An Automatic Lip Sync system using Neural Network classification to generate unique mouth shapes based on speech type

Sanz-Robinson, Jacob, and Vybihal, Joseph

***Abstract*—Creators of multimedia content are commonly faced with the task of generating a sequence of lip movements to match an audio track of human speech, also known as a 'Lip Sync'. A challenge in automatic Lip Sync systems lies in accurately generating mouth shapes that can discern speech type, that's to say when the voice in the audio track is whispering, talking or shouting. We present a novel method that automatically generates lightweight mouth shapes from an excerpt of spoken audio by predicting the speech type.**

Phonemes in the speech input are detected using CMUSphinx, and mapped to visemes according to the Jeffers map. The audio input signal is represented using Mel-Frequency Cepstral Coefficient (MFCC) vectors that are used as features in a Feedforward Neural Network. The neural network then predicts the probability that the audio belong to three categories (whisper, talk, and shout). Mouth shapes are generated by averaging control points in pre-made mouth models according to the Neural Network's predicted probability for the speech type. The resulting mouth shapes are unique, simple to create, and constitute convincing lip sync frames. Their low storage and transmission requirements mean they are well suited for implementing low-bandwidth communication through animation.

***Index Terms*— automatic lip sync, Bezier, feedforward neural network, human voice, lip sync, mouth shape, multi-layer neural network, neural network, phoneme, speech, speech processing, speech type, viseme.**

I. Introduction

HOW much knowledge can you obtain about a person's facial expression based only on the sound of their voice? Many characteristics can be learnt from an audio recording of a person's voice, in particular relating to the shape of the mouth, which the human mind systematically associates with sounds [1]. These characteristics are of interest to multimedia content stakeholders, as the information they contain can assist in generating realistic mouths and facial parts for speech-driven synthetic talking faces (avatars). For instance, hearing an angry shout may conjure the image of angled eyebrows and a wide-open mouth. Furthermore, cognition studies such as Sweeny et al [1] show that when humans hear sounds such as "woo" and "wee", they associate them with horizontally or vertically extended mouth shapes. Ultimately, developing bandwidth-efficient, realistic animated avatars can be used to improve computer-human interactions, finding uses in technologies such as video games and video telephony.

When generating an animation of lip movement to match an audio track of human speech, otherwise known as 'Lip Sync', animators will typically possess prior knowledge about the facial expressions of the avatar they are creating. This allows animators to manually draw an accurate mouth shape for each phoneme in the recording. In Automatic Lip Sync systems, the main focus is to accurately generate an audio to visual mapping from each phoneme (a perceptually distinct unit of sound in speech) in the input speech signal to a corresponding viseme (mouth shape), without relying on manual modification. In the case that no prior knowledge of the facial expression exists, such as real-time Lip Sync generation, the accuracy of generated mouth shapes can be enhanced by modifying the mouth shapes according to cues in the audio recording. We propose that features extracted from the audio speech signal can aid in the construction of uniquely generated mouth shapes for multimedia content.

As such, the aim of this project is to use a Feedforward Neural Network to classify human speech into different categories of speech types (whisper, talk, and shout) and use this classification in conjunction with the phoneme detection system from CMUSphinx [2] to generate a mapping with improved accuracy and detail in the generated mouth shapes. This is an improvement over the pre-made mouth shapes used in many current systems. The unique mouth shapes can be generated with ease, without the need for prior knowledge of the speaker to do so.

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

The next few paragraphs should contain the authors' current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

The final output of the pipeline is a series of timestamps and mouth shapes defined by the control points specified in the MPEG-4 facial animation standard parameters [3]. Importantly, the low data storage and transmission cost of the generated mouths would allow this technique to be used in practical scenarios such as the real-time transmission reconstruction of avatars. It also finds a use in animation studios, and could reduce the cost of producing lip syncs in multimedia productions.

II. Related Work

Creating automated Lip Sync animations from speech has been approached in several different manners, in different fields, and for different applications. Rarely, however, have these works considered the effect of the type of speech on the shape of the mouth, often focusing instead on novel phoneme recognition systems and viseme mappings. Chronologically, amongst the first of these approaches is that seen in [4], where least-squares linear prediction is used to detect phonemes in frames of recorded audio, which are then assigned to pre-made mouth shapes.

In [5], the problem of Lip Sync generation is addressed using a linear model to describe the movement of measured physical articulators of the mouth-shape, with no need for phoneme detection. These articulators control 16 points which define a Bezier patch representation of the face. There are limitations to the results, as the authors point out, such as non-linear speech motions not being well represented. It does, however, indicate success of using control points to manage the output mouth-shape, which shall be used in this project.

In [6], video footage of a speaker was used to generate new footage of the same person speaking synced to a new audio track. This was achieved by reordering the mouth images from the original footage and matching them with the new phonemes labelled through the use of a Hidden Markov Model (HMM). The use of computer vision techniques to track control points on the mouth in the original footage allowed unique speaker mannerisms to be integrated in the final Lip Sync. Another use of HMMs in the field is that of [7], where phoneme labels are not used, and rather, full-facial motion is predicted and represented as facial motion vectors based on the mannerisms of three subjects. Recently, more complex uses of HMMs have been made, such as in [8], where visual gesture articulators (which make use of complex language modelling) are clustered against a learned set, and used to generate ‘dynamic visemes’. Gaussian Mixture Models (GMM) have also been used in conjunction with HMMs, an example of which can be found in [9], where a GMM was used to predict the width and height of the outer contour of the mouth from audio, while the HMM explored a context cue to achieve better mapping performance for isolated words.

The task of phoneme recognition in speech, while widely used in automated Lip Sync generation systems in order to map phonemes to visemes, is not unique to the field. It is also a central part of numerous language processing and speech recognition systems. Recently, the highest success rate in phoneme recognition has been found in systems that use Recurrent Neural Networks (RNN), a successful early example being [10]. In these papers the use of MFCCs as features for the learning models has become a common practice. MFCCs will also be used as features in this project. Other interesting approaches include the use of a variety of Wavelet Transformations to improve the success rate of phoneme identification [11]. Packages and libraries exist with the capacity to perform phoneme identification. One such example is CMUSphynx, that contains “20 years of the CMU research” [2] relating to speech recognition, and will be used in this project.

