University of Massachusetts Amherst

ScholarWorks@UMass Amherst

Doctoral Dissertations | Dissertations and Theses

October 2021

Audio-driven Character Animation

Yang Zhou
University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/dissertations_2

Part of the Artificial Intelligence and Robotics Commons, and the Graphics and Human Computer Interfaces Commons

Recommended Citation
Zhou, Yang, "Audio-driven Character Animation" (2021). *Doctoral Dissertations*. 2393.
https://doi.org/10.7275/23710166 https://scholarworks.umass.edu/dissertations_2/2393

This Open Access Dissertation is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

AUDIO-DRIVEN CHARACTER ANIMATION

A Dissertation Outline Presented

by

YANG ZHOU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2021

College of Information and Computer Sciences

© Copyright by Yang Zhou 2021

All Rights Reserved

AUDIO-DRIVEN CHARACTER ANIMATION

A Dissertation Outline Presented

by

YANG ZHOU

Approved as to style and content by:

Evangelos Kalogerakis, Chair

Rui Wang, Member

Liangliang Cao, Member

Dingzeyu Li, Member

Eli Shechtman, Member

James Allan, Chair
College of Information and Computer Sciences

ACKNOWLEDGMENTS

It is hard to believe my Ph.D. study is reaching an end. It feels like yesterday when I stepped into my lab and started my first project. There are many important things happened during these five years: my marriage, the birth of my daughter, the world-wide pandemic and remote working, all of which caused a big change of my life. Fortunately, I have gone through them with the help and support I have received from so many people, without which this dissertation would not have been possible.

First and foremost, I would like to thank my dear advisor, Evangelos Kalogerakis, for his invaluable advice, continuous support, great trust and patience during my Ph.D. study. His professional guidance and immense knowledge helped me in all the time of my academic research and daily life. For each of my projects, he enthusiastically provided me with insightful suggestions and related articles. For each of my papers, he patiently pointed out logic flaws and unclear descriptions. He taught me how to be a good independent researcher. I can still remember the tremendous help from him during my first thesis-research paper submission. At that time, my daughter was born prematurely one week before the submission deadline. I felt very excited to be a father and was also nervous about finishing my first paper submission. Evangelos understood my difficulty and helped a lot on the paper together with other collaborators. He gave me enough time with my family and took over a lot of the work. The paper was finally successfully submitted and published. To thank and memorize the help of Evangelos and all the other collaborators, I named my daughter *Mia*, which is derived from *Maya*, the name of the software we used throughout this project.

Secondly, I would like to thank Dingzeyu Li and Jimei Yang for mentoring me during my internship at Adobe Research. Due to the collaboration with them, I had great freedom of the research topics and had the opportunity to investigate my passion about the character animation. They have professional experience in these directions and gave me strong guidance and constructive feedback. I also thank Eli Shechtman who gave me great advice on generative models and deep learning. I thank Jose Echevarria who foresaw many potential obstacles in my character animation projects and helped me to overcome them. I also thank Jun Saito who helped me to solve animation kinematics problems and Deepali Aneja who gave me great advice in audio-visual learning. I would also like to thank Wilmot Li and Scott Cohen who helped manage my internship experience and offered me the great opportunity of work position at Adobe. Without their precious support, it would not be possible to conduct this thesis.

Thirdly, I would like to thank my close collaborators Zhan Xu, Chris Landreth and Prof. Karan Singh. We worked on many exciting projects towards better and easier character animation for animators. Zhan Xu is my closest collaborator other than my advisor. Both of us have great passion in the animation research field. I would like to thank him for his kind help and great contribution to my research projects. Chris Landreth is the greatest animator I ever met. He turned my research outcome into gorgeous animation videos. He also gave me lots of insightful suggestions from the animator's perspective view. Prof. Karan Singh is a pioneer in the field of character animation. It is always fun and inspiring to chat and collaborate with him, where he often enlightened me with his great vision in this field.

Besides my collaborators, I would like to thank the rest of my thesis committee: Prof. Rui Wang and Prof. Liangliang Cao for their insightful comments and encouragement which helped me widen my research from various perspectives.

I am glad to be part of the Computer Graphics and Vision lab at UMass Amherst. I would like to thank my fellow lab-mates: Zhan Xu, Difan Liu, Haibin Huang, Zhaolinag Lun, Gopal Sharma, Pratheba Selvaraju, Dmitrii Petrov, Matheus Gadelha, Hang Su, Huaizu Jiang, Chenyun Wu, Zezhou Cheng, Aruni RoyChowdhury, SouYong Jin, Jong-Chyi Su, Tsung-Yu Lin, Ashish Singh for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last five years. Also I thank my friends Xintong Han from Huya Inc., Tim Zhang from Wayfair Inc, Prof. Tomer Weiss from New Jersey Institute of Technology for supporting and enlightening me the direction of my research.

Last but not the least, I would like to give a big thank to my family. My wife, Fan Lu, is a designer and helped me a lot in drawing figures in my thesis. She also gave me a new perspective view of my animation results from a designer. My parents and parents-in-law supported me spiritually throughout writing this thesis and my life in general. I also thank my daughter, Mia, whose smile continuously gives me energy. This dissertation is dedicated to them.

ABSTRACT

AUDIO-DRIVEN CHARACTER ANIMATION

SEPTEMBER 2021

YANG ZHOU

B.E., SHANGHAI JIAO TONG UNIVERSITY

M.E., SHANGHAI JIAO TONG UNIVERSITY

M.E., GEORGIA INSTITUTE OF TECHNOLOGY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Evangelos Kalogerakis

Generating believable character animations is a fundamentally important problem in the field of computer graphics and computer vision. It also has a diverse set of applications ranging from entertainment (e.g., films, games), medicine (e.g., facial therapy and prosthetics), mixed reality, and education (e.g., language/speech training and cyber-assistants). All these applications are all empowered by the ability to model and animate characters convincingly (human or non-human). Existing key-framing or performance capture approaches used for creating animations, especially facial animations, are either laborious or hard to edit. In particular, producing expressive animations from input speech automatically remains an open challenge.

In this thesis, I propose novel deep-learning based approaches to produce speech audio-driven character animations, including talking-head animations for character face rigs and portrait images, and reenacted gesture animations for natural human speech videos.

First, I propose a neural network architecture, called *VisemeNet*, that can automatically animate an input face rig using audio as input. The network has three stages: one that learns to predict a sequence of phoneme-groups from audio; another that learns to predict the geometric location of important facial landmarks from audio; and a final stage that combines the outcome from previous stages to produce animation motion curves for FACS-based (Facial Action Coding System-based) face rigs.

Second, I propose *MakeItTalk*, a method that takes as input a portrait image of a face along with audio, and produces the expressive synchronized talking-head animation. The portrait image can range from artistic cartoons to real human faces. In addition, the method generates the whole head motion dynamics matching the audio stresses and pauses. The key insight of the method is to disentangle the content and speaker identity in the input audio signals, and drive the animation from both of them. The content is used for robust synchronization of lips and nearby facial regions. The speaker information is used to capture the rest of the facial expressions and head motion dynamics that are important for generating expressive talking head animations. I also show that *MakeItTalk* can generalize to new audio clips and face images not seen during training. Both *VisemeNet* and *MakeItTalk* lead to much more expressive talking-head animations with higher overall quality compared to the state-of-the-art.

Lastly, I propose a method that generates speech gesture animation by reenacting a given video to match a target speech audio. The key idea is to split and reassemble clips from an existing reference video through a novel video motion graph encoding valid transitions between clips. To seamlessly connect different clips in the reenactment, I propose a pose-aware video blending network which synthesizes video frames around the stitched frames between two clips. Moreover, the method incorporates an audio-based gesture searching algorithm to find the optimal order of the reenacted frames. The method generates reenactments that are consistent with both the audio rhythms and the speech content. The resulting synthesized videos have much higher quality and consistency with the target audio compared to previous work and baselines.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Motivation

The importance of realistic computer generated human animations cannot be understated. A diverse set of applications ranging from entertainment (movies and games), medicine (facial therapy and prosthetics) and education (language/speech training and cyber-assistants) are all empowered by the ability to realistically model, simulate and animate human characters. Imperfect emulation of even subtle nuance of the character, e.g. facial expressions or hand gestures, can plunge an animated character into the Uncanny Valley, where the audience loses trust and empathy with the character. Paradoxically, the greater the rendered realism of the character, the less tolerant we are of flaws in its animation [89]. In particular, the expressive animation of speech, unsurprisingly, is a critical component of character facial and gesture animation that has been the subject of research for decades [8, 93, 69].

In terms of talking-head (or facial animation), keyframing or performance capture are used for high-end commercial animation. Keyframing by professional animators is both expressive and editable, but laborious and prohibitively expensive in time and effort. Performance capture solutions are the opposite; recording a vocal performance is relatively easy, but hard to further edit or refine, and voice actors often have visually inexpressive faces. Despite recent advances, generating realistic facial animation with little or no manual labor still remains an open challenge in computer graphics. Several key factors contribute to this challenge. Traditionally, the *synchronization* between speech and facial movement is hard to achieve manually. Facial dynamics lie on a high-dimensional manifold, making it nontrivial to find a mapping from audio/speech [39]. Secondly, different talking *styles* in multiple talking-heads can convey different personalities and lead to better viewing experiences [129]. Last but not least, handling lip syncing and facial animation are not sufficient for the perception of *realism* of talking-heads. The entire facial expression considering the correlation between all facial elements and head pose also play an important role [47, 57]. These correlations, however, are less constrained by the audio and thus hard to be estimated.

Moreover, gesture is also a key visual component for human speech communication [69]. It enhances the expressiveness of human performance and helps the audience to better comprehend the speech content [38]. Unlike ahead facial lip motions with phoneme-to-viseme mappings [39, 120, 153], gestures exhibit even more complex relationships with not only acoustics but also semantics of the audio [93]. To bridge the gap between speech audio and natural human video, previous methods [53, 82] predict body pose (i.e., a jointed skeleton) as an intermediate low dimensional representation to drive the video synthesis. However, they dissect the problem into two independent modules (audio-to-pose, and pose-to-video) and produce results suffering from noticeable artifacts, e.g. distorted body parts and blurred appearance. Therefore, it is nontrivial to find a direct cross-modal mapping from audio waveform to gesture videos, even for the same speaker.

The main focus of this thesis is to address the above challenges to produce expressive and realistic talking head and gesture animations directly from input speech audio alone. I also explore different types of the animation output, such as rigged avatar faces for professional animators and 2D cartoon characters and photorealistic human photos for casual users.

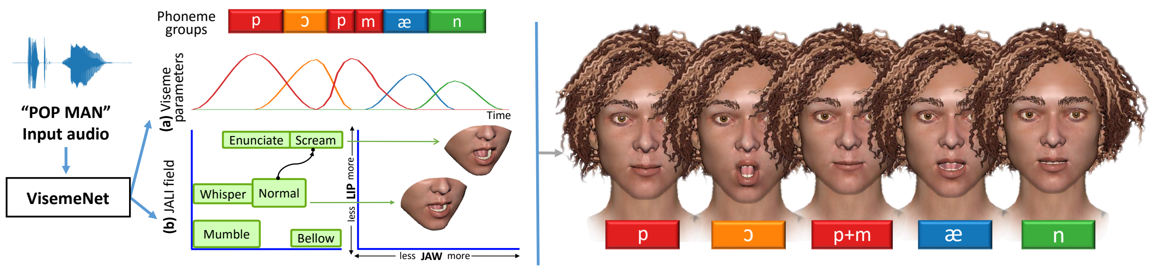


Figure 1.1: VisemeNet is a deep-learning approach that uses a 3-stage LSTM network, to predict compact animator-centric viseme curves with proper co-articulation, and speech style parameters, directly from speech audio in near real-time (120ms lag).

1.2 VisemeNet: Audio-Driven Animator-Centric Speech Animation

I first propose a method, called VisemeNet, to address the problem of producing animator-centric speech animation directly from input audio (Fig. 1.1). The method builds upon JALI [39], a psycho-linguistically inspired face rig approach capable of animating a range of speech styles. In addition, JALI is animator-centric, allowing animators to control speech style (e.g., "mumbling" vs "screaming"), while also providing an interactive lip-synchronization procedure based on audio and speech transcript as input. Unfortunately, JALI requires a text transcript as input and also manual control of speech style or/and speech animation curves through the JALI input parameter space. My work in the first part of my thesis addresses the problem of producing animator-centric speech animation automatically and directly from input audio.

VisemeNet is inspired by the following psycho-linguistic observations:

- While classification of a precise phonetic stream from audio can be difficult due to categorical perception [51], and its dependence on cultural and linguistic context, the problem of predicting a stream of phoneme-groups is simpler. Aurally individual phonemes within a phoneme-group are often hard to distinguish (eg.

pa and ba) [83], but unnecessary for speech animation as they map to near identical visemes [49].

- The Jaw and Lip parameters in the JALI model that capture speech style as a combination of the contribution of the tongue-jaw and face-muscles to the produced speech, are visually manifested by the motion of facial landmarks on the nose, jaw and lips.

- The profile of speech motion curves (attributes like onset, apex, sustain and decay) capture speaker or animator style in professionally keyframed animation. Learning these curves from training data allow us to encode aspects of animator or speaker style.

I thus propose a three-stage network architecture trained end-to-end: one that learns to predict a sequence of phoneme-groups from audio; another that learns to predict the geometric location of important facial landmarks from audio; and a final stage that learns to use phoneme groups and facial landmarks to produce JALI parameter values and sparse speech motion curves, that animate the face.

The contribution of this thesis is thus a deep-learning based architecture to produce state-of-the-art animator-centric speech animation directly from an audio signal in near real-time (120ms lag). The evaluation is five-fold: I evaluate the results quantitatively by cross-validation to ground-truth data; I also provide a qualitative critique of the results by a professional animator; as well as the ability to further edit and refine the animated output; I also show the results to be comparable with recent non-animator-centric audio to lip-synchronization solutions; finally I show that with speech training data that is reasonably diverse in speaker, gender and language, the architecture can provide a truly language agnostic solution to speech animation from audio.

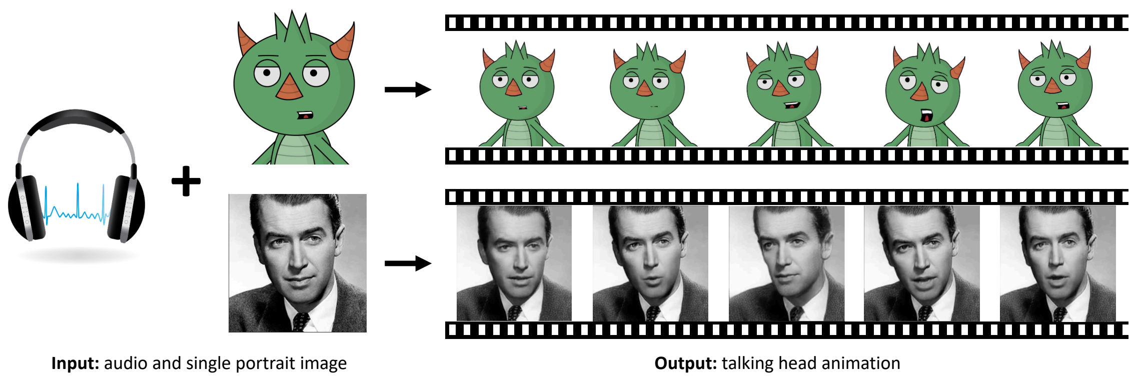

Input: audio and single portrait image
Output: talking head animation

Figure 1.2: Given an audio speech signal and a single portrait image as input (left), the model generates speaker-aware talking-head animations (right). *Both the speech signal and the input face image are not observed during the model training process.* the method creates both non-photorealistic cartoon animations (top) and photorealistic human face videos (bottom).

1.3 MakeItTalk: Speaker-Aware Talking-Head Animation

To animate a 3D model of a head, VisemeNet requires that it is already rigged based on FACS [40] or JALI [39]. Rigging a 3D head model often requires artistic expertise and professional training. In the second part of the thesis, I propose another method, called *MakeItTalk*, that enables 3D talking head animations even for amateur users that do not have any rigging expertise. The method generates talking-heads from a single facial image and audio as the only input (Fig. 1.2). At test time, MakeItTalk is able to produce plausible talking-head animations with both facial expressions and head motions for new faces and voices not observed during training.

Mapping audio to facial animation is challenging, since it is not a one-to-one mapping. Different speakers can have large variations in head pose and expressions given the same audio content. The key insight of the approach is to disentangle the speech content and speaker identity information in the input audio signal. The content captures the phonetic and prosodic information in the input audio and is used for robust *synchronization* of lips and nearby facial regions. The speaker information captures the rest of the facial expressions and *head motion dynamics* that are important for generating expressive talking-head animation. I demonstrate that this disentanglement leads to significantly more plausible and believable head animations.

Another key component of the method is the prediction of facial landmarks as an intermediate representation incorporating speaker-aware dynamics. This is in contrast to previous approaches that attempt to directly generate raw pixels from the audio. Leveraging facial landmarks as the intermediate representation between audio to visual animation has several advantages. First, based on the disentangled representations, the model learns to generate landmarks that capture subtle, speaker-dependent dynamics, sidestepping the learning of low-level pixel appearance that tends to miss those. Second, the degrees of freedom (DoFs) for landmarks is in the order of tens (68 in the implementation), as opposed to millions of pixels in raw video generation methods. As a result, the learned model is also compact, making it possible to train it from moderately sized datasets. Last but not least, the landmarks can be easily used to drive a wide range of different types of animation content, including photorealistic human face images and non-photorealistic cartoon images, such as sketches, 2D cartoon characters, Japanese mangas and stylized caricatures. Specifically, in the case of photorealistic animation, an image-to-image translation framework [68, 104] is implemented to convert landmarks to realistic image sequences can produce plausible videos. In the case of non-photorealistic outputs (e.g., cartoon images), an image deformation approach can work well and preserve feature curves common in vector art.

In summary, given an audio signal and a single portrait image as input (both unseen during training), the method generates expressive talking-head animations. I highlight the following contributions:

- a new deep-learning based architecture is introduced to predict facial landmarks, capturing both facial expressions and overall head poses, from only speech signals.

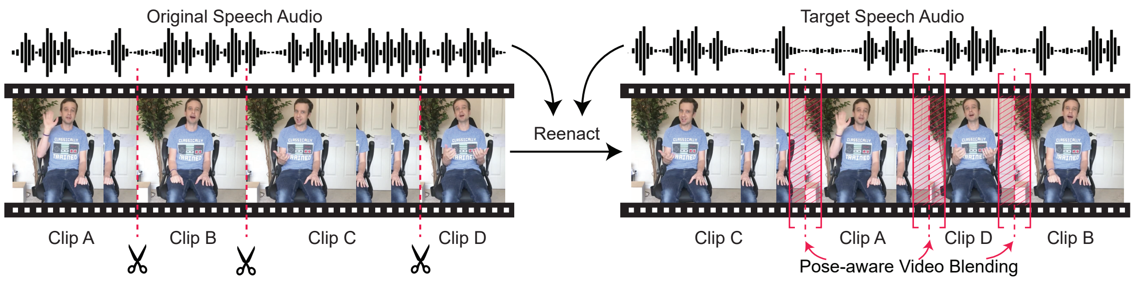


Figure 1.3: Given an input reference video of a speaker (left), the method reenacts it with gestures matching a target speech audio (right). The video is synthesized by re-assembling clips from the reference video and blending the inconsistent boundaries with a pose-aware neural network such that the synthesized video is coherent visually and consistent with both the rhythm and content of the target audio.

- speaker-aware talking-head animations are generated based on disentangled speech content and speaker information, inspired by advances from voice conversion.

- two image synthesis methods are presented for both non-photorealistic cartoon images and photorealistic natural human face images. These methods can handle new faces and cartoon characters not observed during training.

- a set of quantitative metrics and conduct user studies are proposed for evaluation of talking-head animation methods.

1.4 Audio-driven Neural Gesture Reenactment with Video Motion Graph

Human speech is often accompanied by body gestures including arm and hand gestures. To take one more step towards building automatic avatar animations, I propose the third method in my thesis, an audio-driven gesture reenactment system, that is able to synthesize high-resolution, high-quality speech gesture videos directly by cutting, re-assembling, and blending clips from a single input reference video (Fig. 1.3). The process is driven by a novel *video motion graph*, inspired by 3D motion graphs used in character animation [76, 6]. The graph nodes represent frames in the reference video, and edges encode possible transitions between them. Possible valid transitions between frames are discovered, then paths in the graph leading to the generation of a new video are also discovered such that the re-enacted gestures are coherent and consistent with both the audio rhythms and speech content of the target audio.

Direct playback on the discovered paths for an output video can cause temporal inconsistency at the boundary of two disjoint raw frames. Existing frame blending methods cannot easily solve this problem, especially with fast moving and highly deformed human poses. Therefore, I also propose a novel human *pose-aware video blending* network to smoothly blend frames around the temporally inconsistent boundaries to produce naturally-looking video transitions. By doing so, I successfully transform the problem of audio-driven gesture reenactment into the search for valid paths that best match the given audio. The path discovery algorithm is motivated by psychological studies on co-speech gesture analysis. The studies show co-speech gestures can be categorized into rhythmic gestures and referential gestures [93]. While rhythmic gestures are well synchronized with audio onsets [14, 147], referential gestures mostly co-occur with certain phrases, e.g. a greeting gesture of hand-waving appears when a speaker says 'hello' or 'hi' [25, 13]. I analyze the speech of the reference video and detect the audio onset peaks [35] as well as a set of keywords from its transcript [142] as audio features added to the corresponding nodes on the video motion graph. Given the extracted audio onset peaks and keywords from a new audio clip, the optimal paths that best match audio features are used to drive the video synthesis.

The contributions in this part of the thesis are summarized as follows:

- a new system that creates high-quality human speech videos with realistic gestures driven by audio only,

- a novel video motion graph that preserves the video realism and gesture subtleties,
- a pose-aware video blending neural network that synthesizes smooth transitions of two disjoint reference video clips along graph paths and
- an audio-based search algorithm that drives the video synthesis such that the synthesized gesture frames are consistent with both the audio rhythms and the speech content.

CHAPTER 2

LITERATURE REVIEW

In this chapter, I review the most relevant work on *speech* audio-driven character animation, including facial and body gesture animation. In computer graphics and vision, there is a long history of such cross-modal synthesis, as discussed in the following sections.

2.1 Audio-driven Facial Animation

More than two decades ago, Brand *et al.* [15] pioneered *Voice Puppetry* to generating full facial animation from an audio track. Later on, a large body of research explores this direction and it can be broadly classified into *procedural*, *performance-capture*, *data-driven*, and more specifically the recent *deep-learning* based techniques. In the following paragraphs, we overview prior work based on deep learning methods, after a brief overview of the other three approaches.

Procedural. Procedural speech animation segments audio speech into a sequence of phonemes, which are then mapped by rules to visemes. A *viseme* or *visible phoneme* [49] refers to the shape of the mouth at the apex of a given phoneme. The three problems that a procedural approach must solve are: *mapping* a given phoneme to a viseme (in general a many-many mapping based on the spoken context [121]); *co-articulation*, or the overlap in time between successive visemes, resulting from the fluid and energy efficient nature of human speech, often addressed using Cohen and Massaro's [32] seminal *dominance model*; *viseme profile*, or the curve shape that defines the attack, apex, sustain and decay of a viseme over time [7] . JALI [39] defines the state of the art in procedural speech animation, producing compact animator-friendly motion curves that correspond directly to the input phonetic stream. The first part of my thesis is built upon JALI that automatically creates animator-centric motion curves from input audio alone.

Performance-capture. Performance-capture based speech animation transfers motion data captured from a human performer onto a digital face [140]. Performance capture has become increasingly powerful and mainstream with the widespread adoption of cameras, depth sensors, and reconstruction techniques, that can produce a 3D face model from a single image [64]. Real-time performance-based facial animation research [137, 80] and products like *Faceware* (*faceware.com*), are able to create high quality general facial animation, including speech, and can be further complemented by speech analysis [137]. The disadvantage of performance capture is that is visually limited by the actor's performance and is difficult for an animator to edit or refine.

Data-driven. Early data-driven approaches smoothly stitch pieces of facial animation data from a large corpus, to match an input speech track [16], using morphable [45], hidden Markov [132], and active appearance models (AAM) [5, 121]. These data-driven methods tend to be limited in scope to the data available, and the output, like performance-capture, is not animator-centric.

Deep Learning-based Speech Animation. Recent research has shown the potential of deep learning to provide a compelling solution to automatic lip-synchronization simply using an audio signal [119, 72, 120]. To consider different key criteria in talking-head generation, we further overview the prior deep learning-based works on facial landmark synthesis, facial animation, and video generation aspects. Table 2.1 summarizes differences of deep-learning methods that are most related to ours based on a set of key criteria.

	format	facial expression	head pose	style-aware	handle unseen faces
[119]	image	✓	✓	×	×
[120]	face rig	×	×	×	✓
[72]	3D mesh	✓	×	×	×
[151]	image	✓	×	×	✓
[128]	image	✓	×	×	×
[28]	image	✓	×	×	✓
[123]	image	✓	×	✓	×
VisemeNet	face rig	×	×	✓	✓
MakeItTalk	image	✓	✓	✓	✓

Table 2.1: A comparison of related works across to various criteria shown on top. "Handle unseen faces" means handling face images or rigs unobserved during training.

- **Audio-driven Facial Landmark Animation.** Eskimez *et al.* [42, 43] generated synchronized facial landmarks with robust noise resilience using deep neural networks. Later, Chen *et al.* [28] trained decoupled blocks to obtain landmarks first and then generate rasterized videos. Attention masks are used to focus on the most changing parts on the face, especially the lips. Greenwood *et al.* [57] jointly learnt facial expressions and head poses in terms of landmarks from a forked Bi-directional LSTM network. Most previous audio-to-face-animation work focused on matching speech content and left out style/identity information since the identity is usually bypassed due to mode collapse or averaging during training. In contrast, the second proposed method of my thesis disentangles audio content and speaker information, and drives landmarks capturing speaker-dependent dynamics.

- **Audio-driven Facial Lip Synchronization.** With the increasing power of GPUs, we have seen prolific progress on end-to-end learning from audio to video frames, such as predicting lip movement [27, 120], generating full faces with GANs [128, 116] or encoder-decoder CNNs [30], recognizing visemes [153], and estimating blendshape parameters [100]. However, the above methods do not capture speaker identity or style. As a result, if the same sentence is spoken by two different voices, they will tend to generate the same facial animation lacking the dynamics required to make it more expressive and realistic.

- **Audio-driven Style-aware Facial Animation.** Suwajanakorn *et al.* [119] used a re-timing dynamic programming method to reproduce speaker motion dynamics. However, it was specific to a single subject (Obama), and does not generalize to faces other than Obama's. Cudeiro *et al.* [34] attempts to model speaker style in a latent representation. Thies *et al.* [123] encodes personal style in static blendshape bases. Both methods, however, focus on lower facial animation, especially lips, and do not predict head pose. More similar to ours, Zhou *et al.* [151] learned a joint audio-visual representation to disentangle the identity and content from the image domain. However, their identity information primarily focus on static facial appearance and not the speaker dynamics. Speaker awareness encompasses many aspects beyond mere static appearances. The individual facial expressions and head movements are both important factors for speaker-aware animations. Our method addresses speaker identity by learning jointly the static appearance and head motion dynamics, to deliver faithfully animated talking-heads.

- **Audio Content and Speaker Style Disentanglement** Disentanglement of content and style in audio has been widely studied in the voice conversion community. Without diving into its long history (see [118] for a detailed survey), here we only discuss recent methods that fit into our deep learning pipeline. Wan *et al.* [130] developed *Resemblyzer* as a speaker identity embedding for verification purposes across different languages. Qian *et al.* [102] proposed AutoVC, a few-shot voice conversion method to separate the audio into the speech content and the identity information. As a baseline, we use AutoVC for extracting voice content and Resemblyzer for extracting feature embeddings of speaker identities. In the second part of my thesis, we introduce the idea of voice conversion to audio-driven animation and demonstrate the advantages of *speaker-aware* talking-head generation.

