Improved Visual Speech Synthesis using Dynamic Viseme k-means Clustering and Decision Trees

Christiaan F. Rademan, Thomas Niesler

Department of Electrical and Electronic Engineering,
University of Stellenbosch, South Africa
christo@ml.sun.ac.za, trn@sun.ac.za

Abstract

We present a decision tree-based viseme clustering technique that allows visual speech synthesis after training on a small dataset of phonetically-annotated audiovisual speech. The decision trees allow improved viseme grouping by incorporating k-means clustering into the training algorithm.
The use of overlapping dynamic visemes, defined by tri-phone time-varying oral pose boundaries, allows improved modelling of coarticulation effects. We show that our approach leads to a clear improvement over a comparable baseline in perceptual tests.
The avatar is based on the freely available MakeHuman and Blender software components.
Index Terms: conversational agent, talking head, visual speech synthesis, lip animation, coarticulation modelling, CART-based viseme clustering, audio-visual speech data corpus.

1. Introduction

Interaction with sophisticated technologies is increasing the demand for more natural and human-like communication in user interfaces. Part of this movement looks towards virtual avatars as a medium for intuitive, human-like conversational agents. Visual speech synthesis (VSS) plays a critical part in this drive. The work we present here focuses on advancing methods for VSS in computer animated characters.
A phoneme is a classification of a distinct speech sound unit, based on the place and method of articulation. Phonemes can themselves be grouped based on their visual articulation and their pronunciation attributes. When phonemes are grouped based on visually-similar attributes, they are referred to as visemes, a contraction of the words "visual" and "phoneme" [1]. Visemes do not have a one-to-one relation to phonemes [2]. This is due to the inertia of articulatory organs as the shape of the mouth is greatly affected by the articulation of the surrounding units of speech [3]. It is critical to consider this dominance of vocal articulators and their effects on visual speech segments in VSS [4]. It is for this reason that we have chosen to use dynamic visemes.
The dynamic visemes employed in this paper are an adaptation of those described in [5]. Dynamic visemes do not represent a fixed-point pose, but rather a trajectory describing the evolution of displacement with time. We introduce dynamic visemes that are bound to groups of three separate phones (tri-phones). During synthesis, the overlapping dynamic visemes are concatenated using a weighted transition function.
A recent and successful approach to phoneme-to-viseme mapping makes use of classification and regression trees (CART) [6, 7]. Here, clusters of visemes are recursively subdivided based on questions regarding their phonetic properties with the aim of maximising within cluster viseme similarity. We present, on extension to this, a decision tree approach which incorporates k-means clustering to improve viseme similarity.
Gathering all possible examples of naturally occurring tri-phone arrangements would require a huge database. Further more, phonetic annotation is time consuming and expensive. Therefore, our VSS system is based on freely available software components and has been developed and tested using a small dataset and consumer audiovisual equipment. It may therefore be of interest in situations where resources such as annotated data are scarce. For these reasons we have chosen to use a more simplistic model for VSS, whose configuration requires minimal input data.

2. Data

Audiovisual data was captured from a single speaker reading the first 120 sentences (approximately 10 minutes of speech) from the TIMIT corpus [8]. The motivation for using TIMIT is that the sentences it contains are designed to be phonetically diverse, and cover a broad variety of phoneme sequences. A Panasonic HDC-TM900 video camera, at 1920x1080 resolution and 25 frames per second, was used during recordings. The head was stabilised and a mirror, set at a 45° angle, was included for a side view of the face, as shown in Figure 1.

2.1. Phonetic annotation

The audio tracks were segmented and hand labelled using Praat, a speech analysis software [9]. The extended speech assessment methods phonetic alphabet (X-SAMPA) was used for phonetic annotation. Hand labelling was done by a phonetic and orthographic transcription specialist to ensure accuracy and consistency.