The creation of avatars, such as the method presented in [12], is a field that overlaps significantly with the creation of realistic lip sync systems. [12] mentions that one of the main challenges faced is to ensure natural-looking articulations. The generation of facial expressions in the paper depends on a specified emotional state. Other noteworthy attempts in generating realistic and detailed mouth shapes include [13], which directly models the trajectories of articulators using a Dynamic Bayesian Network, describing the synchronization between the visual and auditory aspects of speech, and generating optimal facial parameters using Maximum Likelihood criterion on the model. As seen in [14], the generation of realistic facial expressions in avatars has progressed in the past decade since [12] and [13]. Here, an Expression Map is used to synthesize combinations of basic emotional expressions (such as 40% happiness and 60% surprise) to build more complex ones, which partially inspired the approach of merging viseme models seen in our paper.

In more recent years, Neural Networks have been used to address the problem of lip sync generation. In [15] the input speech is segmented into frames, and the most likely corresponding viseme is found using a neural network. This particular project uses a genetic algorithm to automatically configure the topology of the Network. This paper also maps the visemes to mouth shapes defined with the MPEG-4 facial animation parameters [16], a set of standard points for facial animation. The Lip Syncs in this paper will also be based on the 18 MPEG-4 FA parameters for the mouth shape.

A recent paper in the field is [17], where a Lip Sync for Barrack Obama was generated. Trained on ample footage of the former President, a time delayed RNN is used directly on the input audio to generate visemes or “sparse mouth shapes”, skipping phoneme labeling. The synthesized mouth is overlaid over existent footage to create a Lip Sync. In this method the future mouth-shape is determined by past mouth shapes, and achieves a realistic automatic Lip Sync.

Finally, [18] employs decision trees to generate a sequence of face configurations from a series of phonetic inputs divided into frames. Though the system is reliable and robust, one of the main limitations the author mentions is that of training the

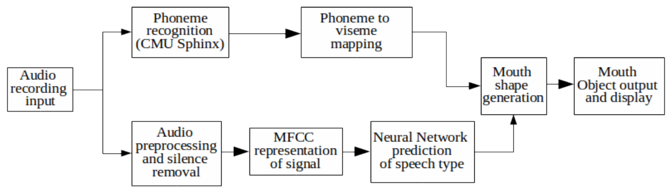

Audio recording input
Phoneme recognition (CMU Sphinx)
Phoneme to viseme mapping
Mouth shape generation
Mouth Object output and display
Audio preprocessing and silence removal
MFCC representation of signal
Neural Network prediction of speech type

Fig. 1. A flowchart illustrating the pipeline of the Lip Sync system

system to deal with different speech tones, such as anger, as opposed to a neutral voice. As such, facial responsivity to different types and tones of speech seems like an interesting and important direction to advance automatic Lip Systems in order to increase their realism in modern applications.

III. Method and System

Given the audio recording of the voice of a speaker, the objective of this project is to generate corresponding mouth shapes, reflecting the type of speech in the generated visemes.

Phoneme Detection is performed by CMUSphinx, and the classification of the input into the categories of whispering, talking, and shouting is performed by a Feedforward Neural Network. To achieve this, the Neural Network is trained with labeled clips of people speaking in each of the aforementioned categories.

The overall pipeline is built in a Jupyter Notebook and works as follows (see Figure 1): The user provides an initial audio recording, which is then passed to CMUSphinx, where the phonemes are detected, and then mapped to a viseme group. In a parallel process, the same initial source audio is broken into chunks, and each chunk has all the silences removed from it. Audio features (MFCCs) are extracted from the resulting dense signal chunks, and fed to the Neural Network. Based on the probabilities the Neural Network predicts for each chunk belonging to a certain speech category, and the corresponding viseme, a unique sparse mouth shape is constructed, and returned as an object. A mouth object is created for every phoneme. These parts are described in greater detail in the following sections.

A. *Audio Preprocessing*

1) *Input Signal Recording*

The audio for which the lip-sync will be generated is captured at the beginning of the pipeline using PyAudio 0.2.11. The user is prompted for speech. This recording is saved as a temporary 16bit, mono, 16kHz Wave file using Python’s Wave module. The recorded audio can be played back and visualized.

2) *Dataset*

The Neural Network which classifies the type of speech is trained and tested using a dataset constructed specifically for this project. The dataset is comprised of speech recordings from 50 different subjects. The recordings of the subjects were then broken into 2 second chunks. Each chunk was manually labeled as belonging to one of three categories: whisper, talk, and shout, or discarded if not suitable. The audio clips were obtained either from subjects elocuting in a high quality recording setup or from online sources such as YouTube videos chosen for this specific purpose. The total time of the dataset is 103 minutes. The dataset was then split, using 70% of the clips for training the Neural Network, and 30% of the clips for testing it.

3) *Dataset Preprocessing*

The same preprocessing that is applied to audio in the dataset is applied to the pipeline's input audio. To be able to feed the recorded audio into the Neural Network the first step is to break the recorded audio into 2 second chunks using Pydub.

Next, all of the silent sections, mainly comprised of the time periods between spoken words, are removed from each chunk so that the only remaining signal is pure speech (Note that the clips may now be less than 2s in length). This is done by running a bash script on each chunk of audio which uses the silence removal tool in the FFmpeg 4.0 libraries. Any sound level below a threshold of -50dB is removed from the chunk. Each chunk is imported into Librosa, whereupon if it is not already in this format, it is converted to a 16bit, mono, 16kHz Wave file. The chunks are also normalized.

