Face-capture with automatic blendshape generation

João Vitor Nogueira*, Bruno Sumar†, Leonardo Carvalho‡,
Universidade Federal Fluminense
Instituto de Ciência e Tecnologia, Departamento de Ciência da Computação
Rio das Ostras, RJ
* jv_nogueira@id.uff.br, † bsumar@id.uff.br, ‡ leonardooc@id.uff.br

***Abstract*—The goal of this paper is the development of a system for the production of performance-driven facial animation that automatically generates a blendshape model from input video frames. This simplifies the production of this kind of model to be used in animation projects. The proposed method is designed to be used without the need for expensive hardware, such that any computer with a webcam can run the system.**

I. Introduction

A great challenge in the field of Computer Graphics is the production of digital characters whose facial movements accurately resemble reality, due to the face's very fine details, and the complexity to represent those.

An efficient technique to control digital faces is Performance-Driven Facial Animation [1]. This method uses face-capture information to calculate facial movements. It can be done densely, through a video sequence, or sparsely, through markers on a captured face, which are used to track movements for the algorithm.

The capture process can be done through a few different ways. One method uses a helmet with cameras worn by an actor, which records their face. This model yields high-quality results and can be done through multiple angles reducing facial obstructions. This is highly used by large-scale productions, given its need for dedicated hardware. However, such equipment is not always readily available, hence simpler methods are needed.

Another approach is to use stationary cameras. It makes the process more accessible, but more computationally demanding, as it requires processing the face's relative position. It also has lower accuracy, given its susceptibility to facial obstructions.

Blendshapes are a simple technique to deform meshes, based on the combination of linear transformations applied to each vertex. A model with high-quality blendshapes yields some of the best results in facial animations, as it grants a great degree of control over fine details. However, this method is very labor intensive, as each transformation has to be defined manually, usually by a 3D artist.

This article proposes a method that automatically generates and fits a blendshape model to a person's facial expression given a video sequence. The model evolves on-the-fly, creating new blendshapes automatically when necessary. It reduces significantly the work required in the production of this kind of model. It also uses video input from a stationary camera for greater accessibility. The proposed method was made into an open-source add-on for Blender, a free 3D modeling software, to help those without access to state-of-the-art technology in their animation endeavors.

II. Related Work

There is a large number of works on face modeling and capture using different approaches. It is beyond the scope of this article to make an exhaustive list of those already published. An overview of several methods developed in this area can be found in [2]–[4]. For a blendshape-specific approach, [5] goes over several methods.

In recent times, several works have been published with the goal of obtaining real-time face capture without the use of facial markers using RGB [6]–[9], and RGB-D cameras [10]–[14] In this article, RGB cameras were used, as the project's goal is to be as accessible as possible.

There is also a vast number of works on performance-driven facial animation, where a person's face is tracked, then its movements and expressions are transmitted to a virtual model. Some of those use Three-dimensional capture data [15], [16] and some others use standard video data [17]–[19]. There are also variations depending on the type of model used, those being principal component analysis [20], blendshapes [1], [17], [18], [21], and a combination of both [16]. In this work, the data was collected from real-time recordings and applied to a blendshape model.

Furthermore, many commercial 3D modeling software have facial animation systems, usually as third-party add-ons that extend the base program's functionality, such as Face Machine Nodes [22] and Facial Animation Toolset [23], for Maya and BonyFace [24] for 3dsMax. Blender is a free alternative to those programs and the chosen platform for implementing and testing the proposed method.

III. Facial Animation With Blendshapes

A blendshape facial model consists on a base model and a set of n facial expressions, where each expression is represented by a blendshape. The weighted combination of these expressions results in the final model. Figure 1 is an example of a blendshape facial model, where the upper left expression is the basis (neutral), and the others correspond to different expressions available for this model, which are slight variations on said basis.

Figure 1. Facial model using blendshapes.

