Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion

TERO KARRAS, NVIDIA
TIMO AILA, NVIDIA
SAMULI LAINE, NVIDIA
ANTTI HERVA, Remedy Entertainment
JAAKKO LEHTINEN, NVIDIA and Aalto University

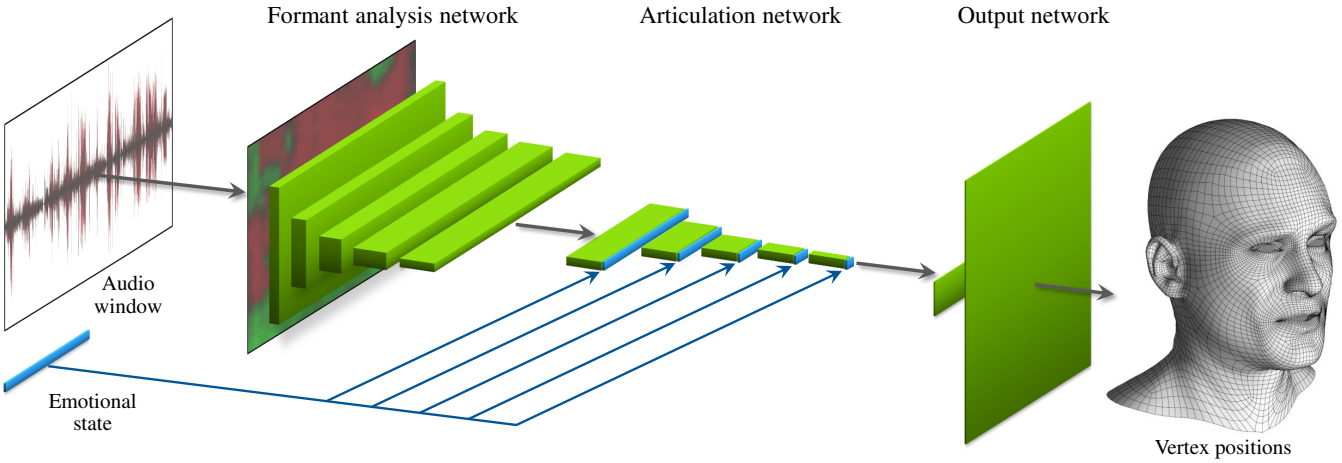

Fig. 1. Our deep neural network for inferring facial animation from speech. The network takes approximately half a second of audio as input, and outputs the 3D vertex positions of a fixed-topology mesh that correspond to the center of the audio window. The network also takes a secondary input that describes the emotional state. Emotional states are learned from the training data without any form of pre-labeling.

We present a machine learning technique for driving 3D facial animation by audio input in real time and with low latency. Our deep neural network learns a mapping from input waveforms to the 3D vertex coordinates of a face model, and simultaneously discovers a compact, latent code that disambiguates the variations in facial expression that cannot be explained by the audio alone. During inference, the latent code can be used as an intuitive control for the emotional state of the face puppet.

We train our network with 3–5 minutes of high-quality animation data obtained using traditional, vision-based performance capture methods. Even though our primary goal is to model the speaking style of a single actor, our model yields reasonable results even when driven with audio from other speakers with different gender, accent, or language, as we demonstrate with a user study. The results are applicable to in-game dialogue, low-cost localization, virtual reality avatars, and telepresence.

CCS Concepts: • **Computing methodologies** → **Animation**; **Neural networks**; *Supervised learning by regression*; *Learning latent representations*;

Additional Key Words and Phrases: Facial animation, deep learning, audio

ACM Reference format:
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 Trans. Graph.* 36, 4, Article 94 (July 2017), 12 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073658

0730-0301/2017/7-ART94 $15.00
DOI: http://dx.doi.org/10.1145/3072959.3073658

1 INTRODUCTION

Expressive facial animation is an essential part of modern computer-generated movies and digital games. Currently, vision-based performance capture, i.e., driving the animated face with the observed motion of a human actor, is an integral component of most production pipelines. While the quality obtainable from capture systems is steadily improving, the cost of producing high-quality facial animation remains high. First, computer vision systems require elaborate setups and often also labor-intensive cleanup and other processing steps. A second, less obvious issue is that whenever new shots are recorded, the actors need to be on location, and ideally also retain their appearance. This may be challenging if, for example, another role requires growing a beard.

While audio-based performance capture algorithms are unlikely to ever match the quality of vision systems, they offer complementary strengths. Most importantly, the tens of hours of dialogue

spoken by in-game characters in many modern games is much too expensive to produce using vision-based systems. Consequently, common practice is to produce only key animations, such as cinematics, using vision systems, and rely on systems based on audio and transcript for producing the bulk of in-game material. Unfortunately, the quality of the animation produced by such systems currently leaves much to be desired. Further, emerging real-time applications in telepresence and virtual reality avatars present additional challenges due to the lack of transcript, wide variability in user voices and physical setups, and stringent latency requirements.

Our goal is to generate plausible and expressive 3D facial animation based exclusively on a vocal audio track. For the results to look natural, the animation must account for complex and codependent phenomena including phoneme coarticulation, lexical stress, and interaction between facial muscles and skin tissue [Edwards et al. 2016]. Hence, we focus on the entire face, not just the mouth and lips. We adopt a data-driven approach, where we train a deep neural network in an end-to-end fashion to replicate the relevant effects observed in the training data.

At first, the problem may seem intractable because of its inherent ambiguity—the same sounds can be uttered with vastly different facial expressions, and the audio track simply does not contain enough information to distinguish between the different variations [Petrushin 1998]. While modern convolutional neural networks have proven extremely effective in various inference and classification tasks, they tend to regress toward the mean if there are ambiguities in the training data.

To tackle these problems, we present three main contributions:

- A convolutional network architecture tailored to effectively process human speech and generalize over different speakers (Sections 3.1 and 3.2).
- A novel way to enable the network to discover variations in the training data that cannot be explained by the audio alone, i.e., apparent emotional state (Section 3.3).
- A three-way loss function to ensure that the network remains temporally stable and responsive under animation, even with highly ambiguous training data (Section 4.3).

Our method produces expressive 3D facial motion from audio in real time and with low latency. To retain maximal independence from the details of the downstream animation system, our method outputs the per-frame positions of the control vertices of a fixed-topology facial mesh. Alternative encodings [Lewis et al. 2014] such as blend shapes or non-linear rigs can be introduced at later pipeline stages, if needed for compression, rendering, or editability. We train our model with 3–5 minutes of high-quality footage obtained using traditional, vision-based performance capture methods. While our goal is to model the speaking style of a single actor, our model yields reasonable results even when driven with audio from other speakers with different gender, accent, or language.

We see uses for this technology in in-game dialogue, low-cost localization, virtual reality, and telepresence. It could also prove useful in accommodating small script changes even in cinematics.

2 RELATED WORK

We will review prior art in systems whose input is audio or text and output is 2D video or 3D mesh animation. We group the approaches into linguistic and machine learning based models, and also review methods that support apparent emotional states.

Models based on linguistics. A large body of literature exists for analyzing and understanding the structure of language, and translating it to anatomically credible facial animation [Lewis 1991; Mattheyses and Verhelst 2015]. Typically, an audio track is accompanied with a transcript that helps to provide explicit knowledge about the phoneme content. The animation is then based on the visual counterpart of phonemes called visemes [Fisher 1968] through complex rules of coarticulation. A well-known example of such a system is the dominance model [Cohen and Massaro 1993; Massaro et al. 2012]. In general, there is a many-to-many mapping between phonemes and visemes, as implemented in the dynamic visemes model of Taylor et al. [2012] and in the recent work dubbed JALI [Edwards et al. 2016]. JALI factors the facial animation to lip and jaw movements, based on psycholinguistic considerations, and is able to convincingly reproduce a range of speaking styles and apparent emotional states independently of the actual speech content.