MFCC representation of the signals is the next step in the audio preprocessing. MFCCs are the coefficients of the real cepstrum of a time-frame of the signal’s estimated spectrum. MFCC use a non-linear frequency scale to approximate human hearing. They contain information about the physical aspects of the speech signal. Since their origin in the 1970s they have been used extensively as features in speech processing applications [19].

For every 30ms window of the preprocessed audio, 14 Mel Frequency Bands are packed into a 14-dimensional MFCC vector, using Librosa. A full 2 second chunk of speech contains 63 such vectors. However, after silence removal, typically, chunks are shorter, so fewer vectors can be generated. Consequently, for each sound chunk a feature array comprised of 45 of these 14-dimensional vectors is fed to the Neural Network. If there are more than 45 MFCC vectors in a chunk, the remaining vectors are truncated and discarded. If there are less than 45 vectors in a chunk, the existing vectors are concatenated to themselves until the feature array is of length 45, so that the chunk can still be fed to the model. This manipulation of the feature array was performed using NumPy.

Additionally, for the purpose of training the Neural Network, the feature arrays are grouped together with their respective labels, and randomly shuffled. The feature arrays are reshaped such that a One-Hot-Encoding can be created using Keras [20].

B. *Neural Network*

The trained Neural Network, which uses Keras as an interface for Tensorflow, is then loaded in the Jupyter Notebook. Details on the Neural Network and its architecture are included in a later section. Every chunk of audio has a corresponding set of MFCC feature vectors. The features are fed into the Neural Network, and a prediction is obtained for the type of speech in each recorded chunk. The network’s output is the probability of the chunk of speech belonging to any of the three speech categories.

C. Phoneme Detection and Viseme Mapping

Phoneme detection is carried out using CMUSphinx, a package featuring a variety of speech recognition systems developed by Carnegie Mellon University. One of the features offered is Phoneme Recognition, which requires a decoder to be set up using a language model (this project uses the US language model). The versions used are Sphinx4-5 and PocketSphinx 0.8. The audio sent to Sphinx must be a normalized 16 bit mono signal with a sampling rate of 16kHz, which was acquired at the beginning of the pipeline.

CMUSphinx identifies each phoneme in the input audio as one of 39 possibilities, based on the ARPABET phonetic transcription codes for American English [21]. It also returns the start and stop times of each phoneme. After the phonemes in the audio have been labelled, they are mapped to one of 12 possible visemes specified by the Jeffers phonemes to viseme map [22]. This is a popular mapping in lip-reading applications, and is a particularly high-performance mapping in regard to its accuracy for consonants [23]. Note that the phoneme detection happens parallel to the audio preprocessing for classification (they occur on different branches in fig. 1 above). In the preprocessing branch the silences are removed from the audio to classify chunks, but in the phoneme detection branch, silences are preserved and are one of the possible visemes we map to. After this step, we have access to all the visemes in the input audio, along with their start and stop times.

D. Mouth Shape Generation

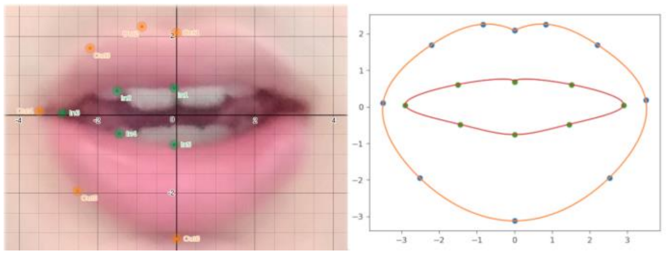


Fig. 2. a) Left, a photo of the subject elocuting a viseme, overlaid on a grid with control points. b) Right, the same viseme displayed as sparse mouth shape created with B-spline interpolations through the control points.

The pipeline outputs the lip sync as a collection of 2-Dimensional Mouth objects, each depicting the sparse mouth shape corresponding to a phoneme in the input audio.
The Mouth objects consist of two sets of coordinates. One of the sets represents the outer shape of the lips, and the other represents the inner border of the lips with the mouth cavity. The coordinates correspond to 18 control points defined in the MPEG-4 facial animation standard parameters [16].

Three models of each viseme specified in the Jeffers mapping were created, one for each speech type, according to these specifications. Photos of the mouth of a subject elocuting the visemes were overlaid on a 2-Dimensional grid (see Fig. 2), such that the control points could be accurately placed manually. By changing the coordinates of the control points, the pipeline can easily be customized to create different mouth-shapes.

Each Mouth object has a display function, which draws the mouth using B-spline interpolation feature from the SciPy package. A B-Spline (or basis spline) is a computationally efficient type of smooth curve which can readily undergo custom shaping, and is commonly used in curve fitting applications. The curve's shape is determined by a set of control points and basis functions [24]. The display of each mouth shape consists of two such interpolations, corresponding to the two sets of coordinates of a Mouth object. The first one uses the coordinates of the outer lip shape as control points, whilst the second uses the coordinates of the inner lip border. Plotting the B-splines through these points results in a 2-Dimensional rendering of a sparse mouth shape to visualize the output of the pipeline. Any computer capable of interpolating B-splines can generate the mouth shapes once it has acquired the coordinate information.

The Mouth object generator averages the coordinates of the three models of a viseme according to the Neural Network's predicted probabilities of speech type for a chunk of audio. If the maximum predicted probability for a given type of speech in a chunk is greater or equal to 0.8, then the pure model of the viseme for this type of speech is outputted. On the other hand, if the threshold of 0.8 is not reached by any of the speech categories, a new Mouth is constructed by combining the viseme models. If the maximum predicted probability belongs to the 'whisper' category, then for every control point in the mouth, the coordinates of this control point in both the 'whisper' and 'talk' models undergo a weighted averaging, using the predicted probabilities as the weights. The horizontal and vertical coordinates are averaged separately to produce new sets of coordinates. Likewise, if the maximum probability belongs to the 'shout' category, the 'shout' control points will be averaged with 'talk'. However, if the highest probability is the 'talk' category, then the control points of the 'talk' viseme are subjected to a weighted average with the control point coordinates of both 'whisper' and 'shout', which results in minor alterations to the initial "talk" viseme. The new set of coordinates resulting from this averaging process is returned as a Mouth object.

