# A modular framework for performance-based facial animation

Carlos Eduardo Rossi Cubas da Silva
*Sao Paulo State University (UNESP)*
Bauru, Brazil
carlos.cubas@gmail.com

Antonio Carlos Sementille
*Department of Computing*
*Sao Paulo State University (UNESP)*
Bauru, Brazil
semente@fc.unesp.br

***Abstract*—In recent decades, interest in capturing human face movements and identifying expressions for the purpose of generating realistic facial animations has increased in both the scientific community and the entertainment industry. We present a modular framework for testing algorithms used in performance-based facial animation. The framework includes the modules used in pipelines found in the literature as a module for creating datasets of blendshapes which are, facial models, where the vectors represent individual facial expressions, an algorithm processing module for identification of weights and, finally, a redirection module that creates a virtual face based on blendshapes. The framework uses a RGB-D camera, the RealSense F200 camera from Intel.**

***Index Terms*—Capture facial movements, Blendshapes, retargeting.**

## I. Introduction

A very active field of research in the area of Computer Graphics is the generation of models of the human face aiming at the creation of realistic facial animations. There are several applications that can benefit from advances in this field, such as: movies for film and television, videogames, videoconferencing using avatars, facial surgery planning and others. In the animation of virtual characters, the accurate reproduction of facial movements is critically important, since it is one of the main sources of emotional information.

According to Fratarcangeli [8], the complexity and sophistication of human head structure increases the difficulty of reproducing a convincing facial animation. High accuracy is required because humans are trained to observe and decode facial expressions from birth, making them experts in detecting small errors in virtual face animation.

The techniques of facial animations can be directed to speech [9] or directed to performance [25], [26].

Performance-based animation systems typically consist of a facial performance capture module and a motion transfer module. To capture facial performance, several systems use multiple cameras and a large number of facial markers on the actors. Although they achieve good results, the use of these markers may not be practical, as well as intrusive to the actors. In addition, such systems typically require a lot of manual intervention [15].

A key trade-off in all systems is the relationship between the quality of the data acquired and the complexity of the configuration of the acquisition. At one end of the spectrum are systems designed for maximum accuracy that lead to impressive virtual avatars, suitable for film production. Because of their robustness, marker-based techniques are widely used for real-time facial animation and often provide enough motion parameters for a compelling redirection to nonhuman creatures or video game characters. At the other end is the realistic scanning of human faces.

For realistic human-face scanning, approaches that do not use markers like real-time 3D scanners are generally more advantageous because of their ability to capture fine-scale dynamics (for example, wrinkles and folds). All of these methods involve highly specialized sensors and / or a controlled studio environment. However, the recent availability of equipment with low-cost but good depth-resolution RGB-D cameras has changed this scenario, making it possible to create environments for the average user.

Since capturing facial movements and performance-based animation are areas of active research in recent years, there are a large number of different processing systems and pipelines that share many fundamental principles as well as specific implementation detail.

We propose, in this work, to create and validate an environment with modular architecture for performance-based facial animation that uses, as a way of capturing the facial movements of an actor, an RGB-D camera, as well as allowing the incorporation of different tracking algorithms. The environment performs the transfer of facial expressions from one user to a different human face model. The generic pipeline of this environment is based on the use of blendshapes, to obtain generic and user-specific expression models, and to animate the output virtual face model (target redirection).

## II. Related Work

Performance-based facial animation, also known as retargeting, introduces the idea of capturing the face of a real actor and its redirection to a virtual actor. This transfer can be applied to a 3D animation created manually by an artist [5].

For Li, Sun, Hu, Zang, Wang and Zhang [13] performance-oriented facial animation refers to the problem of realistically mapping an actor's facial expressions to a digital avatar and compatible with captured performance. It usually consists of a facial tracking step followed by an expression synthesis procedure.

According to Dutreve, Ludovic and Meyer [5], performance-driven facial animation is found in applications such as 3D gaming, man-machine interaction, and the film industry. Movies such as *King Kong* and *Avatar* are examples of the use of facial animation directed at performance [20].

The method presented by Li, Yu, Ye and Bregler [14], shows the face captured through Kinect a device developed by Microsoft which, in addition to capturing images in RGB, also captures the depth images. RGB-D images and 2D video are used in real-time. There is a pre-processing step in its pipeline, done offline, which is the adjustment of the dataset and the capture of the initial data as the neutral face of the actor, and the online process, done in real-time, which adjusts the capture of the actor to the virtual face. In both processes, some algorithms and methods for knitting and redirection are used.

The method is initiated by a face capture in the neutral position. The RGB-D information and 40 automatically captured facial markers that correspond to the points around the mouth and eyes are used.

In [13] a real-time redirection method using Kinect and the 3DS Max modeling software is presented.

The redirection takes place between the real-time capture of a face and a virtual avatar implemented in the 3DS Max software. The transfer of information between the input capture and the avatar takes place through the *MIDI (Musical instrument device interface)* protocol, which is a standard protocol of the music industry [24].

In this method the AAM *(Active Appearance Model)* is used, responsible for face tracking and the extraction of facial points [27].

The method presented by Behrens, Al-Hamadi, Redweik and Niese [1] proposes an automatic system for controlling expressions in a digital avatar in real-time. Face capture is done through Kinect and uses RGB-D images and 2D videos. It has an offline pre-processing stage, which is the generation of the dataset of blendshapes, and an online step, where the processing and calculation of the weights for the generation of the virtual face occurs.

The online process is responsible for the normalization of the captured face, removal of the background and the frontal alignment of the face, through the use of the ICP method, to adjust the captured mesh and lighting.

Through the use of 3D modeling software, a virtual avatar base is generated. This avatar uses blendshapes that are changed to form new custom templates for each user.

Weise, Bouaziz, Li and Pauly [23] have created a real-time face tracking method that uses Kinect. It combines 3D point cloud mapping with 2D markers to generate, from a sequence of animations, a new face through the use of blendshapes.

The goal of this method, according to the authors, is to be flexible and use ease of capturing, with inexpensive cameras and existing animations, created by more complex captures while maintaining the final quality of the process. It has an offline process where a dataset of blendshapes is created with the face of the actor to be captured and an online phase for processing the expressions.

## III. Blendshapes

The term Blendshape was introduced by the printing industry in the 1980s when it became popular in commercial software. [12] defined blendshapes as being facial models where vectors represent individual facial expressions. This consists of creating face poses in various meshes. Each mesh is assigned a shape. One of the meshes is the base shape while the other meshes are called target shapes. The difference between the base form and the target shape is represented by configuration vectors [17]. Many applications used by the animation industry use the technique of changing the forms use of implemented blendshapes (Figure 1).

Fig. 1. Forms of blendshapes.

### A. Algebra and Algorithms

As is shown in [12], blendshapes are a simple vector sum. Consider a facial model composed of $n = 100$ blendshapes, each having $p = 10000$ vertices, with each vertex with three components $x$, $y$, $z$. The blendshape model is expressed by the following equation:

$$f = \sum_{k=0}^{n} w_k b_k \tag{1}$$

or, in matrix notation

$$f = B_W \tag{2}$$

where $f$ is the resulting face, in the form of a $30000x1$ vector, $B$ is a $m = 30000x100$ matrix (m = 3p) where each column of the vector $b_k$, corresponds to an individual blendshape (30000 x 1 vectors) and $w$ is the applied weight (a 100 x 1 vector) in each shape.

A face $b_0$, typically in rest expression, is designated as the neutral face, and the remaining faces, $b_k$, $K = 1...N$ are replaced by the difference between $b_k - b_0$ between the nth $K$ target faces and the neutral face:

$$f = b_0 + \sum_{k=1}^{n} W_k(b_k - b_0) \tag{3}$$

where $b_0$ is the neutral form. This is denoted as:

$$f = b_0 + B_w \tag{4}$$

In this formulation, the weights $w$ are limited to between $[0, 1]$.

