# Trainable Videorealistic Facial Animation

Tony Ezzat
*tonebone@ai.mit.edu*

Gadi Geiger
*gadi@ai.mit.edu*

Tomaso Poggio
*tp@ai.mit.edu*

Center for Biological and Computational Learning
Massachusetts Institute of Technology

## Abstract

We present a technique to perform trainable videorealistic facial animation. A human subject is first recorded using a digital videocamera as he/she utters a pre-determined speech corpus. After processing the corpus automatically and performing necessary training, a visual speech module is produced that is capable of resynthesizing the human subject uttering entirely novel utterances that were not recorded in the original video. The output is videorealistic in the sense that it looks like a video camera recording of the subject. The input to the system can be either real audio sequences or synthetic audio produced by a text-to-speech system, as long as they have been phonetically aligned.

The two key contributions of this paper are 1) the notion of a *multidimensional morphable model* (MMM), which is capable of multidimensional morphing between a small set of mouth image prototypes to synthesize new, previously unseen mouth configurations; and 2) a *trajectory synthesis technique* based on regularization which is automatically trained from the recorded video corpus, and which is capable of synthesizing trajectories in MMM space corresponding to any desired utterance,

**CR Categories:** I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation; I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Video Analysis I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Motion

**Keywords:** facial modelling, facial animation, morphing, optical flow, speech synthesis, lip synchronization.

## 1 Overview

In this work, we present a technique to perform trainable videorealistic facial animation. A human subject is first recorded using a digital videocamera as he/she utters a pre-determined speech corpus. After processing the corpus automatically and performing necessary training, a visual speech module is produced that is capable of re-synthesizing the human subject uttering entirely novel utterances that were not recorded in the original video. The output is videorealistic in the sense that it looks like a video camera recording of the subject.

We choose to focus our efforts in this work on the issues related to the synthesis of novel video, and not on novel audio synthesis. Thus, novel audio needs to be provided as input to our system. This audio can be either real human audio (from the same subject or a different subject), or synthetic audio produced by a text-to-speech system. All that is required by our system is that the audio be phonetically transcribed and aligned. In the case of synthetic audio from TTS systems, this phonetic alignment is readily available from the TTS system itself [7]. In the case of real audio, publicly available phonetic alignment systems [22] may be used.

Our visual speech processing system is composed of two modules: The first module is the *multidimensional morpheable model* (MMM), which is capable of morphing between a small set of prototype mouth images to synthesize new, previously unseen mouth configurations. The second component is a *trajectory synthesis* module, which uses regularization [19] to synthesize smooth trajectories in MMM space for any specified utterance. The parameters of the trajectory synthesis module are trained automatically from the recorded corpus using gradient descent learning.

Recording the video corpus takes on the order of 15 minutes. Processing of the corpus takes on the order of several days, but, apart from the specification of head and eye masks shown in Figure 1, is *fully automatic*, requiring no intervention on the part of the user. The final visual speech synthesis module extracts from recorded corpus a *small* set of prototype images (46 images in the case presented here) which are are used to synthesize all novel sequences. Synthesis of novel utterances takes on the order of about 7 seconds per frame for an image resolution of 624x472 on a circa 1998 450 Mhz Pentium II machine.

Application scenarios for videorealistic facial animation include: user-interface agents, in which they would lend a more human-friendly interface to desktops, TVs, or cell-phones; digital actors in movies, in which a director would be able to reanimate as many takes as he/she desires; virtual avatars, in which they would represent users in a chatroom scenario; very low bitrate coding schemes (such MPEG4), in which they would be used to encode regular video streams more efficiently; studies of visual speech production and perception, in which they would be used to evaluate various scientific models of speech production and perception; aids to the hearing-impaired, in thich they would be used to provide a visual representation of spoken speech in order to assist the deaf; finally, “information-warfare” scenarios, in which fake video is presented to confuse the enemy. The recorded subjects can be regular people, celebrities, ex-presidents, or infamous terrorists.

After reviewing the relevant prior work in Section 2, we begin by first describing the corpus that is recorded in Section 3. The necessary pre-processing of the corpus is discussed in Section 4. Section 5 describes how the MMM is built from the recorded corpus. Section 6 describes how the trajectory synthesis module is built and trained from the corpus. Finally, the last sections describe some of our results and further avenues of exploration.

# 2 Background

## 2.1 3D Facial Modelling and Animation

The main research issue underlying visual speech synthesis is the nature of the facial model to use. One approach is to model the face using classic *3D modeling* methods. Parke [30] was one of the earliest to adopt such an approach by creating a polygonal facial model. To increase the visual realism of the underlying facial model, the facial geometry is frequently scanned in using Cyberware laser scanners. Additionally, a texture-map of the face extracted by the Cyberware scanner may be mapped onto the three-dimensional geometry [26]. Guenter [20] demonstrated recent attempts at obtaining 3D face geometry from multiple photographs using photogrammetric techniques. Pighin et al. [32] captured face geometry and textures by fitting a generic face model to a number of photographs. Blanz and Vetter [8] demonstrated how a large database of Cyberware scans may be used to obtain face geometry from a single photograph.

In terms of animation, techniques have traditionally included both keyframing methods and physics-based methods, and have been extended more recently to include machine learning methods. In keyframing, the animator specifies particular key-frames, and the system generates intermediate values [30] [31] [14] [27]. In physics-based methods, the animator relies on the laws of physics to determine the mouth movement, given some initial conditions and a set of forces for all time. This technique usually requires modelling the underlying facial muscles and skin, and were demonstrated quite effectively [36] [26]. Finally, machine learning methods are a new class of animation tools which are trained from recorded data and then used to synthesize new motion. Examples include hidden markov models (HMMs), which were demonsrated effectively for facial animation by [9] [28] [11].

## 2.2 Image-based Models

An alternative to the 3D modelling approach is to model the talking face using *image-based* techniques, where the talking facial model is constructed using a collection of images captured of the human subject. These methods have the potential of achieving very high levels of videorealism, and are inspired by the recent success of similar sample-based methods for audio speech synthesis [29].

Image-based facial animation techniques need to solve the *video generation* problem: How does one build a generative model of video that is simultaneously *photorealistic*, *videorealistic*, and *parsimonious*? *Photorealism* means that the novel generated images exhibit the correct visual structure of the lips, teeth, and tongue. *Videorealism* means that the generated *sequences* exhibit the correct motion, dynamics, and coarticulation effects [14]. *Parsimony* means that the generative model uses a small amount of training data.