For every control point in a viseme, the weighted averaging is performed as shown below, where n is the number of Mouth models being combined (n=2 or n=3 as described above), x_i is the value of the control point's coordinate, and w_i is the predicted probability of the corresponding chunk of audio belonging to a category (the weight). We need to normalize by dividing the numerator by the sum of predicted probabilities as the denominator doesn't always sum to 1 when n=2 (in the case of whispering or shouting).

$$\frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} = \frac{3.63(0.0102) + 3.52(0.6787) + 3.83(0.3111)}{0.0102 + 0.6787 + 0.3111} = 3.62$$

Taking as an example the x-coordinate of a single control point, say the rightmost point in the talking 'G' phoneme as seen in Table 3. The chunk's predicted class probabilities are

$w_1 = 0.0102$ for whisper, $w_2 = 0.6787$ for talk, and $w_3 = 0.3111$ for shout. The x-coordinate of the mouth models are $x_1 = 3.63$ for whisper, $x_2 = 3.52$ for talk, and $x_3 = 3.83$ for shout. So the final value of the x-coordinate of this control point is 3.62.

E. Integration

The final integration of the pipeline proceeds by iterating through all of the visemes recognized in the recorded input. For each viseme, the timestamps from CMUSphinx are used to calculate the chunk of audio it belongs to. To do this, using integer division, the visemes start and stop times are divided by the chunk duration (2 seconds) to obtain the chunk number. If the resulting chunk numbers are equal to each other for both the start and end timestamps, it is used as the chunk number for the current viseme. On the other hand, if the start and end timestamp chunk numbers are not equal to each other, the pipeline checks whether the largest portion of time the viseme occupies is in the previous, current, or next chunk.

Having obtained the chunk number, the Neural Network can make a prediction for the chunk, and along with the corresponding viseme, is used to generate a Mouth object. This Mouth object is stored in an array along with the start and end timestamps.

IV. Experiments and Discussion

In this section, the details of the implementation of the Neural Network for classifying speech types are discussed, and the results of the overall pipeline are shown and evaluated, along with a discussion about its applications and limitations.

A. Running Times and Hardware

The runtimes for the neural network, the pipeline and all of its parts are reported based on being run on an Intel Core i7-2677M CPU, clocked at 1.80GHz. A typical total running time of the pipeline on a two second chunk of speech is 3.1s. The average breakdown of the run-times is as follows: The preprocessing of the recording takes 0.06s. The detection of 20 phonemes with CMUSphinx, and the mapping to visemes takes 1.6s. Predicting speech types and generating the mouth shapes for the visemes takes 1.4s. If all the mouth shapes were to be plotted and printed using the display function, that takes another 17.2s.

TABLE I
Performance Measures of the Neural Network

Number of Epochs	Average training accuracy	Average training loss	Average testing accuracy
1	0.730	0.627	0.730
2	0.827	0.432	0.763
3	0.850	0.360	0.795
5	0.877	0.317	0.803
8	0.902	0.254	0.832
10	0.919	0.234	0.838
13	0.962	0.150	0.814
15	0.982	0.112	0.820
20	0.996	0.044	0.803

B. Neural Network Architecture and Evaluation

The Feedforward Network consists of 45 nodes, and was trained with a batch size of 32. The input layer has 14 nodes, a sigmoid activation function, and 5% dropout. Two hidden layers have 14 nodes each, and both have RELU activation functions. The output layer has 3 nodes and a softmax function to finalize predictions. It is trained using the ADAMS optimizer [25], implemented in Keras for TensorFlow. Training took 24.7 seconds over the course of 10 epochs. With this configuration the Network achieves 84.2% accuracy.

The variation of loss in training, and the variation of accuracy in the training and testing sets when the number of epochs used was varied is investigated (See Table 1). For the validation, the average accuracy over 10 trials peaks at 83.8% with 10 epochs (See Fig. 3).

The training set size was also varied to determine the effect on validation accuracy. Using 50%, 60%, 65%, and 70% of the dataset for training showed that the accuracy of the network improved with more data.

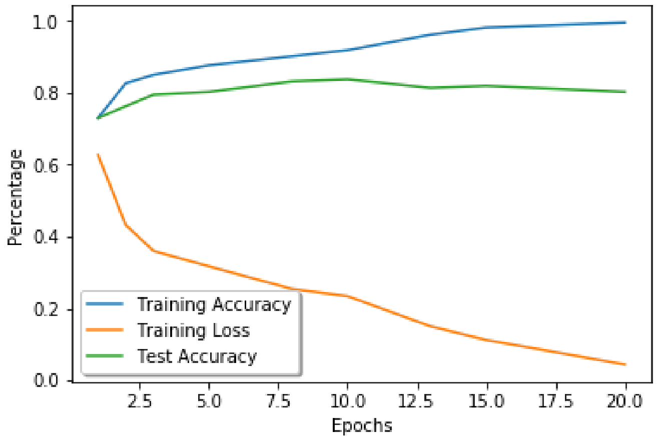


Fig. 3. Displaying the performance measures of the Neural Network

C. Mouth Generation Evaluation and Discussion

All of the figures used in the results were generated using clips from new audio recordings which were not included in the training or validation datasets. The figures below show the output of the pipeline for the phonemes L – G – AH used in the words 'hello' and 'goodbye' for the three speech types. Also shown are the probabilities assigned to the relevant chunk of audio by the Neural Network. To see the full sequence of Mouth shapes for the words 'Hello' and 'Goodbye' spoken using different speech types, and the timestamps, see Appendix A.

