ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING
DEPARTMENT OF BRAIN AND COGNITIVE SCIENCES

A.I. Memo No. 1658 May 1999
C.B.C.L Paper No. 173

Visual Speech Synthesis by Morphing Visemes

Tony Ezzat and Tomaso Poggio

Abstract

We present MikeTalk, a text-to-audiovisual speech synthesizer which converts input text into an audiovisual speech stream. MikeTalk is built using visemes, which are a small set of images spanning a large range of mouth shapes. The visemes are acquired from a recorded visual corpus of a human subject which is specifically designed to elicit one instantiation of each viseme. Using optical flow methods, correspondence from every viseme to every other viseme is computed automatically. By morphing along this correspondence, a smooth transition between viseme images may be generated. A complete visual utterance is constructed by concatenating viseme transitions. Finally, phoneme and timing information extracted from a text-to-speech synthesizer is exploited to determine which viseme transitions to use, and the rate at which the morphing process should occur. In this manner, we are able to synchronize the visual speech stream with the audio speech stream, and hence give the impression of a photorealistic talking face.

This report describers research done at the Center for Biological & Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. This research was sponsored by the Office of Naval Research under contract No. N00014-93-1-0385 and contract No. N00014-95-1-0600. Partial support was also provided by Daimler-Benz AG, Eastman Kodak, Siemens Corporate Research, Inc., and AT&T.

1 Introduction

The goal of the work described in this paper is to develop a text-to-audiovisual speech synthesizer called MikeTalk. MikeTalk is similar to a standard text-to-speech synthesizer in that it converts text into an audio speech stream. However, MikeTalk also produces an accompanying visual stream composed of a talking face enunciating that text. An overview of our system is shown in Figure 1.

Text-to-visual (TTVS) speech synthesis systems are attracting an increased amount of interest in the recent years, and this interest is driven by the possible deployment of these systems as visual desktop agents, digital actors, and virtual avatars. In addition, they may also have potential uses in special effects, very low bitrate coding schemes (MPEG4), and would also be of interest to psychologists who wish to study visual speech production and perception.

In this work, we are particularly interested in building a TTVS system where the facial animation is *video-realistic*: that is, we desire our talking facial model to look as much as possible as if it were a videocamera recording of a human subject, and not that of a cartoon-like human character.

In addition, we choose to focus our efforts on the issues related to the synthesis of the visual speech stream, and not on audio synthesis. For the task of converting text to audio, we have incorporated into our work the Festival speech synthesis system, which was developed by Alan Black, Paul Taylor, and colleagues at the University of Edinburgh [6]. Festival is freely downloadable for non-commercial purposes, and is written in a modular and extensible fashion, which allows us to experiment with various facial animation algorithms.

The Festival TTS system, as with most speech synthesizers, divides the problem of converting text to speech into two sub-tasks, shown in pink in Figure 1: first, a natural language processing (NLP) unit converts the input text into a set of output streams that contain relevant phonetic, timing, and other intonational parameters. Second, an audio signal processing unit converts the NLP output streams into an audio stream in which the input text is enunciated.

Within this framework, our goal in this work, as depicted in red in Figure 1, is two-fold: first, to develop a *visual speech module* that takes as input the phonetic and timing output streams genererated by Festival's NLP unit, and produces as output a visual speech stream of a face enunciating the input text. Secondly, to develop a *lip-sync module* that synchronizes the playback of the audio and visual streams.

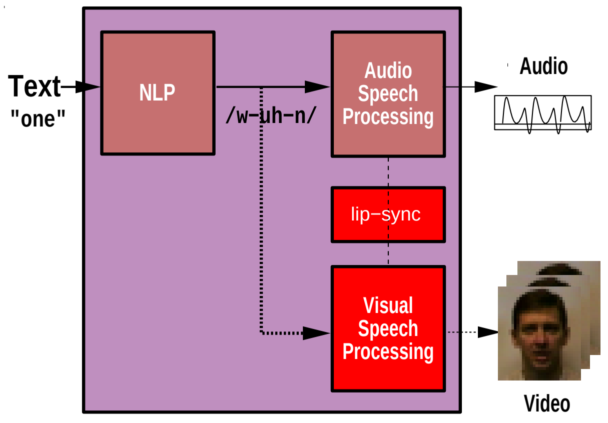

Figure 1. Overview of the MikeTalk TTVS system.

We discuss the facial animation module in Sections 2 through 6, and the lip-sync module in Section 7.

2 Background and Motivation

The main research issue underlying the construction of a TTVS visual stream is the nature of the facial model to use. One approach is to model the face using traditional *3D modeling* methods. Parke [22] was one of the earliest to adopt such an approach by creating a polygonal facial model. The face was animated by interpolating the location of various points on the polygonal grid. Parke's software and topology were subsequently given new speech and expression control software by Pearce, Wyvill, Wyvill, & Hill [23]. With this software, the user could type a string of phonemes that were then converted to control parameters which changed over time to produce the animation sequence. Each phoneme was represented in a table by 11 control parameters, and the system made a transition between two phonemes by interpolating between the control parameters. Recent work on TTVS systems that is based on Parke's models include the work of Cohen & Massaro [10] and LeGoff & Benoit [16].

To increase the visual realism of the underlying facial model, the facial geometry is frequently scanned in using three-dimensional or laser scanners such as those manufactured by Cyberware. Additionally, a texture-map of the face extracted by the Cyberware scanner may be mapped onto the three-dimentional geometry [15]. More advanced dynamic, muscle-based animation mechanisms were demonstrated by Waters [26].

