Automatic generation of a 3D sign language avatar on AR glasses given 2D videos of human signers

Conference Paper · October 2021

CITATIONS
0

READS
689

4 authors, including:

Lan Thao Nguyen
Technische Universität Berlin
3 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Florian Schicktanz
Technische Universität Berlin
2 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Eleftherios Avramidis
Deutsches Forschungszentrum für Künstliche Intelligenz
77 PUBLICATIONS **546** CITATIONS
SEE PROFILE

Automatic generation of a 3D sign language avatar on AR glasses given 2D videos of human signers

Lan Thao Nguyen lan.t.nguyen@campus.tu-berlin.de
Florian Schicktanz
Technische Universität Berlin, Berlin, Germany

Aeneas Stankowski aeneas.stankowski@dfki.de
Eleftherios Avramidis eleftherios.avramidis@dfki.de
German Research Center for Artificial Intelligence (DFKI), Berlin, Germany

Abstract

In this paper we present a prototypical implementation of a pipeline that allows the automatic generation of a German Sign Language avatar from 2D video material. The presentation is accompanied by the source code. We record human pose movements during signing with computer vision models. The joint coordinates of hands and arms are imported as landmarks to control the skeleton of our avatar. From the anatomically independent landmarks, we create another skeleton based on the avatar's skeletal bone architecture to calculate the bone rotation data. This data is then used to control our human 3D avatar. The avatar is displayed on AR glasses and can be placed virtually in the room, in a way that it can be perceived simultaneously to the verbal speaker. In further work it is aimed to be enhanced with speech recognition and machine translation methods for serving as a sign language interpreter. The prototype has been shown to people of the deaf and hard-of-hearing community for assessing its comprehensibility. Problems emerged with the transferred hand rotations, hand gestures were hard to recognize on the avatar due to deformations like twisted finger meshes.

1 Introduction

About one million people in Europe are deaf and mainly communicate using sign language (SL), which consists of gestures of hands, arms and facial expressions. Outside their daily routines, deaf or hard-of-hearing (DHH) people face barriers for independently participating in society. The communication between signers and hearing speakers is often supported by trained SL interpreters. Beyond their respective communities, the availability of these services is often lacking or costly. New technological solutions may provide democratic alternatives to expensive and scarce translation resources and enable an independent communication between hearing and DHH people.

Our approach is based on the assumption that a simultaneous translation from spoken language to SL can enable more direct communication between deaf and hearing people. Through interviews with members of the DHH community, it was apparent that translations are understood even better when the interpreter's signs can be perceived with the speaker's mouth and gestures at the same time. If the person has a good picture of the mouth movement, around 30% of the speech can be read from the lips (Deutscher Gehörlosen-Bund e.V., 2021).

Several phases of the development were conducted with the support of members from the DHH community through a co-operation with the Center for Culture and Visual Communication

of the Deaf in Berlin and Brandenburg (ZFK)[a] as described in our previous publication (Nguyen et al., 2021). In the discovery phase, qualitative interviews were conducted to provide user insights and lead the basic design decisions, including the choice of developing an avatar on AR glasses, as mentioned. In the early implementation phase, a professional deaf SL interpreter was video-recorded to provide material for the SL animation. In the evaluation phase, the efficacy of our proposed solution was evaluated via a wizard-of-Oz experiment, where a prototypical avatar was displayed in a way that it appeared as a fully developed automatic interpreter.

In this paper we focus on the technical implementation of this avatar. The 2D videos of the SL interpreter were analyzed with computer vision methods to extract human joint coordinates (landmarks). After mapping these landmarks on a pre-built avatar model, the virtual interpreter was displayed on a the AR glasses in the deaf person's field of view.

In the next chapter we give an overview of the related work. Chapter 3 provides the details about the implementation. In chapter 4 the evaluation process is described, whereas in chapter 5 we give a conclusion and indications for further research.

