Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2022”

Moscow, June 15–18, 2022

Speaker-agnostic mouth blendshape prediction from speech

Vladislav Korzun[1,3]
Moscow, Russia
korzun@phystech.edu

Gadecky Dmitry[1,3]
Moscow, Russia
gadetskiy.dv@phystech.edu

Vladimir Berzin[1,3]
Moscow, Russia
berzin@phystech.edu

Arkady Ilin[2,3]
Moscow, Russia
arkady.ilin@skoltech.ru

[1]Moscow Institute of Physics and Technology
[2]Skolkovo Institute of Science and Technology
[3]Tinkoff

Abstract

This paper describes a simple end-to-end deep learning approach for automated 3D lip animation from audio. Our solution is speaker-independent, which means that once trained on one voice, the model can be applied to any voice without need for retraining. This solution only requires a small amount of data, which can be easily obtained with a modern iPhone. Along with that we also propose a new combined approach for evaluating blendshape prediction models.

Keywords: 3D, lip sync, neural networks, facial animation
DOI: 10.28995/2075-7182-2022-21-323-332

Спикер-независимое предсказание блендшейпов области рта по речи

Владислав Корзун[1,3]
Москва, Россия
korzun@phystech.edu

Дмитрий Гадецкий[1,3]
Москва, Россия
gadetskiy.dv@phystech.edu

Владимир Берзин[1,3]
Москва, Россия
berzin@phystech.edu

Аркадий Ильин[2,3]
Москва, Россия
arkady.ilin@skoltech.ru

[1]Московский физико-технический институт
[2]Сколковский институт науки и технологий
[3]Тинькофф

Аннотация

В данной статье описывается простой подход глубокого обучения для автоматизированной 3D-анимации губ из аудио. Наше решение является спикер-независимым, что означает, что после обучения на одном голосе модель может быть применена к любому голосу без необходимости переобучения. Представленный алгоритм требует лишь небольшого количества данных, которые можно легко получить с помощью современного iPhone. Наряду с этим мы также предлагаем новый комбинированный подход для оценки моделей предсказания блендшейпов.

Keywords: 3Д, синхронизация губ, нейронные сети, лицевая анимация

1 Introduction

3D lip-sync estimates lip motion corresponding to the audio recording of a person's speech. It is a core problem of avatar head animation extensively studied for decades, as the realistic lip motion heavily affects the perception of liveliness and decreases the uncanny valley effect(Mori et al., 2012).

Several approaches to 3D face animation have been proposed in recent years. The majority of them as in (Karras et al., 2017) and (Cudeiro et al., 2019) aim to create a sequence of entire face meshes based on a speech recording. These models are quite large, and they require high-quality training data, which is difficult to obtain without specialized equipment.

In this paper, we present a simple approach for speaker-agnostic 3D lip-syncs through blendshapes generation. We use an affordable motion capture approach and a small amount of data to train our network. Our system could be trained on one person's speech and used on a different voice as it mostly relies on speaker-independent audio feature encoding and light audio feature to blendshape decoder. By evaluating several audio encoders, we show that good audio features are more useful than a complex model in this task, and there is no need to consider the long context of the phrase.

To sum up, our contributions are as follows:

1. Lightweight speaker agnostic audio-to-blendshape model
2. A new combined approach to evaluating synthesized blendshapes

Our paper is organized in the following way: Section 2 describes related work, section 3 presents our method, section 4 describes experiments and section 5 contains the conclusion.

2 Related Work

The creation of a three-dimensional facial animation based on speech could be done in several ways. First: direct face mesh generation. In this method, a 3D mesh is represented as a set of 3D vertices $\overrightarrow{p} \in R^{3N}$ with fixed topology. In paper (Karras et al., 2017) the coordinates of the facial grid vertices are generated directly from the audio feature input window with an additional emotional state. Thus, the entire animation is generated from the sliding window frame by frame. This system has been trained on 3-5 minutes of high-quality 3D scanning sequences of a particular person. This system could be also used on a different voice, but the face 3D model itself is fixed. At the same time, obtaining 3D scans for other faces is a separate challenge and imposes certain limitations on the application of this approach.

