Representation Analysis and Synthesis of Lip Images Using Dimensionality Reduction

Michal Aharon Ron Kimmel
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

January 14, 2004

Abstract

Understanding facial expressions in image sequences is an easy task for humans. Some of us are capable of lipreading, by interpreting the different motions of mouths. Automatic lipreading by a computer is a challenging task, with so far limited success. The inverse problem of synthesizing real looking lip movements is also highly non-trivial. Today, the technology to automatically generate an image series that imitates natural postures is far from perfect.

We introduce a new framework for facial image representation, analysis, and synthesis (here we refer just to the lower half of the face with a focus on the mouth). It includes interpretation and classification of facial expressions and visual speech recognition, as well as a synthesis procedure of facial expressions that yields natural looking facial movements.

Our facial image analysis and synthesis processes are based on a parametrization of the mouth configuration set of images. These images are represented as points on a two-dimensional flat manifold that enables us to efficiently define the pronunciation of each word and thereby analyze or synthesize the motion of the lips. We present some examples of automatic lips motion synthesis and lipreading, and propose a generalization of our solution to the problem of lipreading different subjects.

1 Introduction

Automatic understanding and synthesizing of facial movements during speech is a complex task that has been intensively investigated [4, 22, 13, 6, 5, 3, 12, 15]. Improving the technology in this area may be useful for various applications such as better accuracy in automatic voice and speech recognition, as well as automatic comprehension of speech in the absence of sound, also known as lipreading. At the other end, generating smooth movements may enhance the animation abilities in, for example, low bit-rate communication devices such as video conference transmission over cellular networks.

In this paper we introduce a framework that handles frontal view facial images, and is capable of representing, synthesizing, and analyzing sequences of facial movements. Our input is a set of frontal facial images. These images are extracted from training sequences of only one person (the model), in which the subject pronounces known syllables. The ascription of the images to their specific syllable is important, and will be used during the synthesis process.

The images are first automatically aligned with respect to the location of the nose. Every two images are compared and a symmetric dissimilarity matrix is computed.

Next, the images are mapped onto a plane, so that each image is represented as a point, while trying to maintain the Euclidean dissimilarities between images. That is, the Euclidean distance between each two points on the plane should be as close as possible to the dissimilarity between the two corresponding images. We justify this flat embedding operation by measuring the relatively small numerical error introduced by this process.

Thereafter, the faces representation plane is uniformly sampled and 'representative key images' are chosen.

Synthesis can now be performed by concatenating the different sequences of images that are responsible for creating the sound, while smoothing the connection between each two sequential sequences.

Using the 'representative key images', new mouth images can be located on the map, and represented by their coordinates. Each word, which is actually a sequence of mouth images, can now be considered as a contour, given by an ordered list of its coordinates. Analysis of a new word is done by comparison of its contour to those of already known words, and selecting the closest as the best match.

All training sequences and their corresponding mapping process were done with a single subject facial images. Nevertheless, we show that the same concept can be generalized with some success to lipreading of other different subjects, by exploiting the fact that the sequence of pronounced phonemes in the same word is similar for all people. This process requires first correlating between the new person images and the model, and then embedding of the new person's pronounced word on the model's lip configuration surface and calculating a new contour. Next, comparison between the calculated contour and contours of already known words, previously calculated for the model, is computed, and the closest word is chosen as the analysis result.

2 Previous Work

Automatic understanding (analysis) and generation (synthesis) of lip movements may be helpful in various applications, and these areas are under intense study. We first review some of the recent results in this field.

2.1 Analysis

The problem of analyzing lip movements, and automatic translation of such movements into meaningful words was addressed in several papers. Some researchers treat lipreading as a stand-alone process [6, 13], while others use it to improve voice recognition systems [5, 4, 15].

Li et al. [13] address the problem of identification of letter's pronunciation. They handled the first ten English letters, and considered each of them as a short word. For training, they used images of a person saying the letters several times. All images were aligned using maximum correlation, and the series of images of each letter were squeezed or stretched to be of the same length. Each such series of images was converted into a $M \times N \times P$ column vector, where $M \times N$ is the size of each image, and P is the number of images in the series (simple concatenate of the series). Several such vectors representing the same letter created a new matrix, A, of size $MNP \times S$, where S is the number of series. The first eigenvectors of the squared matrix AA^T were considered as the principle components of the specific letter's space. Those principle components were called eigen-sequences. Now, when a new series is tested, it is aligned as before and tested for each of the the possible letters. First, the new series is squeezed or stretched to the same length of a possible letter's series. Then, the new series is projected onto this letter's basis, and the amount of preserved energy is tested. The letter which basis preserves most of the new letter energy is chosen as the pronounced letter in the new series. In that paper, an accuracy of about 90% was reported.

An interesting trial for lipreading was introduced by Bregler et al. [6] under the name of 'the bartender problem'. The speaker, as a customer in a bar, is asked to choose between four different drinks, and due to background noise, the bartender's decision of what the customer is asking for is based only on lipreading. Data was collected on the segmented lips' contour, and the area inside the contour. Then, a Hidden Markov Model (HMM) system was trained for each of the four options. With a test set of 22 utterances, the system was reported to make only one error (4.5%).

Acoustically-based automatic speech recognition (ASR) is still not completely speaker independent, its vocabulary is limited, and it is sensitive to noise.

Bregler et al. [6, 4] showed, using a neural network architecture, that visual information of the lip area during speech can significantly improve (up to 50%) the error rate, especially in a noisy environment. In their experiments, they use a neural network architecture in order to learn the pronunciation of letters (each letter is considered as a short word). Their systems were fed, apart from the acoustic information, with visual information of the lip area (grey level values, first FFT coefficients of the area around the lips or data about the segmented lip). The results showed that such hybrid systems can significantly decrease (up to 50%) the error rate. More significant improvement was seen, as expected, when the amount of noise was high, or for speakers with more emphasized lips movements, i.e., speakers that move their lips more while talking.

Duchnowski et al. [9] developed a similar framework for an easy interaction between human and machine. The person, sitting in front of a computer, was recorded and videotaped while pronouncing letters. The person's head and mouth were located and tracked, using a neural network based system. Several types of visual information were extracted, such as gray level values, band-pass Fourier magnitude coefficients, principal components of the down sampled image or linear discriminant analysis coefficients of the down sampled image. The acoustic and visual data was processed by a multi-state time delay neural network system with three layers, and 15 units in the hidden layer. By combining the audio and visual information, they achieved a 20-50% error rate reduction over the acoustic processing alone, for various signal/noise conditions.

2.2 Synthesis

Bregler et al. [3] introduced 'video-rewrite' as an automatic technique for dubbing, i.e. changing a person's mouth deformations according to a given audio track. They preferred handling triples of phones, and so achieved natural connection between each two. Using segmentation of a training audio track, they labelled the facial images, and each sequential three phonemes were handled separately. Next, they segmented the phonemes in the new audio track, and combined triples of phonemes that resembled the segmentation results. The choice of handling triples of phonemes enabled natural connection between all parts of the sentence. They used a 'stitching' process to achieve correspondence between the synthesized mouth movements and the existing face and background in the video.

A different synthesis procedure by Bregler et al. [6] was based on their concept of 'constrained lip configuration space'. They extracted information on the lip contour, and embedded this information in a five-dimensional manifold. Interpolation between different images of the mouth was done by forcing the interpolated images to lie on this constrained configuration space.