A core strength of these methods is the explicit control over the entire process, which makes it possible to, e.g., explicitly guarantee that the mouth closes properly when the puppet is spelling out a bilabial (`/m/`, `/b/`, `/p/`) or that the lower lip touches the upper teeth with labiodentals (`/f/`, `/v/`). Both are difficult cases even for vision-based capture systems. The weaknesses include the accumulated complexity of the process, language-specific rules, need for a near-perfect transcript for good quality (typically done manually [Edwards et al. 2016]), inability to react convincingly to non-phoneme sounds, and lack of a principled way to animate other parts of the face besides the jaw and lips.

FaceFX (`www.facefx.com`) is a widely used commercial package that implements the most widely used linguistic models, including the dominance model.

Models based on machine learning. Here we will group the systems primarily based on how they use machine learning.

Voice puppetry [Brand 1999] is driven exclusively using audio, and does not perform explicit analysis of the structure of the speech. In the training stage, it estimates a hidden Markov model (HMM) based on the observed dynamics of the face in a video. During inference, the HMM is sampled and the most probable sequence is synthesized through trajectory optimization that considers the entire utterance. Subsequent work has improved the trajectory sampling [Anderson et al. 2013; Wang and Soong 2015] and replaced HMM, which does piecewise linear approximation, with alternative representations such as Gaussian process latent variable model [Deena and Galata 2009; Deena et al. 2013], hidden semi-Markov model [Schabus et al. 2014], or recurrent networks [Fan et al. 2016].

Alternatively, machine learning has been used for learning coarticulation [Deng et al. 2006; Ezzat et al. 2002], followed by a concatenation stage to synthesize animation, or for mapping between various stages, such as phoneme classification [Kshirsagar and Magnenat-Thalmann 2000], mapping text to phonemes and phonemes to visemes

[Malcangi 2010], or mapping input audio features to control parameters of a Gaussian mixture model [Hofer and Richmond 2010].

Given that our goal is to produce 3D animation based on audio, we are not inherently interested in the intermediate representations. Instead, we would like to formulate the entire mapping as an end-to-end optimization task. The early experiments with neural networks used audio to directly drive the control parameters of an animated 3D mesh [Hong et al. 2002; Massaro et al. 1999; Öhman and Salvi 1999], but the networks back then were necessarily of trivial complexity. We revisit this end-to-end formulation with deep convolutional networks, latest training methods, and problem-specific contributions.

It is unfortunately difficult to do apples-to-apples comparisons against previous systems that use machine learning. The majority of work has focused on reusing captured video frames with concatenation, blending, and warping. Such image-based methods (e.g., [Anderson et al. 2013; Deena et al. 2013; Ezzat et al. 2002; Fan et al. 2016; Liu and Ostermann 2011; Wang and Soong 2015]) can produce realistic results, but typically need to store a large corpus of frames and are not directly applicable to applications such as games or VR that need to animate and render 3D models from free viewpoints, typically with flexible identities. The systems that do produce 3D are often based on text input instead of audio [Schabus et al. 2014; Wampler et al. 2007]. We are not aware of any publicly available implementations of methods that would suit our needs, e.g. [Deng et al. 2006], and it would not be fair to compare against the result videos of old methods since the quality standards have risen dramatically in sync with the available computing power.

Extracting and controlling the emotional state. The automatic separation of speech and emotional state has been studied by several authors. Chuang et al. [2002] build on the work of Tenenbaum and Freeman [2000] and use a bilinear model for separating the apparent emotional state from speech visemes. This was later generalized in closely related topics to multilinear [Vasilescu and Terzopoulos 2003; Wampler et al. 2007] and non-linear models [Elgammal and Lee 2004], as well as independent component analysis [Cao et al. 2003].

Cao et al. [2005] extract emotions using support vector machines, and synthesize 3D animations based on speech and apparent emotional state. They compute mappings between a set of pre-defined emotional states, and let the user specify the state to be used for animation synthesis. Deng et al. [2006] compute an eigenspace for expressions based on a pre-defined set of emotional states. Wampler et al. [2007] also allow a user-specified emotional state. Anderson et al. [2013] use cluster adaptive training to derive a basis for the emotional state so that it can be interpolated and extrapolated. They also present a user study rating the level of realism in emotion synthesis, covering several methods [Cao et al. 2005; Liu and Ostermann 2011; Melenchon et al. 2009]. Jia et al. [2014] use neural networks to learn a mapping from PAD (pleasure-displeasure, arousal-nonarousal, and dominance-submissiveness) parameters to facial expressions.

What distinguishes our method from all these efforts is that instead of having pre-defined categories for emotions, we let the network learn a latent, low-dimensional descriptor that allows it to explain the data. The descriptor's parameter combinations can be later assigned any number of semantic meanings, e.g., "sad" or "happy", but those have no role in the learning process itself.

Formant analysis network

Layer type	Kernel	Stride	Outputs	Activation
Autocorrelation	-	-	1×64×32	-
Convolution	1×3	1×2	72×64×16	ReLU
Convolution	1×3	1×2	108×64×8	ReLU
Convolution	1×3	1×2	162×64×4	ReLU
Convolution	1×3	1×2	243×64×2	ReLU
Convolution	1×2	1×2	256×64×1	ReLU

Articulation network

Layer type	Kernel	Stride	Outputs	Activation
Conv + concat	3×1	2×1	(256+E)×32×1	ReLU
Conv + concat	3×1	2×1	(256+E)×16×1	ReLU
Conv + concat	3×1	2×1	(256+E)× 8×1	ReLU
Conv + concat	3×1	2×1	(256+E)× 4×1	ReLU
Conv + concat	4×1	4×1	(256+E)× 1×1	ReLU

Output network

Layer type	Kernel	Stride	Outputs	Activation
Fully connected	-	-	150	Linear
Fully connected	-	-	15066	Linear

Table 1. Detailed breakdown of each part of our network. The autocorrelation layer performs fixed-function analysis of the input audio clip (Section 3.2). In the articulation network, we concatenate an E-dimensional vector representing the emotional state to the output of each convolution layer after the ReLU activation, i.e., $\max(0, \cdot)$. The fully-connected layers at the end expand the $256+E$ abstract features to 3D positions of 5022 vertices. In total, the network has 3.7 million scalar weights when $E = 16$.

Residual motion. Almost, but not quite, human-like appearance and motion is often perceived as particularly creepy, an effect known as the uncanny valley [Mori 1970]. Several authors have tried alleviating the effect by deducing additional motion, such as head movements, eye saccades or blinks from audio (e.g., [Deng et al. 2004; Marsella et al. 2013]). In our work, we assume that such residual motion is driven by higher-level procedural controls, such as a game engine, and limit our scope to motion that is directly related to articulation.

3 END-TO-END NETWORK ARCHITECTURE

We will now describe the architecture of our network, along with details on audio processing and the separation of emotional state from the speech content.

Given a short window of audio, the task of our network is to infer the facial expression at the center of the window. We represent the expression directly as per-vertex difference vectors from a neutral pose in a fixed-topology face mesh. Once the network is trained, we animate the mesh by sliding a window over a vocal audio track and evaluate the network independently at each time step. Even though the network itself has no memory of past animation frames, it produces temporally stable results in practice.

3.1 Architecture overview

Our deep neural network consists of one special-purpose layer, 10 convolutional layers, and 2 fully-connected layers. We divide it in three conceptual parts, illustrated in Figure 1 and Table 1.

We start by feeding the input audio window to a *formant analysis network* to produce a time-varying sequence of speech features that will subsequently drive articulation. The network first extracts raw formant information using fixed-function autocorrelation analysis (Section 3.2) and then refines it with 5 convolutional layers. Through training, the convolutional layers learn to extract short-term features that are relevant for facial animation, such as intonation, emphasis, and specific phonemes. Their abstract, time-varying representation is the output of the 5th convolutional layer.

Next, we feed the result to an *articulation network* that consists of 5 further convolutional layers that analyze the temporal evolution of the features and eventually decide on a single abstract feature vector that describes the facial pose at the center of the audio window. As a secondary input, the articulation network accepts a (learned) description of *emotional state* to disambiguate between different facial expressions and speaking styles (Section 3.3). The emotional state is represented as an E-dimensional vector that we concatenate directly onto the output of each layer in the articulation network, enabling the subsequent layers to alter their behavior accordingly.