The blendshapes are combined through the following expression:

$$S = B_0 + \sum_{i=1}^{n} \alpha_i B_i,$$

where B_0 corresponds to the base blendshape vertex group, B_i to ith blendshape vertex group. α_i is a weight factor to control the influence of B_i on the final blendshape vertex group S. The values of α can be defined manually or automatically calculated to arrive at the desired expression.

Controlling said alpha values is a very time-consuming task, hence performance-driven animation methods come into play, capturing a subject's face and automatically adjusting the model's blendshape weights to it. This can be done either in real-time or with a pre-recorded video sequence.

Then, those weights go through the process of retargeting, a transformation of expressions from the original model into another model with a corresponding set of blendshapes. This technique allows the facial fitting to occur using a mesh more similar to the recorded face, thus improving the method's accuracy. Then the results are transmitted to the desired mesh, which can differ greatly from the original face.

IV. Method

The method takes frames from a video sequence as input, obtained either from a real-time camera or a pre-recorded video. Then, it searches for a face in the frame and infers its geometry. That data is used to create a model that fits the face. Finally, the model can be retargeted to a final model. Figure 2 shows the pipeline of the method.

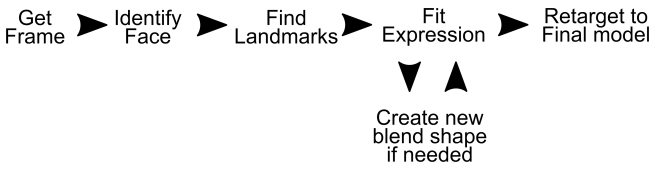

Figure 2. Pipeline of the method.

The first step consists in identifying a face in the frame. This can be done by using Computer Vision and Machine Learning algorithms. In this work it was done with the library MediaPipe [25], using the modules *Face Mesh* and *Face Geometry*, that identifies and obtains the geometric data for a face in a markerless video frame. The result is a set of points called landmarks distributed on the face. These landmarks are given a coordinate system relative to the frame, but using *Face Geometry* module it is also possible to infer 3D coordinates in the real world, given in centimeters, with the face centered at the origin and facing the z-direction, such that, regardless of the subject's distance to the camera, the size of the captured face remains constant. Figure 3 shows the landmark extraction from two frames, and the corresponding 3d mesh obtained from these landmarks.

Figure 3. Landmarks inferred in 2d and 3d mesh.

The landmarks are then used to fit a blendshape model, where the weights α_i can be obtained by solving the following least squares problem: where F_i is the 3d position of i-th landmark, B_{ji} is the position of the i-th vertex in the j-th blendshape, and α_j is the weight related to this blendshape. This minimization problem finds the weights that make the blendshape model as close as possible to the given landmarks.

The blendshape mesh is created on-the-fly during the capture session. The user defines the base mesh, ideally using a neutral expression. New blendshapes can be added manually using the positions of the given landmarks. But it can also be automatically created: if the minimization error is greater than a predetermined threshold, then the current blendshape model does not have enough shapes to reproduce accurately the given expression, in this case, a new blendshape is produced. Figure 4 shows an example of blendshape generation.

To isolate different parts of the face, it is possible to work with vertex groups, such that when a blendshape needs to be created, it is split for each of the groups. For example, defining a group that affects only vertices around the left eye and another group that affects vertices around the right eye, allows the system to create blendshapes that register the movements of each eye separately. Each blendshape j is associated with one vertex group,and every vertex i has a weight w_{ji} associated with this group, such that the position of the corresponding vertex in the blendshape model is given

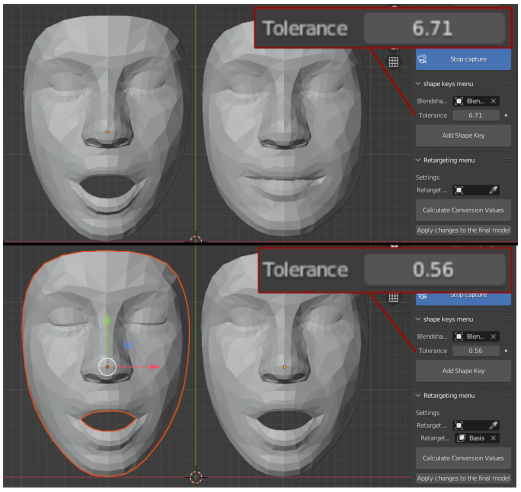

Figure 4. Automatically generated blendshapes. The top half showcases what happens when the threshold is too high. The bottom half showcases that when the tolerance is lowered and a new blendshape is created.

by

$$B_{0i} + \sum_{j=1}^{n} \alpha_j w_{ji} B_{ji}.$$

The minimization problem can be updated to use these vertex weights. Figure 5 shows a set of user-defined vertex groups.