## IV. System Overview

According to Weise, Bouaziz, Li and Pauly [22], performance-oriented facial animation involves two technical challenges: the first is to accurately track the rigid and non-rigid movements of the user's face; the second is to map the tracking parameters to the appropriate controls that will direct the face animation of the virtual character. One approach, found in several methods, is to combine these two problems into a single optimization that finds the most appropriate values of the most likely parameters of a specific expression model based on the observed 2D and 3D data. To define realistic facial space, it is common to derive an initial probability for this optimization from prerecorded animation sequences.

For the creation of the system, the modular architecture illustrated in Figure 2 was adopted. The diversity of the techniques used for the creation of transfer coefficients can be seen. In some cases, two techniques are employed.

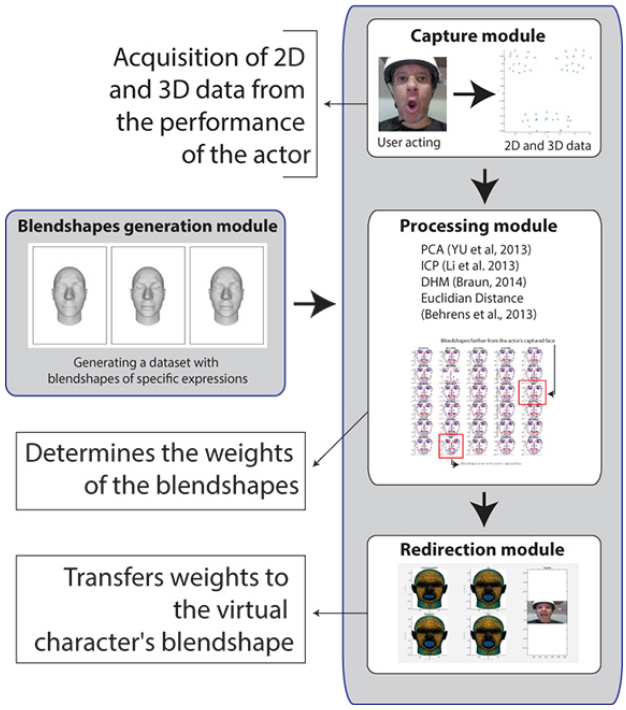


Fig. 2. Architecture and *pipeline* of the developed system.

The system is divided into two subsystems: dataset generation of blendshapes and capture and processing and redirection. The first subsystem generates a dataset of blendshapes with specific expressions for creating realistic animations.

The second subsystem is composed of an Actor Capture Module where the 2D and 3D information will be extracted, which will be processed by the Processing Module, and finally by the Redirection Module, where the facial expression weights are transferred to the blendshape, which represents the facial model of the virtual character.

## V. Implementation of the system prototype

### A. Blendshapes dataset generation subsystem

This subsystem creates a blendshapes database with various types of expressions based on FACS [6] that will be used in redirection. Figure 3 shows the steps required to configure the dataset. The adjustment of the facial markers captured in the 2D image, to the 3D model starts with the normalization of the 3D model and the 2D image points. This normalization is made by a scale adjustment. To adjust the scale, the points corresponding to both eyes are selected manually in the 3D model. In the image captured by the RealSense camera [11], these points are known because the camera automatically marks those points. After highlighting these references between the 3D model and the 2D points, a scale adjustment is made using the distance between the two points and the difference in size between them. At the end of this process, all points are translated using the point of the 3D model closest to the camera, which is the tip of the nose, and the corresponding point of the 2D model, which is reported by the camera. As a database, the blendshapes generated by the application FaceGen Modeller 3.3 [19] were used.

A set of blendshapes generated by FaceGen Modeller 3.3 software [19], is a commercial tool designed for the creation of 3D faces in a realistic way, often used in virtual games. It is based on a database with thousands of human faces scanned in 3D. The created faces vary in gender, age and ethnicity and can be altered through the software manipulation interface.

### B. Capture, processing and redirection subsystem

This subsystem uses the RealSense camera for video capture, the position of the face on the image, and the facial markers and their 3D information. This process is carried out frame by frame. In Figure 4 the implemented pipeline in this subsystem is shown.

#### *Capture*

The first step in the pipeline is capturing the face of an actor using the RealSense camera. A helmet has been developed that ensures the distance between the actor's face and the camera remain unchanged. The camera remains in a stationary position with respect to the movements of the actor's head. Figure 5.

When initiating the capture phase, through software, the actor must stand in front of the camera with a neutral expression during the first few seconds. After this time, the actor can start acting. For the processing step, which will be described below, the first frame captured as the neutral expression is used. It is through this table that the calibration is performed.

In the capture, the 2D and 3D information of the facial markers will be exported and used in the next phases of the pipeline.

#### *Processing*

In this step, the weights of the blendshapes are calculated for each captured frame. This will involve a series of steps to adjust the points before they are submitted to pattern recognition algorithms. The first step of the processing is

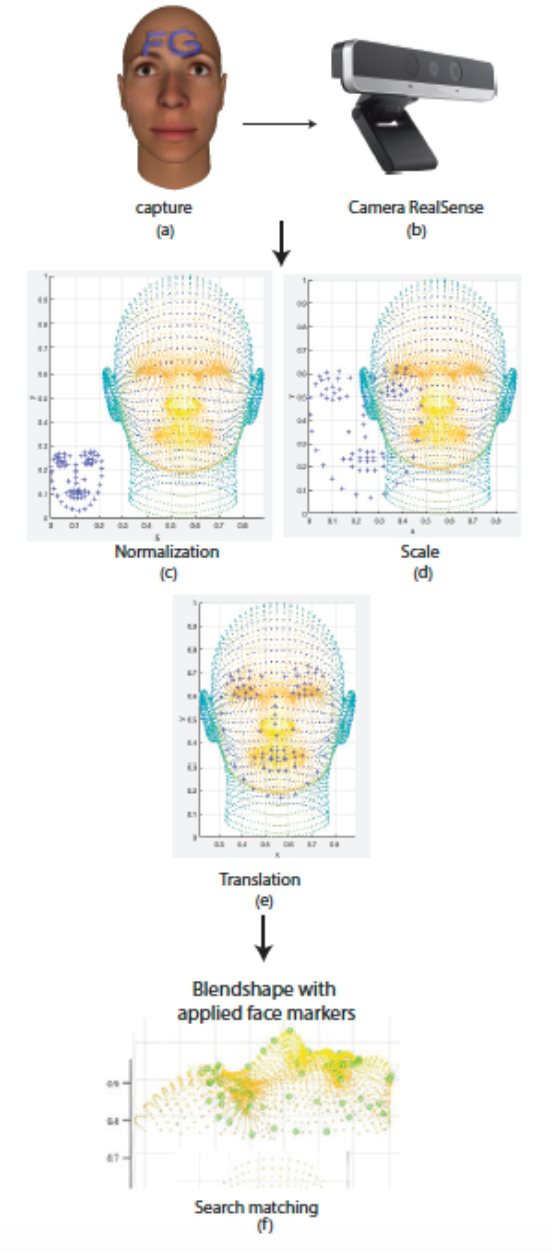


Fig. 3. Steps in generating blendshapes with facial marker associations. (a) 2D image; (b) RealSense camera detects the face and captures the points; (c) standardized model; (d) Adjusting the 2D points captured by the RealSense camera in the 3D model so that the scales between the models remain the same; (e) translocation of the scaled points using the reference point of the z-axis closest to the camera; (f) search for 3D points matches through the proximity of points.