2.2. Feature tracking

Figure 1 shows a frame taken from a recorded sentence in which the facial markers have been identified. Markers on the nose, chin, centre of the top and bottom lip and the left and right corners of the lips were tracked. These six facial feature markers were chosen based on experimentation with animated character motion capture, discussed in Section 3.
The tracking algorithm detects and identifies the white markers' horizontal and vertical positions using relative and previous locations. All trajectories were subsequently manually checked for accuracy. The trajectories of the facial feature markers were normalised relative to the nose, because its movements correlated with those of the performers. We did not capture or model

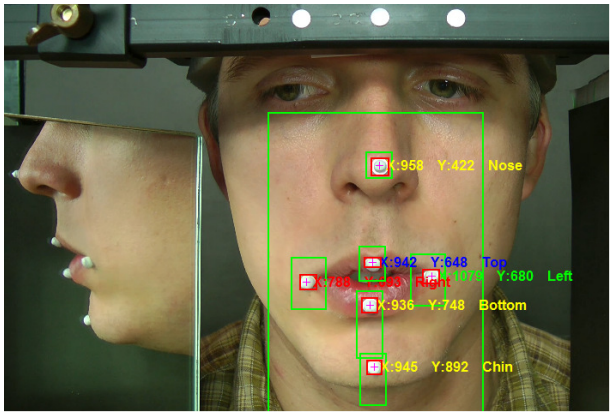

Figure 1: *Facial feature marker trajectory tracking.*

the articulatory movements of the tongue.

2.3. Data processing

The full corpus consists of the original video recordings, X-SAMPA phonetic annotations (including the timing of the phonetic segment boundaries), audio recordings and time-position trajectories for the five features (relative to the nose). The corpus was processed to create a training dataset containing, for each feature, a list of all tri-phone labels, tri-phone duration, true or false values for articulatory phonetic properties (e.g. vowel, schwa, alveolar, fricative, pause etc.) and time-position trajectories, referred to as dynamic visemes in the context of this work. The dynamic visemes were obtained by sampling the marker trajectories of each tri-phone at ten uniformly spaced instances.

3. Animating captured motion

The open source 3D computer graphics software MakeHuman 1.0.2 [10] and Blender 2.70 [11] were used to create and animate the avatars. MakeHuman provides realistic, customisable humanoid character models, which can be imported into the Blender Game Engine (BGE).

Anatomically correct physics-based muscle and skin simulation can produce good VSS, as demonstrated in [12]. This process is very complex. However, we have chosen a much simpler approach that, nevertheless, leads to good results. Our solution uses the rig provided by the MakeHuman model, applying bone driven shape key animation based on tracked feature trajectories.

Rigging is the process of creating a skeleton-like system that consists of bones with which the character can be animated. The function of the bones in a rigged character has been described as "digital orthopaedics", because bones manipulate the areas of the character's mesh to which they are bound, in a way that is reminiscent of how human bones manipulate skin [13]. However, the interaction of multiple bone driven animations becomes cumbersome when trying to achieve the mesh deformations necessary to mimic the subtle facial movements required for speech. Our solution was to use the shape keys (also known as morph targets or blend shapes) provided by MakeHuman.

Animation artists use shape keys to create a library of character mesh deformations with which to speed up animation processes [13]. A shape key is created by saving the deformation of the character's mesh relative to its neutral state. It is then possible to interpolate between the neutral and fully-formed poses of the characters [13]. In the BGE, bones can be used as a proxy to drive the shape keys using a process known as bone driven shape key animation, illustrated in Figure 2 [14]. This process of data-driven animation allows for the motion of the face, captured by the marker trajectories, to be mimicked on the avatar's face. The displacement of the facial markers is used to govern the shape key animation values that manipulate the avatar's mesh, thereby creating the equivalent pose on the avatar's face. The trajectories of the markers at the top, bottom, left and right corners of the mouth, and the chin marker, were scaled to drive shape keys that produced equivalent animations on the avatar. For example, the shape key animation for mouth pursing used the width between the mouth corner markers to animate "O"-shaped poses. To animate bottom lip roll (mouth eversion), the bottom lip to chin distance was used. When compared with traditional motion capture or bone driven animation, our resultant system has the advantage of allowing trajectories to be mimicked using any MakeHuman model.

