Production-Level Facial Performance Capture Using Deep Convolutional Neural Networks

Samuli Laine
NVIDIA

Tero Karras
NVIDIA

Timo Aila
NVIDIA

Antti Herva
Remedy Entertainment

Shunsuke Saito
Pinscreen
University of Southern California

Ronald Yu
Pinscreen
University of Southern California

Hao Li
USC Institute for Creative Technologies
University of Southern California
Pinscreen

Jaakko Lehtinen
NVIDIA
Aalto University

ABSTRACT

We present a real-time deep learning framework for video-based facial performance capture—the dense 3D tracking of an actor's face given a monocular video. Our pipeline begins with accurately capturing a subject using a high-end production facial capture pipeline based on multi-view stereo tracking and artist-enhanced animations. With 5–10 minutes of captured footage, we train a convolutional neural network to produce high-quality output, including self-occluded regions, from a monocular video sequence of that subject. Since this 3D facial performance capture is fully automated, our system can drastically reduce the amount of labor involved in the development of modern narrative-driven video games or films involving realistic digital doubles of actors and potentially hours of animated dialogue per character. We compare our results with several state-of-the-art monocular real-time facial capture techniques and demonstrate compelling animation inference in challenging areas such as eyes and lips.

CCS CONCEPTS

• **Computing methodologies** → **Animation**; **Neural networks**; *Supervised learning by regression*;

KEYWORDS

Facial animation, deep learning

ACM Reference format:
Samuli Laine, Tero Karras, Timo Aila, Antti Herva, Shunsuke Saito, Ronald Yu, Hao Li, and Jaakko Lehtinen. 2017. Production-Level Facial Performance Capture Using Deep Convolutional Neural Networks. In *Proceedings of SCA '17, Los Angeles, CA, USA, July 28-30, 2017,* 10 pages.
https://doi.org/10.1145/3099564.3099581

SCA '17, July 28-30, 2017, Los Angeles, CA, USA

ACM ISBN 978-1-4503-5091-4/17/07…$15.00
https://doi.org/10.1145/3099564.3099581

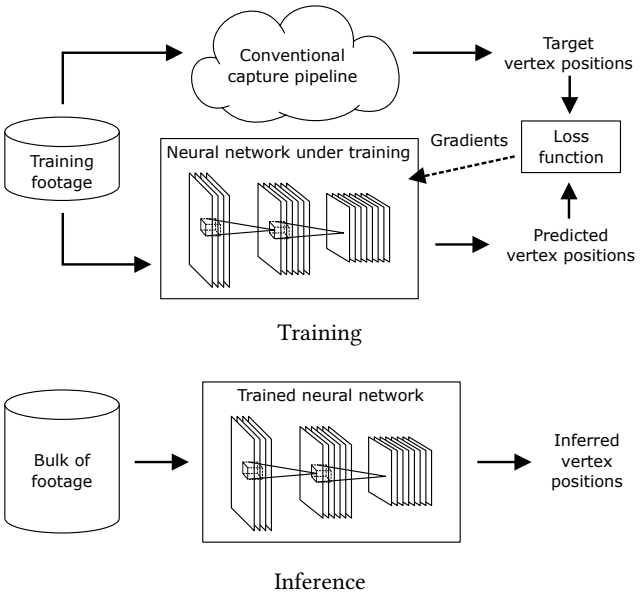


Figure 1: Our deep learning-based facial performance capture framework is divided into a training and inference stage. The goal of our system is to reduce the amount of footage that needs to be processed using labor-intensive production-level pipelines.

1 INTRODUCTION

The use of visually compelling digital doubles of human actors is a key component for increasing realism in any modern narrative-driven video game. Facial performance capture poses many challenges in computer animation and due to a human's innate sensitivity to the slightest facial cues, it is difficult to surpass the uncanny valley, where otherwise believable renderings of a character appear lifeless or unnatural.

Despite dramatic advancements in automated facial performance capture systems and their wide deployment for scalable production, it is still not possible to obtain a perfect tracking for highly complex expressions, especially in challenging but critical areas such as lips

and eye regions. In most cases, manual clean-up and corrections by skilled artists are necessary to ensure high-quality output that is free from artifacts and noise. Conventional facial animation pipelines can easily result in drastic costs, especially in settings such as video game production where hours of footage may need to be processed.

In this paper, we introduce a deep learning framework for real-time and production-quality facial performance capture. Our goal is not to fully eliminate the need for manual work, but to significantly reduce the extent to which it is required. We apply an offline, multi-view stereo capture pipeline with manual clean-up to a small subset of the input video footage, and use it to generate enough data to train a deep neural network. The trained network can then be used to automatically process the remaining video footage at rates as fast as 870 fps, skipping the conventional labor-intensive capture pipeline entirely.

Furthermore, we only require a single view as input during runtime which makes our solution attractive for head cam-based facial capture. Our approach is real-time and does not even need sequential processing, so every frame can be processed independently. Furthermore, we demonstrate qualitatively superior results compared to state-of-the-art monocular real-time facial capture solutions. Our pipeline is outlined in Figure 1.

1.1 Problem Statement

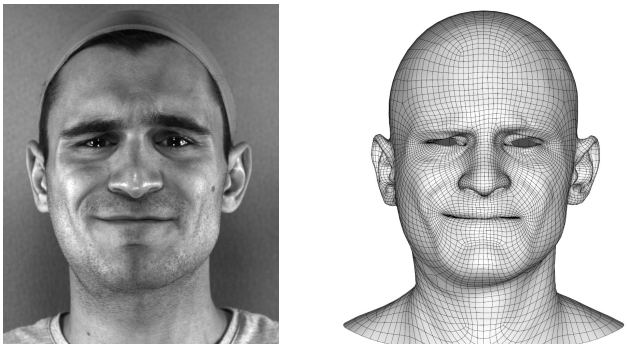
We assume that the input for the capture pipeline is multiple-view videos of the actor's head captured under controlled conditions to generate training data for the neural network. The input to the neural network at runtime is video from a single view. The positions of the cameras remain fixed, the lighting and background are standardized, and the actor is to remain at approximately the same position relative to the cameras throughout the recording. Naturally, some amount of movement needs to be allowed, and we achieve this through data augmentation during training (Section 4.1).

The output of the capture pipeline is the set of per-frame positions of facial mesh vertices, as illustrated in Figure 2. Other face encodings such as blendshape weights or joint positions are introduced in later stages of our pipeline, mainly for compression and rendering purposes, but the primary capture output consists of the positions of approximately 5000 animated vertices on a fixed-topology facial mesh.