To overcome the aforementioned problem with different faces, the parametric face models could be used, such as 3DMM(Blanz and Vetter, 1999), FLAME(Li et al., 2017) or FaceWarehouse(Cao et al., 2013). Parametric face models could be represented as a function $\mathcal{M}(\overrightarrow{\theta}) = \overrightarrow{p} \in R^{3N}$, where $\overrightarrow{\theta}$ - set of parameters. The division of parameters may be different for different models. The FLAME model, for example, separates the parameters into three groups: one represents the face form, another its position, and the third its expression. This division may be useful, for example, for transferring animation from one face to another, changing only face shape parameters.

There are several facial animation models based on parametric face models. For example, VOCA(Cudeiro et al., 2019) uses FLAME by generating 3D meshes in the same topology. As a result, it can be utilized to adjust numerous factors during inference, such as facial shape. Authors also provided VOCASET, a large dataset of 4D scans. Although this method can be used to create facial motion for a variety of faces, the model itself is computationally demanding as it generates the entire face mesh.

Another preferred representation to encode facial animations in CGI production are blenshapes(Lewis et al., 2014). Blendshapes could be used to represent any facial expression as a weighted combination of basis vectors. As a result, any face mesh might be computed as $\overrightarrow{p} = \mathcal{B}w$, where $\mathcal{B} \in R^{3N \times m}$ - basis, $w \in R^m$ - vector of coefficients. The term "basis" usually refers to basic facial expressions such as an open mouth or a closed eye that are chosen to be as independent as possible. It allows w to be meaningful. For example, if we want a facial expression with a half-opened mouth and a closed left eye we take w with a coefficient of 0.5 for the opened mouth and a coefficient of 1 for the left eye. All the other coefficients are set to zero. It's also worth noting that basis vectors usually reflect the margin from idle faces, rather than the mesh itself, i.e. $\overrightarrow{p} = \overrightarrow{p_0} + \mathcal{B}w$. As a result, the facial movement could be

represented as a series of w_i, each of which has a dimension fewer than the number of vertices in the 3D face model.

It is worth noting that blendshapes have already been used as an intermediate representation for neural face puppetry. In (Thies et al., 2020) authors create a latent expression vector from audio. Audio expressions could then be interpreted as blendshapes coefficients of a person-specific generic 3D face model. The mapping from audio-expressions to blendshapes coefficients is trained individually for each person by minimizing the vertex-to-vertex distance between obtained and visually tracked coefficients. Finally, the face mesh obtained from blendshapes is used to generate the final image using neural render.

3 Method

3.1 Problem

In this section, we describe our method for generating facial animations based on blendshape values prediction. First and foremost, let us define the problem. Given a digital audio signal, $w(t), t \in [1, T]$ generate a sequence of corresponding blendshape vectors $\{w_j\}_{j \in 1,M}, w_j \in [0, 1]^m$. Audio signal could be represented as a sequence of audio features $\{a_i\}_{i \in 1,N}$. $a_i \in R^n$. Here N is not equal to M, as the audio features can have different frame rate from target animation.

3.2 Data

First of all, we require appropriate data to train our models. The most challenging part is to acquire blendshapes. A 3D mesh and a basis B are required to obtain the blendshapes weights vector w. Both are difficult to collect and necessitate the use of expensive equipment to scan a 3D mesh from a human face as well as manual adjustments to fine-tune the basis. However, there is an affordable solution to collect blendshapes directly. Using the depth sensor from the most recent iPhones, LiveLinkFace can extract blendshapes from the frontal camera as well as RGB video with audio. We recorded a small dataset with our team members using this application. It is worth noting that we initially attempted to record data during zoom calls. It has the potential to make the collected facial motions more natural and expressive, but we were unable to achieve a significant improvement in results using this source. Then we focused on the method used to collect the GRID dataset (Alghamdi et al., 2018), in which participants were asked to record 100 short phrases. Following this concept, we created a corpus of short phrases found on the internet. Our corpus contains 2500 uttered phrases of two different speakers (2000 and 500 phrases respectfully) ranging in length from three to ten words. We also use additional 20 long phrases for different voices including synthesized ones to test our models.