The images in Table 2 show the output of the pipeline for the phoneme 'L' from the word 'hello' for the 3 different speech types. The probabilities predicted by the Neural Network of the audio chunk belonging to each speech type are also shown. For the whisper, the Neural Network predicts with over 99.99% certainty that the audio clip is a whisper, meaning that the pure, non-averaged, whispering viseme model is used in the output. It depicts the smaller range of motion of the lips which is associated with whispering. For the talking audio, the neural network once again predicts the class with over 99.99% accuracy, meaning the stock model of a talking mouth is used. For 'Shouting', the Neural Network predicts the category with 52.45% certainty. As a result, the control points of the shouting model are subject to the weighted averaging process with the points of the talking model (which received a 46.91% probability). The resulting mouth shapes are convincing, resulting in a wider open mouth with appropriately angled edges for the phonemes.

The images in Tables 3 and 4 show the output for the phonemes 'G' and 'AH' respectively, from the word "goodbye". The whisper is classified with 92.12% certainty, and the stock whisper mouth models are used for the phonemes. For the talking category the 67.87% prediction is below the 0.8 threshold, and so the points with the model are averaged with both those of the whisper (weighted at 1.02%), and the shout (weighted at 31.11%). The shouting class is predicted with 99.62% accuracy, and so is a good example of the pure model shouting mouths.

One possible application for this method is in animations. The user just needs to input an audio clip to obtain accurate mouth shapes with timestamps for use in the animation. It is easy to apply transformations such as rotations or scaling to the control points to integrate the mouth shapes in the footage, saving time and cutting costs for creators. The control points are easily customized to create different mouths for different characters. Furthermore, the low memory requirements for storing and transmitting a Mouth object means that this system is a good candidate for use in the remote reproductions of avatars. It is a simple and memory effective to reconstruct unique avatar mouth shapes on a remote machine based only on the transmission of the control points.

V. Limitations and Future Work

In general, the limitations affecting the pipeline establish the future work that remains to be done on the system. One of the main limitations of the system is the use of CMUSphinx. As the creators of the package warn [26], their phoneme recognition error rate is "considerably" high. The phonemes it outputs are for the most part visually convincing for spoken or shouted words (as long as no distortion is present in the signal), but not for whispered inputs. Furthermore, the phoneme recognition takes a significant amount of time to run, making real-time mouth generation impossible. The most critical future work consists of finding a more robust and quicker system for phoneme detection. Other such systems worth trying are available on packages such as Kaldi [27]. Alternatively, it is also worth considering creating our own specialized phoneme detection neural network tailored to the performance needs of the pipeline.

A failure case is seen when two phonemes mapped to identical visemes are consecutively detected. It is likely these will be located in the same audio chunk, thus possessing the same predicted probabilities, meaning that the resulting viseme will occupy a long span of time without changing, which looks unnatural when animated. A possible solution to this problem could be sampling the audio fed to the classifying neural network at more frequent intervals. Instead of feeding discrete two second chunks of audio to the network, a two second sliding window of audio could be moved forward every 30ms for classification purposes, and might produce more varied and realistic mouths. Another option would be to randomize the second viseme's control points by a small amount, to cause it to change shape.

Another limitation is that the pipeline requires high quality audio files to function properly. Audio recorded with a laptop's microphone rarely produced good results when fed into the neural network for classification. All the audio used was recorded on a high quality recording set up, which might not be available to all users. Also, both CMUSphinx and the generated visemes are limited by virtue of being based on the English language. New language packs and viseme mappings and models would have to be acquired for the system to work with other languages. Beyond the future work stemming from finding solutions to these limitations, a natural continuation to this project consists of having the pipeline output 3D mouth shapes instead of 2D ones. The existing code could easily be modified to produce the 3D mouth shapes by simply adding an extra dimension (for depth) to the existing control points in the MPEG-4 convention, as seen in [3]. The method of generating mouth shapes through averaging class models would not require modifications or new control points, and 3D objects are useful in more diverse applications, while retaining the advantages of our system's uniquely generated mouth objects. In the future, adding control points for the entire face to add realistic facial gestures to the lip sync using this pipeline is an idea worthy of further consideration.

Finally, an interesting question is how accurate a Neural Network would be at predicting speech types in a model with more than three classes. If it were successful, a similar, more nuanced system for generating mouth shapes could be implemented.

VI. Summary

In this article a system has been described for automatic lip synchronization, generating unique mouth shapes according to speech type, using only a speech input. Phonemes in the speech are detected using CMUSphinx. The crucial innovation of the system is the success (84% accuracy) of a Feedforward Neural Network which classifies speech types into three categories, using MFCC vectors as features. Mouth shapes for each viseme of the Jeffers map are generated by merging the control points of premade mouth models according to the Neural Network's predictions of the speech type.

The lip sync system can be used in various applications since the resulting mouth shapes are convincing and uniquely generated. The low storage and transmission requirements and the simplicity of reconstruction of the mouth objects make them a good choice for implementing low-bandwidth communication through animation.
It can be concluded that the pipeline satisfactorily generates detailed, lightweight, automatic lip sync frames, but there is

plenty of room for further work and improvements on the system.

APPENDIX

Tables 5 and 6 show the mouth shapes generated using recordings of the words 'hello' and 'goodbye' for each of the three speech classes. Each column of these tables corresponds to a speech class, as specified in the first row. The second row contains the phoneme and timestamp information generated with CMU Sphinx. The third row shows the probabilities of the recordings belonging to the speech classes in the form [whisper, talk, shout]. The remaining columns show the mouth shapes generated by the pipeline for each of the detected phonemes in the same order of appearance as the second row.

REFERENCES

[1] Sweeny, T.D., Guzman-Martinez, E., Ortega, L., Grabowecky, M. and Suzuki, S., 2012. Sounds exaggerate visual shape. Cognition, 124(2), pp. 194-200

[2] Shmyrev, N., About CMUSphinx. CMUSphinx Open Source Speech Recognition. Available at: https://cmusphinx.github.io/wiki/about/, 2018.

[3] MPEG-4 Face and Body Animation (MPEG-4 FBA) An overview. (2018). Linköping: Visage Technologies, pp.7-8.