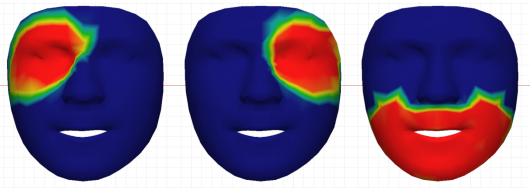

Figure 5. User-defined vertex groups that split the face into three separate regions.

It is also possible to retarget those expressions to another model; calculating an equivalent expression on the target mesh. This is a necessary step, as the desired final model isn't the same used for the capture fitting. This task requires user input, as models and their blendshape sets can vary greatly. Users have to define blendshape equivalencies between models, by selecting one blendshape from the source mesh and a set of weighted blendshapes from the target-mesh, so that both models have an analogous expression. This creates a set of conversion weights α between blendshapes of both models. The blendshape weights of the final model are computed as follows:

$$Bt_i = \sum_{j=1}^{n} \alpha_{ij} Bs_j$$

Where Bt_i is the ith target-mesh blendshape and Bs_j is the jth source-mesh blendshape, and α_{ij} is the conversion from source blendshape j to target blendshape i.

V. Results

Figure 6 shows a face-capture session with a blendshape model that was automatically generated from the proposed method. The method created a set of 12 blendshapes (not shown in the figure).

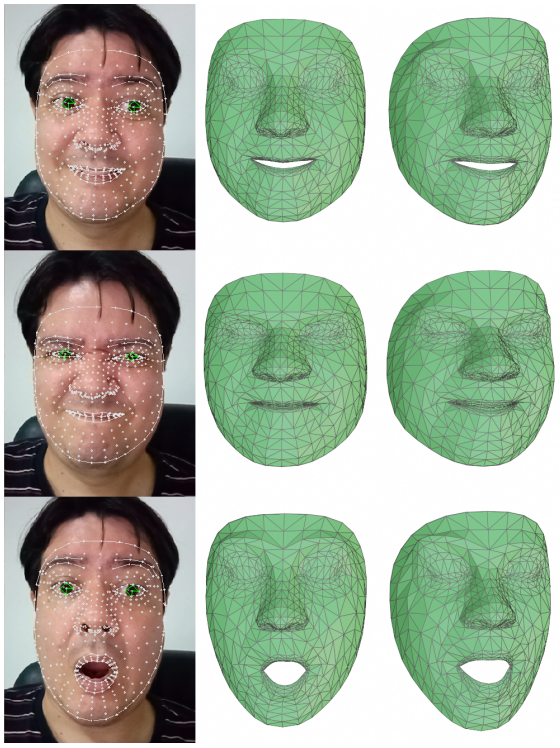

Figure 6. Result of face-capture with an automatically generated blendshape model.

Figure 7 shows the base mesh and four blendshapes that were automatically generated. Figure 8 shows two frames of a face-capture session using a retarget mesh. Figure 9 showcases the retargeting process to a simple mesh.

Figure 7. Set of automatically generated blendshapes. The base mesh is on the left.

Figure 8. Blendshape model with retarget mesh.