Bregler, Covell, el al. [10] describe an image-based system called Video Rewrite in which video generation problem is solved by breaking down the recorded video corpus into a set of smaller audiovisual basis units. Each one of these short sequences is a *triphone* segment, and a large database with all the acquired triphones is built. A new audiovisual sentence is constructed by *concatenating* the appropriate triphone sequences from the database together. Photorealism is adressed by only using recorded sequences to generate the novel video. Videorealism is achieved by using triphone contexts to model coarticulation effects, but in order to handle all the possible triphone contexts, the system requires a library with tens and possibly hundreds of thousands of subsequences, which seems to be an overly-redundant and non-parsimonious sampling of human lip configurations. Essentially, Video Rewrite trades off parsimony in favor of videorealism: since it does not have the capacity to *generate* novel lip imagery from a few recorded images, it relies on the re-sequencing of a vast amount of original video.

Cosatto and Graf [16] describe an approach which attempts to reduce this redundancy by embedding the space of mouth images in a parameterized space. The imposed dimensions of this lip space are lip width, position of the upper lip, and position of the lower lip. This 3-dimensional lip space is then populated by using the images from the recorded corpus. Synthesis is performed by traversing Cohen-Massaro [14] trajectories in this imposed lip space. The authors use a pixelwise blend to generate novel lip images, but this generative technique is clearly not photorealistic, producing undesirable ghosting effects. To avoid the effects of pixel blending, the parameterized space needs to be populated by thousands of images in order for the synthesis to look realistic. In this manner, parsimony is sacrificed in favor of photorealism. Videorealism is addressed by using Cohen-Massaro coarticulation rules, but these rules are hand-crafted and not trained from the recorded data, which renders them difficult to use in a general setting.

Scott, Kagels, et al. [34] [37] and Ezzat and Poggio [18] explore 2D morphing for image-based facial animation. Their facial models are composed of a small set of *visemes* (the visual manifestation of phonemes). Scott, Kagels, et al. used a set of 40-50 visemes, while Ezzat and Poggio used a set of 17 visemes. To animate the face, a 2D viseme-to-viseme morphing algorithm is developed which is capable of transitioning smoothly between the various mouth shapes. While 2D morphing was shown to be adequate at achieving parsimony and photorealism, it was not capable of addressing videorealism: In viseme-to-viseme morphing there is no notion of a *space* in which it is possible to synthesize smooth trajectories which address coarticulation effects and dynamics. For example, output from the Ezzat-Poggio system looked considerably jerky and required slowing the audio down.

## 2.3 Multidimensional Morphable Models

In this work, we introduce the notion of a *multidimensional morphable model* (MMM), which is capable of addressing all three issues of photorealism, videorealism, and parsimony simultaneously. MMM’s have already been introduced in other works [24] [5] [17] [23] [8]. We develop an MMM variant in this work and show its utility for facial animation.

The basic underlying assumption of the MMM is that the complete set of mouth images associated with human speech lie in a low-dimensional space whose axes represent mouth *appearance* variation and mouth *shape* variation. Mouth appearance refers to the raw pixel values of the images, and is represented in the MMM as a set of prototype images extracted from the recorded corpus. Mouth shape refers to the motion of the mouth with respect to a reference image, and is represented in the MMM as a set of optical flow vectors [21] computed automatically from the recorded corpus. In the work presented here, 46 images are extracted and 46 correspondences are computed. The low-dimensional MMM space is paramaterized by shape parameters $\alpha$ and appearance parameters $\beta$.

The MMM maybe viewed as a "black box" capable of performing two tasks: Firstly, given as input a set of parameters $(\alpha, \beta)$, the MMM is capable of *synthesizing* an image of the subject's face with that shape-appearance configuration. Synthesis is performed by morphing the various prototype images to produce novel, previously unseen mouth images which correspond to the input parameters $(\alpha, \beta)$. Conversly, the MMM can also perform *analysis*: given an input lip image, the MMM computes shape and appearance parameters $(\alpha, \beta)$ that represent the position of that input image in MMM space. In this manner, it is possible to *project* the entire recorded corpus onto the constructed MMM, and produce a time series of $(\alpha_t, \beta_t)$ parameters that represent trajectories of mouth motion in MMM space. We term this operation *analyzing the recorded corpus*.

Photorealism is achieved in an MMM because novel images are generated by morphing extracted prototype images. Parsimony is achieved because only a small set of prototypes (46 in this case) are needed. What about videorealism?

### 2.4 Trajectory Learning and Synthesis

Videorealism is achieved in our visual speech synthesis system through the *trajectory synthesis* module. This module maps from an input stream of phonemes (with their respective frame durations) to a *trajectory* of MMM shape-appearance parameters. This trajectory is then fed into the MMM to synthesize the final visual stream that represents the talking face.

The basic assumption made by the trajectory synthesis module is that visual speech is composed of a set of *phonemes* which are concatenated together in rapid succession to form the visual utterance. Each phomeme is represented as a localized Gaussian target region in MMM space. For a given utterance, the trajectory synthesis module synthesizes a trajectory which passes through the relevant phonetic regions in MMM space while maintaining an appropriate level of smoothness. Mathematically, a *regularization* framework [19] is employed to synthesize this trajectory. Coarticulation effects are regulated via the magnitude of the variance for each phoneme: small variance means low coarticulatory effects, large variance means high coarticulatory effects. Dynamics of each phoneme are regulated via the input phone stream, which determines how fast the system must pass through each phoneme region.

The trajectory synthesis module is itself trained from the analyzed corpus parameters. The training process determines the optimal position and shape of the phonetic regions in MMM space using a gradient descent procedure.

## 3 Corpus

A large audiovisual corpus of a human subject (Meredith) uttering various utterances was recorded. Recording was performed at a TV studio against a blue "chroma-key" background with a standard Sony analog TV camera. The data was subsequently digitized at a 29.97 fps NTSC frame rate with an image resolution of 640 by 480 and an audio resolution of 44.1KHz. The final sequences were stored as Quicktime sequences compressed using a Sorenson coder. The recorded corpus lasts for 15 minutes, and is composed of approximately 30000 frames.

The recorded corpus consisted of 1-syllable and 2-syllable words, such as `''bed''` and `''dagger''`. A total of 152 1-syllable words and 156 2-syllable words were recorded. The subject was asked to utter all sentences in a neutral expression. In addition, the sentences themselves were designed to elicit no emotions from the subject.