- **Image Translation for Human Face Synthesis.** Neural image translation approach is widely used recently for talking face synthesis and editing [68, 135, 134, 73]. Face2Face and VDub are among the early explorers to demonstrate robust appearance transfer between two talking-head videos [124, 52]. Later, adversarial training was adopted to improve the quality of the transferred results. For example, Kim *et al.* [74] used cycle-consistency loss to transfer styles and showed promising results on one-to-one transfers. Zakharov *et al.* [148] developed a few-shot learning scheme that leveraged landmarks to generate realistic faces. Based on these prior works, we also employ an image-to-image translation network to generate realistic talking-head animations. Unlike Zakharov *et al.* [148], the second part of my thesis, MakeItTalk, handles generalization to faces unseen during training without the need of fine-tuning. Additionally, we are able to generate non-photorealistic cartoon images through an image deformation module.

2.2 Audio-driven Human Gesture Animation

Several approaches for audio-driven speech animation of body gestures have been proposed in the recent years [3, 78, 146, 4]. They propose learning methods to solve the multi-modal mapping from audio to 3D human gestures. They represent synthesized gestures with 3D skeletons, which can drive a 3D character model. Yet, these methods are not able to synthesize video of a target speaker unless they are also provided with an extremely detailed, textured, and rigged 3D model for that speaker. Since such input is not easily obtained, their demonstrated results lack photorealism. Instead, in the third part of my thesis, the proposed method is a reenactment system, that is able to synthesize high-resolution, high-quality photorealistic speech gesture videos directly by cutting, re-assembling, and blending clips from a single input reference video. The method is related to previous work on motion graphs, human video synthesis, and video frame blending. In the following paragraphs, we overview prior work in such directions.

Motion Graph. The idea of motion graphs was first proposed in [76, 6] to create realistic and controllable animation based on a pre-captured motion. It is broadly used in generating 3D character animations [61, 111, 10, 79, 103, 94, 107, 77]. However, these approaches only work on 3D human skeleton representations and cannot be directly applied to video animation in image space. While blending re-assembled motions requires interpolating 3D joint positions in character animation, in our case blending requires synthesizing whole image frames to create a coherent video.

[108, 2] propose motion graph in pixel space and solve this issue by de-ghosting [112] and gradient-domain compositing [131] based on pixel warping. However, these approaches focus on simple periodical scene scenarios, e.g. pendulum, waterfalls, etc. and cannot work on complex human motions. [50, 143, 81, 149] generates controllable human action videos by retrieving and warping nearest candidate frames. However, they require additional motion capture resources such as physical markers, multi-view or RGB-D cameras. [23, 22, 66] also introduce human video synthesis based on reconstruction of human meshes from pre-captured multi-view camera datasets. However, these methods are not suitable for monocular camera videos.

Human Gesture Video Synthesis. [53, 82] translate predicted skeletal gesture motions to photo-realistic speaker videos by utilizing recent neural image translation approaches similar to facial animations [68, 135, 134, 73]. However, neural image translation is not artifact-free. In particular, in human gesture video generation, disconnected moving object parts, as well as incoherent texture appearance are known issues [134]. For example, first, skeleton-based image translation methods can synthesize plausible results for most areas of human bodies, but they cannot perform well on details such as fingers. Second, since human gestures have a lot of freedom, the outlier gestures which occur more frequently, can easily lead to unpleasant artifacts, e.g. broken faces, arms and/or fingers. Due to their large number of parameters in their generator and discriminator, these methods also require large datasets. It is possible for celebrities with tons of video recordings, while it is impossible for other normal people to create their own speech videos. Few-shot solutions [148, 133] do not have such large dataset requirements, yet they suffer from various artifacts, in particular for human pose synthesis, such as blurred appearance and distorted body parts [133]. [138, 85, 84, 113, 139] fit human body model or/and texture parameters to a training video to improve the appearance of body shapes and texture at test time. Yet, inaccurate fitting easily results in artifacts and lose of body or cloth subtleties, especially in the presence of loose clothing and detailed body part deformations, e.g. fingers. In the third part of my thesis, the proposed method follows a largely different approach from all the above prior works: instead of per-frame neural translation, the video of a speaker is generated by re-assembling clips from a short, few minute long reference video. Because most of the frames originate from the reference video, the synthesized video preserves gesture realism as well as body and cloth subtleties. As a result, the problem is simplified to blending video frames. Our neural blending network focuses on solving this particular task, instead of learning to generate all frames from scratch.

Video Frame Blending. The choice of the frame blending strategy significantly impacts the quality of the video generated from re-assembling clips. Naive weighted averaging of video frames easily result in ghost effect [108, 95]. More advanced frame interpolation methods [86, 70, 96, 58] based on optical flow estimation [67, 9, 122] have been proposed to synthesize intermediate frames between two consecutive frames, in particular for slow motion videos. However, such methods fail if two frames are very different from each other and the optical flow estimation is not accurate enough. They work for generic content, yet do not consider human motion as a prior for our task. The third proposed method in this thesis uses a human pose-aware neural network for frame blending that produces significantly better quality video compared to prior such work, as demonstrated in our experiments.

CHAPTER 3

VISEMENET: AUDIO-DRIVEN ANIMATOR-CENTRIC SPEECH ANIMATION

The first part of this thesis discusses VisemeNet, a novel deep-learning based approach to producing animator-centric speech motion curves that drive a JALI or standard FACS-based production face-rig, directly from input audio [153]. [1] Our three-stage Long Short-Term Memory (LSTM) network architecture is motivated by psycho-linguistic insights: segmenting speech audio into a stream of phonetic-groups is sufficient for viseme construction; speech styles like mumbling or shouting are strongly co-related to the motion of facial landmarks; and animator style is encoded in viseme motion curve profiles. Our contribution is an automatic real-time lip-synchronization from audio solution that integrates seamlessly into existing animation pipelines. We evaluate our results by: cross-validation to ground-truth data; animator critique and edits; visual comparison to recent deep-learning lip-synchronization solutions; and showing our approach to be resilient to diversity in speaker and language.

3.1 Algorithm Design

Our approach is designed to achieve high-quality, animator-editable, style-aware, language agnostic, real-time speech animation from audio.

Following the current animation practice of FACS-like face rigs, and state-of-the-art animator-centric speech [39], our network uses an input audio signal to predict a

[1]This work is published at the ACM Transactions on Graphics, Vol. 37, No. 4, 2018, and was also presented in the Proceedings of ACM SIGGRAPH 2018.

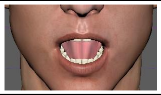
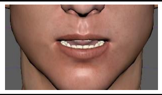
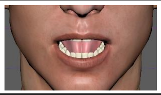
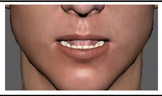
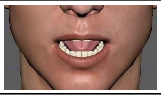
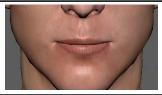
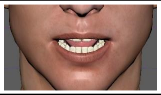
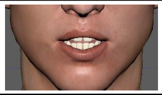
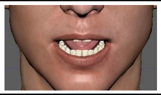
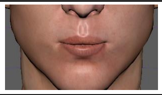
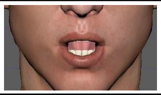
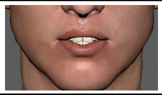
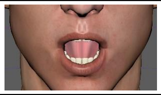
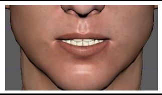
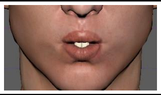
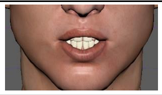
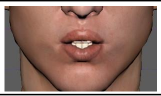
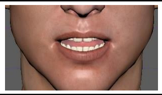
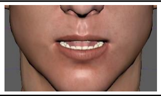
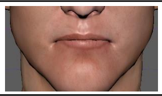
Viseme	Phoneme	Output	Viseme	Phoneme	Output
Ah	ɑ, ɔ, a		LNTD	l, n, t, d, ʆ, ʟ, ɾ	
Aa	æ		GK	g, k, ŋ, q, ɢ	
Eh	e, ɛ		MBP	b, m, p	
Ee	i		R	ɹ	
Ih	ɪ		WA_PEDAL	w, ʋ, ʍ	
Oh	o, ɒ		JY	j, dʒ, c, ɟ	
Uh	ʊ, ʌ, ɞ, ɐ, ɶ, ʊ̈, or, ɨ		S	s, z, ɣ	
U	u		ShChZh	ʃ, tʃ, ʒ, ɬ, ɮ,	
Eu	œ, y, ɯ, ø, ɵ		Th	θ, ð	
Schwa	ə, ɘ		FV	f, v, ɱ	

Figure 3.1: List of visemes along with groups of phonemes (in International Phonetic Alphabet format) and corresponding lower face rig outputs that our architecture produces.

sparse and compact set of viseme values (see Figure 3.1), and jaw and lip parameters, over time. Our neural network architecture (see Figure 3.2) is designed to exploit psycho-linguistic insights and make the most effective use of our training data.

Phoneme Group Prediction A large part of our network, the "phoneme group stage" in Figure 3.2 (top left box), is dedicated to map audio to phonemes groups corresponding to visemes. For example, the two labio-dental phonemes /*f* and *v*/ form a group that maps to a single, near-identical viseme [39], where the lower lip

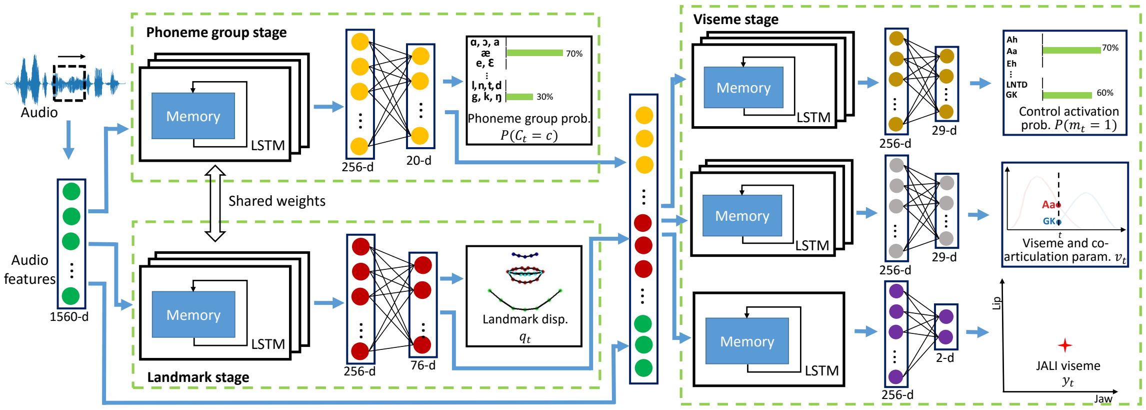


Figure 3.2: Our architecture processes an audio signal (left) to predict JALI-based viseme representations: viseme and co-articulation control activations (top right), viseme and co-articulation rig parameters (middle right), and 2D JALI viseme field parameters (bottom right). Viseme prediction is performed in three LSTM-based stages: the input audio is first processed through the phoneme group stage (top left) and landmark stage (bottom left), then the predicted phoneme group and landmark representations along with audio features are processed through the viseme stage.

is pressed against the upper teeth in Figure 3.1 (last row, right). We identified 20 such visual groups of phonemes expressed in the International Phonetic Alphabet (IPA) in Figure 3.1. Our network is trained to recognize these phoneme groups from audio without any text or phonetic transcript. By predicting only phonetic groups relevant to animation, our task is simpler and less sensitive to linguistic context. The network can also be trained with less data than most speech processing pipelines. As demonstrated in the evaluation (Sec. 3.4), using off-the-shelf audio-to-text techniques to extract individual phonemes that subsequently predict visemes, leads to lower performance than our end-to-end architecture.

Speech Style Prediction Phoneme groups alone have no information of vocal delivery and cannot predict visemes, specially for expressive emotional speech. The same phoneme might be pronounced conversationally, or screamed. These style attributes of the audio performance can be captured using jaw and lip parameters [39]. These parameters are also strongly correlated to the (2D frontal) jaw and lip landmark positions in a visual capture of the vocal performance. The "landmark stage" of our network in Figure 3.2 (bottom-left box) is designed to predict a set of jaw and lip landmark positions over time given input audio.

Viseme Prediction The last part of our network, the "viseme stage" in Figure 3.2(right-box), combines the intermediate predictions of phoneme groups, jaw and lip parameters, as well as the audio signal itself to produce visemes. By training our architecture on a combination of data sources containing audio, 2D video, and 3D animation of human speech, we are able to predict visemes accurately. We represent visemes based on the JALI model [39], comprising a set of intensity values for 20 visemes and 9 co-articulation rules, and **JA**W and **LI**P parameters that capture speaking styles. The viseme and co-articulation values over time can animate standard FACS-based production rigs, with the JA-LI parameters for a rig adding control over speech style. Notably, these are precisely the controls professional animators keyframe, and are thus directly amenable to editing and refinement.

Transfer Learning from Audiovisual Datasets. We need reliable sources of diverse training data with compelling variation in terms of different speakers, speech styles, and emotional content, to train a network that generalizes well. An end-to-end source of training data would be audio clips with corresponding streams of 3D facial rig parameters. Such a large coherent dataset with enough variability is not easily available, and would be too expensive to create with professional animators. On the other hand, there is a wealth of publicly available audiovisual corpora (2D audio+video+text transcript clips), such as BIWI [48], SAVEE [60], and GRID [33]. Although these corpora do not contain any facial rig parameters, they are nonetheless valuable, in the context of our network architecture:

- The faces in the video clips can be accurately detected and annotated with landmarks through modern computer vision. The extracted facial landmarks corresponding to the speech audio are then useful to train the landmark stage of our network.

- The text transcripts can be automatically aligned with the audio clip [91], to provide training phonemes for the phoneme group stage of our network.

To make use of the large amounts of data in these audiovisual datasets, we employ a *transfer learning* procedure to train our network. We first "pre-train" the phoneme group and landmark stages of our network based on training phoneme groups and landmarks extracted from the above audiovisual datasets. We then initialize these two stages according to their pre-trained parameters, and then jointly train the whole network end-to-end. To perform this joint training, we still need a dataset of audio clips with associated streams of facial rig parameters. Mapping phoneme groups and facial landmarks to visemes however, is significantly easier than mapping general speech audio to phonemes and landmarks. Further, the phoneme groups are strongly correlated to visemes, while the landmarks are strongly correlated to jaw and lip parameters. Thus, to train the viseme stage, a much smaller dataset of example 3D animations with face rig parameters is required for sufficient generalization. We empirically observed that using our transfer learning procedure results in a better generalization performance than simply training the entire network on a small dataset of audio clips with rig parameters. We also found that adapting a Multi-Task Learning (MTL) procedure to train the network simultaneously according to multiple objectives involving phoneme group, landmark, viseme and other rig parameter prediction was also important to achieve high performance.

Memory-enabled Networks We adapt a memory-enabled neural network architecture based on Long Short-Term Memory units (LSTMs) [98, 63] for all stages of our network. We believe that memory-based networks are important to correctly capture co-articulation and speech context from input audio, which even for a human listener, is challenging from isolated audio fragments. Memory-enabled networks explicitly store and represent a large amount of context in the input signal that is useful to reliably infer the spoken context. Finally, another advantage of our architecture is that it can predict viseme motion curves in near real-time (120ms or 12 frame lag), given the input audio on modern GPUs. In the following sections, we discuss the network architecture and training procedure in more detail.

3.2 Network Architecture

Our network has an end-to-end architecture that takes an audio signal as input and outputs viseme representations based on JALI. As discussed in the previous section and shown in Figure 3.2, our network has a three stage-architecture: the input audio is first processed through the phoneme group and landmark stages, then the predicted phoneme group and landmark representations along with audio features are processed through the viseme prediction stage. All branches are based on a combination of memory-based LSTM units, which encode context in the input signal, and fully connected layers, which decode the memory of the units into time-varying predictions. Below we discuss our input audio representation and the three stages of our network in more detail.

Input Audio Representation. Given an audio signal as input, we extract a feature vector for each frame encoding various power spectrum and audio frequency signal characteristics. Our feature vector concatenates 13 Mel Frequency Cepstral Coefficients (MFCCs) [36] that have been widely used for speech recognition, 26 raw Mel Filter Bank (MFB) features that have been shown to be particularly useful for emotion discrimination in audio [19], and finally 26 Spectral Subband Centroid features that often result in better speech recognition accuracy when they are used in conjuction with MFCCs [99]. The resulting 65-dimensional feature vector is passed as input to all three stages of our network for further processing. The features are extracted every 10 ms, or in other words feature extraction is performed at a 100 FPS rate. The frequency analysis is performed within windows of size 25 ms in the input audio.

3.2.1 Phoneme Group Stage

The phoneme group stage takes as input the audio features concatenated from 12 frames before the current frame, and also the audio features of the current frame plus 11 frames after the current one. This means that given real-time audio inputs, the network will infer visemes with a lag of 120 ms, plus the required time to extract audio features and perform viseme inference given the audio features per frame (feature extraction and network inference take 1 ms per frame measured on a TitanX GPU, allowing real-time inference with the abovementioned lag). The concatenation produces a 1560-dimensional feature vector $\mathbf{x}_t$ per frame t (65 features x 24 frames) covering audio signal information in a window of $240ms$. The rationale behind using this window is that it approximately matches the average duration of a phoneme in normal speech. We also found that that such window size represented a good trade-off between fast processing time and high phoneme group prediction accuracy.

The feature vector $\mathbf{x}_t$ passes through three layers of unidirectional LSTM units that hierarchically update their internal memory state (encoded with a 256-dimensional vector in our implementation) based on the input information in the audio features. We found that at least three layers (i.e., a deep network) were necessary to achieve sufficient generalization. The LSTM units can choose to either store in their memory cells representations of the incoming features, or alternatively erase representations from their memory cells. The choices of erasing or storing, as well as the transformations of the input features are controlled through non-linear functions (sigmoid and hyperbolic tangent functions) with learnable parameters (for their exact form, we refer to the popular tutorial [98] and [63]).

At each frame, the memory state of the last LSTM layer is decoded towards probabilistic predictions of phoneme groups. The decoding is performed through two non-linear transformations. The first transformation involves a fully connected layer that takes as input the representation of the uppermost LSTM layer, applies a linear transformation on it to produce a 256-dimensional output, which is further processed through the commonly used REctified Linear Unit (RELU) non-linearity. The second layer takes the resulting vector, applies another linear transformation on it producing a 20-dimensional vector $\mathbf{z}_t$, which is then passed through a softmax function to output a per-frame probability for each phoneme group listed in Figure 3.1.

Overall, this network stage can be seen as a non-linear function f that considers audio features up to the current frame $\mathbf{x}_{1:t}$ (by exploiting the LSTM recurrent connections) to output phoneme group probabilities: $P(C_t = c) = f(\mathbf{x}_{1:t}, \boldsymbol{\theta}, \boldsymbol{\phi})$ where C_t is a discrete random variable whose possible values c are our phoneme groups, $\boldsymbol{\theta}$ are the LSTM parameters, and $\boldsymbol{\phi}$ are the decoder parameters. We note that we also experimented with a time-delayed LSTM, as proposed in [56], yet we did not perceive any noticeable differences in the output predictions. We suspect this is because we consider audio features from a large window containing both past and future frames.

3.2.2 Landmark Stage

This stage of our network takes as input the 1560-dimensional feature vector $\mathbf{x}_t$ per frame (same input as in the phoneme group part), passes it through three LSTM layers and decodes the uppermost layer memory into a sparse set of 38 2D facial landmarks, representing the jaw, lip, and nose configuration per frame. Ground-truth and predicted landmarks are visualized in Figure 3.3. The landmarks do not aim at capturing the morphology of a particular face, but instead approximately capture the shape of the lips, positions of jaw and nose of an average face. We found that predicting these visual cues are particularly useful to infer correct visemes that

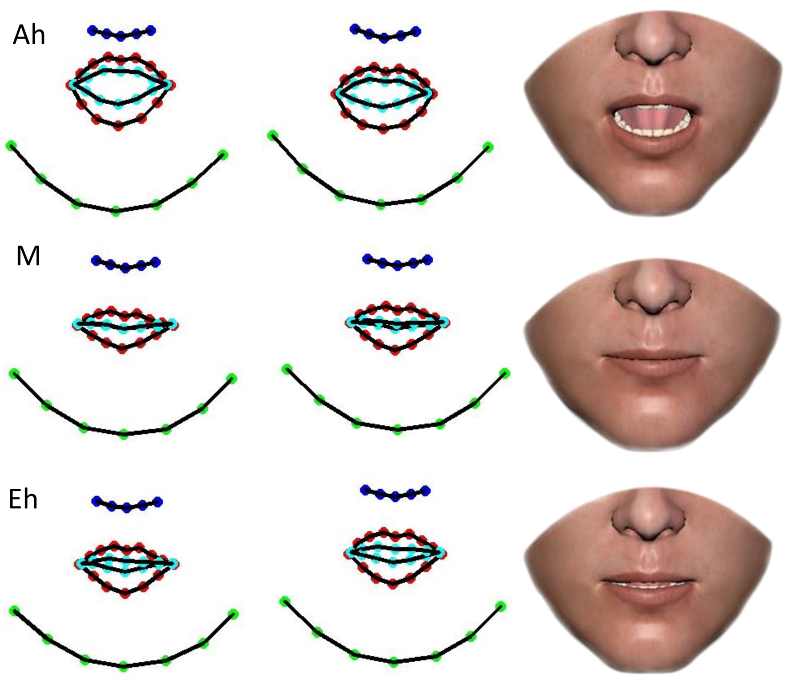


Figure 3.3: Landmark output predictions for a few test frames. The spoken phoneme is shown on the top left. The first column shows ground-truth landmarks. The second column shows landmark predictions from the landmark stage of our network for the same frames. The third column shows the corresponding predicted rig outputs.

reflect speech style (e.g., mumbling, screaming). In particular, the advantage of using these visual cues is that we can exploit audio and landmarks extracted from video available in large, public audiovisual datasets as additional supervisory signal to train the LSTM layers, as described in the next section.

Since phonetic groups and lower face shape are correlated, the three LSTM layers are shared between the phoneme group and landmark stages. Sharing representations is a common strategy in multi-task learning [21], which helps generalization when tasks are correlated. The decoder of the landmark stage is specific to landmark predictions and has its own learned parameters. It is composed of two transformations, implemented as a fully connected layer followed by RELUs, and a second fully connected layer that outputs landmark displacements per frame. The displacements are expressed relative to landmarks representing an average lower face in neutral expression. The displacements are stored in a 76-dimensional vector $\mathbf{q}_t$ per frame t, which simply concatenates displacement coordinates of all the landmarks. Given the neutral face landmarks $\mathbf{b}$, the animated landmark positions can be computed as $\mathbf{b} + \mathbf{q}_t$ per frame. Overall, this part of our network can be seen as another non-linear function h that considers audio features up to the current frame $\mathbf{x}_{1:t}$ and outputs landmark displacements per frame: $\mathbf{q}_t = h(\mathbf{x}_{1:t}, \boldsymbol{\theta}, \boldsymbol{\omega})$ where $\boldsymbol{\theta}$ are the shared LSTM parameters, and $\boldsymbol{\omega}$ are the decoder parameters of this stage.

3.2.3 Viseme Stage

The viseme stage takes as input the produced phoneme group representations $\mathbf{z}_t$ (i.e., phoneme group activations before applying softmax), landmark displacements $\mathbf{q}_t$, and also the audio features $\mathbf{x}_t$ and outputs JALI-based rig parameters and controls that determine the visemes per frame. Here, we use the audio features as additional input since phoneme groups and landmarks might not entirely capture all speech style-related information existing in audio (for example, fast breathing due to a nervous style of speech will not be captured in landmarks or phonemes, yet will manifest in audio features).

The viseme stage produces the following outputs per frame t (see also Figure 3.2, right):

- 29 continuously-valued viseme animation and co-articulation parameters present in JALI (we refer to [39] for more details). The rig parameters are represented by a 29-dimensional continuous vector $\mathbf{v}_t$.

- 29 binary random variables, represented as a vector $\mathbf{m}_t$, that indicate whether each of the above viseme and co-articulation control parameters is active per frame. The underlying reason for using these binary variables is that the activations of viseme and co-articulation rig parameters are largely sparse (Figure 1.1a), since at a given frame only one viseme is dominantly active. If we train the network to match the viseme and co-articulation parameters in the training data, without considering these binary indicator variables, then the predicted values of these action units are often biased towards low or zero values since most of the time their corresponding training values are zero.

- an output 2-dimensional vector $\mathbf{y}_t$ representing the 2D JALI viseme field values capturing speech style per frame t.

Given the inputs $\{\mathbf{z}_t, \mathbf{q}_t, \mathbf{x}_t\}$ concatenated as a single vector, the viseme stage uses a three-layer LSTM-based architecture to produce the above outputs. Here, we found that using separate LSTM layers (i.e., without shared parameters) for each output type offers the best performance, probably due to the fact that the JALI viseme field is designed to be independently controllable from the rest of rig parameters [39], and also because the continuous rig parameter values vary widely given their activation state. Each LSTM layer uses units with a 256-dimensional memory state. The rig parameters $\mathbf{v}_t$ and JALI viseme field values $\mathbf{y}_t$ are computed by decoding their corresponding uppermost LSTM layer memory through two dedicated fully connected layers with a RELU non-linearity in-between. The binary indicator variables are predicted by decoding their corresponding uppermost LSTM layer memory, followed by two dedicated fully connected layers with a RELU non-linearity in-between, and finally a sigmoid function that produces the probability of each rig parameter to be active or not. At test time, we first predict these activation probabilities. Then we produce the continuous values of the corresponding viseme and co-articulation rig parameters whose activation probability is above a threshold, which we automatically set during training.

Overall, this part of our network can be seen as a set of three non-linear functions g_1, g_2, g_3 that considers phoneme group predictions $\mathbf{z}_{1:t}$, landmark predictions $\mathbf{q}_{1:t}$, and audio features $\mathbf{x}_{1:t}$ up to the current frame and output probabilities of rig parameter activations $P(\mathbf{m}_t) = g_1(\mathbf{z}_{1:t}, \mathbf{q}_{1:t}, \mathbf{x}_{1:t}, \boldsymbol{\xi}_1)$, rig parameter values $\mathbf{v}_t =$

$g_2(\mathbf{z}_{1:t}, \mathbf{q}_{1:t}, \mathbf{x}_{1:t}, \boldsymbol{\xi}_2)$, and viseme field values $\mathbf{y}_t = g_3(\mathbf{z}_{1:t}, \mathbf{q}_{1:t}, \mathbf{x}_{1:t}, \boldsymbol{\xi}_3)$, where $\boldsymbol{\xi} = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3\}$ are learned parameters of each corresponding set of LSTM layer and decoder. In the following section, we describe how all the parameters of our network are learned.

3.3 Training

We follow a two-step, transfer learning procedure to train our network. Motivated by the observation that there are large valuable amounts of audiovisual data with text transcripts available in public repositories, we first "pre-train" the phoneme group and landmark stages of our network based on 15 hours of recorded audio along with ground-truth phoneme groups and tracked facial landmarks from video as supervisory signal. After this pre-training step, we fine-tune the whole network jointly, end-to-end, to accurately predict visemes based on a smaller, painstakingly created dataset consisting of one hour of audio with exemplar streams of JALI rig parameter values. Below we explain our datasets and pre-processing, then we discuss pre-training and joint training procedures. Training and test splits are discussed in the results section (Sec. 3.4).

Audiovisual Dataset. This dataset contains audiovisual data from three repositories. First, we use the GRID dataset [33], which contains transcripts, audio and video recordings of 1000 sentences spoken by 34 speakers (18 male, 16 females) in a neutral style of speech with total time duration of about 14 hours. The sentences are deliberately chosen to cover common phonemes in English. Second, we use the SAVEE dataset [136], which contains transcripts, audio and video recordings of 480 sentences spoken by 4 male speakers expressing different emotions, including anger, disgust, fear, sadness and surprise. The total clip duration is 30 minutes. Third, we used the BIWI 3D audiovisual corpus dataset [48], which contains transcripts, audio, video and RGBD recordings of 1109 sentences spoken by 14 speakers (6 males and 8 females) in various emotional and neutral styles of speech with total time duration of about 1 hour. We pre-processed the videos of these datasets to extract facial landmarks involving lips, jaw and nose (Figure 3.3, left column) through DLib [75] and FaceWare Analyzer [46]. The landmarks were aligned with our average face template in neutral poses and normalized to be invariant to face location, orientation and size. Then we extracted training landmark displacements for each frame relative to the average face. Given the provided text transcripts in these datasets, we used the Montreal Forced Aligner (MFA) [91] to align audio with text and extract phonemes along with corresponding phoneme groups.