The input data was captured at 60 frames per second and included 61 blendshapes. We only use 33 of them, picking only those responsible for the lower half of the face. As a result, we do not cover blinking and eye movements as according to recent works (Chen et al., 2020), movements in these areas cannot be unambiguously determined from speech. We also downsampled our data to 30 frames per second.

3.3 Audio processing

For our model to be used on a different voice, the audio encoder should be able to produce speaker-agnostic audio features. We discovered two major approaches to acquiring such features.

For starters, we could use pretrarined embeddings, which are commonly used in speech recognition. Wav2vec 2.0 (Baevski et al., 2020) - a framework trained to produce context representations of audio features - is one of them. This model consists of two major components: a multi-layer convolutional feature encoder and a transformer that builds contextual representations. These representations are used to train our model as audio features.

Wav2vec model showed state-of-the-art results on speech recognition, but its audio embeddings may have some drawbacks in our task. To demonstrate the possible issue we use two subsets of our data. Both have the same phrases recorded by two different speakers. Then, over these subsets, we construct 2D t-SNE (Van der Maaten and Hinton, 2008) projections for audio features and corresponding blendshapes. Finally, we highlight points corresponding to the same phoneme of the one phrase for both speakers with different colors. We found that audio embeddings are projected to almost the same points for both

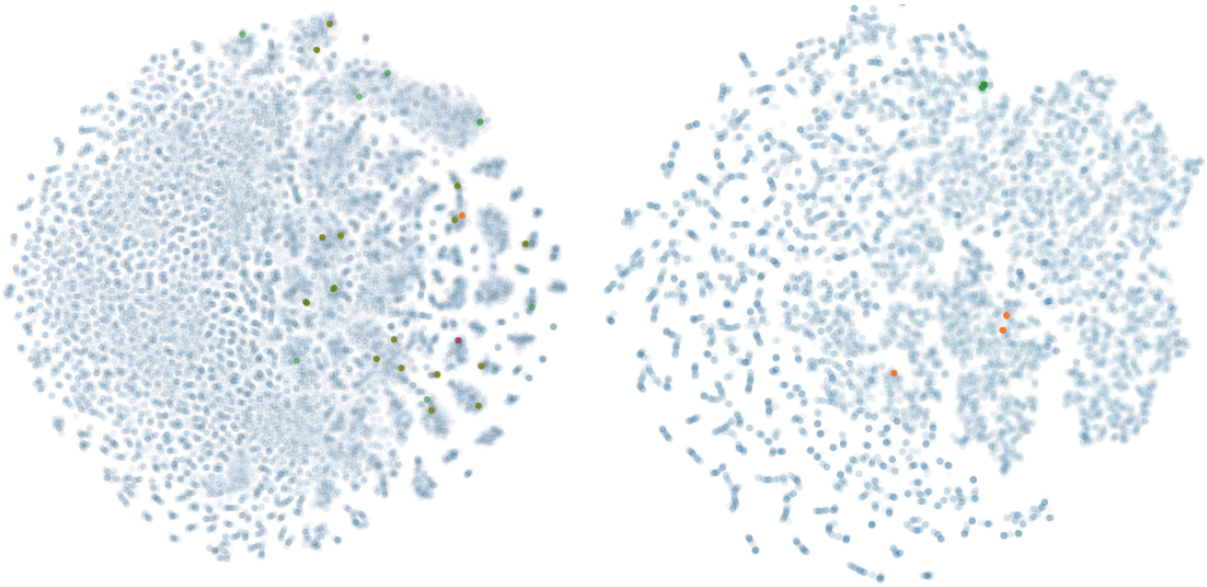


Figure 1: t-SNE projections for wav2vec features (left) and blendshapes (right). Green (first speaker) and Orange (second speaker) colors represent small windows around the same phoneme for two speakers. Dark Green - the intersection of windows for two speakers (orange+green), and Purple - intersection phonemes embeddings itself

speakers while corresponding blendshape projections are far from each other (Figure 1). This difference in the value of the target variable could affect the convergence of our models when we use data from more than one speaker or the same speaker but in different conditions.

