Real-time face retargeting and a face rig on the web

Fig. 1. Screen captures of the web-based real-time facial retargeting with different actors and virtual characters.

Eva Valls Garolera
Interactive Technologies Group
Universitat Pompeu Fabra
Barcelona, Spain
eva.valls@upf.edu

Gerard Llorach
Hörzentrum Oldenburg GmbH
Oldenburg, Germany
g.llorach@hoerzentrum-oldenburg.de

Javier Agenjo
Interactive Technologies Group
Universitat Pompeu Fabra
Barcelona, Spain
javi.agenjo@upf.edu

Josep Blat
Interactive Technologies Group
Universitat Pompeu Fabra
Barcelona, Spain
josep.blat@upf.edu

***Abstract*—Virtual characters are a key element in human-computer interaction, as these enhance the communication process via facial expressions, gestures and body postures. The process of creating facial animations is a time-consuming and laborious task. In this paper we present our recent advances in our web-based tools to animate faces. Our tools try improve the control of the facial expression through simple interfaces. We demonstrate that real-time facial retargeting on the web is possible and we present a facial rig based on natural neighbor interpolation between predefined facial expressions.**

Keywords—facial retargeting, facial rig, web, real-time, 3D graphics.

I. Introduction

The web has become the standard platform to share information due to its ubiquity and interoperability. And although the web started as a way to share text information and multimedia content, currently thanks to the standard APIs WebGL it is possible to render high-quality 3D content on the Web, which open new ways of interactivity between the user and the content [1].

In recent years, virtual characters have been increasingly appearing in different applications becoming a key component in human-computer interaction [2,3]. It is well known that non-verbal behavior such as facial expressions, the gaze or blinking plays an important role in this type of communication. So, to do this plausible, the digital models have to move and express themselves in a natural way. These behaviors need to be created and generated with animations. There are many kinds of techniques to animate 3D characters, but the most known are motion capture and keyframe animation.

Current real-time motion capture systems map the actor's performance to their equivalent 3D model. These systems can speed up the process of animation, as they can generate high-precision animations in real-time and these can be recorded. Facial motion capture systems track low-level features such as facial landmarks. These captured features of the actor need to be transferred to the corresponding 3D model. The lower the captured features are, the more complex and individual the facial retargeting will be. The Facial Action Coding System (FACS) [4] offers a good compromise as the features, the Facial Action Units (AUs), are independent of the person and describe facial muscle movements.

Keyframe animation is a more artistic and slow process. The animator decides where, when and how each part of the character will move. The most common low-level controls for facial keyframe animation are blend shapes or skeletal animation with bones/joints. Blend shapes, also called morph targets, have one control parameter and usually define a specific facial action or expression, e.g. smile, eyebrows up, surprise. With skeletal animation, each bone has 9 control parameters (translation, rotation, scale) and usually affects a region of the face, e.g. superior eyelid, right mouth corner. As the low-level control parameters can escalate quite quickly, the animation process can be quite time consuming. High-level controls such as facial rigs usually ease the creation of animations. They permit the control of several low-level control parameters through simple and intuitive interfaces.

Facial rigs are commonly platform dependent and usually cannot be transferred to other applications. Even more, in some cases the final animations created with rigs cannot be exported and used in other platforms. This issue is particularly important for web applications, as web technologies to animate virtual characters are quite scarce. Currently there are applications that are capable of reproducing facial animations on the web. Talking heads [5][6], tutoring agents [7] and embodied conversational agents [8] use blend shapes, skeletal animation and standards such as the MPEG-4 to generate facial expressions [1]. Nevertheless, some of these applications rely on external software to create animations. For simplistic animations where rigs are not required it is more efficient to use already established animation software. But for animations with complex rigs that are platform dependent, these applications are limited.

One example of a facial rig paradigm is [9], which has been implemented in the web by [10]. The idea behind this facial rig is to use the valence-arousal emotional space and to control the facial expression by moving a point in a 2D space. Nevertheless, in the aforementioned literature the

[1] Stickman Ventures, "Ginger WebGL Morph Demo". Available at: https://sv-ginger.appspot.com/. Accessed July 2019.

facial rig is limited to predefined set of facial expressions. Due to its interpolation formula, it is not possible to include additional facial expressions.