2 Related Work

In the last 30 years, diverse approaches to the automatic generation of SL have emerged. Tokuda and Okumura (1998) presented a prototype system for a word-to-word translation of Japanese to Japanese Sign Language (JSL) by finger spelling. By the beginning of the century, Elliott et al. (2000) specified the first framework for producing avatar based SL from text. They proposed the Signing Gesture Markup Language (SIGML; Elliott et al., 2004), an XML-formatted SL sequence description to drive the animation of an avatar in a web browser. It is built on the Hamburg Notation System (HamNoSys Hanke, 2004) that allows phonetic representations of signs.

SiGML is still applied in recent sign generation concepts (Kaur and Singh, 2015; Verma and Kaur, 2015; Rayner et al., 2016; Sugandhi et al., 2020). It is used as input language for the SL animation system Java Avatar Signing (JASigning) (Elliott et al., 2010), the successor of SiGMLSigning (Elliott et al., 2004). This was applied by Rayner et al. (2016) for their open online SL translation application development platform and Sugandhi et al. (2020), who developed a system that produces Indian Sign Language (ISL) from English text while considering ISL grammar. For the correct grammar they created a HamNoSys database before converting the representations to SiGML.

Another architecture was introduced by Heloir and Kipp (2010). They presented the Embodied Agents Behavior Realizer (EMBR) engine for robust real-time avatar animation. It is controlled by the EMBRScript which defines key poses for animation sequences. The EMBR system was later extended by Kipp et al. (2011) to build a sign animation tool based on a gloss database.

More similar to our approach, there is recent work that builds upon open source machine learning solutions which track human body keypoints from 2D video material. McConnell et al. (2020) animated a two-dimensional virtual human that is able to sign the British Sign Language (BSL) alphabet, based on body landmarks estimated by the OpenPose library (Cao et al., 2019). Although we also focus on a method requiring no special hardware or costly computing resources, we produce translations on a sentence level and we animate an avatar on all three dimensions.

Our work uses the three-dimensional landmark prediction of the lightweight and fast MediaPipe framework (Lugaresi et al., 2019). This was also considered for the automatic recognition of SL (Harditya, 2020; Chaikaew et al., 2021; Halder and Tayade, 2021; Bagby et al., 2021; Bansal et al., 2021) in previous work, but not for producing SL. MediaPipe is compatible

[a]Zentrum für Kultur und visuelle Kommunikation Gehörloser in Berlin & Brandenburg e.V., Potsdam, Germany

to more hardware systems and has less requirements than OpenPose. With the 2D skeleton generated by OpenPose and generative adversarial networks, Stoll et al. (2020) and Ventura et al. (2020) elaborated novel approaches to generate signing videos with photo realistic humans.

Regarding the animation of 3D avatars based on 2D recordings, Brock et al. (2020) developed a pipeline that creates skeletal data from sign videos with the help of multiple recurrent neural networks, proposing a new pose estimation framework specially designed for SL. The authors also compare the features of different available architectures, including OpenPose and MediaPipe. They provide a solution that covers all compared capabilities. To drive their avatar, angular and positional joint displacements are calculated with inverse kinematics, the results are then saved in a BVH file. User experiments showed that the synthesized avatar signing is comprehensible.

For this paper we implemented as a first step a more simplistic method, using an existing machine learning architecture and driving a virtual avatar by raw bone rotation data calculated in a 3D graphic suite. We are aiming to reach likewise comprehensibility results with a more refined animation approach in the future.

3 Implementation

The goal of the implementation was to create a SL avatar that is displayed in a HoloLens 1[b] and can be used for a proof of concept where we access the acceptability and comprehensibility of a simulated real-time translation on AR glasses among people from the DHH community.

Converting 2D SL video to a 3D avatar has multiple benefits (Kipp et al., 2011). In a simple use case, one can reside to this kind of conversion to create avatars out of pre-recorded SL speeches, while preserving the anonymity of the speakers. In a more advanced use case, such as the one of the automatic SL interpretation that we aim at, a big amount of sequences of SL gestures recorded from human speakers are required as graphical linguistic units. These will be used by a unified pipeline that generates full SL sentences, including methods of speech recognition and machine translation, to be implemented in further work. Collecting bigger amounts of human SL speaker recordings is aided if 2D cameras are used, as this is straightforward, does not require advanced equipment and allows using videos of different signers.