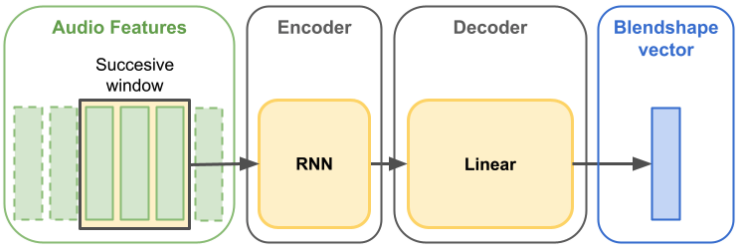
To address this issue, we should employ features that preserve the *timbre* of the voice while removing the speaker's identity. Voice conversion is one of the more straightforward approaches that could be used: given an input speech recording, convert it to sound like a different speaker. Furthermore, the same approach was previously applied to 2D face animation. The authors of MakeItTalk (Zhou et al., 2020) used AutoVC (Qian et al., 2019) to convert input speech to the fixed speaker. Following the same procedure, we were able to train our model on one speaker and then apply it to different voices without having to retrain the entire pipeline.

AutoVC is a tricky intermediate space autoencoder. In this model, the encoder encodes the input Mel-spectrogram and passes it along with the speaker embedding to the decoder. The main idea is to find a dimension of intermediate space that allows only the content of speech to be encoded while retaining the speaker style information from embeddings. We take a pre-trained AutoVC model from MakeItTalk and convert all input Mel-spectrograms, including training data, to a single voice.

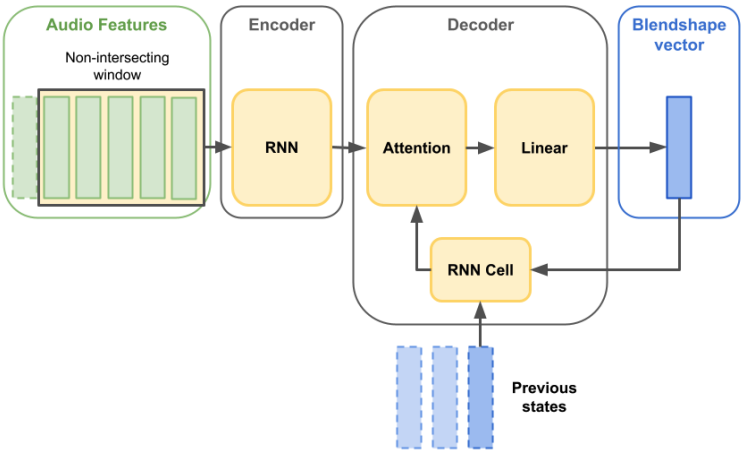
3.4 Blendshapes vector prediction

We suggest that our problem is similar to the generation of gestures for 3D skeletons. Both problems can be thought of as sequence-to-sequence problems where sequence audio features are considered as input and a sequence of real-value vectors as output. For gesture generation, these output vectors reflect joint rotation angles, and in our task, these vectors will contain blendshape values. As a result, we can facilitate the approach that produced good results in gesture generation.

First, like in (Kucherenko et al., 2019) we attempted to generate a blendshapes vector from a fixed-length window of audio features. We encode a window of audio features using a recurrent neural network, then pass this encoding through a simple perceptron to obtain a blendshapes vector for a single frame. Here we do not use additional encodings of blendshapes, such as those from autoencoder. We also use the Savitzky–Golay filter (Savitzky and Golay, 1964) during inference to smooth predicted animation.



(a) FeedForward model



(b) Seq2seq model

Figure 2: Architectures of FeedForward and Seq2seq models

The full architecture of the FeedForward model is visualized in Fig. 2 (a)

The second model is similar to the one proposed in (Korzun et al., 2021). Following this work, we employ a modified variant of sequence to sequence model. It is made up of two primary parts. First is a recurrent audio encoder which produces contextual features. Second - a recurrent decoder with dot-product encoder-decoder attention uses these features to predict final blendshapes vectors. As in the aforementioned paper, we initialize decoder states with the outputs of the previous state. Figure 2 (b) depicts the entire architecture of the Seq2seq model.