JALI-annotated Dataset An experienced animator created rigs according to the JALI template for the BIWI dataset (total $1h$ of rig motion curves with corresponding audio involving the 14 BIWI speakers).

3.3.1 Pre-training

Given N audio clips with sequences of landmark displacements $\hat{\mathbf{q}}_{1:t_n}$ (where t_n is number of frames of the n^{th} audio clip, $n = 1...N$), and corresponding phoneme groups $\hat{c}_{1:t_n}$ extracted per frame from the training audiovisual datasets, the goal of the pre-training step is to estimate the decoder parameters $\boldsymbol{\phi}$ of the phoneme group stage, the decoder parameters $\boldsymbol{\omega}$ of the landmark stage, and the parameters $\boldsymbol{\theta}$ of their shared LSTM layers. The parameters are learned such that the predicted phoneme groups match the training ones as much as possible, the predicted landmark coordinates are as close as possible to the training ones, and also the predicted landmarks do not change over time abruptly. The goals can be expressed with a combination of a classification loss $L_c(\boldsymbol{\theta}, \boldsymbol{\phi})$ for phoneme groups, a regression loss $L_q(\boldsymbol{\theta}, \boldsymbol{\omega})$ for landmarks, and another loss $L'_q(\boldsymbol{\theta}, \boldsymbol{\omega})$ that promotes smoothness in the predicted landmark movement. This combination is expressed as the following multi-task loss:

$$L_1(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}) = w_c \ L_c(\boldsymbol{\theta}, \boldsymbol{\phi}) + w_q L_q(\boldsymbol{\theta}, \boldsymbol{\omega}) + w'_q L'_q(\boldsymbol{\theta}, \boldsymbol{\omega}) \tag{3.1}$$

where weights of the three losses are set as $w_c = 0.75$, $w_q = 0.25$, $w'_q = 0.1$ in all our experiments. The classification loss $L_c(\boldsymbol{\theta}, \boldsymbol{\phi})$ favors parameters that maximize the probability of the training phoneme groups, or equivalently minimize their negative log-probability. It can be expressed as the popular cross-entropy multi-class loss:

$$L_c(\boldsymbol{\theta}, \boldsymbol{\phi}) = -\frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{t_n}\sum_{t=1}^{t_n}\log P(C_t = \hat{c}_t)\right) \tag{3.2}$$

The regression loss $L_q(\boldsymbol{\theta}, \boldsymbol{\omega})$ is expressed as the absolute differences (i.e., L_1-norm loss) between the training and predicted landmark coordinates (their total number is $M = 76$ coordinates from 38 2D landmarks):

$$L_q(\boldsymbol{\theta}, \boldsymbol{\omega}) = \frac{1}{N}\frac{1}{M}\sum_{n=1}^{N}\left(\frac{1}{t_n}\sum_{t=1}^{t_n}||\mathbf{q}_t - \hat{\mathbf{q}}_t||_1\right) \tag{3.3}$$

The smoothness loss $L'_q(\boldsymbol{\theta}, \boldsymbol{\omega})$ penalizes large absolute values of landmark motion derivatives with respect to time:

$$L'_q(\boldsymbol{\theta}, \boldsymbol{\omega}) = \frac{1}{N}\frac{1}{M}\sum_{n=1}^{N}\left(\frac{1}{t_n}\sum_{t=1}^{t_n}||\dot{\mathbf{q}}_t||_1\right) \tag{3.4}$$

where $\dot{\mathbf{q}}_t$ represents the derivative of the predicted landmark displacements over time. The derivative is computed through central finite differences in our implementation.

Minimizing the above multi-task loss function is done through batch gradient descent with batch size 256, learning rate 0.00001, momentum set to 0.9, over $2M$ iterations.

3.3.2 Joint Training

For joint training, we initialize the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}$ of the phoneme group and landmark stages to their pretrained values, which are already expected to be close to a desired local minimum. Then we estimate all the parameters of the whole network jointly, including the parameters $\boldsymbol{\xi} = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3\}$ of the viseme prediction branch, such that based on the JALI-annotated dataset, we satisfy the above goals: (a) the predicted viseme and co-articulation parameter activations match the ground-truth ones through a binary classifications loss $L_a(\boldsymbol{\xi}_1)$, (b) the predicted viseme and co-articulation parameters are as close as possible to the ground-truth ones when these units are active through a regression loss $L_v(\boldsymbol{\xi}_2)$ modified to consider these activations, (c) the predicted 2D JALI viseme field values are also as close as possible to the ground-truth ones through a regression loss $L_j(\boldsymbol{\xi}_3)$, (d) the rig parameters and JALI field values do not change abruptly over time through two smoothness losses $L'_v(\boldsymbol{\xi}_2)$ and $L'_j(\boldsymbol{\xi}_3)$, and finally (e) the predicted phoneme groups and landmarks remain close to the ground-truth ones (as done in the pre-training step). This goal can be expressed again as a multi-task loss:

$$\begin{aligned} L_2(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\xi}) = L_1(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}) + w_a L_a(\boldsymbol{\xi}_1) + w_v L_v(\boldsymbol{\xi}_2) \\ + w_j L_j(\boldsymbol{\xi}_3) + w'_v L'_v(\boldsymbol{\xi}_2) + w'_j L'_j(\boldsymbol{\xi}_3) \end{aligned} \tag{3.5}$$

where $L_1(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega})$ is the loss of Eq. 3.1 (same as pre-training, but now evaluated in the JALI-annotated dataset). The loss weights are set in all our experiments as follows: $w_a = 0.1, w_v = 0.2, w_j = 0.2, w'_v = 0.15$, and $w'_j = 0.15$. Note that this loss function is not decomposable because the predictions (and in turn, the losses) associated with the viseme branch depend on the predicted phonemes and landmarks of the other two stages during training. Below we describe the individual loss functions in detail.

The loss function $L_a(\boldsymbol{\xi}_1)$ penalizes disagreements between predicted parameter activations $\mathbf{m}_t$ and ground-truth parameter activations $\hat{\mathbf{m}}_t$ for each training frame t. Since multiple rig parameters can be active at a given time, this loss function attempts to maximize the probability of correct, individual activations per parameter (or equivalently minimize their negative log-probability). It can be expressed as a sum of $A = 29$ binary cross-entropy losses, one per rig parameter:

$$L_a(\boldsymbol{\xi}_1) = -\frac{1}{N}\frac{1}{A}\sum_{n=1}^{N}\sum_{a=1}^{N}\left(\frac{1}{t_n}\sum_{t=1}^{t_n}[\hat{m}_{a,t}=1]\log P(m_{a,t}=1)\right.$$
$$\left.\frac{1}{t_n}\sum_{t=1}^{t_n}[\hat{m}_{a,t}=0]\log P(m_{a,t}=0)\right) \tag{3.6}$$

where $[\hat{m}_{a,t}=1], [\hat{m}_{a,t}=0]$ are binary functions indicating whether the rig parameter a is active or not at frame t.

The loss function $L_v(\boldsymbol{\xi}_2)$ measures absolute differences between the training values $\hat{v}_{a,t}$ and predicted values $v_{a,t}$ of each viseme and co-articulation rig parameter a when these are active according to the ground-truth binary activity indicator functions:

$$L_v(\boldsymbol{\xi}_2) = \frac{1}{N}\frac{1}{A}\sum_{n=1}^{N}\sum_{a=1}^{A}\left(\frac{1}{t_{n,a}}\sum_{t=1}^{t_n}[\hat{m}_{a,t}=1]\cdot|v_{a,t}-\hat{v}_{a,t}|\right) \tag{3.7}$$

where $t_{n,a}$ is the number of frames where the rig parameter a is active per clip n in the ground-truth (i.e., $t_{n,a} = \sum_t[\hat{m}_{a,t}=1]$) An alternative approach would be to evaluate rig parameter differences when these are inactive (i.e., pushing the predictions towards 0 in these cases). However, we found that this degrades the prediction quality of the rig parameters because the network over-focuses on making correct predictions in periods where visemes are inactive. The smoothness loss $L'_v(\boldsymbol{\xi}_2)$ penalizes large derivatives of predicted viseme and co-articulation parameters over time:

$$L_q(\boldsymbol{\xi}_2) = \frac{1}{N}\frac{1}{A}\sum_{n=1}^{N}\sum_{a=1}^{A}\left(\frac{1}{t_{n,a}}\sum_{t=1}^{t_n}[\hat{m}_{a,t}=1]\cdot|\dot{v}_{a,t}|\right) \tag{3.8}$$

Finally, the loss function $L_j(\boldsymbol{\xi}_3)$ measures absolute differences between the training values $\hat{\mathbf{y}}_t$ and predicted values $\mathbf{y}_t$ of the 2D JALI viseme field, while $L'_j(\boldsymbol{\xi}_3)$ penalizes large changes in the viseme field values over time:

$$L_j(\boldsymbol{\xi}_3) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{t_n}\sum_{t=1}^{t_n}||\mathbf{y}_t - \hat{\mathbf{y}}_t||_1\right) \tag{3.9}$$

$$L'_j(\boldsymbol{\xi}_3) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{t_n}\sum_{t=1}^{t_n}||\dot{\mathbf{y}}_t||_1\right) \tag{3.10}$$

Minimizing the above multi-task loss function is done through batch gradient descent with with batch size 256, learning rate 0.00001, momentum set to 0.9, over 200K iterations.

Thresholding Activation Probabilities. In the training stage, we also compute a threshold thr_a for each rig parameter a that determines when to activate it based on the rig control activation probability produced by our network, i.e., check $P(m_{a,t} > thr_a)$. One potential choice could be to simply set $thr_a = 0.5$. Yet, we found that optimizing the threshold per rig parameter through a dense grid search in a small hold-out validation dataset and selecting the value yielding the best precision and recall in rig activations in that dataset offered better performance.

Implementation and Running Times. Our network is implemented in Tensorflow. Pre-training takes $30h$ and joint training takes $15h$ in our training datasets measured on a TitanX GPU. At test time, audio feature extraction and network inference (forward propagation) is performed at 1000 FPS (1ms per frame) with a lag of 120 ms relative to the current frame (see Sec. 3.2, phoneme branch paragraph). Our code for training and testing the network, trained models, datasets, and results will become publicly available upon acceptance.

3.4 Evaluation

We evaluated our method and alternatives both quantitatively and qualitatively. In this section, we primarily focus on quantitative evaluation. We refer the reader to the video for qualitative results and comparisons.

Methodology. We focused on the BIWI 3D audiovisual dataset to validate our method and alternatives. As mentioned in the previous section, exemplar JALI-based motion curves were provided by an artist for the whole dataset, thus we can compare predicted rig parameters to ground-truth. In addition, each of the 14 speakers of the BIWI dataset speaks the same sentence in both neutral and expressive styles, conveying emotions such as anger, sadness, fear, nervousness, and excitement. Thus, we can compare our method and alternatives on how well they handle different styles of speech.

Because this JALI-annotated dataset has only 14 speakers, we perform the evaluation through a leave-one-out approach: for each of the 14 BIWI speakers, we perform pre-training on our audiovisual dataset (including GRID, SAVEE, and BIWI but excluding that BIWI speaker), and then perform joint training on the JALI-annotated BIWI dataset using the other 13 speakers. As a result, we form 14 training and test splits, where the test splits always involve a speaker not observed during training. Since we aim at learning a speaker-independent, generic model, we believe that this is a more compelling and practically useful generalization scenario, compared to training and testing on the same speaker.

Quantitative Evaluation Measures. The first evaluation measure we use is the rig parameter *activation precision*, which measures how often we activate the right viseme and co-articulation rig parameters based on the ground-truth. Specifically, given a binary variable $\hat{m}_{a,t}$ indicating whether the rig parameter a is activated or not at frame t in the ground-truth, and given our predicted binary variable $m_{a,t}$ for that parameter and frame, the precision is calculated as the number of times we correctly predict activations for the rig parameters (i.e., $\sum_{a,t}[\hat{m}_{a,t} = 1 \,\&\, m_{a,t} = 1]$, or in other words, the number of true positives) normalized by the total number of predicted activations (i.e., $\sum_{a,t}[m_{a,t} = 1]$). We also evaluate the rig parameter *activation recall*, which measures out of all the ground-truth rig parameter activations, what fraction

Table 3.1: Precision and recall for activation of rig controls for our full method and degraded versions of it for neutral and expressive speech styles, averaged over all test splits (higher precision and recall are better). We also report the average standard deviation (SD) of precision and recall for each variant.

	neutral			expressive		
	precision (%)	recall (%)	SD	precision (%)	recall (%)	SD
full method	**89.5**	**92.2**	1.9	**90.1**	**92.3**	2.2
landmark-based	73.6	82.0	5.1	74.4	82.7	4.9
phoneme-based	87.8	91.5	1.8	88.2	91.6	2.0
audio-based	68.7	81.3	5.2	69.3	82.2	4.9
no transfer learning	85.8	89.5	2.3	86.1	89.6	2.2
no shared weights (LP)	88.5	91.5	1.0	88.8	91.6	1.9
shared weights (V)	89.0	91.9	1.7	89.3	91.9	1.9
ASR-based	87.6	89.2	1.6	88.4	89.8	1.4
GRU-based	87.4	91.2	2.1	87.8	91.1	2.3
sliding window-based	78.1	77.8	1.5	78.6	78.2	1.5

of them we predict correctly. The recall is calculated as the number of times we correctly predict activations for the rig parameters (again, number of true positives) divided by the total number of ground-truth activations $\sum_{a,t}[\hat{m}_{a,t} = 1]$. In the ideal scenario, precision and recall should be both 100%.

We also measure the *motion curve differences*, which evaluates the absolute differences of our predicted rig parameter values (viseme, co-articulation, and JALI field parameters) compared to their ground-truth values averaged over the test frames where the corresponding rig parameters are active either in the ground-truth or in the predictions. We note that we do not consider inactive parameters in the evaluation of motion curve differences because the motion curves are very sparse; zero-values would dominate the measure otherwise. Since all the rig parameters are normalized between $[0, 1]$ during training and testing, these differences can be treated as percentages.

Quantitative Comparisons. We compare our network with the following alternative architectures: (a) **landmark-based:** we eliminate the phoneme group stage of

Table 3.2: Percentage difference of motion curves (including viseme, co-articulation, and JALI field parameters) for our method and degraded versions of our architecture for neutral and expressive styles of speech, averaged over all test splits (lower difference is better). We also report the standard deviation (SD) of the percentage differences for each variant.

	neutral		expressive	
	motion curve differences (%)	SD	motion curve differences (%)	SD
full method	**7.8**	0.8	**7.6**	0.9
landmark-based	13.6	1.8	13.2	1.5
phoneme-based	9.0	0.7	8.8	0.7
audio-based	14.5	1.8	14.2	1.6
no transfer learning	9.7	0.9	9.6	0.8
no shared weights (LP)	8.8	0.6	8.7	0.7
shared weights (V)	9.5	0.7	9.2	0.7
ASR-based	9.1	0.6	8.8	0.6
GRU-based	9.1	0.7	9.0	0.8
sliding window-based	15.4	0.3	15.2	0.4

our network i.e., visemes are predicted based on landmarks and audio features only, (b) **phoneme-based:** we eliminate the landmark stage of our network i.e., visemes are predicted based on phoneme groups and audio features only, (c) **audio-based:** we eliminate both the landmark and phoneme group stages i.e., visemes are predicted directly from audio features alone, which also implies that there is no pre-training since no training phoneme groups or landmarks can be used in this condition (d) **no transfer learning:** we keep all the stages of our network, yet we train only on the JALI-annotated BIWI training split and not on the rest of the audiovidual datasets, (e) **no shared weights (LP):** we disable weight sharing between the landmark and phoneme stages i.e., the LSTM layers of these two stages have independent parameters in this variant, (f) **shared weights (V):** we force weight sharing between the layers of the three LSTM modules used for predicting rig control activations, viseme/co-articulation parameters, and JALI parameters in the viseme stage (this is in contrast to our proposed architecture that uses LSTMs without shared weights in this stage).

(g) **ASR-based:** instead of using our phoneme group prediction part, we pass the input audio through the popular Google Cloud automatic speech recognition engine [54] to extract text, then extract phonemes based on the MFA forced aligner [91], form the phoneme groups of Figure 3.1, encode them into a binary vector with 1s corresponding to present phoneme groups and 0*s* for the non-present ones per frame, and pass this vector to our viseme branch (instead of our phoneme group representations). This alternative architecture tests the condition where phonemes are acquired automatically in separate stages through existing off-the-shelf tools, (h) **GRU-based:** we use Gated Recurrent Units (GRUs) [29] instead of LSTMs as memory modules, (i) **sliding window-based:** instead of using LSTMs in our three stages, we experimented with the memory-less neural network modules based on three fully connected hidden layers operating on sliding windows, as proposed in [120]. We note that we further benefited this approach by using the same type of inputs in each stage (i.e., the viseme stage receives audio, landmarks, and phonemes as input instead of using phonemes alone as done in [120]) and also using the same pre-training and transfer learning procedure as in our approach (without these enhancements, the performance was worse). We also increased the number of hidden nodes per layer so that the number of learnable parameters is comparable to the one in our architecture (using the original number of hidden nodes also resulted in worse performance).

We trained these alternative architectures based on the same corresponding loss functions in the same training sets as our method, performed the same hyper-parameter tuning procedure as in our method, and evaluated them in the same test splits. Table 3.1 and Table 3.2 report the abovementioned evaluation measures for our method and the alternatives for neutral and expressive styles of speech. Our full method offers the best performance in terms of all evaluation measures and different styles of speech. Based on the results, if one attempts to skip our intermediate phoneme group and landmark stages, and predict visemes from audio features directly ("audio-based" condition), then the performance degrades a lot (see also accompanying video and Figure 3.4 for qualitative comparisons). Skipping the phoneme group stage ("landmark-based" condition) also results in a large performance drop, which indicates that recognizing phoneme groups is crucial for predicting correct visemes, as the psycholinguistic literature indicates. Skipping landmarks ("phoneme-based" condition) also results in a noticeable drop in performance for both neutral and expressive styles of speech. Using off-the-shelf tools for viseme recognition ("ASR-based" condition) also results in worse performance than our approach. Note also that this ASR-based approach is language-dependent and requires an input language-based phoneme model specification, while our approach is language-agnostic since our phoneme groups are based on the International Phonetic Alphabet (IPA). Furthermore, we observed that transfer learning and weight sharing in the landmark and phoneme stages improve performance. Using GRUs results in worse performance compared to LSTMs. Finally, replacing the LSTMs with fully connected network modules operating on sliding windows causes a large drop in performance.

Qualitative Comparisons. Our video shows facial animation results produced by our method and degraded versions of our architecture as well as comparisons with previous works [72], [119], and [120]. Quantitative comparisons with these previous works are not possible because their used test rigs are not FACS-enabled, and their implementation is not publicly available. In contrast to our approach, none of these previous methods produce editable, animator-centric viseme curves or facial action units. We also demonstrate generalization to speech animation involving different languages.

3.5 Conclusion

We presented an animator-centric, deep learning approach that maps audio to speech motion curves. There are various avenues for future work. Our implemen-

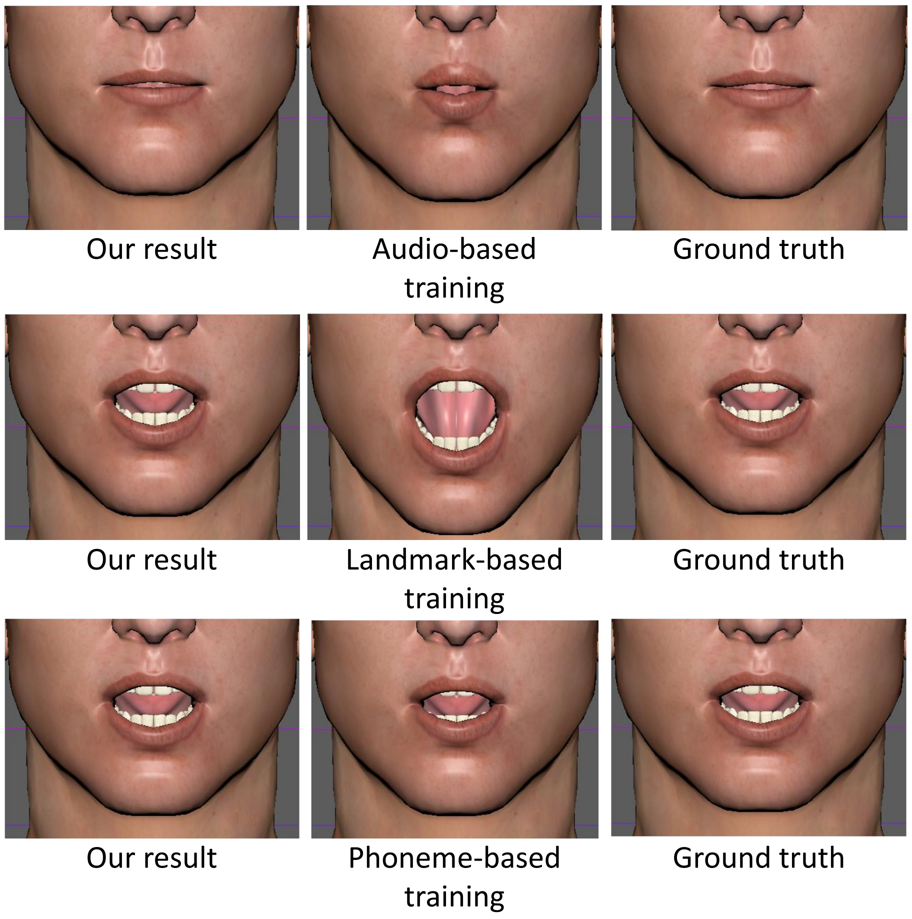


Figure 3.4: Characteristic outputs of alternative architectures versus our method for a frame from our test splits. **(top)** Comparison with using audio features alone ("audio-based" training) **(middle)** Comparison with using landmarks and audio features alone ("landmark-based" training) **(bottom)** Comparison with using phoneme groups and audio features alone ("phoneme-based" training). Ground-truth for the corresponding test frame is shown on the right column.

tation currently uses hand-engineered audio features. Replacing them with learned features, similarly to what is done in image and shape processing pipelines, could help improving performance. Another interesting extension would be to incorporate a discriminator network that would attempt to infer the quality of the generated animations, and use its predictions to boost the performance of our viseme generator network, as done in cGAN-based approaches [68] for image and shape synthesis. Finally, our method is able to drive only the lower part of the face. Learning to control the upper face e.g., eyes, without explicit supervisory signals would also be a fruitful direction.

The marriage between animator-centric techniques and deep-learning has the potential to fundamentally alter current facial animation practice in film and game studios, leaving animators free to focus on the creative and nuanced aspects of character expression. We believe our solution is a significant step in this direction.

CHAPTER 4

MAKEITTALK: SPEAKER-AWARE TALKING-HEAD ANIMATION

The second part of my thesis focuses on "MakeItTalk", a method that generates expressive talking-head videos from a single facial image with audio as the only input [152].[1] In contrast to previous attempts to learn direct mappings from audio to raw pixels for creating talking faces, our method first disentangles the content and speaker information in the input audio signal. The audio content robustly controls the motion of lips and nearby facial regions, while the speaker information determines the specifics of facial expressions and the rest of the talking-head dynamics. Another key component of our method is the prediction of facial landmarks reflecting the speaker-aware dynamics. Based on this intermediate representation, our method works with many portrait images in a single unified framework, including artistic paintings, sketches, 2D cartoon characters, Japanese mangas, and stylized caricatures. In addition, our method generalizes well for faces and characters that were not observed during training. We present extensive quantitative and qualitative evaluation of our method, in addition to user studies, demonstrating generated talking-heads of significantly higher quality compared to prior state-of-the-art methods.

[1]This work is published at the ACM Transactions on Graphics, Vol. 39, No. 6, 2020, and was also presented in the Proceedings of ACM SIGGRAPH ASIA 2020.

4.1 Overview

As summarized in Figure 4.1, given an audio clip and a single facial image, our architecture, called "MakeItTalk", generates a *speaker-aware* talking-head animation synchronized with the audio. In the training phase, we use an off-the-shelf face 3D landmark detector to preprocess the input videos to extract the landmarks [18]. A baseline model to animate the speech content can be trained from the input audio and the extracted landmarks directly. However, to achieve high-fidelity dynamics, we found that landmarks should instead be predicted from a disentangled content representation and speaker embedding of the input audio signal.

Specifically, we use a voice conversion neural network to disentangle the speech content and identity information [102]. The *content* is speaker-agnostic and captures the general motion of lips and nearby regions (Figure 4.1, *Speech Content Animation*, Section. 4.2.1). The *identity* of the speaker determines the specifics of the motions and the rest of the talking-head dynamics (Figure 4.1, *Speaker-Aware Animation*, Section. 4.2.2). For example, no matter who speaks the word 'Ha!', the lips are expected to be open, which is speaker-agnostic and only dictated by the content. As for the exact shape and size of the opening, as well as the motion of nose, eyes and head, these will depend on who speaks the word, i.e., identity. Conditioned on the content and speaker identity information, our deep model outputs a sequence of predicted landmarks for the given audio.

To generate rasterized images, we developed two algorithms for the landmark-to-image synthesis (Section. 4.2.3). For non-photorealistic images like paintings or vector arts (Fig. 4.7), we use a simple image warping method based on Delaunay triangulation (Figure 4.1, *Face Warp*). For photorealistic ones (Fig. 4.6), we devised an image-to-image translation network (similar to pix2pix [68]) to animate the given natural human face image with the underlying landmark predictions (Figure 4.1, *Image2Image Translation*). Combining all the image frames and input audio together

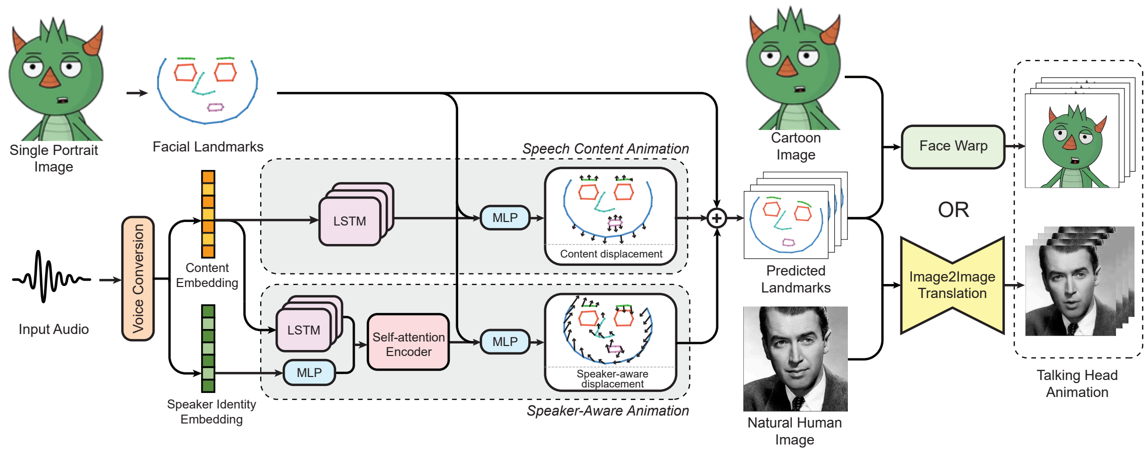


Figure 4.1: Pipeline of our method ("MakeItTalk"). Given an input audio signal along with a single portrait image (cartoon or real photo), our method animates the portrait in a speaker-aware fashion driven by disentangled content and speaker embeddings. The animation is driven by intermediate predictions of 3D landmark displacements. The "speech content animation" module maps the disentangled audio content to landmark displacements synchronizing the lip, jaw, and nearby face regions with the input speech. The same set of landmarks is further modulated by the "speaker-aware animation" branch that takes into account the speaker embedding to capture the rest of the facial expressions and head motion dynamics.

gives us the final talking-head animations. In the following sections, we describe each module of our architecture.

4.2 Method

4.2.1 Speech Content Animation

To extract the speaker-agnostic content representation of the audio, we use AutoVC encoder from [102]. The AutoVC network utilizes an LSTM-based encoder that compresses the input audio into a compact representation (bottleneck) trained to abandon the original speaker identity but preserve content. In our case, we extract a content embedding $\mathbf{A} \in \mathcal{R}^{T \times D}$ from AutoVC network, where T is the total number of input audio frames, and D is the content dimension.