Our process of creating a SL avatar for AR glasses consists of four steps. First, we collected video footage of the predefined phrases by a professional interpreter (section 3.1). Then, based on the collected video footage, motion capture was used to convert the gestures and facial expressions into tracking points, using a motion tracking model that analyzes images and extracts body landmark positions (section 3.2). Based on the motion tracking points, the animations were transferred to the skeleton of the avatar after calculating rotation vectors (section 3.3). Finally, an application was developed to display the animated avatar on the AR glasses (section 3.4). The application, scripts and other material are published as open source.[cd]

3.1 Collection of video material

We filmed a professional deaf interpreter translating written German sentences to German Sign Language (Deutsche Gebärdensprache; DGS). Since this footage is needed to track all human joint landmarks in every single video frame, it was necessary to capture high-resolution video with as few motion blur as possible. Using a single-lens reflex camera, this was achieved by setting a very short exposure time of 1/1600, a relatively low f-number of 2.8 and a very high ISO value of 3200 for strong light sensitivity. Moreover, a 1920x1080 full HD resolution and a frame rate of 25 FPS were chosen. The recording took place at a professional film studio of the

[b] https://docs.microsoft.com/de-de/hololens/hololens1-hardware

[c] https://github.com/lanthaon/sl-animation-blender

[d] https://github.com/lanthaon/sl-roleplay-unity

ZFK, where translations for the public TV are recorded as well. The recording lasted 4 hours and was supported by a professional film team. Three cameras filmed the interpretation from 3 different angles, but in this paper we only use the frontal camera. We will consider using the full footage including the side cameras in order to improve detection precision in future work.

3.2 Generating Motion Capture Data

The footage was processed with a pipeline integrating three machine learning models for computer vision that analyze the images and generate body landmarks with 3D coordinates for the upper body, hands and face. These are depicted in image A of figure 1. Face detection combines a compact feature extractor convolutional neural network with the anchor scheme of the Single Shot Multibox Detector (Liu et al., 2016; Bazarevsky et al., 2019). The upper body was analyzed via a body pose tracking model that uses an encoder-decoder heatmap-based network and a subsequent regression encoder network (Bazarevsky et al., 2020). The hands were analyzed with a palm detector, combined with a hand landmark detection model via multi-view bootstrapping (Simon et al., 2017; Zhang et al., 2020).

We used the tools including the pre-trained models of MediaPipe Holistic (Grishchenko and Bazarevsky, 2020) and wrote a Python script for the open source 3D graphic suite *Blender version 2.91.2*[e] to create exportable motion capture data based on the landmarks. For the scripting, we worked with the *Blender Python module*, *MediaPipe version 0.8.3.1* and *OpenCV version 4.5.1.48* under *Python 3.7*.

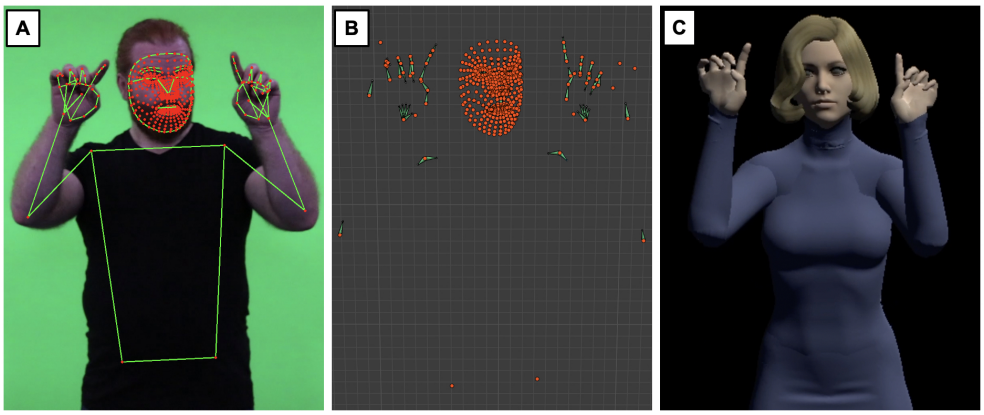

Figure 1: Holistic tracking applied to a video frame. A) is the annotated original footage where the red dots are the tracked landmarks, while the green lines connect the joints. B) is the same frame in Blender with landmarks plotted as orange spheres and cyan coloured bones. C) shows the motion capture data applied to an avatar.