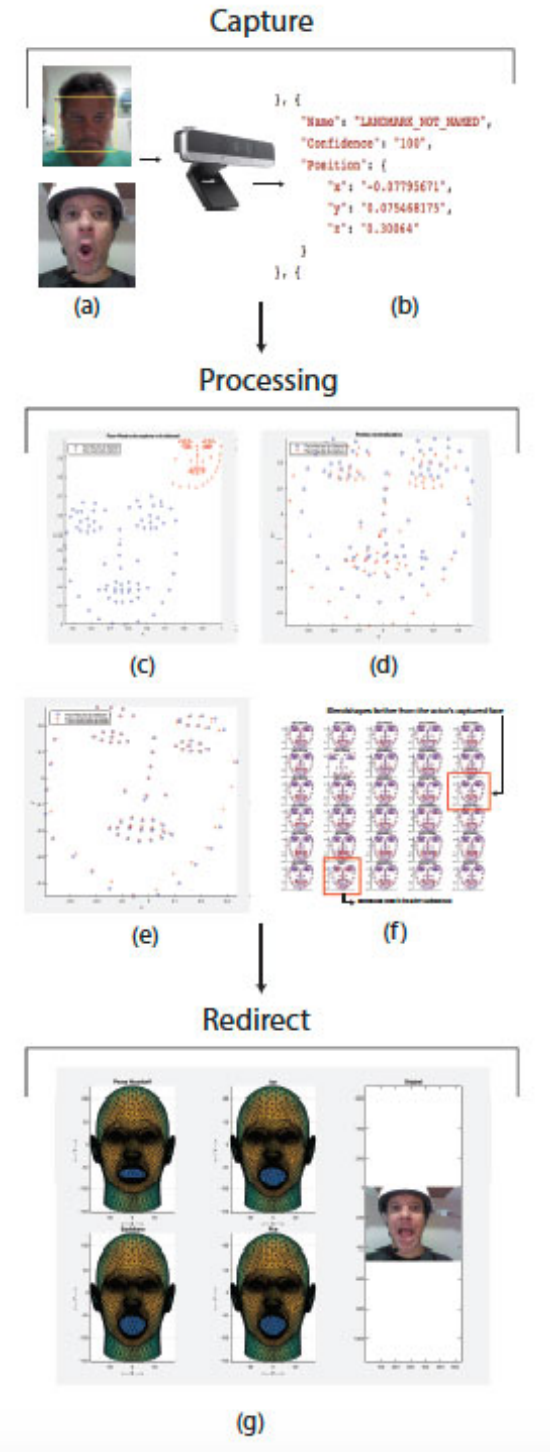


Fig. 4. Pipeline of the capture, processing, and redirection subsystem. (a) frame-by-frame capture of the face of the actor; (b) exported file with 2D and 3D facial marker information; (c) initial adjustment of the neutral face of the actor with the neutral face of the dataset; (d) scale adjustment between faces; (e) translating between the points of the face of the actor and the points of the virtual face; (f) calculation of weights between dataset blendshapes and (g) redirection of weights to the virtual face.

the adjustment of the neutral face from the capture with the adjustment of the neutral face of the dataset. This adjustment is made by normalizing the 3D points of each capture together with each expression of the dataset.

The next step of the adjustment is the scale. It is calculated through a set of points that have a correspondence between them. In this case, eye points were selected from the set of previously captured and adjusted facial markers in the dataset.

The final positioning is the centering of the points through the closest point of the camera. This translation occurs to

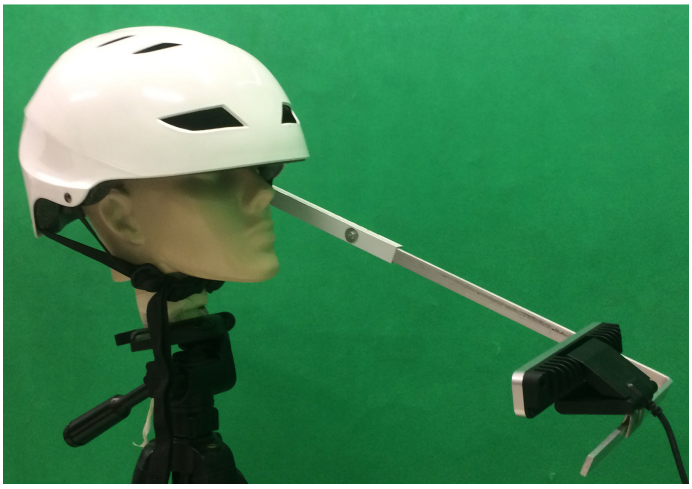

Fig. 5. Helmet used to capture the videos in the initial phase of the process.

minimize the distances between the points of the captured face and the faces of the dataset. After the adjustment, a displacement is applied to the neutral face of the dataset so that it stays with the points in the same position as the points captured by the face of the actor. This improves the calculation of the weights.

*Redirection*

The redirection module uses the values found in the distance calculations between actor face points and blendshapes. This phase generates a new model that, from the neutral face of the dataset, mixes several other models taking into account the proximity of the captured face.

The weights correspond to the calculation of the distance between the blendshapes and the captured face. For the applications of weights, the values of the process of calculating distance between the points is normalized between the values zero and one, with the value one being the closest to the expression. A result of this calculation can be seen in Figure 6.

For the redirection, the algorithms found in the literature are used. For this work ICP, an algorithm used to minimize the difference between two point clouds. [2], PCA, which is an algorithm that reduces the dimensionality of data set with the least loss of information. [3], DHM, the maximum distance from one set to the nearest 2D point of another set. [10] and Euclidean Distance, as you calculate the distance between two points in a vector space [18] were chosen, since they are the most used for this type of system. The weights correspond to the computation of the distance between the blendshapes and the captured face. In this step, various combinations of datasets with different numbers of expressions were tested. Depending on the types of expressions present in the dataset, the result can be changed.

For the applications of weights, the values of the distance calculation process between the points is normalized between the values zero and one, the value one being the closest to

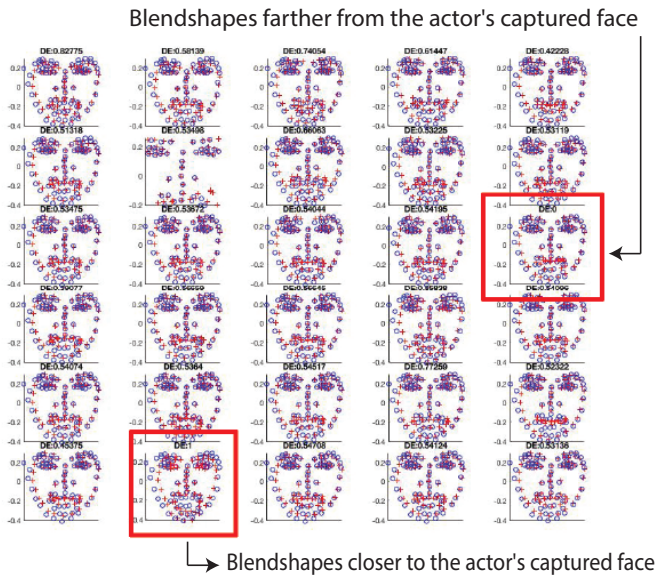


Fig. 6. Example of calculating the distances between the points of the captured face of the actor with the blendshapes

the expression of the dataset and zero being the farthest. The following equation was used:

$$f = \sum_{k=0}^{n} W_k B_k \tag{5}$$

where $W_k$ is the value found in the calculation of the distances and $B_k$ the corresponding blendshape. These values are summed and in the end result is added to the neutral face of the dataset. The result can be seen in Figure 7.