The goal of the content animation component is to map the content embedding $\mathbf{A}$ to facial landmark positions with a neutral style. In our experiments, we found that recurrent networks are much better suited for the task than feedforward networks, since they are designed to capture such sequential dependencies between the audio content and landmarks. We experimented with vanilla RNNs and LSTMs [55], and found that LSTMs offered better performance. Specifically, at each frame t, the LSTM module takes as input the audio content $\mathbf{A}$ within a window $[t \to t+\tau]$. We set $\tau = 18$ frames (a window size of 0.3s in our experiments). To animate any input 3D static landmarks $\mathbf{q}$, where $\mathbf{q} \in \mathcal{R}^{68\times 3}$ that are extracted using a landmark detector, the output from LSTM layers is fed into a Multi-Layer Perceptron (MLP) and finally predicts displacements $\Delta\mathbf{q}_t$, which put the input landmarks in motion at each frame.

To summarize, the speech content animation module models sequential dependencies to output landmarks based on the following transformations:

$$\mathbf{c}_t = LSTM_c\big(\mathbf{A}_{t\to t+\tau}; \mathbf{w}_{lstm,c}\big), \tag{4.1}$$

$$\Delta\mathbf{q}_t = MLP_c(\mathbf{c}_t, \mathbf{q}; \mathbf{w}_{mlp,c}), \tag{4.2}$$

$$\mathbf{p}_t = \mathbf{q} + \Delta\mathbf{q}_t, \tag{4.3}$$

where $\{\mathbf{w}_{lstm,c}, \mathbf{w}_{mlp,c}\}$ are learnable parameters for the LSTM and MLP networks respectively. The LSTM has three layers of units, each having an internal hidden state vector of size 256. The decoder MLP network has three layers with internal hidden state vector size of 512, 256 and 204 (68×3), respectively.

4.2.2 Speaker-Aware Animation

Matching just the lip motion to the audio content is not sufficient. The motion of the head or the subtle correlation between mouth and eyebrows are also crucial clues to generate realistic talking-heads. For example, Figure 4.2 shows our speaker-aware

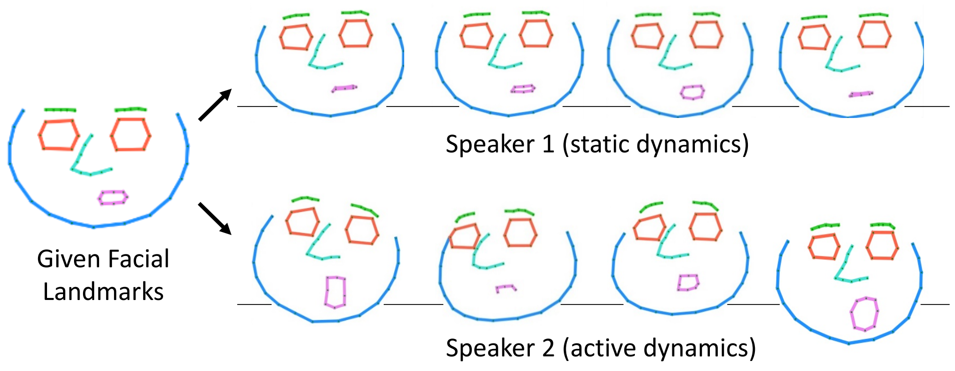


Figure 4.2: Landmark prediction for different speaker identities. Left: static facial landmarks from a given portrait image. Right-top: predicted landmark sequence from a speaker who tends to be conservative in terms of head motion. Right-bottom: predicted landmark sequence from another speaker who tends to be more active.

predictions for two different speaker embeddings: one originates from a speaker whose head motion tends to be more static, and another that is more active. Our method successfully differentiates the head motion dynamics between these two speakers.

To achieve this, we extract the speaker identity embedding with a speaker verification model [130] which maximizes the embedding similarity among different utterances of the same speaker, and minimizes the similarity among different speakers. The original identity embedding vector $\mathbf{s}$ has a size of 256. We found that reducing its dimensionality from 256 to 128 via a single-layer MLP improved the generalization of facial animations especially for speakers not observed during training. Given the identity embedding $\mathbf{s}$ extracted, we further modulate the per-frame landmarks $\mathbf{p}_t$ such that they reflect the speaker's identity. More specifically, the landmarks are perturbed to match head motion distribution and facial expression dynamics observed in speakers during training.

As shown in the bottom stream of Figure 4.1, we first use an LSTM to encode the content representation within time windows, which has the same network architecture and time window length as the LSTM used in the speech content animation module.

We found, however, that it is better to have different learned parameters for this LSTM, such that the resulting representation $\tilde{\mathbf{c}}_t$ is more tailored for capturing head motion and facial expression dynamics:

$$\tilde{\mathbf{c}}_t = LSTM_s\big(\mathbf{A}_{t \to t+\tau}; \mathbf{w}_{lstm,s}\big), \tag{4.4}$$

where $\mathbf{w}_{lstm,s}$ are trainable parameters. Then, the following model takes as input the speaker embedding $\mathbf{s}$, the audio content representation $\tilde{\mathbf{c}}_t$, and the initial static landmarks $\mathbf{q}$ to generate speaker-aware landmark displacement. Notably, we found that producing coherent head motions and facial expressions requires capturing longer time-dependencies compared to the speech content animation module. While phonemes typically last for a few tens of milliseconds, head motions, e.g., a head swinging left-to-right, may last for one or few seconds, several magnitudes longer. To capture such long and structured dependencies, we adopted a self-attention network [126, 37]. The self-attention layers compute an output expressed as a weighted combination of learned per-frame representations i.e., the audio content representation $\tilde{\mathbf{c}}_t$ extracted by the above LSTM concatenated with the speaker embedding $\mathbf{s}$. The weight assigned to each frame is computed by a compatibility function comparing all-pairs frame representations within a window. We set the window size to $\tau' = 256$ frames (4 sec) in all experiments. The output from the last self-attention layer and the initial static landmarks are processed by an MLP to predict the final per-frame landmarks.

Mathematically, our speaker-aware animation models structural dependencies to perturb landmarks that capture head motion and personalized expressions, which can be formulated as follows:

$$\mathbf{h}_t = Attn_s(\tilde{\mathbf{c}}_{t \rightarrow t+\tau'}, \mathbf{s}; \mathbf{w}_{\text{attn},s}), \tag{4.5}$$

$$\Delta \mathbf{p}_t = MLP_s(\mathbf{h}_t, \mathbf{q}; \mathbf{w}_{mlp,s}), \tag{4.6}$$

$$\mathbf{y}_t = \mathbf{p}_t + \Delta \mathbf{p}_t, \tag{4.7}$$

where $\{\mathbf{w}_{\text{attn},s}, \mathbf{w}_{mlp,c}\}$ are trainable parameters of the self-attention encoder and MLP decoder, $\mathbf{p}_t$ is computed by Eq. (4.3), and $\mathbf{y}_t$ are the final per-frame landmarks capturing both speech content and speaker identity. In our implementation, the attention network follows the encoder block in [126]. More details about its architecture are provided in the appendix.

4.2.3 Single-Image Animation

The last step of our model creates the final animation of the input portrait. Given an input image $\mathbf{Q}$ and the predicted landmarks $\{\mathbf{y}_t\}$ for each frame t, we produce a sequence of images $\{\mathbf{F}_t\}$ representing the facial animation. The input portrait might either depict a cartoon face, or a photorealistic human face image. We use different implementations for each of these two types of portraits. In the next paragraphs, we explain the variant used for each type.

Cartoon Images (Non-photorealistic). These images usually have sharp feature edges, e.g., from vector arts or flat shaded drawings. To preserve these sharp features, we propose a morphing-based method to animate them, avoiding pixel-level artifacts. From the input image, we extract the facial landmarks using [145]. We then run Delaunay triangulation on these landmarks to generate semantic triangles. By mapping the initial pixels as texture maps to the triangles, the subsequent animation process becomes straightforward. As long as the landmark topology remains the same, the textures on each triangle naturally transfer across frames. An illustration is shown in Figure 4.3. An analogy with our approach is the vertex and fragment shader pipeline in rendering. The textures are bind to each fragment at the very

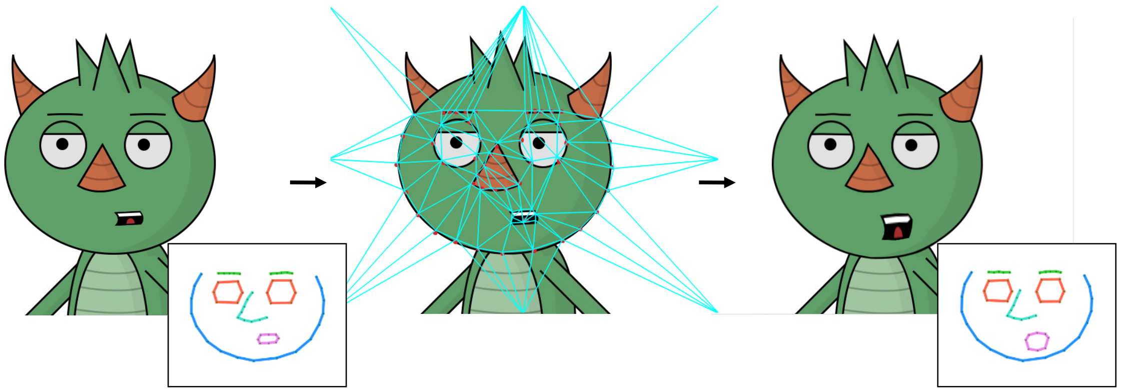


Figure 4.3: Cartoon image face warping through facial landmarks and Delaunay Triangulation. Left: Given cartoon image and its facial landmarks. Middle: Delaunay triangulation. Right: Warped image guided by the displaced landmarks.

beginning and from then on, only the vertex shader is changing the location of these vertices (landmark positions). In practice, we implement a GLSL-based C++ code that uses vertex/fragment shaders and runs in real-time.

Photorealistic Images. The goal here is to synthesize a sequence of frames given the input photo and the predicted animated landmarks from our model. Inspired by the landmark-based facial animation from [148], we first create an image representation $\mathbf{Y}_t$ of the predicted landmarks $\mathbf{y}_t$ by connecting consecutive facial landmarks and rendering them as line segments of predefined color (Figure 4.1). The image $\mathbf{Y}_t$ is concatenated channel-wise with the input portrait image $\mathbf{Q}$ to form a 6-channel image of resolution 256×256. The image is passed to an encoder-decoder network that performs image translation to produce the image $\mathbf{F}_t$ per frame. Its architecture follows the generators proposed in [44] and [59]. Specifically, the encoder employs 6 convolutional layers, where each layer contains one 2-strided convolution followed by two residual blocks, and produces a bottleneck, which is then decoded through symmetric upsampling decoders. Skip connections are utilized between symmetric layers of the encoder and decoder, as in U-net architectures [104]. The generation proceeds for each frame. Since the landmarks change smoothly over time, the output images formed as an interpolation of these landmarks exhibit temporal coherence. Examples of generated image sequences are shown in Figure 4.6.

4.3 Training

We now describe our training procedure to learn the parameters of each module in our architecture.

Voice Conversion Training. We follow the training setup described in [102] with the speaker embedding initialized by the pretrained model provided by [130]. A training source speech from each speaker is processed through the content encoder. Then another utterance of the same source speaker is used to extract the speaker embedding, which is passed to the decoder along with the audio content embedding to reconstruct the original source speech. The content encoder, decoder, and MLP are trained to minimize the self-reconstruction error of the source speech spectrograms [102]. Training is performed on the *VCTK* corpus [127], which is a speech dataset including utterances by 109 native English speakers with various accents.

4.3.1 Speech Content Animation Training

Dataset. To train a content-based animation model, we use an audio-visual dataset that provides high-quality facial landmarks and corresponding audio. To this end, we use the *Obama Weekly Address* dataset [119] containing 6 hours of video featuring various Obama's speeches. Due to its high resolution and relatively consistent front facing camera angle, we can obtain accurate facial landmark detection results using [18]. We also register the facial landmarks to a front-facing standard facial template [11] using a best-estimated affine transformation [109]. This also results in factoring out the speaker-dependent head pose motion, which we will address in Section. 4.3.2. We emphasize that one registered speaker is enough to train this module, since our goal here is to learn a mapping from audio content to facial landmarks.

Loss Function. To learn the parameters $\{\mathbf{w}_{lstm,c}, \mathbf{w}_{mlp,c}\}$ used in the LSTM and MLP, we minimize a loss function that evaluates (a) the distance between the registered reference landmark positions $\hat{\mathbf{p}}_t$ and predicted ones $\mathbf{p}_t$, and (b) the distance between their respective graph Laplacian coordinates, which promotes correct placement of landmarks with respect to each other and preserves facial shape details [117]. Specifically, our loss is:

$$L_c = \sum_{t=1}^{T}\sum_{i=1}^{N} \left\|\mathbf{p}_{i,t} - \hat{\mathbf{p}}_{i,t}\right\|_2^2 + \lambda_c \sum_{t=1}^{T}\sum_{i=1}^{N} \left\|\mathcal{L}(\mathbf{p}_{i,t}) - \mathcal{L}(\hat{\mathbf{p}}_{i,t})\right\|_2^2, \tag{4.8}$$

where i is the index for each individual landmark, and λ_c weighs the second term ($\lambda_c = 1$ in our implementation, set through hold-out validation). We use the following graph Laplacian $\mathcal{L}(\mathbf{p}_t)$:

$$\mathcal{L}(\mathbf{p}_{i,t}) = \mathbf{p}_{i,t} - \frac{1}{|\mathcal{N}(\mathbf{p}_i)|} \sum_{\mathbf{p}_j \in \mathcal{N}(\mathbf{p}_i)} \mathbf{p}_{j,t}, \tag{4.9}$$

where $\mathcal{N}(\mathbf{p}_i)$ includes the landmark neighbors connected to $\mathbf{p}_i$ within a distinct facial part (Figure 4.4). We use 8 facial parts that contain subsets of landmarks predefined for the facial template.

4.3.2 Speaker-Aware Animation Training

Dataset. To learn the speaker-aware dynamics of head motion and facial expressions, we need an audio-visual dataset featuring a diverse set of speakers. We found the *VoxCeleb2* dataset is well-suited for our purpose since it contains video segments from a variety of speakers [31]. VoxCeleb2 was originally designed for speaker verification. Since our goal is different, i.e., capturing speaker dynamics for talking head

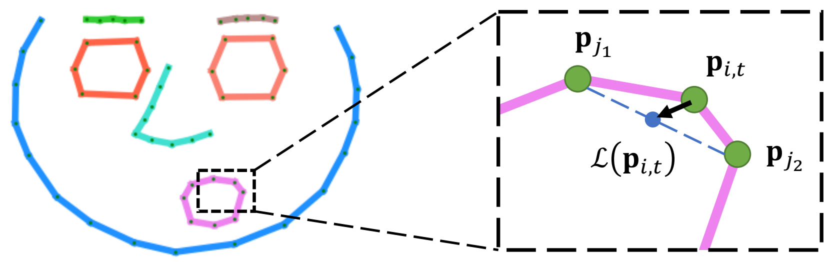


Figure 4.4: Graph Laplacian coordinates illustration. Left: 8 facial parts that contain subsets of landmarks. Right: Zoom-in graph Laplacian vector and related neighboring landmark points.

synthesis, we chose a subset of 67 speakers with a total of 1,232 videos clips from VoxCeleb2. On average, we have around 5-10 minutes of videos for each speaker. The criterion for selection was accurate landmark detection in the videos based on manual verification. Speakers were selected based on Poisson disk sampling on the speaker representation space. We split this dataset as 60% / 20% / 20% for training, hold-out validation and testing respectively. In contrast to the content animation step, we do not register the landmarks to a front-facing template since here we are interested in learning the overall head motion.

Adversarial Network. Apart from capturing landmark position, we also aim to match the speaker's head motion and facial expression dynamics during training. To this end, we incorporate a GAN approach. Specifically, we create a discriminator network $Attn_d$ which follows a similar structure with the self-attention generator network in Section. 4.3.2. More details about its architecture are provided in the appendix. The goal of the discriminator is to find out if the temporal dynamics of the speaker's facial landmarks look "realistic" or fake. It takes as input the sequence of facial landmarks within the same window used in the generator, along with audio content and speaker's embedding. It returns an output characterizing the "realism" r_t per frame t:

$$r_t = Attn_d(\mathbf{y}_{t \to t+\tau'}, \tilde{\mathbf{c}}_{t \to t+\tau'}, \mathbf{s}; \mathbf{w}_{\text{attn},d}), \tag{4.10}$$

where $\mathbf{w}_{\text{attn},d}$ are the parameters of the discriminator. We use the LSGAN loss function [90] to train the discriminator parameters treating the training landmarks as "real" and the generated ones as "fake" for each frame:

$$L_{gan} = \sum_{t=1}^{T} (\hat{r_t} - 1)^2 + r_t^2, \tag{4.11}$$

where $\hat{r_t}$ denotes the discriminator output when the training landmarks $\hat{\mathbf{y}}_t$ are used as its input.

Loss Function. To train the parameters $\mathbf{w}_{\text{attn},s}$ of the self-attention generator network, we attempt to maximize the "realism" of the output landmarks, and also consider the distance to the training ones in terms of absolute position and Laplacian coordinates:

$$\begin{aligned} L_s = & \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| \mathbf{y}_{i,t} - \hat{\mathbf{y}}_{i,t} \right\|_2^2 + \lambda_s \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| \mathcal{L}(\mathbf{y}_{i,t}) - \mathcal{L}(\hat{\mathbf{y}}_{i,t}) \right\|_2^2 \\ & + \mu_s \sum_{t=1}^{T} (r_t - 1)^2, \end{aligned} \tag{4.12}$$

where $\lambda_s = 1$ and $\mu_s = 0.001$ are set through hold-out validation. We alternate training between the generator (minimizing L_s) and discriminator (minimizing L_{gan}) to improve each other as done in GAN approaches [90].

4.3.3 Image-to-Image Translation Training

Finally, we train our image-to-image translation module to handle photorealistic animation outputs. The encoder/decoder pair used for image translation is first trained on paired video frames from VoxCeleb2. Then, we fine-tune the network on a subset which contains high-resolution video crops provided by [114]. In particular, based on a video of a talking person in the dataset, we randomly sample a frame pair: a source training frame $\hat{\mathbf{Q}}_{src}$ and a target frame $\hat{\mathbf{Q}}_{trg}$ of this person. The facial landmarks of the target face are extracted and rasterized into an RGB image $\hat{\mathbf{Y}}_{trg}$ based on the approach described in Section. 4.2.3. The encoder/decoder network takes as input the concatenation of $\hat{\mathbf{Q}}_{src}$ and $\hat{\mathbf{Y}}_{trg}$ and outputs a reconstructed face $\mathbf{Q}_{trg}$. The loss function aims to minimize the $\mathcal{L}_1$ per-pixel distance and perceptual feature distance between the reconstructed face $\mathbf{Q}_{trg}$ and training target face $\hat{\mathbf{Q}}_{trg}$ as in [71]:

$$L_a = \sum_{\{\text{src,trg}\}} ||\mathbf{Q}_{\text{trg}} - \hat{\mathbf{Q}}_{\text{trg}}||_1 + \lambda_a \sum_{\{\text{src,trg}\}} ||\phi(\mathbf{Q}_{\text{trg}}) - \phi(\hat{\mathbf{Q}}_{\text{trg}})||_1,$$

where $\lambda_a = 1$, and ϕ concatenates feature map activations from the pretrained VGG19 network [115].

4.3.4 Implementation Details

All landmarks in our dataset are converted to 62.5 frames per second and audio waveforms are sampled under $16K$ Hz frequency. Both of these rates followed [102], i.e. 62.5 Hz for the mel-spetrogram and 16 kHz for the speech waveform. We experimented with other common frame rates, and we found the above worked well for the entire pipeline. We note that the facial landmarks are extracted from the input video at its original frame rate and the interpolation is performed on landmarks rather than the original pixels. We trained both the speech content animation and speaker-aware animation modules with the Adam optimizer using PyTorch. The learning rate was set to 10^{-4}, and weight decay to 10^{-6}. The speech content animation module contains $1.9M$ parameters and took 12 hours to train on a Nvidia 1080Ti GPU. The speaker-aware animation module has $3.8M$ parameters and took 30 hours to train on the same GPU. The single-face animation module for generating realistic human faces was also trained with the Adam optimizer, a learning rate of 10^{-4}, and a batch

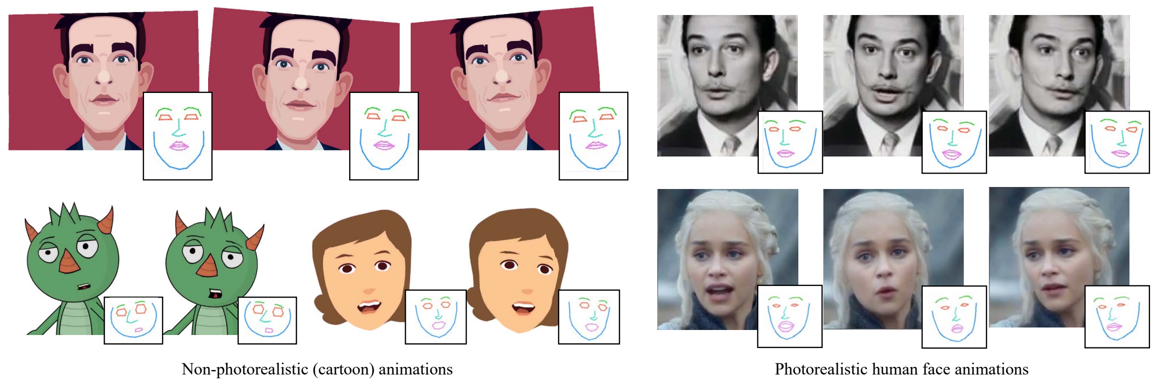

Non-photorealistic (cartoon) animations
Photorealistic human face animations

Figure 4.5: Generated talking-head animation gallery for non-photorealistic cartoon faces (left) and photorealistic human faces (right). The corresponding intermediate facial landmark predictions are also shown on the right-bottom corner of each animation frame. Our method synthesizes not only facial expressions, but also different head poses.

size of 16. The network has 30.7M parameters and was trained for 20 hours on 8 Nvidia 1080Ti GPUs.

4.4 Results

With all the pieces in place, we now present results of talking-head videos from a single input image and a given audio file. Figure 4.5 shows a gallery of our generated animations for cartoon and photorealistic images. We note that the resulting animations include both full facial expressions and dynamic head poses.

In the following sections, we discuss more results for generating photorealistic video and non-photorealistic animations, along with qualitative comparisons. Then we present detailed numerical evaluation, an ablation study, and applications of our method.

4.4.1 Animating Photorealistic Images

Figure 4.6 shows synthesized photorealistic videos featuring talking people as well as comparisons with state-of-the-art video generation methods [28, 128]. The ground-

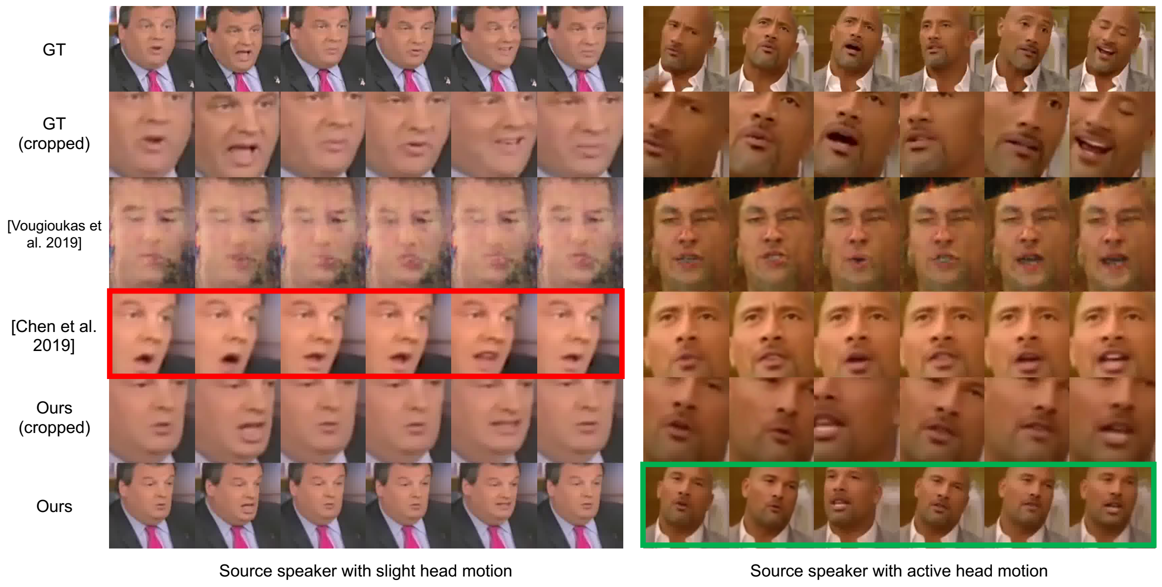

GT
GT (cropped)
[Vougioukas et al. 2019]
[Chen et al. 2019]
Ours (cropped)
Ours
Source speaker with slight head motion
Source speaker with active head motion

Figure 4.6: Comparison with state-of-the-art methods for video generation of photo-realistic talking-heads. The compared methods crop the face and predict primarily the lip region while ours generates both facial expression and head motion. GT and our results are full faces and are cropped for a better visualization of the lip region. Left example: [28] has worse lip synchronization for side-faces (see the red box). Right example: our method predicts speaker-aware head pose dynamics (see the green box). Note that the predicted head pose is different than the one in the ground-truth video, but it exhibits similar *dynamics* that are characteristic for the speaker.

truth (GT) and our results are cropped to highlight the differences in the lip region (see row 2 and 5). We notice that the videos generated by [128, 28] predict primarily the lip region on cropped faces and therefore miss the head poses. [128] does not generalize well to faces unseen during training. [28] lacks synchronization with the input audio, especially for side-facing portraits (see the red box). Compared to these methods, our method predicts facial expressions more accurately and also captures a plausible head motion (see the green box).

Our supplementary video also includes a comparison with the concurrent work by [123]. Given the same audio and target speaker image at test time, we found that our lip synchronization appears to be more accurate than their method. We also emphasize that our method learns to generate head poses, while in their case the head

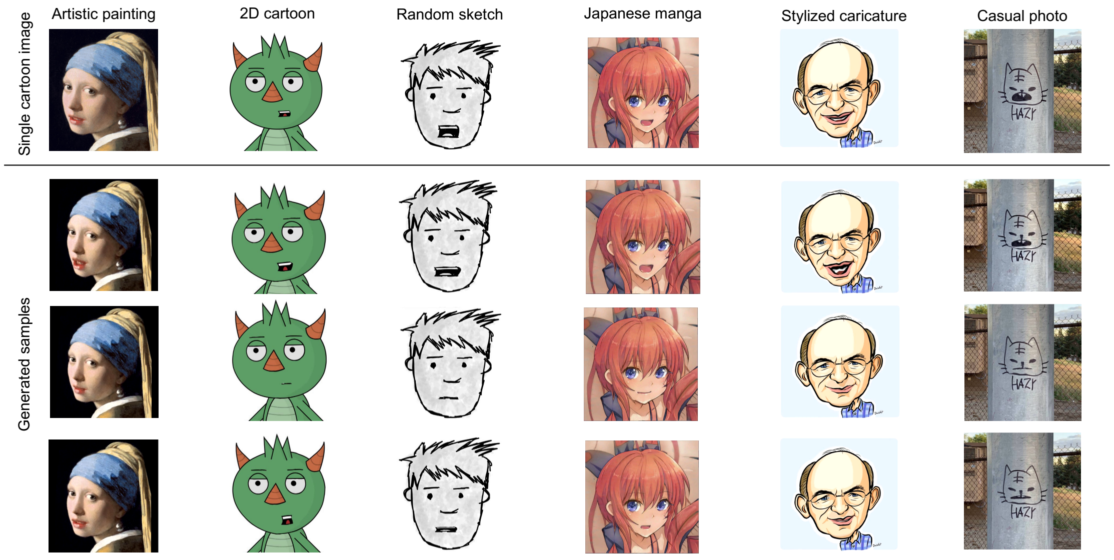


Figure 4.7: Our model works for a variety types of non-photorealistic (cartoon) portrait images, including artistic paintings, 2D cartoon characters, random sketches, Japanese mangas, stylized caricatures and casual photos. Top row: input cartoon images. Next rows: generated talking face examples by face warping. Please also see our supplementary video.

pose is not explicitly handled or is added back heuristically in a post-processing step (not detailed in their paper). Their synthesized video frames appear sharper than ours perhaps due to their neural renderer of their 3D face model, and additional target speaker-specific training. However, this limits the applicability of their method when no target speaker videos are available. Our network instead requires only a single target speaker image to generate the talking-head animation without retraining. As a result, our method has the advantage of driving a diverse set of animations, including animations of cartoon characters, casual photos, paintings, and sketches, for which training videos are often not available.