Van Gool et al. [12, 22] chose to handle 3D faces. They worked with a system called "ShapeSnatcher", that uses a structured light technique, in order to acquire 3D facial data. The 3D structure has an advantage over flat images in both analysis and synthesis. It better captures facial deformations, as it is independent of head rotations and translations, and when synthesizing, it can create animation of speech from several view points, and with various head positions.

Kalberer and Van Gool created 'eigenfacemasks'. In [12], the mask consists of 124 vertices, from which 38 are located around the lip area. A mask is represented by a coordinates vector of these 124 vertices. They acquired 3D facial data from a single person pronouncing various phonemes. Each frame was analyzed separately, and represented as a mask. In order to fit the mask's vertices to the correct points on the face, 124 black dots were placed on the face of the speaking subject. After acquiring several such sequential masks, the first 10 eigenvectors were extracted. The space these eigenvectors spanned was considered as the space of intra-subject facial deformations during speech. For animation of a certain word, its viseme[1] face masks were displayed, and spline interpolation between the coefficients of the eigenvectors was used to smooth the transitions. This interpolation was executed between the coefficients of the projection of the different visemes masks on the chosen eigenvectors, which meant that each intermediate mask was embedded in the eigenmask space. The eigenfacemasks' compact space requirements enabled an easy generation of intermediate masks, that looked realistic.

In the latter two papers the use of a small lip configuration space is seen, and transitions between two configurations are restricted to that space. This creates a more natural transition than a simple linear interpolation, and this idea is used also in this paper. To emphasize this idea, which we also encounter in our framework, see Figure 1, where the surface illustrates a limited 3D lip configuration space, and points 'A' and 'B' are two specific lip configurations on that surface. These two configurations are different, so sequential presentation of them might cause a 'jerky' effect. Simple interpolation between the two con-

[1]The term *'viseme'* [10] is a compound of the words *'visual'* and *'phoneme'*, and here represents the series of visual face deformations that occur during pronunciation of a certain phoneme.

figuration will create images off the restricted space (the dashed line), and would therefore, look un-natural. A better synthesis with a smooth and natural transition between the two configuration, pass through the restricted space alone (the consecutive line in the figure).

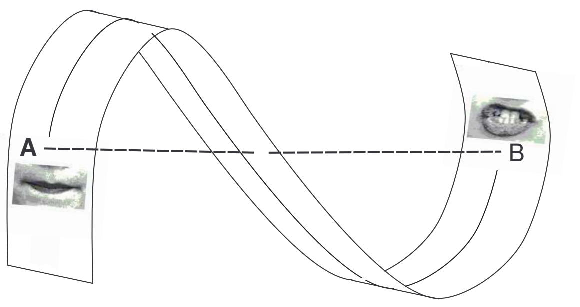

Figure 1: Smoothing the transition between different lip configuration
Illustration of a restricted lip configuration space, and two possible transitions between two different configurations in this space. The dashed line indicates a linear interpolation, while the consecutive line represents a more natural transition, since it is restricted to the configurations manifold.

3 Visual Speech Synthesis and Lipreading by Flat Embedding

Different people pronounce the same vowels differently. Even the pronunciation of the same person in different scenarios may change. We chose to handle a specific situation, of a subject speaking to the camera and slightly accentuating his words.

Each vowel is pronounced differently when said in different parts of a word. For example, the vowel ‘A’ in ‘America’ looks different from the vowel ‘A’ in ‘Los Angeles’. This difference occurs (among other subjective reasons) due to the location of the syllable ‘A’ in the word, and the syllables that appear before and after it. To be more specific, we found that the main reason for different pronunciation of the same vowel is the formation of the mouth just before and just after this syllable is said.

In our framework, we divide each word into isolated parts, each containing a consonant and a vowel, or a consonant alone, e.g. ‘ba’, ‘ku’, ‘shi’, ‘r’ etc. Each of these sounds is considered a syllable. We assume that each such syllable has its own ‘*visual articulation signature*’ (VAS in short), i.e. the series of mouth motions that must occur in order for the sound to be vocalized. These mouth motions may differ from one person to another. Other parts of the full visual pronunciation of a syllable can be neglected. Figure 2 shows a series of images of a mouth pronouncing the syllable ‘sha’. The VAS is defined by images 11-19. Here, the identification of the VAS from a series of images was done manually. It can be done automatically by using an ASR system that points to locations where a specific sound was heard.

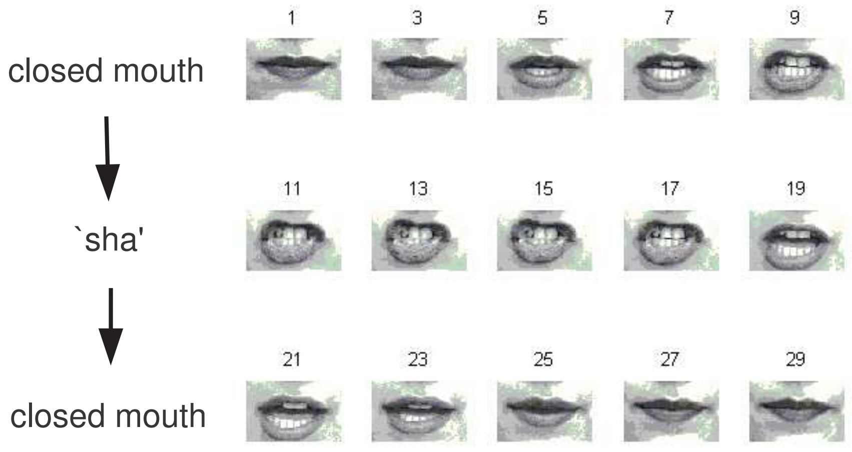

Figure 2: One syllable image sequence
Images sequence of a subject saying the syllable 'sha'. Images 11-19 define the 'visual articulation signature' (VAS).

3.1 The Input Data

A single subject was videotaped while pronouncing 20 sounds, each sound was pronounced six times, each time as a different vowel (A, E, I, O, U, and 'sheva', i.e. a consonant that carries an ultra-short vowel or no vowel sound). Each of these 120 sequences started and ended with a closed mouth. An example of such a sequence can be seen in Figure 2. Moreover, for each series, the indices of the VAS were registered and recorded. The total number of front view face images was about 3450.

3.2 Comparing Images

Alignment: The images were taken using a stationary camera, while the subject was sitting. Nevertheless, slight movements of the head are unavoidable, and the images were first aligned. The nose was chosen as the alignment object, based on the assumption that the nose is stable while talking. The mouth area deforms a lot, the eyes blink, and the ears might be concealed by hair, or when the head rotates. Each image was translated, using an Affine Motion detector algorithm [14, 1] (see also Appendix A), so that the nose is completely stable. After the alignment process is finished, only the mouth-area sub-images (as seen in Figure 2) were considered.
Comparison Measure: As a measure for comparison between images we chose a variation on the Jacobs, Belhumeur, and Basri measure [11], We refer to this measure as JBB. The

suggested distance between two images is

$$E(I,J) = \int\int I \cdot J \left|\nabla\left(\frac{I}{J}\right)\right| \cdot \left|\nabla\left(\frac{J}{I}\right)\right| dxdy, \tag{1}$$

where $I(x,y)$ and $J(x,y)$ are two images and $E(I,J)$ is the relative distance between them.

