Two Dimensional Sign Language Agent

Matthew McConnell
School of Computing Science, University of Glasgow
Matthew.McConnell@glasgow.ac.uk

Mary Ellen Foster
School of Computing Science, University of Glasgow
MaryEllen.Foster@glasgow.ac.uk

CCS CONCEPTS

• **Computing methodologies** → **Procedural animation**; • **Human-centered computing** → *Accessibility technologies*; • **Social and professional topics** → People with disabilities.

KEYWORDS

2D, Sign Language, Virtual Agent, Comprehensibility

ACM Reference Format:
Matthew McConnell and Mary Ellen Foster. 2020. Two Dimensional Sign Language Agent. In *IVA '20: Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents (IVA '20), October 19–23, 2020, Virtual Event, Scotland Uk.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3383652.3423898

1 INTRODUCTION

Throughout the world, deaf people communicate through sign language; each country has a distinct sign language, and there is no universal sign language. Some people with hearing disabilities use solutions such as hearing aids or interpreters to help them communicate with hearing people; however, these solutions can't be used by all people and sometimes can only be used sparingly. A sign language conversational agent would have many applications for the deaf community, particularly as an interpreter. In this paper we focus on creating a sign language animation synthesis pipeline; such a pipeline could have potential use within a full, end-to-end conversational agent.

There is some previous work in automatically generating sign language animations. An early example is TESSA [2], which translated from spoken language to British Sign Language in the post-office domain. More recent approaches have adopted sequence-to-sequence approaches to translate a sequence of text into the corresponding signs [3, 5]. However, these examples of automatic sign language generation all suffer from the use of technologies that are not highly available (motion capture, depth cameras, and large computational power respectively) leading to them being less general, portable, scalable, and ultimately usable.

In this work, we aim to create an animated agent that can produce sign language animations without the need for costly human intervention or costly computational resources. Our work only attempts this on the British Sign Language (BSL) alphabet[1] which contains the same 26 letters of the English alphabet. However, our work can be easily extended and applied to a full dataset of signs (including words) and other sign languages. The key attributes we were aiming for within our sign language synthesis pipeline were:

- **lightweight** - requires few computational resources
- **versatile** - able to sign everything
- **general** - applicable to any sign language by learning from examples
- **comprehensible** - understood by sign language speakers

2 IMPLEMENTATION

2.1 Building the Sign Language Dictionary

The first stage of the animation pipeline was to build a sign dictionary from video examples. Video examples of sign language are the most abundantly available within the sign language community. Thus, it is the best dataset to work from to allow our pipeline to be *general*. We built a small BSL alphabet video dataset that contained 8 examples of each letter compiled mostly of self-recordings. Three different people in different settings were part of the video examples, with one set of examples coming from SignBSL[2]. We used OpenPose [1] on frames of example videos to get a list of body and hand keypoints for each frame - in other words, a keypoints sequence for each example video. OpenPose is a fast, accurate, real-time pose estimator. To collect the keypoint sequences, we created a Python script to go through directories of videos and output a JSON dictionary for each example sign in the video dataset.

2.2 Virtual Human Agent Model

For the model, we created a simple 2D human (see figure 1a) in Inkscape[3]. We used a 2D model instead of a 3D model because OpenPose can only estimate a 2D pose from a single camera perspective. Although a 3D model may be more comprehensible, it is also more difficult to obtain accurate 3D movement from 2D video examples and thus we believe it is better to leave the 2D to 3D motion 'processing' to the user as they will likely much better comprehend 2D motion into 3D motion. A 2D model is also more *lightweight* to animate. Our model is based upon the keypoints produced by OpenPose and only contains the upper body as the lower body is not used in the BSL alphabet. We created the model to be a collection of body parts that when combined into a pose, look like a human. By having the model being composed of simple, individual body parts, it made it easy to position, rotate, and scale the body parts such that the model can be *versatile*.

[1] https://www.british-sign.co.uk/wp-content/uploads/2013/05/BSL-Fingerspelling-Right-Handed-1024x724.png

[2] https://www.signbsl.com/

[3] https://inkscape.org/

IVA '20, October 19–23, 2020, Virtual Event, Scotland Uk

ACM ISBN 978-1-4503-7586-3/20/09.
https://doi.org/10.1145/3383652.3423898

2.3 Animating the model

We animated our model using Pyglet[4], a 2D Python game engine. Our implementation translates OpenPose keypoints into body part positions to mimic the sequence of poses in the video examples given. We had two types of body parts: dynamic and static. Dynamic body parts are assigned two OpenPose keypoints: a pivot point that is anchored near the top of the body part; and another point that the body part is to point to or *'draw a line'* to. To move a dynamic body part into a new position: the anchor of the body part is set to the position of one of the keypoints; trigonometry is used to determine the angle (rotation) and magnitude (scale) between the keypoints; the body part rotation and scale (length) are set accordingly. Static body parts differ in only two ways when compared to dynamic body parts: they have an anchor point in a more central location and use the midpoint between its two given keypoints for its position.