For training, we implement simple MSE loss with no additional continuous and variance losses as in mentioned work for several reasons. To begin with, lip motion does not need to be as varied as body motion. As a result, we were able to give up on variance loss. Second, our experiments revealed that predicted motion is already smooth enough. Although, there are jerks between sequences. To eliminate them, we use the Savitzky–Golay filter.

3.5 Training

To train our models we use the PytorchLightning framework. We use every 10th sample of the first speaker data as validation data. And also every 10th sample of the second speaker as test data. Thus, we have 1800 and 200 samples of first data as train and validation respectively along with 450 samples of the second speaker for additional train data. The remaining 50 samples are used as a test subset. All our models were trained with Adam optimizer with a learning rate set to 1e-3. We also use the early stopping technique with a focus on validation loss.

4 Experiments

In this section we first present our experimental setup, then we evaluate our models and discuss the efficacy of our approaches.

We train our models on several subsets of data to find the dependency of generated animation quality on data quantity. Let us define designations for our models and datasets. There are feedforward (ff) and sequence-to-sequence (s2s) models trained on audio features from Wav2vec2 (w2v) and transferred Mel spectrograms via AutoVC (avc). There are 4 different models in total: ff_avc, ff_w2v, s2s_avc and s2s_w2v. We also train our models on different breakdowns of the training dataset: subsets 1-3 consist of various tracks of the first speaker with 450, 900, and 1800 samples respectively. It is worth mentioning that subsets were recorded separately from each other in several takes with a significant time gap between them. Subset 4 contains all the training data for the first speaker (like subset 3), along with 450 samples of the second speaker's tracks. Therefore, there are 16 experiments in total.

Evaluation metrics For quantifying the performance of our method, we compute the following three metrics.

- $\mathbf{L_2}$ **distance:** We calculate simple L_2 distance between each generated blendshapes vector and corresponding target one;
- **Mean landmark distance (LMD):** Following the method first described in (Chen et al., 2018) and utilizing the dlib model (King, 2009), we detect the sequences of landmark positions from the 3D face model animation based on real blendshapes, collected from a real speaker, and separately detect landmark positions from animation based on the blendshapes, predicted from corresponding speakers speech. As in (Zhou et al., 2020) we also normalize landmarks over mouth width. Then we calculate the distance between the resulting sequences.
- **SyncNet confidence and minimum distance:** Following (Chung and Zisserman, 2016) we use SyncNet model to obtain audio and video embeddings and measure synchronization between audio sequence and facial movements. To evaluate the change in behavior of this metric on the animated 3D model as opposed to the real face we also take additional measurements of this metric on permuted audio sequences and animations, where the audio is intentionally taken from one track and the facial movements are from a completely different track. See section 4.2 for more details.

4.1 LMD and L2

Here we show tables with results for different experiments. Firstly, standard L_2 norm and landmark distance metrics can be calculated between real blendshapes and predicted ones. The results are showed in tables 1 and 2. To understand the trustworthiness of LMD we also calculated the distance for permuted corresponding samples. This test showed an increase in the measured distance when applied to asynchronous data. Here LMD for permuted data was equal to 0.75.

model	subset_1	subset_2	subset_3	subset_4
ff_avc	2.79	2.37	2.62	1.50
ff_w2v	2.58	2.14	2.82	0.77
s2s_avc	2.17	2.06	2.24	0.95
s2s_w2v	2.28	1.90	2.54	0.77

Table 1: L2 distance, 1e-3

subset_1	subset_2	subset_3	subset_4
0.590	0.549	0.543	0.467
0.453	0.450	0.460	0.356
0.546	0.555	0.506	0.468
0.462	0.463	0.500	0.377

Table 2: LMD

Here we can see the correlation between L_2 and LMD. It's also worth noting that as the amount of data for one speaker increases, the metrics do not improve. Adding extra data, especially for wav2vec features, will only degrade the output. Only by including data from a second speaker (subset 4) are we able to enhance metrics significantly. This impact could be attributable to two factors: the test dataset and subset 4 share the same speaker, or the model's generalization ability has improved. To prove this hypothesis, we test our models on different voices in section 4.3. It is also worth noticing that wav2vec features give better results and the difference between seq2seq and feedforward models is not noticeable. That could mean that lip motion depends only on a short context.