Despite these improvements, the generated facial animations still lack video realism. One Cyberware texture scan alone does not suffice to capture the complex time-varying appearance of a human face, and usually is not able to capture the 3D structure of human hair. Furthermore, overlaying the texture scan on top of a 3D polygonal or muscle-based model exposes the deficiencies of these models in terms of their ability to animate human motion.

An alternative 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 speech synthesis [19].

Bregler, Covell, el al. [7] describe such an image-based approach in which the talking facial model is composed of a set of audiovisual sequences extracted from a larger audiovisual corpus. 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. To handle all the possible triphone contexts, however, the system requires a library with tens and possibly hundreds of thousands of images, which seems to be an overly-redundant sampling of human lip configurations.

Cosatto and Graf [11] describe an approach which attempts to reduce this redundancy by parameterizing the space of lip positions. 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 trajectories in this imposed lip space. The trajectories are created using Cohen-Massaro's coarticulation rules [10]. If the lip space is not populated densely, the animations produced may be jerky. The authors use a pixelwise blend to smooth the transitions between the lip images, but this can produce undesirable ghosting effects.

An even simpler image-based lip representation was used by Scott, Kagels, et al. [24] [27]. Their facial model is composed of a set of 40-50 *visemes*, which are the visual manifestation of phonemes. To animate the face, a 2D morphing algorithm is developed which is capable of transitioning smoothly between the various mouth shapes. While this produces smooth transitions between the visemes, the morphing algorithm itself requires considerable user intervention, making the process tedious and complicated.

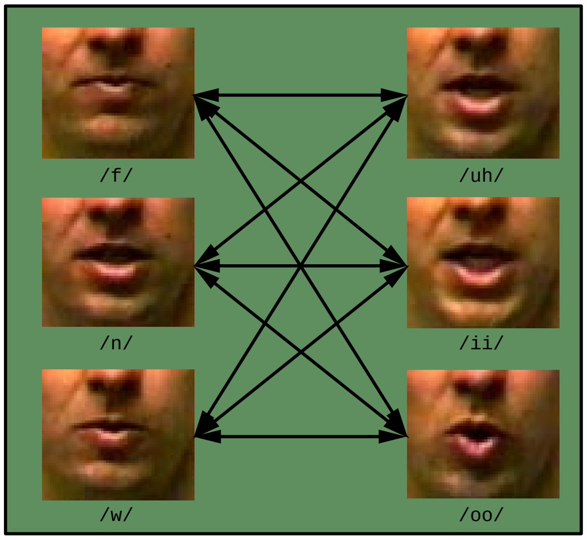

Figure 2. The MikeTalk facial model.

Our work explores further the use of this viseme-morphing representation for synthesis of human visual speech. Instead of using a manual morphing method, however, we employ a method developed by Beymer, Shashua, Poggio [5]. This morphing algorithm required little or no user intervention, and was shown to be capable of modeling rigid facial transformations such as pose changes, as well as non-rigid transformations such as smiles.

3 Our Facial Model

Our facial modelling approach may best be summarized as an *image-based*, *morphing* method, and is close in spirit to the work of [5] [24] [27]. We summarize the three main aspects of our facial model as follows:

Corpus and Viseme Acquisition: First, a visual corpus of a subject enunciating a set of key words is initially recorded. Each key word is chosen so that it visually instantiates one American English phoneme. Because there are 40-50 American English phonemes [20], the subject is asked to enunciate 40-50 words. One single image for each phoneme is subsequently identified and manually extracted from the corpus sequence. In this work, we use the term *viseme* to denote the lip image extracted for each phoneme. A few of the viseme images are depicted in Figure 2.

Viseme Morph Transformation: Secondly, we construct, in a manner described in more detail below, a *morph transformation* from each viseme image to every other viseme image. This transformation allows us to smoothly and realistically transition between any two visemes, creating intermediate lip shape images between the two endpoints. For N visemes in our final viseme set, we define N^2 such transformations. The arrows between the viseme images in Figure 2 are a figurative depiction of these transformations.

Concatenation Finally, to construct a novel visual utterance, we *concatenate* viseme morphs. For example, in terms of Figure 2, the utterance for the word `one` is constructed by morphing from `\w\` viseme to the `\uh\` viseme, followed by a morph from the `\uh\` viseme to the `\n\` viseme. For any input text, we determine the appropriate sequence of viseme morphs to make, as well as the rate of the transformations by utilizing the output of the natural language processing unit of the Festival TTS system.

In a graph-theoretic sense, our facial model may be viewed as an *N-node clique*, where each node represents one viseme, and the directed edges between nodes represent the N^2 viseme transformations. From an animator's perspective, the visemes in our model represent *keyframes*, and our transformations represent a method of *interpolating* between them.

In the following sections, we describe the various aspects of our approach in detail.

4 Corpus and Viseme Data Acquisition

The basic underlying assumption of our facial synthesis approach is that the complete set of mouth shapes associated with human speech may be reasonably spanned by a finite set of *visemes.* The term *viseme* itself was coined initially by Fisher [12] as an amalgamation of the words "visual" and "phoneme". To date, there has been no precise definition for the term, but in general it has come to refer to a speech segment that is *visually* contrastive from another. In this work, we define a viseme to be a *static lip shape image that is visually contrastive from another.*

Given the assumption that visual speech is spanned by a set of visemes, we would like to design a particular visual corpus which elicits one instantiation for each viseme. One possible strategy to adopt is to assume a *one-to-one mapping* between the set of phonemes and the set of visemes, and design the corpus so that there is at least one word uttered which instantiates each phoneme.