In this work we implemented a web-based real-time markerless facial retargeting system. We used the library Beyond Reality Face[2] (BRFv4) for facial landmark tracking and the method proposed by Sheng and Kay (2016) [12] for facial retargeting. The system transfers the facial expressions of the user's face to a 3D character rendered in real-time. Our first contribution is that our system works in the client's browser with WebGL and that we only require a webcam instead of a depth camera as in [12]. We evaluated this facial retargeting system and listed possible issues and tracking errors. Our second contribution is the improvement of an existing facial rig [10] by changing the interpolation formula between facial expressions. This new facial rig permits customization and addition of new facial expressions. Our tools have been implemented in WebGLStudio [11], an open source platform, and are easily transferable between virtual characters thanks to the node-based system developed.

II. Method

Our system offers two methods for facial control of a virtual actor: **facial retargeting** and **facial rig with 2D interpolation**. Facial retargeting consists on the reproduction of the user's facial expressions on a virtual character in real-time using a simple webcam. This method allows the user to control the character with his/her own body as a mirror. Contrary to it, the facial rig uses preconfigured expressions of the character which are controlled through an interface, e.g. a gamepad or a web interface.

In this work we use blend shapes to generate facial expressions in the virtual actor. The methods presented here are also compatible with skeletal animation. WebGLStudio supports skeletal animation and blend shapes using the GPU, thus permitting real-time generation of complex facial expressions and animations.

A. Facial Retargeting

In this section we describe the method we used for real-time facial retargeting system. We used the library BRFv4 and the method proposed by Sheng and Kay (2016) [12]. One important difference from the aforementioned method is that we used 2D landmarks instead of 3D. An overview of the facial retargeting system can be seen in Figure 2.

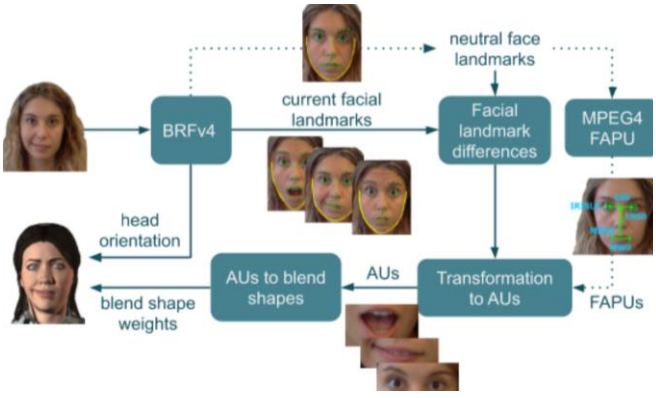

Fig. 2. System overview of the real-time facial retargeting. Dotted lines represent the initialization phase. Straight lines represent the real-time path. The facial landmarks and the head orientation are computed with the BRFv4 library. The AUs are extracted using the facial landmark differences and the FAPUs. The AUs are transformed to blend shape weights in order to modify the facial expression of the virtual character.

[2] Beyond Reality Face, "BRFv4". Available at: https://github.com/Tastenkunst/brfv4_javascript_examples. Accessed July 2019.

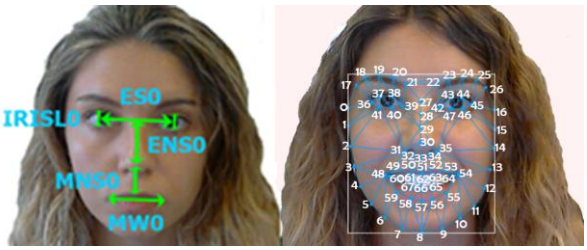

Fig. 3. FAPUs (left) and landmarks (right) provided by the BRFv4.

The BRFv4 library tracks the head orientation and provides 2D facial landmarks at every frame (see Fig 3). A neutral face is pre-initialized in our application. During runtime, the current landmarks and the landmarks of the pre-initialized neutral face are used to calculate landmark differences associated to each Action Unit (AU), as shown in the following formula:

$$D_i = \left| \frac{S_i^c - S_i^n}{N_i} \right|$$

where D_i is the displacement of the i-th AU, S_i^c is the state of the i-th AU of the current expression, S_i^n is the state of the neutral expression and N_i is the normalization factor for the i-th AU. The state of an AU is defined by distances between landmarks associated to the AU (see Table I). The normalization factor is computed using distances between landmarks in the face defined in MPEG-4 as FAPUs (Face Animation Parameter Units) [13] (see Fig. 3 and Table II).