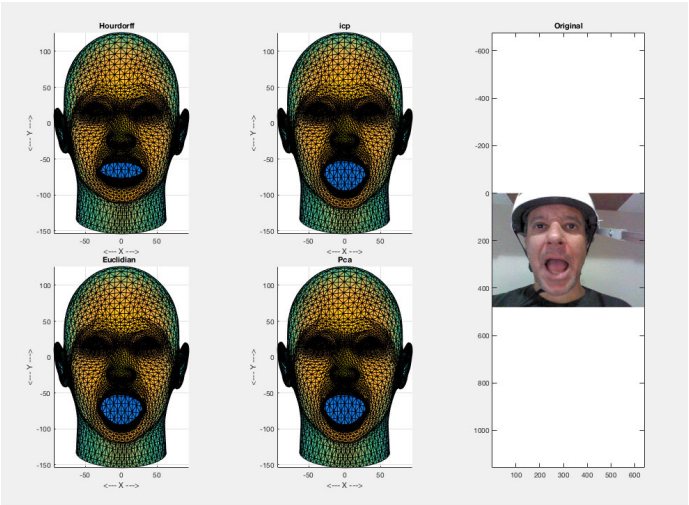


Fig. 7. Example of applying weights in blendshapes.

## VI. Results and Evaluation

Our framework was developed using an Asus computer with a 64-bit operating system, x64-based processor, Windows 10

Pro, Intel Core (Tm) i5-3317U CPU (1.7GHz), four gigabytes of RAM and 500GB of hard disk. An Intel RealSense camera was also used.

There was also use of the development language Java 1.8 and the Matlab software version R2016b with the following installed plugins: MATLAB and Simulink Student Suite, Computer Vision System Toolbox and Neural Network Toolbox.

To perform the tests, two datasets were used. The first was generated by the FaceGen Modeller 3.3 software, comprising 82 FACS-based facial expressions (one of them being the neutral expression) and the second, a personalized dataset with 80 facial expressions that, through the capture of an actor with a neutral expression, were cloned from the FaceGen Modeller 3.3 dataset, using the method of [21], being a personalized dataset and used in the initial pipeline module implemented for the prototype. These expressions correspond to basic emotions, such as joy, sadness, fear, disgust, amazement and crying, among others.

For the tests, through the use of the prototype and the helmet shown in Figure 5, a video of approximately three minutes was made at a rate of 14 frames per second, resulting in a sample of 2,691 frames

In order to determine the accuracy of the screening algorithms selected for the test, the difference between the positioning of the facial markers captured in the initial module (considered to be the reference or ground truth) and the positioning of the markers obtained after the application of the algorithms, in the processing module, was calculated. The calculation of this error was made by means of the Euclidean distance.

For the experiment, 4 sets of blendshapes were randomly selected. The first containing about 25 percent of the total dataset (21 blendshapes), the second set with 50 percent of the total (41 blendshapes), the third with 75 percent of the total (62 blendshapes) and the last with 100 percent of the blendshapes set (82 blendshapes).

We selected 52 facial markers shown in Figure 8 that correspond to the internal points of the face that cover the area of the mouth, eyes and eyebrows. These are the areas used by [23] and [14] in their respective pipelines.

### A. *Results*

The results obtained in the processing of the frames captured using the set of blendshapes generated by the FaceGen Modeller 3.3 software, submitted to selected algorithms and presented from the literature, are demonstrated. Through the graphics shown in Figures 10 and 12, the accuracy of each algorithm in relation to each frame analyzed can be observed. An example of this analysis can be seen in Figure 9, where a highlighted frame shows the great difference in results between the algorithms.

The first experiment uses the framework to test the algorithms selected from the literature. The results can be seen in Figure 10.

Table 1 shows the evolution of the results as a function of the amount of blendshapes selected. As more blendshapes were included in the dataset, the error dropped. This can also be viewed in Figure 11.

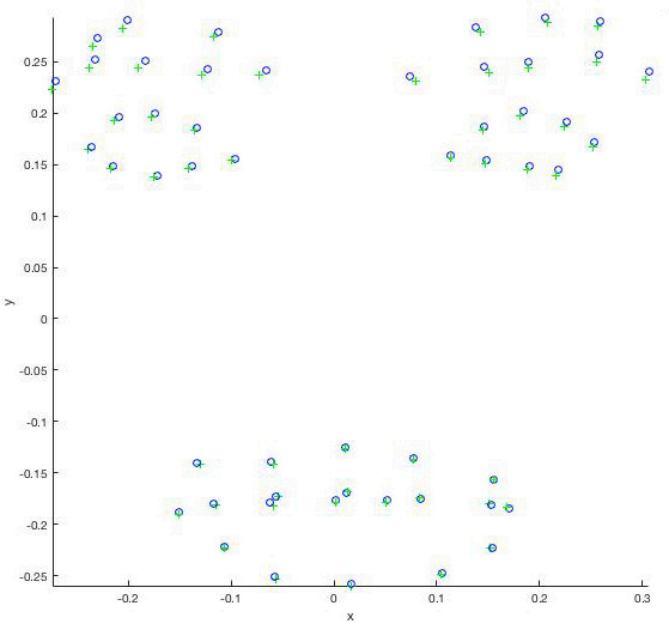


Fig. 8. Points selected for the tests, totaling 52 facial markers.

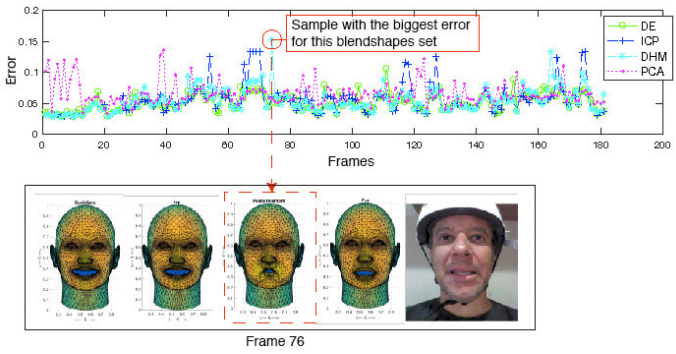


Fig. 9. Example of algorithm analysis for each captured frame. It is noted that the highlighted frame present poor DHM algotithm recognition.

TABLE I
RESULTS IN MILLIMETERS OF THE PROCESSING OF RANDOMLY GENERATED SETS OF BLENDSHAPES.

| Blendshapes | Euclidean | ICP | DHM | PCA |
|---|---|---|---|---|
| 25% | 0.57576 | 0.56825 | 0.56898 | 0.57959 |
| 50% | 0.5704 | 0.5833 | 0.058498 | 0.58094 |
| 75% | 0.61059 | 0.53811 | 0.60951 | 0.61059 |
| 100% | 0.61059 | 0.055849 | 0.60605 | 0.61059 |

The second experiment was carried out to measure the time it took each algorithm to calculate the weights of the blendshapes. The same methodology was applied, using the four sets of randomly selected blendshapes, following the same criteria of the previous experiment. The results can be visualized in the graphs of Figure 12.