In order to extract the landmarks frame by frame, we created the method `get_video_frames` that splits the video into an array of frames. It passes a video file URL as string to the `VideoCapture` class of OpenCV, which then decodes the video frames.

The resulting frame list can then be used in method `get_landmarks`. Within the function, a for-loop is called which iterates the frame array, calculates the landmarks with MediaPipe Holistic and saves them for each frame and type into the corresponding arrays for pose, left hand, right hand and face. In the optimal case, after the last iteration the arrays should contain n lists of tracked landmarks, where n is the number of frames. If for example the left

[e] https://www.blender.org/download/releases/2-91/

Mesh type	*Polygon count*
Hair	13.035
Shoes	4808
Dress	149.122
Shape	16.720
Total	$183.685 < 200.000$

Table 1: Polygon count of the avatar mesh divided into categories

hand could not be tracked in the first frame, a `NoneType` object is added to the array instead of a landmark list.

After gathering the landmarks into arrays, they can be passed to `load_landmarks_into_scene`. While iterating the specified landmark array, sphere objects are created once for each existing landmark and are named uniquely after their corresponding MediaPipe landmark model designation. To update the spheres' locations analogous to the video frame, keyframes are set in each iteration. The XYZ coordinates are multiplied with a factor of either 30 or 40 to stretch the distances between the points, enabling to better analyze the results visually. Image B in figure 1 shows the landmark cloud, highlighted in orange, for an exemplary keyframe.

The last step is to built an armature with bones based on the animated landmark cloud which is done in method `create_bones`. For this, lists of string tuples with names of start and target landmarks were defined to specify between which two landmarks a bone should be created. The first tuple item is the name of the start landmark. There, the bone's root will be located. The second item is the target landmark that determines the direction to where the bone's end joint should point at. These rules were implemented as bone constraints to which the bones adapt automatically in each keyframe.

Finally, the resulting skeleton was exported as a Biovision Hierarchy (BVH) file, which is a common file type for motion capture data. After that, the next step was to apply the animation data in the BVH file to a 3D character, like depicted in C of figure 1. This will be described further in the following subsection.

3.3 Creating and animating the avatar

For the creation of the 3D avatar, the 3D modelling software *Daz Studio*[f] has been used. The Daz community provides free 3D content as well as 3D content for a fee. Only free 3D content has been chosen for the character shape and assets like hair, cloth and shoes. To work with the character in Blender, it has been exported with the *Daz to Blender Bridge*[g].

When developing apps for the HoloLens, it should be considered that the device is a self-contained computer with the processing power of a mobile phone. It is recommended to limit the overall polygon count to under 200,000 faces. We could keep the polygon count under this level by choosing less elaborately designed assets. The number of faces for each asset are summarized in Table 1.

3D characters created in Daz are fully rigged and prepared for animation purposes. The character's bone structure has been analyzed to figure out the important upper body bones to which our BVH skeleton had to be adjusted. To apply the produced BVH data, both the character and the BVH armature were imported into a Blender scene. In a Python script we defined the matching bone pairs between their skeletons and set bone constraints on the