1.2 Offline Capture Pipeline

The training data used for the deep neural network was generated using Remedy Entertainment's in-house capture pipeline based on a cutting edge commercial DI4D PRO system [Dimensional Imaging 2016] that employs nine video cameras.

First, an unstructured mesh with texture and optical flow data is created from the images for each frame of an input video. A fixed-topology template mesh is created prior to the capture work by applying Agisoft [Photoscan 2014], a standard multi-view stereo reconstruction software, on data from 26 DSLR cameras and two cross polarized flashes. The mesh is then warped onto the unstructured scan of the first frame. The template mesh is tracked using optical flow through the entire sequence. Possible artifacts are manually fixed using the DI4DTrack software by a clean-up artist. The position and orientation of the head are then stabilized using a few key vertices of the tracking mesh. The system then outputs the positions of each of the vertices on the fixed-topology template mesh.



Input video frame Output

Figure 2: Input for the conventional capture pipeline is a set of nine images, whereas our network only uses a cropped portion of the center camera image converted to grayscale. Output of both the conventional capture pipeline and our network consists of ~5000 densely tracked 3D vertex positions for each frame.

Additional automated deformations are later applied to the vertices to fix remaining issues. For instance, the eyelids are deformed to meet the eyeballs exactly and to slide slightly with motion of the eyes. Also, opposite vertices of the lips are smoothly brought together to improve lip contacts when needed. After animating the eye directions the results are compressed for runtime use in Remedy's Northlight engine using 416 facial joints. Pose space deformation is used to augment the facial animation with detailed wrinkle normal map blending. These ad-hoc deformations were not applied in the training set.

2 RELATED WORK

The automatic capture of facial performances has been an active area of research for decades [Blanz and Vetter 1999; Mattheyses and Verhelst 2015; Pughin and Lewis 2006; Williams 1990a], and is widely used in game and movie production today. In this work we are primarily interested in real-time methods that are able to track the entire face, without relying on markers, and are based on consumer hardware, ideally a single RGB video camera.

2.1 Production Facial Capture Systems

Classic high-quality facial capture methods for production settings require markers [Bickel et al. 2008; Guenter et al. 1998; Williams 1990b] or other application specific hardware [Pighin and Lewis 2006]. Purely data-driven high-quality facial capture methods used in a production setting still require complicated hardware and camera setups [Alexander et al. 2009; Beeler et al. 2011; Bhat et al. 2013; Borshukov et al. 2005; Fyffe et al. 2014; Vlasic et al. 2005; Weise et al. 2009a; Zhang et al. 2004] and a considerable amount of computation such as multi-view stereo or photometric reconstruction of individual input frames [Beeler et al. 2011; Bradley et al. 2010; Furukawa and Ponce 2009; Fyffe et al. 2011; Shi et al. 2014; Valgaerts

et al. 2012] that often require carefully crafted 3D tracking model [Alexander et al. 2009; Borshukov et al. 2005; Vlasic et al. 2005]. Many production setting facial performance capture techniques require extensive manual post-processing as well.

Specialized techniques have also been proposed for various subdomains, including eyelids [Bermano et al. 2015], gaze direction [Zhang et al. 2015], lips [Garrido et al. 2016], handling of occlusions [Saito et al. 2016], and handling of extreme skin deformations [Wu et al. 2016].

Our method is a "meta-algorithm" in the sense that it relies on an existing technique for generating the training examples, and then learns to mimic the host algorithm, producing further results at a fraction of the cost. As opposed to the complex hardware setup, heavy computational time, and extensive manual post-processing involved in these production setting techniques, our method is able to produce results with a single camera, reduced amounts of manual labor, and at a rate of slightly more than 1 ms per frame when images are processed in parallel. While we currently base our system on a specific commercial solution, the same general idea can be built on top of any facial capture technique taking video inputs, ideally the highest-quality solution available.

2.2 Single-View Real-time Facial Animation

Real-time tracking from monocular RGB videos is typically based either on detecting landmarks and using them to drive the facial expressions [Cootes et al. 2001; Saragih et al. 2011; Tresadern et al. 2012] or on 3D head shape regression [Cao et al. 2015, 2014, 2013; Hsieh et al. 2015a; Li et al. 2010; Olszewski et al. 2016; Thies et al. 2016; Weise et al. 2009b]. Of these methods, the regression approach has delivered higher-fidelity results, and real time performance has been demonstrated even on mobile devices [Weng et al. 2014]. The early work on this area [Cao et al. 2013; Weng et al. 2014] require an actor-specific training step, but later developments have relaxed that requirement [Cao et al. 2014] and also extended the method to smaller-scale features such as wrinkles [Cao et al. 2015; Ichim et al. 2015].

Most of these methods are targeting "in-the-wild" usage and thus have to deal with varying lighting, occlusions, and unconstrained head poses. Thus, these methods are typically lower quality in detail and accuracy. These methods are also usually only able to infer low-dimensional facial expressions—typically only a few blendshapes—reliably. More problems also arise in appearance based methods such as [Thies et al. 2016]. For example, relying on pixel constraints makes it possible only to track visible regions, making it difficult to reproduce regions with complex interactions such as the eyes and lips accurately. Additionally, relying on appearance can lead to suboptimal results if the PCA model does not accurately encode the subject's appearance such as in the case of facial hair.

In contrast, we constrain the setup considerably in favor of high-fidelity results for one particular actor. In our setup, all of the lighting and shading as well as gaze direction and head poses are produced at runtime using higher-level procedural controls. Using such a setup, unlike the other less constrained real-time regression-based methods, our method is able obtain high quality results as well as plausible inferences for the non-visible regions and other difficult to track regions such as the lips and eyes.

Olszewski et al. [2016] use neural networks to regress eye and mouth videos separately into blend shape weights in a head-mounted display setup. Their approach is closely related to ours with some slight differences. First of all, their method considers the eye and mouth separately while our method considers the whole face at once. Also, they use blendshapes from FACS [Ekman and Friesen 1978] while our system produces vertex coordinates of the face mesh based on a 160-dimensional PCA basis. Moreover, our system can only process one user at a time without retraining while the method of Olszewski et al. [2016] is capable of processing several different identities. However, our method can ensure accurate face tracking while theirs is only designed to track the face to drive a target character.

2.3 Alternative Input Modalities

Alternatively, a host of techniques exists for audio-driven facial animation [Brand 1999; Cohen and Massaro 1993; Edwards et al. 2016; Taylor et al. 2012], and while impressive results have been demonstrated, these techniques are obviously not applicable to non-vocal acting and also commonly require an animator to adjust the correct emotional state. They continue to have important uses as a lower-cost alternative, e.g., in in-game dialogue.