This one-to-one strategy is a reasonable approach in light of the fact that we plan on using the Festival TTS system to produce the audiovisual sequence. In doing so, Festival's NLP unit will produce a stream of phonemes corresponding to the input text. Consequently, there is a need to *map* from the set of phonemes used by the TTS to a set of visemes so as to produce the visual stream. A one-to-one mapping between phonemes and visemes thus ensures that a unique viseme image is associated with each phoneme label. Since most speech textbooks and dictionaries contain a list of phonemes and example words which instantiate them, the corpus may thus be designed to contain those example words.

Our recorded corpus is shown in Figure 3. The example words uttered are obtained from [20], and are generally categorized into example words which instantiate consonantal, monophthong vocalic, and diphthong vocalic phonemes. After the whole corpus is recorded and digitized, one lip image is extracted as an instance of that viseme. In general, the viseme image extracted was chosen as the image occurring at the point where the lips were judged to be at their *extremal* position for that sound.

It should be noted that diphthongs are treated in a special manner in this work. Since diphthongs are vocalic phonemes which involve a quick transition between *two* underlying vowel nuclei, it was decided that two viseme images were necessary to model that diphthong visually: one to represent the first vowel nucleus, and the other to represent the second. Consequently, we extract two images for every diphthong from the recorded corpus.

The one-to-one mapping strategy thus leads to the extraction of 52 viseme images in all: 24 representing the consonants, 12 representing the monophthongs, and 16 representing the diphthongs.

Since a large number of the extracted visemes looked similar, it was decided to further reduce the viseme set by grouping them together. This was done in a subjective manner, by comparing the viseme images visually to assess their similarity. This is in keeping with the current viseme literature, which indicates that the mapping between phonemes and visemes is, in fact, *many-to-one*: there are many phonemes which look alike visually, and hence they fall into the same visemic category. This is particularly true, for example, in cases where two sounds are identical in manner and place of articulation, but differ only in voicing characteristics. For example, `\b\` and `\p\` are two bilabial stops which differ only in the fact that the former is voiced while the latter is voiceless.This difference, however, does not manifest itself visually, and hence the two phonemes should be placed in the same visemic category. In grouping the visemes subjectively, the authors were guided by the conclusions in [21] for the case of consonantal visemes, and in [18] for the case of vocalic visemes.

It is also important to point out that that the map from phonemes to visemes is also *one-to-many*: the same phoneme can have many different visual forms. This phenomenon is termed *coarticulation*, and it occurs because the neighboring phonemic context in which a sound is uttered influences the lip shape for that sound. For example, the viseme associated with `\t\` differs depending on whether the speaker is uttering the word `two` or the word `tea`. In the former case, the `\t\` viseme assumes a rounded shape in anticipation of the upcoming `\uu\` sound, while the latter assumes a more spread shape in anticipation of the upcoming `\ii\` sound. To date, and largely due to the large number of influencing factors, the nature and scope of coarticulation remains an open research problem. The reader is referred to [10] for an in-depth discussion on the theories behind coarticulation, and to [21] for a study on consonantal perception in various vocalic contexts. At the present stage of our work, we have decided for the sake of simplicity to ignore coarticulation effects,

The final reduced set of visemes are shown in Figures 4 and 5. Note that while our figures display only the region around the mouth, our viseme imagery capture the entire face.

In all, there are 16 final visemes. Six visemes represent the 24 consonantal phonemes. Seven visemes represent the 12 monophthong phonemes. In the case of diphthongs, it was found that all vowel nuclei could be represented by corresponding monophthong visemes. The only exception to this occurred in the case of two nuclei: the second nucleus of the `\au\` dipththong, which we call the `\w-au\` viseme, and the first nucleus of the `\ou\` dipththong, which we call the `\o-ou\` viseme. Finally, one extra viseme was included to represent silence, which we call `\#\`.

5 Viseme Morphing

In constructing a visual speech stream, it is not sufficient to simply display the viseme images in sequence. Doing so would create the disturbing illusion of very abrupt mouth movement, since the viseme images differ significantly from each other in shape. Consequently, a mechanism of transitioning from each viseme image to every other viseme image is needed, and this transition must be smooth and realistic.

In this work, a *morphing* technique was adopted to create this transition. Morphing was first popularized by Beier & Neely [3] in the context of generating transitions between different faces for Michael Jackson's *Black or White* music video. Given two images I_0 and I_1, morphing generates intermediate images I_α, where α is a parameter ranging from 0 to 1. These intermediate images are generated by *warping* I_0 towards I_1, *warping* I_1 towards I_0, and *cross-dissolving* the warped images to produce the final desired image. Intuitively, warping maybe viewed as an interpolation in *shape*, while cross-dissolving maybe viewed as an interpolation in *texture*.

In the following sections, we discuss each of the above steps in detail, and describe their effect on viseme images in particular.

monophthongs	
ii	bead
i	bid
e	bed
a	bad
o	body
aa	father
uh	bud
oo	baud
u	book
uu	boot
@	about
@@	bird

diphthongs	
ou	boat
ei	bait
au	bout
ai	bide
oi	boyd
e@	there
i@	near
u@	moor

consonants	
r	ride
l	light
w	wide
y	yacht
m	might
n	night
ng	song
b	bite
d	dog
g	get
p	pet
t	tea
k	key
v	veal
dh	then
z	zeal
zh	garage
f	feel
th	thin
s	seal
sh	shore
h	head
jh	jeep
ch	chore

Figure 3. The recorded visual corpus. The underlined portion of each example word identifies the target phoneme being recorded. To the left of each example word is the phonemic transcription label being used.