[4] Lewis, J.P. and Parke, F.I., 1987, May. Automated lip-synch and speech synthesis for character animation. In ACM SIGCHI Bulletin (Vol. 17, No. SI, pp. 143-147). ACM.

[5] Koster, B.E., Rodman, R.D. and Bitzer, D., 1994. Automated lip-sync: Direct translation of speech-sound to mouth-shape. In Signals, Systems and Computers, 1994. 1994 Conference Record of the Twenty-Eighth Asilomar Conference on (Vol. 1, pp. 583-586). IEEE.

[6] Bregler, C., Covell, M. and Slaney, M., 1997, August. Video rewrite: Driving visual speech with audio. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (pp. 353-360). ACM Press/Addison-Wesley Publishing Co.

[7] Brand, M., 1999, July. Voice puppetry. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (pp. 21-28). ACM Press/Addison-Wesley Publishing Co.

[8] Taylor, S.L., Mahler, M., Theobald, B.J. and Matthews, I., 2012, July. Dynamic units of visual speech. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (pp. 275-284). Eurographics Association.

[9] Huang, F.J. and Chen, T., 1998, December. Real-time lip-synch face animation driven by human voice. In Multimedia Signal Processing, 1998 IEEE Second Workshop on (pp. 352-357). IEEE.

[10] Graves, A., Mohamed, A.R. and Hinton, G., 2013, May. Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on (pp. 6645-6649). IEEE.

[11] Hibare, R. and Vibhute, A., 2014. Feature extraction techniques in speech processing: a survey. International Journal of Computer Applications, 107(5).

[12] Tang, Hao, Yun Fu, Jilin Tu, M. Hasegawa-Johnson, and T.s. Huang, 2008. Humanoid Audio–Visual Avatar With Emotive Text-to-Speech Synthesis. IEEE Transactions on Multimedia 10, no. 6 (pp. 969–81). IEEE.

[13] Xie, Lei, and Zhi-Qiang Liu, 2007. Realistic Mouth-Synching for Speech-Driven Talking Face Using Articulatory Modelling. IEEE Transactions on Multimedia 9, no. 3 (pp. 500–510). IEEE

[14] Agarwal, Swapna, and Dipti Prasad Mukherjee, 2019. Synthesis of Realistic Facial Expressions Using Expression Map. IEEE Transactions on Multimedia 21, no. 4 (pp. 902–14). IEEE.

[15] Zoric, G. and Pandzic, I.S., 2005, July. A real-time lip sync system using a genetic algorithm for automatic neural network configuration. In Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on (pp. 1366-1369). IEEE.

[16] MPEG-4 Face and Body Animation (MPEG-4 FBA) An overview. (2018). Linköping: Visage Technologies, pp.7-8.

[17] Suwajanakorn, S., Seitz, S.M. and Kemelmacher-Shlizerman, I., 2017. Synthesizing Obama: learning lip sync from audio. ACM Transactions on Graphics (TOG), 36(4), p.95.

[18] Kim, T., Yue, Y., Taylor, S. and Matthews, I., 2015, August. A decision tree framework for spatiotemporal sequence prediction. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 577-586). ACM.

[19] Huang, Xuedong, et al. Spoken Language Processing: a Guide to Theory, Algorithm, and System Development. Prentice Hall PTR, 2001 (pp. 313-316).

[20] Chollet, Francois, et al. Keras. https://keras.io, 2015.

[21] CMU, The CMU Pronouncing Dictionary. The CMU Pronouncing Dictionary. Available at: http://www.speech.cs.cmu.edu/cgi-bin/cmudict, 2018.

[22] Cappelletta, L. and Harte, N., 2011, August. Viseme definitions comparison for visual-only speech recognition. In Signal Processing Conference, 2011 19th European (pp. 2109-2113). IEEE.

[23] Bear, H.L. and Harvey, R., 2017. Phoneme-to-viseme mappings: the good, the bad, and the ugly. Speech Communication, 95, pp.40-67.

[24] De Boor, C., 2001. A practical guide to splines, revised Edition, Vol. 27 of Applied Mathematical Sciences. Mechanical Sciences, year

[25] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[26] Shmyrev, N., Phoneme Recognition (caveat emptor) . CMUSphinx Open Source Speech Recognition. Available at: https://cmusphinx.github.io/wiki/phonemerecognition, 2018

[27] Povey, Daniel. Kaldi ASR. https://kaldi-asr.org, 2020.

TABLE 2
THE PREDICTED PROBABILITIES AND RESULTING MOUTH SHAPES FOR THE 'L' PHONEME.

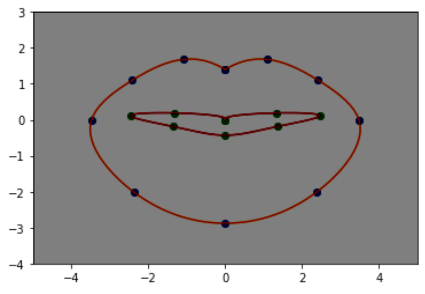
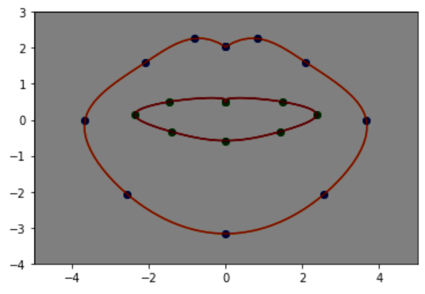
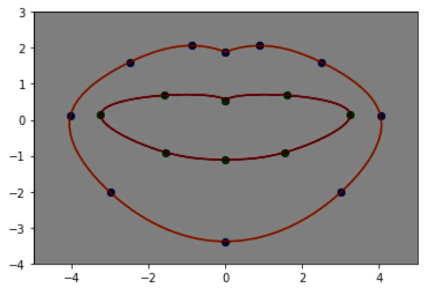
Whisper	Talk	Shout
[0.9999 Whisper, 1.299E-5 Talk, 9.211E-8 Shout]	[1.616E-5 Whisper, 0.9999 Talk, 3.989E-5 Shout]	[0.0064 Whisper, 0.4691 Talk, 0.5245 Shout]
		

TABLE 3
THE PREDICTED PROBABILITIES AND RESULTING MOUTH SHAPES FOR THE 'G' PHONEME.