A lot of work has also been done for RGB-D sensors, such as Microsoft Kinect Fusion, e.g., [Bouaziz et al. 2013; Hsieh et al. 2015b; Li et al. 2013; Thies et al. 2015; Weise et al. 2011]. Recently Liu et al. also described a method that relies on RGB-D and audio inputs [Liu et al. 2015].

2.4 Convolutional Neural Networks (CNN)

We base our work on deep CNNs that have received significant attention in the recent years, and proven particularly well suited for large-scale image recognition tasks [Krizhevsky et al. 2012; Simonyan and Zisserman 2014]. Modern CNNs employ various techniques to reduce the training time and improve generalization over novel input data, including data augmentation [Simard et al. 2003], dropout regularization [Srivastava et al. 2014], ReLU activation functions, i.e., $max(0, \cdot)$, and GPU acceleration [Krizhevsky et al. 2012]. Furthermore, it has been shown that state-of-the-art performance can be achieved with very simple network architectures that consist of small 3×3-pixel convolutional layers [Simonyan and Zisserman 2014] that employ strided output to reduce spatial resolution throughout the network [Springenberg et al. 2014].

3 NETWORK ARCHITECTURE

Our input footage is divided into a number of shots, with each shot typically consisting of 100–2000 frames at 30 FPS. Data for each input frame consists of a 1200×1600 pixel image from each of the nine cameras. As explained above, the output is the per-frame vertex position for each of the ∼5000 facial mesh vertices.

As input for the network, we take the 1200×1600 video frame from the central camera, crop it with a fixed rectangle so that the face remains in the picture, and scale the remaining portion to 240×320 resolution. Furthermore, we convert the image to grayscale, resulting in a total of 76800 scalars to be fed to the network. The resolution may seem low, but numerous tests confirmed that increasing it did not improve the results.

3.1 Convolutional Network

Our convolutional network is based on the all-convolutional architecture [Springenberg et al. 2014] extended with two fully connected layers to produce the full-resolution vertex data at output. The input is a whitened version of the 240×320 grayscale image. For whitening, we calculate the mean and variance over all pixels in the training images, and bias and scale the input so that these are normalized to zero and one, respectively.

Note that the same whitening coefficients, fixed at training time, are used for all input images during training, validation, and production use. If the whitening were done on a per-image or per-shot basis, we would lose part of the benefits of the standardized lighting environment. For example, variation in the color of the actor's shirt between shots would end up affecting the brightness of the face. The layers of the network are listed in the table below.

Name	Description
input	Input $1 \times 240 \times 320$ image
conv1a	Conv 3×3, $1 \rightarrow 64$, stride 2×2, ReLU
conv1b	Conv 3×3, $64 \rightarrow 64$, stride 1×1, ReLU
conv2a	Conv 3×3, $64 \rightarrow 96$, stride 2×2, ReLU
conv2b	Conv 3×3, $96 \rightarrow 96$, stride 1×1, ReLU
conv3a	Conv 3×3, $96 \rightarrow 144$, stride 2×2, ReLU
conv3b	Conv 3×3, $144 \rightarrow 144$, stride 1×1, ReLU
conv4a	Conv 3×3, $144 \rightarrow 216$, stride 2×2, ReLU
conv4b	Conv 3×3, $216 \rightarrow 216$, stride 1×1, ReLU
conv5a	Conv 3×3, $216 \rightarrow 324$, stride 2×2, ReLU
conv5b	Conv 3×3, $324 \rightarrow 324$, stride 1×1, ReLU
conv6a	Conv 3×3, $324 \rightarrow 486$, stride 2×2, ReLU
conv6b	Conv 3×3, $486 \rightarrow 486$, stride 1×1, ReLU
drop	Dropout, $p = 0.2$
fc	Fully connected $9720 \rightarrow 160$, linear activation
output	Fully connected $160 \rightarrow N_{\text{out}}$, linear activation

The output layer is initialized by precomputing a PCA basis for the output meshes based on the target meshes from the training data. Allowing 160 basis vectors explains approximately 99.9% of the variance seen in the meshes, which was considered to be sufficient. If we fixed the weights of the output layer and did not train them, that would effectively train the remainder of the network to output the 160 PCA coefficients. However, we found that allowing the last layer to be trainable as well improved the results. This would seem to suggest that the optimization is able to find a slightly better intermediate basis than the initial PCA basis.

4 TRAINING

For each actor, the training set consists of four parts, totaling approximately 5–10 minutes of footage. The composition of the training set is as follows.

Extreme Expressions. In order to capture the maximal extents of the facial motion, a single range-of-motion shot is taken where the actor goes through a pre-defined set of extreme expressions. These include but are not limited to opening the mouth as wide as possible, moving the jaw sideways and front as far as possible, pursing the lips, and opening the eyes wide and forcing them shut.

FACS-Like Expressions. Unlike the range-of-motion shot that contains exaggerated expressions, this set contains regular FACS-like expressions such as squinting of the eyes or an expression of disgust. These kind of expressions must be included in the training set as otherwise the network would not be able to replicate them in production use.

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, which are sentences that are designed to contain as many different phonemes as possible, in several different emotional tones. A pangram fitting the emotion would be optimal but in practice this is not always feasible.

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. This material is composed of the preliminary version of the script, or it may be otherwise prepared for the training. 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 isn't perfect or if completely novel or out of character acting is encountered.

The training set is typically comprised of roughly 10% of range-of-motion and expression shots, 30% of pangrams across emotional states, and 60% of in-character performances of varying intensity and scenario.

4.1 Data Augmentation

We perform several transformations to the input images during training in order to make the network resistant to variations in input data. These transformations are executed on CPU concurrently with network evaluation and training that occurs on the GPU. Augmentation is not used when evaluating the validation loss or when processing unseen input data in production use. Examples of augmented input images are shown in Figure 3.

The main transformations are translation, rotation and zoom, which account for the motion of the actor's head during capture. The magnitudes of these augmentations are set so that they cover at least all of the variation expected in the input data. This kind of image-based augmentation does not cover large-scale changes in head pose, and thus our method does not tolerate that unless such effects are present in the training data.

In addition to geometric transformations, we vary the brightness and contrast of the input images during training, in order to account for variations in lighting over the capture process. Our cameras pick a slight periodic flicker from the 50 Hz LED lights in the capture room, and it is possible that some of the bulbs degrade during the capture period that may take place over several days or weeks.