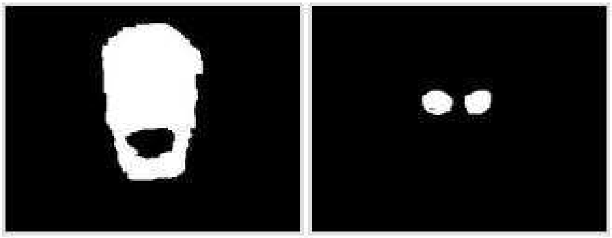

Figure 1: The head and eye masks used to normalize the corpus imagery

## 4 Pre-Processing

The recorded corpus data needs to be pre-processed in several ways before it may be processed effectively for re-animation. Firstly, the audio needs to be phonetically aligned in order to be able to associate a phoneme for each image in the corpus. Secondly, each image in the corpus needs to be *normalized* so that the head in the image does not exhibit any head movement or eye movement. This normalization step produces corpus imagery where the only movement occurring in the entire frame is the mouth movement associated with speech.

### 4.1 Audio Alignment

We perform audio alignment on all the recorded sequences using the CMU Sphinx system [22], which is publically available. Given an audio sequence and an associated text transcript of the speech being uttered, alignment systems use Viterbi search to find the optimal start and end of phonemes for the given audio sequence. The alignment task is easier than the speech recognition task because the text of the audio being uttered is known apriori. Regular speech recognition systems have to figure out what is being uttered.

### 4.2 Head Movement

Although the subject was instructed to keep her head steady during recording, residual head movement nevertheless still exists in the final recorded sequences. Since the head motion is small, we make the simplifying assumption that it can be approximated as the perspective motion of a plane lying on the surface of the face. Planar perspective deformations [38] have 8 degrees of freedom, and can be inferred using 8 corresponding points between a reference frame and the current frame. We employ optical flow (see Appendix A) to extract correspondences for 640x480 pixels, and use least squares to solve the overdetermined system of equations to obtain the 8 parameters of the perspective warp. Among the 640x480 correspondences, only those lying withing the *head mask* shown in Figure 1 are used. Pixels from the background area are not used because they do not exhibit any motion at all, and those from the mouth area exhibit non-rigid motion associated with speech.

After computing the 8 planar perspective parameters, the image is warped towards the reference frame. This is performed for all images in the corpus. After warping the images are cropped to a dimension of 624x420 to eliminate the border artifacts associated with the warp.

### 4.3 Eye Movement

The images in the corpus also exhibit residual eye movement and eye blinks which need to be removed. An eye mask is created (see Figure 1) which allows just the eyes from a single frame to be pasted onto the rest of the corpus imagery. The eye mask is blurred at the edges to allow a seamless blend between the pasted eyes and the rest of face.

## 5 Multidimensional Morpheable Models

At the heart of our visual speech synthesis approach is the *multidimensional morpheable model* representation, which is capable of *morphing* between various lip images to synthesize new, previously unseen lip configurations. In the following sections, we describe how a multidimensional morpheable model is defined, how it may be acquired automatically from a recorded video corpus, how it may be used for synthesis, and, finally, how such a morpheable model may be used for analysis.

### 5.1 Definition

An MMM consists of a *set of prototype images* $\{I_i\}_{i=1}^N$ that represent the various lip textures that will be encapsulated by the MMM. One image is designated arbitrarily to be the *reference image* $I_1$.

Additionally, the MMM consists of a set of *prototype flows* $\{C_i\}_{i=1}^N$ that represent the correspondences between the reference image $I_1$ and the other prototype images in the MMM. The correspondence from the reference image to itself, $C_1$, is designated to be an empty, zero, flow.

In this work, we choose to represent the correspondence maps using relative displacement vectors:

$$C_i(\mathbf{p}) = \{d_x^i(\mathbf{p}), d_y^i(\mathbf{p})\} \tag{1}$$

A pixel in image $I_1$ at position $\mathbf{p} = (x, y)$ corresponds to a pixel in image $I_i$ at position $(x + d_x^i(x,y), y + d_y^i(x,y))$.

Previous methods at computing correspondence [2] [34] [25] have adopted *feature-based* approaches, in which a set of high-level *shape features* common to both images is specified. When it is done by hand, however, this feature specification process can become quite tedious and complicated, especially in cases when a large amount of imagery is involved. In our case the images to be morphed belong to an object that is undergoing mouth motion, so the problem of specifying correspondence between images thus reduces to the problem of estimating the motion field of the underlying moving object. This observation [4] [13] allows us to make use of a large number of *automatic* motion estimation algorithms for the purpose of computing the desired correspondence between two images. In this work, we make use of *optical flow* algorithms to estimate this motion (see Appendix A). This motion is captured as a two-dimensional array of displacement vectors, in the same exact format shown in Equation 1.

### 5.2 Building an MMM

An MMM must be constructed automatically from a recorded corpus of $\{I_j\}_{j=1}^S$ images. The two main tasks involved are to choose the image prototypes $\{I_i\}_{i=1}^N$, and to compute the correspondence $\{C_i\}_{i=1}^N$ between them. We discuss the steps to do this briefly below.

#### 5.2.1 PCA

Principal component analysis (PCA) is first performed on all the images of the recorded video corpus. PCA allows each image in the video corpus to be represented using a set of low-dimensional parameters.

Performing PCA using classical autocovariance methods [6], however, usually requires loading all the images and computing a very large autocovariance matrix, which requires a lot of memory. To avoid this, we adopt an on-line PCA method, termed EM-PCA [33] [35], which allows us to peform PCA on the images in the corpus without loading them all into memory. EM-PCA is iterative, requiring several iterations, but is guaranteed to converge in the limit to the same principal components that would be extracted from the classical autocovariance method. The EM-PCA algorithm is typically run in this work for 10 iterations.

Performing EM-PCA produces a set of $D$ 624x472 principal components and a matrix $\Sigma$ of eigenvalues. In this work, $D = 15$ PCA bases are retained. The images in the video corpus are subsequently projected on the principal components, and each image $I_j$ is represented with a $D$-dimensional parameter vector $p_j$.

#### 5.2.2 K-means Clustering

Selection of the prototype images is performed using *k-means clustering* [6]. The algorithm is applied directly on the $\{p_j\}_{j=1}^S$ low dimensional PCA parameters, producing $N$ cluster centers. Typically the cluster centers extracted by k-means clustering do not coincide with actual image datapoints, so the nearest images in the dataset to the computed cluster centers are chosen to be the final image prototypes $\{I_i\}_{i=1}^N$ for use in our MMM.