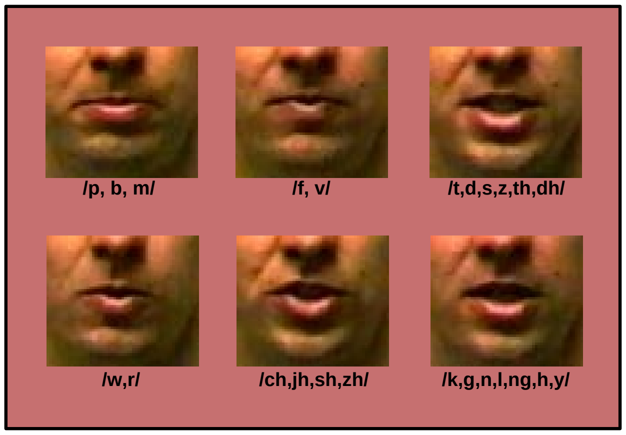

Figure 4. The 6 consonant visemes

5.1 Correspondence

As a first step, all morphing methods require the specification of *correspondence* maps $C_0 : I_0 \Rightarrow I_1$ and $C_1 : I_1 \Rightarrow I_0$ relating the images I_0 and I_1 to each other. These maps serve to ensure that the subsequent warping process preserves the desired correspondence between the geometric attributes of the objects to be morphed.

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

$$C_0(\mathbf{p_0}) = \{d_x^{0\rightarrow 1}(\mathbf{p_0}), d_y^{0\rightarrow 1}(\mathbf{p_0})\} \tag{1}$$

$$C_1(\mathbf{p_1}) = \{d_x^{1\rightarrow 0}(\mathbf{p_1}), d_y^{1\rightarrow 0}(\mathbf{p_1})\} \tag{2}$$

A pixel in image I_0 at position $\mathbf{p_0} = (x, y)$ corresponds to a pixel in image I_1 at position $(x + d_x^{0\rightarrow 1}(x, y), y + d_y^{0\rightarrow 1}(x, y))$. Likewise, a pixel in image I_1 at position $\mathbf{p_1} = (x, y)$ corresponds to a pixel in image I_0 at position $(x + d_x^{1\rightarrow 0}(x, y), y + d_y^{1\rightarrow 0}(x, y))$. As discussed in [25], two maps are usually required because one map by itself may not be one-to-one.

The specification of the correspondence maps C_0 and C_1 between the images is typically the hardest part of the morph. Previous methods [3] [24] [14] have adopted *feature-based* approaches, in which a set of high-level *shape features* common to both images is specified. The correspondences for the rest of the points are determined using interpolation.

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 addition, the process of specifying the feature regions usually requires choosing among a large number of fairly arbitrary geometric primitives such as points, line segments, arcs, circles, and meshes.

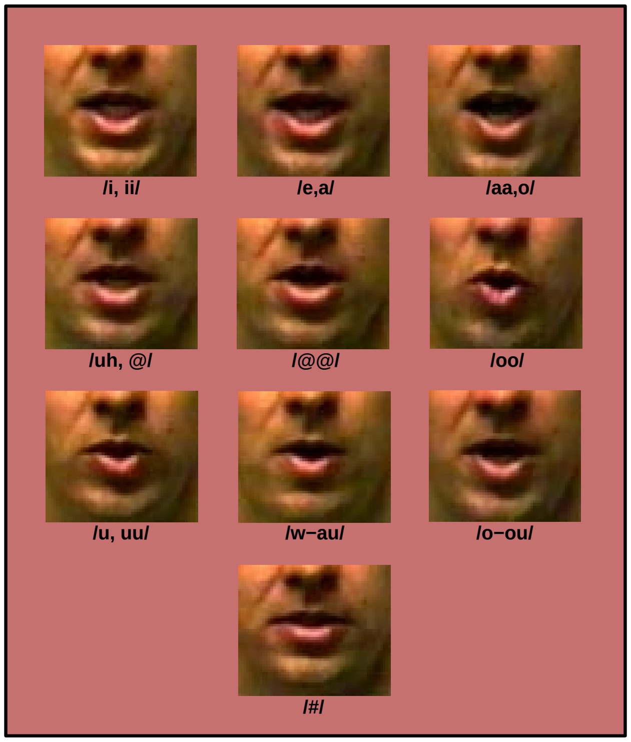

Figure 5. The 7 monophthong visemes, 2 diphthong visemes, and the silence viseme.

However, in our case the images to be morphed belong to *one single object that is undergoing motion*: a talking face. The problem of specifying correspondence between two images thus reduces to the problem of estimating the motion field of the underlying moving object! This observation, made in [5] and [9], is extremely significant because it 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.

5.2 Optical Flow

Optical flow [13] 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 Equations 1 and 2. Given two images I_0 and I_1, computing optical flow with I_0 as reference image produces a correspondence map C_0, while computing optical flow with I_1 as reference produces a correspondence map C_1.

Optical flow is thus of clear importance because it allows for the *automatic* determination of correspondence maps. In addition, since each pixel is effectively a feature point, optical flow allows us to bypass the need for choosing any of the afore-mentioned geometric feature primitives. In this sense, optical flow is said to produce *dense, pixel* correspondence.

There is currently a vast literature on this subject (see for example [2] 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 [4]. 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 [8], where initial displacement estimates are obtained at coarse resolutions, and then propagated to higher resolution levels of the pyramid. Given the size of our viseme imagery, we have found that these methods are capable of estimating displacements on the order of 5 pixels between two images.