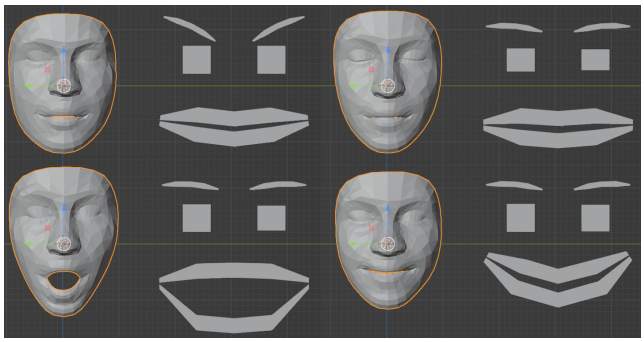

Figure 9. Retargeting to a simple model.

Figure 10 shows how fitting error changes along time. Every time the error goes over the threshold, a new blendshape is added to the model, cause a substantial error reduction on the next frames. After some time, when there is enough blendshapes, the error is always below the threshold.

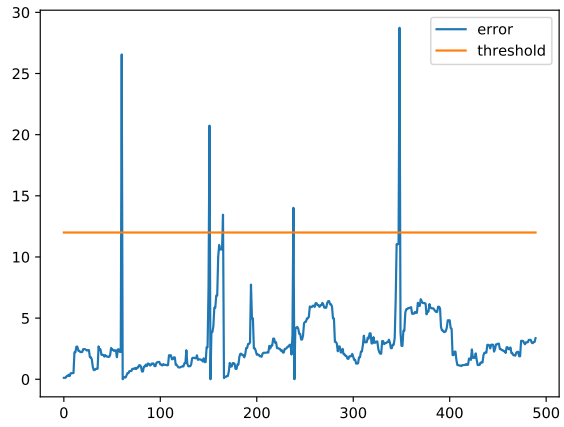

Figure 10. Error evolution during capture.

Lastly, Figure 11, shows the distance between equivalent vertices in the captured and blendshape meshes. The left column shows neutral expressions, close to the base mesh, while the right one shows extreme ones. The first row shows the results given a tolerance of 2, the second with a tolerance of 10, and the bottom one 50. It was empirically determined that intermediate tolerance values yield the best results.

Figure 11. Blendshape ajustment error. Each vertex is painted in a colored scale, where dark blue is an error of 0 cm and bright red of 1 cm or more.

VI. Conclusion

This work proposes a technique for the creation of performance-driven facial animations with automatic generation of a blendshape model from marker-less input video frames. The method uses Computer Vision methods to extract landmark points on the face's surface and fits them in a blendshape model, which is incrementally improved whenever an expression that can not be accurately represented is found.

Despite showing promising results the system still has some limitations. As the number of blendshapes, increases some errors compound and become more noticeable. Such errors are pretty noticeable in the lip areas, that are almost always gaping. Figure11 showcases this phenomenon, the row with the smallest tolerance had worse results than the one with and intermediate tolerance.

Another problem that stems from the number of blendshapes, is how cumbersome the manual adjustments required for retargeting become, as each blendshape is ajusted manually.

Lastly, the method is computationally costly, it's hard to do it at a high frame rate. In our experiments, the system performed at around 20 frames per second in an 8-core AMD Ryzen 5 processor. Most of the processing time (around 98%) is due to the frame processing and landmark extraction done by Mediapipe. The proposed blendshape fitting method uses only 2% of processing time.

For future works, we plan on including gaze tracking. The user interface can be refined to streamline the usability. Finally, the retargeting process needs and overhaul. Rather than having to map blendshapes one by one, it would be better if it were possible to match whole expressions with many blendshapes.

This could be done by solving a least squares problem for the values of α_{ij}.

REFERENCES

[1] L. Williams, "Performance-driven facial animation," in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '90. New York, NY, USA: Association for Computing Machinery, 1990, p. 235–242. [Online]. Available: https://doi.org/10.1145/97879.97906

[2] Z. Deng and J. Noh, *Computer Facial Animation: A Survey*. London: Springer London, 2008, pp. 1–28. [Online]. Available: https://doi.org/10.1007/978-1-84628-907-1_1

[3] V. Orvalho, P. Bastos, F. Parke, B. Oliveira, and X. Alvarez, "A Facial Rigging Survey," in *Eurographics 2012 - State of the Art Reports*, M.-P. Cani and F. Ganovelli, Eds. The Eurographics Association, 2012.