It should be noted that k-means clustering requires the use of an internal distance metric with which to compare distances between datapoints and the chosen cluster centers. In our case, since the image parameters are themselves produced by PCA, the appropriate distance metric between two points $p_m$ and $p_n$ is the *Mahalanobis distance metric*:

$$d(p_m, p_n) = (p_m - p_n)^T \Sigma^T \Sigma (p_m - p_n) \tag{2}$$

where $\Sigma$ is the afore-mentioned matrix of eigenvalues extracted by the EM-PCA procedure.

We selected $N = 46$ image prototypes in this work, which are partly shown in Figure 2. The top left image is the reference image $I_1$.

#### 5.2.3 Dijkstra

After the $N = 46$ image prototypes are chosen, the next step in building an MMM is to compute correspondence between the reference image $I_1$ and all the other prototypes. Although it is strictly possible to compute *direct* optical flow between the images, we have found that direct application of optical flow is not capable of estimating good correspondence when the underlying lip displacements between images are greater than 5 pixels.

It is possible to use *flow concatenation* to overcome this problem. Since the original corpus is digitized at 29.97 fps, there are many intermediate frames that lie between the chosen viseme prototypes. A series of consecutive optical flow vectors between each intermediate image and its successor may be computed and *concatenated* into one large flow vector that defines the global transformation between the chosen visemes (see Appendix B for details on flow concatenation).

Typically, however, viseme prototype images are very far apart in the recorded visual corpus, so it is not practical to compute concatenated optical flow between them. The repeated concatenation that would be involved across the hundreds or thousands of intermediate frames leads to a considerably degraded final flow.

To compute good correspondence between viseme prototypes, a method is needed to figure out how to compute the path from the reference example $I_1$ to the chosen image prototypes $I_i$ without

Figure 2: 24 of the 46 image prototypes included in the MMM. The reference image is the top left frame.

repeated concatenation over hundreds or thousands of intermediates frames. We accomplish this by constructing the *corpus graph* representation of the corpus: A corpus graph is an S-by-S sparse adjacency graph matrix in which each frame in the corpus is represented as a node in a graph connected to $k$ nearest images. The $k$ nearest images are chosen *k-nearest neighbors* algorithm [6], and the distance metric used is the Mahalanobis distance in Equation 2 applied to the PCA parameters $p$. Thus, an image is connected in the graph to the $k$ other images that look most similar to it. The edge-weight between a frame and its neighbor is the value of the Mahalanobis distance.

After the corpus graph is computed, the *Dijkstra* shortest path algorithm [15] is used to compute the shortest path between the reference example $I_1$ and the other chosen image prototypes $I_i$. Each shortest path produced by the Dijkstra algorithm is a list of images from the corpus that cumulatively represent the shortest deformation path from $I_1$ to $I_i$ as measured by the Mahalanobis distance. Concatenated flow from $I_1$ to $I_i$ is then computed along the intermediate images produced by the Dijkstra algorithm. Since there are 46 images, $N = 46$ correspondences $\{C_i\}_{i=1}^{N}$ are computed in this fashion from the reference image $I_1$ to the other image prototypes $\{I_i\}_{i=1}^{N}$.

## 5.3 Synthesis

The goal of synthesis is to map from the multidimensional parameter space $(\alpha, \beta)$ to an image which lies at that position in MMM space. Since there are 46 correspondences, $\alpha$ is a 46-dimensional parameter vector that controls mouth shape. Similarly, $\beta$ is a 46-dimensional parameter vector that controls mouth texture. The total dimensionality of $(\alpha, \beta)$ is 92.

Synthesis first proceeds by synthesizing a new correspondence $C^{synth}$ using *linear combination* of the prototype flows $C_i$:

$$C_1^{synth} = \sum_{i=1}^{N} \alpha_i C_i \tag{3}$$

The subscript 1 in Equation 3 above is used to emphasize that $C_1^{synth}$ originates from the reference image $I_1$, since all the prototype flows are taken with $I_1$ as reference.

Forward warping may be used to push the pixels of the reference image $I_1$ along the synthesized correspondence vector $C_1^{synth}$. Notationally, we denote the forward warping operation as an operator $\mathbf{W}(I, C)$ that operates on an image $I$ and a correspondence map $C$ (see Appendix C for details on foward warping).

However, a single forward warp will not utilize the image texture from *all* the examples. In order to take account all image texture, a *correspondence re-orientation* procedure first described in [4] is adopted that re-orients the synthesized correspondence vector $C_1^{synth}$ so that it originates from each of the other example images $I_i$. This allows us to subsequently forward warp all the examples along their respective re-oriented correspondence vectors.

Reorientation of the synthesized flow $C_1^{synth}$ proceeds in two steps, shown figuratively in Figure **??**: first, the synthesized flow $C_1^{synth}$ is subtracted from $C_i$ to yield a flow that contains the correct flow geometry, but which originates from the reference example $I_1$ rather than the desired example image $I_i$. To move the flow into the correct reference frame, this flow vector is *itself warped* along $C_i$. The resulting re-oriented flows $C_i^{synth}$ may be computed as:

$$C_i^{synth} = \mathbf{W}(C_i - C_1^{synth}, C_i) \tag{4}$$

Re-orientation is performed for all examples in the example set.

The third step in synthesis is to warp the prototype images $I_i$ along the re-oriented flows $C_i^{synth}$ to generate a set of $N$ warped image textures $I_i^{warped}$:

$$I_i^{warped} = \mathbf{W}(I_i, C_i^{synth}); \tag{5}$$

The fourth and final step is to blend the warped images $I_i^{warped}$ using the $\beta$ parameters to yield the final morphed image:

$$I^{morph} = \sum_{i=1}^{N} \beta_i I_i^{warped} \tag{6}$$

Combining Equations 3 through 6 together, our MMM synthesis may be written as follows:

$$I^{morph}(\alpha, \beta) = \sum_{i=1}^{N} \beta_i \mathbf{W}(I_i, \mathbf{W}(C_i - \sum_{j=1}^{N} \alpha_j C_j, C_i)) \tag{7}$$

Synthesis takes on the order of about 7 seconds per frame for an image resolution of 624x472 on a circa 1998 450 Mhz Pentium II machine. Typically only the image prototypes $I_i$ which are associated with higher values of $\beta_i$ are warped, which yields a considerable savings in computation time. We typically choose the top 10 $\beta_i$ prototypes.