For even larger displacements between visemes, we have found that a *flow concatenation* procedure is extremely useful in estimating correspondence. Since the original visual corpus is digitized at 30 fps, there are many intermediate frames that lie between the chosen viseme images. The pixel motions between these consecutive frames are small, and hence the gradient-based optical flow method is able to estimate the displacements. Consequently, we compute a series of consecutive optical flow vectors between each intermediate image and its successor, and *concatenate* them all into one large flow vector that defines the global transformation between the chosen visemes. Details of our flow concatenation procedure itself are found in the appendix.

It is not practical, however, to compute concatenated optical flow between viseme images that are very far apart in the recorded visual corpus. The repeated concatenation that would be involved across the hundreds of intermediate frames leads to a considerably degraded final flow. Consequently, we have found that the best procedure for obtaining good correspondences between visemes is actually a *mixture of both direct and concatenated flow computations*: typically, an intermediate frame is chosen that is simultaneously similar *in shape* to the chosen starting viseme, and also close *in distance* to the chosen ending viseme. Direct optical flow is then computed between the starting viseme and this intermediate frame, and concatenated optical flow is computed from the intermediate up to the ending viseme. The final flow from the starting viseme to the ending viseme is then itself a concatenation of both of these direct and concatenated subflows.

```
for j = 0...height,
   for i = 0...width,
      x = RND (i + $\alpha$dx(i,j) );
      y = RND (j + $\alpha$dy(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 αdx and αdy to produce I^{warped}.

5.3 Forward Warping

Given two viseme images I_0 and I_1, the first step of our morphing algorithm is to compute the correspondence map $C_0(\mathbf{p_0}) = \{d_x^{0\rightarrow 1}(\mathbf{p_0}), d_y^{0\rightarrow 1}(\mathbf{p_0})\}$ as discussed in the previous section, and then to *forward warp* I_0 along that correspondence.

Forward warping may be viewed as "pushing" the pixels of I_0 along the computed flow vectors. By scaling the flow vectors uniformly by the parameter α between 0 and 1, one can produce a series of warped intermediate images $I_0^{warped}(\alpha)$ which approximate the transformation between visemes I_0 and I_1. Formally, the forward warp W_0 and the warped image I_0^{warped} are computed as

$$W_0(\mathbf{p_0}, \alpha) = \mathbf{p_0} + \alpha C_0(\mathbf{p_0}) \tag{3}$$

$$I_0^{warped}(W_0(\mathbf{p_0}, \alpha)) = I_0(\mathbf{p_0}) \tag{4}$$

A procedural version of our forward warp is shown in Figure 6.

Several such intermediate warps are shown in Figure 7a, where I_0 is the `\m\` viseme and I_1 is the `\aa\` viseme. The black *holes* which appear in the intermediate images 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.

In symmetric fashion, it is also possible to forward warp I_1 towards I_0. First, the reverse correspondence map C_1 is estimated by computing optical flow from I_1 towards I_0. The forward warp W_1 and warped intermediate image $I_1^{warped}(\beta)$ are then generated analogously as in Equations 7 and 8:

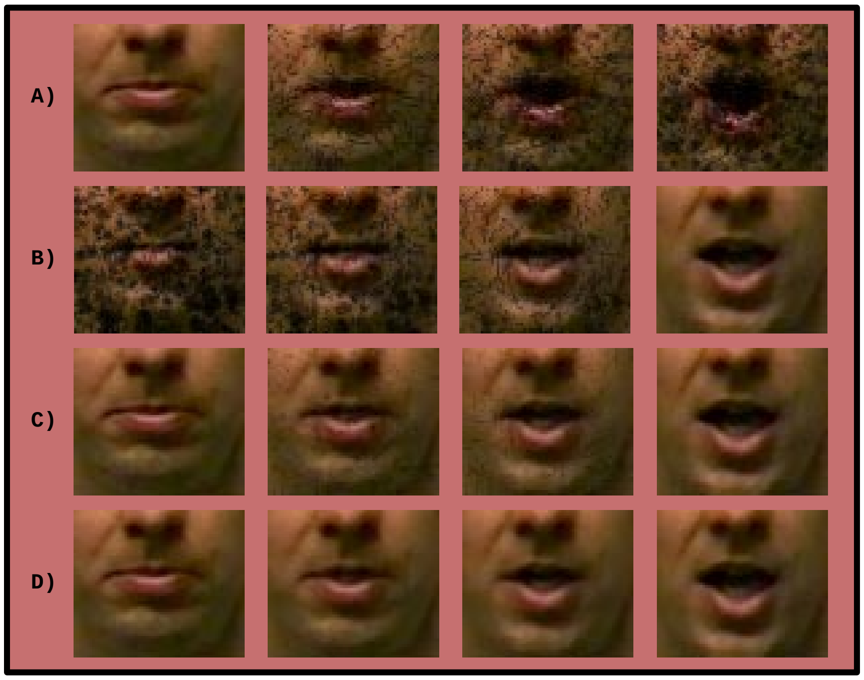

Figure 7. A) Forward warping viseme I_0 (first image) towards I_1. B) Forward warping viseme I_1 (last image) towards I_0. C) Morphing the images in I_0 and I_1 together. D) The same morphed images as in C), after hole-filling and median-filtering. Note that our morphing algorithm operates on the entire facial image, although we only show the region around the mouth for clarity.

$$W_1(\mathbf{p_1}, \beta) = \mathbf{p_1} + \beta C_1(\mathbf{p_1}) \tag{5}$$

$$I_1^{warped}(W_1(\mathbf{p_1}, \beta)) = I_1(\mathbf{p_1}) \tag{6}$$

Note that in order to align the two forward warps with respect to each other, β is set to $1 - \alpha$. In this manner, as α moves from 0 to 1, the warped intermediates $I_1^{warped}(1-\alpha)$ start out with the fully warped versions and move towards the viseme I_1. Several intermediate images of this reverse forward warp are shown in Figure 7b.

5.4 Morphing

Because forward warping can only move pixels around, *it cannot model the appearance of new pixel texture*. As is evident from the sequence in Figure 7a, a forward warp of viseme I_0 along the flow vectors of C_0 can never produce a final image that looks like viseme I_1, since viseme I_1 itself contains a large amount of novel texture from the inside of the mouth.

Morphing overcomes this "novel pixel texture" problem by combining the texture found in both forward warps. This combination is performed by scaling the warped intermediate images with respective *cross-dissolve* or *blending* parameters, and then adding to produce the final morphed image $I^{morph}(\alpha)$:

$$I^{morph}(\mathbf{p}, \alpha) = (1-\alpha) I_0^{warped}(\mathbf{p}, \alpha) + \alpha I_1^{warped}(\mathbf{p}, (1-\alpha)) \tag{7}$$

By interpolating the blending parameters the morph "fades out" the warped versions of the starting viseme and "fades in" the warped versions of the ending viseme. The blending process thus allows the two warps to be effectively combined, and the "new" pixels of the second viseme to become involved in the viseme transition itself.

Due to the presence of holes in the warped intermediates, we adopt a slightly more sophisticated blending mechanism: Whenever *one* forward warp predicts a pixel value in an intermediate image while the other leaves a hole, we set the final pixel value in the morphed image to be the same as that predicted by the single warp, without any blending. Whenever *both* forward warps predict pixel values in an intermediate image, we resort to the standard blending approach of Equation 7. Holes are detected in the images by pre-filling the destination images with a special reserved color prior to warping. Our morphed intermediate image is thus synthesized as:

$$I^{morph}(\mathbf{p}, \alpha) = \left\{ \begin{array}{ll} I_0^{warped}(\mathbf{p}, \alpha) & \textit{if } I_1^{warped}(\mathbf{p}, (1-\alpha)) \textit{ is a hole} \\ I_1^{warped}(\mathbf{p}, (1-\alpha)) & \textit{if } I_0^{warped}(\mathbf{p}, \alpha) \textit{ is a hole} \\ (1-\alpha) I_0^{warped}(\mathbf{p}, \alpha) + \alpha I_1^{warped}(\mathbf{p}, (1-\alpha)) & \textit{otherwise} \end{array} \right\}$$

Figure 7c depicts several morphed images constructed in this manner. Note that our morphing algorithm operates on the entire facial image, although we only show the region around the mouth for clarity.

As may be seen in the images of Figure 7c, there are locations for which *neither* warp predicts a pixel value, leaving a set of visible holes in the morphed images. To remedy this, a hole-filling algorithm proposed in [9] was adopted. The algorithm traverses the destination image in scanline order and fills in the holes by interpolating linearly between their non-hole endpoints. This approach works reasonably well whenever the holes are small in size, which is the case here.