To go from posing to animating the agent, we set up a loop that got the next pose from the OpenPose keypoint sequence and moved the agent into this pose. We then allowed the agent to sign any text by building a sequence of OpenPose keypoints for each letter of the text and concatenating them, giving us a sequence of poses for the entire text provided.

(a) Virtual agent **(b) Video frame the virtual agent pose is based on**

Figure 1: Comparison of the virtual and human agent in the same pose. See the virtual agent in action go to https://youtu.be/7p8U6BFdHBI.

Animations were smoothed using a moving average across the OpenPose keypoint frames. Pauses were also added between words by repeating the last keypoint frame of the last letter signed, leading to a gradual motion pause when combined with the smoothing. The drawing order is fixed in our implementation, meaning that certain body parts are always drawn on top of others. This doesn't adhere to the *versatile* characteristic and thus is a weakness that should be worked on in future work. Finally, we also dynamically scaled the width of some of the body parts if OpenPose points were available to do so - this helped introduce a sense of depth to the body parts, e.g. how close the face is or if the hand is flipping over.

You can access our code repository[5] to explore the code base.

[4]http://pyglet.org/
[5]https://gitlab.com/MatthewMcConnell/sign-al

3 EVALUATION

Our evaluation design is based on previous work on the comprehensibility of a virtual sign-language agent [4]. We designed our evaluation to compare the comprehensibility of our virtual agent to a real human. We performed a within-subjects evaluation where users would see both types of agents individually and attempt to write down what the agents signed.

The independent variable of this evaluation is the type of agent; the virtual agent was projected on a laptop screen, while the human agent signed in person in front of the participants. We removed a potential confounding variable of the human and virtual agent having different signing styles by having the animation synthesis pipeline use only example videos of the human agent.

In this evaluation, we used ten short English phrases that could be signed. The phrases were distinct, and some of the phrases are uncommon or unlikely to be said in everyday conversation. To ensure that every letter of the alphabet was seen multiple times, each letter was within at least three phrases. The ten phrases were:

(1) Hello, I saw a gentle cat
(2) I would like a tea jug from you
(3) Very Beautiful Keyboard
(4) We are from New Zealand
(5) Extraordinary Quebec TV
(6) Here is your IQ and x-ray
(7) Can you open this jar for me?
(8) Quiz extravaganza prize
(9) Many unbuckled seat belts
(10) People are jerks in a zoo

Each user saw both the virtual and human agent sign five phrases each. The ten phrases were randomly divided to the agents such that they would each get five phrases. Thus, every user saw all of the ten phrases but may have seen phrases on a different type of agent compared to other users. The order the agents signed their allocated phrases was also randomised so that the potential impact of the learning effect was neutralised. The type of agent shown first to the user was also alternated.

For this study, we recruited participants with some knowledge of the BSL alphabet. In total, 11 people participated in the study - a small sample size for most evaluations, but more representative of the population than usual due to the small population size of people who know the BSL alphabet. Users consisted mainly either of beginners or BSL experts.

4 RESULTS

After carrying out the evaluation, we computed a comprehensibility metric for every user, as follows. First, we computed the Levenshtein edit distance between the user's answers and the actual phrases signed. These values were then normalised by dividing by the actual phrase length. Finally, we took the mean for every user on each agent type to compute the final comprehensibility; note that a lower score on this metric corresponds to better comprehensibility.

Table 1 shows the results for each user on this metric, suggesting that every user comprehended the human agent better than the virtual agent. The data is paired and independent, with one independent variable, and was found to be non-normal by the Wilko-Shapiro test. Therefore, we used a one-tailed Wilcoxon

Table 1: The mean comprehensibility of the virtual and human agent for every user and the difference between them

User	Virtual Mean (SD)	Human Mean (SD)	Diff.
1	0.60 (0.11)	0.26 (0.30)	0.35
2	0.47 (0.09)	0.14 (0.08)	0.33
3	0.43 (0.24)	0.23 (0.12)	0.20
4	0.47 (0.04)	0.20 (0.17)	0.27
5	0.46 (0.07)	0.15 (0.07)	0.31
6	0.23 (0.18)	0.11 (0.07)	0.11
7	0.55 (0.17)	0.14 (0.18)	0.41
8	0.58 (0.15)	0.41 (0.11)	0.17
9	0.30 (0.13)	0.13 (0.06)	0.17
10	0.74 (0.29)	0.41 (0.17)	0.33
11	0.42 (0.12)	0.19 (0.15)	0.22

signed-rank test with $p = 0.05$ to test our hypothesis that the virtual agent was less comprehensible than the human agent. The statistical test produced a Wilcoxon value of $W = 66$ and a p-value of $p = 0.00192 < 0.05$. Thus, we can reject the null hypothesis, confirming that the virtual agent was less comprehensible.