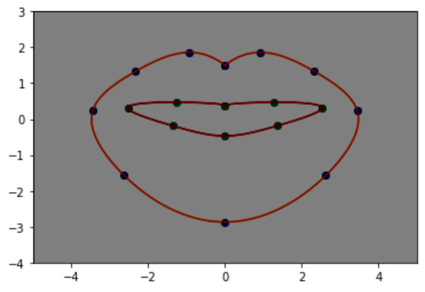
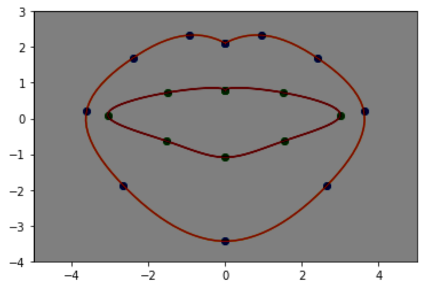
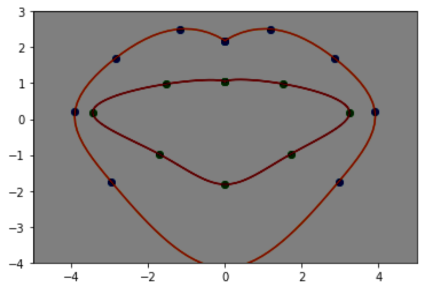
Whisper	Talk	Shout
[0.9211 Whisper, 0.0702 Talk, 0.0085 Shout]	[0.0102 Whisper, 0.6787 Talk, 0.3111 Shout]	[0.0012 Whisper, 0.0025 Talk, 0.9961 Shout]
		

TABLE 4
THE PREDICTED PROBABILITIES AND RESULTING MOUTH SHAPES FOR THE 'AH' PHONEME.

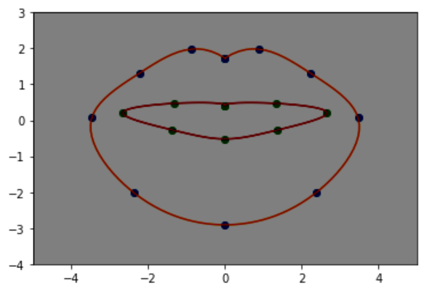
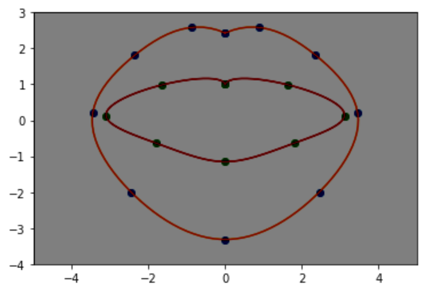
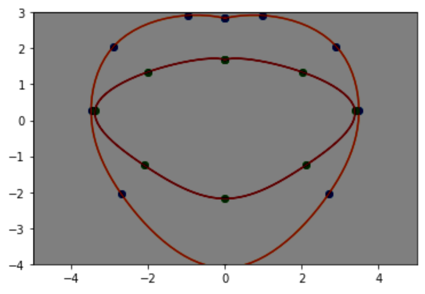
Whisper	Talk	Shout
[0.9211 Whisper, 0.0702 Talk, 0.0085 Shout]	[0.0102 Whisper, 0.6787 Talk, 0.3111 Shout]	[0.0012 Whisper, 0.0025 Talk, 0.9961 Shout]
		