In addition, the morphed image occasionally exhibits "salt-and-pepper"-type noise. This occurs whenever there is a slight mismatch in the brightness values of neighboring pixels predicted by different viseme endpoints. To remove this, we convolve the hole-filled morphed image with a 3-by-3 median filter [17]. Figure 7d shows the same set of morphed intermediates as in Figure 7c, but with the holes filled and median-filtered.

Overall, we have found that the above morphing approach produces remarkably realistic transitions between a wide variety of viseme imagery, including the typically "hard" morph transitions between open and closed mouths shown in Figure 7. It should be noted that our algorithm is essentially *linear*, in the sense that the warping and blending functions depend linearly on the parameter α. As in [5] [14], it is possible to use more complicated, non-linear warping and blending functions. We discuss the utility of using such functions briefly in the next section.

6 Morph Concatenation

To construct a visual stream in which a word or a sentence is uttered, we simply *concatenate* the appropriate viseme morphs together. For example, the word `one`, which has a phonetic transcription of \`w-uh-n`\, is composed of the two viseme morphs \`w-uh`\ and \`uh-n`\ put together and played seamlessly one right after the other. It should be noted that while this concatenation method guarantees G_0 geometric continuity, it does not guarantee continuity of speed, velocity, or acceleration. Future work must address this issue, since discontinuities in mouth motion are objectionable to the viewer. More advanced morph transition rates [14] are required to deal with this issue.

7 Audiovisual Synchronization

As discussed earlier, we have incorporated the Festival TTS system [6] into our work. In a manner that is completely analogous to our method for concatenating viseme morphs, Festival constructs the final audio speech stream by concatenating *diphones* together. Diphones are short audio sequences which sample the transitions between the middle of one phone to the middle of another phone. Given the presence of about 40-50 phonemes in English, most diphone-based TTS systems record a corpus of about 1600-2500 diphones. Shown figuratively at the top in figure 8 is an audio stream for the word `one`, which is composed of the diphones \`w-uh`\ and \`uh-n`\ concatenated together.

In order to produce a visual speech stream in synchrony with the audio speech stream, our lip-sync module first extracts the duration of each diphone D_i as computed by the audio module. We denote this duration (in seconds) as $l(D_i)$. Additionally, we compute the total duration T of the audio stream as $T = \sum_{i=1}^{N} l(D_i)$.

Next, the lip-sync module creates an intermediate stream, called the *viseme transition* stream. A viseme transition is defined to be the collection of two endpoint visemes and the optical flow correspondences between them. The lip-sync module loads the appropriate viseme transitions into the viseme transition stream by examining the audio diphones. For example, if D_i is a \`uh-n`\ diphone, then the corresponding viseme transition V_i loaded by the lip-sync module is composed of the \`uh`\ viseme, the \`n`\ viseme, and the optical flow vectors between them.

Two additional attributes are also computed for each viseme transition. The first is the duration of each viseme transition, $l(V_i)$, which is set to be equal to the duration of the corresponding diphone $l(D_i)$. For reasons that will become clear shortly, it is also useful to compute the start index in time of each viseme transition. We denote this as $s(V_i)$, and compute it as

$$s(V_i) = \left\{ \begin{array}{ll} 0 & i = 0 \\ s(V_{i-1}) + l(V_{i-1}) & \textit{otherwise} \end{array} \right\} \qquad (8)$$

Thirdly, the lip-sync module creates the *video*

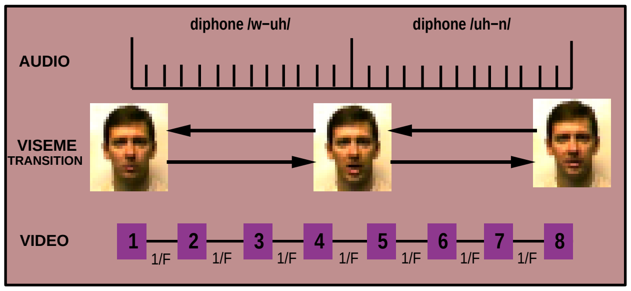

Figure 8. LIP-SYNC diagram

stream, which is composed of a sequence of *frames* which sample the chosen viseme transitions. Given a chosen frame rate F, the lip-sync module is required to create TF frames. As shown in figure 8, this naturally implies that the start index in time of the k'th frame is

$$s(F_k) = k/F \tag{9}$$

Given the start index of each viseme transition (Equation 8) and the start index of each frame (Equation 9), the lip-sync algorithm determines how to synthesize each frame F_k by setting the morph parameter α_k for that frame to be

$$\alpha_k = \frac{s(F_k) - s(V_i)}{l(V_i)} \quad \text{if } s(F_k) - s(V_i) < l(V_i) \tag{10}$$

The morph parameter is thus simply the length of time elapsed from the start of a viseme transition to the frame, divided by the entire duration of the viseme transition itself. The condition on the right hand side of Equation 10 is there to ensure that the correct viseme is chosen to synthesize a particular frame. In terms of figure 8, this condition would ensure that frames 1, 2, 3, and 4 are synthesized from the `\w-uh\` viseme transition, while frames 5, 6, 7, 8 are synthesized from the `\uh-n\` viseme transition.

As a final step, each frame is synthesized using the morph algorithm discussed in Section 5.4.

We have found that the use of TTS timing and phonemic information in this manner produces very good quality lip synchronization between the audio and the video. However, since the video sampling rate is constant and independent of mouth displacements, frequent undersampling of large mouth movement occurs, which can be very objectionable to the viewer. Accordingly, we have found it necessary to increase our sampling rate adaptively based on the optical flow vectors in each viseme transition. In a scheme similar to one suggested by [9], we pre-compute the maximal offset vector for each viseme transition, and use it to determine if our constant sampling rate undersamples a viseme transition. If so, then we add more samples to the viseme transition until the motion rate is brought down to an acceptable level. Typically, we've found that oversampling a viseme transition to ensure no more than 2.5-3.0 pixel displacements between frames leads to acceptably smooth mouth motion. When a viseme transition is oversampled, the corresponding audio diphone is lengthened to ensure that synchrony between audio and video is maintained.

8 Summary

In summary, our talking facial model may be viewed as a *collection of viseme imagery and the set of optical flow vectors defining the morph transition paths from every viseme to every other viseme.*

We briefly summarize the individual steps involved in the construction of our facial model:

Recording the Visual Corpus: First, a visual corpus of a subject enunciating a set of key words is recorded. An initial one-to-one mapping between phonemes and visemes is assumed, and the subject is asked to enunciate 40-50 words. One single image for each viseme is identified and extracted manually from the corpus. The viseme set is then subjectively reduced to a final set of 16 visemes.

Building the Flow Database: Thirdly, we build a database of optical flow vectors that specify the morph transition from each viseme image to every other viseme image. Since there are 16 visemes in our final viseme set, a total of 256 optical flow vectors are computed.

