

Auslan Jam: A Graphical Sign Language Display System

Nick Lowe, James Strauss, Sam Yeates, Eun-Jung Holden
Department of Computer Science & Software Engineering
The University of Western Australia
35 Stirling Highway, Crawley, W.A. 6009, Australia.
{nickl, james, samy, eunjung}@cs.uwa.edu.au

Abstract

Australian sign language, Auslan, uses a combination of hand shapes, positions and movements, as well as facial expressions to communicate. Our ongoing research is to develop an automatic translator that graphically translates English to Auslan. Currently, we have implemented object-oriented C++ graphics libraries to build a whole upper body model. The model is a kinematic tree that allows physiologically possible movements that are necessary for Auslan sign display. We have experimented with two angle representations: Euler and quaternion. Using interpolation algorithms, we determined that the quaternion angle representation is more reliable than the Euler angle representation. This paper presents the design and implementation of the Auslan Jam system and our research into angle representation techniques. It also discusses future development of the model and the sign translator interface, and identifies possible improvements.

1 Introduction

Australian deaf communities use a sign language called Auslan. Signers use a combination of hand movements which change in shape and location relative to the upper body, as well as facial expressions. Auslan is different from other sign languages, though it is related to the British sign language. Auslan has different grammar from English, which makes it difficult for the deaf to master English. Thus, communication between the deaf and the hearing often relies on human sign translators who know both sign language and spoken language.

Automatic translation between Auslan and English is an active area of research within our department. Research includes a number of Artificial Intelligence (AI) topics, such as visual gesture recognition to translate Auslan to English [5]; audio and visual speech recognition to translate English to Auslan [4] [3]; and a language mapping system between

Auslan and English. Developing practical applications involves devising output facilities such as text or audio English output for hearing people, and graphical sign display for the deaf.

One aspect of our ongoing development is to build an automatic sign translator that graphically translates English to Auslan. It uses English text input which is then translated into Auslan by displaying signs through computer graphics. It will not perform all aspects of translation, such as the semantic interpretation of the signs. However, given that the system establishes a sufficient database of signs, some mapping between signs and letters/words/sentences could provide an adequate translation for certain interactions (for example, conferences) and informal ones (for example, restaurants), as well as a tool for learning signs for non-signers. It could also be useful in emergency situations such as in hospitals or police stations, where urgent information could be conveyed without having to wait for a human translator, and where written communication is inappropriate.

There are two common methods used to implement the display and animation of signs in translator systems [2]. The first stores sign representations as frames of a digital movie, while the other uses stored parametric data allowing run-time generation of sign movement through interpolation and display of computer graphics primitives.

Unlike the first method, the second method can generate a smooth interpolation from one sign to another, providing more realistic output. Our translator uses the parametric computer-generated sign translation technique. Previously, we developed a graphical sign translator system that animates two-handed movement to display signs on a Sun workstation [6]. The current system uses whole upper body movement and is developed to run on a domestic PC. This system is developed in three stages. The first stage is to build a kinematic model of the whole upper body, where manipulation of the body joints is possible. The second stage is to devise an effective technique to manipulate the model for animation. The final stage is to build an interface

where English text input is entered and the model is used to animate the translated signs.

Currently, we have designed and implemented a graphical kinematic upper body model called *Auslan Jam*.

The kinematic model of the upper human body was developed to display the signs. This was implemented using an object-oriented technique where body joints are considered *node* objects, each containing local orientation parameters. These joints are connected to form a kinematic tree, where each joint location can be determined from the ancestor joint orientations and known body segment lengths. For example, the elbow location is calculated by the orientations of the body and shoulder, and the lengths from body to shoulder and from shoulder to elbow. The body is rendered by displaying a polygonal model for each segment.

We experimented with various interpolation methods using Euler and quaternion angle representations in order to manipulate Auslan Jam.

This paper explains the model implementation, reports the results of interpolation experiment, and finally describes the ongoing development of a model manipulation technique and the sign translator interface.

2 Graphical Human Model Design

2.1 Requirements

The biomechanics of the human upper body are very complex, however, the representation of Auslan requires only a coarse approximation. A non-uniform simplification of the biomechanical model allows complexity to be retained in areas where it contributes to the visual appearance of the model. For example, many Auslan poses involve intricate finger positioning. Thus, the model requires many nodes to provide full degrees-of-freedom (DOFs) allowing all physiologically possible movements of the hand. In contrast, Auslan does not require significant spinal movement. Thus the DOFs of the spine can be minimised and fewer nodes are required to simulate spinal movement.