4.4.2 Animating Non-Photorealistic Images

Figure 4.7 shows a gallery of our generated non-photorealistic animations. Each animation is generated from input audio and a single portrait image. The portrait images can be artistic paintings, random sketches, 2D cartoon characters, Japanese mangas, stylized caricatures and casual photos. Despite being only trained on human facial landmarks, our method can successfully generalize to a large variety of stylized cartoon faces. This is because our method uses landmarks as intermediate representations, and also learns relative landmark displacements instead of absolute positions.

4.4.3 Evaluation Protocol

We evaluated MakeItTalk and compared with related methods quantitatively and with user studies. We created a test split from the VoxCeleb2 subset, containing 268 video segments from 67 speakers. The speaker identities were observed during training, however, their test speech and video are different from the training ones. Each video clip lasts 5 to 30 seconds. Landmarks were extracted using [18] from test clips and their quality was manually verified. We call these as “reference landmarks” and use them in the evaluation metrics explained below.

Evaluation Metrics. To evaluate how well the synthesized landmarks represent accurate lip movements, we use the following metrics:

- **Landmark distance for jaw-lips (D-LL)** represents the average Euclidean distance between predicted facial landmark locations of the jaw and lips and reference ones. The landmark positions are normalized according to the maximum width of the reference lips for each test video clip.

- **Landmark velocity difference for jaw-lips (D-VL)** represents the average Euclidean distance between reference landmark velocities of the jaw and lips and predicted ones. Velocity is computed as the difference of landmark locations between consecutive frames. The metric captures differences in first-order jaw-lips dynamics.

- **Difference in open mouth area (D-A:)**: the average difference between the area of the predicted mouth shape and reference one. It is expressed as percentage of the maximum area of the reference mouth for each test video clip.

To evaluate how well the landmarks produced by our method and others reproduce overall head motion, facial expressions, and their dynamics, we use the following metrics:

- **Landmark distance (D-L)**: the average Euclidean distance between all predicted facial landmark locations and reference ones (normalized by the width of the face).

- **Landmark velocity difference (D-V)**: the average Euclidean distance between reference landmark velocities and predicted ones (again normalized by the width of the face). Velocity is computed as the difference of landmark locations between consecutive frames. This metric serves as an indicator of landmark motion dynamics.

- **Head rotation and position difference (D-Rot/Pos)**: the average difference between the reference and predicted head rotation angles (measured in degrees) and head position (again normalized by the width of the face). The measure indicates head pose differences, like nods and tilts.

4.4.4 Content Animation Evaluation

We begin our evaluation by comparing MakeItTalk with state-of-the-art methods for synthesis of facial expressions driven by landmarks. Specifically, we compare with [42], [153], and [28]. All these methods attempt to synthesize facial landmarks, but cannot produce head motion. Head movements are either generated procedurally or copied from a source video. Thus, to perform a fair evaluation, we factor out head motion from our method, and focus only on comparing predicted landmarks under

Methods	D-LL ↓	D-VL ↓	D-A ↓
Zhou *et al.* [153]	6.2%	0.85%	15.2%
Eskimez *et al.* [42]	4.0%	0.46%	7.5%
Chen *et al.* [28]	5.0%	0.49%	5.0%
Ours (no speaker ID)	2.2%	0.45%	5.9%
Ours (no content)	3.1%	0.61%	10.2%
Ours (full)	**2.0%**	**0.44%**	**4.2%**

Table 4.1: Quantitative comparison of facial landmark predictions of MakeItTalk versus state-of-the-art methods.

an identical "neutral" head pose for all methods. For the purpose of this evaluation, we focus on the lip synchronization metrics (**D-LL**, **D-VL**, **D-A**), and ignore head pose metrics. Quantitatively, Table 4.1 reports these metrics for the above-mentioned methods and ours. We include our full method, including two reduced variants: (a) "Ours (no speaker ID)", where we train and test the speech content branch alone without the speaker-aware branch, (b) "Ours (no content)", where we train and test the speaker-aware branch alone without the speech content branch. We discuss these two variants in more detail in our ablation study (Section 4.4.6). The result shows that our method achieves the lowest errors for all measures. In particular, our method has 2x times less **D-LL** error in lip landmark positions compared to [42], and 2.5x times less **D-LL** error compared to [28].

Figure 4.8 shows characteristic examples of facial landmark outputs for the above methods and ours from our test set. Each row shows one output frame. [153] is only able to predict the lower part of the face and cannot reproduce closed mouths accurately (see second row). [42] and [28], on the other hand, tend to favor conservative mouth opening. In particular, [28] predicts bottom and upper lips that sometimes overlap with each other (see second row, red box). In contrast, our method captures facial expressions that match better the reference ones. Ours can also predict subtle facial expressions, such as the lip-corner lifting (see first row, red box).

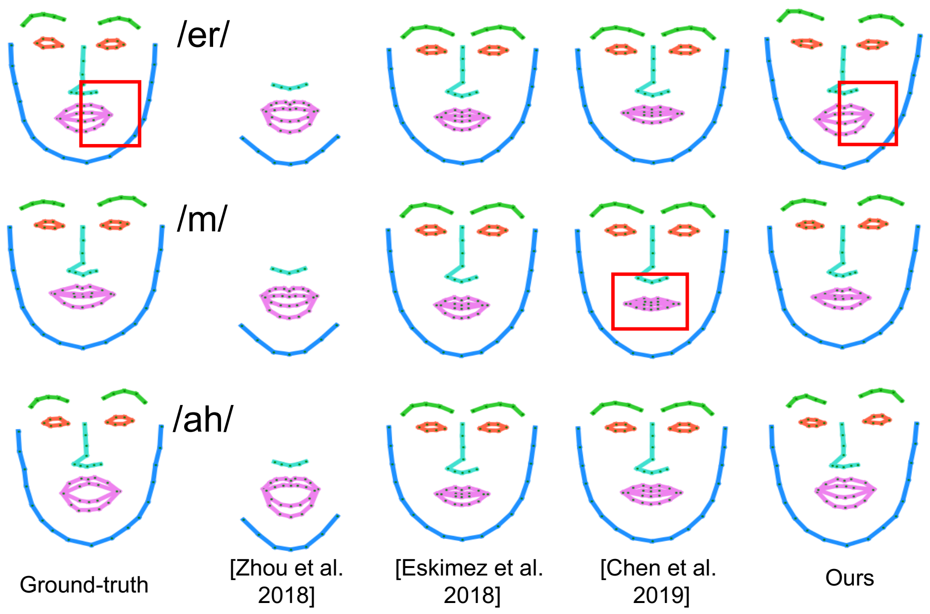


Figure 4.8: Facial expression landmark comparison. Each row shows an example frame prediction for different methods. The GT landmark and uttered phonemes are shown on left.

4.4.5 Speaker-Aware Animation Evaluation

Head Pose Prediction and Speaker Awareness Existing speech-driven facial animation methods do not synthesize head motion. Instead, a common strategy is to copy head poses from another existing video. Based on this observation, we evaluate our method against two baselines: *"retrieve-same ID"* and *"retrieve-random ID"*. These baselines retrieve the head pose and position sequence from another video clip randomly picked from our training set. Then the facial landmarks are translated and rotated to reproduce the copied head poses and positions. The first baseline *"retrieve-same ID"* uses a training video with the same speaker as in the test video. This strategy makes this baseline stronger since it re-uses dynamics from the same speaker. The second baseline *"retrieve-random ID"* uses a video from a different random speaker. This baseline is useful to examine whether our method and alternatives produce head pose and facial expressions better than random or not.

Table 4.2 reports the **D-L**, **D-V**, and **D-Rot/Pos** metrics. Our full method achieves much smaller errors compared to both baselines, indicating our speaker-aware

Methods	D-L ↓	D-V ↓	D-Rot/Pos ↓
retrieve-same ID	17.1%	1.1%	10.3/8.1%
retrieve-random ID	20.8%	1.2%	21.4/9.2%
Ours (random ID)	33.0%	1.8%	28.7/12.3%
Ours (no speaker ID)	13.8%	**0.7%**	12.6/6.9%
Ours (no content)	12.5%	0.8%	8.6/5.7%
Ours (full)	**12.3%**	**0.7%**	**8.0/5.4%**

Table 4.2: Head pose prediction comparison with the baseline methods in S4.4.5 and our variants based on our head pose metrics.

prediction is more faithful compared to merely copying head motion from another video. In particular, we observe that our method produces 2.7× less error in head pose (D-Rot), and 1.7× less error in head position (D-Pos) compared to using a random speaker identity (see *"retrieve-random ID"*). This result also confirms that the head motion dynamics of random speakers largely differ from ground-truth ones. Compared to the stronger baseline of re-using video from the same speaker (see *"retrieve-same ID"*), we observe that our method still produces 1.3× less error in head pose (D-Rot), and 1.5× less error in head position (D-Pos). This result confirms that re-using head motion from a video clip even from the right speaker still results in significant discrepancies, since the copied head pose and position does not necessarily synchronize well with the audio. Our full method instead captures the head motion dynamics and facial expressions more consistently w.r.t. the input audio and speaker identity.

Figure 4.5 shows a gallery of our generated cartoon images and natural human faces under different predicted head poses. The corresponding generated facial landmarks are also shown on the right-bottom corner of each image. The demonstrated examples show that our method is able to synthesize head pose well, including nods and swings. Figure 4.9 shows another qualitative validation of our method's ability to capture personalized head motion dynamics. The figure embeds 8 representative speakers from our dataset based on their variance in Action Units (AUs), head pose

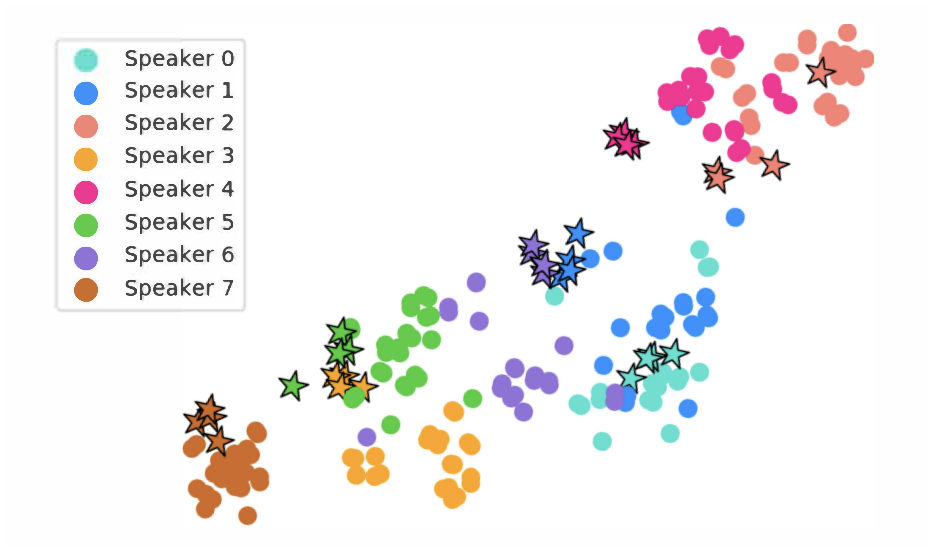


Figure 4.9: t-SNE visualization for AUs, head pose and position variance based on 8 reference speakers videos (solid dots) and our predictions (stars). Different speakers are marked with different colors as shown in the legend.

and position variance. The AUs are computed from the predicted landmarks based on the definitions from [40]. The embedding is performed through t-SNE [88]. These 8 representatives were selected using furthest sampling i.e., their AUs, head pose and position differ most from the rest of the speakers in our dataset. We use different colors for different speakers and use *solid dots* for embeddings produced based on the reference videos in AUs, head pose and position variance, and *stars* for embeddings resulting from our method. The visualization demonstrates that our method produces head motion dynamics that tend to be located more closely to reference ones.

4.4.6 Ablation Study

Individual Branch Performance. We performed an ablation study by training and testing reduced variants of our network: (a) the first variant, called *"Ours (no speaker ID)"* uses only the *speech content animation* branch i.e. omitting the *speaker-aware animation* branch during training/testing, (b) the second variant, called *"Ours (no content)"* uses only the *speaker-aware animation* branch i.e. omitting the *speech* *content animation* branch during training/testing. The aim of this variant is to check whether a single network can jointly learn both lip synchronization and speaker-aware head motion. The results of these two variants and our full method are shown in in Table 4.1 and Table 4.2. We also refer readers to the supplementary video for more visual comparisons.

The variant *"Ours (no speaker ID)"* only predicts facial landmarks from the audio content without considering the speaker identity. It performs well in terms of capturing the lip landmarks, since these are synchronized with the audio content. The variant is slightly worse than our method based on the lip evaluation metrics (see Table 4.1). However, it results in 1.6x larger errors in head pose and 1.3$\times$ larger error in head position (see Table 4.2) since head motion is a function of both speaker identity and content.

The results of the variant *"Ours (no content)"* has the opposite behaviour: it performs well in terms of capturing head pose and position (it is slightly worse than our method, see Table 4.2). However, it has 1.6$\times$ higher error in jaw-lip landmark difference and 2.4$\times$ higher error in open mouth area difference (see Table 4.1), which indicates that the lower part of face dynamics are not synchronized well with the audio content. Figure 4.10 demonstrates that using the speaker-aware animation branch alone i.e., the "Ours (no content)" variant results in noticeable artifacts in the jaw-lip landmark displacements.

Using both branches in our full method offers the best performance according to all evaluation metrics.

Random Speaker ID Injection We tested one more variant of our method called *"Ours (random ID)"*. For this variant, we use our full network, however, instead of using the correct speaker embedding, we inject another random speaker identity embedding. The result of this variant is shown in Table 3. Again we observe that the performance is significantly worse (3.6x more error for head pose). This indicates

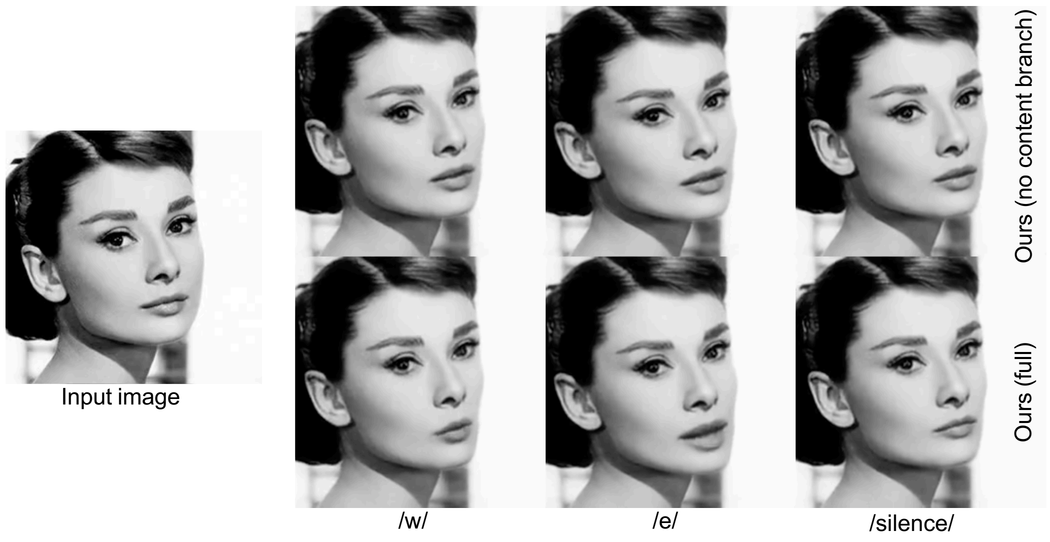


Figure 4.10: Comparison to *"Ours (no content)"* variant (right-top) which uses only the *speaker-aware animation* branch. The full model (right-bottom) result has much better articulation in the lower-part of the face. It demonstrates that a single network architecture cannot jointly learn both lip synchronization and speaker-aware head motion.

that our method successfully splits the content and speaker-aware motion dynamics, and captures the correct speaker head motion dynamics (i.e., it does not reproduce random ones).

4.4.7 User Studies

We also evaluated our method through perceptual user studies via Amazon Mechanical Turk service. We obtained 6480 query responses from 324 different MTurk participants in our two different user studies described below.

User Study for Speaker Awareness To evaluate the *speaker awareness* of different variants of our method, we assembled a pool of 300 queries displayed on different webpages. On top of the webpage, we showed a reference video of a real person talking, and on the bottom we showed two cartoon animations: one cartoon animation generated using our full method and another cartoon animation based on one of the two variants discussed above: (*"Ours (random ID)"* and *"Ours (no speaker ID)"*. The two cartoon videos were placed in randomly picked left/right positions for each webpage. Each query included the following question: "Suppose that you want to see the real person of the video on the top as the cartoon character shown on the bottom. Which of the two cartoon animations best represents the person's talking style in terms of facial expressions and head motion?" The MTurk participants were asked to pick one of the following choices: "left animation", "right animation", "can't tell - both represent the person quite well", "can't tell - none represent the person well". Each MTurk participant was asked to complete a questionnaire with 20 queries randomly picked from our pool. Queries were shown at a random order. Each query was repeated twice (i.e., we had 10 unique queries per questionnaire), with the two cartoon videos randomly flipped each time to detect unreliable participants giving inconsistent answers. We filtered out unreliable MTurk participants who gave two different answers to more than 5 out of the 10 unique queries in the questionnaire, or took less than a minute to complete it. Each participant was allowed to answer one questionnaire maximum to ensure participant diversity. We had 90 different, reliable MTurk participants for this user study. For each of our 300 queries, we got votes from 3 different MTurk participants. Since each MTurk participant voted for 10 unique queries twice, we gathered 1800 responses (300 queries $\times$ 3 votes $\times$ 2 repetitions) from our 90 MTurk participants. Figure 4.11 (top) shows the study result. We see that the majority of MTurkers picked our full method more frequently, when compared with either of the two variants.

User Study for Photorealistic Videos. To validate our landmark-driven *photo-realistic animation* method, we conducted one more user study. Each MTurk participant was shown a questionnaire with 20 queries involving random pairwise comparisons out of a pool with 780 queries we generated. For each query, we showed a single frame showing the head of a real person on top, and two generated videos below (ran-

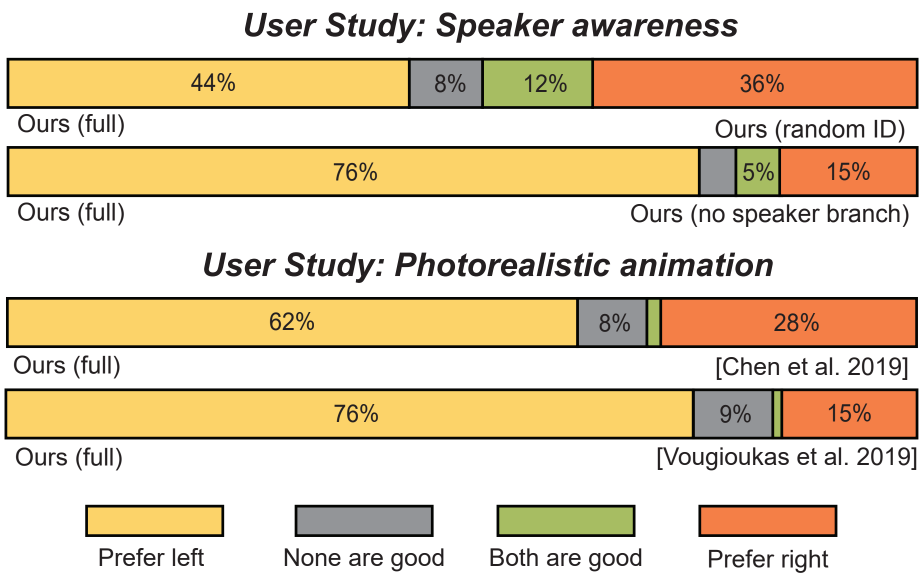


Figure 4.11: User study results for speaker awareness (top) and photorealistic animation (bottom).

domly placed at left/right positions): one video synthesized from our method, and another from either [128] or [28]. The participants were asked which person's facial expression and head motion look more realistic and plausible. We also explicitly instruct them to ignore the particular camera position or zoom factor and focus on the face. Participants were asked to pick one of four choices ("left", "right", "both" or "none") as in the previous study. We also employed the same random and repeated query design and MTurker consistency and reliability checks to filter out unreliable answers. We had 234 different MTurk participants for this user study. Like in the previous study, each query received votes from 3 different, reliable MTurk participants. As a result, we gathered 780 queries $\times$ 3 votes $\times$ 2 repetitions = 4680 responses from our 234 participants. Figure 4.11 (bottom) shows the study result. Our method was voted as the most "realistic" and "plausible" by a large majority, when compared to [28] or [128].

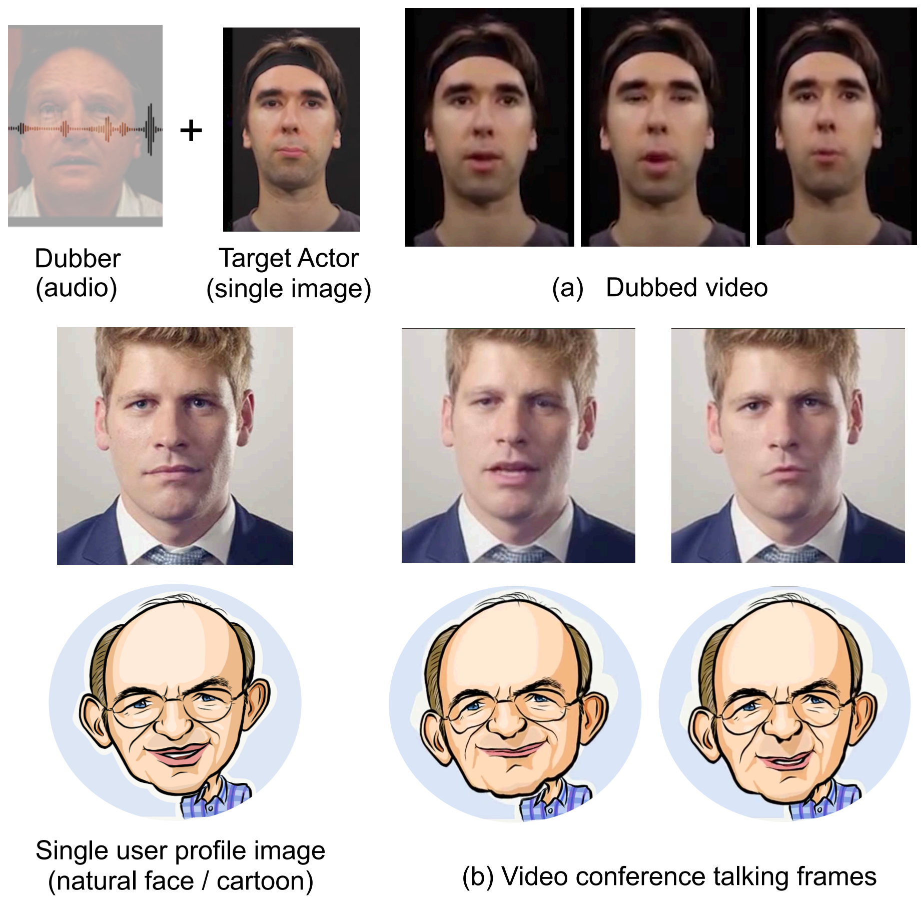


Figure 4.12: Applications. Top row: video dubbing for target actor given only audio as input. Middle and bottom row: bandwidth-limited video conference for photorealistic and non-photorealistic user profile images. Please also see our supplementary video.

4.4.8 Applications

Synthesizing realistic talking-heads has numerous applications [74, 151]. One common application is dubbing using voice from a person different from the one in the original video, or even using voice in a different language. In Figure 4.12(top), given a single frame of the target actor and a dubbing audio spoken by another person, we can generate a video of the target actor talking according to that other person's speech.

Another application is bandwidth-limited video conference. In scenarios where the visual frames cannot be delivered with high fidelity and frame-rate, we can make use only of the audio signal to drive the talking-head video. Audio signal can be preserved under much lower bandwidth compared to its visual counterpart. Yet, it is still important to preserve visual facial expressions, especially lip motions, since they heavily contribute to understanding in communication [92]. Figure 4.12(middle) shows that we can synthesize talking heads with facial expressions and lip motions with only the audio and an initial high-quality user profile image as input. Figure 4.12(bottom) shows an example of non-photorealistic talking-head animation that can be used in teleconferencing for entertainment reasons, or due to privacy concerns related to video recording. We also refer readers to the supplementary video.

Our supplementary video also demonstrates a text-to-video application, where we synthesize photorealistic video from text input, after converting it to audio through a speech synthesizer [97].

Finally, our video demonstrates the possibility of interactively editing the pose of our synthesized talking heads by applying a rotation to the intermediate landmarks predicted by our network.

4.5 Ethical Considerations

"Deepfake videos" are becoming more prevalent in our everyday life. The general public might still think that talking head videos are hard or impossible to generate synthetically. As a result, algorithms for talking head generation can be misused to spread misinformation or for other malicious acts. I hope the method proposed will help people understand that generating such videos is entirely feasible. My main intention is to spread awareness and demystify this technology. The corresponding code for this method is publicly released on the Github[2] which includes a watermark to the generated videos making it clear that they are synthetic.

4.6 Conclusion

We have introduced a deep learning based approach to generate speaker-aware talking-head animations from an audio clip and a single image. Our method can handle new audio clips and new portrait images not seen during training. Our key insight was to predict landmarks from disentangled audio content and speaker, such that they capture better lip synchronization, personalized facial expressions and head motion dynamics. This led to much more expressive animations with higher overall quality compared to the state-of-the-art.

[2]https://github.com/yzhou359/MakeItTalk

CHAPTER 5

AUDIO-DRIVEN NEURAL GESTURE REENACTMENT WITH VIDEO MOTION GRAPHS

Human speech is often accompanied by body gestures including arm and hand gestures. The third part of my thesis focuses on generating plausible speech gestures while human talking.[1] We present a method that reenacts a high-resolution and high-quality video with gestures matching a target speech audio. The key idea of our method is to split and re-assemble clips from a reference video through a novel video motion graph encoding valid transitions between clips. To seamlessly connect different clips in the reenactment, we propose a pose-aware video blending network which synthesizes video frames around the stitched frames between two clips. Moreover, we developed an audio-based gesture searching algorithm to find the optimal order of the reenacted frames. Our system generates reenactments that are consistent with both the audio rhythms and the speech content. We evaluate our synthesized video quality quantitatively, qualitatively, and with user studies, demonstrating that our method produces videos of much higher quality and consistency with the target audio compared to previous work and baselines.