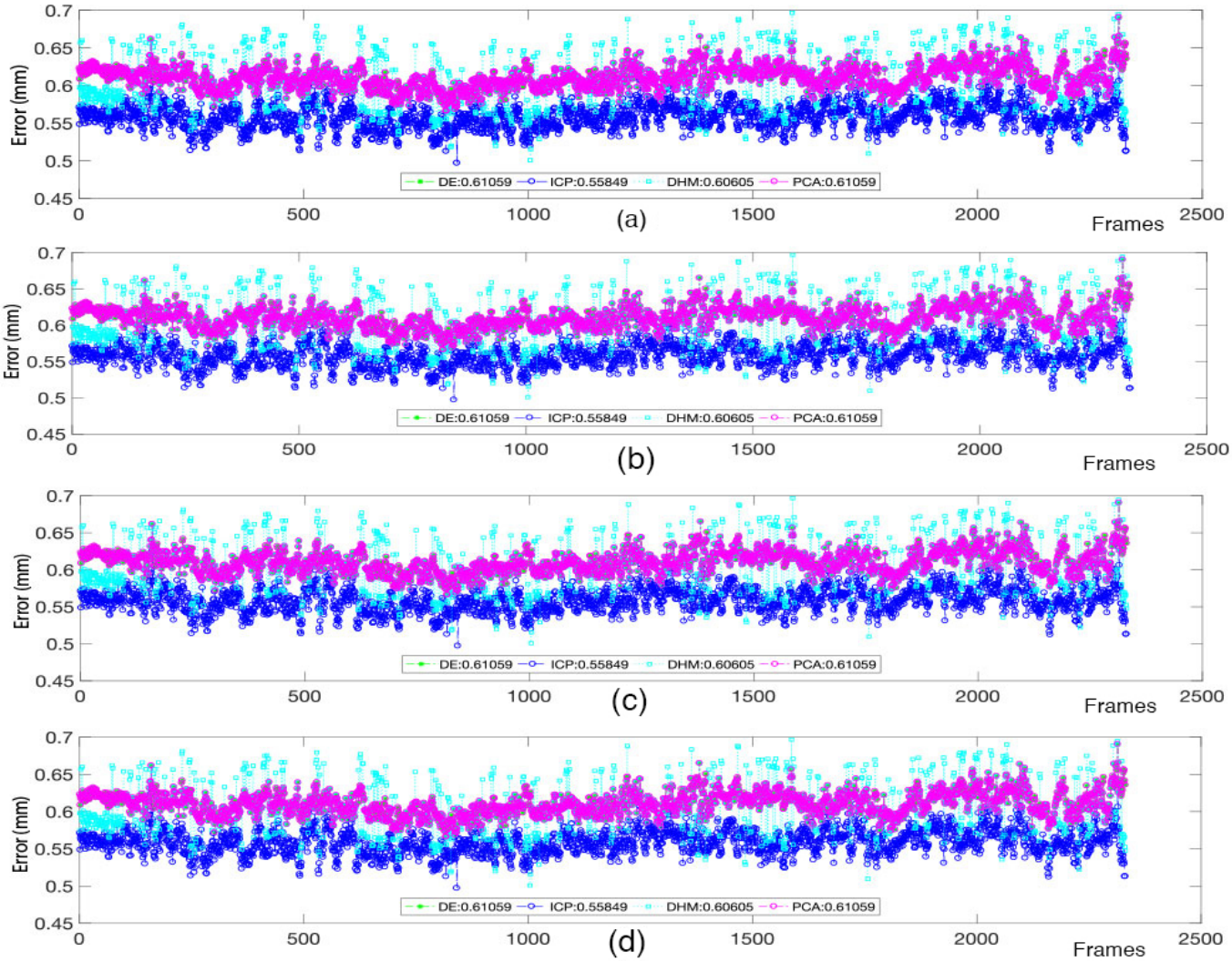


Fig. 10. Graphs showing the calculation of the error among the tested algorithms. (a) 25 percent (21 blendshapes); (b) 50 percent (41 blendshapes); (c) 75 percent (62 blendshapes) and (d) 100 percent (81 blendshapes).

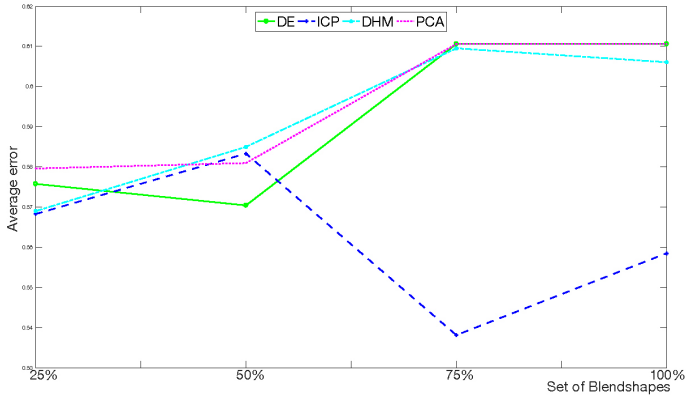


Fig. 11. Visualization of the performance of the algorithms according to the amount of blendshapes

Table 2 shows the result obtained by the algorithms as a function of the processing time, in a summarized form.

TABLE II
RESULTS IN SECONDS OF THE PROCESSING OF THE BLENDED SETS ACCORDING TO THE TIME OF EACH ALGORITHM.

| Blendshapes | Euclidean | ICP | DHM | PCA |
|---|---|---|---|---|
| 25% | 0.0066629 | 0.11487 | 0.0080718 | 0.0070316 |
| 50% | 0.01636 | 0.25739 | 0.018847 | 0.01604 |
| 75% | 0.020346 | 0.32098 | 0.024317 | 0.020492 |
| 100% | 0.026757 | 0.43506 | 0.032258 | 0.028331 |

For the second dataset, the results obtained in the frame processing can be seen in Figure 13. In this figure, the accuracy of each algorithm in relation to each frame analyzed is shown. This dataset is formed by a set of cloned blendshapes using the neutral face of the actor as a base, being a set of blendshapes closer to the geometry of the face analyzed (at variance to the previous dataset that was a more generic set). Through the graphs shown in Figure 15, the time each algorithm took to