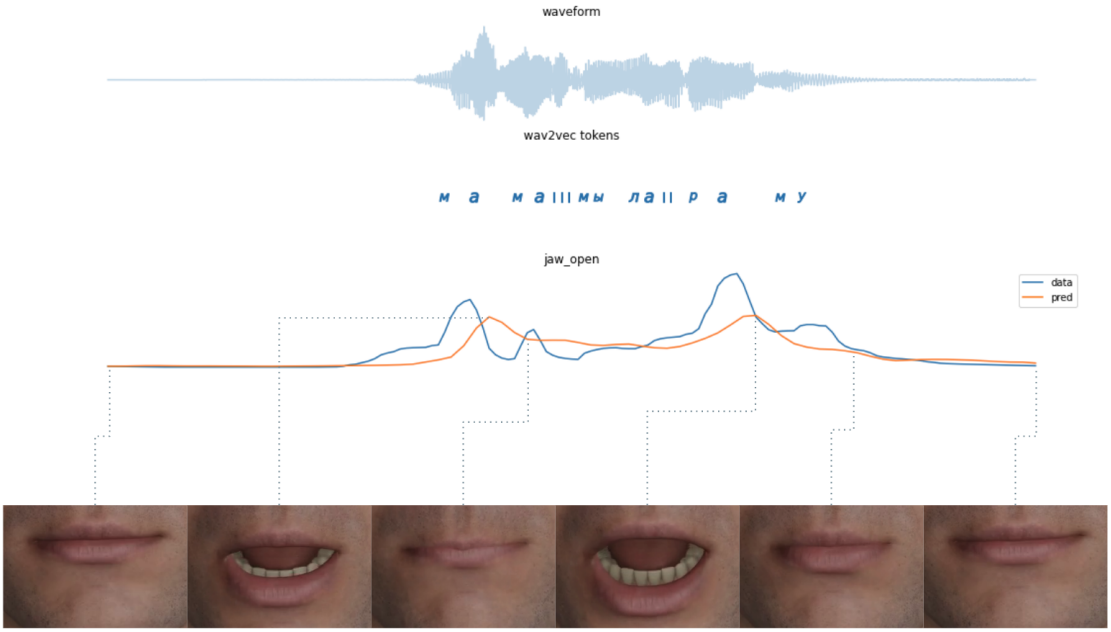


Figure 3: Sample of our model

In figure 3, we demonstrate how one of our models operates on a test sample. For this example, we use a feedforward model with a wav2vec audio encoder (ff_w2v) trained only on a single speaker (subset 3). There are no such phrases nor this speaker in the training dataset.

Figure 3 contains the input raw audio, corresponding tokens acquired by wav2vec, values of *jaw_open* blendshape values for real data (blue) and predicted result vakue (orange) along with several relevant 3D model renders. Despite modest differences in blendshape value lines, the model was able to reproduce the majority of phonemes.

Even though it has lost the second peak corresponding to the second vowel, this difference did not strike the eye during manual viewing of the produced video, but was plainly visible upon closer study.

4.2 Evaluation of SyncNet metric

Next we consider SyncNet confidence and distance collected during this experiment. The results can be seen in section tables 3 and 4.

model	subset_1	subset_2	subset_3	subset_4
ff_avc	10.21	9.89	9.61	9.24
ff_w2v	9.17	9.25	8.89	8.39
s2s_avc	10.36	10.61	9.69	9.12
s2s_w2v	9.64	9.54	9.91	8.54

Table 3: SyncNet distance

subset_1	subset_2	subset_3	subset_4
3.71	3.94	4.15	4.76
5.05	4.85	5.31	5.65
3.37	3.23	4.25	4.54
4.43	4.23	4.21	5.37

Table 4: SyncNet confidence

The SyncNet confidences also show growth if we add the second speaker. It also proves the assumption that the wav2vec features are better and the seq2seq model does not outperform the more simple feedforward model.