Let us briefly explain the concept behind the JBB measure. Assume that an object $\{x, y, f(x,y)\}$ is viewed from direction $(0,0,-1)$, its surface normals are $(f_x, f_y, 1)/\sqrt{f_x^2 + f_y^2 + 1}$. When this object, assumed to be Lambertian, is illuminated by one light source from direction (s^x, s^y, s^z), the intensity image is given by

$$I(x,y) = \alpha(x,y)\frac{-(s^x, s^y, s^z)\cdot(f_x, f_y, 1)}{\sqrt{f_x^2 + f_y^2 + 1}}, \tag{2}$$

where $\alpha(x,y)$ is the albedo function of the object.

When dividing two images of the same object, taken under different illumination conditions, the albedos and the normalization components cancel out one another. Generally speaking, the resulting ratio is simpler than the ratio of two images of different objects. A simple measure of the complexity of the ratio image is the integral over its squared gradients $\left|\nabla\left(\frac{I}{J}\right)\right|^2$. Symmetry consideration, and overcoming singularities in shadowed areas lead to the above measure.

To illustrate the advantages of the JBB measure, we compared it to the L_1 and L_2 norms and to the correlation measure, all calculated on a slightly smoothed (with a Gaussian kernel of size 5 and standard deviation 0.8) version of the images. We took one mouth image and compared it to different mouth images, selected at random, from various pronunciations, taken at different times, and under slightly different illumination conditions. The results of each norm were normalized between 0 (most similar) and 1 (most different), as shown in Figure 3.

We can clearly see that the JBB and the correlation measures best identify images number 1, 10 and 11 as the closest to the reference. The JBB measure was chosen, as it also assigns high differences to the rest of the images.

Next, using the JBB measure, we calculated the differences between each two images in the input set. We thereby obtained an $N \times N$ symmetric matrix of differences (dissimilarity measures), where N is the total number of images.

3.3 Flattening

Our next goal is to embed all the images as points on a finite dimensional Euclidean space, such that the Euclidean distance between each two images is as close as possible to the dissimilarity between the images. This flat embedding offers a compact representation method that enables us to handle the recognition process. It seems that for our applications, 'local' accuracy may be more significant than a global one. That is, keeping the accuracy of smaller differences is more important. The reason is that we are interested in representing one image using another that is close to it on the flat surface. The accurate distance between two different images is less important, as long as they are far from each other on the parametrization

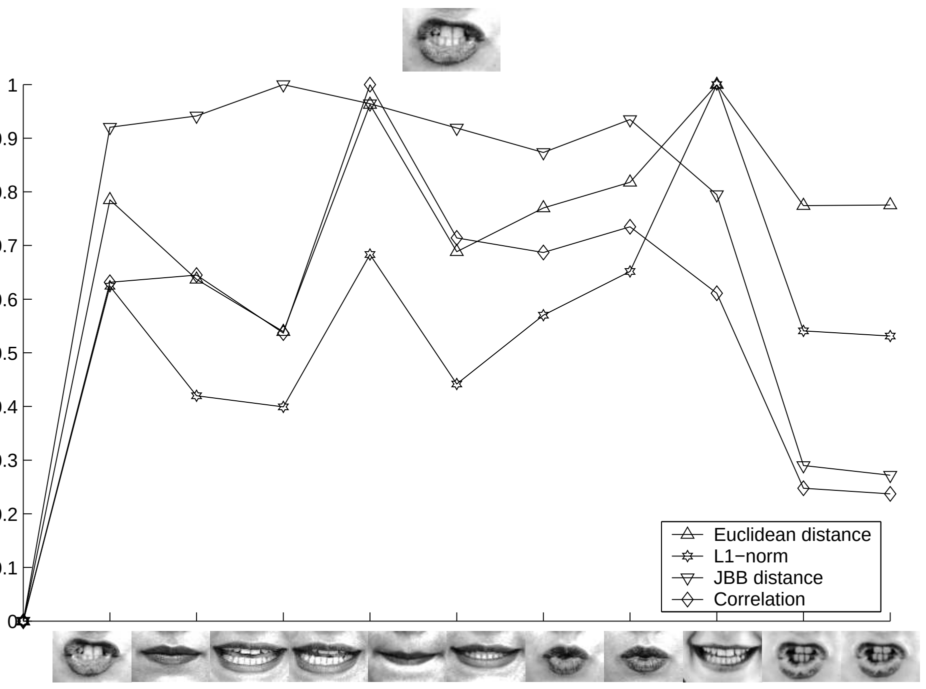

Figure 3: Comparison between various measures for distance between images
A single image (at the top) was compared with 11 randomly chosen images (at the bottom) using the four comparison measures: L_1-norm, L_2-norm (Euclidean distance), correlation, and the JBB measure.

plane. A related flattening procedure was explored by Roweis and Saul [16] with full face images, using Locally Linear Embedding.

Figure 4 shows the screen of a tool we built in order to explore the properties of the flattening process. The embedding flat surface, on which the blue circles are located, is seen in the middle. Each blue circle represents an image, and similar looking images are close to each other. The red contour represents a sequence of mouth images saying the word ‘Emi’. We see that this path is divided into two parts, one for each of the two different syllables that compose this word. The green stars represent the images that are shown when synthesizing the word, in order to create a smooth transition between the two syllables. The stars lie almost on a straight line, which connects the two parts of the word.

It is interesting to note that the flattening procedure we used maps the open mouth images to the upper part of the screen, while close mouth images can be found at bottom. Images that contain teeth are mapped to the right, while images without teeth are found at the left part.

We next investigate 3 flattening method - Locally linear Embedding, Classical Scaling and Least Squares Multidimensional Scaling. Each of these methods was tried on our data base images, and their results were compared. The chosen embedding is the one seen in Figure 4. It was found using least squares MDS with classical scaling initialization.

3.3.1 Locally linear embedding

Locally linear embedding [18] is a flattening method designed for preserving the local structure of the data, and addressing the problem of nonlinear dimensionality reduction. The resulted embedding is optimized to preserve the local configurations of nearest neighbors, while assuming a local linear dependence between them. The ’neighborhood’ definitions of each point is set by the user, and may include all points which distances from the specific point is smaller than a certain value, a fixed number of closest points, or any other reasonable decision. The embedding procedure is divided into three parts:

- Identify the neighbors of each data point, X_i.
- Compute the weights W_{ij} that best reconstruct each data points X_i from its neighbors, by minimizing the cost function (least squares problem)

$$E(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2 . \tag{3}$$

- Compute the vectors Y_i best reconstructed by the weights W_{ij}, by minimizing the equation (eigenvalue problem)

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2 . \tag{4}$$

A more detailed description of this method is given in Appendix B.