4.2 Training Parameters

We train the network for 200 epochs using the Adam [Kingma and Ba 2014] optimization algorithm with parameters set to values recommended in the paper. The learning rate is ramped up using a geometric progression during the first training epoch, and then decreased according to $1/\sqrt{t}$ schedule. During the last 30 epochs we ramp the learning rate down to zero using a smooth curve,

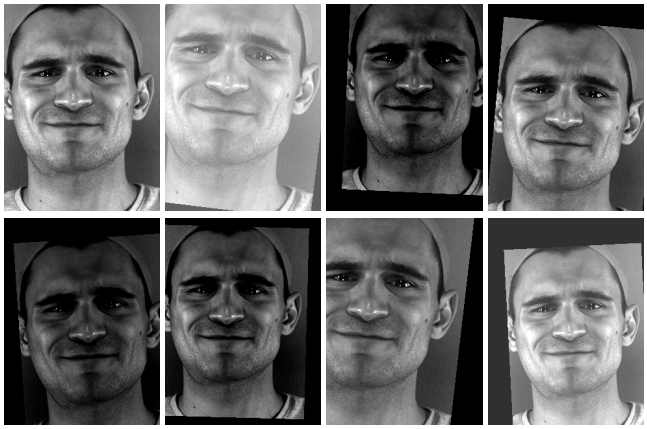


Figure 3: Examples of augmented inputs presented to the network during training. Top left image is the 240×320 crop from the same input video frame as was shown in Figure 2, and the remaining images are augmented variants of it.

and simultaneously ramp Adam 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. Minibatch size is set to 50, and each epoch processes all training frames in randomized order. Weights are initialized using the initialization scheme of He et al. [2015], except for the last output layer which is initialized using a PCA transformation matrix as explained in Section 3.1.

The strength of all augmentation transformations is ramped up linearly during the first five training epochs, starting from zero. This prevented a rare but annoying effect where the network fails to start learning properly, and gets stuck at clearly sub-optimal local minimum. The augmentation ramp-up process can be seen as a form of curriculum learning [Bengio et al. 2009].

Our loss function is simply the mean square error between the predicted vertex positions produced by the network and the target vertex positions in the training data.

Our implementation is written in Python using Theano [Theano Development Team 2016] and Lasagne [Dieleman et al. 2015]. On a computer with a modern CPU and a NVIDIA Titan X GPU, the training of one network with a typical training set containing 10000–18000 training frames (~5–10 minutes at 30Hz) takes approximately 5–10 hours.

5 RESULTS

We tested the trained network using footage from a later session. The lighting conditions and facial features exhibited in the training set were carefully preserved. The inference was evaluated numerically and perceptually in relation to a manually tracked ground truth.

We will first evaluate our choices in the design and training of our neural network, followed by examination of the numerical results. We then turn to visual comparisons with recent monocular real-time facial performance capture methods. Finally, we explore the limitations of our pipeline.

The quality of the results can be best assessed from the accompanying video. In the video an interesting observation is that our results are not only accurate but also perfectly stable temporally despite the fact that we do not employ recurrent networks or smooth the generated vertex positions temporally in any way. It is very difficult for human operators to achieve similar temporal stability as they inevitably vary in their work between sequences.

5.1 Network Architecture Evaluation

All results in this paper were computed using the architecture described in Section 3.1. It should be noted that the quality of the results is not overly sensitive to the exact composition of the network. Changing the dimensions of the convolutional layers or removing or adding the 1×1 stride convolution layers only changed performance by a slight margin. The architecture described in Section 3.1 was found to perform slightly better compared to other all-convolutional architectures that could be trained in a reasonable amount of time, so it was chosen for use in production.

In addition to using an all-convolutional neural network, we also experimented with fully connected networks. When experimenting with fully connected networks, we achieved the best results by transforming the input images into 3000 PCA coefficients. The PCA basis is pre-calculated based on the input frames from the training set, and the chosen number of basis images captures approximately 99.9% of the variance in the data. The layers of the network are listed in the table below.

Name	Description
input	Input 3000 image PCA coefficients
fc1	Fully connected 3000 $\rightarrow$ 2000, ReLU activation
drop1	Dropout, $p = 0.2$
fc2	Fully connected 2000 $\rightarrow$ 1000, tanh activation
drop2	Dropout, $p = 0.2$
fc3	Fully connected 1000 $\rightarrow$ 160, linear activation
output	Fully connected 160 $\rightarrow$ N_{out}, linear activation

The position and the orientation of the head in the input images varies, which in practice necessitates stabilizing the input images prior to taking the PCA transformation. For this we used the facial landmark detector of Kazemi and Sullivan [2014]. Rotation angle and median line of the face were estimated from the landmark points surrounding the eyes. Because these were found to shift vertically during blinking of the eyes, the vertical position of the face was determined from the landmark points on the nose. The image was then rotated to a fixed orientation, and translated so that the point midway between the eyes remained at a fixed position.

Even though the network may seem overly simplistic, similarly to the all-convolutional architecture, we did not find a way to improve the results by adding more layers or changing the widths of the existing layers. We experimented with different regularization schemes, but simply adding two dropout layers was found to yield the best results. The output layer is initialized using a PCA basis for the output meshes computed as in the convolutional network.

Ultimately, the only aspect in which the fully connected network remained superior to the convolutional network was training time. Whereas the convolutional network takes 8–10 hours to train in a typical case, the fully connected network would converge in as

Figure 4: Convergence of the convolutional network for Character 1 during 200 training epochs. The effects of learning rate and β_1 rampdown can be seen in the final 30 epochs. This training run had 15173 training frames and 1806 validation frames and took 8 hours to finish.

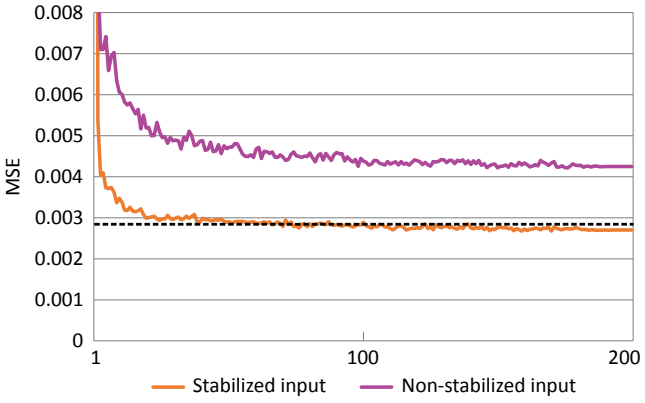


Figure 5: Convergence of the validation loss of the fully-connected network. The training took approximately one hour, and the training set was the same as for the convolutional network. The dashed horizontal line shows the converged validation loss of the convolutional network, and we can see that with stabilized input images, the fully connected network reaches slightly smaller loss than the all-convolutional network. However, visually the results were clearly inferior. Using non-stabilized input images yielded much worse results both numerically and visually.

little as one hour. Even though it was initially speculated that fast training times could be beneficial in production use, it ended up not mattering much as long as training could be completed overnight.