TABLE I. AU'S STATE AND FAPUS DEFINITIONS

AU	State
Mouth Stretch (AU27)	$(dist(61,67) + dist(62,66) + dist(63,65))/3$
Smile (AU12)	Left = $dist(54,33) - dist(42,54)$
	Right = $dist(48,33) - dist(39,48)$
Mouth sideways (AU14)	Left = $dist(54,45) - dist(54,33)$
	Right = $dist(48,36) - dist(48,33)$
Brow Left Up/Down (AU2, AU4)	$(dist(22,42) + dist(23,47) + dist(24,46))/3$
Brow Right Up/Down (AU2, AU4)	$(dist(21,39) + dist(20,40) + dist(19,41))/3$
Eye-Lid Left Open/Closed (AU5, AU43)	$(dist(44,46) + dist(43,47))/2$
Eye-Lid Right Open/Closed (AU5, AU 43)	$(dist(38,40) + dist(37,41))/2$
FAPU	**Definition**
MSN	$dist(33,62)$
MW	$dist(48,54)$
ES	$0.5dist(36,39) + 0.5dist(42,45) + dist(39,42)$
ENS	$dist(27,33)$
IRIS_L	$dist(44,46)$
IRIS_R	$dist(37,41)$

In order to define the displacement range of the AU in this normalized space, the weight of each one is computed in the following form:

$$w_i = \begin{cases} 1, & D_i^{max} \leq D_i \\ \frac{D_i}{D_i^{max}}, & 0 \leq D_i < D_i^{max} \end{cases}$$

where D_i^{max} is the maximum of the i-th AU displacement. The maximum displacement is defined manually (see Table II).

TABLE II. PARAMTERS FOR COMPUTING AUS AND WEIGHTS

AU	N_i (FAPU)	D_i^{max}
Mouth Stretch (AU27)	MNS	1
Smile Right/Left (AU12)	MNS	0.4
Mouth sideways Right/Left (AU14)	MW	0.2
Brow Raiser Right/Left (AU2)	ENS	0.2
Brow Lowerer Right/Left (AU4)	ES	0.1
Eye-Lid Raised Right/Left (AU5)	IRIS	0.6
Eye-Lid Closed Right/Left (AU 43)	IRIS	0.4

The AUs weights are then smoothed by interpolating between frames and then mapped to the blend shapes of our virtual character. The facial expression is defined by a vector of blend shape weights.

We configure manually the transformation weights from AUs to blend shapes and the offsets of the head orientation. Usually each virtual character has a different set of blend shapes and a different neutral head orientation. Thus, this latter step would always be required when setting up the system for a new character. In our virtual character most of the AUs have an associate blend shape with the same name. But, for example, in the case of the *Brow Raiser (AU2)* we combine the blend shapes *Brows Up* and *Eyes Wide* at the same time in order to get the desired movement.

B. Facial Rig With 2D Interpolation

In this section we briefly explain the 2D valence-arousal emotional space to control the character's facial expression and introduce the different interpolation method for the facial rig. In the valence-arousal space we predefine several facial expressions for different positions (see Fig. 4). Some of these

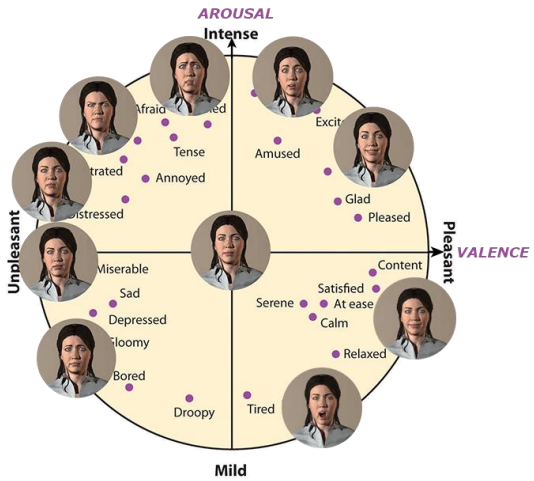

Fig. 4. Valence-Arousal representation within preconfigured expressions.

predefined expressions are based on the work from [9] and [14].

The user can select any point of the 2D space and a facial expression is generated automatically according to the surrounding predefined facial expressions. In order to interpolate between the predefined facial expressions and to generate a facial expression at any given point of the 2D space, we implemented natural neighbor interpolation (NNI) [15] (see Fig. 5). Given a point in a 2D space, this method finds the nearest neighboring points and assigns a normalized weight to each neighbor according to their respective distance.

We implemented two different strategies for NNI. One uses the method proposed by [16, 17], where Voronoi diagrams are created by the rasterization of cones using the GPU. To compute NNI we check the differences of two Voronoi diagrams stored in textures: one with the neighboring points and one with the additional interpolation point (Fig. 5).