Each layer l outputs $F_l \times W_l \times H_l$ activations, where F_l is the number of abstract feature maps, W_l is dimension of the time axis, and H_l is the dimension of the formant axis. We use strided 1×3 convolutions in the formant analysis network to gradually reduce H_l while increasing F_l, i.e., to push raw formant information to the abstract features, until we have $H_l = 1$ and $F_l = 256$ at the end. Similarly, we use 3×1 convolutions in the articulation network to decrease W_l, i.e., to subsample the time axis by combining information from the temporal neighborhood. We chose the specific parameters listed in Table 1 because we found them to consistently perform well in our datasets while leading to reasonable training times. The results are not hugely sensitive to the exact number of layers or feature maps, but we found it necessary to organize the convolutions in two distinct phases to avoid overfitting. Importantly, the formant analysis network performs the same operation at every point along the time axis, so that we can benefit from the same training samples at different time offsets.

The articulation network outputs a set of 256+E abstract features that together represent the desired facial pose. We feed these features to an *output network* to produce the final 3D positions of 5022 control vertices in our tracking mesh. The output network is implemented as a pair of fully-connected layers that perform a simple linear transformation on the data. The first layer maps the set of input features to the weights of a linear basis, and the second layer calculates the final vertex positions as a weighted sum over the corresponding basis vectors. We initialize the second layer to 150 precomputed PCA components that together explain approximately 99.9% of the variance seen in the training data. In principle, we could use a fixed basis to effectively train the earlier layers to output the 150 PCA coefficients. However, we have found that allowing the basis vectors themselves to evolve during training yields slightly better results.

3.2 Audio processing

The main input to our network is the speech audio signal that we convert to 16 kHz mono before feeding it to the network. In our experiments, we normalize the volume of each vocal track to utilize the full [-1,+1] dynamic range, but we do not employ any other kind of processing such as dynamic range compression, noise reduction, or pre-emphasis filter.

The autocorrelation layer in Table 1 converts the input audio window to a compact 2D representation for the subsequent convolutional layers. Our approach is inspired by the source–filter model of speech production [Benzeghiba et al. 2007; Lewis 1991], where the audio signal is modeled as a combination of a *linear filter* (vocal tract) and an *excitation signal* (vocal cords). The resonance frequencies (formants) of the linear filter are known to carry essential information about the phoneme content of the speech. The excitation signal indicates the pitch, timbre, and other characteristics of the speaker's voice, which we hypothesize to be far less important for facial animation, and thus we focus primarily on the formants to improve the generalization over different speakers.

The standard method for performing source–filter separation is linear predictive coding (LPC). LPC breaks the signal into several short frames, solves the coefficients of the linear filter for each frame based on the first K autocorrelation coefficients, and performs inverse filtering to extract the excitation signal. The resonance frequencies of the filter are entirely determined by the autocorrelation coefficients, so we choose to skip most of the processing steps and simply use the autocorrelation coefficients directly as our representation of the instantaneous formant information. This makes intuitive sense, since the autocorrelation coefficients essentially represent a compressed version of signal whose frequency content approximately matches the power spectrum of the original signal. The representation is a natural fit for convolutional networks, as the layers can easily learn to estimate the instantaneous power of specific frequency bands.

In practice, we use 520ms worth of audio as input, i.e., 260ms of past and future samples with respect to the desired output pose. We chose this value to capture relevant effects like phoneme coarticulation without providing too much data that might lead to overfitting. The input audio window is divided into 64 audio frames with 2x overlap, so that each frame corresponds to 16ms (256 samples) and consecutive frames are located 8ms (128 samples) apart. For each audio frame, we remove the DC component and apply the standard Hann window to reduce temporal aliasing effects. Finally, we calculate $K = 32$ autocorrelation coefficients to yield a total of 64×32 scalars for the input audio window. Although much fewer autocorrelations, e.g. $K = 12$, would suffice to identify individual phonemes, we choose to retain more information about the original signal to allow the subsequent layers to also detect variations in pitch.

Our approach differs from most of the previous work on speech recognition, where the analysis step is typically based on a specialized techniques such as Mel-frequency cepstral coefficients (MFCC), perceptual linear prediction (PLP), and RASTA filtering [Benzeghiba et al. 2007]. These techniques have enjoyed wide adoption mainly because they lead to good linear separation of phonemes and consequently work well with hidden Markov models. In our early tests,

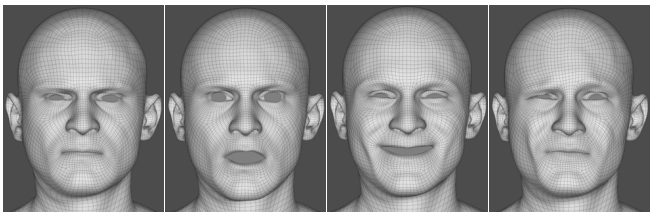

Fig. 2. What does silence look like? These are example frames from our training set where the actor does not speak.

we tried several different input data representations and observed that the autocorrelation coefficients were clearly superior for our case.

3.3 Representation of emotional states

Inferring facial animation from speech is an inherently ambiguous problem, because the same sound can be produced with very different facial expressions. This is especially true with the eyes and eyebrows, since they have no direct causal relationship with sound production. Such ambiguities are also problematic for deep neural networks, because the training data will inevitably contain cases where nearly identical audio inputs are expected to produce very different output poses. Indeed, Figure 2 shows several examples of conflicting training data where the input audio clip consists entirely of silence. If the network has nothing else to work with besides the audio, it will learn to output the statistical mean of the conflicting outputs.

Our approach for resolving these ambiguities is to introduce a secondary input to the network. We associate a small amount of additional, latent data with each training sample, so that the network has enough information to unambiguously infer the correct output pose. Ideally, this additional data should encode all relevant aspects of the animation in the neighborhood of a given training sample that cannot be inferred from the audio itself, including different facial expressions, speaking styles, and coarticulation patterns. Informally, we wish the secondary input to represent the *emotional state* of the actor. Besides resolving ambiguities in the training data, the secondary input is also highly useful for inference—it allows us to mix and match different emotional states with a given vocal track to provide powerful control over the resulting animation.

One way to implement emotional states would be to manually label or categorize the training samples based on the apparent emotion [Anderson et al. 2013; Cao et al. 2005; Deng et al. 2006; Wampler et al. 2007]. This approach is not ideal, however, because there is no guarantee that a pre-defined labeling actually resolves ambiguities in the training data to a sufficient degree. Instead of relying on predefined labels, we adopt a data-driven approach where the network automatically learns a succinct representation of the emotional state as a part of the training process. This allows us to extract meaningful emotional states even from in-character footage, as long as a sufficient range of emotions is present.

We represent the emotional state as an E-dimensional vector, where E is a tunable parameter that we set to 16 or 24 in our tests, and initialize the components to random values drawn from a Gaussian distribution. One such vector is allocated for each training sample, and we refer to the matrix that stores these latent variables as the *emotion database*. As illustrated in Figure 1, the emotional state is appended to the list of activations of all layers of the articulation network. This makes it a part of the computation graph of the loss function (Section 4.3), and as a trainable parameter, it gets updated along with the network's weights during backpropagation. The dimensionality of E is a tradeoff between two effects. If E is too low, the emotional states fail to disambiguate variations in the training data, leading to weak audio response. If E is too high, all emotional states tend to become too specialized to be useful for general inference (Section 5.1).

One potential concern with the emotion database is that unless we constrain it in a meaningful way, it might learn to explicitly store information that is also present in the audio. In a pathological case, it could store E blend shape weights that determine much of the facial expression, thus diminishing the role of audio and making the network useless for processing material not seen during training.