2.2 Design

The biomechanical structure of the upper body can be well simulated by a kinematic tree. A suitable kinematic tree consists of a series of nodes, each with a parent and an arbitrary number of children. A node represents a joint in the body, containing a transformation from the parent node's local axes to its own. The translation component remains unchanged to represent rigid body segments connecting each joint. The rotation component may be modified to alter the orientation of the body segment. The local axes of any node can be calculated via forward kinematics as shown in Figure 1.

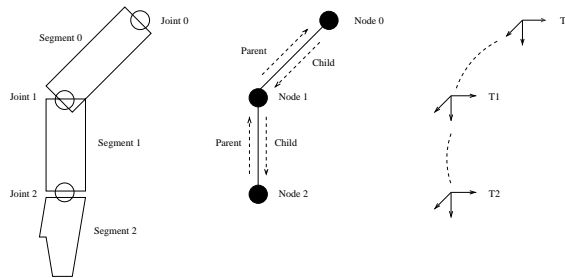


Figure 1. A kinematic model of the human arm. The human arm can be represented as a series of joints and body segments. This can be kinematically modelled by associating nodes in a kinematic tree with joints in the arm. Each node contains its own coordinate frame in terms of its parent's coordinate frame. Therefore the local axis of any node can be calculated using the coordinate frames of its ancestors. This is known as forward kinematics. (In this example, $T_2 = T_1.T_0$)

In order to provide a realistic appearance of the human upper body, each node has an associated polygonal model representing the surface of the body segment directly following the joint. Thus, a node can be rendered by transforming the associated polygonal model to the node's local coordinate space and rendering it. All segments following a node in the kinematic tree can be rendered by performing a depth-first traversal of the tree from that node, and rendering each node encountered as shown in Figure 2. The entire tree can be rendered by traversing from the root node.

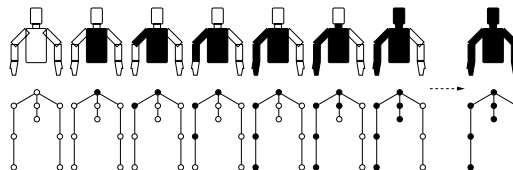


Figure 2. Rendering the human upper body model. Auslan Jam renders the human upper body model by depth-first traversal of the kinematic tree. When a node is encountered, its associated polygonal model is displayed of the screen. Once the traversal is complete, the entire model has been rendered.

We adopted an object-oriented approach for the kinematic modelling and rendering system. Each node in the kinematic tree is an instance of a node class. The node class

defines pointers to a parent and child nodes to describe the hierarchy of the kinematic tree. Each node also contains a pointer to a polygonal model object that represents a body segment. For example, the elbow node contains a pointer to its parent node (the shoulder node), its child node (the wrist node) and a polygonal model representing the forearm. We have included a display method in the polygonal model class to simplify rendering of the entire tree.

To traverse and render a tree we created a node render method that utilises the node's pointers to its children. The node render method calculates its local coordinate space, transforms its polygonal shape into this space, and then calls the shape display method. Then, it calls the render methods of all its children. This recursive behaviour will render any forward kinematic tree by depth-first traversal when called from the root node. Thus it can render any human upper body pose represented in a kinematic tree.

Our object-oriented approach facilitates modularity and extensibility. The system is modular because functionality of the system depends upon interfaces between classes. Thus, technologically different classes are interchangeable because the implementation of the classes is independent of the interfaces. For example, the basic polygonal model class can be replaced with a model class that provides a different render method or geometrical data format. This is an example of the extensibility supported by object-oriented design.

All data are maintained independent of the kinematics and graphics system. In this way, we allow for future improvement of data without the need for changes to the system. For example, the kinematic resolution of the human upper body model can be modified without altering the workings of the system. Similarly, the geometric resolution of body segments can also be improved.

2.3 Implementation

We developed the kinematic tree and rendering module in C++ because it meets our object-oriented design requirements and can easily be linked to other components of the two-way Auslan-English translator system. C++ has an advantage over languages such as VRML, a popular display method [11] [8], which is difficult to link with other systems.

We first created a set of fundamental geometric classes, and then built classes for nodes, models, trees, and poses. OpenGL was used for rendering because it provides a fast, standard set of basic rendering functions.

The basic model of the human upper body was created by using manual measurement of a human participant. It contains 39 nodes, each with an associated body segment. There are three nodes to represent the spine. These nodes represent the base of the spine, the base of the neck, and the base of the head. Figure 3 shows this.