The second method uses a bidimensional matrix of 64x64. It stores for every cell of the matrix the influence of each neighboring point. This matrix is only recalculated when a new neighbor is added in the space. In our work the neighboring points are the predefined facial expressions and the interpolation point is the facial expression to generate.

C. Implementation

We have integrated our solution inside WebGLStudio [11] and implemented our facial animation pipeline using a node-based system (Fig. 6). The properties of the scene and objects can be chosen to appear in the graph interface. Snippets of code and functions can be written and shown as nodes with their corresponding inputs and outputs.

We have decided to use this type of approach both because it is an intuitive and easy to connect and develop nodes, and it helps less technical and more artistic profiles to improve our system, and due to the nature of our work we require those skills to be present in our pipeline.

We used Adobe Fuse CC[3] to create the virtual characters. This tool provides characters with 52 automatically generated blend shapes. For a review of free tools to create 3D virtual characters for the web please refer to [10]. In our work the facial retargeting makes use of 13 blend shapes, most of them associated to an AU. Out of these 13 blend shapes, 4 are used to control the eyebrows, 4 are used to control the eyelids and 5 are used to control the mouth. The facial rig uses 29 blend shapes: 8 of them are used to control the eyebrows, 6 are used to control the eyelids, 13 are used to

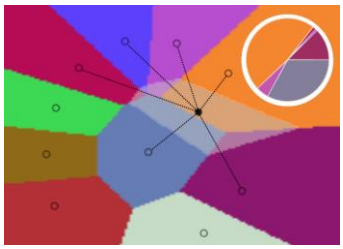

Fig. 5. NNI with a Voronoi diagram. Filled black circle in the center indicates the facial expression to generate. Unfilled black circles indicate the predefined facial expressions. Dashed lines show the relationship between the predefined facial expressions and the desired expression. The pie chart (top-right) shows the influence percentage of each predefined facial expression.

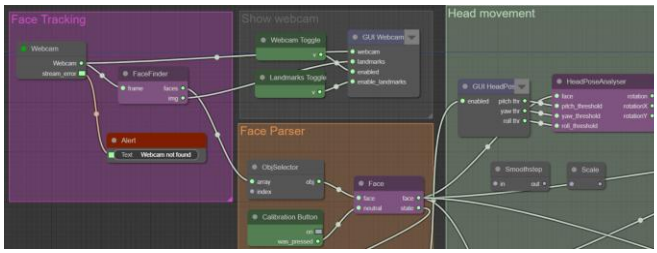

Fig. 6. Screenshot of a section of the facial retargeting graph. The paths and nodes shown are the webcam's input, the webcam GUI with facial landmarks, the extraction of facial features and the head orientation.

control the mouth and 2 are used to control the nose. In our system we additionally integrated a web-based lip-sync algorithm [18] which uses 5 blend shapes for the mouth control.

We exported the character through the Mixamo [4] web platform in FBX format. Our web application only supports the COLLADA format, thus we used Blender [19] to convert from FBX to COLLADA. Although Mixamo supports directly COLLADA, we found some issues with the skinning of the character and used FBX instead.

III. Evaluation

We evaluated the precision of the facial retargeting system by testing the system with 8 participants. They were instructed to test the system and try different facial expressions for approximately one minute. We used OpenFace 2.1.0 [20, 21] to extract the Facial Action Units of the videos of each actor and its corresponding animated virtual character.

We computed the mean-squared error (MSE) of each AUs (see Fig. 7). The AUs that had more error are related to the eyebrows (*AU4 - Brow Lowerer*, *AU1 - Inner Brow Raiser*) and the eyelids (*AU7 - Lid Tightener*). Other AUs with relevant MSEs were AUs 25 and 26, which represent a similar action (Lips part and Jaw Drop), and AUs 15 and 17 (*Lip Corner Depressor* and *Chin Raiser*). In the case of AUs 25 and 26, they were correlated and most of the error came from the confusion between them.

IV. Results and Discussion

An evaluation with experienced and inexperienced animators should be conducted in order to get a better assessment of the tools provided in this work. Although we did not conduct a user experiment with animators, we observed that the facial retargeting and the facial rig provide a fast method to create facial animations. In this section we explain the precision, advantages, disadvantages and possible improvements of the tools presented.