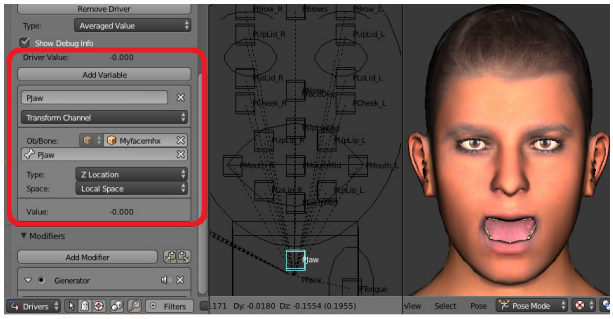

Figure 2: *Example of bone driven shape key animation interpolating between the neutral and open jaw pose.*

4. Minimum deviation decision trees

In this section we describe the decision tree-based viseme clustering methods first proposed in [6], and subsequently expanded to many-to-many phoneme-to-viseme mappings in [7]. Both contributions discuss the application of regression trees to the grouping of static visemes. Clusters of static visemes are split by querying their phonetic context or properties. Figure 3 illustrates the decision tree training algorithm extended to use our dynamic visemes. Since the decision tree algorithms test more than one attribute when attempting to split a group of visemes in a leaf node, they can be classified as multivariate CART algorithms [15, 16, 17].

The decision tree described in [7] applies all possible phonetic context questions to the static visemes grouped in a decision tree's leaf node. The algorithm then measures how homogeneous the resulting child nodes are. An active appearance model (AAM) is used for automatic markerless facial tracking, generating the parameters that numerically describe the static visemes.

Equation 1 is applied to each phoneme instance p_i in a leaf node, where $d(p_i, p_j)$ is the Euclidean distance between instances p_i and p_j and N is the number of phonemes in the node. The smallest value μ_{best} and variance σ_{best} are then selected. Equation 2 is then used to determine the subset impurity I_Z, in which λ is a scaling factor. This procedure is repeated to find the question whose subset best minimizes the impurity of the AAM parameters in the child nodes.

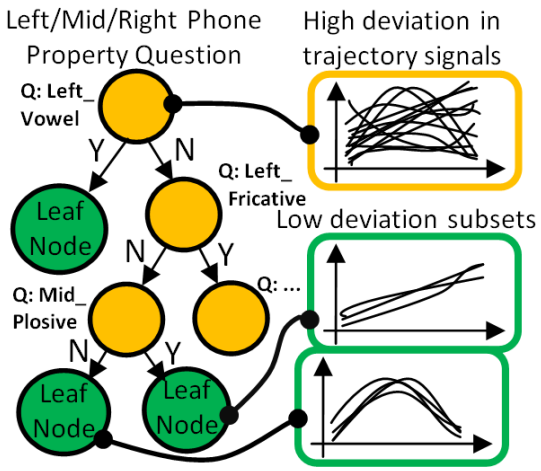

Figure 3: *Illustration of the intended resultant effect of phonetically grouped dynamic visemes. Static visemes, as used in [7], would be represented by points of similar displacement in leaf nodes.*

$$\mu_i = \frac{\sum_{j=1}^{N} d(p_i, p_j)}{N-1} \quad (1)$$

$$I_Z = N \times (\mu_{best} + \lambda \times \sigma_{best}) \quad (2)$$

Our baseline is an adaptation of the work described above. We consider time-dependent visemes, and measure all distances relative to a mean viseme calculated using Equation 3. We then minimise the average deviation D_{Avg} from this mean to split the leaf nodes, using Equation 4. This alleviates the need for an arbitrary scaling factor. This process is repeated for every possible phonetic question. The question resulting in subsets with minimum average deviation in their constituent dynamic viseme subset is chosen to form the child nodes. A minimum occupancy count, as well as a minimum reduction in deviation, are used as stopping criteria.

$$\mu_n = \frac{1}{N} \sum_{i=1}^{N} P_i \quad (3)$$

$$D_{Avg} = \frac{1}{M} \sum_{j=1}^{M} \frac{1}{N} \sum_{i=1}^{N} \mid P_{j,i} - \mu_i \mid \quad (4)$$

5. *k*-means decision trees

We now present an alternative way of choosing optimal decision tree questions. We use k-means clustering [18], with $k = 2$, to classify similar dynamic visemes in a parent node. A separate CART then tries to find the phonetic attribute question which would split the data into subsets that best match these two classes. For this, entropy was used as an impurity measure for discriminating information gain between phonetic attributes [19].