[f] https://www.daz3d.com/get_studio

[g] https://www.daz3d.com/daz-to-blender-bridge

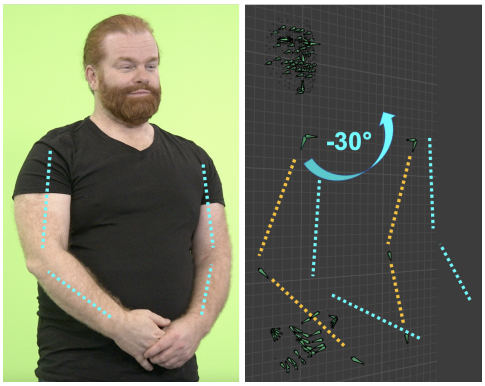

Figure 2: Original video frame compared to BVH skeleton based on MediaPipe tracking. The arms have a narrower angle than the original, while hands and face were tracked properly. Starting from a perfect vertical alignment, a more natural orientation was achieved by rotating the skeleton -30° along the corresponding axis.

avatar to copy the rotation of the assigned BVH bones. In listing 1 is an example variable `pose_bones_mapping` that defines the mapping for the arm rotations calculated based on the pose landmarks. The left entries are the bone names from the motion capture skeleton, the right entries bone names from the Daz avatar. We named the BVH bones after their target landmark.

Listing 1: Variable defining matching arm bones between the Daz avatar skeleton and the BVH armature

```
pose_bones_mapping = [
    ("LEFT_ELBOW", "lShldrBend"),
    ("LEFT_WRIST", "lForearmBend"),
    ("RIGHT_ELBOW", "rShldrBend"),
    ("RIGHT_WRIST", "rForearmBend")
]
```

Analyzing the resulting motion capture data in Blender revealed that hand, arm and face landmarks were not tracked coherently by MediaPipe, they rather seemed disconnected from each other. In figure 2 the face, arm and hand positions produced by the machine learning solution can be compared to the original recorded person. While hands and face seem to have proper results, the arms appear to be angled narrower which indicates an inaccurate depth estimation. Since our method copies only bone rotations and not bone positions, no full alignment is necessary, but the skeleton had to be adjusted due to the unnatural orientation of the arms before applying the rotations to the avatar. This was solved by writing a short function that adapts the global rotation for the BVH skeleton as visualized on the right image of figure 2.

The mapping of the bone rotations was done in method `map_bones` which iterates through all bone pairs of a tuple list like in listing 1. There the rotations from the BVH armature in the scene are to copied to the avatar's bones. In general, all XYZ axes are considered for copying the rotation, except for the head and neck bones, where the y axis is excluded. Because there was still the problem of the too narrow arm angle, the influence of the rotation copy constraint for the forearm bone has been reduced. In some animations the avatar's forearms or the hands would otherwise penetrate other body parts like the face.

For the proof of concept, only rotations for arms, hand and experimentally XZ rotations for head and neck bones were transferred to the avatar. The implementation of facial expressions

constitutes additional challenges that have to be taken on in future work, since they represent an essential element of the SL. Head and neck bones were created between specific face bones, but in principle, no tracking landmarks exist for the posture.

To continue with the development of the HoloLens application, the animated avatar was exported as a Filmbox (FBX) file that can be imported by the required Unity game engine[h].

3.4 HoloLens Application Development

We used *Unity version 2019.4.20f1 (LTS)* for the development together with *Microsoft's Mixed Reality Toolkit (MRTK) for Unity version 2.5.3*[i]. The toolkit provides a bounding box into which we embedded our avatar, enabling to position, rotate and scale the virtual character in space. With this feature we could place the avatar independently of the environment in a way that the study participant could see both the speaking person and the virtual interpreter avatar. To avoid that the 3D character is accidentally moved somewhere else during the user study, we implemented a virtual start button that disables the bounding box control before reaching the AR glasses to our participants.

An additional physical clicker that connects to the HoloLens 1 via Bluetooth allowed us to trigger actions during the study without wearing the AR glasses ourselves. The main camera's pointer handler of the Unity scene listens to the global event *On Pointer Clicked*. When this event occurs, a method is called that plays the next animation defined in a list containing the 3D avatar's motion clip names.

After our first application prototype we experienced that the AR glasses' computational limits were exceeded when more than ten avatars had to be handled, whereby each of them signed a different sentence. App size and computational complexity increased due to the many objects to a level that the HoloLens could not handle. It crashed each time after starting the app. Thus, our solution was to merge all animation clips needed for the role play in the user study into one avatar. The Unity app was finally deployed on the HoloLens with the *Community Edition of Visual Studio 2019*[j]. Figure 3 illustrates how the avatar was positioned in the study room.