The information provided by the audio is limited to short-term effects within the 520ms interval by design. Consequently, a natural way to prevent the emotional states from containing duplicate information is to forbid them from containing short-term variation. Having the emotional states focus on longer-term effects is also highly desirable for inference—we want the network to produce reasonable animation even when the emotional state remains fixed. We can express this requirement by introducing a dedicated regularization term in our loss function to penalize quick variations in the emotion database, which leads to incremental smoothing of the emotional states over the course of training. One important limitation of our approach is that we cannot model blinking and eye motion correctly since they do not correlate with the audio and cannot be represented using the slowly varying emotional state.

While it may seem redundant to append the emotional state to all layers of the articulation network, we have found this to improve the results significantly in practice. We suspect that this is because the emotional state aims to control the animation on multiple abstraction levels, and the higher abstraction levels are generally more difficult to learn. Connecting to the earlier layers provides nuanced control over subtle animation features such as coarticulation, whereas connecting to the later layers provides more direct control over the output poses. It makes intuitive sense that the early stages of training will need to concentrate on the latter, while the later stages can concentrate on the former once the individual poses are reasonably well represented.

The learned emotional states are just data without semantic meanings ("sad", "happy", etc.). We defer the discussion about semantics to Section 5.1 because they do not play a role in the network architecture or training.

4 TRAINING

We will now describe the aspects relevant to training our network: how the training targets were obtained, what our dataset consists of, dataset augmentation, loss function, and training setup.

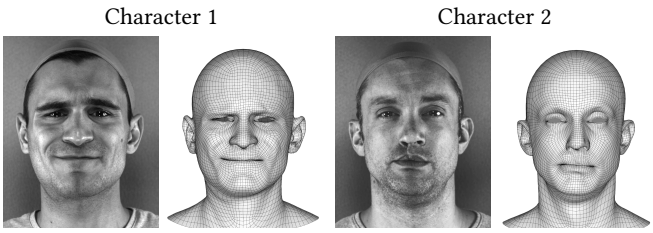

Fig. 3. The training targets were obtained using a vision-based pipeline that uses streams from 9 HD cameras to output 3D positions of animated control vertices for each frame.

4.1 Training targets

We obtained the 3D vertex positions used as training targets using the commercial DI4D PRO system (www.di4d.com) that employs nine synchronized video cameras at 30Hz to directly capture the nuanced interactions of the skull, muscles, fascia and skin of an actor, excluding high frequency details such as wrinkles. The benefit of this system is that it allows us to bypass complex and expensive facial rigging and tissue simulations for digital doubles. The input and output data is illustrated in Figure 3.

As the first step, an unstructured mesh with texture and optical flow data is reconstructed from the nine images captured for each frame. A fixed-topology template mesh, created prior to the capture work using a separate photogrammetry pipeline, is then projected on to the unstructured mesh and associated with the optical flow. The template mesh is tracked across the performance and any issues are fixed semi-automatically in the DI4DTrack software by a tracking artist. The position and orientation of the head are stabilized using a few key vertices of the tracking mesh. Finally, the vertex positions of the mesh are exported for each frame in the shot. These positions—or more precisely the deltas from a neutral pose—are the desired outputs of our network when given a window of audio during training.

One limitation of video-based capture setups is that they cannot capture the tongue since it is typically not visible in the images. While the exact control of the tongue is highly relevant for speech production, it is rarely visible so clearly that it would have to be animated in detail. We thus leave the tongue as well as wrinkles, eyes and residual head/body motion to higher level procedural animation controls.

4.2 Training dataset

For each actor, the training set consists of two parts: pangrams and in-character material. In general, the inference quality increases as the training set grows, but a small training set is highly desirable due to the cost of capturing high-quality training data. We feel that 3–5 minutes per actor represents a practical sweet spot.

Pangrams. This set attempts to cover the set of possible facial motions during normal speech for a given target language, in our case English. The actor speaks one to three pangrams, i.e., sentences that are designed to contain as many different phonemes as possible, in several different emotional tones to provide a good coverage of the range of expression.

In-character material. This set leverages the fact that an actor's performance of a character is often heavily biased in terms of emotional and expressive range for various dramatic and narrative reasons. In case of a movie or a game production, this material can be composed of the preliminary version of the script. Only the shots that are deemed to support the different aspects of the character are selected so as to ensure that the trained network produces output that stays in character even if the inference is not perfect, or if completely novel or out of character voice acting is encountered.

For Character 1, our training set consists of 9034 frames (5min 1s), of which 3872 come from pangrams and 5162 from in-character material. Additionally, 1734 frames are reserved for validation.

For Character 2, our training set consists of 6762 frames (3min 45s), of which 1722 come from pangrams and 5040 from in-character material. Additionally, we have 887 frames for validation.

4.3 Loss function

Given the ambiguous nature of our training data, we must take special care to define a meaningful loss function that we wish to optimize. We use a specialized loss function that consists of three distinct terms: a *position term* to ensure that the overall location of each output vertex is roughly correct, a *motion term* to ensure that the vertices exhibit the right kind of movement under animation, and a *regularization term* to discourage the emotion database from containing short-term variation.

Simultaneous optimization of multiple loss terms tends to be difficult in practice, because the terms can have wildly different magnitudes and their balance may change in unpredictable ways during training. The typical solution is to associate a pre-defined weight with each term to ensure that none of them gets neglected by the optimization. However, choosing optimal values for the weights can be a tedious process of trial and error and it typically needs to be repeated whenever the training set changes. To overcome these issues, we introduce a normalization scheme that automatically balances the loss terms with respect to their relative importance. As a result, we automatically put an equal amount of effort into optimizing each term and there is consequently no need to specify any additional weights.

Position term. Our primary error metric is the mean of squared differences between the desired output y and the output produced by the network $\hat{y}$. For a given training sample x, we express this using the position term $P(x)$:

$$P(x) = \frac{1}{3V} \sum_{i=1}^{3V} \left(y^{(i)}(x) - \hat{y}^{(i)}(x) \right)^2 \tag{1}$$

Here, V represents the total number of output vertices and $y^{(i)}$ denotes the ith scalar component of $y = (y^{(1)}, y^{(2)}, \ldots, y^{(3V)})$. The total number of output components is $3V$, because our network outputs the full 3D position for each vertex.

Motion term. Even though the position term ensures that the output of our network is roughly correct at any given instant in time, it is not sufficient to produce high-quality animation. We have found that training the network with the position term alone leads to a considerable amount of temporal instability, and the response

to individual phonemes is generally weak. This motivates us to optimize our network in terms of vertex motion as well: a given output vertex should only move if it also moves in the training data, and it should only move at the right time. We thus need a way to measure vertex motion as a part of our loss function, similar to Brand's work on HMMs [1999], where both position and velocity are optimized.

The standard approach for training neural networks is to iterate over the training data in minibatches, where each minibatch consists of B randomly selected training samples $x_1, x_2, \ldots, x_B$. To account for vertex motion, we draw the samples as $B/2$ *temporal pairs*, each consisting of two adjacent frames. We define operator $m[\cdot]$ as the finite difference between the paired frames. We can now define the motion term $M(x)$:

$$M(x) = \frac{2}{3V} \sum_{i=1}^{3V} \left(m\left[y^{(i)}(x)\right] - m\left[\hat{y}^{(i)}(x)\right] \right)^2 \tag{2}$$

The factor 2 appears because M is evaluated once per temporal pair.

Regularization term. Finally, we need to ensure that the network correctly attributes short-term effects to the audio signal and long-term effects to the emotional state as described in Section 3.3. We can conveniently define a regularization term for our emotion database using the same finite differencing operator as above:

$$R'(x) = \frac{2}{E} \sum_{i=1}^{E} m\left[e^{(i)}(x)\right]^2 \tag{3}$$

$e^{(i)}(x)$ denotes the ith component stored by the emotion database for training sample x. Note that this definition does not explicitly *forbid* the emotion database from containing short-term variation—it merely discourages excess variation *on the average*. This is important, because our training data contains legitimate short-term changes in the emotional state occasionally, and we do not want the network to incorrectly try to explain them based on the audio signal.