Entropy of a set $H(S)$, is dependant on the number of elements of each discrete class x_i in the class domain (x_1, x_2), as outlined in Equation 5. High entropy is defined by having a large portion p of elements of a class appearing in a set. Information gain IG is calculated by finding the entropy difference between subsets and the original set. Subsets are created by asking

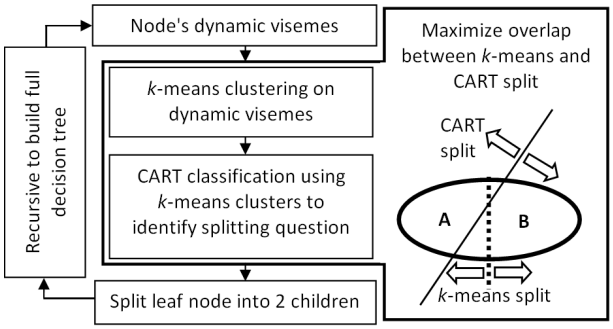

Figure 4: *Illustration of how the k-means (k=2) classifier is used to choose questions and grow a decision tree. The CART classification algorithm finds the question that splits the dynamic visemes into two nodes that best match the clusters produced by the k-means algorithm.*

all phonetic questions. The discrete attribute phonetic question PQ resulting in the set S with maximum information gain is found using Equation 6. In Equation 6, $p(c_j)$ is the proportion of elements in the child node C to the number of elements in the parent node P.

$$H(x_{1,2}, S) = -\sum_{x_i \in (x_1, x_2)} p(x_j) \log_2 p(x_i) \quad (5)$$

$$argmax IG(PQ, S) = H(P) - \sum_{c_j \in (PQ)} p(c_j) H(C) \quad (6)$$

Once the phonetic attribute leading to the greatest information gain is identified, it is used to split the parent node and populate the child nodes. This process is repeated at each node, as illustrated in Figure 4. For our k-means CART algorithm, only the minimum number of dynamic visemes in a parent node was specified as a stopping criterion.

This method was inspired by the work of DeMartino *et al.* [20], who applied k-means clustering to find static poses of fiduciary points in speech according to geometric similarity. By considering nonsense CVCV words, a set of context-dependent visemes could be found for VSS. Our algorithm extends this work by integrating k-means clustering into a phonetic-based decision tree.

6. Dynamic viseme concatenation

For synthesis, a sequence of tri-phone labels, as well as tri-phone start and end times, is provided. For each tri-phone, the decision tree is traversed and the corresponding mean dynamic viseme is retrieved from the leaf node. The dynamic visemes of successive tri-phones are then concatenated using a weighted concatenation function.

Our tri-phone based visemes overlap each other in thirds. A weighted concatenation function tapers each viseme in a piecewise linear fashion, assigning the greatest weight to the dynamic viseme's centre, as show in Figure 5. The weighting functions are applied after each dynamic viseme is scaled to match the duration of the tri-phone in the synthesized utterance. This weighted concatenation approach was used to prevent visemes from dominating whole segments of synthesized speech, while conserving contextual coarticulation effects. The methods used by [4, 5, 20] interpolate between individual static or dynamic visemes, which could be problematic if the chosen

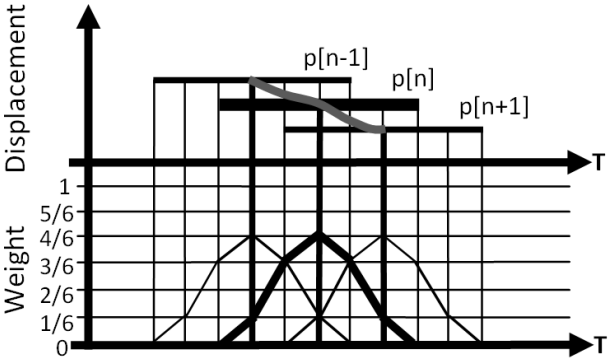

Figure 5: *Illustration of how three successive dynamic visemes p[n-1], p[n] and p[n+1] are concatenated to create a feature's trajectory. The overlapping portions of the visemes are combined using a linear weighting that accentuates their centres.*

viseme misrepresents the speech segment. Our system avoids this potential pitfall by including the effects of the pre and post dynamic viseme context during synthesis, therefore capturing and reproducing more natural coarticulation effects.
The concatenated dynamic visemes are scaled and used to drive the bone driven shape key animations, as discussed in Section 3.2, to produce synthesized visual speech.