All joints in the arms (shoulder, elbow, and wrist) are

represented by nodes, and there are 15 nodes per hand. The fingers are modelled using nodes representing the metacarpophalangeal (MP), the proximal interphalangeal (PIP), and distal interphalangeal (DIP) joints. The thumb is similarly modelled with nodes representing the carpometacarpal (CM), MP, and interphalangeal (IP) joints. Thus, the kinematics of our virtual hand closely approximate a real hand.

Each node allows full three DOF orientation. Physiological constraints of movement can be imposed by the animation system rather than the underlying graphics system.

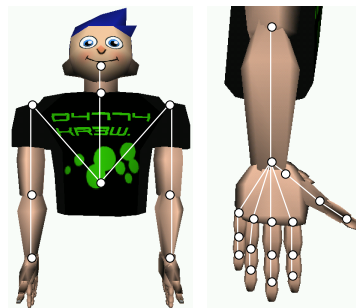


Figure 3. The initial kinematic tree representing the human upper body. This tree is parametrised using measurements from a human participant. The white circles and lines represent nodes and arcs of the tree respectively.

3. Joint Orientation Representation

Realistic sign language animation requires accurate configuration of the upper body into poses as well as smooth animation between poses. In order to achieve this, it is important to determine an efficient means for representing joint orientations. We compared two orientation representation methods: Euler and quaternion. The effectiveness of these representations are investigated by combining them with various interpolation techniques.

Euler representation describes the orientation of an object using three rotations around three axes of rotation. Although a simple and intuitive method for describing object orientation, the Euler representation method has some problems. Euler angles suffer from *gimbal lock*, which occurs when one axis of rotation lines up with another axis of rotation. Consequently, a degree of freedom is lost from the rotation when this is the case. When directly specifying the Euler orientation parameters this is not a problem, however some calculations dealing with Euler angles become ill-defined near gimbal lock positions. Furthermore, in our sign display system we graphically manipulate the model

to define each pose, and the same pose may be generated by using a different combination of Euler angles. This can cause problems when animating between poses, possibly requiring remapping of angles before animation.

Quaternions use a scalar, w , and the three coefficients of imaginary terms, x , y and z to represent orientations. That is,

$$\mathbf{q} = w + xi + yj + zk,$$

$$\text{where } i^2 = j^2 = k^2 = -1, x, y, z \in \mathbb{R}.$$

While quaternions are a four dimensional structure, rotations are three dimensional. An extra condition requiring quaternions to be of unit length is required to limit the usable quaternion space to three dimensions. This means that quaternion rotations lie on a unit sphere in 4 dimensional space. For a full discussion of quaternions and comparison of angle representations, see Dam, Koch and Lillholm [1].

Unlike simpler Euler angles, examining the parameters of a quaternion does not provide an intuitive method of understanding the resulting orientation. However, the benefits of using quaternions outweigh this, and include the ability to combine quaternion rotations simply by multiplying quaternions, and freedom from the multiple definition problem (where there can be more than one set of euler angles representing a rotation) and the gimbal lock problem of Euler orientation representations [10].

4. Joint Interpolation

4.1 Interpolation Method

Simple interpolation algorithms were chosen to compare the Euler and quaternion orientation techniques.

Euler representation was tested using simple linear interpolation of the three angles between an initial and a final pose.

Two interpolation methods were tested for quaternions:

- The linear interpolation (lerp) formula that linearly interpolates between the four quaternion parameters. Since the initial and final orientations lie on the surface of a unit sphere, this interpolation “cuts through” the inside of the sphere. After interpolation, the quaternion must be renormalised to ensure it remains a legal quaternion rotation.
- The spherical linear interpolation method (slerp) is slightly more complex than the lerp method. Instead of interpolating linearly between the two positions on the unit sphere, it directly calculates the position on the surface of the sphere. This results in the same physical path being traversed as the lerp method, however

the angular velocity of animation between the two approaches is different. The formula to interpolate along the great circle between the initial orientation \mathbf{q}_i and final orientation \mathbf{q}_f is given below. Near the final orientation, when $\sin(\theta) \approx 0$, linear interpolation is used.

$$\mathbf{q}(t) = \frac{\mathbf{q}_i * \sin((1-t)\theta) + \mathbf{q}_f * \sin(t\theta)}{\sin(\theta)},$$

$$\text{where } \theta = \arccos(\mathbf{q}_i \cdot \mathbf{q}_f) \text{ and } 0 \leq t \leq 1.$$

4.2 Practical Comparison

Our experiments were similar to that of Dam et al. [1] but the interpolation process was visualised by animating the right shoulder joint of the Auslan Jam model. A simple user interface was created to graphically manipulate the model, and was used to generate two poses for each interpolation test. An animation between these poses was displayed, and the angular distance to the final pose and angular velocity of the animation were recorded for 200 frame interpolations.