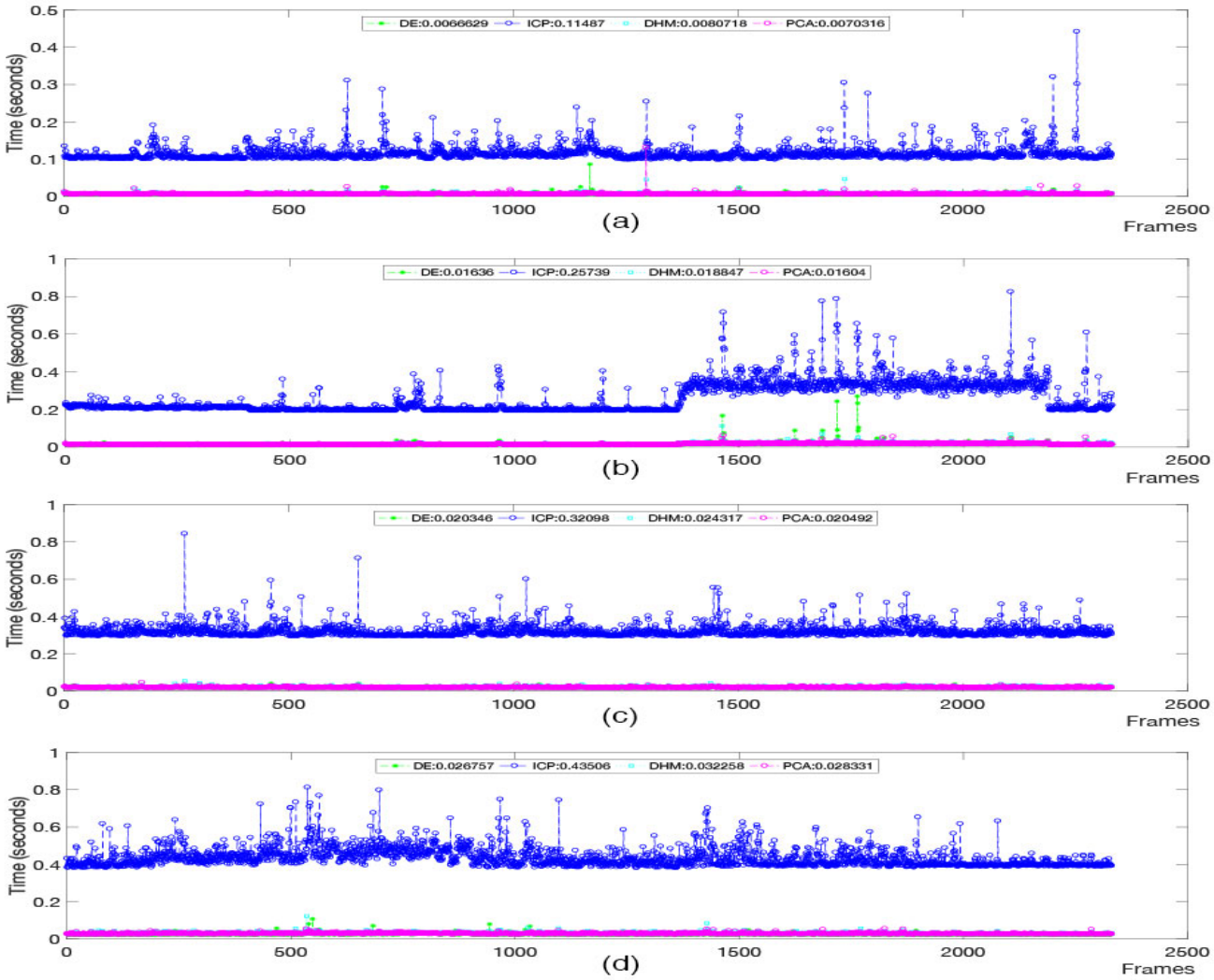


Fig. 12. Graphs showing the time spent per algorithm, frame by frame, for the amount of selected blendshapes. (a) 25 percent (21 blendshapes); (b) 50 percent (41 blendshapes); (c) 75 percent (62 blendshapes) and (d) 100 percent (81 blendshapes).

process the blendshapes for this dataset can be seen.

Table 3 shows the evolution of the results in relation to the quantity of blendshapes selected. It is clear that the error rate has dropped in comparison to the results of the first dataset. The result can also been seen graphically in Figure 14.

TABLE III
RESULTS IN MILLIMETERS OF THE PROCESSING OF RANDOMLY GENERATED SETS OF BLENDSHAPES.

| Blendshapes | Euclidean | ICP | DHM | PCA |
|---|---|---|---|---|
| 25% | 0.49399 | 0.50768 | 0.49026 | 0.51372 |
| 50% | 0.49355 | 0.48413 | 0.53793 | 0.51383 |
| 75% | 0.4913 | 0.47216 | 0.53823 | 0.50697 |
| 100% | 0.4907 | 0.4929 | 0.55334 | 0.4907 |

When comparing the two results, the generic dataset and the dataset customized for the author's face, it can be seen, according to the results obtained, that, in most cases, the dataset which has faces similar to those captured performs

TABLE IV
RESULTS IN SECONDS OF THE PROCESSING OF THE BLENDED SETS ACCORDING TO THE TIME OF EACH ALGORITHM.

| Blendshapes | Euclidean | ICP | DHM | PCA |
|---|---|---|---|---|
| 25% | 0.0067576 | 0.1087 | 0.0077405 | 0.0070019 |
| 50% | 0.014907 | 0.23638 | 0.01702 | 0.014889 |
| 75% | 0.022562 | 0.34619 | 0.02558 | 0.02257 |
| 100% | 0.025639 | 0.40697 | 0.029538 | 0.025699 |

better in terms of distance error between the frame capture and the final blendshape generated through the chosen algorithms. This can be seen in Figures 16, 17, 18 and 19, which compare the two results.

It is also worth noting that according to [16], in general, 30 frames per second is the minimum needed to achieve real time. In these circumstances, based on the values obtained, it can be affirmed that the algorithms that give frame processing performance of less than 0.3 seconds fit into the time needed to

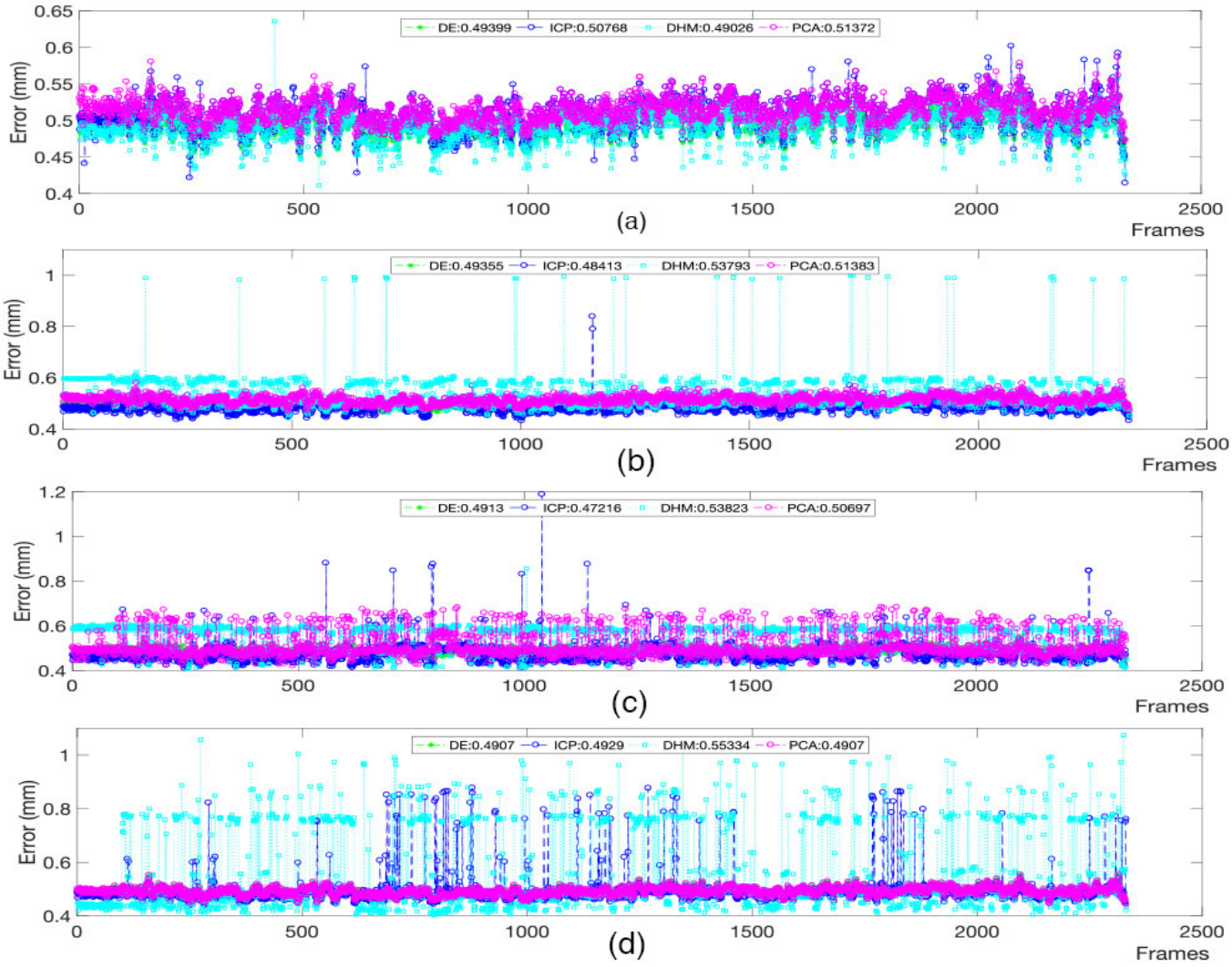


Fig. 13. Graphs showing the calculation of the error among the tested algorithms. (a) 25 percent (21 blendshapes); (b) 50 percent (41 blendshapes); (c) 75 percent (62 blendshapes) and (d) 100 percent (80 blendshapes).