Even though we found the virtual agent to have lower scores than the human agent, it is important to note that the data shows that users did not find the virtual agent completely incomprehensible. Many of the differences between the human agent and virtual agent comprehensibility within users were not large; In particular, users 6 and 9 comprehended the virtual agent very well.

As the BSL experience of the participants varied widely, including both experts and beginners, we also investigated whether the user experience influenced the agent comprehensibility. Figure 2 shows the mean of the comprehensibility metric of users grouped by experience—note that user 10 (BSL expert) was an extreme outlier and was not included in this graph. As seen in the figure, there were no significant differences in comprehension among the user groups.

During the user evaluation, users also gave informal feedback about the virtual agent. Many users said that it was difficult to distinguish between certain signs in the BSL alphabet, partly due to how similar body parts looked. As shown in Figure 1a, our model has no difference between the back and front of the hand, so knowing which side of the hand is showing is difficult. Also, users mentioned that it was difficult to see what body part is supposed to be on top of another, partly due to the fixed drawing order employed.

5 CONCLUSIONS

We have created an animation synthesis pipeline that created 2D British Sign Language alphabet animations without any costly manual animation or computational resources required using OpenPose, a 2D model and Pyglet. Our animation synthesis pipeline successfully had the attributes of being: *lightweight* as it ran in real-time on conventional hardware; and *general* since it could be applied to any sign language using video examples.

We completed an evaluation comparing the comprehensibility of the virtual agent to a human agent. The results of this evaluation showed that although the virtual agent was less comprehensible

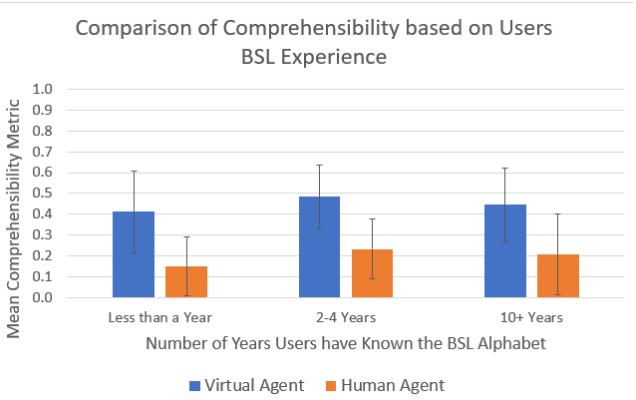

Figure 2: Mean comprehensibility of the virtual and human agent for user groups categorised by experience

than the human agent, users were still generally able to understand the signs it made, meaning that we had partial success at being *comprehensible*; comprehensibility also did not depend on the user experience with BSL. The user feedback in the evaluation also identified issues that could have affected the agent comprehensibility. More comprehensibility could be found by adding more features while keeping the pipeline *lightweight*. One example not explored in our work is facial expressions which are important within sign language and can allow for lip-reading.

Our pipeline struggled to be *versatile*. Thus, potential future work could be done such as obtaining and drawing an outline on the virtual agent to improve depth and help distinguish which body part is on top of another on the virtual agent. A further developed version of the animation synthesis pipeline could also attempt to sign a robust average of many examples of the same sign.

In summary, we believe the lightweight animation synthesis pipeline we have created is novel and promising, even though it is not as comprehensible as a human. Thus, we believe that future work improving this pipeline would be beneficial and that it could be suitable for eventual use within an end-to-end sign language conversational agent applicable to any sign language.

REFERENCES

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *arXiv:1812.08008 [cs]* (May 2019). http://arxiv.org/abs/1812.08008 arXiv: 1812.08008.

[2] Stephen Cox, Michael Lincoln, Judy Tryggvason, Melanie Nakisa, Mark Wells, Marcus Tutt, and Sanja Abbott. 2002. Tessa, a System to Aid Communication with Deaf People. In *Proceedings of the Fifth International ACM Conference on Assistive Technologies (Assets '02)*. Association for Computing Machinery, New York, NY, USA, 205–212. https://doi.org/10.1145/638249.638287

[3] Amanda Cardoso Duarte. 2019. Cross-modal Neural Sign Language Translation. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM. https://doi.org/10.1145/3343031.3352587

[4] Michael Kipp, Alexis Heloir, and Quan Nguyen. 2011. Sign Language Avatars: Animation and Comprehensibility. In *Intelligent Virtual Agents (Lecture Notes in Computer Science)*, Hannes Högni Vilhjálmsson, Stefan Kopp, Stacy Marsella, and Kristinn R. Thórisson (Eds.). Springer, Berlin, Heidelberg, 113–126. https://doi.org/10.1007/978-3-642-23974-8_13

[5] Stephanie Stoll, Necati Cihan Camgoz, Simon Hadfield, and Richard Bowden. 2020. Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks. *International Journal of Computer Vision* 128, 4 (Jan 2020), 891–908. https://doi.org/10.1007/s11263-019-01281-2