However we produced two index fingers slightly crossing each other making contact. The direction of the fingertips of the index fingers was confusing. This problem can be solved by improving the input script, to for example, indicate which parts of the hands should make contact and to determine the direction of the fingers.

2) The different finger lengths of the different handshapes. The flat handshape is longer than the fist handshape. A sign of a fist handshape that touches the mouth requires the hand to move further up than the flat handshape. Since the lengths of these two handshapes vary, the movements of the hands need further consideration. Collision detection can be used to solve this problem.

3) Rotational limits of the joints. The joint restrictions enable the avatar to perform all rotations that are possible to humans. A flat handshape of a right hand rotated at an angle of 270° (see Figure 3) is possible, however the rotation is unnatural and may produce an unnatural animation, which may be difficult to recognize.

VII. Conclusion

We have presented the implementation of our system that renders an animated avatar from SignWriting notation. Our implementation allows an automatic rendering of signing animations, which can be recognized and understood by deaf people. We obtained an accuracy of 66% and recognition of 82%. We have identified the factors that affect the accuracy. The automatic rendering of signing animations can render any sign that can be written in SignWriting in contrast to systems that can only perform a limited number of signs.

VIII. Future work

We are currently working on extending and improving our system. We are improving the accuracy of producing the exact signing animations. Currently our system uses SWML as input. We wish to adapt our system to be able to also use as input the recent XML encoding of SignWriting called (BSWML) Binary SignWriting Markup Language, which currently is not supported by our system. A possible extension of the system would be to animate any H-Anim compliant avatar.

References

[1] SASL Project. Integration of Signed and Verbal Communication: South African Sign Language recognition and animation, http://www.cs.uwc.ac.za/.

[2] S. Prillwitz, R. Leven, H. Zienert, T. Hanke, J. Henning, HamNoSys Version 2.0: Hamburg Notation System for Sign Language – An introductory Guide. International Studies of Sign Language and the Communication of the Deaf, Volume 5.

[3] R Kennaway, Experience with and requirements for a gesture description language for synthetic animation. Lecture Notes in Computer Science Vol. 2915, pp. 300-311

[4] M. Papadogiorgaki, N. Grammalidis, L. Makris, N.Sarris, and M. G. Strintzis, VSigns – A Virtual Sign Synthesis Web Tool, Proc. 6th COST 276, Workshop on Information and Knowledge Management for Integrated Media Communication, pp. 25-31.

[5] V. Sutton (2000, January). Sutton Movement Writing & Shorthand: A global writing system for a global age. [Online]. Available: http://www.valeriesutton.org/

[6] SWML-Overview, [Online]. Available http://web.archive.org/web/20040407141949/swml.ucpel.tche.br/overview.htm

[7] S. E. Slevinski, SignWriting Symbol Lessons [Online] Available: http://www.signbank.org/swis/iswa/

[8] S. E. Slevinski, SignWriting Image Server, Open Font and Rendering software. [Online]. Available: http://www.signbank.org/swis/iswa/

[9] D. van Wyk, J Connan, High Quality Flexible H-Anim Hands for Sign Language Visualization.

[10] G. Caridakis, K. Karpouzis G. Sapountzaki, S. Fotinea, E. Efthimiou, A dynamic environment for Greek Sign Language Synthesis using virtual characters.

[11] D. E. van Wyk, Virtual Human Modelling and Animation for Sign Language Visualisation, MSc thesis, University of the Western Cape

[12] N. Burtnyk and M. Wein, Interactive skeleton techniques for enhancing motion dynamics in keyframe animation. Communications of the ACM, 19, 564-584.

[13] SignPuddle, [Online]. Available: http://www.signbank.org/SignPuddle1.5/searchword.pthp?ui=1&sgn=5

[14] Blender. [Online]. Available: http://www.blender.org

Kgatlhego A Moemedi is a Telkom Centre of Excellence M.Sc student at the University of The Western Cape. Currently she is doing research on sign language visualization and SignWriting. Her interests include artificial intelligence, sign language and 3D animations.

James Connan heads up the South African Sign Language research group at the University of The Western Cape. He has a wide range of interests that include: computer vision and machine learning.