[4] F. I. Parke and K. Waters, *Computer Facial Animation*. A K Peters, 2008.

[5] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, "Practice and Theory of Blendshape Facial Models," in *Eurographics 2014 - State of the Art Reports*, S. Lefebvre and M. Spagnuolo, Eds. The Eurographics Association, 2014.

[6] C. Cao, Y. Weng, S. Lin, and K. Zhou, "3d shape regression for real-time facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 41:1–41:10, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2462012

[7] C. Cao, D. Bradley, K. Zhou, and T. Beeler, "Real-time high-fidelity facial performance capture," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 46:1–46:9, Jul. 2015. [Online]. Available: http://doi.acm.org/10.1145/2766943

[8] Y. Weng, C. Cao, Q. Hou, and K. Zhou, "Real-time facial animation on mobile devices," *Graphical Models*, vol. 76, no. 3, pp. 172 – 179, 2014, computational Visual Media Conference 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1524070313000295

[9] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Demo of face2face: Real-time face capture and reenactment of rgb videos," in *ACM SIGGRAPH 2016 Emerging Technologies*, ser. SIGGRAPH '16. New York, NY, USA: ACM, 2016, pp. 5:1–5:2. [Online]. Available: http://doi.acm.org/10.1145/2929464.2929475

[10] T. Weise, S. Bouaziz, H. Li, and M. Pauly, "Realtime performance-based facial animation," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 77:1–77:10, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964972

[11] Y.-L. Chen, H.-T. Wu, F. Shi, X. Tong, and J. Chai, "Accurate and robust 3d facial capture using a single rgbd camera." in *ICCV*. IEEE Computer Society, 2013, pp. 3615–3622. [Online]. Available: http://dblp.uni-trier.de/db/conf/iccv/iccv2013.html#ChenWSTC13

[12] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 42:1–42:10, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2462019

[13] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 40:1–40:10, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461976

[14] P.-L. Hsieh, C. Ma, J. Yu, and H. Li, "Unconstrained realtime facial performance capture." in *CVPR*. IEEE Computer Society, 2015, pp. 1675–1683. [Online]. Available: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#HsiehMYL15

[15] B. Choe, H. Lee, and H. Ko, "Performance-driven muscle-based facial animation," *Comput. Animat. Virtual Worlds*, vol. 12, pp. 67–79, 2001.

[16] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann, "Animating blendshape faces by cross-mapping motion capture data," in *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, ser. I3D '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 43–48. [Online]. Available: https://doi.org/10.1145/1111411.1111419

[17] F. Pighin, R. Szeliski, and D. Salesin, "Resynthesizing facial animation through 3d model-based tracking," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, 1999, pp. 143–150 vol.1.

[18] B. Choe and H.-S. Ko, "Analysis and synthesis of facial expressions with hand-generated muscle actuation basis," in *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No.01TH8596)*, 2001, pp. 12–19.

[19] T. Rhee, Y. Hwang, J. D. Kim, and C. Kim, "Real-time facial animation from live video tracking," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 215–224. [Online]. Available: https://doi.org/10.1145/2019406.2019435

[20] V. Blanz, C. Basso, T. Poggio, and T. Vetter, "Reanimating faces in images and video," *Comput. Graph. Forum*, vol. 22, pp. 641–650, 09 2003.

[21] E. Chuang, F. Deshpande, and C. Bregler, "Facial expression space learning," in *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, 2002, pp. 68–76.

[22] Anzovin. (2012) Face machine nodes. [Online]. Available: https://anzovin.squarespace.com/tfmn

[23] Animationsinstitut. (2012) Facial animation toolset. [Online]. Available: https://animationsinstitut.de/en/research/tools/facial-animation-toolset

[24] Scriptattack. (2012) Bony face: Facial animation system for 3dsmax. [Online]. Available: http://www.scriptattack.com/maxscripts/bonyface/index_eng.html

[25] Google. (2020) Mediapipe. [Online]. Available: https://mediapipe.dev