Empirically we have found that the MMM synthesis technique is capable of surprisingly realistic re-synthesis of the lips as well as the teeth. The tongue is somewhat problematic, but this is because only a few of the image prototypes in the MMM display the tongue. A solution would thus be to include more images which display the tongue in the MMM. The blending of multiple images in the MMM for synthesis also averages out camera image sensor noise, which lends a smoother quality to the final syntesized image sequences.

### 5.4 Analysis

The goal of analysis is to *project* the entire recorded corpus $\{I_j\}_{j=1}^{S}$ onto the constructed MMM, and produce a time series of $(\alpha_j, \beta_j)_{j=1}^{S}$ parameters that represent trajectories of the original mouth motion in MMM space.

One possible approach for analysis of images is to perform *analysis-by-synthesis*: In this approach, used in various forms in [5] [17] [23] [8], the synthesis algorithm is used to synthesize an image $I^{synth}(\alpha, \beta)$, which is then compared to the novel image using an error metric (ie, the L2 norm). Gradient-descent is then usually performed to change the parameters in order to minimize the error, and the synthesis process is repeated. The search ends when a local minimum is achieved. Analysis-by-synthesis, however, is very slow in the case when a large number of images are involved.

In this work we choose another method that is capable of extracting parameters $(\alpha, \beta)$ in one iteration. In addition to the image $I^{novel}$ to be analyzed, the method requires that the correspondence $C^{novel}$ from the reference image $I_1$ in the MMM to the novel image $I^{novel}$ be computed beforehand. In our case, most of the novel imagery to be analyzed will be from the recorded video corpus itself, so we employ the Dijkstra approach discussed in the Section 5.2.3 to compute good quality correspondences between the reference image $I_1$ and $I_{novel}$.

Given a novel image $I^{novel}$ and its associated correspondence $C^{novel}$, the first step of the analysis algorithm is to estimate the parameters $\alpha$ which mimimize

$$\|C^{novel} - \sum_{i=1}^{N} \alpha_i C_i\| \tag{8}$$

This is solved using the famous pseudo-inverse:

$$\alpha = (C^T C)^{-1} C^T C^{novel} \tag{9}$$

where C above is a matrix containing all the prototype correspondences $\{C_i\}_{i=1}^{N}$.

After the parameters $\alpha$ are estimated, $N$ image warps are synthesized in the same manner as described in Section 5.3 using flow-reorientation and warping:

$$I_i^{warp} = \mathbf{W}(I_i, \mathbf{W}(C_i - \sum_{i=1}^{N} \alpha_i C_i, C_i)) \tag{10}$$

The final step in analysis is to estimate the values of $beta$ as the values which minimize

$$\|I^{novel} - \textstyle\sum_{i=1}^{N} \beta_i I_i^{warp}\| \quad \texttt{subject to} \quad \beta_i > 0 \;\; \forall i \quad \texttt{and} \quad \textstyle\sum_{i=1}^{N} \beta_i = 1 \tag{11}$$

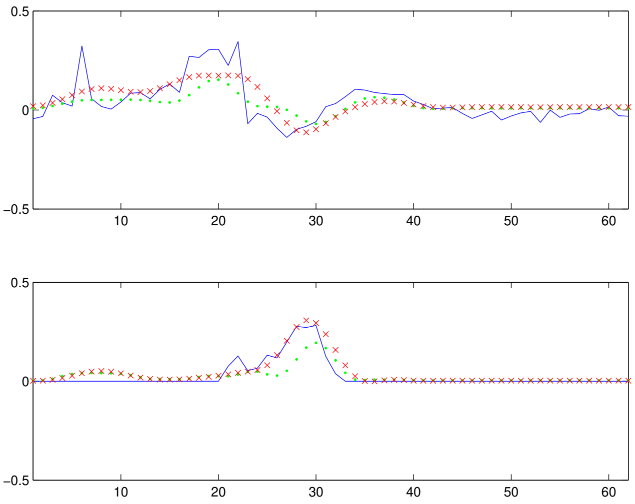


Figure 3: Top: The analyzed trajectory for $\alpha_{12}$ (in solid blue), compared with the synthesized trajectory for $\alpha_{12}$ before training (in green dots) and after training (in red crosses). Bottom: Same as above, but the trajectory is for $\beta_{28}$. Both trajectories are from the word `tabloid`.

The nonnegativity constraint above on the $\beta_i$ parameters is needed because pixel values cannot be negative. The convexity constraint is needed to normalize the $beta$ weights computed for each frame, which prevents brightness flickering in the final images. Equation 11 involves the minimization of a quadratic cost function subject to constraints, and is thus solveable using quadratic programming methods. In this work, we use the Matlab function `quadprog`, which is an interior point algorithm.

Each utterance in the corpus is analyzed with respect to the 92-dimensional MMM created in Section 5.2, yielding a set of $z_t = (\alpha_t, \beta_t)$ parameters for each utterance. Analysis takes on the order of 15 seconds per frame (on a circa 1998 450 Mhz Pentium II). Shown in Figure 3 in solid blue are example analyzed trajectories for $\alpha_{12}$ and $\beta_{28}$ computed for the word `tabloid`.

## 6 Trajectory Synthesis

### 6.1 Overview

The goal of trajectory synthesis is to map from an input *phone stream* to a *trajectory* of $(\alpha, \beta)$ parameters in MMM space. After the parameters are synthesized, Equation 7 from Section 5.3 is used to create the final visual stream that represents the talking face.

The phone stream is a stream of phonemes $\{P_t\}$ representing that phonetic transciption of the utterance. For example, the word `one` may be represented by a phone stream $\{P_t\}_{t=1}^{15}$ = `(\w\, \w\, \w\, \w\, \uh\, \uh\, \uh\, \uh\, \uh\, \uh\, \n\, \n\, \n\, \n\, \n\)`. Each element in phone stream represents one image frame. We define $T$ to be the length of the entire utterance in frames.