Figure 3: Final user test setup. A participant (left) is wearing a HoloLens 1 displaying a virtual avatar in front of the table by the impersonated doctor (right), while a sign interpreter (middle) is prepared to translate the participant's feedback.

[h]https://unity.com/de

[i]https://docs.microsoft.com/de-de/windows/mixed-reality/mrtk-unity/

[j]https://visualstudio.microsoft.com/de/vs/

4 Evaluation

Our evaluation comprised a user study and discovery phase prior to the implementation (Nguyen et al., 2021). In the discovery phase we reached out to the Center for Culture and Visual Communication of the Deaf in Berlin and Brandenburg (ZFK), which is a major contact point for deaf people in the area. Four qualitative interviews were conducted and recorded to gather insights about the DHH communities' needs.

We could identify doctor appointments as a difficult situation for deaf people without an interpreter. Additionally, the interviewees rated different mock-ups, showing three translation options on three different technical devices. AR glasses combined with a virtual avatar were considered as one of the two most useful options. The option of having a live interpreter displayed in AR glasses was rated as equally helpful, yet would only slightly improve the status quo, as it would make such service locally dependent and the limited availability of interpreters still remains a problem.

To test the prototype system, we conducted a user study similar to a wizard-of-Oz-experiment (Nguyen et al., 2021). The use case of a medical examination was chosen during which the doctor asks the patient several questions. In that context, an examination dialog with predefined sentences was written and translated to SL. In the context of the dialog, the doctor is expecting certain reactions from the DHH patient, which decrease the risk of wrong understanding e.g. *"Show me, where you do have pain?"*. One of the experiment supervisors performed the role of the doctor and read aloud the prepared sentences, while another supervisor used an external clicker to trigger the corresponding animations on the AR glasses that were worn by the participants. Speech recognition will have to be integrated in future implementations for an independent live translation system.

We are conscious that medical usage requires high precision which is hard to be achieved by the state of the art. Nevertheless, we proceeded with this case for the experiment, since it was suggested during the discovery phase interviews as one of high importance for the users' community and could therefore motivate better the evaluation of the full concept.

The whole scene was filmed and a survey of standardized questions was asked at the end of a user test. Additional information about the qualitative results may be found in the prior published report (Nguyen et al., 2021).

Among the three female and five male participants from which six were deaf, there were also one hearing and one hard-of-hearing person. Results showed a high acceptance rate of the presented solution, even though the comprehensibility of the avatar's signing was rather low. Participants had difficulties to identify the movements of the fingers, as the actual positions of the individual finger joints were harder to distinguish through the often warped or deformed hand mesh. Sometimes the movements of the avatar's hands were furthermore jittery or incorrect due to technical reasons concerning the fidelity of the implementation. Besides, the study participants felt affected by the missing lip movements and facial expressions, which are relevant for performing grammatical functions. For example, raising the eyebrows indicates a question in DGS. With the chosen level of quality the purpose of conducting a proof of concept was achieved, but can be improved in the future.

To summarize, sentences that were signed mainly with finger movements showed a poorer comprehensibility rate than sentences with prominent arm movements that relied less on specific hand gestures. Although the arms seem to have been tracked less accurately than the hands, after being transferred to the avatar their movements were easier to recognize than finger movements, as the overall size and length of the movements mitigate the lower landmark precision. The mesh of the avatar's fingers was often deformed or twisted after copying the rotation of a bone created between two MediaPipe landmarks. We assume that the approach needs to be optimized mathematically, since the bone rotations calculated automatically with Blender are not a fully

accurate solution to generate SL from 2D videos. In addition, there were some inaccuracies in the depth data estimated by MediaPipe, which led to altered distances between different parts of the body, like hands and face. Also, the spine is not tracked, thus the posture of the avatar cannot be influenced yet if for example the head is shifted forward in the video.