A major caveat with Eq. 3 is that $R'(x)$ can be brought arbitrarily close to zero by simply decreasing the magnitude of $e(x)$ while increasing the corresponding weights in the network. Drawing on the idea of batch normalization [Ioffe and Szegedy 2015], we remove this trivial solution by normalizing $R'(x)$ with respect to the observed magnitude of $e(x)$:

$$R(x) = R'(x) \Big/ \left(\frac{1}{EB} \sum_{i=1}^{E} \sum_{j=1}^{B} e^{(i)}(x_j)^2 \right) \tag{4}$$

Normalization. To balance our three loss terms, we leverage the properties of the Adam optimization method [Kingma and Ba 2014] that we use for training our network. In effect, Adam updates the weights of the network according to the gradient of the loss function, normalized in a component-wise fashion according to a long-term estimate of its second raw moment. The normalization makes the training resistant to differences in the magnitude of the loss function, but this is only true for the loss function as a whole—not for the individual terms. Our idea is to perform similar normalization for each loss term individually. Using the position term as an example, we estimate the second raw moment of $P(x)$ for each minibatch and maintain a moving average v_t^P across consecutive minibatches:

$$v_t^P = \beta \cdot v_{t-1}^P + (1 - \beta) \cdot \frac{1}{B} \sum_{j=1}^{B} P(x_j)^2 \tag{5}$$

Here, t denotes the minibatch index and β is a decay parameter for the moving average that we set to 0.99. We initialize $v_t^P = 0$ and correct our estimate to account for startup bias to get $\hat{v}_t^P = v_t^P / (1 - \beta^t)$. We then calculate the average $P(x)$ over the current minibatch and normalize the value according to $\hat{v}_t^P$:

$$\ell^P = \left(\frac{1}{B} \sum_{j=1}^{B} P(x_j) \right) \Big/ \left(\sqrt{\hat{v}_t^P} + \epsilon \right) \tag{6}$$

ϵ is a small constant that we set to 10^{-8} to avoid division by zero. Repeating Equations 5 and 6 for $M(\cdot)$ and $R(\cdot)$, we express our final loss function as a sum over the individual terms $\ell = \ell^P + \ell^M + \ell^R$. Although it would be possible to further fine-tune the importance of the loss terms through additional weights, we have not found this to be beneficial.

4.4 Training data augmentation

To improve temporal stability and reduce overfitting, we employ random *time-shifting* for our training samples. Whenever we present a minibatch to the network, we randomly shift the input audio window by up to 16.6ms in either direction (±0.5 frames at 30 FPS). To compensate, we also apply the same shift for the desired output pose through linear interpolation. We shift both training samples in a temporal pair by the same amount, but use different random shift amounts for different pairs. We also tried cubic interpolation of outputs, but it did not work as well as linear.

As a crucial step to improve generalization and avoid overfitting, we apply multiplicative noise to the input of every convolutional layer [Srivastava et al. 2014]. The noise has the same magnitude for every layer, and it is applied on a per-feature map basis so that all activations of a given feature map are multiplied by the same factor. We apply identical noise to paired training samples to get a meaningful motion term. The formula for our noise is $1.4^{\mathcal{N}(0,\,1)}$.

We do not apply any other kind of noise or augmentation on our training samples besides the time-shifting of input/outputs and multiplicative noise inside the network. We experimented with adjusting the volume, adding reverb (both long and short), performing time-stretching and pitch-shifting, applying non-linear distortion, random equalization, and scrambling the phase information but none of these improved the results further.

4.5 Training setup and parameters

We have implemented our training setup using Theano [Theano Development Team 2016] and Lasagne [Dieleman et al. 2015] that internally use cuDNN [Chetlur et al. 2014] for GPU acceleration.

We train the network for 500 epochs using Adam [Kingma and Ba 2014] with the default parameters. Each epoch processes all training samples in a randomized order in minibatches of 100 training samples (50 temporal pairs). The learning rate is ramped up tenfold using a geometric progression during the first training epoch, and

it is then decreased gradually according to $1/\sqrt{t}$ schedule. During the last 30 epochs we ramp the learning rate down to zero using a smooth curve, and simultaneously ramp Adam's β_1 parameter from 0.9 to 0.5. The ramp-up removes an occasional glitch where the network does not start learning at all, and the ramp-down ensures that the network converges to a local minimum. The total training time is 2h 55min for Character 1 and 1h 30min for Character 2 on an NVIDIA Pascal Titan X.

The network weights are initialized following He et al. [2015]. The emotion database is initialized with zero-mean Gaussian noise with $E = 16, \sigma = 0.01$ for Character 1 and $E = 24, \sigma = 0.002$ for Character 2. We had to hand-tune these parameters per actor, while other parameters required no tuning. The differences are probably explained by the fact that Character 1 mostly had the same expression throughout the training set, while Character 2 was much more lively and did various extraneous movements with his face.

5 INFERENCE AND RESULTS

Once trained, the network can be evaluated at arbitrary points in time by selecting the appropriate audio window, leading to facial animation at the desired frame rate.

On our Theano implementation, inference takes 6.3ms for a single frame and 0.2ms/frame when a batch of 100 frames is processed at once. Given that Theano is known for its large overheads, we are confident that it would be possible to push the single frame performance, which matters for real-time use, below 1ms by using a more efficient framework, e.g., cuDNN directly.

The latency of our method is determined by the audio window size, which currently reaches 260ms to the future. Coarticulation sets a lower bound for the look-ahead; we have confirmed experimentally that we can limit the look-ahead to 100ms during training with minor degradation in quality, even though some coarticulation effects are known to be longer [Schwartz and Savariaux 2014]. Shortening the look-ahead further than this leads to a quick drop in perceived responsiveness, so a realistic lower bound for the latency of our method therefore seems to be around 100ms. Methods relying on explicit trajectory optimization (e.g. [Brand 1999; Deena et al. 2013; Fan et al. 2016]) have substantially higher latency.

5.1 Emotional states

When inferring the facial pose for novel audio, we need to supply the network with an emotional state vector as a secondary input. As part of training, the network has learned a latent E-dimensional vector for each training sample, and our strategy is to mine this emotion database for robust emotion vectors that can be used during inference.

During training, the network attempts to separate out the latent information—i.e., everything that is not inferable from the audio alone—into the emotion database. However, this decomposition is not perfect and some amount of crosstalk remains between articulation and the overall expression. In practice, many of the learned emotion vectors are only applicable in the neighborhood of their corresponding training frames and are not necessarily useful for general inference. This is to be expected, because our training data will necessarily be too limited to cover all phonemes and coarticulation effects for every observed emotional state.

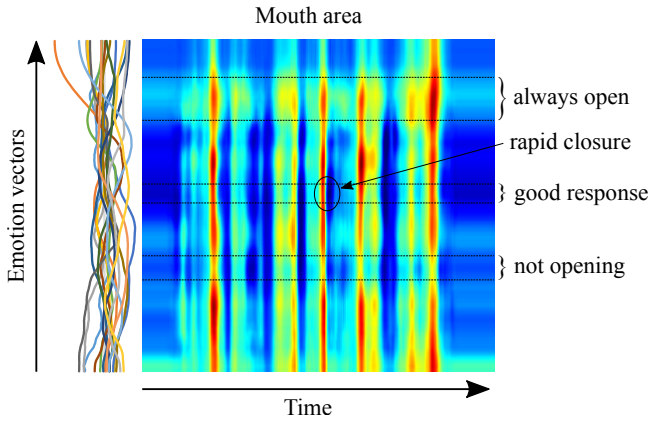

Fig. 4. Visualization of the opening of the mouth during the course of an audio clip (x-axis) under different constant emotion vector inputs. Each point on the y-axis represents a different emotion vector extracted from the learned database for the same clip. Blue and red indicate that the mouth is closed and open, respectively. We observe that many emotion vectors have problems with opening or closing the mouth properly. As the first step of our database mining process, we rapidly cull emotion vectors that do not respond well to audio in this way.