Synthesizing the New Audiovisual Sentence: Finally, we utilize a text-to-speech system [6] to convert input text into a string of phonemes, along with duration information for each phoneme. Using this information, we determine the appropriate sequence of viseme transitions to make, as well as the rate of the transformations. The final visual sequence is composed of a *concatenation* of the viseme transitions, played in synchrony with the audio speech signal generated by the TTS system.

9 Results

We have synthesized several audiovisual sentences to test our overall approach for visual speech synthesis and audio synchronization described above. Our results may be viewed by accessing our World Wide Web home page at `http://cuneus.ai.mit.edu:8000/research/miketalk/miketalk.html`. The first author may also be contacted for a video tape which depicts the results of this work.

10 Discussion

On the positive side, our use of actual images as visemes allows to achieve a significant level of *videorealism* in the final facial model. And since we only need to sample the visemes themselves, and not all the possible transitions paths between them, the size of the visual corpus which needs to be recorded is small. The transitions between the visemes are computed in an off-line manner automatically using our optical flow techniques.

The use of concatenated optical flow as a method to compute correspondence between visemes automatically seems to work very well, allowing us to overcome the difficulties associated with other correspondence methods which are manual and very tedious. In addition, the representation of a viseme transition as an optical flow vector allows us to morph as many intermediate images as necessary to maintain synchrony with the audio produced by the TTS.

Despite these advantages, there is clearly a large amount of further work to be done. First, there is no coarticulation model included in our facial synthesis method. This has the effect of producing visual speech that looks overly articulated. There is a clear need to use more sophisticated techniques to learn the dynamics of facial mouth motion.

Also, there is a clear need to incorporate into our work nonverbal mechanisms in visual speech communication such as eye blinks, eye gaze changes, eyebrow movements, and head nods. These communication mechanisms would serve to make the talking facial model more lifelike.

Since our method is image-based, we are constrained by the pose of the face in the imagery captured. Future work must address the need to synthesize the talking facial model in different poses, and various recent image-based warping and morphing methods may be explored in this regard [1] [25].

References

[1] S. Avidan, T. Evgeniou, A. Shashua, and T. Poggio. Image-based view synthesis by combining trilinear tensors and learning techniques. In *VRST '97 Proceedings*, pages 103–109, Lausanne, Switzerland, 1997.

[2] 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.

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

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

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

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

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

[8] 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.

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

[10] 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.

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

[12] C. G. Fisher. Confusions among visually perceived consonants. *Jour. Speech and Hearing Research*, 11:796–804, 1968.

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

[14] 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.

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

[16] 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.

[17] J. Lim. *Two-dimensional signal and image processing.* Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[18] A. Montgomery and P. Jackson. Physical characteristics of the lips underlying vowel lipreading performance. *Jour. Acoust. Soc. Am.*, 73(6):2134–2144, June 1983.

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

[20] J. Olive, A. Greenwood, and J. Coleman. *Acoustics of American English Speech: A Dynamic Approach.* Springer-Verlag, New York, USA, 1993.

[21] E. Owens and B. Blazek. Visemes observed by hearing-impaired and normal-hearing adult viewers. *Jour. Speech and Hearing Research*, 28:381–393, September 1985.

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

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

[24] 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.

[25] S. Seitz and C. Dyer. View morphing. In *SIGGRAPH '96 Proceedings*, pages 21–30, 1996.

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

[27] 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.

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

A Appendix: Flow Concatenation

We briefly discuss the details of the *flow concatenation* algorithm mentioned in section 5.2. Given a series of consecutive images $I_0, I_1, \ldots I_n$, we would like to construct the correspondence map $C_{0(n)}$ relating I_0 to I_n. Direct application of the optical flow algorithm may fail because the displacements in the images are too large. Because the images $I_0, I_1, \ldots I_n$ are the result of a dense sampling process, the motion between consecutive frames is small, and hence we can compute optical flow between the consecutive frames to yield $C_{01}, C_{12}, \ldots C_{(n-1)n}$. The goal is to concatenate $C_{01}, C_{12}, \ldots C_{(n-1)n}$ together to yield an approximation to the desired $C_{0(n)}$ map. We can view this problem as the algebraic problem of adding vector fields.

We focus on the case of the 3 images I_{i-1}, I_i, I_{i+1} and the correspondences $C_{(i-1)i}, C_{i(i+1)}$, since the concatenation algorithm is simply is an iterative application of this 3-frame base case. Note that it is not correct to construct $C_{(i-1)(i+1)}(\mathbf{p})$ as the simple addition of $C_{(i-1)i}(\mathbf{p}) + C_{i(i+1)}(\mathbf{p})$ because the two flow fields are with respect to two different reference images, I_{i-1} and I_i. 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* [28] 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:

$$C_{(i-1)(i+1)}(\mathbf{p}) = C_{(i-1)i}(\mathbf{p}) + C^{warped}_{i(i+1)}(\mathbf{p}) \tag{11}$$

A procedural version of our backwarp warp is shown in figure 9. `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 9. BACKWARD WARP algorithm, which warps I backwards along `dx` and `dy` to produce I^{warped}

There are two algorithmic variants for concatenation of n images, shown below. The first variant, **CONCATENATION-DOWN**, iteratively

computes
concatenated flows $C_{(n-1)(n)}, C_{(n-2)(n)}, \ldots, C_{0(n)}$ using the method discussed above. The desired final concatenated flow is $C_{0(n)}$. The second variant, **CONCATENATION-UP**, iteratively computes concatenated flows $C_{0(1)}, C_{0(2)}, \ldots, C_{0(n)}$. The desired final concatenated flow is $C_{0(n)}$.