Since the audio is aligned, it is possible to examine all the flow and texture parameters for any particular phoneme. Shown in Figure 6.1 are histograms for the $\alpha_1$ parameter for the `\w\`, `\m\`, `\aa\` and `\ow\` phones. Evaluation of the analyzed parameters from the corpus reveals that parameters representing the same phoneme tend to *cluster* in MMM space. We represent each phoneme $p$ is mathematically as a multidimensional Gaussian with mean $\mu_p$ and diagonal covariance $\Sigma_p$. Separate means and covari-

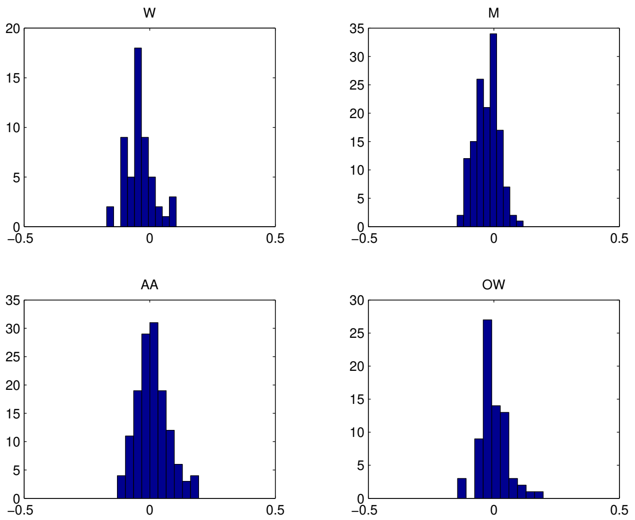


Figure 4: Histograms for the $\alpha_1$ parameter for the `\w\`, `\m\`, `\aa\` and `\ow\` phones.

ances are estimated for the flow and texture parameters [1].

The trajectory synthesis problem is framed mathematically as a *regularization* problem [19]: The goal is to synthesize a trajectory $y$ which minimizes an objective function $E$ consisting of a *target term* and a *smoothness term*:

$$E = \underbrace{(y-\mu)^T D^T (\Sigma^T \Sigma)^{-1} D (y-\mu)}_{target term} + \lambda \underbrace{y^T W^T W y}_{smoothness} \qquad (12)$$

The target term consists of the relevant means $\mu$ and covariances $\Sigma$ constructed from the phone stream:

$$\mu = \begin{bmatrix} \mu_{P_t} \\ \vdots \\ \mu_{P_T} \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{P_t} & & \\ & \ddots & \\ & & \Sigma_{P_T} \end{bmatrix} \qquad (13)$$

The matrix $D$ is a duration-weighting matrix which emphasizes the shorter phonemes and de-emphasizes the longer ones, so that the objective function is not heavily skewed by the phonemes of longer duration:

$$D = \begin{bmatrix} \sqrt{I - \frac{D_{P_1}}{T}} & & & \\ & \sqrt{I - \frac{D_{P_2}}{T}} & & \\ & & \ddots & \\ & & & \sqrt{I - \frac{D_{P_T}}{T}} \end{bmatrix} \qquad (14)$$

One possible smoothness term consists of the first order difference operator:

$$W = \begin{bmatrix} -I & I & & & \\ & -I & I & & \\ & & & \ddots & \\ & & & -I & I \end{bmatrix} \qquad (15)$$

Higher orders of smoothness are formed by repeatedly multiplying $W$ with itself: $W^T W^T W W$, $W^T W^T W^T W W W$, and so on.

Finally, the regularizer $\lambda$ determines the tradeoff between both terms.

Taking the derivative of Equation 12 and minimizing yields the following equation for synthesis:

$$(D^T \Sigma^T \Sigma^{-1} D + \lambda W^T W) y = D^T (\Sigma^T \Sigma)^{-1} D \mu \qquad (16)$$

Given known means $\mu$, covariances $\Sigma$, and regularizer $\lambda$, synthesis is simply a matter of plugging into Equation 16 and solving for $y$ using Gaussian elimination. This is done separately for the flow and the texture parameters.

## 6.2 Training

The means $\mu_p$ and covariances $\Sigma_p$ for each phone $p$ are initialized directly from the data using sample means and covariances. However, the sample estimates tend to average out the mouth movement so it looks under-articulated. As a result, there is a need to adjust the means and variances to better reflect the training data.

Gradient descent learning [6] is employed to adjust the mean and covariances. First, the Euclidean error metric is chosen to represent the error between the original utterance $z$ and the synthetic utterance $y$:

$$E = (z-y)^T (z-y) \qquad (17)$$

The parameters $\{\mu_p, \Sigma_p\}$ need to be changed to minimize this objective function $E$. The chain rule may be used to derive the relationship between E and the parameters:

$$\frac{\partial E}{\partial \mu_i} = \left(\frac{\partial E}{\partial y}\right)^T \left(\frac{\partial y}{\partial \mu_i}\right) \qquad (18)$$

$$\frac{\partial E}{\partial \sigma_{ij}} = \left(\frac{\partial E}{\partial y}\right)^T \left(\frac{\partial y}{\partial \sigma_{ij}}\right) \qquad (19)$$

$\frac{\partial E}{\partial y}$ may be obtained from Equation 17:

$$\frac{\partial E}{\partial y} = -2(z-y) \qquad (20)$$

Since $y$ is defined according to Equation 16, we can take its derivative to compute $\frac{\partial y}{\partial \mu_i}$ and $\frac{\partial y}{\partial \sigma_{ij}}$:

$$(D^T \Sigma^T \Sigma^{-1} D + \lambda W^T W) \frac{\partial y}{\partial \mu_i} = D^T (\Sigma^T \Sigma)^{-1} D \frac{\partial \mu}{\partial \mu_i} \qquad (21)$$

$$(D^T \Sigma^T \Sigma^{-1} D + \lambda W^T W) \frac{\partial y}{\partial \sigma_{ij}} = 2 D^T \Sigma^{-1} \Sigma^{-1} \frac{\partial \Sigma}{\partial \sigma_{ij}} \Sigma^{-1} D (y-\mu) \qquad (22)$$

Finally, gradient descent is performed by changing the old values of the parameters according to the computed gradient:

$$\mu^{new} = \mu^{old} - \eta \frac{\partial E}{\partial \mu} \qquad (23)$$

$$\Sigma^{new} = \Sigma^{old} - \eta \frac{\partial E}{\partial \Sigma} \qquad (24)$$

[1] Technically, since the texture parameters are non-negative, they are best modelled using Gamma distributions not Gaussians. In that case, Equation 12 needs to be re-written for Gamma distributions. In practice, however, we have found Gaussians to work well enough to merit their use here for texture parameters.