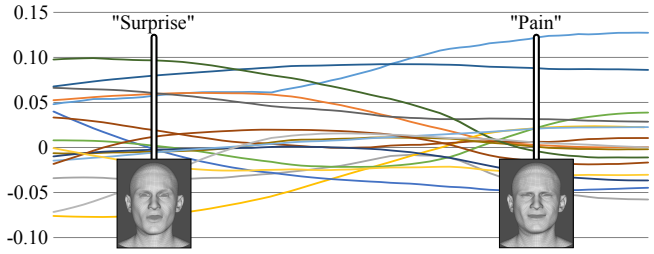

Fig. 5. Human-understandable semantic meanings for different emotion vectors are assigned manually by examining the animation they produce. The neural network itself is not interested in these semantic meanings; as far it is concerned, the emotion vector is just data that helps to disambiguate the audio.

We manually mine for robust emotion vectors using a three-step process. The main problem in most learned emotion vectors is that they deemphasize the motion of the mouth: when such a vector is used as a constant input when performing inference for novel audio, the apparent motion of the mouth is subdued. Our first step is therefore to pick a few audio windows from our validation set that contain bilabials and a few that contain vowels, for which the mouth should be closed and open, respectively. We then scan the emotion database for vectors that exhibit the desired behavior for all chosen windows. Performing this preliminary culling for Character 1 resulted in 100 candidate emotion vectors for further consideration. Figure 4 illustrates how the response varies with different emotion vectors. The depicted training shot contains one region of highly responsive emotion vectors, from which one candidate was chosen for further consideration.

The second step in the culling process is to play back the validation audio tracks and inspect the facial motion inferred with each

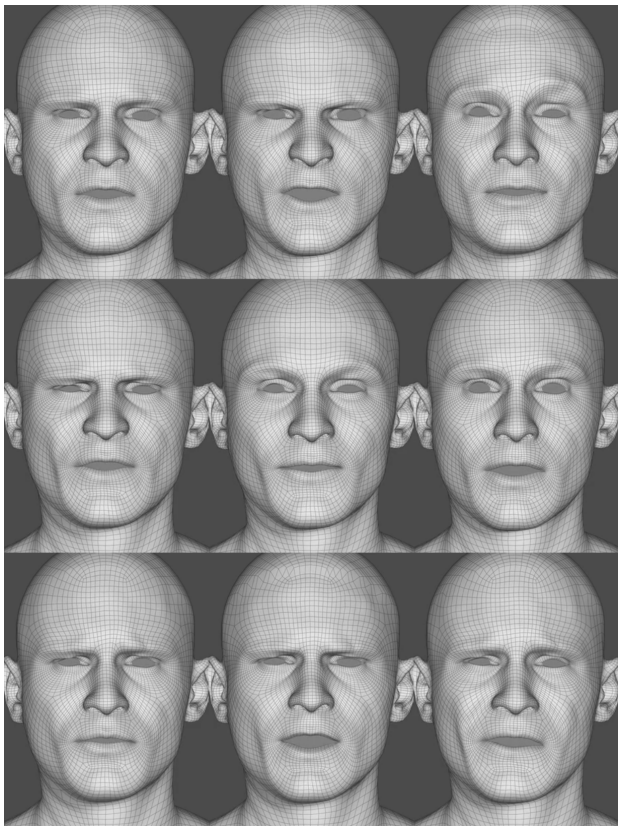

Fig. 6. The emotional state has a large effect on the animation, as shown on the accompanying video. These nine poses are inferred from the same audio window using different emotion vectors.

of the candidate emotion vectors. At this stage, we discard emotion vectors that result in subdued or spurious, unnatural motion, indicating that the emotion vector is tainted with short-term effects. This stage narrowed the set to 86 candidate emotion vectors for Character 1. As the third and final step, we run inference on several seconds of audio from a different speaker and eliminate emotion vectors with muted or unnatural response. We have found that severely attenuated response to a different speaker is a sign of lack of generalization power, and tends to cause problems even with the same speaker under varied articulation styles. With Character 1, this step left 33 emotion vectors.

We then examine the output of the network for several novel audio clips with every remaining emotion vector, and assign a semantic meaning (e.g., "neutral", "amused", "surprised", etc.) to each of them, depending on the emotional state they convey (Figure 5). Which semantic emotions remain depends entirely on the training material, and it will not be possible to extract, e.g., a "happy" emotion if the training data does not contain enough such material to be generalizable to novel audio. Figure 6 shows inferred facial poses for Character 1 using the same audio window but different emotion vectors. As can be seen, even after removing all but the best performing emotion vectors there is still substantial variation to choose from.

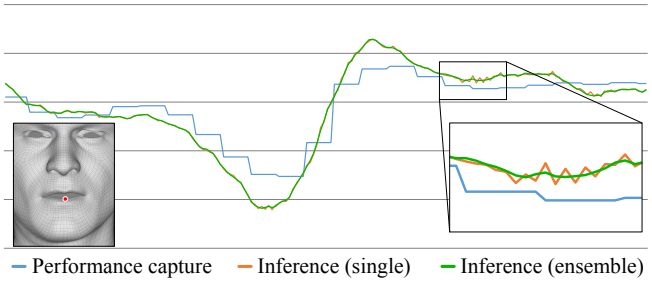

Fig. 7. The graph shows the y-coordinate of the vertex marked with the red dot as a function of time for performance capture and inference. We can see that an ensemble prediction helps to remove small-scale jitter from the inference results.

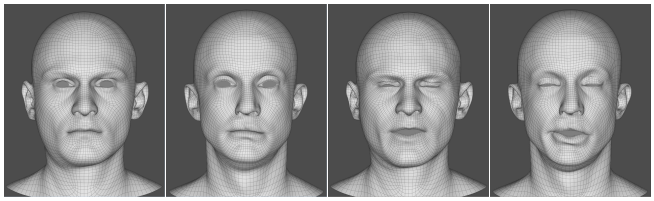

Fig. 8. Deformation transfer from Character 1 to Character 2.

Interestingly, we have found that emotion vectors that are mined in this way behave well under interpolation, i.e., sweeping from one emotion vector to another tends to produce natural-looking results. This means that it is possible to vary the emotional state during inference based on high-level information from a game engine, or by manual keyframing.

5.2 Temporal stability

As can be seen from the accompanying video, the results are stable in animation. The primary sources of temporal stability are the motion term ℓ^M and time-shift augmentation, but even with these techniques there is still a small amount of jitter left in the lip area at 4ms timescale for some inputs. This is likely due to aliasing between neighboring audio frames around stops and plosives. We fix this via *ensembling*: the network is evaluated twice for a given animation frame, 4 ms apart in time, and the predictions are averaged. Related approaches have been used before, at varying timescales, e.g. [Lewis and Parke 1987; Taylor et al. 2016]. Figure 7 illustrates the effect.

Even with the motion term, time-shifting, and ensembling some ambiguities remain in how the eyebrows should move. We thus employ additional temporal smoothing for the upper part of the face using a Gaussian filter with $\sigma_t = 0.1$ seconds.

5.3 Retargeting

When we train our model, the output network becomes specialized for a particular mesh. In many applications we would like to drive several different meshes with audio, and we support that via deformation transfer [Sumner and Popović 2004], as shown in Figure 8.

	PC vs. Ours		PC vs. DM		Ours vs. DM	
Character 1						
Clip 1	14	6	19	1	19	1
Clip 2	16	4	20	0	20	0
Clip 3	17	3	20	0	19	1
Clip 4	16	4	19	1	18	2
Clip 5	9	11	16	4	17	3
Clip 6	19	1	19	1	17	3
Clip 7	15	5	20	0	17	3
Clip 8	13	7	19	1	15	5
Character 2						
Clip 1	17	3	19	1	15	5
Clip 2	15	5	18	2	15	5
Clip 3	15	5	19	1	20	0
Clip 4	18	2	19	1	16	4
Clip 5	17	3	19	1	17	3
Total						
Votes	201	59	246	14	225	35
Ratio	**77%**	**23%**	**95%**	**5%**	**87%**	**13%**

Table 2. Results of the blind user study where we compare our method ("Ours") against video-based performance capture ("PC") and dominance model based animation ("DM") through pairwise quality comparisons. The input audio clips were taken from the held-out validation dataset of the corresponding actor. See text for details of the setup.