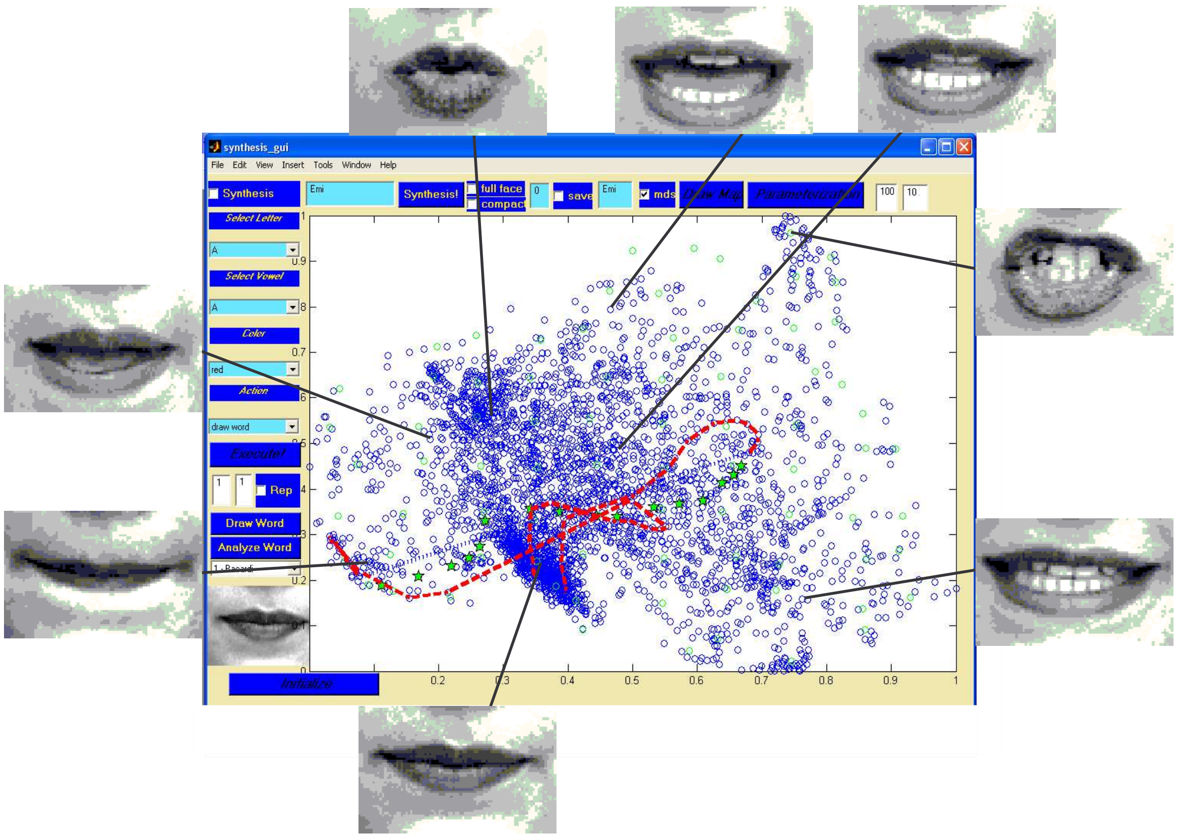

Figure 4: The Embedding Space
A screen of the tool used for synthesis and analysis of the mouth images. In the middle we can see the representation space, in which each circle represents an image.

In our case, the input to the LLE algorithm was the matrix of pairwise distances between each two points, and not the initial coordinates of each point (which would have been the image gray-level values). We therefore derived from this matrix the neighborhood relations and the weights calculations, as described in [18] and in the appendix.

An improvement to the LLE algorithm and the related Isomap [7] was proposed by Donoho and Grimes [8] by the name of 'Hessian Eigenmaps'. Their method can successfully handle the case of a connected non-convex parametrization space. We did not experiment with this method.

3.3.2 Classical Scaling

Multidimensional scaling (MDS), [2], is a family of methods that try to represent similarity measurements between pairs of objects, as distances between points in a low-dimensional space. This allows us to visually capture the geometric structure of the data, and perform dimensionality reduction.

First we tried *classical Scaling* [2] (See Appendix C). *Classical Scaling* assumes that the dissimilarities between the images are Euclidean distances (in some D-dimensional world). Based on this assumption it reveals a centered configuration of points in a d-dimensional world, that best preserves, subject to Frobinius norm, those distances. Classical scaling's solution in a d-dimensional world minimizes the following function,

$$\|B - \tau_1(\Delta_2)\|_F^2 , \quad s.t. \quad B \in \Omega_n(d), \tag{5}$$

where $\Omega_n(d)$ is the set of symmetric $n \times n$ positive semi-definite matrices that have rank no greater than d, $\Delta_2 = [\delta_{ij}^2]$, $\tau_1(D) = -\frac{1}{2}\left(I - \mathbf{1}\mathbf{1}'\right) D \left(I - \mathbf{1}\mathbf{1}'\right)$ is the double centering operator, and $\mathbf{1} = [1, ..., 1]' \in \Re^n$.

The concepts on this method are simple, and includes 4 stages. Let $\Delta^{(2)}$ be the matrix of squared dissimilarities. Do,

- Apply double centering: $B_\Delta = \tau_1(\Delta^{(2)})$.
- Compute Eigendecomposition of $B_\Delta = Q\Lambda Q'$
- Sort the eigenvalues, and denote
$$\Lambda_{ii}^+ = \begin{cases} \Lambda_{ii} & \text{if } \Lambda_{ii} > 0, i < d \\ 0 & \text{otherwise} \end{cases}$$
- Extract the centered coordinates by $X = Q\Lambda^{+1/2}$.

If the distances were indeed taken out of a d-dimensional configuration of points, than the solution classical scaling provides in a d-dimensional world is exact. Otherwise, classical Scaling provides only an approximation, and not necessarily the one we would like.

In our application, we tried classical scaling solution in a 2-dimestional space, by taking the first two eigenvectors (scaled by the eigenvalues) of the distances matrix, after double centering, as the coordinates of the points in the flat configuration. This method reveals the accuracy of the representation captured by the first two eigenvalues, which can be measured by the following 'energy' term, (a variation of the Frobenius norm)

$$E = \sqrt{\frac{\sum_{i=1}^{2} \lambda_i^2}{\sum_{i=1}^{N} \lambda_i^2}}, \tag{6}$$

where λ_i is the i-th largest eigenvalue of the distances matrix, after applying double centering. The first two eigenvalues capture approximately 95% of the energy. This figure validates the fact that mouth images can indeed be embedded in a plane with insignificant distortion, which is somewhat surprising.

3.3.3 Stress definitions

We next try to evaluate the quality of the flattening procedure [2]. We define the representation error as

$$e_{ij}^2 = (\delta_{ij} - d_{ij})^2, \tag{7}$$

where δ_{ij} and d_{ij} in Equation (7) are the dissimilarity and the Euclidean distance between points i and j, respectively. The configuration's total error of representation is measured as the sum of e_{ij}^2 over all i and j, that yields the *stress*

$$\sigma(X) = \sum_{i<j} (\delta_{ij} - d_{ij})^2. \tag{8}$$

Here d_{ij} is the Euclidean distance between points i and j in the configuration X. In order to refer differently to smaller and larger distances as required by our application, we consider a weighted sum of errors

$$\sigma_W(X) = \sum_{i<j} w_{ij} (\delta_{ij} - d_{ij})^2. \tag{9}$$

In order to have a comparable measure for various configurations, we normalize the stress,

$$\hat{\sigma}_W(X) = \frac{\sum_{i<j} w_{ij} (\delta_{ij} - d_{ij})^2}{\sum_{i<j} w_{ij} \cdot \delta_{ij}^2}. \tag{10}$$

Using this measure, we can compare between various flattening methods. The stress results for classical scaling and LLE, calculated with a unified weight matrix, and with weights $w_{ij} = (1/\delta_{ij}^2)$ are given in Tables 1 and 2.