One disadvantage of using a fully connected network is that stabilizing the input images for the fully connected network turned out to be problematic because of residual motion that remained due to inaccuracies in the facial landmark detection. This residual jitter of input images sometimes caused spurious and highly distracting motion of output vertices. We tried hardening the fully connected network to this effect by applying a similar jitter to inputs during training in order to present the same stabilized input image to the network in slightly different positions, but this did not help.

We suspect that it may be too difficult for the fully connected network to understand that slightly offset images should produce the same result, perhaps partially due to the input PCA transform. Nonetheless, the results with input PCA transform were better than using the raw image as the input.

On the other hand, the convolutional network, when trained with proper input augmentation (Section 4.1), is not sensitive to the position and orientation of the actor's head in the input images. Hence the convolutional network carries an advantage in that no image stabilization is required as a pre-process.

We see in Figures 4 and 5 that the fully connected network often produced numerically better results than the convolutional network, but visually the results were significantly worse as the fully connected network appeared to generally attenuate the facial motion. Even in individual shots where the fully connected network produced a numerically clearly superior result, the facial motion was judged to lack expressiveness and was not as temporally stable compared to the results produced by the convolutional network. We further discuss this general discrepancy between numerical and visual quality below.

5.2 Training Process Evaluation

In Section 4.1 we described several data augmentations we performed that made the network more resistant to variations in input data and eliminated the need for stabilization as a pre-process for our all-convolutional network. Additionally, we also tried augmenting the data by adding noise to the images and applying a variable gamma correction factor to approximate varying skin glossiness. However, both of these augmentations were found to be detrimental to learning. Similarly, small 2D perspective transformations—an attempt to crudely mimic the non-linear effects of head rotations—were not found to be beneficial.

5.3 Numerical Results

Figure 4 shows the convergence of the network for Character 1, trained using 15173 input frames. The training set for Character 2 contained 10078 frames. As previously explained, our loss function is the MSE between the network output and target positions from the training/validation set. The vertex coordinates are measured in centimeters in our data, so the final validation loss of 0.0028 corresponds to RMSE of 0.92 millimeters. With longer training the training loss could be pushed arbitrarily close to zero, but this did not improve the validation loss or the subjective quality of the results.

Figure 7 illustrates the numerical accuracy of our trained network on a selection of interesting frames in validation shots that were not used in training. Note that the RMSE of the frames shown in the figure are higher than average since the validation data mostly consist of more neutral material than the frames shown in the figure. Per-frame RMSE plot for the validation shots for Character 1 is shown in Figure 6.

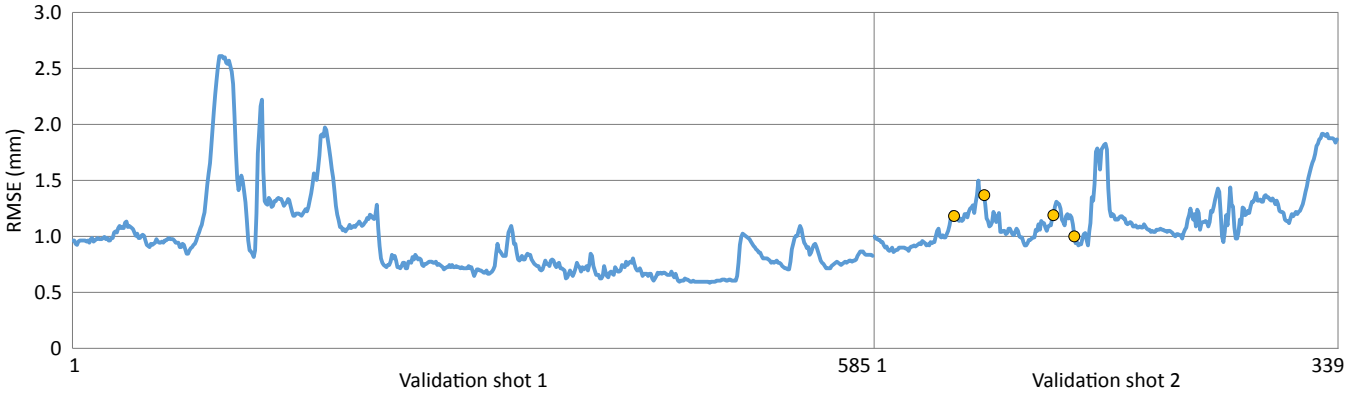


Figure 6: Per-frame RMSE in the first two validation shots of Character 1. Frame index advances from left to right. The orange dots indicate frames used in the top four rows of Figure 7. These validation shots are included in the accompanying video.

We found that the neural network was very efficient in producing consistent output even when there were variations in the input data because of inaccuracies in the conventional capture pipeline. In the first four rows of Figure 7, we can see that, especially in the regions around the hairline and above the eyebrows, the target data obtained from the conventional capture pipeline sometimes contained systematic errors that the capture artists did not notice and thus did not correct. Because a neural network only learns the consistent features of the input-output mapping as long as overfitting is avoided, our network output does not fluctuate in the same way as the manual target positions do. In fact, visually it is often not clear whether the manually tracked target positions or the inferred positions are closer to the ground truth. We believe this explains some of the numerical discrepancies between our output and the validation data.

Given the inevitable variability in the manual work involved in using the conventional capture pipeline, we could not hope that our network would reach a numerically exact match with manually prepared validation data. The goal of performance capture is to generate believable facial motion, and therefore the perceptual quality of the results—as judged by professional artists—is ultimately what determines whether a capture system is useful or not in our production environment.