5 Conclusion

We developed a method to generate SL from 2D video recordings with the help of machine learning methods of computer vision and a 3D creation suite. A skeleton was built from the detected body landmarks by creating bones between pairs of specified landmarks. This was then exported as BVH data and applied to a rigged avatar. The avatar's bone constraints were set to copy the rotations of the matching bones from the BVH skeleton.

Some problems became apparent after mapping the bone rotations to the avatar. Besides modified distances between hands or arms to the body, the most severe issue was the twisting of the finger meshes which altered their appearance significantly. The user study with our prototype system showed that the avatar's hand gestures were hard to recognize for most participants. This led to poorer comprehensibility for sentences where the attention had to be paid predominantly to the finger movements, while sentences with discernible arm movements were recognized by the majority.

As it has been noted, the animation approach needs to be optimized to achieve more comprehensible results. Facial expressions and if possible lip movements should be enabled in future implementations. Still, a high acceptance level could be observed for the concept of displaying a SL avatar in AR glasses. We believe that a well developed automated speech to SL system could enable more freedom and flexibility for deaf people in situations where an interpretation can enhance communication significantly, but would be normally not affordable due to the scarce availability of interpreters.

Even if our prototype system for the experiment is static through the predefined, manually triggered questions, it opens the door for several use cases where content and vocabulary are restricted, e.g. a museum tour. Moreover, it was important for us to create an animation solution that requires no special hardware and enables other researchers as well as non-professionals an open-source tool for producing three-dimensional virtually performed SL with an avatar. Our vision is to create the basis for an animation data-set which can grow with the addition of more scenarios and can be used for the further development of a system allowing the real-time translation of arbitrary conversations, in conjunction with methods from speech recognition and machine translation.

Acknowledgements

This research was supported by Uwe Schönfeld, Mathias Schäfer, Helene Sudermann, Lukas Gesche, Johann Carl Reischies, Marco Richardson and Willi Reuter. It has been accomplished as a semester project at the Technische Universität Berlin, hosted by the Quality and Usability Lab and organized by Neslihan Iskender under the chair of Prof. Sebastian Möller. Supervision from the side of DFKI has been supported by the project SocialWear (Socially Interactive Smart Fashion) funded by the German Ministry of Education (BMBF), with the support of Fabrizio Nunnari and Esther Zahn.

References

Bagby, B., Gray, D., Hughes, R., Langford, Z., and Stonner, R. (2021). Simplifying sign language detection for smart home devices using google mediapipe. https://bradenbagby.com/Portfolio/Resources/PDFs/ResearchPaper.pdf.

Bansal, D., Ravi, P., So, M., Agrawal, P., Chadha, I., Murugappan, G., and Duke, C. (2021). Copycat: Using sign language recognition to help deaf children acquire language skills. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–10.

Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., and Grundmann, M. (2020). BlazePose: On-device Real-time Body Pose tracking. *CoRR*, abs/2006.1.

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., and Grundmann, M. (2019). BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. *CoRR*, abs/1907.0.

Brock, H., Law, F., Nakadai, K., and Nagashima, Y. (2020). Learning three-dimensional skeleton data from sign language video. *ACM Trans. Intell. Syst. Technol.*, 11(3).

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., and Sheikh, Y. (2019). Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186.

Chaikaew, A., Somkuan, K., and Yuyen, T. (2021). Thai sign language recognition: an application of deep neural network. In *2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering*, pages 128–131.

Deutscher Gehörlosen-Bund e.V. (2021). Wie viele Gehörlose gibt es in Deutschland? https://www.gehoerlosen-bund.de/faq/geh%C3%B6rlosigkeit.

Elliott, R., Bueno, J., Kennaway, R., and Glauert, J. (2010). Towards the integration of synthetic sl animation with avatars into corpus annotation tools. In *4th Workshop on the Representation and Processing of Sign Languages: Corpora and Sign Language Technologies, Valletta, Malta*, page 29.