3.3.4 Least Squares Multidimensional Scaling

Least-Square MDS [2] is a flattening method that minimizes the stress value from Equation (10). The optimization method that we used is called 'Iterative Majorization', and a detailed description of the process is given in Appendix D. Nevertheless, the initial configuration of the least squares MDS is crucial, due to the existence of many local minima. In our examples, when initialized with a random configuration, the resulting stress value was actually worse than the one achieved by classical scaling. We thus initialized the algorithm with the configuration that was found by classical scaling. That yielded a significant improvement (see Table 1). An illustration of this combination can be seen in Figure 5. Moreover, this combination of the two methods is fast and easy to implement. We also tried to multiply the classical scaling configuration by a scalar t_2,

$$t_2 = \left(\frac{\langle D_2(X), \Delta_2 \rangle_F}{\|D_2(X)\|_F^2} \right)^{1/2} \tag{11}$$

where $D_2(X)$ is the squared Euclidean distances matrix of a configuration X, Δ_2 is the squared dissimilarity matrix and $\langle\rangle_F$ and $\|\|_F$ are the Frobenious inner product and norm, respectively. This procedure was recommended by Malone, Tarazaga and Trosset [21] as a better initial configuration for the least squares procedure. Using this did not yield any improvement of the final result in our application.

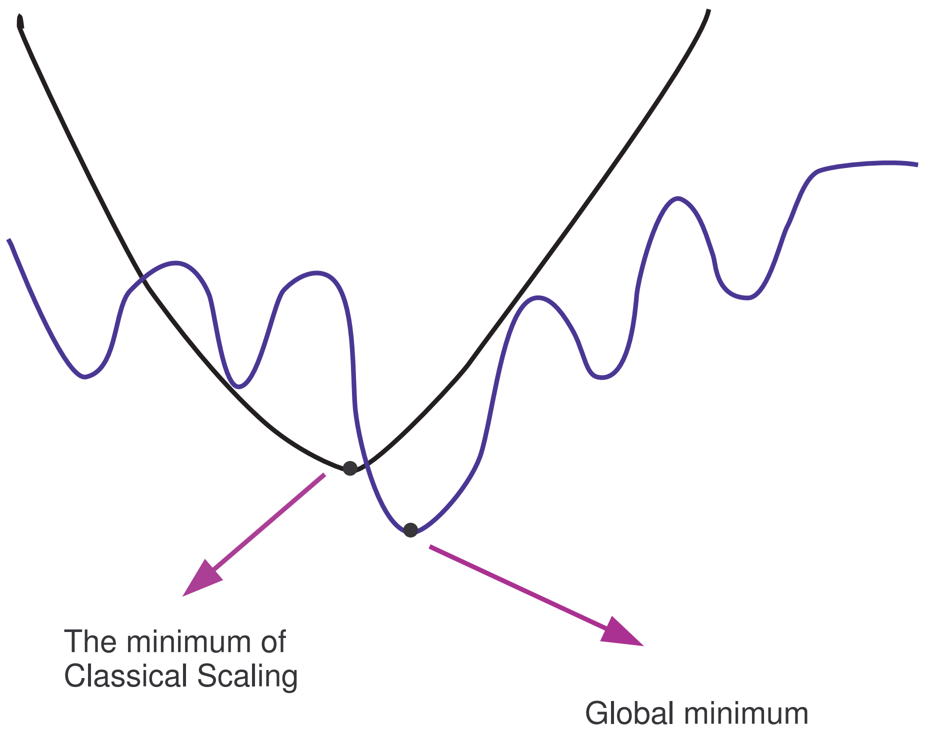

Figure 5: An illustration of the combination of classical scaling and Least Squares MDS.

We search for a configuration that better preserves small distances, and gives higher accuracy. For that end, we defined a weight matrix, that is derived from the dissimilarities matrix as follows,

$$w_{ij} = 1/\delta_{ij}^2, \tag{12}$$

where w_{ij} is the weight assigned to the distance between images i and j, and δ_{ij} is the dissimilarity between these images. These weights enable error of the same ratio in all distances. That is, larger errors are allowed at larger distances.

3.3.5 Flattening Methods - Comparison

All the three methods were checked on our data base. The stress values (weighted and un-weighted) of classical scaling and least squares MDS (with different initial configurations) can be seen in Table 1.

Table 1: Stress values for different variations of MDS
The stress is calculated with a unified weight matrix, and with weight $W_{ij} = 1/\delta_{ij}^2$.

Method	Un-weighted Stress	Weighted Stress
Classical MDS	0.095	0.153
Least Squares MDS with random initialization	0.159	0.0513
Least Squares MDS with LLE initialization	0.022	0.0550
Least Squares MDS with Classical Scaling initialization	0.022	0.0361

All stress values were calculated with the same weights that were used during the iterative process.

We also applied LLE using three different neighborhood definitions:

1. 5 nearest neighbors for each point.
2. 20 nearest neighbors for each point.
3. All neighbors which distances to the point is less than 0.019 (between 1 and 1102 neighbors for each point).

The results were first tested by calculating the un-weighted stress value, and the weighted stress value with the same weights as were used before ($w_{ij} = 1/\delta_{ij}^2$). The results are presented in Table 2. We see that the stress values of the LLE method are relatively high. While the weighted least-squares MDS is designed to minimize the stress value, LLE works for a more general goal - preserving the local configurations of nearest neighbors. In particular, the chosen weights values, used for the calculation of the stress, are completely not connected to the method itself.

Therefore, we defined another stress function more suitable to the LLE principle,

$$stress_k(Y) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{k(i)} \left(d(Y_i, Y_{ij}) - \delta(Y_i, Y_{ij})\right)^2}{\sum_{i=1}^{N} k(i)}. \tag{13}$$

Where N is the total number of points, and $k(i)$ is the number of neighbors of the i-th point (this number can be fixed, or might change when i changes). $d(X, Y)$ and $\delta(X, Y)$ are the

Table 2: Stress values for different variations of LLE
The stress is calculated with a unified weight matrix, and with weight $W_{ij} = 1/\delta_{ij}^2$.

Method	**Un-weighted Stress**	**Weighted Stress**
Fixed number of neighbors (5)	0.951	0.948
Fixed number of neighbors (20)	0.933	0.948
Fixed Threshold (0.019)	0.927	0.930

Euclidean distance and dissimilarity measure between points X and Y respectively. Using this new stress definition, we compared between the configuration found by the LLE and the configuration previously found by the LS-MDS (with the previous weights definition). The results can be seen in Table 3. For each row of the Table, the stress value is calculated as described in 13, in the specific way the LLE was configured (fixed 5 neighbors, fixed 20 neighbors, and all neighbors which distances from the point is less than 0.019).

Table 3: LLE adapted stress values for the results of LLE procedures.

Neighborhood definition	Stress value for the LLE configuration	Stress value for the LS-MDS configuration
Fixed number of neighbors (5)	0.0011	0.0003
Fixed number of neighbors (20)	0.0023	0.022
Fixed Threshold (0.019)	0.00058	0.00011

We can still see that the stress values of the LLE configurations are worse than those archived by the LS-MDS. This implies that the LLE is less suitable for our task.

Another recent method for dimensionality reduction, which we did not investigate, is known as *'Isometric feature mapping'* or *ISOMAP* [7]. This method assumes that the smaller measured distances well approximate real geodesic distances, and using those values, geodesic distances between faraway points are calculated. Than, classical scaling is used in order to flatten the points to a space of the desired dimension. In the ISOMAP, regular sampling of the configuration manifold could cause a problem with the convergence of the method and could impose a non-geometric metric related to the resulting graph. Actually, a similar idea, was used by [19] to flatten manifolds.