5.4 Comparison

We visually compare our method in Figure 8 to Thies et al. [2016] and Cao et al. [2014], two state-of-the-art monocular real-time facial performance capture methods that do not require a rig. Since the comparison methods generalize to any identity and our method assumes the identity of one user, in order to make the comparison more fair, we use the method of Li et al. [2010] to fix the identity mesh for the comparison methods and only track the expressions. Visually our method appears to be more accurate than [Thies et al. 2016] and [Cao et al. 2014], but we note that they bear significant advantages in that they do not require per-user calibration and allow for less constrained head movements. We also note that if we allowed the comparison methods to retrain for new identities and restricted head movement in all their inputs, their accuracy could be improved to more closely match our levels. An advantage our method poses over the comparison methods is that it is capable of inferring plausible animations for self-occluded or difficult to track regions such as details surrounding the mouth and eyes. In a production setting where we have resources to constrain the head movement and perform per-user training and would like to capture the user as accurately and plausibly as possible across all regions of the head, our method is advantageous over other existing methods.

Method	Cao et al. [2014]	Thies et al. [2016]	Our method (online)	Our method (batched)
Frames/s	28	28	287	870

Table 1: Throughput comparison with other facial performance capture methods.

5.5 Performance

Our system runs comfortably in real-time. As seen in Table 1, we achieve 287 frames per second when used online and up to 870 frames per second if batch processing is used offline with a batch size of 200 frames. Meanwhile, other real-time methods are only able to achieve 28 frames per second. This allows our system to process a large amount of footage in a short amount of time.

5.6 Limitations

We have proposed a system that can achieve high levels of accuracy for facial performance capture while drastically reducing the amount of manual work involved in a production setting. However, we have observed several limitations in our system and suggest future work we can explore.

Non-Optimal Loss Function. In validation shots for which numerical quality results could be computed, the visual quality of the network output did not always follow the value of the loss function. For example, a shot with a low loss value might exhibit unnatural movement of lips or eyes, whereas a shot with a higher loss value may look more believable. This suggests that our current loss function does not get the optimal results possible from a deep neural

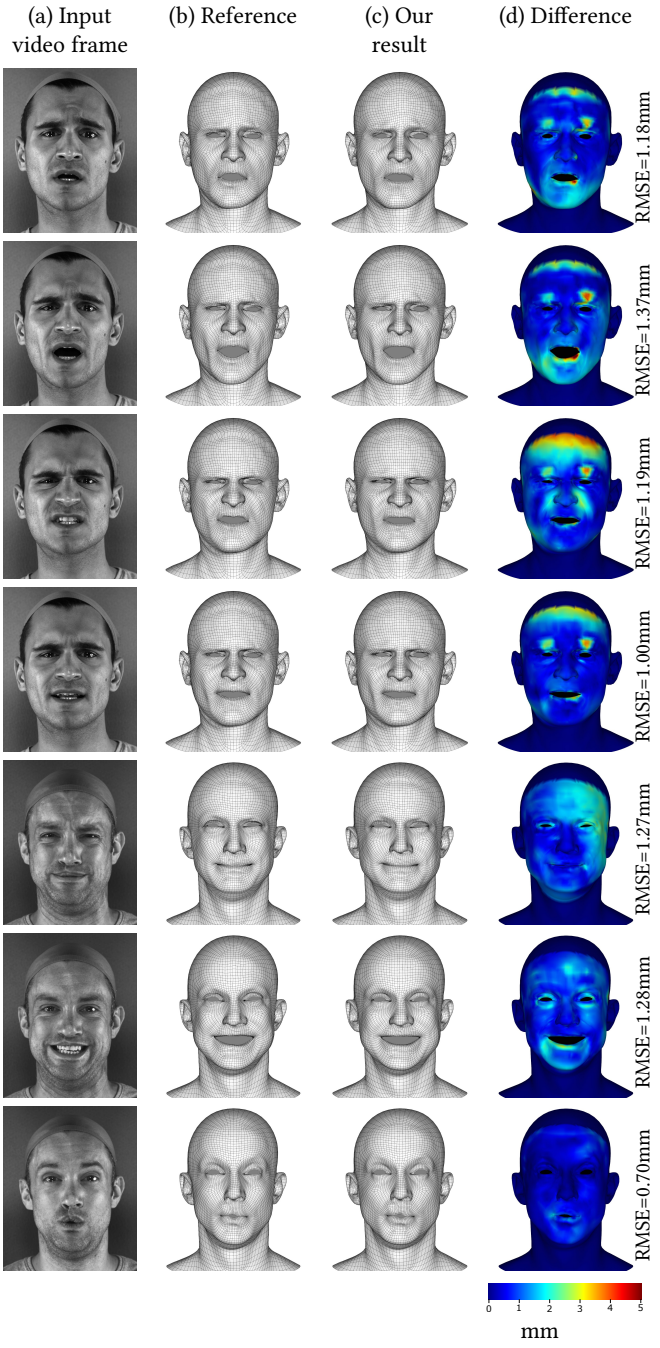


Figure 7: A selection of frames from two validation shots. (a) Crop of the original input image. (b) The target positions created by capture artists using the existing capture pipeline at Remedy Entertainment. (c) Our result inferred by the neural network based solely on the input image (a). (d) Difference between target and inferred positions. The RMSE is calculated over the Euclidean distances between target and inferred vertex positions with only the animated vertices taken into account.

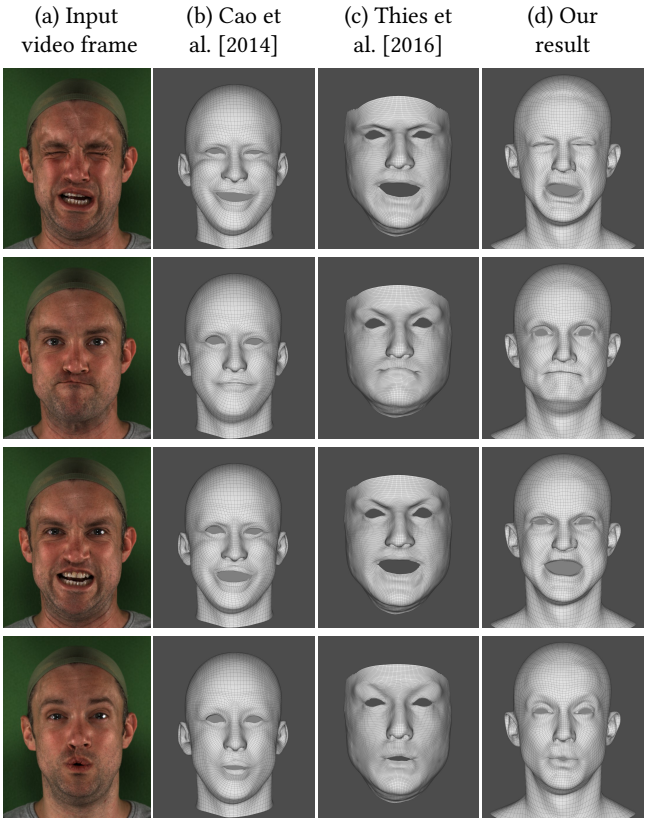


Figure 8: We compare our method (d) with Thies et al. [2016] (c) and Cao et al. [2014] (b) on various input images (a). We see that our method is more accurate and able to better capture details around the difficult mouth and eye regions.

network, and it should be beneficial to design a more perceptually oriented loss function for facial expressions similar in spirit of how structural similarity metrics have been developed for comparing images [Wang et al. 2004]. It seems clear that a better loss function would result in a more capable network, as it would learn to focus more on areas that require the highest precision.