```
for i = n-1 downto 0 do,
    if i = n - 1 then
        compute $C_{i(n)}(\mathbf{p})$ using optical flow
    else
        compute $C_{i(i+1)}(\mathbf{p})$ using optical flow
        warp $C_{(i+1)n}(\mathbf{p})$ backwards along $C_{i(i+1)}(\mathbf{p})$
            to produce $C^{warped}_{(i+1)n}(\mathbf{p})$
        set $C_{i(n)}(\mathbf{p}) = C_{i(i+1)}(\mathbf{p}) + C^{warped}_{(i+1)n}(\mathbf{p})$
```

Figure 10. CONCATENATION-DOWN algorithm

```
for i = 1 to n do,
    if i = 1 then
        compute $C_{0(i)}(\mathbf{p})$ using optical flow
    else
        compute $C_{(i-1)i}(\mathbf{p})$ using optical flow
        warp $C_{(i-1)i}(\mathbf{p})$ backwards along $C_{0(i-1)}(\mathbf{p})$
            to produce $C^{warped}_{(i-1)i}(\mathbf{p})$
        set $C_{0(i)}(\mathbf{p}) = C_{0(i-1)}(\mathbf{p}) + C^{warped}_{(i-1)i}(\mathbf{p})$
```

Figure 11. CONCATENATION-UP algorithm