7. Testing and evaluation

Of the 120 sentences in our dataset, twelve were randomly chosen and reserved as an independent test set, and the remaining 108 used for training. The twelve test sentences were synthesized using both minimum deviation and k-means decision tree algorithms for both objective and subjective evaluations.

7.1. Objective testing

For quantitative analyses, the synthesized feature trajectories were compared to the original trajectories by calculating an averaged root mean square error (RMSE).
The training set was divided into 12 subsets. Each subset was then used to synthesize the 12 test sentences. The RMSE was calculated by measuring the difference between the synthesized and original trajectories every 2.4 milliseconds. The RMSE was averaged over every feature for all 12 test sentences. This calculation was repeated for every subset. The subsets were then merged into groups of increasing size, and the procedure repeated. In this way 1, 2, 3, 6 and 12 (i.e. all) subsets were used as training material.
Figure 6 shows the RMSE as a function of training set. Table 1 reveals the RMSE and standard deviation when using all 108 sentences for training. From these results it can be seen that both algorithms show continuous improvement as the number of training sentences increases, with neither clearly outperforming the other. Table 1 shows that the minimum deviation algorithm performed marginally better than our k-means decision trees, but that the latter had a slightly lower RMSE standard deviation, indicating better dynamic viseme clustering.

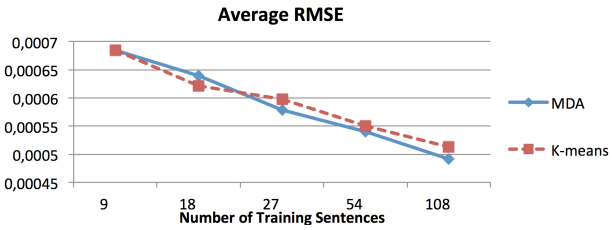

Figure 6: *RMSE using incremental training subsets for minimum deviation algorithm (MDA) and k-means algorithm.*

Table 1: *RMSE and RMSE standard deviation (STD) using the twelve random test sentences with 108 training sentences.*

	Minimum deviation decision tree	k-means decision tree
RMSE	0.000491	0.000513
RMSE STD	0.000238	0.000220

7.2. Subjective testing

The twelve test sentences were also used for subjective evaluation in the form of human perceptual tests. Avatars were animated using both minimum deviation and k-means decision trees. As a baseline, an avatar was also animated using the trajectories obtained from motion capture directly. Test participants were required to watch a sequence of videos, each showing 2 of the 3 possible avatars, randomly chosen. In each video, the first avatar speaks, followed the second, and finally both speak together, as illustrated in Figure 7. Participants were then asked to indicate which avatar was perceived to best articulate the spoken sentence. This was repeated three times for each sentence, allowing all combinations of avatars to be compared for each test sentence. In total, each of the 40 test participants therefore evaluated $12 \times 3 = 36$ videos. Participants were permitted to re-view videos before making a decision. To prevent guessing, participants could also indicate when they could not differentiate between the two avatars.
Figure 8 presents the perceptual evaluation results, showing an improvement of over 12% of our VSS method against the baseline algorithm. Our method also afforded a more favourable assessment than the minimum deviation algorithm, with approximately 15% improvement, when compared with the motion capture baseline.