Per-User Calibration. Despite our ability to capture details more accurately than other methods, one strong limitation of our method compared to other state-of-the-art monocular facial performance capture methods is that we require performing a per-user calibration of retraining the network for each new identity. In the future, we would like to further reduce the amount of manual labor involved in our pipeline and create a system that can achieve the same level of accuracy while also generalizing to all users.

6 CONCLUSION

We have presented a neural network based facial capture method that has proven accurate enough to be applied in an upcoming game production based on thorough pre-production testing while also requiring much less labor than other current facial performance capture pipelines used in game production. Another advantage our method holds in the production setting is that building the dataset

for the network enables tracking work to start any time, so pick-up shoots and screenplay changes are much faster to deliver with high quality.

We have evaluated our network architecture and training pipeline against other network and pipeline variations, and we determined the proposed architecture and augmentation methods to yield a very good balance between optimal visual results and reasonable training time for production purposes.

We have also shown our method to surpass other state-of-the-art monocular real-time facial performance capture methods in our ability to infer a plausible mesh around regions that are invisible or difficult to track such as the area surrounding the eye and mouth. However, our system has a significant drawback as we require per-user calibration. The 5–10 minute dataset required for each new identity for high-quality output typically means that the actor needs an important enough role in the game to justify the cost.

Even though the convolutional network may seem like an opaque building block, our approach retains all of the artistic freedom because we output simple 3D point clouds that can be further edited using standard tools, and compressed into standard character rigs. We feel that many other aspects of the production pipelines of modern games could benefit from similar, selective use of deep learning for bypassing or accelerating manual processing steps that have known but tedious or expensive solutions.

Future Work. Future work may include addressing the limitations of our system mentioned earlier and developing a more accurate pipeline that does not require per-user calibration. Additional work will also focus on capturing datasets using helmet-mounted cameras for true performance capture of the face and body simultaneously. Nevertheless, we have presented a system that has drastically reduced the amount of manual work in high quality facial performance capture, and our system represents an important step in the direction of fully automated, high quality facial and body capture.

REFERENCES

Oleg Alexander, Mike Rogers, William Lambeth, Matt Chiang, and Paul Debevec. 2009. The Digital Emily project: photoreal facial modeling and animation. In *ACM SIGGRAPH Courses (SIGGRAPH '09)*. ACM, New York, NY, USA, Article 12, 15 pages. https://doi.org/10.1145/1667239.1667251

Thabo Beeler, Fabian Hahn, Derek Bradley, Bernd Bickel, Paul Beardsley, Craig Gotsman, Robert W. Sumner, and Markus Gross. 2011. High-quality Passive Facial Performance Capture Using Anchor Frames. *ACM Trans. Graph.* 30, 4, Article 75 (2011), 10 pages. https://doi.org/10.1145/2010324.1964970

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proc. ICML*. 41–48. https://doi.org/10.1145/1553374.1553380

Amit Bermano, Thabo Beeler, Yeara Kozlov, Derek Bradley, Bernd Bickel, and Markus Gross. 2015. Detailed Spatio-temporal Reconstruction of Eyelids. *ACM Trans. Graph.* 34, 4 (2015), 44:1–44:11.

Kiran S. Bhat, Rony Goldenthal, Yuting Ye, Ronald Mallet, and Michael Koperwas. 2013. High Fidelity Facial Animation Capture and Retargeting with Contours. In *Proc. Symposium on Computer Animation*. 7–14.

Bernd Bickel, Manuel Lang, Mario Botsch, Miguel A. Otaduy, and Markus Gross. 2008. Pose-space animation and transfer of facial details. In *ACM SCA*. 57–66.

Volker Blanz and Thomas Vetter. 1999. A Morphable Model for the Synthesis of 3D Faces. In *Proc. ACM SIGGRAPH*. 187–194.

George Borshukov, Dan Piponi, Oystein Larsen, J. P. Lewis, and Christina Tempelaar-Lietz. 2005. Universal capture - image-based facial animation for "The Matrix Reloaded". In *ACM SIGGRAPH Courses (SIGGRAPH '05)*. ACM, New York, NY, USA, Article 16. https://doi.org/10.1145/1198555.1198596

Sofien Bouaziz, Yangang Wang, and Mark Pauly. 2013. Online Modeling for Realtime Facial Animation. *ACM Trans. Graph.* 32, 4 (2013), 40:1–40:10.

Derek Bradley, Wolfgang Heidrich, Tiberiu Popa, and Alla Sheffer. 2010. High resolution passive facial performance capture. *ACM TOG* 29, Article 41 (2010), 10 pages. Issue 4. https://doi.org/10.1145/1778765.1778778

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

Chen Cao, Derek Bradley, Kun Zhou, and Thabo Beeler. 2015. Real-time High-fidelity Facial Performance Capture. *ACM Trans. Graph.* 34, 4 (2015), 46:1–46:9.

Chen Cao, Qiming Hou, and Kun Zhou. 2014. Displaced Dynamic Expression Regression for Real-time Facial Tracking and Animation. *ACM Trans. Graph.* 33, 4 (2014), 43:1–43:10.

Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D Shape Regression for Real-time Facial Animation. *ACM Trans. Graph.* 32, 4 (2013), 41:1–41:10.

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

Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. 2001. Active Appearance Models. *IEEE TPAMI* 23, 6 (2001), 681–685.

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

Dimensional Imaging. 2016. DI4D PRO System. http://www.di4d.com/systems/di4d-pro-system/. (2016).

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.

P. Ekman and W. Friesen. 1978. *The Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Consulting Psychologists Press, Palo Alto.

Yasutaka Furukawa and Jean Ponce. 2009. Dense 3D motion capture for human faces. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Graham Fyffe, Tim Hawkins, Chris Watts, Wan-Chun Ma, and Paul Debevec. 2011. Comprehensive facial performance capture. In *Computer Graphics Forum*, Vol. 30. 425–434.