To further justify the application of the SyncNet metric to evaluate our model, we conducted the following experiment on the test dataset. First, we used all 61 blendshapes in the model to assess confidence and distance over the original test tracks with no alterations (Table 5, *All BS* column). Following that, we

Metric	All BS	Mouth BS	Mouth BS / wrong track
Distance	8.55	8.58	11.95
Confidence	5.69	5.66	2.34

Table 5: SyncNet distance and confidence on consistent and inconsistent tracks

used the same tracks but limited the number of blendshapes to those corresponding to the mouth region motions, i.e., to 33. (Table 5, *Mouth BS* column). Finally, after animating the same 33 blendshapes, we swapped their respective audio tracks at random. (Table 5, *Mouth BS / wrong track* column). Finally, within each group, the confidence and distance values obtained during each experiment are averaged.

As a result of these findings, limiting the number of blendshapes used in our model's training and inference has no significant impact on the metric's behavior. Furthermore, the metric unambiguously identifies cases of audio recording substitution or divergence.

4.3 Synthetic data

In addition to the recordings of a real speech, artificially synthesized speech can also be fed into our model. As an example, we dubbed 20 longer tracks with the voices of Alena and Dorofeev from Tinkoff's VoiceKit speech synthesis service, and then added the same lines taken from a human speaker for comparison. The SyncNet scores can be seen in the tables 6 and 7. For each model, we use the subset with the best score.

model	Real	Alena	Dorofeev
ff_avc (subset_1)	9.50	11.44	10.85
ff_w2v (subset_1)	9.11	11.17	11.22
s2s_avc (subset_3)	9.60	12.66	12.22
s2s_w2v (subset_4)	8.88	10.96	11.24

Table 6: SyncNet distance for real and synthetic recordings

Real	Alena	Dorofeev
3.30	3.14	4.26
3.82	3.61	4.13
3.06	1.83	2.67
3.82	3.28	3.54

Table 7: SyncNet confidence for real and synthetic recordings

When compared to samples from the test dataset, this result shows some metric degradation on these tracks. This includes the voice of the real person. The main difference, we believe, lies in the length of the tracks. The test dataset is primarily comprised of short phrases, whereas these tracks are at least twice as long. Visual inspection does not reveal a significant reduction in motion quality.

Despite this, synthetic voices can sometimes outperform natural voices in terms of SyncNet scores. It could be explained by the synthetic voice's speech smoothness. Women's synthetic voices, on the other hand, have a significant drop in scores, possibly due to the different timbre. It's also worth noting that the best results came from different subsets. It could lead to the conclusion that the best results on subset 4 were obtained in section 4.1 due to the presence of the same speaker, and that these models could be trained on just one speaker.

We also try to find out an explanation of SyncNet behavior. We manually examine the phrase where models have been scored differently. In the prediction with a lower score, the mouth did not close completely, while in the other it did. Nevertheless, even the first model was relatively correct in reproducing phonemes visually.

4.4 User study

We also provided short blind comparison between some of the models. We record a quite long phrase with the length of 22 seconds for synthetic Dorofeev's synthetic voice. Then we render videos for all models and subjectively pick the best four of them. To compare models left we ask users to choose the best from them in a following way. First, we pick a first pair of videos randomly and show them to the

user simultaneously to choose the best of two. Then, we do the same with a second pair left. Finally, we ask user to pick the best video between the winners of a previous stage. The results of the user study are shown in table 8.