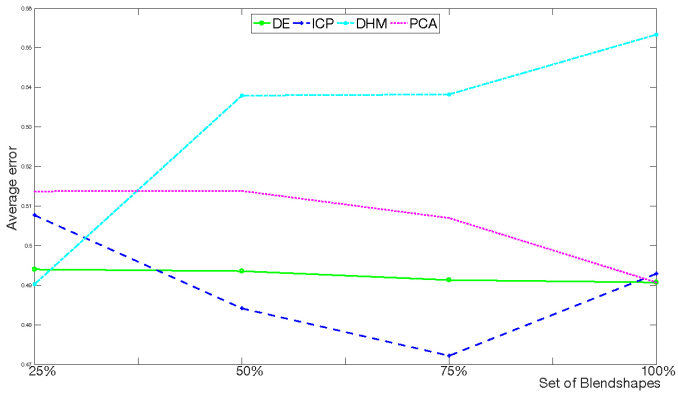


Fig. 14. Visualization of the performance of the algorithms according to the amount of blendshapes

obtain real time. This can be seen in the majority of the cases presented in the two types of dataset analyzed. For gaming, the number of frames per second, in some cases, reaches a rate of 60. In this case, the algorithms which have a processing speed of up to 0.016 seconds could be used, as is the case of the Euclidean Distance using 25% and 50% of the blendshapes and ICP using 25% and 50% of the blendshapes, for example.

## VII. Conclusion and Future Work

An important step in the performance-based facial animation process is the quality with which the redirection of the captured face of the actor is transferred to a virtual face. For this type of task, specific pipelines were created, but most are in the industry, not allowing for their reproduction because they have proprietary algorithms.

These pipelines seek as ground truth the creation of virtual faces very close to the real face because the main idea is to deceive the human eye, which is an expert in detecting details and a simple error can compromise the entire work.

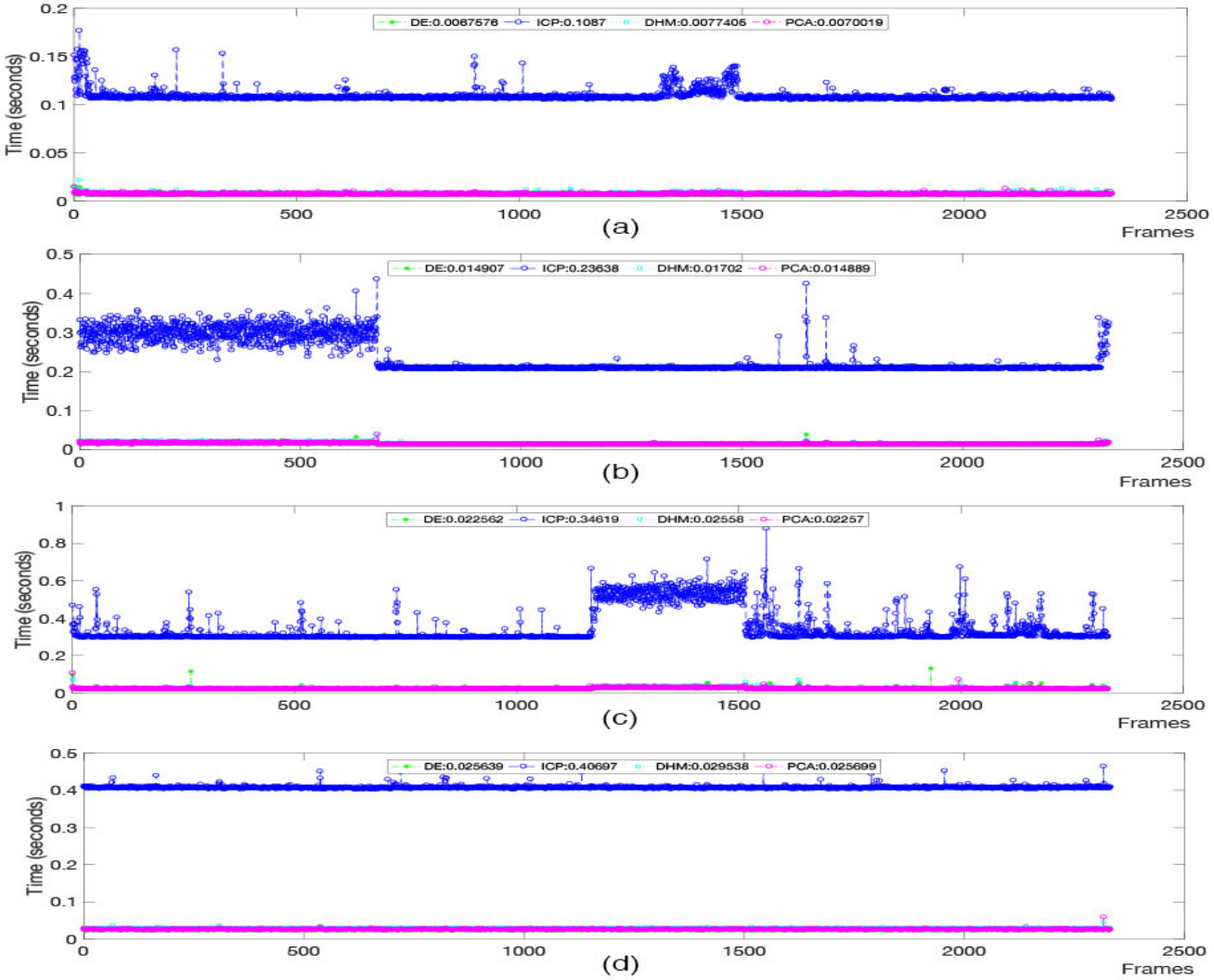


Fig. 15. Graphs showing the time spent per algorithm, frame by frame, for the amount of selected blendshapes. (a) 25 percent (21 blendshapes); (b) 50 percent (41 blendshapes); (c) 75 percent (62 blendshapes) and (d) 100 percent (80 blendshapes).

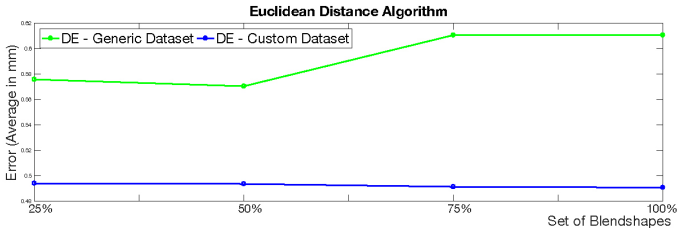


Fig. 16. Graph showing the error difference between the generic dataset and customized dataset processing for the Euclidean Distance algorithm.

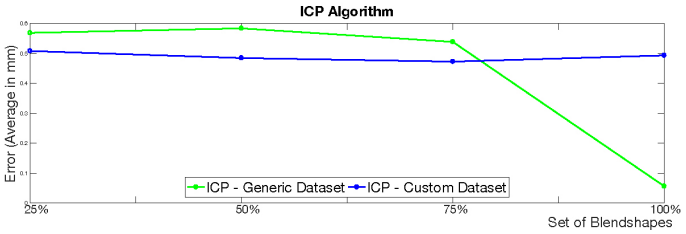


Fig. 17. Graph showing the error difference between the generic dataset and customized dataset processing for the ICP algorithm.