Figure 7: *Example frame taken from a video used in the perceptual test. In this case the left and right avatars were driven by k-means and minimum deviation decision trees, respectively. The frames show the articulation of the sound "K".*

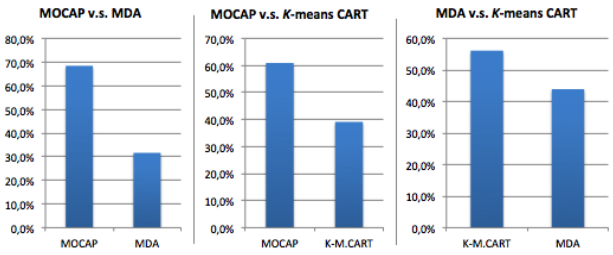

Figure 8: *Perceptual test results comparing direct motion capture (MOCAP), minimum deviation (MDA) and k-means decision tree (K-M.CART) avatar animations.*

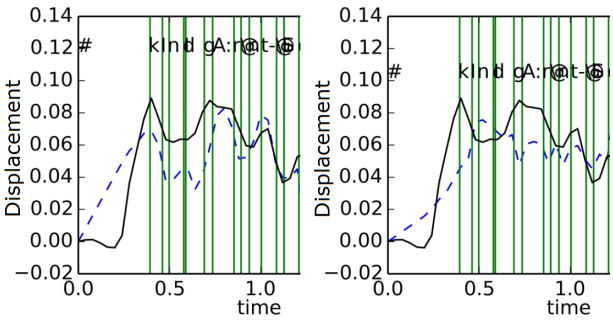

Figure 9: *Synthesised chin trajectories for sentence starting: "Kindergarten children...". The left and right graphs use the k-means and minimum deviation decision trees, respectively, to generate the synthesised blue dashed line. The original chin trajectory is marked by the solid black line. The vertical green lines indicate phone boundaries.*

7.3. Discussion

The baseline algorithm first splits the nodes based on phonetic properties, then tries to find the subset whose visemes are most alike. Our algorithm takes the opposite approach: it first finds a commonality between the visemes and then asks which phonetic property would best preserve this. Although the objective assessment, in terms of RMSE, shows that trajectories synthesized using the baseline algorithm tend to be spatially slightly closer to the original trajectories, RMSE may not be a good indicator for comparing the shape similarity of the synthesized and original trajectories.

As an example, Figure 9 shows the synthesized chin trajectories for the two algorithms. The trajectory synthesized by the minimum deviation algorithm is closer to the original trajectory in terms of RMSE. However, the shape of the trajectory synthesized by the k-means decision tree algorithm better matches that of the original trajectory. This greater qualitative shape similarity appears to be reflected in the perceptual tests.

It is possible that the use of a non-professional voice actor affected the intelligibility of the avatars' speech. With a professional, the recordings may have been more enunciated and better repeated, likely improving dynamic viseme clustering results and the avatars' visual speech. An investigation of this remains for future work.

8. Conclusion and future work

We successfully implemented two improved decision tree algorithms for VSS using dynamic visemes. The trees are trained using tracked and phonetically-annotated audiovisual data. The decision trees were subsequently used to synthesize oral feature trajectories for avatar animation using phonetically-annotated audio alone.

The proposed algorithm incorporates k-means clustering into the decision tree training process. Leaf nodes are split into two child nodes by a process that selects phonetically-based questions which best agree with the k-means clustering result. In this way, the selected decision tree questions best explain the groups seen in the data. The trajectories synthesized using our decision trees led to a slight increase in mean square error relative to a baseline. However, perceptual tests showed a clear improvement over the same baseline. Furthermore, informal qualitative assessment of the trajectories themselves showed that their character better corresponded to the ground truth, even through this was not captured by the mean squared error.

The VSS and animation techniques we present have been shown to work on a small dataset. The algorithms gave good synthesis results even when trained on just 108 sentences. This dataset is sparse when compared with the 1199 and 2542 utterances used in related work [5, 7]. Furthermore, the number of tracked features (6) is far smaller than those typically employed in alternative systems, such as AAMs. Hence, our system may be attractive in situations where neither advanced video capturing equipment nor a lot of data is available. This is typically the case in under-resourced language environments, a category in which much of sub-Saharan Africa falls. By making use of freely-available software tools, such as MakeHuman and Blender, it is possible for small research groups in poorly resourced environments to produce a flexible avatar for use in human-computer interaction.