In our application, using Least-Squares MDS enabled us to increase dramatically the weight of the large distances, so that their exact value has a smaller influence on the solution. Nevertheless, it introduces a free parameter that sets the neighborhood size, and prevents us from comparing reliably between the various methods. We believe that weighting the importance of the flattened distances can replace, in some cases, the need to re-define the large distances, as is aimed to in Isomap. Moreover, as we show empirically, the small stress value computed from our flat embedding via Least-Squares MDS validates the numerical correctness of the method we used for the lips images.

3.4 Space Parametrization

Thousands of images were flattened to a plane, and generated different density areas, as can be seen in Figure 4. In order to locate a new given image in the plane, we first select 'representative key images' by uniformly sampling the plane. We use only these images to locate new image coordinates. In our experiments we selected 81 images (out of 3450) to sample the representation plane. This was done by dividing the plane into 100 squares (10 squares in each row and column). For each square that contained images, the image with the median coordinates was selected as a 'representative key image'. Next, in order to locate the coordinates of a new image in the parametrization plane the following steps were followed.

1. The nose in the new image is aligned, by comparing to one of the previously taken images, using an affine motion tracker algorithm.

2. The JBB distances between the new image, and each of the 'representative key images' were calculated.

3. The coordinates of the new image are set as a weighted sum of the representatives' coordinates, according to the distance from each representative.

$$x_{new} = \frac{\sum_{i=1}^{N} w_i \cdot x_i}{\sum_{i=1}^{N} w_i}, \tag{14}$$

   where N is the number of 'representative key images', x_i is the x coordinate of the $i-th$ representative key image and the weight w_i is set to be $1/\delta_i^3$, where δ_i is the JBB distance between the new image, and the $i-th$ representative key image. The y coordinate was calculated in a similar way.

3.5 Sentence Synthesis

A simple way to synthesize sequences using the facial images is by concatenating the VAS of the syllables that combine the sentence, so that the 'crucial' part of each syllable is seen. The first and last part of the sequence of pronunciation of each syllable appears only if this syllable is at the beginning or the end of the synthesized word, respectively.

This concatenating procedure results in unnatural speaking image sequences because the connection between the different partial sequences is not smooth enough. An example can be seen in Figure 6. There, a simple synthesis of the name "Emi" is performed as described above, and the transition between the two syllables (images 14 and 15) can be easily detected.

A possible solution to this problem is by defining a clique weighted graph, i.e. a graph in which a weighted arc exists between each two of its vertices. The vertices represent the input images and the weight of the arc between vertex i and j is the dissimilarity measure between the two corresponding images. A smooth transition between images A and B can be performed by presenting the images along the shortest path between A and B. The shortest path is easily found using the Dijkstra algorithm. The shortest path between an image at the end of the VAS of the first syllable, and an image at the beginning of the VAS of the next syllable is used to synthesis smooth transactions, as shown in Figure 7. There, 16 images (those marked as 'new' in the figure) were found by the Dijkstra algorithm as the

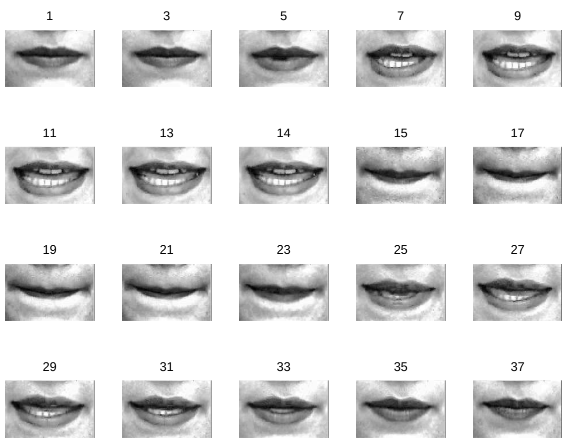

Figure 6: Sampled sequence of the synthesis of the name "Emi"

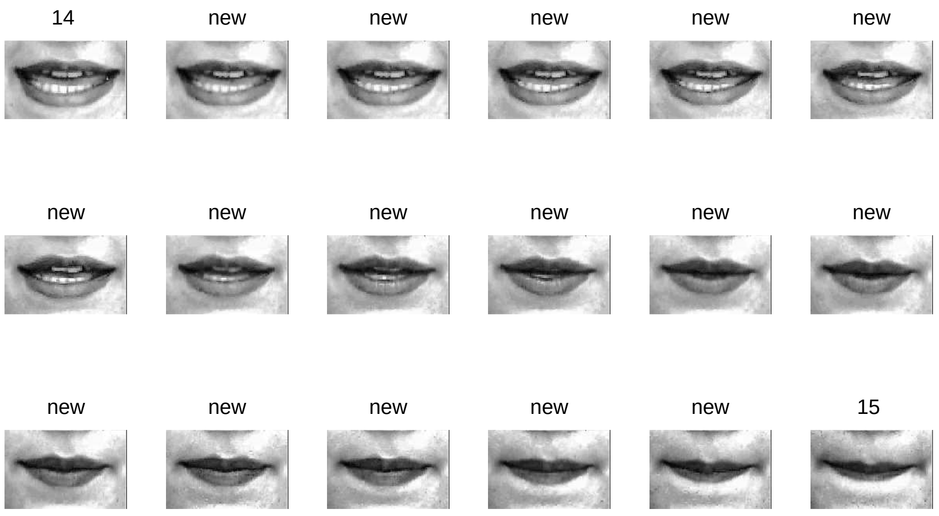

Figure 7: Smooth transition between images 14 and 15 in Figure 6.

shortest weighted path between the last image of the viseme signature of the phoneme 'E' (number 14) and the first image of the viseme signature of the phoneme 'Mi' (number 15). This smooth connection between two different lip configurations is completely embedded in the constrained lip configuration space.

In this solution, a problem may occur if the path that is found includes too many images. Merging those images may slow down the pronunciation, whereas the duration of the pronunciation and synchronization with the sound is crucial when synthesizing speech. We, therefore, control the number of images that are presented by re-sampling the new images. An example of a shorter smooth transition is shown in Figure 8.

Another solution, that exploits the embedding surface and the chosen representative key images is to define a clique weighted graph which nodes are the representative key images, and the two images between which the smoothing should be done. The weight of the arc

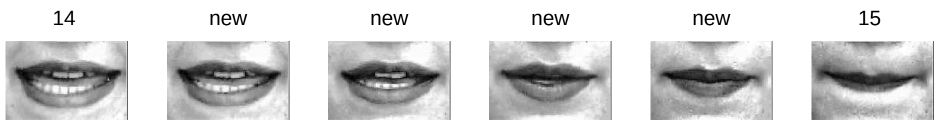

Figure 8: Sampled transition between images 14 and 15 in Figure 6.

that connects images i and j is the distances measure between the two images. The smooth transition is now the images that appear on the shorted path between the two images. This solution is faster, as the dijkstra algorithm runs on a much smaller graph, and the paths that are found rarely needs resampling, as they are much shorter than in the previous solution. An example of the synthesis of the name 'Emi' appears in Figure 9.