The evaluation with OpenFace 2.1.0 should be taken with care, as the system is not trained for tracking AUs in virtual characters. Additionally, the relationship between AUs and blend shapes in the virtual character was done manually without any optimization process. Nevertheless, we expected to get a coarse idea of the performance of the system with this evaluation. For example, the eyebrows' error could be due to the fact that our virtual character does not have wrinkles, which appear naturally when frowning.

[4] Adobe, "Mixamo". Available at: https://www.mixamo.com/. Accessed July 2019.

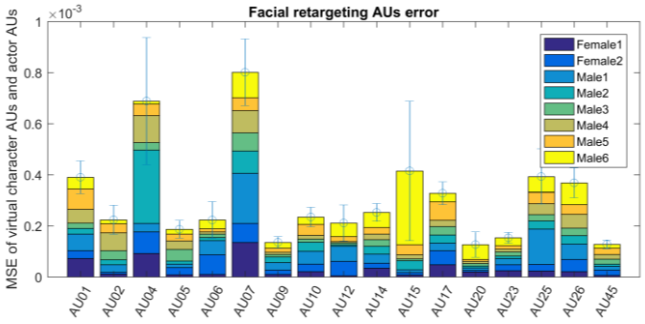

Fig. 7. MSE of each AU. Standard error is shown on top of each bar as blue thin lines. The error means are shown as blue circles. The divisions inside the columns represent the error percentage of each participant.

The facial retargeting errors could also be caused by the landmark detection of the BRFv4 library. We noticed that when the mouth or the eyes are closed the markers are not close enough together and that there are errors on the mouth corners' detection. which was confirmed by the evaluation with OpenFace 2.1.0. When blinking, the eyelids did not open and close symmetrically, which created a bizarre effect. Currently we opted for automatically generated blinks in our applications.

The manual transformation from AUs to blend shapes could also be improved and automated using OpenFace 2.1.0 or similar software. The AUs weights extracted from the webcam image and those of the generated image of the character could be compared and used to reduce and optimize the error by changing the transformation function.

Our facial retargeting system uses 2D landmarks instead of 3D landmarks as in [12]. This difference should affect the performance of our system. Additionally, we do not take into account head rotation in the landmark distances, thus when the head is rotated the AU's weights change with the same facial expression.

Although the BRFv4 library is the best implementation in the web we have found, we noticed some noise in the mouth and eyelids landmarks. Thus, we believe that in the incoming years the precision of the face detection and tracking libraries on the web could be improved and it will achieve the accuracy levels of the offline SDKs. As it is improved, the errors of our facial retargeting system will be greatly reduced. A further addition to our system would be gaze retargeting, which is missing in our application.

Regarding the facial rig, one of its advantages is that it only requires two values to generate a facial expression. This facial rig (without NNI) was previously used in [2], where valence and arousal values were given to generate facial expressions in an embodied conversational agent. With NNI the facial rig is not limited to the valence-arousal space. Because the 2D space of the facial rig is customizable, we believe that it has potential for animating not just emotions, but actions. For example, the action of chewing could be characterized with four extreme facial expressions: jaw open, jaw clenched, jaw to the left, jaw to the right. In the 2D space one could find infinite combinations of these four facial expressions and thus create varied animations by just dragging a point in a 2D space. An animator's workspace could contain several facial rig configurations for different actions and emotions.

Our tools are implemented inside WebGLStudio [11], a web-based open-source 3D authoring tool. The software not only provides libraries for rendering with WebGL, but also an interface to create scenes and develop new web components. Thanks to this interface, we are able to test different virtual characters quite easily and to modify and integrate our tools (see Fig. 1). Moreover, this platform has a graph component interface. By only changing the links between the graph nodes, virtual characters with different facial features can be integrated.

Although it is relatively easy to exchange virtual characters using the graph interface, the transformation from AUs to blend shapes, bones and the facial features of the virtual character is always required. Only when using software that generates virtual characters with the same facial parameters, this transformation process has to be done only once, as the faces should have the same properties.

V. Conclusions

In this work we presented a facial retargeting system and a new interpolation function for a facial rig. We demonstrated that it is possible to implement such technologies in the web browser with a web-cam and to use these tools with different virtual characters. Further evaluations with users and animators should be conducted.

Our application not only supports these technologies, but it is implemented in an open source platform, giving access to others to use and modify the tools. We believe that the methods presented here will open new possibilities for facial animation on the web and will improve the animation pipelines, thus permitting higher quality web-based interactive virtual agents.