5.1 Method

5.1.1 Overview.

The goal of our method is to synthesize a new video for a reference speaker given a target speech audio from the same or different speaker. Our video synthesis is

[1]This work is submitted to ICCV 2021 and is currently under review

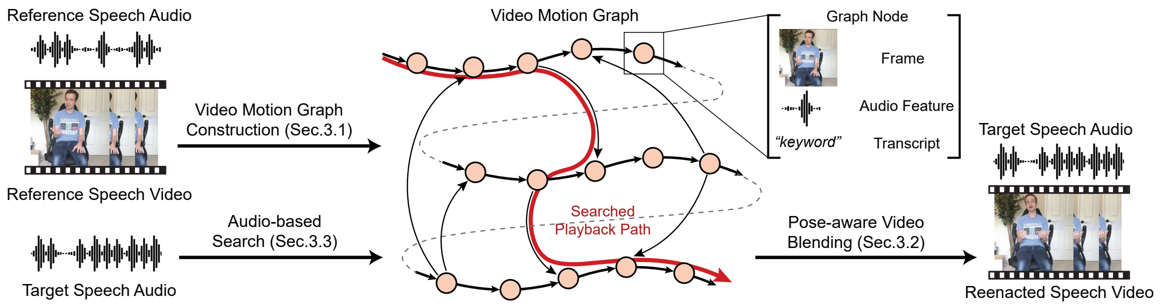


Figure 5.1: System overview. The reference video is first encoded into a directed graph where nodes represent video frames and audio features, and edges represent transitions. The transitions include original ones between consecutive reference frames, and synthetic ones between disjoint frames. Given a unseen target audio at test time, a beam search algorithm finds plausible playback paths such that gestures best match the target speech audio. Synthetic transitions along disjoint frames are neurally blended to achieve temporal consistency.

guided by a novel *video motion graph* created from an input reference video of the speaker (Section. 5.1.2). The video motion graph is a directed graph that encodes how the reference video may be split and re-assembled in different graph paths (see Fig. 5.1 for an illustration). The graph node representations are defined as the raw reference video frames and corresponding audio features. The edges are defined as the transitions between frames, including *natural transitions* between the original consecutive frames in the input video and *synthetic transitions* connecting disjoint clips. Synthetic transitions are introduced to expand the graph connectivity and enable nonlinear video playback.

However, a direct nonlinear playback along synthetic transitions does not guarantee smooth video rendering due to the abrupt changes of disjoint frames in image space. Thus, we design a novel *pose-aware video blending* network to re-render and interpolate neighboring frames required by the synthetic transitions (Section. 5.1.3). We develop an *audio-based searching* method to find optimal paths in the video motion graph that best match the target audio features both rhythmically and semantically

(Section. 5.1.4). To generate new videos, we retrieve the raw input video frames at natural transitions and synthesize neural blended frames at synthetic transitions.

5.1.2 Video Motion Graph

The key idea of our video motion graph is to create synthetic transitions based on the similarity of the speaker's pose in the reference video frames. Our pose similarity metric relies on 3D space and image space cues. Given a reference video, our first step is to extract pose parameters $\boldsymbol{\theta}$ of the SMPL model [87] for all frames with an off-the-shelf motion capture method [141]. We further smooth the pose parameters with [24] to promote temporally coherent results.

3D Space Pose Similarity. Based on the pose parameters, we compute the 3D positions and velocities in world space for all joints via forward kinematics. For each pair of frames with indices (m, n) (including non-consecutive ones), we evaluate pose dissimilarity $d_{feat}(m, n)$ by calculating the Euclidean distance of their feature vectors produced by concatenating the SMPL pose parameters, positions, and velocities of all joints.

Image Space Pose Similarity. To obtain the pose similarity in image space, for each frame m, we project the fitted 3D SMPL human mesh onto image space using known camera parameters from [141], and mark the mesh surface area which is visible on image after projection as S_m. Then for each pair of frames (m, n), we compute their image space dissimilarity based on the the Intersection-over-Union (IoU) between their corresponding visible surface areas: $d_{img}(m, n) = 1 - (S_m \cap S_n)/(S_m \cup S_n)$. The lower $d_{img}(m, n)$ is, the higher the IoU, thus larger overlap exists in the surface area in two meshes, indicating higher pose similarity in terms of image rendering.

Based on these two distance measurements, we create graph synthetic transitions between any pair of reference video frames (nodes in our graph) if their distance

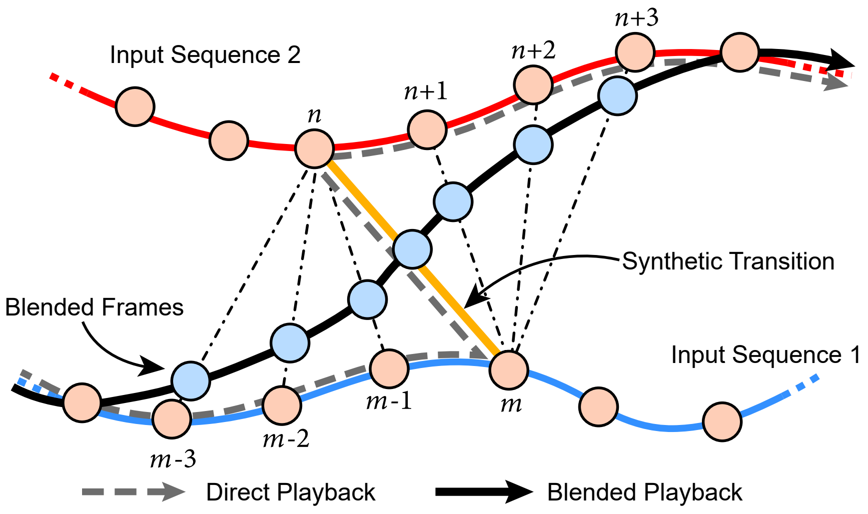
$d_{feat}(m,n)$ and $d_{img}(m,n)$ are below predefined thresholds (both distance for natural transitions are defined as 0). Here we follow [144] to set the thresholds as the corresponding average distance between frames $(m, m+l)$ in the reference video. Larger frame offset l results in higher thresholds, thus more synthetic transitions, increasing the possible number of paths in the resulting denser graph. This also results in larger computational cost for the path search algorithm of Section. 5.1.4. Our experiments use $l = 4$ which we practically find to achieve a good balance between computational cost and number of available paths in the graph.

5.1.3 Pose-aware Video Blending

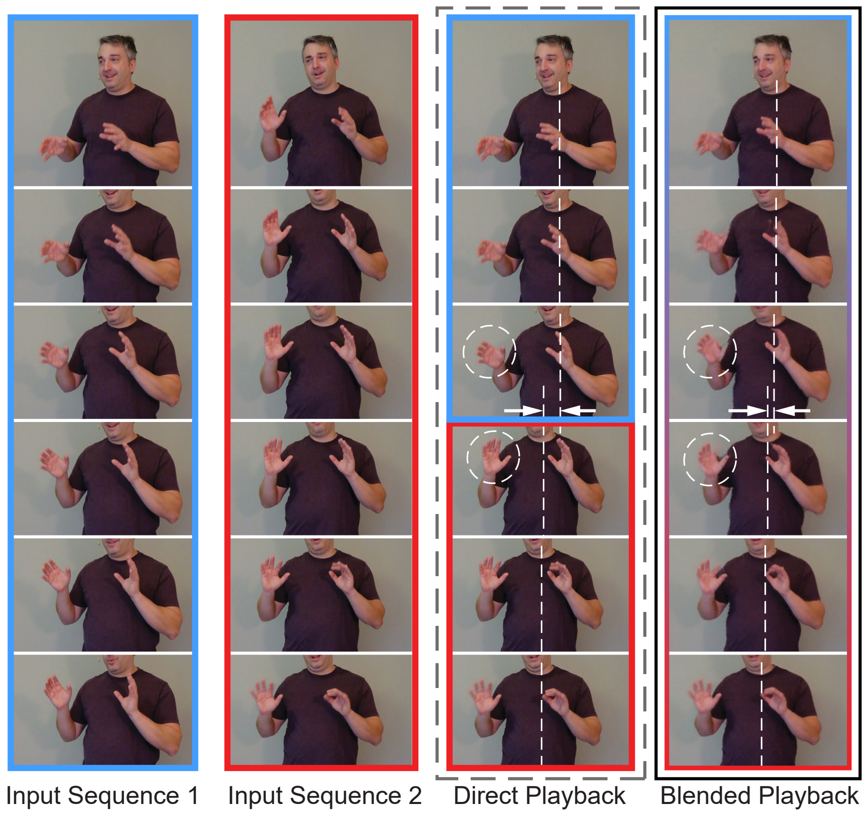
A mere playback of connected frames at synthetic transitions easily results in noticeable jittering artifacts (see direct playback in Fig. 5.2(a) grey dashed path and Fig. 5.2(b) third column).

To solve this problem, we synthesize blended frames to replace original frames around a small temporal neighborhood of a synthetic transition so that the video can smoothly blend from the first sequence to the other (see Fig. 5.2(a) solid black path and Fig. 5.2(b) last column). For a synthetic transition connecting frames m, n, we define the neighborhood using the frame range $[m-k, m]$ and $[n, n+k]$ with a neighborhood size k.

We designed a *pose-aware video blending network* to synthesize frames within the above neighborhood. Given two frames with indices i, j (where $i \in [m-k, m]$ and $j \in [n, n+k]$) and their corresponding raw RGB image representations I_i and $I_j \in \mathbb{R}^{H \times W \times 3}$ from the reference video, the network synthesizes each blended frame in the neighborhood with a target blended weight $\alpha \in [0, 1/K, 2/K, ..., 1]$, where $K = 2k$.



(a) Illustration of pose-aware blended playback.



(b) Our blended playback generates smoother transition.

Figure 5.2: Compared to direct playback along synthetic transitions which have severe horizontal shift for body poses and abrupt change in hand rotations (see dashed lines and circles in (b)), our blending strategy generates natural transitions between clips.

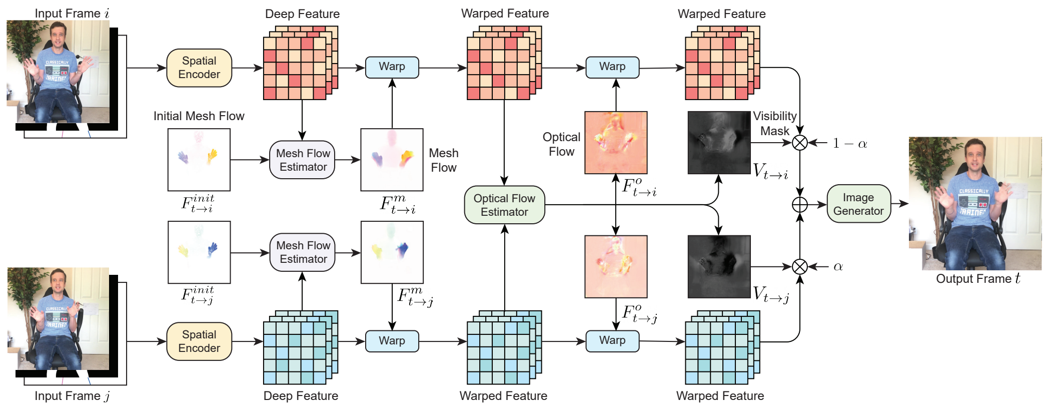


Figure 5.3: Pose-aware neural blending network architecture. Two source frames are encoded into deep feature maps and then warped based on the predicted flows from two stages: a 3D mesh-based flow stage for coarse feature map alignment, followed by an optical flow-based stage further refining the warping. Finally, the warped features are blended with predicted visibility masks to generate the target frame.

As a first step, we use the blending weight to estimate the SMPL pose parameter $\boldsymbol{\theta}_t$ for a blended frame t as: $\boldsymbol{\theta}_t = (1-\alpha)\boldsymbol{\theta}_i + \alpha\boldsymbol{\theta}_j$, where $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are the SMPL pose parameters captured from two input frames respectively.

Our network processes the images I_i, I_j, the body foreground masks extracted from the silhouettes of the fitted SMPL meshes, and the pose parameter $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \boldsymbol{\theta}_t$. Processing takes place in two stages. The first stage warps foreground human body image features based on a 3D motion field computed from vertex displacements of the fitted SMPL meshes. The second stage further refines the warping by computing the residual optical flow between the warped body image features produced by the first stage, and the optical flow for the rest of the image (i.e., background). An image translation network transforms the refined warped image features to the image I_t representing the target output frame t. The network architecture is shown in Fig. 5.3.

Mesh Flow Stage. The first stage has two parallel streams, each producing image deep feature maps encoding the warping for the input images I_i and I_j. To produce these features, we first compute an initial 3D motion field, which we refer to as initial "mesh flow", from the SMPL body mesh displacements between the two frames. To this end, we first find the body mesh vertex positions $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_t$ from the SMPL pose parameters $\boldsymbol{\theta}_i, \boldsymbol{\theta}_j, \boldsymbol{\theta}_t$ respectively. Then we obtain the initial mesh flow $F^{init}_{t\to i}$ and $F^{init}_{t\to j}$ as the displacement of the corresponding mesh vertices $\mathbf{v}_t - \mathbf{v}_i$ and $\mathbf{v}_t - \mathbf{v}_j \in \mathbb{R}^{N\times 3}$ respectively. We note that we only consider here the displacements from visible vertices found via perspective projection onto image plane. These displacements are projected and rasterized as image-space motion field $\mathbb{R}^{N\times 3} \to \mathbb{R}^{H\times W\times 2}$. Since the vertex sampling does not match the image resolution, the resulting flow fields are rather sparse. Thus, we diffuse them with a Gaussian kernel with σ set to 8 in our experiments.

These initial motion fields are far from perfect. This is because the boundaries of the projected mesh often do not exactly align with the boundaries of the human body in the input frames. Thus, we refine these fields with a neural module. The module has two streams, each refining the corresponding motion field for frame i and j. The first stream processes as inputs the RGB image I_i, the foreground mask I_{mask}, and an image containing the rendered skeleton I_{skel} representing the SMPL pose parameters. It then encodes them into an image deep feature map $\mathbf{x}_i$:

$$\mathbf{x}_i = E_s(I_i, I_{mask}, I_{skel}; \mathbf{w}_s) \tag{5.1}$$

where $\mathbf{w}_s$ are learnable weights. Similarly, the second stream produces an image deep feature map $\mathbf{x}_j$ for frame j. The two streams share the same network based on 8 stacked CNN residual blocks [17]. More details are provided in the supplementary material.

We then estimate the refined motion fields through another network E_m,

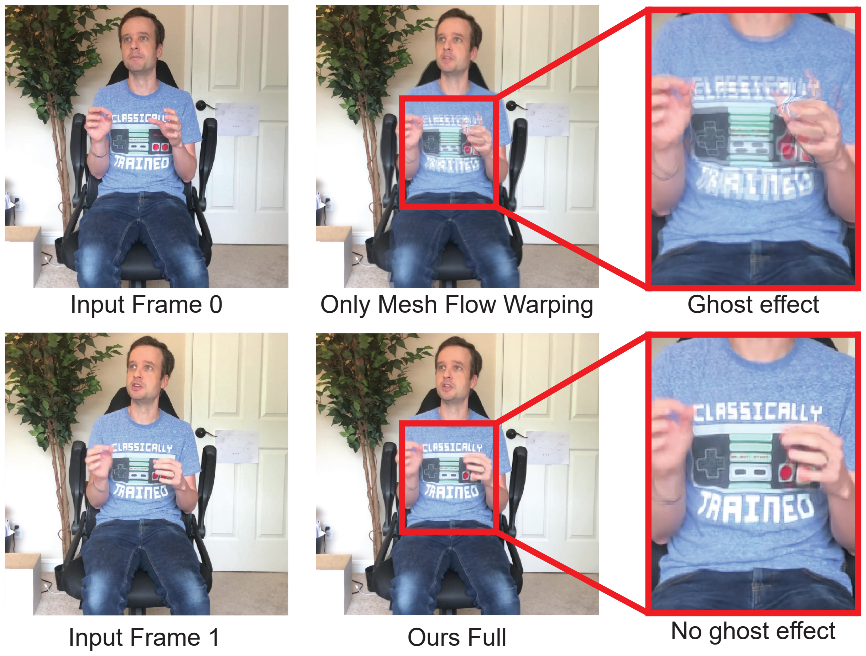


Figure 5.4: A ghost effect example. **Left**: two input frames. **Top-right**: ghost effect from using mesh flow only. **Bottom-right**: sharp features with further warping by optical flow.

$$F^m_{t\to i} = E_m(\mathbf{x}_i, F^{init}_{t\to i}; \mathbf{w}_m), \tag{5.2}$$

$$F^m_{t\to j} = E_m(\mathbf{x}_j, F^{init}_{t\to j}; \mathbf{w}_m). \tag{5.3}$$

where $\mathbf{w}_m$ are learnable weights. This network is designed based on UNet [104]. More details are provided in the supplementary material. We then backwards warp the above image feature maps with the above motion fields to obtain the warped deep features $\mathbf{x}'_i$ and $\mathbf{x}'_j$.

Optical Flow Stage. Synthesizing the final target frame directly from the two warped feature maps $\mathbf{x}'_i$ and $\mathbf{x}'_j$ suffers from ghost effect (Fig. 5.4). This is because the motion field calculated in the previous stage is based on the SMPL model which ignores details such as textures on clothing.

Our second stage aims to further warp the deep feature maps x'_i and x'_j based on optical flow computed throughout the image including the background. At this stage, the warped features already represent bodies that are roughly aligned. We found that an off-the-shelf frame interpolation network based on optical flow [70] can reproduce the missing pixel-level details and remedy the ghost effect. The network predicts optical flow $F^o_{t\to i}$ and $F^o_{t\to j}$ to further warp the features from $\mathbf{x}'_i$ and $\mathbf{x}'_j$ to $\mathbf{x}''_i$ and $\mathbf{x}''_j$ respectively. It also estimates soft visibility maps [70] $V_{t\to i}$ and $V_{t\to j}$ used for blending to obtain a deep feature map for frame t:

$$x''_t = (1-\alpha)V_{t\to i} \odot x''_i + \alpha V_{t\to j} \odot x''_j. \tag{5.4}$$

Finally, we take as input the above blended deep feature map to synthesize the target image I_t. This is performed with a generator network G following a UNet image translation network architecture from Chapter. 4:

$$\hat{I}_t = G(x''_t; \mathbf{w}_g) \tag{5.5}$$

where $\mathbf{w}_g$ are learnable weights.

Fig. 5.5 and Fig. 5.6 show output images from the video blending network for different blending weights, along with results from our intermediate stages.

Training. To train our pose-aware video blending network, we sample triplets of frames in the reference video. Given a target frame e.g., frame t, we randomly sample two other nearby frames with indices $t-k_0$ and $t+k_1$, $k_0, k_1 \in [1, 8]$ to form triplets. The corresponding blending weight α is computed as $k_1/(k_1+k_2)$. We train the entire network end-to-end with losses defined to better estimate the flows and reconstruct the final image.

More specifically, we first have an L1 reconstruction loss L_{rec} and a perceptual loss L_{per} between the synthesized image $\hat{I}_t$ and I_t:

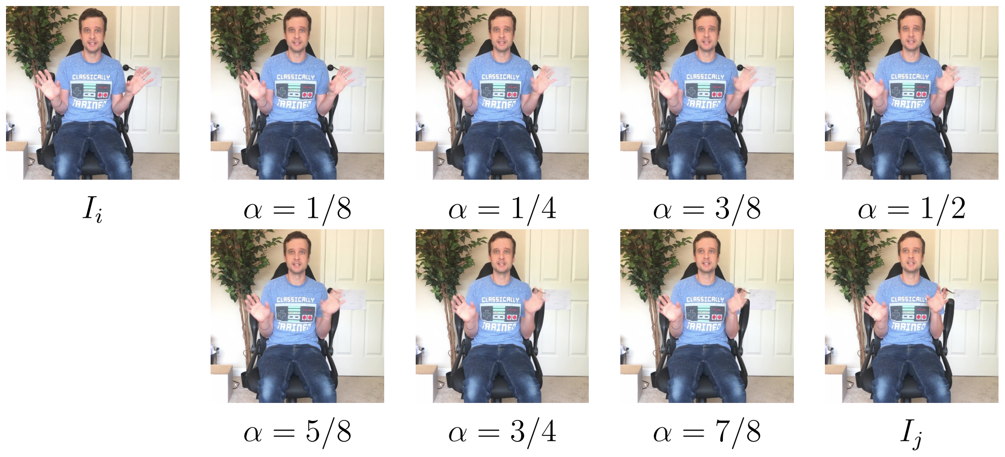


Figure 5.5: Pose-aware video blending results of synthesized in-between frames with blended human gestures for different blending weights $\alpha \in (0, 1)$.

$$L_{rec} = \mathcal{L}_1(I_t, \hat{I}_t) \tag{5.6}$$

$$L_{per} = \mathcal{L}_1(\phi(I_t), \phi(\hat{I}_t)) \tag{5.7}$$

where $\phi(\cdot)$ concatenates feature map activations from a pre-trained VGG19 network [115].

We then adopt another L1 reconstruction loss $\mathcal{L}^b_{rec}$ promoting better frame reconstruction directly from the warped deep features x''_i and x''_j after these pass through our generator network G. This helped predict warped deep features such that they lead to generating frames as close as possible to ground-truth in the first place. We also empirically observed faster convergence with this loss:

$$L^b_{rec} = \mathcal{L}_1(I_t, G(x''_i)) + \mathcal{L}_1(I_t, G(x''_j)) \tag{5.8}$$

Further, we have warping loss L^m_{warp} and L^o_{warp} by measuring the L1 reconstruction error between the target image and the source images I_i and I_j after being warped through the motion field $F^m_{t\to i}$ and also the optical flow $F^o_{t\to i}$:

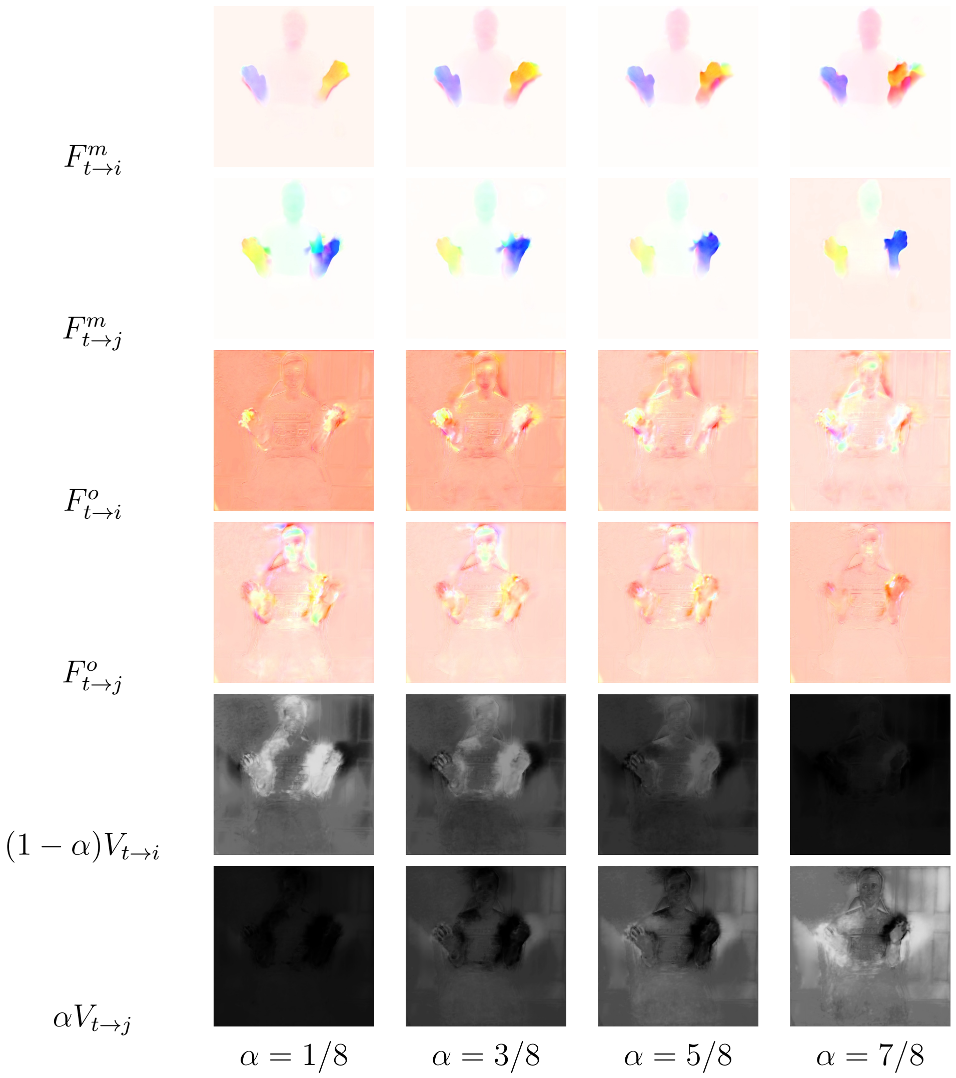


Figure 5.6: Pose-aware video blending results of intermediate mesh flows, optical flows and visibility maps results for corresponding blending weights $\alpha \in (0, 1)$.

$$\begin{aligned} L^m_{warp} = \mathcal{L}_1(I_t, \mathcal{W}(I_i, F^m_{t\to i})) + \\ \mathcal{L}_1(I_t, \mathcal{W}(I_j, F^m_{t\to j})) \end{aligned} \tag{5.9}$$

$$\begin{aligned} L^{o}_{warp} = \mathcal{L}_1(I_t, \mathcal{W}(\mathcal{W}(I_i, F^{m}_{t\to i}), F^{o}_{t\to i}))+ \\ \mathcal{L}_1(I_t, \mathcal{W}(\mathcal{W}(I_j, F^{m}_{t\to j}), F^{o}_{t\to j})) \end{aligned} \tag{5.10}$$

where $\mathcal{W}(I, F)$ applies backward warping flow F on image I.

Finally, we follow [70] and include a smoothness loss for both mesh flow and optical flow:

$$L_{sm} = ||\nabla F^{m}_{t\to i}||_1 + ||\nabla F^{m}_{t\to j}||_1 + \tag{5.11}$$

$$||\nabla F^{o}_{t\to i}||_1 + ||\nabla F^{o}_{t\to j}||_1 \tag{5.12}$$

The overall loss $\mathcal{L}$ is defined as the weighted sum of all losses described above, then averaged over all training frames.

$$\begin{aligned} \mathcal{L} = \mathcal{L}_{rec} + \lambda_p \mathcal{L}_{per} + \lambda_b \mathcal{L}^{b}_{rec} + \\ \lambda_m \mathcal{L}^{m}_{warp} + \lambda_o \mathcal{L}^{o}_{warp} + \lambda_s \mathcal{L}_{sm} \end{aligned} \tag{5.13}$$

The weights have been set empirically using the validation set as $\lambda_p = 0.01,\ \lambda_b = 0.25,\ \lambda_m = 0.25,\ \lambda_o = 0.25, \lambda_s = 0.01$.

To train the entire model, we first train the mesh flow estimator network with L^{m}_{warp} as a "warming" stage. Then we load a pre-trained optical flow model from [70]. Finally, we train the entire network end-to-end with the loss mentioned above. The network weights are optimized with Adam optimizer using PyTorch. The learning rate is set to 10^{-4} and weight decay to 10^{-6}. The training process is performed on 4 Nvidia GeForce 1080Ti GPUs. Each speaker video takes around 10k iterations to converge.

5.1.4 Audio-based Search

Given a speech audio at test time, we develop a graph search algorithm to find plausible paths along which gestures match the speech audio both rhythmically and semantically. Previous studies have shown that speech gestures can be classified into two categories: 1) referential gestures that appear together with specific, meaningful keywords, and 2) rhythmic gestures which respond to the audio prosody features [93]. More specifically, the key stroke of a rhythmic gesture appear at the same time as (or within a very short of period of) an *audio onset* within a phonemic clause [41]. To find precise gestures on the right timings, or frame indices, we define a pair of audio features for input speech: *audio onset feature* and *keyword feature*. The audio frame indices match the video frame rate.

Audio Onset and Keyword Feature. We define the audio onset feature as a binary value indicating the activation of an audio onset for each frame detected with a standard audio processing algorithm [12]. To extract keyword features, we first use the Microsoft Azure speech-to-text engine [142] to convert the input audio into transcripts with corresponding start and end time for each word. We create a dictionary of common words for referential gestures, which we call *keywords* (see supplementary for a list). If a keyword appears at a frame (or node), we set its keyword feature to that word. Otherwise, we simply set it to *empty* (no keyword).

Target Speech Audio Segmentation. We split the target speech audio into segments starting and ending with the frames where the audio onset or keyword feature is activated. Let $\{a_s\}_{s=1}^{S}$ be the frame indices of such frames, where S is their total number. Segments are represented as $a_s \to a_{s+1}$, and their duration are $L_s = a_{s+1} - a_s$ (number of frames). We also add two extra endpoints $a_0 = 1$ and $a_{S+1} = N_t$ indicating the first and last frame of the target audio respectively to form the complete segment list, i.e. $a_s \to a_{s+1}$, $s = 0, 1, .., S$.