Graham Fyffe, Andrew Jones, Oleg Alexander, Ryosuke Ichikari, and Paul Debevec. 2014. Driving High-Resolution Facial Scans with Video Performance Capture. *ACM Trans. Graph.* 34, 1 (2014), 8:1–8:14.

Pablo Garrido, Michael Zollhöfer, Chenglei Wu, Derek Bradley, Patrick Pérez, Thabo Beeler, and Christian Theobalt. 2016. Corrective 3D Reconstruction of Lips from Monocular Video. *ACM Trans. Graph.* 35, 6 (2016), 219:1–219:11.

Brian Guenter, Cindy Grimm, Daniel Wood, Henrique Malvar, and Fredric Pighin. 1998. Making faces. In *ACM SIGGRAPH*. 55–66. https://doi.org/10.1145/280814.280822

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

Pei-Lun Hsieh, Chongyang Ma, Jihun Yu, and Hao Li. 2015a. Unconstrained Realtime Facial Performance Capture. In *IEEE CVPR*. 1675–1683.

Pei-Lun Hsieh, Chongyang Ma, Jihun Yu, and Hao Li. 2015b. Unconstrained realtime facial performance capture. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. 1675–1683.

Alexandru Eugen Ichim, Sofien Bouaziz, and Mark Pauly. 2015. Dynamic 3D Avatar Creation from Hand-held Video Input. *ACM Trans. Graph.* 34, 4 (2015), 45:1–45:14.

Vahid Kazemi and Josephine Sullivan. 2014. One Millisecond Face Alignment with an Ensemble of Regression Trees. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proc. NIPS*. 1097–1105.

Hao Li, Thibaut Weise, and Mark Pauly. 2010. Example-based facial rigging. In *Acm transactions on graphics (tog)*, Vol. 29. ACM, 32.

Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. 2013. Realtime Facial Animation with On-the-fly Correctives. *ACM Trans. Graph.* 32, 4 (2013), 42:1–42:10.

Yilong Liu, Feng Xu, Jinxiang Chai, Xin Tong, Lijuan Wang, and Qiang Huo. 2015. Video-audio Driven Real-time Facial Animation. *ACM Trans. Graph.* 34, 6 (2015), 182:1–182:10.

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

Kyle Olszewski, Joseph J. Lim, Shunsuke Saito, and Hao Li. 2016. High-fidelity Facial and Speech Animation for VR HMDs. *ACM Trans. Graph.* 35, 6 (2016), 221:1–221:14.

Photoscan. 2014. Agisoft. (2014). http://www.agisoft.com/

Fred Pighin and J. P. Lewis. 2006. Performance-driven facial animation. In *ACM SIGGRAPH Courses (SIGGRAPH '06)*.

F. Pughin and J. P. Lewis. 2006. Performance-driven facial animation. In *ACM SIGGRAPH 2006 Courses*.

Shunsuke Saito, Tianye Li, and Hao Li. 2016. Real-Time Facial Segmentation and Performance Capture from RGB Input. *CoRR* abs/1604.02647 (2016). http://arxiv.org/abs/1604.02647

Jason M. Saragih, Simon Lucey, and Jeffrey F. Cohn. 2011. Deformable Model Fitting by Regularized Landmark Mean-Shift. *IJCV* 91, 2 (2011), 200–215. https://doi.org/10.1007/s11263-010-0380-4

Fuhao Shi, Hsiang-Tao Wu, Xin Tong, and Jinxiang Chai. 2014. Automatic Acquisition of High-fidelity Facial Performances Using Monocular Videos. *ACM Trans. Graph.* 33, 6 (2014), 222:1–222:13.

Patrice Y. Simard, Dave Steinkraus, and John Platt. 2003. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *Proc. ICDAR*,

Vol. 3. 958–962.
Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).
Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. *arXiv preprint arXiv:1412.6806* (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.
Sarah L. Taylor, Moshe Mahler, Barry-John Theobald, and Iain Matthews. 2012. Dynamic Units of Visual Speech. In *Proc. SCA*. 275–284.
Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016).
Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. 2015. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.* 34, 6 (2015), 183.
J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. 2016. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *Proc. Computer Vision and Pattern Recognition (CVPR)*.
P. A. Tresadern, M. C. Ionita, and T. F. Cootes. 2012. Real-Time Facial Feature Tracking on a Mobile Device. *Int. J. Comput. Vision* 96, 3 (2012), 280–289.
Levi Valgaerts, Chenglei Wu, Andrés Bruhn, Hans-Peter Seidel, and Christian Theobalt. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM TOG* 31, 6, Article 187 (Nov. 2012), 11 pages. https://doi.org/10.1145/2366145.2366206
Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. 2005. Face transfer with multilinear models. In *ACM SIGGRAPH (SIGGRAPH '05)*. ACM, New York, NY, USA, 426–433. https://doi.org/10.1145/1186822.1073209
Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE TIP* 13, 4 (2004), 600–612.
Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime Performance-based Facial Animation. *ACM Trans. Graph.* 30, 4 (2011), 77:1–77:10.
Thibaut Weise, Hao Li, Luc Van Gool, and Mark Pauly. 2009a. Face/Off: Live Facial Puppetry. In *Proc. SCA*. 7–16.
Thibaut Weise, Hao Li, Luc Van Gool, and Mark Pauly. 2009b. Face/off: Live facial puppetry. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer animation*. ACM, 7–16.
Yanlin Weng, Chen Cao, Qiming Hou, and Kun Zhou. 2014. Real-time facial animation on mobile devices. *Graphical Models* 76, 3 (2014), 172–179.
Lance Williams. 1990a. Performance-driven Facial Animation. *SIGGRAPH Comput. Graph.* 24, 4 (1990), 235–242.
Lance Williams. 1990b. Performance-driven facial animation. *ACM SIGGRAPH* 24, 4 (1990), 235–242. https://doi.org/10.1145/97880.97906
Chenglei Wu, Derek Bradley, Markus Gross, and Thabo Beeler. 2016. An Anatomically-constrained Local Deformation Model for Monocular Face Capture. *ACM Trans. Graph.* 35, 4 (2016), 115:1–115:12.
Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. 2004. Spacetime Faces: High Resolution Capture for Modeling and Animation. *ACM Trans. Graph.* 23, 3 (2004), 548–558. https://doi.org/10.1145/1015706.1015759
Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2015. Appearance-based Gaze Estimation in the Wild. In *Proc. Computer Vision and Pattern Recognition (CVPR)*. 4511–4520.