	Character 1				Character 2			
	Male		Female		Male		Female	
	Ours	DM	Ours	DM	Ours	DM	Ours	DM
English 1	12	8	12	8	14	6	17	3
English 2	16	4	12	8	18	2	16	4
English 3	16	4	17	3	19	1	16	4
French	16	4	18	2	15	5	16	4
German	16	4	12	8	16	4	16	4
Italian	14	6	13	7	17	3	17	3
Spanish	19	1	13	7	20	0	19	1
Total								
Votes	109	31	97	43	119	21	117	23
Ratio	**78%**	**22%**	**69%**	**31%**	**85%**	**15%**	**84%**	**16%**

Table 3. Results of the second user study where we compare our method ("Ours") against dominance model ("DM") with several different speakers and languages. Each row represents one male speaker and one female speaker, evaluated separately for Character 1 and Character 2.

6 EVALUATION

To assess the quality of our results, we conducted a blind user study with 20 participants who had no professional experience on animation techniques. In the study, we compared the output of our method ("Ours") against video-based performance capture from the DI4D system ("PC") and against dominance model based animation [Cohen and Massaro 1993; Massaro et al. 2012] produced using FaceFX software ("DM"). The audio clips used as inputs were taken from the held-out validation dataset of the corresponding actor, i.e., they were not used as part of training data when training our network.

The study featured two animated characters with a total of 13 audio clips that were 3–8 seconds long. For each clip a video was rendered with each of the three methods. Out of these, three A vs. B pairs were created (PC vs. Ours, PC vs. DM, Ours vs. DM), totaling 39 pairs of videos presented to the participants. These pairs were presented as A vs. B choices in random order, also randomizing which method was A and which was B for each individual question. The participants, unaware of which videos originated from which method, progressed through the video pairs, choosing the more natural-looking animation of each pair before moving to the next one. The study took approximately 10–15 minutes to complete. All videos were presented at 30 frames per second with audio. For our method, we assigned each audio clip a constant emotion vector that was mined from the emotion database as explained in Section 5.1.

FaceFX creates the animation by interpolating between pre-defined target poses: the neutral pose, 6 targets for the mouth, 3 for the tongue, 8 for head rotation, and 3 for the eyes (blink, squint, eyebrow raise). Ignoring the tongue, head rotation, and blinking, we supplied FaceFX with 9 targets that we manually selected from our training set. We spent about 2 hours per character to find targets that matched the examples in FaceFX documentation as closely as possible. For mouth-related target poses, the upper part of the face was smoothly masked out so that no unwanted motion occurred near the eyes, and the lower part of the face was similarly masked for the two eye-related poses. We supplied FaceFX with the transcript for each audio clip to be used as a basis for viseme analysis.

Table 2 shows the full results of the study, summed over the 20 participants. As expected, the output of video-based performance capture was generally perceived as more natural than the animations synthesized by our method or the dominance model. The fifth clip of Character 1, where our method is on par with performance capture, is an interesting exception to this rule. Because both our method and dominance model reach their highest scores against performance capture in this clip, it appears that the surprising result is caused by unnatural look of the performance capture data instead of exceptionally good performance of our method.

Our method clearly outperforms the dominance model, winning 87% of the pairwise comparisons, and even in the worst-case clips 75% of the participants preferred our method (15 vs. 5 votes). Additionally, our method fares much better in comparisons against video-based performance capture. Estimating an objective quality ranking for each of the methods would require a more elaborate user study, but we can still observe that the vote between our method and performance capture is closer to a tie than the vote between our method and dominance model.

6.1 Generalization

To assess the capability of our network to generalize over different speakers and languages, we conducted a second user study using 14 representative audio clips that we extracted from public domain audio books hosted by LibriVox (`www.librivox.org`). We did not look at the output of our network when selecting the audio clips, and we did not use them in any way when training our network, tuning the parameters, or mining the emotional states. We selected a total of 6 clips for English, and 8 for French, German, Italian, and Spanish. We further organized the clips by gender to form 7 pairs consisting of one male speaker and one female speaker each. We used a similar setup as in the first user study, and asked 20 participants to compare

videos produced by our method ("Ours") and dominance model ("DM") and choose the more natural-looking one. There were a total of 28 pairs of videos presented to each participant, one for each audio clip and each character. The videos were 8–13 seconds long and the study took approximately 15–20 minutes to complete.

The results of the study are summarized in Table 3. For male speakers, the number of participants that preferred our method is roughly the same as in the first study. The number is somewhat lower for female speakers, however. The likely explanation is that our comparison videos featured a male character even for audio from a female speaker, and it is possible that several participants perceived the output of both methods as equally unnatural in this case. The results also indicate that the variation between different languages is considerably lower than the variation between different speakers of the same language. This suggests that our method is not overly sensitive to the input language per se—the capability to generalize to novel audio is more likely related to the voice, speaking style, and tempo of the particular speaker.

6.2 Accompanying video

Since it is impossible to demonstrate the quality of our animation results using text and images, we refer the reader to the accompanying video. The video includes comparisons with video-based performance capture and dominance model, as well as a final render from a game engine. We also compare our results against dynamic visemes and JALI using the original video and audio footage from Taylor et al. [2012] and Edwards et al. [2016]. Note that in these comparisons we drive our network using a speaker different from the training set. We observe that our results are fluent, expressive, and have good temporal stability in the entire face region.

Because our method generalizes well over different speakers, it can also be used with synthetic audio. As an experiment, we ran our method with audio that was synthesized using WaveNet [van den Oord et al. 2016], a deep neural network that generates audio based on input text. As shown in the video, our method produces natural-looking animation from both male and female synthetic voices.

To further probe the limits our method, we ran it for the speakers and languages featured in Table 3 with several different emotional states. As shown in the video, the English-trained network responds surprisingly well in most cases, but it can have trouble keeping up if the input audio has very different tempo compared to the training data.

7 FUTURE WORK

Our primary focus is on generating high-quality facial animation for in-game dialogue, and we plan to continue evaluating the practical value of our method in a production setting. We have also performed several informal experiments to gauge the suitability of our method for more general-purpose use, using audio clips recorded in a casual setting with consumer-grade equipment and varying levels of background noise. In general, our method appears to remain responsive as long as the input volume level is normalized to roughly match the training data, and the animation looks plausible as long as it is displayed in sync with the audio and the tempo of the speech is not too fast. In the future, we hope to see a more principled study of these and related effects in a realistic interactive setting with two or more speakers.

We feel that the main shortcoming of our method is the lack of fine detail present in the performance capture data. Combining our approach with generative neural networks might enable better synthesis of such detail and possibly also residual motion for, e.g., the eyes. While our method is able to produce plausible results for several different emotional states based on just 5min of training data, increasing the size of the dataset would likely improve the results even further. It would be particularly interesting to train the network simultaneously for several different characters in attempt to learn a latent, unified representation of character identity. Conceivably, one could also deduce the emotional state automatically during inference based on a longer-term audio context.

ACKNOWLEDGEMENTS

We wish to thank Julian Kostov and Derek Hagen for acting, and Markus Holtmanns for German voice acting; Pauli Kemppinen for the ambient occlusion renderings, our colleagues at NVIDIA Helsinki for participating in the user studies, and the deep learning team for compute resources; JALI authors for comparison videos and helpful suggestions on evaluation, Dynamic Visemes authors for comparison videos, and WaveNet authors and LibriVox for audio clips; David Luebke for helpful comments and Lance Williams for pointers to related work.

REFERENCES

Robert Anderson, Björn Stenger, Vincent Wan, and Roberto Cipolla. 2013. Expressive visual text-to-speech using active appearance models. In *Proc. CVPR*. 3382–3389.

Mohamed Benzeghiba, Renato De Mori, Olivier Deroo, Stephane Dupont, Teodora Erbes, Denis Jouvet, Luciano Fissore, Pietro Laface, Alfred Mertins, Christophe Ris, and others. 2007. Automatic speech recognition and speech variability: A review. In *Speech Communication*, Vol. 49. 763–786.