Future work will include the addition of tongue data capture and animation, and the incorporation of multiple facial features to allow for expressions or gestures typically used in conversations.

9. Acknowledgements

This work was supported in part by the National Research Foundation of the Republic of South Africa (grant TP13081327740). The authors would like to thank Alison Wileman for her phonetic annotations and the Blender and Python online communities for sharing their knowledge.

10. References

[1] Fisher. C.G., "Confusion among visually perceived consonants", Journal for Speech and Hearing Research, 11: 796-804, 1968.

[2] Turkmani, A., "Visual analysis of viseme dynamics", Ph.D. dissertation, Dept. Eng. and Physical Sciences, University of Surrey, Surrey, 2008.

[3] Krňoul, Z., Železný, M., Müller, L., and Kanis, J., "Training of coarticulation models using dominance functions and visual unit selection methods for audio-visual speech synthesis", Proc. Interspeech, 585-588, 2006.

[4] Cohen, M.M., and Massaro, D.W., "Modeling coarticulation in synthetic visual speech", in *Models and Techniques in Computer Animation*, Magnenat-Thalmann, M., and Thalmann, D., Eds., Tokyo: Springer-Verlag, 1993, pp. 139-156.

[5] Taylor, S.L., Mahler, M., Theobald, B.J., Matthews, I. "Dynamic units of visual speech". Proc. 11th ACM SIGGRAPH/Eurographics Conf. Computer Animation. Eurographics Association, 275-284, 2012.

[6] Galanes, F., Unverferth, J., Arslan, L., Talkin D., "Generation of lip-synched synthetic faces from phonetically clustered face movement data" Proc. Int. Conf. Auditory-visual Speech Processing, 191-194, 1998.

[7] Mattheyses, W., Latacz, L., and Verhelst, W., "Comprehensive many-to-many phoneme-to-viseme mapping and its application for concatenative visual speech synthesis", Speech Communication, 55 (7-8): 857-876, 2013.

[8] Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., and Dahlgren, N., "The DARPA TIMIT acoustic-phonetic continuous speech corpus", CD-ROM, National Institute of Standards and Technology, 1986.

[9] Boersma, P., and Weenink, D., "Praat: doing phonetics by computer", Computer program, Version 5.4.08, Online: http://www.praat.org/, accessed on 10 Dec 2015.

[10] blender.org, Blender v2.70, Online: http://www.blender.org, accessed on 19 Feb 2014.

[11] makehuman.org, MakeHuman v1.0.2, Online: http://www.makehuman.org/, accessed 25 Feb 2014.

[12] Sifakis, E., Selle, A., Robinson-Mosher, A., Fedkiw, R. "Simulating speech with a physics-based facial muscle model", Proc. Symposium on Computer Animation (SCA), 261270, 2006.

[13] Hess, R. "Blender foundations: the essential guide to learning Blender 2.6", Amsterdam: Focal Press, 2010.

[14] Thames, C., "Tutorials for Blender 3D", Online: http://www.tutorialsforblender3d.com/, accessed on 19 Feb 2014.

[15] Breiman, L., Friedman, J. H., Olshen, R.A, and Stone, C. J., "Classification and regression trees", Monterey, CA: Wadsworth, 1984.

[16] Witten, I. H., and Frank E., "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", San Francisco, Morgan Kaufmann, 1999.

[17] Quinlan, J.R., "C4.5: Programs for Machine Learning", San Francisco, Morgan Kaufmann, 1993.

[18] Pedregosa, G., Varoquaux, A., Gramfort, V., Michel, B., Thirion, O., Grisel, M., Blondel, P., Prettenhofer, W.R., and Dubourg, V., "scikit-learn: machine learning in Python", in Journal of Machine Learning Research, 12:2825-2830, 2011.

[19] Rokach, L., and Maimon, O., "Data mining with decision trees: theory and applications", Second edition. Hackensack, New Jersey: World Scientific, 2015.

[20] DeMartino, J.M., Magalhaes, L.P., and Violaro, F., "Facial animation based on context-dependent visemes," Computers and Graphics, 30, 971-980, 2006.