Figures 4, 5 and 6 illustrate two important advantages of quaternion interpolation methods over Euler interpolation methods. The initial orientation of the shoulder for each animation is $(0^\circ, 180^\circ, 0^\circ)$, while the final position is an euler orientation of $(90^\circ, 270^\circ, 90^\circ)$. This selection of orientations illustrates that Euler interpolation does not always take the most direct path from initial to final points. An Euler interpolation from $(0^\circ, 180^\circ, 0^\circ)$ to $(0^\circ, 270^\circ, 0^\circ)$ would produce a different animation path, even though $(90^\circ, 270^\circ, 90^\circ)$ and $(0^\circ, 270^\circ, 0^\circ)$ are equivalent orientations. The quaternion interpolation methods produce the same animation path with either set of endpoints. The angular velocity of an interpolation is an important consideration. The angular velocity graph for the Euler interpolation shows that it has a non-constant velocity. While the quaternion lerp algorithm also produces a non-constant velocity, it will not suffer from the corresponding problems of multiple endpoint definition of Euler interpolation. The quaternion slerp algorithm is slightly more complex than the lerp algorithm, but exhibits a constant angular velocity for all interpolations.

Thus, quaternion angle representation provides mathematical simplicity in storing orientations, and freedom from less desirable aspects of the Euler angle representation method. In our experiment, quaternion interpolation methods produced satisfactory interpolation between two key frames, which will be extended to interpolate multi-keyframes [1] [9].

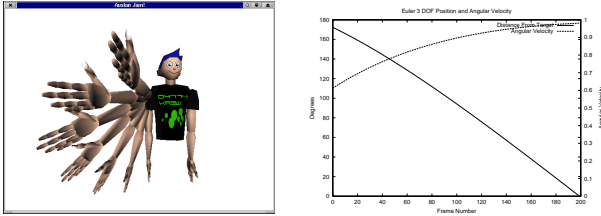


Figure 4. A screen capture and velocity analysis of a 3 degree of freedom Euler interpolation. The arm moves from the default position (downwards) to a position towards the front of the model’s body, with a twist of the arm.

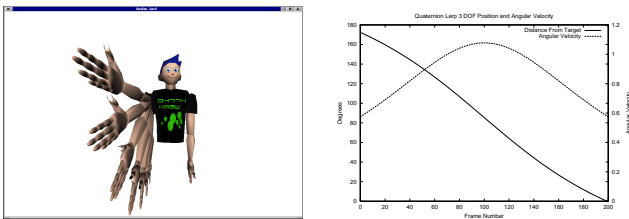


Figure 5. A screen capture and velocity analysis of a 3 degree of freedom lerp interpolation. The arm moves from the default position (downwards) to a position towards the front of the model’s body, with a twist of the arm.

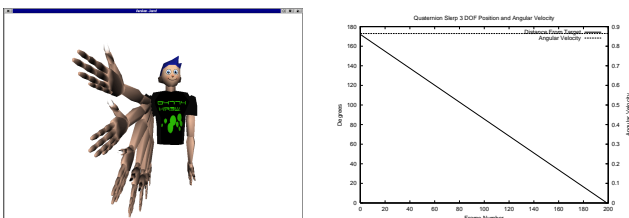


Figure 6. A screen capture and velocity analysis of a 3 degree of freedom slerp interpolation. The arm moves from the default position (downwards) to a position towards the front of the model’s body, with a twist of the arm.

5 On-going development

We presented Auslan Jam, a system that uses an object-oriented kinematic model of the human upper body to graphically display Auslan. Auslan Jam’s graphics libraries provide facilities to easily extend the system to any biome-

chanical simulation. We use quaternion representation for natural interpolation from one pose to another. Ongoing research includes improvement of the model, device of model manipulation techniques to define poses, as well as translator interface design.

5.1 Model improvements

The Auslan Jam model currently uses a fast and simple method to render the model figure, since it is designed to run on a domestic PC without a fast graphics card. However, the rendering method can be improved to provide a more realistic model figure.