Matthew Brand. 1999. Voice Puppetry. In *Proc. ACM SIGGRAPH*. 21–28.

Yong Cao, Petros Faloutsos, and Frédéric Pighin. 2003. Unsupervised Learning for Speech Motion Editing. In *Proc. SCA*. 225–231.

Yong Cao, Wen C. Tien, Petros Faloutsos, and Frédéric Pighin. 2005. Expressive Speech-driven Facial Animation. *ACM Trans. Graph.* 24, 4 (2005), 1283–1302.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cuDNN: Efficient Primitives for Deep Learning. *arXiv:1410.0759* (2014).

E. S. Chuang, F. Deshpande, and C. Bregler. 2002. Facial expression space learning. In *Proc. Pacific Graphics*. 68–76.

Michael M. Cohen and Dominic W. Massaro. 1993. Modeling Coarticulation in Synthetic Visual Speech. In *Models and Techniques in Computer Animation*. 139–156.

Salil Deena and Aphrodite Galata. 2009. Speech-Driven Facial Animation Using a Shared Gaussian Process Latent Variable Model. In *Proc. Symposium on Advances in Visual Computing: Part I*. 89–100.

S. Deena, S. Hou, and A. Galata. 2013. Visual Speech Synthesis Using a Variable-Order Switching Shared Gaussian Process Dynamical Model. *IEEE Transactions on Multimedia* 15, 8 (2013), 1755–1768.

Zhigang Deng, Shri Narayanan, Carlos Busso, and Ulrich Neumann. 2004. Audio-based Head Motion Synthesis for Avatar-based Telepresence Systems. In *Proc. Workshop on Effective Telepresence*. 24–30.

Zhigang Deng, Ulrich Neumann, J. P. Lewis, Tae-Yong Kim, Murtaza Bulut, and Shrikanth Narayanan. 2006. Expressive Facial Animation Synthesis by Learning Speech Coarticulation and Expression Spaces. *IEEE TVCG* 12, 6 (2006), 1523–1534.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, and others. 2015. Lasagne: First release. (2015).

Pif Edwards, Chris Landreth, Eugene Fiume, and Karan Singh. 2016. JALI: An Animator-centric Viseme Model for Expressive Lip Synchronization. *ACM Trans. Graph.* 35, 4 (2016), 127:1–127:11.

A. Elgammal and Chan-Su Lee. 2004. Separating style and content on a nonlinear manifold. In *Proc. CVPR*, Vol. 1. 478–485.

Tony Ezzat, Gadi Geiger, and Tomaso Poggio. 2002. Trainable Videorealistic Speech Animation. *ACM Trans. Graph.* 21, 3 (2002), 388–398.

Bo Fan, Lei Xie, Shan Yang, Lijuan Wang, and Frank K. Soong. 2016. A deep bidirectional LSTM approach for video-realistic talking head. *Multimedia Tools and Applications* 75, 9 (2016), 5287–5309.

Cletus G. Fisher. 1968. Confusions Among Visually Perceived Consonants. *JSLHR* 11 (1968), 796–804.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852* (2015).

Gregor Hofer and Korin Richmond. 2010. Comparison of HMM and TMDN Methods for Lip Synchronisation. In *Proc. Interspeech*. 454–457.

Pengyu Hong, Zhen Wen, and T. S. Huang. 2002. Real-time Speech-driven Face Animation with Expressions Using Neural Networks. *Trans. Neur. Netw.* 13, 4 (2002), 916–927.

Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167* (2015).

Jia Jia, Zhiyong Wu, Shen Zhang, Helen M. Meng, and Lianhong Cai. 2014. Head and facial gestures synthesis using PAD model for an expressive talking avatar. *Multimedia Tools and Applications* 73, 1 (2014), 439–461.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980* (2014).

S. Kshirsagar and N. Magnenat-Thalmann. 2000. Lip synchronization using linear predictive analysis. In *Proc. ICME*, Vol. 2. 1077–1080.

John Lewis. 1991. Automated lip-sync: Background and techniques. *The Journal of Visualization and Computer Animation* 2, 4 (1991), 118–122.

J. P. Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Fred Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. In *Eurographics (State of the Art Reports)*.

J. P. Lewis and F. I. Parke. 1987. Automated Lip-synch and Speech Synthesis for Character Animation. In *Proc. SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*. 143–147.

K. Liu and J. Ostermann. 2011. Realistic facial expression synthesis for an image-based talking head. In *Proc. ICME*. 1–6.

M. Malcangi. 2010. Text-driven avatars based on artificial neural networks and fuzzy logic. *Int. J. Comput.* 4, 2 (2010), 61–69.

Stacy Marsella, Yuyu Xu, Margaux Lhommet, Andrew Feng, Stefan Scherer, and Ari Shapiro. 2013. Virtual Character Performance from Speech. In *Proc. SCA*. 25–35.

D. W. Massaro, J. Beskow, M. M. Cohen, C. L. Fry, and T. Rodriguez. 1999. Picture my voice: Audio to visual speech synthesis using artificial neural networks. In *Proc. AVSP*. #23.

D. W. Massaro, M. M. Cohen, R. Clark, and M. Tabain. 2012. Animated speech: Research progress and applications. In *Audiovisual Speech Processing*. 309–345.

Wesley Mattheyses and Werner Verhelst. 2015. Audiovisual speech synthesis: An overview of the state-of-the-art. *Speech Communication* 66 (2 2015), 182–217.

J. Melenchon, E. Martinez, F. De La Torre, and J. A. Montero. 2009. Emphatic Visual Speech Synthesis. *IEEE Transactions on Audio, Speech, and Language Processing* 17, 3 (2009), 459–468.

M. Mori. 1970. Bukimi no tani (The uncanny valley). *Energy* 7, 4 (1970), 33–35.

T. Öhman and G. Salvi. 1999. Using HMMs and ANNs for mapping acoustic to visual speech. *IEEE Journal of Selected Topics in Signal Processing* 40, 1 (1999), 45–50.

Valery A. Petrushin. 1998. How well can People and Computers Recognize Emotions in Speech?. In *Proc. AAAI Fall Symp.* 141–145.

D. Schabus, M. Pucher, and G. Hofer. 2014. Joint Audiovisual Hidden Semi-Markov Model-Based Speech Synthesis. *IEEE Journal of Selected Topics in Signal Processing* 8, 2 (2014), 336–347.

JL Schwartz and C Savariaux. 2014. No, there is no 150 ms lead of visual speech on auditory speech, but a range of audiovisual asynchronies varying from small audio lead to large audio lag. *PLoS Computational Biology* 10, 7 (2014).

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958.

Robert W. Sumner and Jovan Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Trans. Graph.* 23, 3 (2004), 399–405.

Sarah Taylor, Akihiro Kato, Ben Milner, and Iain Matthews. 2016. Audio-to-Visual Speech Conversion using Deep Neural Networks. In *Proc. Interspeech*. 1482–1486.

Sarah L. Taylor, Moshe Mahler, Barry-John Theobald, and Iain Matthews. 2012. Dynamic Units of Visual Speech. In *Proc. SCA*. 275–284.

Joshua B. Tenenbaum and William T. Freeman. 2000. Separating Style and Content with Bilinear Models. *Neural Comput.* 12, 6 (2000), 1247–1283.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv:1605.02688* (2016).

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499* (2016).

M. Alex O. Vasilescu and Demetri Terzopoulos. 2003. Multilinear Subspace Analysis of Image Ensembles. In *Proc. CVPR*, Vol. 2. 93–99.

Kevin Wampler, Daichi Sasaki, Li Zhang, and Zoran Popović. 2007. Dynamic, Expressive Speech Animation from a Single Mesh. In *Proc. SCA*. 53–62.

Lijuan Wang and Frank K. Soong. 2015. HMM trajectory-guided sample selection for photo-realistic talking head. *Multimedia Tools and Applications* 74, 22 (2015), 9849–9869.