Acknowledgments

This work was partially funded by the EU's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement 675324 (ENRICH), and is also related to the projects KRISTINA (645012) and SAUCE (780470). We would like to show our gratitude to Sergio Sayago for the interest and the contributions made in this work.

References

[1] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, and J. Blat. "3D graphics on the web: A survey." Computers & Graphics 41 (2014): 43-61.

[2] L. Wanner, E. André, J. Blat, S. Dasiopoulou, M. Farrús, T. Fraga, E. Kamateri, F. Lingenfelser, G. Llorach, O. Martínez and G. Meditskos. "Kristina: A knowledge-based virtual conversation agent." In International conference on practical applications of agents and multi-agent systems, pp. 284-295. Springer, Cham, 2017.

[3] M. Valstar, T. Baur, A. Cafaro, A. Ghitulescu, B. Potard, J. Wagner, E. André et al. "Ask Alice: an artificial retrieval of information agent." In Proceedings of the 18th ACM International Conference on Multimodal Interaction, pp. 419-420. ACM, 2016.

[4] R. Ekman. What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS). Oxford University Press, USA, 1997.

[5] GR. Leone and P. Cosi. "LUCIA-webGL: a web based Italian MPEG-4 talking head." In Auditory-Visual Speech Processing 2011. 2011.

[6] RZ. Buda, G. Boldizsár, A. Tóth, S. Szeghalmy, R. Tornai and R. Kunkli. "Extended capabilities for a WebGL based talking head system." In 2014 5th IEEE Conference on Cognitive Infocommunications (CogInfoCom), pp. 459-459. IEEE, 2014.

[7] H.V. Diez, S. García, J.R. Sánchez, and M. del Puy Carretero. "3D animated agent for tutoring based on WebGL." In Proceedings of the 18th International Conference on 3D Web Technology, pp. 129-134. ACM, 2013.

[8] G. Llorach and J. Blat. "Say Hi to Eliza. " In International Conference on Intelligent Virtual Agents, pp. 255-258. Springer, Cham, 2017.

[9] M. Romeo . Automated processes and intelligent tools in CG media production. Diss. Universitat Pompeu Fabra, 2016.

[10] G. Llorach, J. Agenjo, J. Blat and S. Sayago. "Web-Based Embodied Conversational Agents and Older People." In Perspectives on Human-Computer Interaction Research with Older People, pp. 119-135. Springer, Cham, 2019.

[11] J. Agenjo, A.Evans and J. Blat. "WebGLStudio: a pipeline for WebGL scene creation." In Proceedings of the 18th International Conference on 3D Web Technology, pp. 79-82. ACM, 2013.

[12] Sheng G, "Archived - Intel® RealSense™ SDK-Based Real-Time Face Tracking and Animation," In Intel Developer Zone. Available at: https://software.intel.com/en-us/articles/intel-realsense-sdk-based-real-time-face-tracking-and-animation. Accessed July 2019.

[13] M. Escher, I. Pandzic and N.M. Thalmann. "Facial deformations for MPEG-4." In Proceedings Computer Animation'98 (Cat. No. 98EX169), pp. 56-62. IEEE, 1998.

[14] D. Matsumoto and P. Ekman, 2008. Facial expression analysis. Scholarpedia, 3(5), p.4237.

[15] R. Sibson. "A brief description of natural neighbour interpolation." Interpreting multivariate data (1981).

[16] Hoff III, E. Kenneth, J. Keyser, M. Lin, D. Manocha and T. Culver. "Fast computation of generalized Voronoi diagrams using graphics hardware." In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 277-286. ACM Press/Addison-Wesley Publishing Co., 1999.

[17] A. Beutel, T. Mølhave and P.K. Agarwal. "Natural neighbor interpolation based grid DEM construction using a GPU." In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 172-181. ACM, 2010.

[18] G. Llorach, A. Evans, J. Blat, G. Grimm and V. Hohmann. "Web-based live speech-driven lip-sync." In 2016 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), pp. 1-4. IEEE, 2016.

[19] T. Roosendaal . "Blender." Blender Foundation. Available at: https://www.blender.org. Accessed July 2019.

[20] T. Baltrusaitis, A. Zadeh, YC. Lim and L. Morency. "Openface 2.0: Facial behavior analysis toolkit." In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 59-66. IEEE, 2018.

[21] T. Baltrušaitis, M. Mahmoud and P. Robinson. "Cross-dataset learning and person-specific normalisation for automatic action unit detection." In 2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), vol. 6, pp. 1-6. IEEE, 2015.