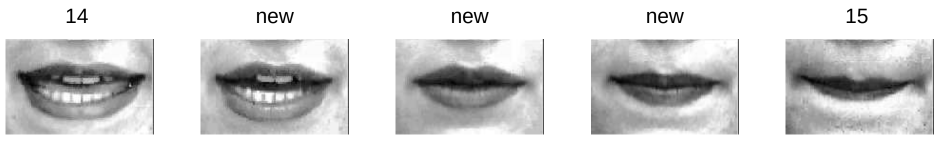

Figure 9: Smooth transition between images 14 and 15 in Figure 6, using the embedded-based synthesis method.

This synthesis procedure is completely automatic. The input is defined by the text to be synthesis and possibly the time interval of each syllable pronunciation, as well as pauses between the words. The results look natural as they all consist of only realistic, aligned images, smoothly connected to each other.

3.6 Lipreading

Here we extend the 'bartender problem' proposed by Bregler et al. [6]. We chose sixteen different names of common drinks[2], and videotaped a single subject (the same person that pronounced the syllables in the training phase) saying each word six times. The first utterance of each word pronunciation was chosen as reference, and the other utterances were analyzed, and compared to all the other reference sequences. After the coordinates of each image in each word sequence (training and test cases) are located on the plane, each word can be represented as a contour. Analyzing a new word reduces to comparing between two such contours on the parametrization representation plane.

Comparing Contours: The words' contours, as an ordered list of coordinates, usually include a different number of images. In order to compare two sequences we first fit a sequence length to the one it is compared to. We do so by using a version of the Dynamic Time Warping Algorithm (DTW) of Sakoe and Chiba [17] with a slope constraint of one. This algorithm is commonly used in the field of speech recognition [13]. The main idea behind the DTW algorithm is that different utterances of the same word are rarely performed at

[2] The tested drink names: Bacardi, Coffee, Tequila, Cola, Martini, Champagne, Bloody Mary, Milk Shake, Orange Juice, Vodka, Cappuccino, French Vanilla, Lemonade, Liqueur, Sprite, Sunrise.

Table 4: Illustration of a possible table results for the dynamic programming algorithm. The first value in each cell indicates the distance, and the second value indicates the index of the minimum value chosen for the above distance.

	1	2	3
1	$(0,-1)$	$(\infty,1)$	$(\infty,1)$
2	$(\infty,1)$	$(1,2)$	$(2,1)$
3	$(\infty,1)$	$(2,3)$	$(3,2)$
4	$(\infty,1)$	$(\infty,1)$	$(5,3)$

the same speaking rate across the entire utterance. Therefore, when comparing different utterances of the same word, the speaking rate and the duration of the utterance should not contribute to the dissimilarity measurement.

Let us denote the two sequences of points as: $A = [a_1, a_2, ...a_m]$, and $B = [b_1, b_2, ...b_n]$, where $a_i = \{x_i, y_i\}$ are the x and y coordinates of the $i - th$ image in the sequence. We first set the difference between images a_1 and b_1 as $g(1,1) = 2d(1,1)$, where $d(i,j)$ is the Euclidean distance $\|a_i - b_j\|$. Then, recursively, we denote

$$g(i,j) = \min \begin{bmatrix} g(i-1,j-2) + 2d(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \\ g(i-2,j-1) + 2d(i-1,j) + d(i,j) \end{bmatrix}. \tag{15}$$

$g(i,j)$, where i or j is smaller than 1, is defined as infinity. The distance between sequences A and B is $g(m,n)$. The indices of the minimum chosen values (each index can vary from 1 to 3, for the 3 possible values of $g(i,j)$) indicates the new parametrization of the sequence A, in order to be as close as possible to the parametrization of the sequence B.

An imaginary example for such a computation can be seen in Table 4, which we denote as T. There, a cell $T_{i,j}$ contains two numbers (a,b) where a is the distance $g(i,j)$, and b is the index of the minimum value chosen for this distance value. This example illustrates the new parametrization of a path with 4 points into a path with 3 points. The chosen points will be numbered 1,2 and 4 (the index of cell $T_{4,3}$ is 3, which means the next cell to look for is $T_{2,2}$, which index is 2, and therefore points to cell $T_{1,1}$).

Using dynamic programming methods, the maximum number of Euclidean distance computations is $m \cdot n$, and therefore, the computation is efficient.

When a new parametrization s is available, the first derivative of sequence A is calculated using a backward approximation scheme $x'^A_s = x^A_s - x^A_{s-1}$, and second derivatives using a central scheme $x''^A_s = x^A_{s+1} - 2x^A_s + x^A_{s-1}$. In this new parametrization the number of elements in each sequence is the same, as well as the number of elements of the first and second derivatives, that can now be easily compared. Next, three different distance measures between the two contours are computed

$$\begin{array}{l} G(A,B) = g(m,n), \\ P(A,B) = \sum_{s=1}^{n} \left(\left({X'}_s^A - {X'}_s^B \right)^2 + \left({Y'}_s^A - {Y'}_s^B \right)^2 \right), \\ Q(A,B) = \sum_{s=1}^{n} \left(\left({X''}_s^A - {X''}_s^B \right)^2 + \left({Y''}_s^A - {Y''}_s^B \right)^2 \right). \end{array} \tag{16}$$

Those measures indicate on the closest reference word to a new pronounced word.

We now summarize the whole analysis process. When receiving a new image sequence N,

1. Find the path that corresponds to this sequence by finding the representation plane coordinates of each image in the sequence as described in section 3.4.
2. For each reference sequence R_j, for $j = 1$ to k, where k is the number of reference sequences (16 in our experiments) do:
   (a) Compute the DTW between the sequence N and R_j.
   (b) Use these results to compute the distances $G(N,R_j)$, $P(N,R_j)$, $Q(N,R_j)$.
3. Normalize each distance by
$$\begin{array}{l} \tilde{G}(N,R_j) = G(N,R_j) / \sum_{i=1}^{k} G(N,R_i), \\ \tilde{P}(N,R_j) = P(N,R_j) / \sum_{i=1}^{k} P(N,R_i), \\ \tilde{Q}(N,R_j) = Q(N,R_j) / \sum_{i=1}^{k} Q(N,R_i). \end{array} \tag{17}$$
4. For each reference sequence, compute the distances
$$D_j(N) = \tilde{G}(N,R_j) + \alpha \cdot \tilde{P}(N,R_j) + \beta \cdot \tilde{Q}(N,R_j). \tag{18}$$
   In our experiments, we empirically found that $\alpha = \beta = \frac{1}{2}$ give the best classification results.
5. Select the reference sequence with the smallest distance $D_j(N)$ as the result of the analysis process.

The combination of the integral Euclidean distance with the first and second derivatives is an approximation of the Sobolev Spaces norm, defined as

$$\|f\|_{H^2}^2 = \sum_{j=0}^{k} \left\| f^{(j)} \right\|_{L^2}^2 = \|f\|^2 + \|f'\|^2 + \|f''\|^2. \tag{19}$$

We next show that this hybrid norm gives better classification results than each of its components alone.