Cross-validation sessions were performed to evaluate the appropriate value of $\lambda$ and the correct level of smoothness $W$ to use. The learning rate $\eta$ was set to 0.001 for all trials, and 10 iterations performed. Comparison between batch and online updates indicated that online updates perform better, so this method used throughout training. Testing was performed on a set composed of 1-syllable words, 2-syllable words, and sentences *not contained in the training set.* Shown in Figure **??** are the testing errors for various values of $\lambda$ and smoothness kernel $W$. The Euclidean norm between the synthesized trajectories and the original trajectories was used to measure error. The results indicate that the optimal smoothness kernel is $W = 4$ and the optimal regularizer is $\lambda = 10000$. Figure 3 depicts synthesized trajectories for the $\alpha_{12}$ and $\beta_{28}$ parameters before training (in green dots) and after training (in red crosses) for these optimal values of $W$ and $\lambda$.

# 7 Discussion

We have synthesized numerous examples using our system, spanning the entire range of 1-syllable, 2-syllable, short sentences, and long sentences. In addition, we have synthesized songs and foreign speech examples.

Coarticulation effects in our system are regulated via the magnitude of the variance $\Sigma$ for each phoneme. Coarticulation refers to the effect of previous and following phonemes on the current phone. In our case, the magnitude of the variance determines the importance with which a trajectory must pass through that phoneme location. Small variance means the trajectory *must* pass through that region in phoneme space, and hence neighboring phonemes have little coarticulatory effect. On the other hand, large variance means the trajectory has a lot of flexibility in choosing a path through a particular phonetic region, and hence it will choose to pass through regions which are closer to a phoneme's neighbors. The phoneme will thus experience large coarticulatory effects.

Phonetic dynamics refers to the *speed* with which a particular phonetic region is traversed. In our systen, phonetic dynamics are regulated via the input phone stream, which determines how long each phoneme lasts for. Given the locations $\mu$ and variance $\Sigma$ for each phoneme in a trajectory, Equation 12 determines the optimal speed to pass through a phonetic region which balances nearness to the phoneme with smoothness of the overall trajectory. In general, we find the trajectories speed up in regions of small duration and small variance (ie plosives), while they slow down in regions of large duration and large variance (ie silences).

There are two ways to evaluate the synthesis results. The first, shown in Figure **??**, is by comparing the synthesized trajectories with the original trajectories using the Euclidean norm. A perhaps more useful method of comparison is to perform visual "Turing tests" with human subjects to see whether they can differentiate between real sequences and synthetic ones. Experiments in this regard are underway, and preliminary results are encouraging. Specifically, we perform:

1. **Hard Turing Test:** In this experiment, subjects will be asked to view *one* visual sequence, and asked to identify whether it is synthetic or real.

2. **Easy Turing Test:** In this experiment, subjects will be asked view real and synthetic visual sequences, and asked to identify explicitly which one is synthetic and which one is real.

3. **Intelligibility Test:** Normal-hearing subjects will be asked to lip-read short sequences, and confusion matrices will compare the lipreading performance on the real sequences vs. the synthetic ones.

# 8 Further Work

We mention two clear avenues of exploration.

Clearly the face is used as a conduit to transmit emotion, so one possible avenue to explore is the synthesis of speech under various emotional states. It is possible to record various corpora under different emotional states and create MMMs for each state. During synthesis, the appropriate MMM is selected. An open question to explore is *emotional dynamics*: how does one transition from a happy MMM to a sad MMM?

Another avenue to explore is the generalization of our approach to 3D. It is possible to envision a real-time 3D scanner that is capable of recording a 3D video corpus of speech. In addition, techniques such as those presented in [20] [32] [8] can be used to map a 2D video corpus into 3D. Afterwards the techniques presented here may be used to create and animate a 3D MMM.

# References


[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[2] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92 Proceedings*, pages 35–42, Chicago, IL, 1992.

[3] J.R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, Princeton, New Jersey, April 1990.

[4] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. Technical Report 1431, MIT AI Lab, 1993.

[5] David Beymer. Vectorizing face images by interleaving shape and texture computations. A.I. Memo No. 1537, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1995.

[6] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[7] A. Black and P. Taylor. *The Festival Speech Synthesis System*. University of Edinburgh, 1997.

[8] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 187–194, Los Angeles, 1999. Addison Wesley Longman.

[9] Matthew Brand. Voice puppetry. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 21–28, Los Angeles, 1999. Addison Wesley Longman.

[10] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *SIGGRAPH '97 Proceedings*, Los Angeles, CA, August 1997.

[11] N.M. Brooke and S.D. Scott. Computer graphics animations of talking faces based on stochastic models. In *Intl. Symposium on Speech, Image Processing, and Neural Networks*, Hong Kong, April 1994.

[12] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. on Communications*, COM-31(4):532–540, April 1983.

[13] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93 Proceedings*, pages 279–288, Anaheim, CA, August 1993.

[14] M. M. Cohen and D. W. Massaro. Modeling coarticulation in synthetic visual speech. In N. M. Thalmann and D. Thalmann, editors, *Models and Techniques in Computer Animation*, pages 139–156. Springer-Verlag, Tokyo, 1993.

[15] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989.

[16] E. Cosatto and H. Graf. Sample-based synthesis of photorealistic talking heads. In *Proceedings of Computer Animation '98*, pages 103–110, Philadelphia, Pennsylvania, 1998.

[17] Tony Ezzat and Tomaso Poggio. Facial analysis and synthesis using image-based models. In *Proceedings Int. Conf. on Face and Gesture Recognition*, pages 116–121, Killington, Vermont, 1996.

[18] Tony Ezzat and Tomaso Poggio. Miketalk: A talking facial display based on morphing visemes. In *Proc. Computer Animation Conference*, pages 96–102, Philadelpha, Pennsylvania, 1998.

[19] F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers, and basis functions: From regularization to radial, tensor, and additive splines. Technical Report 1430, MIT AI Lab, June 1993.

[20] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH '98 Proceedings*, pages 55–66, Orlando, FL, 1998.

[21] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[22] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld. The SPHINX-II speech recognition system: an overview (http://sourceforge.net/projects/cmusphinx/). *Computer Speech and Language*, 7(2):137–148, 1993.

[23] M. Jones and T. Poggio. Multidimensional morphable models: A framework for representing and maching object classes. In *Proceedings of the International Conference on Computer Vision*, Bombay, India, 1998.