Beam Search. We initialize a beam search [106] procedure in the video motion graph to find K plausible paths matching the target speech audio segments. We set K to 20. The beam search initializes K paths starting with K random nodes as the first frame a_0 for the target audio, then expands in a breadth-first-search manner to find paths ending at a *target graph node* whose audio feature matches the target audio feature at the endpoint of the first segment a_1, associated with either an activated audio onset or the same non-empty keyword feature. Note that there can be multiple target graph nodes sharing the same audio feature with a_1.

During the beam search, all the explored paths are sorted based on a *path transition cost*, plus a *path duration cost*. The path transition cost is defined as the sum of node distances (see Section. 5.1.2) between all consecutive nodes m, n along the path, i.e. $\sum_{m,n} (d_{feat}(m, n) + d_{img}(m, n))$. The cost of synthetic transitions are always higher than natural ones. Thus, the path cost prevents using implausible paths with too many synthetic transitions.

When a path reaches a *target* graph node, we check its duration. Due to the sparsity of the graph, there may not be any path matching exactly the target audio segment length L_i. Still, the path length should be similar to L_i, otherwise one would need to accelerate or decelerate the path too much to adjust it to the exact length, leading to unnaturally fast or slow gestures. We only accept paths with duration $L'_s \in [0.9L_s, 1.1L_s]$ since these can be slightly adjusted, e.g. re-sampled, to match the target segment duration. For the above range, we observed that the motion still looks natural. Nevertheless, we also add a path duration cost $|1 - L'_s/L_s|$ to favor paths during beam search with duration closer to the target duration.

After processing the first segment $a_0 \rightarrow a_1$, we start another beam search for the next segment $a_1 \rightarrow a_2$. Here, the path expansion starts with the last node of the K paths discovered from previous iteration. The expansion continues with the same search procedure as above. In order, the searches run iteratively for all the rest segments $a_s \rightarrow a_{s+1}$, $s \in [1, S]$ while always keeping the most plausible K paths. All searched K paths can be used to generate various plausible results for the same target speech audio (see the supplementary video). The best path is picked in our experiments.

Video Synthesis. We generate a video along with the final path in the motion graph discovered by the beam search executions, and use the blending network to handle synthetic transitions (see Fig. 5.2 for an example). As explained above, for each synthesized video segment corresponding to target audio segment $a_s \rightarrow a_{s+1}$, we adjust its speed to match the target duration. Finally, we post-process our result by adopting [101] to synchronize the lips of the speaker to match the corresponding speech audio.

5.2 Results and Evaluation

Dataset. We collected seven speech videos. Each speaker is asked to tell a personal story in front of a static camera, either standing or sitting. Speakers are encouraged to use their gestures while telling the stories. The length of the video varies between 2-10 minutes depending on the story.

5.2.1 Audio-driven Reenactment Results

Given a reference video from speaker A and a target audio clip randomly from another speaker B in our dataset, we reenact the reference video with our system to generate a new speech video with A's appearance and B's voice. We generate 127 such videos of 25 seconds in length and 512×512 in resolution, each of which contains expressive speech gestures. Example results are provided in the supplementary video.

Evaluation. To evaluate the consistency of such reenacted videos to target speech audio, we perform a perceptual user study via the Amazon Mechanical Turk service.

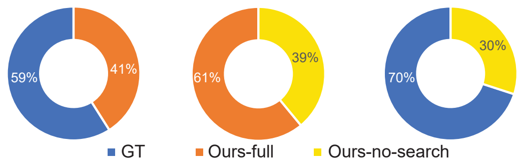


Figure 5.7: Pairwise comparison results from our user study. The comparison of *Ours-full* against *Ours-no-search* shows the effectiveness of the proposed audio-based search algorithm.

We compare the results from our full system (**Ours-full**) against ground-truth (**GT**), which are original reference video clips of speaker B, and results from a baseline system (**Ours-no-search**), which randomly finds paths along video motion graphs without audio-based search. All videos are post-processed by masking face regions with a solid white box to prevent facial motions from dominating visual perception.

We design the user study questionnaire by providing a list of queries involving pairwise comparisons of results from two out of three methods mentioned above. We have a pool of 381 queries (127 videos from each method $\times$ 3 comparison pairs). For each query, we show two videos in parallel randomly placed at left/right positions. The participants are asked which speaker's gestures are more consistent with the speech audio and vote for one of the two choices: "left animation", "right animation". We also explicitly instruct them to focus on the speakers' hand gestures and ignore the masked facial area. Detailed user study setup are found in the supplementary material. Each participant is asked to complete a questionnaire with 10 queries randomly picked from our pool and altogether 1130 votes from 113 valid participants are gathered. We plot the statistics in Fig. 5.7.

The preference (61% vs. 39%) of *Ours-full* over *Ours-no-search* shows the effectiveness of the audio-based search algorithm. Although no audio guidance is used, 30% votes received by *Ours-no-search* against *GT* also suggest our video motion graph and frame blending approach is able to generate high-quality and realistic videos. The relative higher votes (41%) given to *Ours-full* against *GT* demonstrates our full system generates better though not perfect gesture videos that are coherent with the audio.

5.2.2 Video Blending Evaluation

We also numerically evaluate the proposed video blending network. To this end, we split each video in our dataset in three parts: the first part (80% of the original video duration) is used for training the network, the second and third part are used for hold-out validation and testing respectively (each has 10% of the original video duration).

Given two frames $t-k$ and $t+k$ in the test split of each video, we synthesize blended frames with the blending weight $\alpha = 0.5$ and compare its quality with the ground-truth frame t. All the compared frames are multiplied with ground-truth human masks to compare the foreground human results only.

We compare our method with the state-of-the-art frame interpolation methods **FeatureFlow**[58] and **SuperSlMo**[70], as well as human pose-based image synthesis methods **vUnet**[44] and **EBDance**[26] which uses a Pix2PixHD backbone [135]. For the pose-based image synthesis methods, we estimate the interpolated human skeleton by averaging joint positions. We retrain all the comparison methods on our dataset for a fair comparison. We also evaluate two network alternatives: **Ours w/ mesh** which only uses mesh-based warping flows and **Ours w/ optical** which only uses optical flows.

Image Quality. We evaluate the quality of synthesized images via four common metrics: Image Error (IE) - average absolute pixel difference between two images; Peak Signal-to-Noise Ratio (PSNR) and LPIPS [150].

Table 5.1 shows the comparison for methods mentioned above. It shows our full model consistently outperforms all frame interpolation methods [58, 70] and pose-based image synthesis methods [44, 26]. Fig. 5.8 shows two examples of synthesized

Method	IE↓	PSNR↑	LPIPS↓
FeatureFlow[58]	1.18	33.5	0.015
SuperSlMo[70]	1.04	35.0	0.012
vUnet[44]	1.20	33.6	0.013
EBDance[26]	1.75	30.7	0.020
Ours w/ mesh	0.87	35.2	0.009
Ours w/ optical	0.97	34.6	0.009
Ours	**0.76**	**36.1**	**0.007**

Table 5.1: Image quality assessment of video blending results.

Method	MOVIE[110] $\times 10^{-2}$			FID↓
	S↓	T↓	S-T↓	
FeatureFlow[58]	2.17	0.97	0.22	19.1
SuperSlMo[70]	1.84	0.84	0.17	15.4
vUnet[44]	2.06	0.78	0.19	15.6
EBDance[26]	3.04	1.9	0.43	20.5
Ours w/ mesh	1.80	0.63	0.14	15.1
Ours w/ optical	1.84	0.74	0.16	13.2
Ours	**1.67**	**0.68**	**0.13**	**13.0**

Table 5.2: Video quality assessment of video blending results.

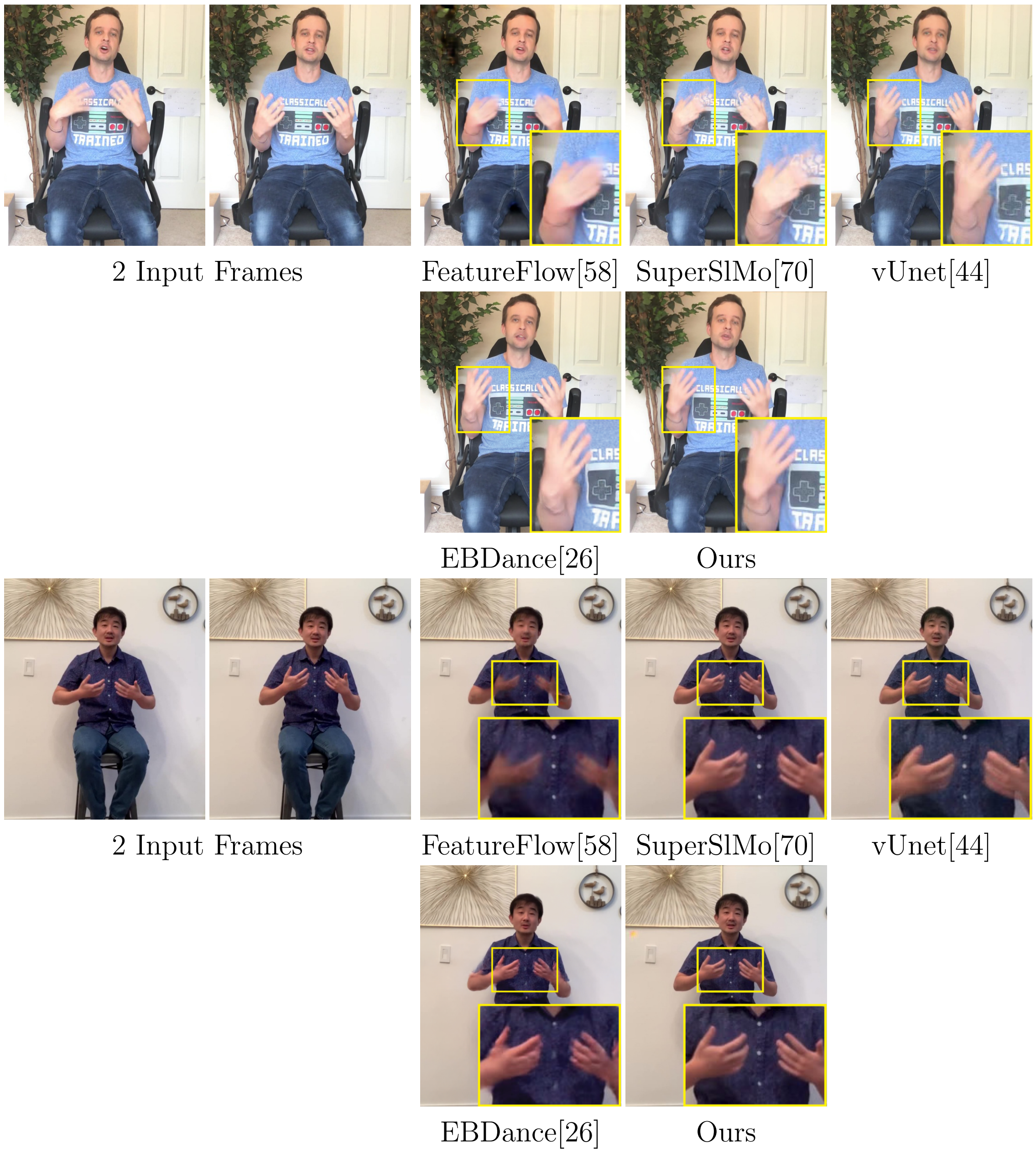


Figure 5.8: Comparison of blended frame synthesis using different methods. Note the natural look of details such as fingers in our method.

frames by ours and other methods. In the top example, the inputs are two frames with larger gesture difference. The existing frame interpolation methods [58, 70] cannot estimate the flow field, and thus result in broken and blurred hand results. The pose-based image synthesis methods [44, 26] preserve hand structures but have artifacts around fingers and clothing. Ours achieves the best quality for both hands and clothing. The lower example shows frames with smaller gesture differences. [70, 44, 26] preserve hands better but still suffer from broken and blurred texture. Ours generates clear and sharp finger results.

Video Quality. To evaluate the quality of the generated video, we adopt the metric, MOVIE [110] index, to evaluate the distortion in spatial (S), temporal (T), and spatio-temporal (S-T) aspects. We also follow [134] to evaluate the visual quality of the video and temporal consistency with Fréchet Inception Distance (FID) scores [62]. We use the pre-trained video recognition CNN model to get features from synthesized video clips [20]. Table 5.2 shows our method can achieve the best video quality in the temporal domain. It demonstrates that the synthesized blended frames seamlessly connect reenacted frames with much less temporal artifacts. We provide additional synthesized clips to showcase blending results in the supplementary video.

5.3 Conclusion

We propose a novel system based on video motion graphs to generate new videos that best preserve high image synthesis quality and speaker gesture motion subtleties. To seamlessly reenact disjoint frames from the input video, we introduce a neural pose-aware video blending method to smoothly blend inconsistent transition frames. We show the superior performance of the proposed system comparing to the state-of-the-art methods and baselines via both numerical experiments and perceptual user studies.

CHAPTER 6
CONCLUSION

This thesis explored three different neural network architectures for generating various audio-driven character animations, ranging from *character facial animations* of FACS-based face rigs and cartoon/natural human face images to *character gesture animations* of human speech videos.

In Chapter 3 and 4, I presented two different approaches to create character facial animation based on a single speech audio clip as input. Both approaches learnt a cross-modality mapping from audio signals to talking-head visual animations. Chapter 3 specially introduced speech motion curves of professional face rigs as the output representation, which are commonly used by character animators. The solution of the proposed approach can be seamlessly integrated into existing facial animation practice in film and game studios, leaving animators free to focus on the creative and nuanced aspects of character expression. Chapter 4 presented an audio-driven facial animation solution for a more generic character representation: portrait images, which are easily accessed by most users. By taking as input either natural human portraits or 2D cartoon character faces, the proposed approach is able to synthesize corresponding talking-head animation videos based on the input speech audio. The key insight is to animate faces from disentangled audio content and speaker identity, such that they capture better lip synchronization, personalized facial expressions and head motion dynamics. This leads to much more expressive animations with higher overall quality compared to the state-of-the-art. I believe these two solutions are significant steps towards automatic character talking-head animation.

Apart from facial animation, human speech is often accompanied by body gestures including arm and hand gestures, which enhances the expressiveness of human performance and helps the audience to better comprehend the speech content. Therefore, Chapter 5 focused on speech gestures and presented a novel system to generate character speech videos with plausible gestures while talking. Instead of synthesizing videos from scratch, the proposed system was built upon splitting and re-assembling clips from a reference video through a novel video motion graph, which is able to preserve high image synthesis resolution and speaker gesture motion subtleties. To seamlessly reenact disjoint frames from the input video, I introduced a neural pose-aware video blending method to smoothly blend inconsistent transition frames which would otherwise suffer from temporal inconsistency. The better performance of the proposed system is demonstrated by comparing to the state-of-the-art methods and baselines via both numerical experiments and perceptual user studies.

6.1 Future Work

6.1.1 Audio-driven Character Facial Animation

In Chapters 3 and 4, we investigated the use of facial landmarks as a key intermediate representation to help generating the final animation. Landmarks capture the shape outlines of animated lips, jaw, and other key facial parts, and can be trained from massive 2D audiovisual datasets available online. Despite its advantages, the intermediate facial landmark representation, can still not capture all fine nuances of facial animations. Expanding the facial landmark representations to more fine-grained facial representations can be a plausible future work direction to improve the quality and expressiveness of the photorealistic video synthesis. I suspect that the new representations would be useful for animating heads based on a richer set of utterances, e.g., laughter, singing, and so on.

In addition, although the method proposed in the second part of my thesis captures aspects of the speaker's style e.g., predicting head swings reflecting active speech, there are several other factors that can influence head motion dynamics. For example, the speaker's mood can also play a significant role in determining head motion and facial expressions. Further incorporating sentiment analysis into the animation pipeline to make the animated character more vivid and expressive is another promising research direction.

The synthetic image generation and manipulation, especially the "deepfake videos" synthesized via deep neural networks, have raised significant concerns about their potential misuse. A promising and urgent future work direction is digital media forensics, especially the detection of deepfake videos involving facial or lip region manipulation. Forgery videos can be identified by detecting synthesized pixel artifacts on each frame around the manipulated regions, e.g. blurred teeth, double chins [105, 125, 1]. Based on the work of this thesis, personal facial expression dynamics could also be considered i.e., synthesized deepfake talking-head videos may have inconsistent speaker-specific talking styles. Analyzing high level facial motion kinematics, e.g. lip motions, facial expression motions, may be useful to detect forgeries in the temporal space.

6.1.2 Audio-driven Character Gesture Animation

In Chapter 5, we use a pre-defined common keyword dictionary for the keyword features, which may fail when an invididual uses words outside that vocabulary. Using richer audio features learnt through data might help with more accurate gesture matching and might also remove the dependency on having an input transcript. There is an inevitable trade-off between the quality and the variety of synthesized animations – increasing the graph edge density can increase the transitions variety, yet may retrieve frames that are harder to blend. Even though the result from Chapter 5 is not as realistic as the ground truth, it is significantly better than all existing approaches. I believe a hybrid framework of video motion graph and neural reenactment is a promising future work direction for high-quality controllable digital human animations.

APPENDIX A
MAKEITTALK SUPPLEMENTARY MATERIALS

In this section we describe the supplementary materials including the details of neural network architectures used in Chapter. 4.

A.1 Speaker-Aware Animation network

The attention network follows the encoder block structure in Vaswani *et al.* [126]. It is composed of a stack of $N = 2$ identical layers. Each layer has (a) a multi-head self-attention mechanism with $N_{head} = 2$ heads and dimensionality $d_{model} = 32$ and (b) a fully connected feed-forward layer whose size is the same to the input size. We also use the embedding layer (a one-layer MLP with hidden size 32) and the position encoder layer as mentioned in [126].

The discriminator network has a similar network architecture to the attention network. The difference is that the discriminator also includes a three-layer MLP network which has 512, 256, 1 hidden size respectively.

A.2 Image-to-image translation network

The layers of the network architecture used to generate photorealistic images are listed in Table A.1. In this table, the left column indicates the spatial resolution of the feature map output. **ResBlock down** means a 2-strided convolutional layer with 3×3 kernel followed by two residual blocks, **ResBlock up** means a nearest-neighbor upsampling with a scale of 2, followed by a 3×3 convolutional layer and then two

	Landmark Representation $\mathbf{Y}_t$
256×256	Input Image $\mathbf{Q}$
128×128	ResBlock down $(3 + 3) \rightarrow 64$
64×64	ResBlock down $64 \rightarrow 128$
32×32	ResBlock down $128 \rightarrow 256$
16×16	ResBlock down $256 \rightarrow 512$
8×8	ResBlock down $512 \rightarrow 512$
4×4	ResBlock down $512 \rightarrow 512$
8×8	ResBlock up $512 \rightarrow 512$
16×16	Skip + ResBlock up $(512 + 512) \rightarrow 512$
32×32	Skip + ResBlock up $(512 + 512) \rightarrow 256$
64×64	Skip + ResBlock up $(256 + 256) \rightarrow 128$
128×128	Skip + ResBlock up $(128 + 128) \rightarrow 64$
256×256	Skip + ResBlock up $(64 + 64) \rightarrow 3$
256×256	Tanh

Table A.1: Generator architecture for synthesizing photorealistic images.

residual blocks, and **Skip** means a skip connection that concatenates the feature maps of an encoding layer and decoding layer with the same spatial resolution.

APPENDIX B

NEURAL GESTURE REENACTMENT SUPPLEMENTARY MATERIALS

In this section we describe the supplementary materials for Chapter. 5.

B.1 Dictionary of Common Keywords

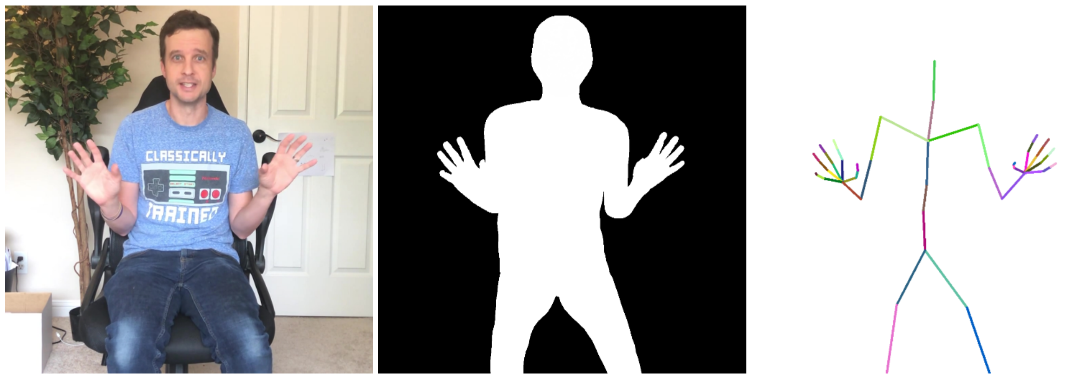
Referential gestures, especially iconic and metaphoric gestures, have strong correlations with the transcript [93, 146]. They usually appear together with certain keywords, such as action verbs, concrete objects, abstract concepts, and relative quantities to co-express the speech content [65]. We gather a few frequently used such keywords co-occurring with referential gestures in our speaker videos, as shown in Table. B.1.

B.2 Spatial Encoder Details

The spatial encoder network E_s takes as input the RGB image I_i, the foreground mask I_{mask}, and an image containing the rendered skeleton I_{skel} representing the SMPL pose parameters. Fig. B.1 shows an example of these input images. The spatial encoder network consists of 8 residual blocks [17] with no up- or down-sampling. Table B.2 shows the blocks in detail. The left column of the table indicates the spatial resolution of the feature map. The numbers before and after $\rightarrow$ in the right column are the numbers of input and output channels, respectively.

Category	Keywords
greeting	hey, hi, hello
counting	one, two, three, first, second, third
direction	east, west, north, south, back, front, away, here, around
sentiment	crazy, incredible, surprising, screaming
action	walk, drive, ride, enter, open, attach, take, move
relative	more, less, much, few
others	called

Table B.1: Dictionary of common keywords.



(a) (b) (c)

Figure B.1: An example of inputs to our spatial encoder network (a) input image frame, (b) corresponding foreground human mask and (c) rendered skeleton image.

B.3 Mesh Flow Estimator Details

We show our *Mesh Flow Estimator* network structure for generating the mesh flow warping field in Table B.3. In this table, the left column indicates the spatial resolution of the feature map output. The *ResBlock down* block is a 2-strided convolutional layer with a 3×3 kernel followed by two residual blocks. The *ResBlock up* block is a nearest-neighbor upsampling with a scale of 2, followed by a a 3×3 convolutional layer and then two residual blocks. The term *Skip* means skip connection that concatenates the feature maps of an encoding layer and decoding layer with the same spatial resolution.

512×512	Input image I_i Foreground body mesh mask I_{mask} Rendered skeleton image I_{skel}
512×512	ResBlock $(3+1+3) \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$
512×512	ResBlock $16 \rightarrow 16$

Table B.2: Spatial Encoder network structure.

B.4 Image Generator Network Details

The image generator network follows the structure of the *Mesh Flow Estimator* network (see Table B.3). There are three differences. The first difference is that the number of input feature channels is 16 (instead of 18), which represents the number of deep feature x'' channels. The second difference is that the number of output channels is changed from 2 to 3 for generating RGB images. Finally, the network uses in the end a $tanh(\cdot)$ activation to regularize the image values between $[0, 1]$.

B.5 User Study Details

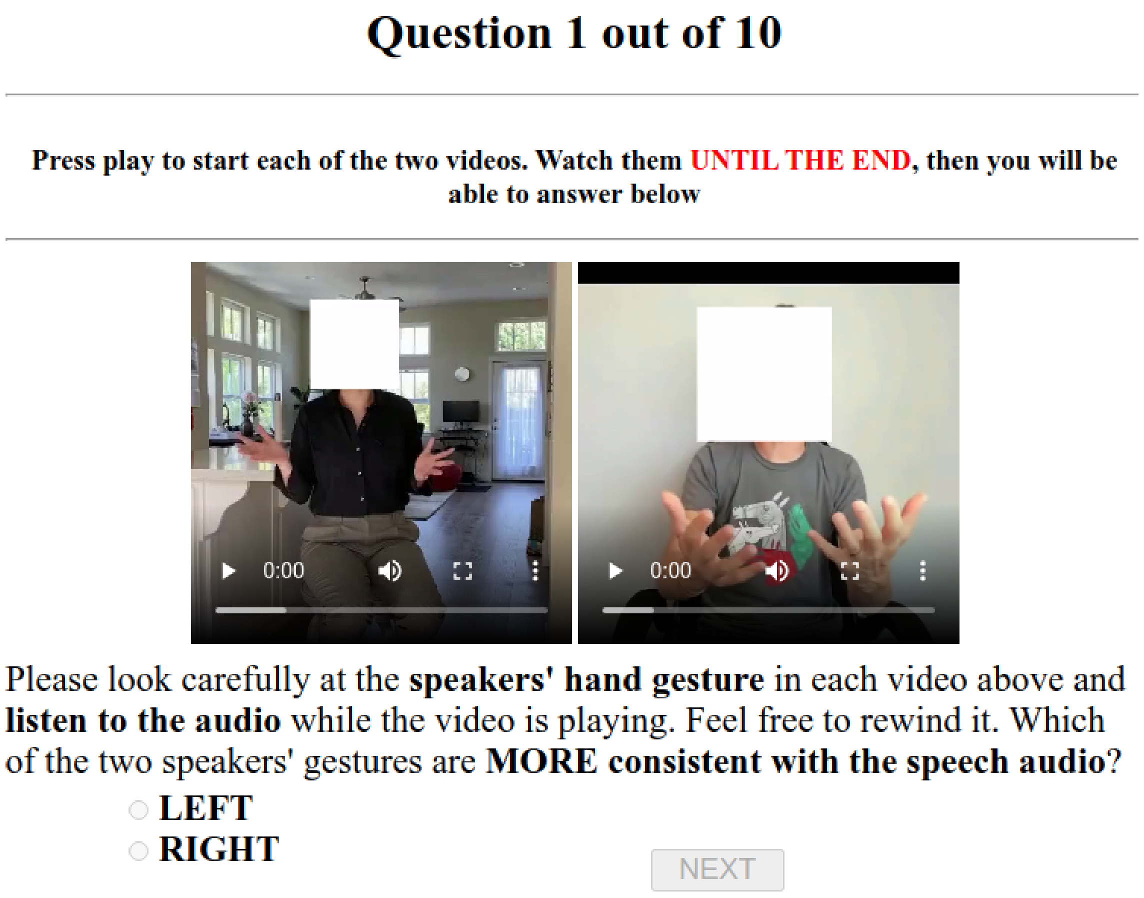
We provide here more details about the user study. Fig. B.2 shows the webpage layout used in our questionnaires. The layout shows two video results to the participants, a question on the bottom and two choices ("left"/"right"). To enable the selection of either choice, the users must watch both videos until the end. Our questionnaires also include a similar page layout showing tutorial examples in the beginning. The tutorial shows a pair of videos with clear differences: one video is from ground-truth in which the speaker's gestures are naturally consistent with the audio; the other video is a failure case, which shows gestures that are inconsistent

512×512	Input image deep feature x'_i Input initial mesh flow $F^{init}_{t \to i}$
256×256	ResBlock down $(16 + 2) \to 32$
128×128	ResBlock down $32 \to 64$
64×64	ResBlock down $64 \to 128$
32×32	ResBlock down $128 \to 256$
16×16	ResBlock down $256 \to 512$
8×8	ResBlock down $512 \to 512$
4×4	ResBlock down $512 \to 512$
8×8	ResBlock up $512 \to 512$
16×16	Skip + ResBlock up $(512 + 512) \to 512$
32×32	Skip + ResBlock up $(512 + 512) \to 256$
64×64	Skip + ResBlock up $(256 + 256) \to 128$
128×128	Skip + ResBlock up $(128 + 128) \to 64$
256×256	Skip + ResBlock up $(64 + 64) \to 32$
512×512	Skip + ResBlock up $(32 + 32) \to 2$

Table B.3: Mesh Flow Estimator network structure.

with audio at some places. For these tutorial cases only, we let the participants pick an answer first and then let them know whether their answer is correct or wrong and explain why.