The model can be extended to provide visual speech simulation while signing. English speech of the corresponding signs can be simulated in the face of the model, providing an additional modality as often used in deaf communication.

5.2 Model manipulation technique

Auslan signs consists of key poses and their in-between interpolations. There are three ways of producing the key poses that consist of 3D orientation parameters of the model kinematic tree. The first method uses a motion capture device such as a Virtual Reality (VR) ware, or a vision based system where the joint angles are generated automatically. The second method graphically manipulates the joint angles of the model to store the pose. The third method graphically specifies the path of the model actuators, and allows inverse kinematics to automatically calculate the poses. While we currently use the second method, the third method needs to be investigated to specify hand locations.

Smooth multiple keyframe interpolation is important for generation of realistic output. The chosen algorithm must successfully animate between keyframes within a sign (predictable), and between arbitrary signs (less predictable). A large body of work has been carried out in regard to interpolating positions of individual objects, which must be adapted to efficiently interpolate orientations for the articulated Auslan Jam model.

5.3 Design of the translator interface

The proposed structure of the translator interface is shown in Figure 7.

The translator interface requires a database where kinematic configurations of sign poses and motion are stored. It requires two databases: a hand shape database and a sign database. The hand shape database stores Auslan basic hand shapes [7], which separates Auslan signs like the alphabet in English, with their corresponding configurations. The sign database stores Auslan signs by using basic hand shapes as well as other upper body joint configurations. In

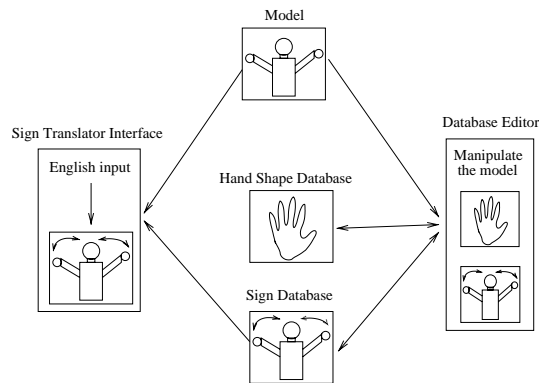


Figure 7. Structure of the translator system.

order to generate and modify these databases, we will develop a database editor that enables the kinematic model to be graphically manipulated to allow storage of its configuration in the database. The translator interface will extract signs from the database and display a sequence of signs by animating the in-between motion.

Acknowledgement We would like to thank Robyn Owens and Amitava Datta for their input. This work is supported by the Australian Research Council.

References

- [1] E. B. Dam and M. M. L. Koch. Quaternions, interpolation and animation. Technical report, Department of Computer Science, University of Copenhagen, 1998.
- [2] N. Frishberg, S. Corazza, L. Day, S. Wilcox, and R. Schulmeister. Sign language interfaces. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 194 – 197, 1993.
- [3] E. J. Holden, G. Loy, and R. Owens. Accommodating for 3d head movement in visual lipreading. In *Proceedings of IASTED International conference on Signal and Image Processing*, pages 166–171, 2000.
- [4] E. J. Holden and R. Owens. Visual speech recognition using cepstral images. In *Proceedings of IASTED International conference on Signal and Image Processing*, pages 331–336, 2000.
- [5] E. J. Holden and R. Owens. Visual sign language recognition. In R. Klette, T. Huang, and G. Gimenez, editors, *Multi-Image Search and Analysis*. Springer, Berlin, 2001.
- [6] E. J. Holden and G. G. Roy. The graphical translation of english text into signed english in the hand sign translator system. *Computer Graphics Forum*, 11(3):C357–C366, 1992.
- [7] T. A. Johnston. *Auslan Dictionary: A dictionary of the sign language of the Australian deaf community*. Deafness Resources, Australia, 1989.
- [8] T. Naka, Y. Mochizuki, and S. Asahara. Wonderspace: web based humanoid animation. In *Elsevier Future Generation Computer Systems 17*, pages 57 – 64, 2000.
- [9] R. Ramamoorthi and A. H. Barr. Fast construction of accurate quaternion splines. *Computer Graphics*, 31(Annual Conference Series):287–292, 1997.
- [10] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH’85*, pages 245–254, 1985.
- [11] A. Su and R. K. Furuta. Vml-based representations of asl fingerspelling on the world wide web. In *Proceedings of the Third International ACM Conference on Assistive Technologies*, pages 43 – 45, 1998.