In addition to the model's victory in the second stage when compared to the winner of another pair (first column - Abs Winner), we collect statistics on how many times the model passed to the next stage regardless of the model's victory in the second stage (second column - Pairwise winner).

model	Abs Winner	Pairwise winner
ff_avc (subset_1)	13	**39**
ff_w2v (subset_4)	8	28
ff_w2v (subset_1)	**14**	36
s2s_w2v (subset_4)	8	26

Table 8: User study

More simple models performed better in a user study on a synthetic phrase. This result is highly correlated with SyncNet scores on the same voice. This observation could provide yet another reason to use SyncNet as an objective metric for facial animation generation.

5 Conclusion

We propose a simple approach for automatic lip-sync for 3D face models. We show that good audio features are more useful than a complex model in this task and that no lengthy context is required. We also make some observations about the audio encoders that are used and data dependency. Our method has its flaws, and generated motions aren't perfect, but they're good enough to avoid catching the eye and annoying.

We are also looking for ways to make predicted motion even better. First, we are thinking about conducting a more in-depth user study of our models. Second, we are looking for a way to incorporate more data from videos with expressive facial movements.

We believe that our work will aid in the creation of automatic character animation and will serve as a solid foundation for future research.

6 Acknowledgements

The reported study was funded by RFBR according to the research project № 20-31-90051

References

Najwa Alghamdi, Steve Maddock, Ricard Marxer, Jon Barker, and Guy J Brown. 2018. A corpus of audio-visual lombard speech with frontal and profile views. *The Journal of the Acoustical Society of America*, 143(6):EL523–EL529.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Volker Blanz and Thomas Vetter. 1999. A morphable model for the synthesis of 3d faces. // *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, P 187–194.

Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. 2013. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425.

Lele Chen, Zhiheng Li, Ross K Maddox, Zhiyao Duan, and Chenliang Xu. 2018. Lip movements generation at a glance. // *Proceedings of the European Conference on Computer Vision (ECCV)*, P 520–535.

Lele Chen, Guofeng Cui, Celong Liu, Zhong Li, Ziyi Kou, Yi Xu, and Chenliang Xu. 2020. Talking-head generation with rhythmic head motion. // *ECCV*.

Joon Son Chung and Andrew Zisserman. 2016. Out of time: automated lip sync in the wild. // *Asian conference on computer vision*, P 251–263. Springer.

Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael J Black. 2019. Capture, learning, and synthesis of 3d speaking styles. // *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, P 10101–10111.

Tero Karras, Timo Aila, Samuli Laine, Antti Herva, and Jaakko Lehtinen. 2017. Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Transactions on Graphics (TOG)*, 36(4):1–12.

Davis E. King. 2009. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10(60):1755–1758.

Vladislav Korzun, Dimov Ilya, and Zharkov Andrew. 2021. Audio and text-driven approach for conversational gestures generation. *Dialogue*, 20:425–432.

Taras Kucherenko, Dai Hasegawa, Gustav Eje Henter, Naoshi Kaneko, and Hedvig Kjellström. 2019. Analyzing input and output representations for speech-driven gesture generation. // *Proceedings of the ACM International Conference on Intelligent Virtual Agents*, IVA '19, P 97–104.

John P Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H Pighin, and Zhigang Deng. 2014. Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, 1(8):2.

Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. 2017. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1.

Masahiro Mori, Karl F. MacDorman, and Norri Kageki. 2012. The uncanny valley [from the field]. *IEEE Robotics Automation Magazine*, 19(2):98–100.

Kaizhi Qian, Yang Zhang, Shiyu Chang, Xuesong Yang, and Mark Hasegawa-Johnson. 2019. Autovc: Zero-shot voice style transfer with only autoencoder loss. // *International Conference on Machine Learning*, P 5210–5219. PMLR.

Abraham Savitzky and Marcel JE Golay. 1964. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639.

Justus Thies, Mohamed Elgharib, Ayush Tewari, Christian Theobalt, and Matthias Nießner. 2020. Neural voice puppetry: Audio-driven facial reenactment. // *European conference on computer vision*, P 716–731. Springer.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Yang Zhou, Xintong Han, Eli Shechtman, Jose Echevarria, Evangelos Kalogerakis, and Dingzeyu Li. 2020. Makelttalk: speaker-aware talking-head animation. *ACM Transactions on Graphics (TOG)*, 39(6):1–15.