To this end, pipelines, in addition to using algorithms such as those presented in this work, combine several algorithms, in several different stages, between offline and online processes to improve the final quality, including in real-time (which is a necessity of the film and games industry).

Thus, this work presents a framework to test the algorithms used in performance-based facial animation and has a blendshapes processing module, a weights calculation module and a module for the creation of Virtual faces incorporating the modules normally found in the literature for this type of task. In addition, it introduces the use of an RGB-D camera, the RealSense from Intel that has several algorithms implemented for image processing, such as face detection and detection of facial markers.

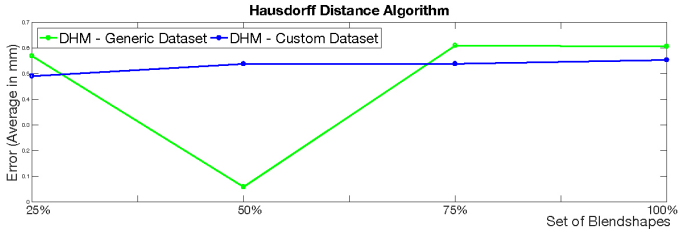


Fig. 18. Graph showing the error difference between the generic dataset and customized dataset processing for the Hausdorff Distance algorithm.

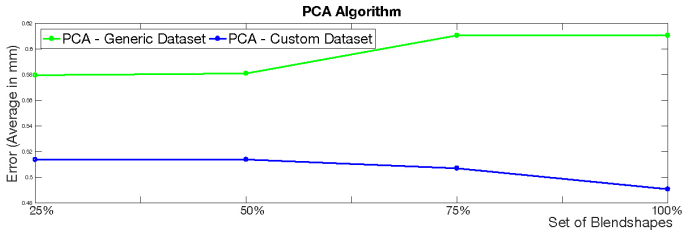


Fig. 19. Graph showing the error difference between the generic dataset and customized dataset processing for the PCA algorithm.

Possible developments resulting from this work may be:

1) Improvement of the dataset creation module, incorporating the algorithms found in the literature for the transfer of shapes between the meshes, allowing the creation of customized datasets for each actor;
2) Improvement of the calculation of the weights applied to the blendshapes, by means of other algorithms of tracking and the use of neural networks;
3) Incorporation of available blend data bases for research that are labeled according to the FACs, such as the database Bosphorus [7] and FaceWarehouse [4];
4) Inclusion of a module for face rendering, using, for example, a game engine such as Unity 3D;
5) Using other cameras to capture data such as Microsoft's Kinect; and
6) Treatment of the characteristics of the human face that were not considered in this work, such as: eyes, teeth, tongue and hair.

## REFERENCES

[1] S. Behrens, A. Al-Hamadi, E. Redweik, and R. Niese. Automatic realtime user performance-driven avatar animation. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pp. 2694–2699. IEEE, 2013.

[2] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn. A survey of rigid 3d pointcloud registration algorithms. In *Fourth International Conference on Ambient Computing, Applications, Services and Technologies*, pp. 8–13, 2014.

[3] A. Braun. *Aprendizado e utilização do estilo de movimento facial na animação de avatares*. PhD thesis, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2014.

[4] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3d facial expression database for visual computing. *Visualization and Computer Graphics, IEEE Transactions on*, 20(3):413–425, March 2014. doi: 10.1109/TVCG.2013.249

[5] L. Dutreve, A. Meyer, and S. Bouakaz. Feature points based facial animation retargeting. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pp. 197–200. ACM, 2008.

[6] P. Ekman and E. L. Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, 1997.

[7] T. Fang, X. Zhao, O. Ocegueda, S. Shah, and I. Kakadiaris. 3d facial expression recognition: A perspective on promises and challenges. In *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 603–610, March 2011. doi: 10.1109/FG.2011.5771466

[8] M. Fratarcangeli. Computational models for animating 3 d virtual faces. 2013.

[9] P. Hong, Z. Wen, and T. S. Huang. Real-time speech-driven face animation with expressions using neural networks. *Neural Networks, IEEE Transactions on*, 13(4):916–927, 2002.

[10] M. J. Hossain, M. A. A. Dewan, K. Ahn, and O. Chae. A linear time algorithm of computing hausdorff distance for content-based image analysis. *Circuits, Systems, and Signal Processing*, 31(1):389–399, 2012. doi: 10.1007/s00034-011-9284-y

[11] Intel. Realsense. https://communities.intel.com/docs/DOC-24012. Accessed: 22/07/2017.

[12] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng. Practice and theory of blendshape facial models. In *Eurographics (State of the Art Reports)*, pp. 199–218, 2014.

[13] D. Li, C. Sun, F. Hu, D. Zang, L. Wang, and M. Zhang. Real-time performance-driven facial animation with 3ds max and kinect. In *Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on*, pp. 473–476, Nov 2013. doi: 10.1109/CECNet.2013.6703372

[14] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.*, 32(4):42:1–42:10, July 2013. doi: 10.1145/2461912.2462019

[15] C. Luo, J. Yu, C. Jiang, R. Li, and Z. Wang. Real-time control of 3d facial animation. In *Multimedia and Expo (ICME), 2014 IEEE International Conference on*, pp. 1–6, July 2014. doi: 10.1109/ICME.2014.6890231

[16] Microsoft. Understanding frames per second (fps). https://support.microsoft.com/en-us/kb/269068, 2003. Accessed: 2017-01-15.

[17] M. B. Nendya, E. M. Yuniarno, and S. Gandang. Facial rigging for 3d character. *Int. J. Comput. Graph. Animat*, 4(3):21–29, 2014.

[18] S. Robison. Geometria plana. https://goo.gl/NOUxIl, 2014. Accessed: 2016-12-05.

[19] E. B. Roesch, L. Tamarit, L. Reveret, D. Grandjean, D. Sander, and K. R. Scherer. Facsgen: A tool to synthesize emotional facial expressions through systematic manipulation of facial action units. *Journal of Nonverbal Behavior*, 35(1):1–16, 2011.

[20] Y. Seol, J. P. Lewis, J. Seo, B. Choi, K. ichi Anjyo, and J. yong Noh. Spacetime expression cloning for blendshapes. *ACM Trans. Graph.*, 31:14:1–14:12, 2012.

[21] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23(3):399–405, 2004.

[22] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Kinect-based facial animation. In *SIGGRAPH Asia 2011 Emerging Technologies*, SA '11, pp. 1:1–1:1. ACM, New York, NY, USA, 2011. doi: 10.1145/2073370.2073371

[23] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. In *ACM Transactions on Graphics (TOG)*, vol. 30, p. 77. ACM, 2011.

[24] S. Yuan, Y. Lu, and H. He. Midi-based software for real-time network performances. In *Cryptography and Network Security, Data Mining and Knowledge Discovery, E-Commerce & Its Applications and Embedded Systems (CDEE), 2010 First ACIS International Symposium on*, pp. 226–230. IEEE, 2010.

[25] M. Zeng, L. Liang, X. Liu, and H. Bao. Video-driven state-aware facial animation. *Computer animation and virtual worlds*, 23(3-4):167–178, 2012.

[26] Q. Zhang, Z. Liu, G. Quo, D. Terzopoulos, and H.-Y. Shum. Geometry-driven photorealistic facial expression synthesis. *Visualization and Computer Graphics, IEEE Transactions on*, 12(1):48–60, 2006.

[27] M. Zhou, L. Liang, J. Sun, and Y. Wang. Aam based face tracking with temporal matching and face segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 701–708. IEEE, 2010.