TABLE 5
'HELLO' IN DIFFERENT SPEECH TYPES

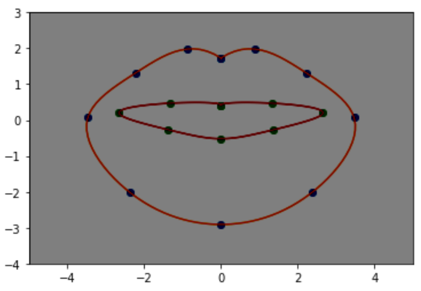
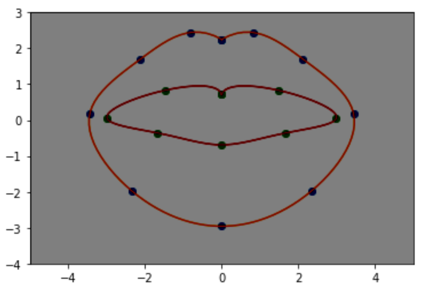
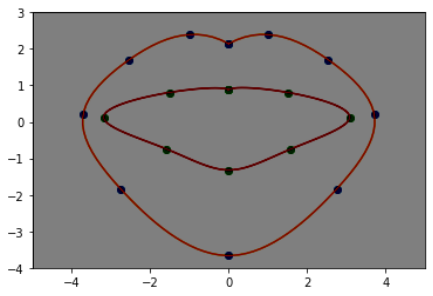
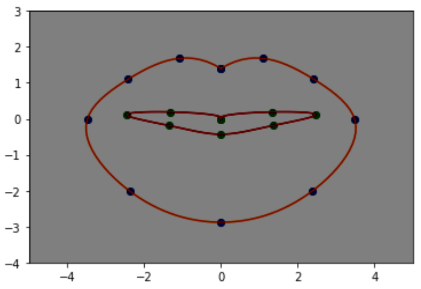
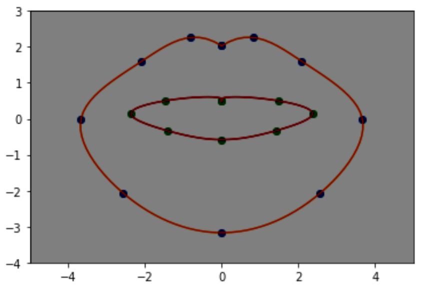
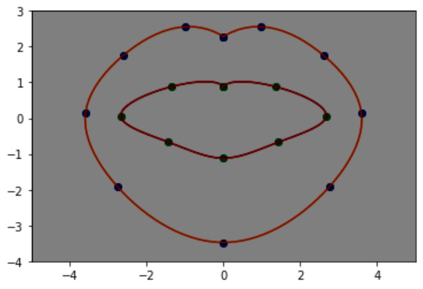
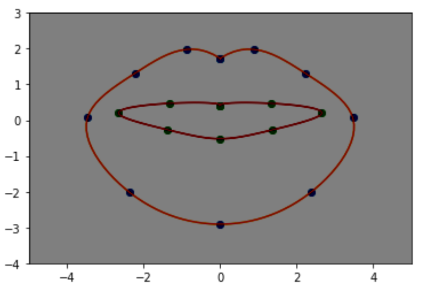
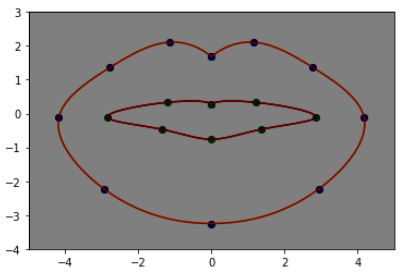
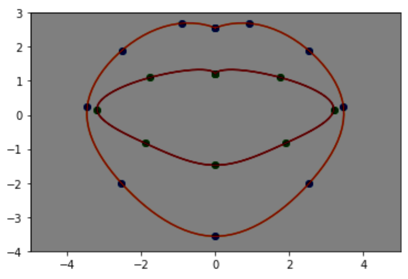
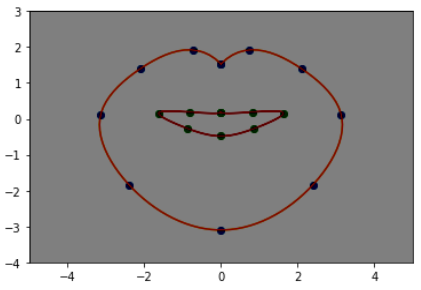
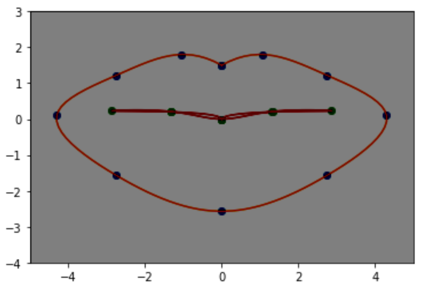
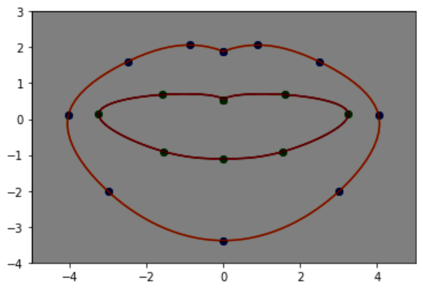
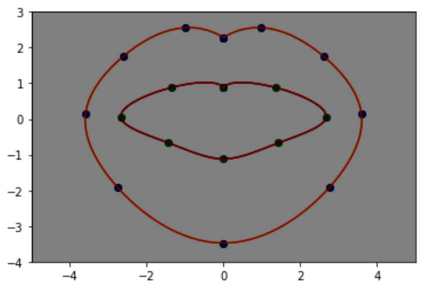
Whisper	Talk	Shout
Phonemes and timestamps: [('SIL', 0.0, 0.03), ('AE', 0.04, 0.11), ('L', 0.12, 0.18), ('AA', 0.19, 0.38), ('HH', 0.39, 0.43), ('AO', 0.44, 0.48)]	Phonemes and timestamps: [('SIL', 0.0, 0.02), ('AH', 0.03, 0.07), ('L', 0.08, 0.17), ('OW', 0.18, 0.34)]	Phonemes and timestamps: [('SIL', 0.0, 0.02), ('G', 0.03, 0.1), ('OW', 0.11, 0.23), ('AA', 0.24, 0.36), ('AE', 0.37, 0.39), ('L', 0.4, 0.47), ('OW', 0.48, 0.93)]
Speech class probabilities: [9.9998689e-01, 1.2992635e-05, 9.2109254e-08]	Speech class probabilities: [1.6160682e-05, 9.9994385e-01, 3.9889543e-05]	Speech class probabilities: [0.00641645, 0.46911407, 0.5244695]
		
		
		
		
		

TABLE 6
'GOODBYE' IN DIFFERENT SPEECH TYPES

Whisper	**Talk**	**Shout**
Phonemes and timestamps: [('SIL', 0.67, 0.75), ('K', 0.76, 0.82), ('UW', 0.83, 0.86), ('DH', 0.87, 0.92), ('SIL', 0.93, 1.06), ('AH', 1.07, 1.1), ('AA', 1.11, 1.28)]	Phonemes and timestamps: [('SIL', 0.0, 0.02), ('G', 0.03, 0.1), ('UH', 0.11, 0.15), ('B', 0.16, 0.21), ('P', 0.22, 0.25), ('B', 0.26, 0.3), ('AA', 0.31, 0.44), ('AY', 0.45, 0.56)]	Phonemes and timestamps: [('SIL', 0.0, 0.03), ('G', 0.04, 0.09), ('OW', 0.1, 0.18), ('D', 0.19, 0.24), ('OW', 0.25, 0.28), ('AA', 0.29, 0.56), ('AY', 0.57, 0.82), ('AH', 0.83, 0.89)]
Speech class probabilities: [0.9211595 , 0.0702494 , 0.00859113]	Speech class probabilities: [0.01024215, 0.67867833, 0.3110795]	Speech class probabilities: [0.00127534, 0.00252765, 0.996197]