We also adapt a user validation check to filter out unreliable MTurkers. Specifically, after the tutorial, our questionnaires showed 10 queries in a random order. 3 of the queries were repeated twice (i.e., we had 7 unique queries per questionnaire). We randomly flipped the two videos each time to detect unreliable participants giving inconsistent answers. We filter out unreliable MTurk participants who give different answers to two (or more) of the repeated queries in the questionnaire or took less than 5 minutes to complete it. Each participant was allowed to answer one questionnaire maximum to ensure participant diversity. We collected answers from 113 reliable participants for our user study.



Question 1 out of 10

Press play to start each of the two videos. Watch them UNTIL THE END, then you will be able to answer below

Please look carefully at the **speakers' hand gesture** in each video above and **listen to the audio** while the video is playing. Feel free to rewind it. Which of the two speakers' gestures are **MORE consistent with the speech audio**?

- ○ **LEFT**
- ○ **RIGHT**

NEXT

Figure B.2: User study questionnaire page.

BIBLIOGRAPHY

[1] Agarwal, Shruti, Farid, Hany, Gu, Yuming, He, Mingming, Nagano, Koki, and Li, Hao. Protecting world leaders against deep fakes. In *Proc. CVPRW* (2019).

[2] Agarwala, Aseem, Zheng, Ke Colin, Pal, Chris, Agrawala, Maneesh, Cohen, Michael, Curless, Brian, Salesin, David, and Szeliski, Richard. Panoramic video textures. In *ACM Trans. on Graphics (TOG)*. 2005.

[3] Ahuja, Chaitanya, Lee, Dong Won, Nakano, Yukiko I, and Morency, Louis-Philippe. Style transfer for co-speech gesture animation: A multi-speaker conditional-mixture approach. In *Proc. ECCV* (2020).

[4] Alexanderson, Simon, Henter, Gustav Eje, Kucherenko, Taras, and Beskow, Jonas. Style-controllable speech-driven gesture synthesis using normalising flows. In *Computer Graphics Forum* (2020).

[5] Anderson, Robert, Stenger, Björn, Wan, Vincent, and Cipolla, Roberto. Expressive Visual Text-to-Speech Using Active Appearance Models. In *Proc. CVPR* (2013).

[6] Arikan, Okan, and Forsyth, David A. Interactive motion generation from examples. *ACM Trans. on Graphics (TOG)* (2002).

[7] Bailly, Gérard. Learning to speak. sensori-motor control of speech movements. *Speech Communication* (1997).

[8] Bailly, Gérard, Perrier, Pascal, and Vatikiotis-Bateson, Eric. *Audiovisual Speech Processing*. Cambridge University Press, 2012.

[9] Baker, Simon, Scharstein, Daniel, Lewis, JP, Roth, Stefan, Black, Michael J, and Szeliski, Richard. A database and evaluation methodology for optical flow. *IJCV* (2011).

[10] Beaudoin, Philippe, Coros, Stelian, van de Panne, Michiel, and Poulin, Pierre. Motion-motif graphs. In *Proc. ACM SCA* (2008).

[11] Beeler, Thabo, and Bradley, Derek. Rigid stabilization of facial expressions. *ACM Trans. Graph.* (2014).

[12] Bello, Juan Pablo, Daudet, Laurent, Abdallah, Samer, Duxbury, Chris, Davies, Mike, and Sandler, Mark B. A tutorial on onset detection in music signals. *IEEE Trans on Speech and Audio Processing* (2005).

[13] Bergmann, Kirsten, and Kopp, Stefan. Gnetic–using bayesian decision networks for iconic gesture generation. In *International Workshop on Intelligent Virtual Agents* (2009).

[14] Bozkurt, Elif, Yemez, Yücel, and Erzin, Engin. Multimodal analysis of speech and arm motion for prosody-driven synthesis of beat gestures. *Speech Communication* (2016).

[15] Brand, Matthew. Voice puppetry. In *Proc. SIGGRAPH* (1999).

[16] Bregler, Christoph, Covell, Michele, and Slaney, Malcolm. Video rewrite: Driving visual speech with audio. In *Proc. SIGGRAPH* (1997).

[17] Brock, Andrew, Donahue, Jeff, and Simonyan, Karen. Large scale gan training for high fidelity natural image synthesis. In *Proc. ICLR* (2018).

[18] Bulat, Adrian, and Tzimiropoulos, Georgios. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *Proc. ICCV* (2017).

[19] Busso, Carlos, Lee, Sungbok, and Narayanan, Shrikanth S. Using neutral speech models for emotional speech analysis. In *Proc. InterSpeech* (2007).

[20] Carreira, Joao, and Zisserman, Andrew. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. CVPR* (2017).

[21] Caruana, Rich. Multi-task learning. *Machine Learning* (1997).

[22] Casas, Dan, Richardt, Christian, Collomosse, John, Theobalt, Christian, and Hilton, Adrian. 4d model flow: Precomputed appearance alignment for real-time 4d video interpolation. In *Computer Graphics Forum* (2015).

[23] Casas, Dan, Volino, Marco, Collomosse, John, and Hilton, Adrian. 4d video textures for interactive character appearance. In *Computer Graphics Forum* (2014).

[24] Casiez, Géry, Roussel, Nicolas, and Vogel, Daniel. 1 € filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proc. SIGCHI on Human Factors in Computing Systems* (2012).

[25] Cassell, Justine, Stone, Matthew, and Yan, Hao. Coordination and context-dependence in the generation of embodied conversation. In *Proc. International Conference on Natural Language Generation* (2000).

[26] Chan, Caroline, Ginosar, Shiry, Zhou, Tinghui, and Efros, Alexei A. Everybody dance now. In *Proc. ICCV* (2019).

[27] Chen, Lele, Li, Zhiheng, K Maddox, Ross, Duan, Zhiyao, and Xu, Chenliang. Lip movements generation at a glance. In *Proc. ECCV* (2018).

[28] Chen, Lele, Maddox, Ross K, Duan, Zhiyao, and Xu, Chenliang. Hierarchical cross-modal talking face generation with dynamic pixel-wise loss. In *Proc. CVPR* (2019).

[29] Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. EMNLP* (2014).

[30] Chung, Joon Son, Jamaludin, Amir, and Zisserman, Andrew. You said that? In *Proc. BMVC* (2017).

[31] Chung, Joon Son, Nagrani, Arsha, and Zisserman, Andrew. Voxceleb2: Deep speaker recognition. In *Proc. INTERSPEECH* (2018).

[32] Cohen, Michael M, and Massaro, Dominic W. Modeling Coarticulation in Synthetic Visual Speech. *Models and Techniques in Computer Animation* (1993).

[33] Cooke, Martin, Barker, Jon, Cunningham, Stuart, and Shao, Xu. An audio-visual corpus for speech perception and automatic speech recognition. *J. Acoustical Society of America* (2006).

[34] Cudeiro, Daniel, Bolkart, Timo, Laidlaw, Cassidy, Ranjan, Anurag, and Black, Michael J. Capture, learning, and synthesis of 3d speaking styles. In *Proc. CVPR* (2019).

[35] Davis, Abe, and Agrawala, Maneesh. Visual rhythm and beat. *ACM Trans. Graph.* (2018).

[36] Davis, Steven B., and Mermelstein, Paul. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoustics, Speech and Signal Processing* (1980).

[37] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805* (2018).

[38] Driskell, James E, and Radtke, Paul H. The effect of gesture on speech production and comprehension. *Human factors* (2003).

[39] Edwards, Pif, Landreth, Chris, Fiume, Eugene, and Singh, Karan. Jali: an animator-centric viseme model for expressive lip synchronization. *ACM Trans. Graph.* (2016).

[40] Ekman, Paul, and Friesen, Wallace V. Facial action coding system: a technique for the measurement of facial movement.

[41] Elhoseiny, Mohamed, Cohen, Scott, Chang, Walter, Price, Brian, and Elgammal, Ahmed. Sherlock: Scalable fact learning in images. In *Proc. AAAI* (2017).

[42] Eskimez, Sefik Emre, Maddox, Ross K, Xu, Chenliang, and Duan, Zhiyao. Generating talking face landmarks from speech. In *Proc. LVA/ICA* (2018).

[43] Eskimez, Sefik Emre, Maddox, Ross K, Xu, Chenliang, and Duan, Zhiyao. Noise-resilient training method for face landmark generation from speech. *IEEE/ACM Trans. Audio, Speech, and Language Processing* (2019).

[44] Esser, Patrick, Sutter, Ekaterina, and Ommer, Björn. A variational u-net for conditional appearance and shape generation. In *Proc. CVPR* (2018).

[45] Ezzat, Tony, Geiger, Gadi, and Poggio, Tomaso. Trainable videorealistic speech animation. In *Proc. SIGGRAPH* (2002).

[46] Faceware. Analyzer. http://facewaretech.com/products/software/analyzer, 2017.

[47] Faigin, Gary. *The artist's complete guide to facial expression*. Watson-Guptill, 2012.

[48] Fanelli, G., Gall, J., Romsdorfer, H., Weise, T., and Gool, L. Van. A 3-d audio-visual corpus of affective communication. *IEEE Trans. Multimedia* (2010).

[49] Fisher, Cletus G. Confusions among visually perceived consonants. *J. Speech, Language, and Hearing Research 11*, 4 (1968).

[50] Flagg, Matthew, Nakazawa, Atsushi, Zhang, Qiushuang, Kang, Sing Bing, Ryu, Young Kee, Essa, Irfan, and Rehg, James M. Human video textures. In *Proc. Symposium on Interactive 3D Graphics and Games* (2009).

[51] Fugate, Jennifer MB. Categorical perception for emotional faces. *Emotion Review* (2013).

[52] Garrido, Pablo, Valgaerts, Levi, Sarmadi, Hamid, Steiner, Ingmar, Varanasi, Kiran, Perez, Patrick, and Theobalt, Christian. Vdub: Modifying face video of actors for plausible visual alignment to a dubbed audio track. In *Computer graphics forum* (2015).

[53] Ginosar, Shiry, Bar, Amir, Kohavi, Gefen, Chan, Caroline, Owens, Andrew, and Malik, Jitendra. Learning individual styles of conversational gesture. In *Proc. CVPR* (2019).

[54] Google. Google cloud voice. https://cloud.google.com/speech, 2017.

[55] Graves, Alex, and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *Proc. ICML* (2014).

[56] Graves, Alex, and Schmidhuber, Jürgen. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* (2005).

[57] Greenwood, David, Matthews, Iain, and Laycock, Stephen. Joint learning of facial expression and head pose from speech. Interspeech.

[58] Gui, Shurui, Wang, Chaoyue, Chen, Qihua, and Tao, Dacheng. Featureflow: robust video interpolation via structure-to-texture generation. In *Proc. CVPR* (2020).

[59] Han, Xintong, Wu, Zuxuan, Huang, Weilin, Scott, Matthew R, and Davis, Larry S. Finet: Compatible and diverse fashion image inpainting. In *Proc. ICCV* (2019).

[60] Haq, S., and Jackson, P.J.B. Speaker-dependent audio-visual emotion recognition. In *Proc. AVSP* (2009).

[61] Heck, Rachel, and Gleicher, Michael. Parametric motion graphs. In *Proc. Symposium on Interactive 3D Graphics and Games* (2007).

[62] Heusel, Martin, Ramsauer, Hubert, Unterthiner, Thomas, Nessler, Bernhard, and Hochreiter, Sepp. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proc. NeurIPS* (2017).

[63] Hochreiter, Sepp, and Schmidhuber, Jürgen. Long short-term memory. *Neural computation* (1997).

[64] Hu, Liwen, Saito, Shunsuke, Wei, Lingyu, Nagano, Koki, Seo, Jaewoo, Fursund, Jens, Sadeghi, Iman, Sun, Carrie, Chen, Yen-Chun, and Li, Hao. Avatar digitization from a single image for real-time rendering. *ACM Trans. Graph.* (2017).

[65] Huang, Chien-Ming, and Mutlu, Bilge. Modeling and evaluating narrative gestures for humanlike robots. In *Robotics: Science and Systems* (2013).

[66] Huang, Peng, Tejera, Margara, Collomosse, John, and Hilton, Adrian. Hybrid skeletal-surface motion graphs for character animation from 4d performance capture. *ACM Trans. on Graphics (TOG)* (2015).

[67] Ilg, Eddy, Mayer, Nikolaus, Saikia, Tonmoy, Keuper, Margret, Dosovitskiy, Alexey, and Brox, Thomas. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. CVPR* (2017).

[68] Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR* (2017).

[69] Iverson, Jana M, and Goldin-Meadow, Susan. Why people gesture when they speak. *Nature* (1998).

[70] Jiang, Huaizu, Sun, Deqing, Jampani, Varun, Yang, Ming-Hsuan, Learned-Miller, Erik, and Kautz, Jan. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proc. CVPR* (2018).

[71] Johnson, Justin, Alahi, Alexandre, and Fei-Fei, Li. Perceptual losses for real-time style transfer and super-resolution. In *Proc. ECCV* (2016).

[72] Karras, Tero, Aila, Timo, Laine, Samuli, Herva, Antti, and Lehtinen, Jaakko. Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Trans. Graph.* (2017).

[73] Karras, Tero, Laine, Samuli, and Aila, Timo. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[74] Kim, Hyeongwoo, Elgharib, Mohamed, Zollhöfer, Michael, Seidel, Hans-Peter, Beeler, Thabo, Richardt, Christian, and Theobalt, Christian. Neural style-preserving visual dubbing. *ACM Trans. Graph.* (2019).

[75] King, Davis E. Dlib-ml: A machine learning toolkit. *J. of Machine Learning Research* (2009).

[76] Kovar, Lucas, Gleicher, Michael, and Pighin, Frédéric. Motion graphs. In *ACM Trans. on Graphics (TOG)*. 2008.

[77] Krüger, Björn, Tautges, Jochen, Weber, Andreas, and Zinke, Arno. Fast local and global similarity searches in large motion capture databases. In *Proc. ACM SCA* (2010).

[78] Kucherenko, Taras, Jonell, Patrik, van Waveren, Sanne, Henter, Gustav Eje, Alexandersson, Simon, Leite, Iolanda, and Kjellström, Hedvig. Gesticulator: A framework for semantically-aware speech-driven gesture generation. In *Proc. ICMI* (2020).

[79] Lee, Yongjoon, Wampler, Kevin, Bernstein, Gilbert, Popović, Jovan, and Popović, Zoran. Motion fields for interactive character locomotion. In *ACM Trans. on Graphics (TOG)*. 2010.

[80] Li, Hao, Yu, Jihun, Ye, Yuting, and Bregler, Chris. Realtime Facial Animation with on-the-Fly Correctives. *ACM Trans. Graph.* (2013).

[81] Li, Kun, Yang, Jingyu, Liu, Leijie, Boulic, Ronan, Lai, Yu-Kun, Liu, Yebin, Li, Yubin, and Molla, Eray. Spa: Sparse photorealistic animation using a single rgb-d camera. *IEEE Trans. on CSVT* (2016).

[82] Liao, Miao, Zhang, Sibo, Wang, Peng, Zhu, Hao, Zuo, Xinxin, and Yang, Ruigang. Speech2video synthesis with 3d skeleton regularization and expressive body poses. In *Proc. ACCV* (2020).

[83] Liberman, Alvin M, Harris, Katherine Safford, Hoffman, Howard S, and Griffith, Belver C. The discrimination of speech sounds within and across phoneme boundaries. *J. Experimental Psychology* (1957).

[84] Liu, Lingjie, Xu, Weipeng, Zollhoefer, Michael, Kim, Hyeongwoo, Bernard, Florian, Habermann, Marc, Wang, Wenping, and Theobalt, Christian. Neural rendering and reenactment of human actor videos. *ACM Trans on Graphics (TOG)* (2019).

[85] Liu, Wen, Piao, Zhixin, Min, Jie, Luo, Wenhan, Ma, Lin, and Gao, Shenghua. Liquid warping gan: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *Proc. ICCV* (2019).

[86] Liu, Ziwei, Yeh, Raymond A, Tang, Xiaoou, Liu, Yiming, and Agarwala, Aseem. Video frame synthesis using deep voxel flow. In *Proc. ICCV* (2017).

[87] Loper, Matthew, Mahmood, Naureen, Romero, Javier, Pons-Moll, Gerard, and Black, Michael J. Smpl: A skinned multi-person linear model. *ACM Trans. on Graphics (TOG)* (2015).

[88] Maaten, Laurens van der, and Hinton, Geoffrey. Visualizing data using t-sne. *JMLR* (2008).

[89] MacDorman, Karl F., Green, Robert D., Ho, Chin-Chang, and Koch, Clinton T. Too real for comfort? uncanny responses to computer generated faces. *Computers in Human Behavior* (2009).

[90] Mao, Xudong, Li, Qing, Xie, Haoran, Lau, Raymond YK, Wang, Zhen, and Paul Smolley, Stephen. Least squares generative adversarial networks. In *Proc. ICCV* (2017).

[91] McAuliffe, Michael, Socolof, Michaela, Mihuc, Sarah, Wagner, Michael, and Sonderegger, Morgan. Montreal forced aligner: trainable text-speech alignment using kaldi. In *Proc. Interspeech* (2017).

[92] McGurk, Harry, and MacDonald, John. Hearing lips and seeing voices. *Nature* (1976).

[93] McNeill, David. *Hand and mind: What gestures reveal about thought*. University of Chicago press, 1992.

[94] Min, Jianyuan, and Chai, Jinxiang. Motion graphs++ a compact generative model for semantic motion analysis and synthesis. *ACM Trans. on Graphics (TOG)* (2012).

[95] Niklaus, Simon, and Liu, Feng. Softmax splatting for video frame interpolation. In *Proc. CVPR* (2020).

[96] Niklaus, Simon, Mai, Long, and Liu, Feng. Video frame interpolation via adaptive separable convolution. In *Proc. ICCV* (2017).

[97] Notevibes. Text to Speech converter. `https://notevibes.com/`, 2020.

[98] Olah, Christopher. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015.

[99] Paliwal, Kuldip K. Spectral subband centroid features for speech recognition. In *Proc. ICASSP* (1998).

[100] Pham, Hai Xuan, Wang, Yuting, and Pavlovic, Vladimir. End-to-end learning for 3d facial animation from speech. In *Proc. ICMI* (2018).

[101] Prajwal, KR, Mukhopadhyay, Rudrabha, Namboodiri, Vinay P, and Jawahar, CV. A lip sync expert is all you need for speech to lip generation in the wild. In *Proc. ACM International Conference on Multimedia* (2020).

[102] Qian, Kaizhi, Zhang, Yang, Chang, Shiyu, Yang, Xuesong, and Hasegawa-Johnson, Mark. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *Proc. ICML* (2019).

[103] Reitsma, Paul SA, and Pollard, Nancy S. Evaluating motion graphs for character animation. *ACM Trans. on Graphics (TOG)* (2007).

[104] Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI* (2015).

[105] Rossler, Andreas, Cozzolino, Davide, Verdoliva, Luisa, Riess, Christian, Thies, Justus, and Nießner, Matthias. Faceforensics++: Learning to detect manipulated facial images. In *Proc. ICCV* (2019).

[106] Rubin, Steven M, and Reddy, Raj. The locus model of search and its use in image interpretation. In *IJCAI* (1977).

[107] Safonova, Alla, and Hodgins, Jessica K. Construction and optimal search of interpolated motion graphs. *ACM Trans. on Graphics (TOG)* (2007).

[108] Schödl, Arno, Szeliski, Richard, Salesin, David H, and Essa, Irfan. Video textures. In *Proc. Conference on Computer Graphics and Interactive Techniques* (2000).

[109] Segal, Aleksandr, Haehnel, Dirk, and Thrun, Sebastian. Generalized-icp. In *Proc. Robotics: science and systems* (2009).

[110] Seshadrinathan, Kalpana, and Bovik, Alan Conrad. Motion tuned spatio-temporal quality assessment of natural videos. *IEEE Trans. Image Processing* (2009).

[111] Shin, Hyun Joon, and Oh, Hyun Seok. Fat graphs: constructing an interactive character with continuous controls. In *Proc. ACM SCA* (2006).

[112] Shum, H-Y, and Szeliski, Richard. Construction of panoramic image mosaics with global and local alignment. In *Panoramic vision*. 2001.

[113] Shysheya, Aliaksandra, Zakharov, Egor, Aliev, Kara-Ali, Bashirov, Renat, Burkov, Egor, Iskakov, Karim, Ivakhnenko, Aleksei, Malkov, Yury, Pasechnik, Igor, Ulyanov, Dmitry, et al. Textured neural avatars. In *Proc. CVPR* (2019).

[114] Siarohin, Aliaksandr, Lathuilière, Stéphane, Tulyakov, Sergey, Ricci, Elisa, and Sebe, Nicu. First order motion model for image animation. In *Proc. NeurIPS* (2019).

[115] Simonyan, Karen, and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR* (2014).

[116] Song, Yang, Zhu, Jingwen, Li, Dawei, Wang, Andy, and Qi, Hairong. Talking face generation by conditional recurrent adversarial network. In *Proc. IJCAI* (2019).

[117] Sorkine, Olga. Differential representations for mesh processing. In *Computer Graphics Forum* (2006).

[118] Stylianou, Yannis. Voice transformation: a survey. In *Proc. ICASSP* (2009).

[119] Suwajanakorn, Supasorn, Seitz, Steven M, and Kemelmacher-Shlizerman, Ira. Synthesizing obama: learning lip sync from audio. *ACM Trans. Graph.* (2017).

[120] Taylor, Sarah, Kim, Taehwan, Yue, Yisong, Mahler, Moshe, Krahe, James, Rodriguez, Anastasio Garcia, Hodgins, Jessica, and Matthews, Iain. A deep learning approach for generalized speech animation. *ACM Trans. Graph.* (2017).

[121] Taylor, Sarah L., Mahler, Moshe, Theobald, Barry-John, and Matthews, Iain. Dynamic units of visual speech. In *Proc. SCA* (2012).

[122] Teed, Zachary, and Deng, Jia. Raft: Recurrent all-pairs field transforms for optical flow. In *Proc. ECCV* (2020).

[123] Thies, Justus, Elgharib, Mohamed, Tewari, Ayush, Theobalt, Christian, and Nießner, Matthias. Neural voice puppetry: Audio-driven facial reenactment. In *Proc. CVPR, to appear* (2020).

[124] Thies, Justus, Zollhofer, Michael, Stamminger, Marc, Theobalt, Christian, and Nießner, Matthias. Face2face: Real-time face capture and reenactment of rgb videos. In *Proc. CVPR* (2016).

[125] Tolosana, Ruben, Vera-Rodriguez, Ruben, Fierrez, Julian, Morales, Aythami, and Ortega-Garcia, Javier. Deepfakes and beyond: A survey of face manipulation and fake detection. *Information Fusion* (2020).

[126] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. In *Proc. NeurIPS* (2017).

[127] Veaux, Christophe, Yamagishi, Junichi, MacDonald, Kirsten, et al. Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit.

[128] Vougioukas, Konstantinos, Petridis, Stavros, and Pantic, Maja. Realistic speech-driven facial animation with gans. *IJCV* (2019).

[129] Walker, Marilyn A, Cahn, Janet E, and Whittaker, Stephen J. Improvising linguistic style: Social and affective bases for agent personality. In *Proc. IAA* (1997).

[130] Wan, Li, Wang, Quan, Papir, Alan, and Moreno, Ignacio Lopez. Generalized end-to-end loss for speaker verification. In *Proc. ICASSP* (2018).

[131] Wang, Hongcheng, Raskar, Ramesh, and Ahuja, Narendra. Seamless video editing. In *Proc. ICPR* (2004).

[132] Wang, Lijuan, Han, Wei, and Soong, Frank K. High Quality Lip-Sync Animation for 3D Photo-Realistic Talking Head. In *Proc. ICASSP* (2012).

[133] Wang, Ting-Chun, Liu, Ming-Yu, Tao, Andrew, Liu, Guilin, Catanzaro, Bryan, and Kautz, Jan. Few-shot video-to-video synthesis. In *Proc. NeurIPS* (2019).

[134] Wang, Ting-Chun, Liu, Ming-Yu, Zhu, Jun-Yan, Liu, Guilin, Tao, Andrew, Kautz, Jan, and Catanzaro, Bryan. Video-to-video synthesis. In *Proc. NeurIPS* (2018).

[135] Wang, Ting-Chun, Liu, Ming-Yu, Zhu, Jun-Yan, Tao, Andrew, Kautz, Jan, and Catanzaro, Bryan. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proc CVPR* (2018).

[136] Wang, Wenwu. *Machine Audition: Principles, Algorithms and Systems: Principles, Algorithms and Systems*. IGI Global, 2010.

[137] Weise, Thibaut, Bouaziz, Sofien, Li, Hao, and Pauly, Mark. Realtime performance-based facial animation. In *ACM Trans. Graph.* (2011).

[138] Weng, Chung-Yi, Curless, Brian, and Kemelmacher-Shlizerman, Ira. Photo wake-up: 3d character animation from a single photo. In *Proc. CVPR* (2019).

[139] Weng, Chung-Yi, Curless, Brian, and Kemelmacher-Shlizerman, Ira. Vid2actor: Free-viewpoint animatable person synthesis from video in the wild. *arXiv preprint arXiv:2012.12884* (2020).

[140] Williams, Lance. Performance-driven facial animation. In *Proc. SIGGRAPH* (1990).

[141] Xiang, Donglai, Joo, Hanbyul, and Sheikh, Yaser. Monocular total capture: Posing face, body, and hands in the wild. In *Proc. CVPR* (2019).

[142] Xiong, Wayne, Wu, Lingfeng, Alleva, Fil, Droppo, Jasha, Huang, Xuedong, and Stolcke, Andreas. The microsoft 2017 conversational speech recognition system. In *Proc. ICASSP* (2018).

[143] Xu, Feng, Liu, Yebin, Stoll, Carsten, Tompkin, James, Bharaj, Gaurav, Dai, Qionghai, Seidel, Hans-Peter, Kautz, Jan, and Theobalt, Christian. Video-based characters: creating new human performances from a multi-view video database. In *ACM Trans. on Graphics (TOG)*. 2011.

[144] Yang, Yanzhe, Yang, Jimei, and Hodgins, Jessica. Statistics-based motion synthesis for social conversations. In *Computer Graphics Forum* (2020).

[145] Yaniv, Jordan, Newman, Yael, and Shamir, Ariel. The face of art: landmark detection and geometric style in portraits. *ACM Trans. Graph.* (2019).

[146] Yoon, Youngwoo, Cha, Bok, Lee, Joo-Haeng, Jang, Minsu, Lee, Jaeyeon, Kim, Jaehong, and Lee, Geehyuk. Speech gesture generation from the trimodal context of text, audio, and speaker identity. *ACM Trans. on Graphics (TOG)* (2020).

[147] Yunus, Fajrian, Clavel, Chloé, and Pelachaud, Catherine. Sequence-to-sequence predictive models: from prosody to communicative gestures. In *Workshop sur les Affects, Compagnons artificiels et Interactions* (2020).

[148] Zakharov, Egor, Shysheya, Aliaksandra, Burkov, Egor, and Lempitsky, Victor. Few-shot adversarial learning of realistic neural talking head models. In *Proc. ICCV* (2019).

[149] Zhang, Haotian, Sciutto, Cristobal, Agrawala, Maneesh, and Fatahalian, Kayvon. Vid2player: Controllable video sprites that behave and appear like professional tennis players. *arXiv preprint arXiv:2008.04524* (2020).

[150] Zhang, Richard, Isola, Phillip, Efros, Alexei A, Shechtman, Eli, and Wang, Oliver. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR* (2018).

[151] Zhou, Hang, Liu, Yu, Liu, Ziwei, Luo, Ping, and Wang, Xiaogang. Talking face generation by adversarially disentangled audio-visual representation. In *Proc. AAAI* (2019).

[152] Zhou, Yang, Li, Dingzeyu, Han, Xintong, Kalogerakis, Evangelos, Shechtman, Eli, and Echevarria, Jose. Makeittalk: Speaker-aware talking head animation. *ACM Trans. Graph* (2020).

[153] Zhou, Yang, Xu, Zhan, Landreth, Chris, Kalogerakis, Evangelos, Maji, Subhransu, and Singh, Karan. Visemenet: Audio-driven animator-centric speech animation. *ACM Trans. Graph.* (2018).