Elliott, R., Glauert, J. R., Jennings, V., and Kennaway, J. (2004). An overview of the sigml notation and sigmlsigning software system. In *Fourth International Conference on Language Resources and Evaluation, LREC*, pages 98–104.

Elliott, R., Glauert, J. R. W., Kennaway, J. R., and Marshall, I. (2000). The development of language processing support for the visicast project. In *Proceedings of the Fourth International ACM Conference on Assistive Technologies*, Assets '00, page 101–108, New York, NY, USA. Association for Computing Machinery.

Grishchenko, I. and Bazarevsky, V. (2020). MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device. http://ai.googleblog.com/2020/12/mediapipe-holistic-simultaneous-face.html.

Halder, A. and Tayade, A. (2021). Real-time vernacular sign language recognition using mediapipe and machine learning. *International Journal of Research Publication and Reviews*, 2(5):9–17.

Hanke, T. (2004). Hamnosys-representing sign language data in language resources and language processing contexts. In *LREC*, volume 4, pages 1–6.

Harditya, A. (2020). Indonesian sign language (bisindo) as means to visualize basic graphic shapes using teachable machine. In *International Conference of Innovation in Media and Visual Design (IMDES 2020)*, pages 1–7. Atlantis Press.

Heloir, A. and Kipp, M. (2010). Real-time animation of interactive agents: Specification and realization. *Applied Artificial Intelligence*, 24(6):510–529.

Kaur, S. and Singh, M. (2015). Indian sign language animation generation system. In *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, pages 909–914. IEEE.

Kipp, M., Heloir, A., and Nguyen, Q. (2011). Sign language avatars: Animation and comprehensibility. In Vilhjálmsson, H. H., Kopp, S., Marsella, S., and Thórisson, K. R., editors, *Intelligent Virtual Agents*, pages 113–126. Springer Berlin Heidelberg.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9905 LNCS, pages 21–37. Springer Verlag.

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M. G., Lee, J., Chang, W.-T., Hua, W., Georg, M., and Grundmann, M. (2019). Mediapipe: A framework for building perception pipelines.

McConnell, M., Foster, M. E., and Ellen, M. (2020). Two Dimensional Sign Language Agent. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*, New York, NY, USA. ACM.

Nguyen, L. T., Schicktanz, F., Stankowski, A., and Avramidis, E. (2021). Evaluating the translation of speech to virtually-performed sign language on ar glasses. In *Proceedings of the Thirteenth International Conference on Quality of Multimedia Experience (QoMEX). International Conference on Quality of Multimedia Experience (QoMEX-2021), June 14-17*. IEEE.

Rayner, E., Bouillon, P., Gerlach, J., Strasly, I., Tsourakis, N., and Ebling, S. (2016). An openweb platform for rule-based speech-to-sign translation. In *54th annual meeting of the Association for Computational Linguistics (ACL)*.

Simon, T., Joo, H., Matthews, I., and Sheikh, Y. (2017). Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1145–1153.

Stoll, S., Camgoz, N. C., Hadfield, S., and Bowden, R. (2020). Text2Sign: Towards Sign Language Production Using Neural Machine Translation and Generative Adversarial Networks. *International Journal of Computer Vision*, 128(4):891–908.

Sugandhi, Kumar, P., and Kaur, S. (2020). Sign Language Generation System Based on Indian Sign Language Grammar. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 19(4):1–26.

Tokuda, M. and Okumura, M. (1998). Towards automatic translation from japanese into japanese sign language. In *Assistive Technology and Artificial Intelligence*, pages 97–108. Springer.

Ventura, L., Duarte, A., and i Nieto, X. G. (2020). Can Everybody Sign Now? Exploring Sign Language Video Generation from 2D Poses. In *ECCV 2020 Workshop on Sign Language recognition, Production and Translation (SLRTP)*.

Verma, A. and Kaur, S. (2015). Indian sign language animation generation system for gurumukhi script. *International Journal of Computer Science and Technology*, 6(3):117–121.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., and Grundmann, M. (2020). MediaPipe Hands: On-device Real-time Hand Tracking. *CoRR*, abs/2006.1.