[24] A. Lanitis, C.J. Taylor, and T.F. Cootes. A unified approach to coding and interpreting face images. In *Proceedings of the International Conference on Computer Vision*, pages 368–373, Cambridge, MA, June 1995.

[25] S. Y. Lee, K. Y. Chwa, S. Y. Shin, and G. Wolberg. Image metemorphosis using snakes and free-form deformations. In *SIGGRAPH '92 Proceedings*, pages 439–448, 1992.

[26] Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *SIGGRAPH '95 Proceedings*, pages 55–62, Los Angeles, California, August 1995.

[27] B. LeGoff and C. Benoit. A text-to-audiovisual-speech synthesizer for french. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Philadelphia, USA, October 1996.

[28] T. Masuko, T. Kobayashi, M. Tamura, J. Masubuchi, and K. Tokuda. Text-to-visual speech synthesis based on parameter generation from hmm. In *ICASSP*, 1998.

[29] E. Moulines and F. Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, 9:453–467, 1990.

[30] F. I. Parke. *A parametric model of human faces*. PhD thesis, University of Utah, 1974.

[31] A. Pearce, B. Wyvill, G. Wyvill, and D. Hill. Speech and expression: A computer solution to face animation. In *Graphics Interface*, 1986.

[32] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98 Proceedings*, Orlando, FL, 1998.

[33] Sam Roweis. EM algorithms for PCA and SPCA. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[34] K.C. Scott, D.S. Kagels, S.H. Watson, H. Rom, J.R. Wright, M. Lee, and K.J. Hussey. Synthesis of speaker facial movement to match selected speech sequences. In *Proceedings of the Fifth Australian Conference on Speech Science and Technology*, volume 2, pages 620–625, December 1994.

[35] Michael E. Tipping and Christopher M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[36] K. Waters. A muscle model for animating three-dimensional facial expressions. In *SIGGRAPH '87 Proceedings*, pages 17–24, July 1987.

[37] S.H. Watson, J.R. Wright, K.C. Scott, D.S. Kagels, D. Freda, and K.J. Hussey. An advanced morphing algorithm for interpolating phoneme images to simulate speech. Jet Propulsion Laboratory, California Institute of Technology, 1997.

[38] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA., 1990.

# A Appendix: Optical Flow

Optical flow [21] was originally formulated in the context of measuring the apparent motion of objects in images. This apparent motion is captured as a two-dimensional array of displacement vectors, in the same exact format shown in Equation 1.

There is currently a vast literature on this subject (see for example [1] for a recent review), and several different methods for computing flow have been proposed and implemented. In this work, we utilize the *coarse-to-fine, gradient-based* optical flow algorithms developed by [3]. These algorithms compute the desired flow displacements using the spatial and temporal image derivatives. In addition, they embed the flow estimation procedure in a multiscale pyramidal framework [12], where initial displacement estimates are obtained at coarse resolutions, and then propagated to higher resolution levels of the pyramid.

# B Appendix: Flow Concatenation

Given a series of consecutive images $I_0, I_1, \dots I_n$, we would like to construct the correspondence map $C_{0(n)}$ relating $I_0$ to $I_n$. We focus on the case of the 3 images $I_{i-1}, I_i, I_{i+1}$ since the concatenation algorithm is simply is an iterative application of this 3-frame base case. Optical flow is first computed between the consecutive frames to yield $C_{(i-1)i}, C_{i(i+1)}$. Note that it is not correct to construct $C_{(i-1)(i+1)}$ as the simple addition of $C_{(i-1)i} + C_{i(i+1)}$ because the two flow fields are with respect to two different reference images. Vector addition needs to be performed with respect to a common origin.

Our concatenation thus proceeds in two steps: to place all vector fields in the same reference frame, the *correspondence map* $C_{i(i+1)}$ *itself* is warped *backwards* [38] along $C_{(i-1)i}$ to create $C^{warped}_{i(i+1)}$. Now $C^{warped}_{i(i+1)}$ and $C_{(i-1)i}$ are both added to produce an approximation to the desired concatenated correspondence:

$$C_{(i-1)(i+1)} = C_{(i-1)i} + C^{warped}_{i(i+1)} \tag{25}$$

A procedural version of our backwarp warp is shown in figure 5. `BILINEAR` refers to bilinear interpolation of the 4 pixel values closest to the point `(x,y)`

```
for j = 0...height,
   for i = 0...width,
      x = i + dx(i,j);
      y = j + dy(i,j);
      I^warped(i,j) = BILINEAR (I, x, y);
```

Figure 5: **BACKWARD WARP** algorithm, which warps I backwards along `dx` and `dy` to produce $I^{warped}$

# C Appendix: Forward Warping

Forward warping may be viewed as "pushing" the pixels of an image $I$ along the computed flow vectors $C$. Notationally, we denote the forward warping operation as an operator $\mathbf{W}(I, C)$ that operates on an image $I$ and a correspondence map $C$, producing a warped image $I^{warped}$ as final output. A procedural version of our forward warp is shown in Figure 6.

Forward warping produces *black holes* which occur in cases where a destination pixel was not filled in with any source pixel value. The main reason for this is that the computed optical flow displacements exhibit nonzero divergence, particularly around the region where the mouth is expanding. To remedy this, a hole-filling algorithm first proposed by Chen & Williams [13] was adopted. The algorithm pre-fills the destination images with a special reserved background color. After performing the forward warp, the hole-filling algorithm traverses the destination image in rasterized order and fills in the holes by interpolating linearly between their non-hole endpoints. For simplicity of notation, the hole-filling procedure is subsumed into our operator $\mathbf{W}$.

It is also possible to forward warp a correspondence map $C'$ along another correspondence $C$, which we denote as

$\mathbf{W}(C', C)$. In this scenario, the $x$ and $y$ components of $C'(\mathbf{p}) = \{d'_x(\mathbf{p}), d'_y(\mathbf{p})\}$ are treated as separate images, and warped individually along $C$: $\mathbf{W}(dx', C)$ and $\mathbf{W}(dy', C)$. Hole-filling is performed on the warped $x$ and $y$ components separately.

```
for j = 0...height,
   for i = 0...width,
      x = ROUND (i + αdx(i,j) );
      y = ROUND (j + αdy(i,j) );
      if (x,y) are within the image
         I^warped(x,y) = I(i,j);
```

Figure 6: **FORWARD WARP** algorithm, which warps $I$ forward along flow vectors $\alpha$dx and $\alpha$dy to produce $I^{warped}$.