Results: We tested 96 sequences (16 words, 6 utterances of each word, one of which was selected as the reference sequence). The accuracy rate is 94% (only 6 errors). The results of the different measures (Euclidean, first, second derivatives, and their combination) can be

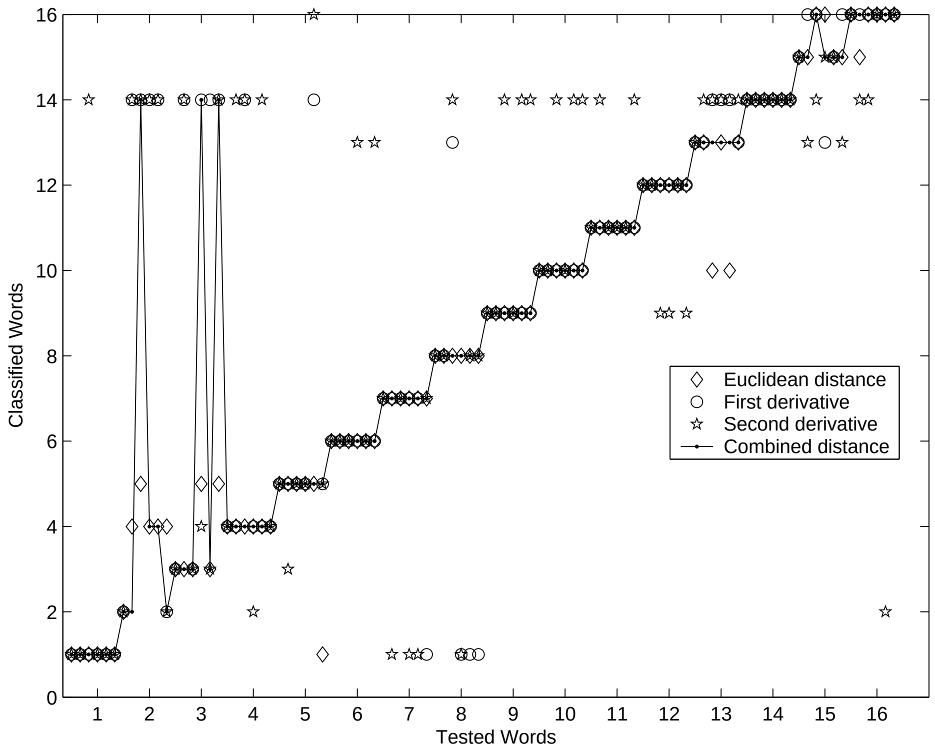

Figure 10: Analysis results of the different distance measures

viewed in Figure 10. The 96 utterances are divided into 16 groups along the x axis. The diamond, circle, and star icons indicate the analysis results computed with the Euclidean, first derivative, and second derivative distance, respectively. The red line indicates the result of approximated Sobolev norm that combines the three measures. The six miss-classifications are easily detected as the deviations from the staircase structure. We see that the Euclidean distance is more accurate than the noisy first and second derivative distances. That was the reason we increased its relative weight in the hybrid Sobolev norm. Either way, the combination of the three yields the best results.

3.7 Generalization: lipreading other people

Up until now, we handled facial images of only one person (female). Next, we present a generalization of our framework, in which we intend to lip read other people. Instead of performing the whole learning process from the beginning, we exploit the fact that different people say the different words in a similar way (the sequence of pronounced phonemes is equal, when saying the same word). Therefore, after a proper matching between the lip configurations images of the model and the new person, we expect the representing contours of the same word to look similar.

For that end, we took pictures of another person (male), pronouncing the various drinks' names, three times each word. In Figure 11 we can see the two people pronouncing the word 'Coffee'.

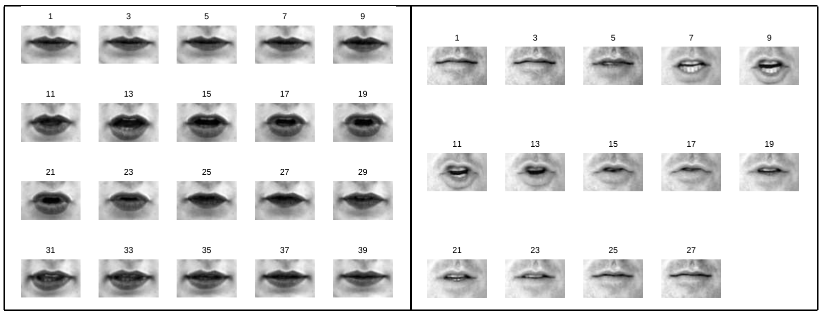

Figure 11: Pronunciation of two different people
Two different people pronouncing the word 'Coffee'. The left woman's images were used for the training and surface representation.

Comparing mouth area images of two different people might be deceptive because of different facial features such as the lips thickness, skin texture or teeth structure. Moreover, different people pronounce the same phonemes differently, and gray level or mouth's contour comparison between the images might not reveal the true similarity between phonemes.

For that end, we aligned the new person's nose to the nose of the model using Euclidean version of Kanade-Lucas. An affine transformation here may cause distortion of the face due to different nose structures. Next, the rest of the images are aligned to the first image (of the same person) using affine Kanade-Lucas algorithm, so that all the mouth area images can be extracted easily.

Then, we relate between images of different persons by defining a set of phonemes, and assigning each phoneme a representing image (also known as viseme). The visemes of the model are placed on the surface using the method described in Section 3.4. The other person's visemes are assigned **exactly the same coordinates**. The process of assigning an image for each phoneme was done here manually, but it can also be done automatically using an ASR system, that detects the image that was taken exactly when the specific phoneme is heard. Figure 12 shows part of the visemes we assigned for the model and the other person.

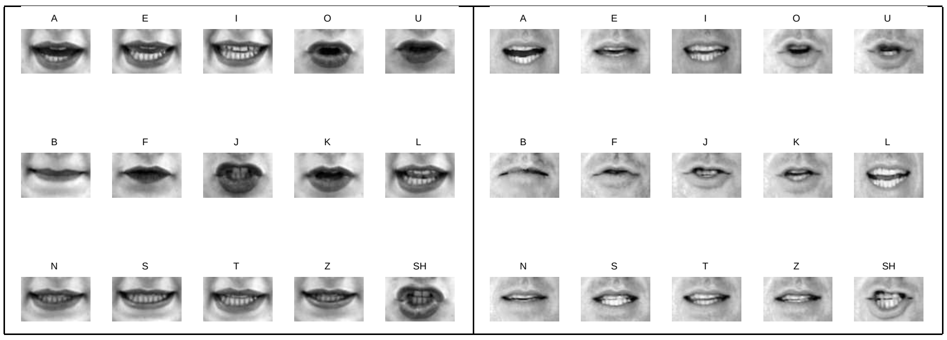

Figure 12: Visemes assignment
Partial group of the visemes we assigned for the model (left) and the other person.

Next, the location of the new person's images on the surface is found using the following procedure,

- the image is compared to all the assigned visemes of the same person, resulting the similarity measures $\{\delta_i\}_{i=1}^{N}$, where N is the number of visemes.
- The new coordinates of the image is set by

$$x_{new} = \frac{\sum_{i=1}^{N} w_i \cdot x_i}{\sum_{i=1}^{N} w_i}, \tag{20}$$

  where x_i is the x coordinate of the $i-th$ viseme and the weight w_i is set to be $w_i = 1/\delta_i^2$. The y coordinate is set in a similar way.

In the above procedure, only images of the same person are compared. By using it, each new person's image can be located on the surface, and each new pronounced word is described as a contour which can be compared with all the other contours. In Figure 13 four contours are shown, representing the pronunciation of the words 'Cappuccino' and 'Sprite' by two different people - the model on the left, and the second person on the right.

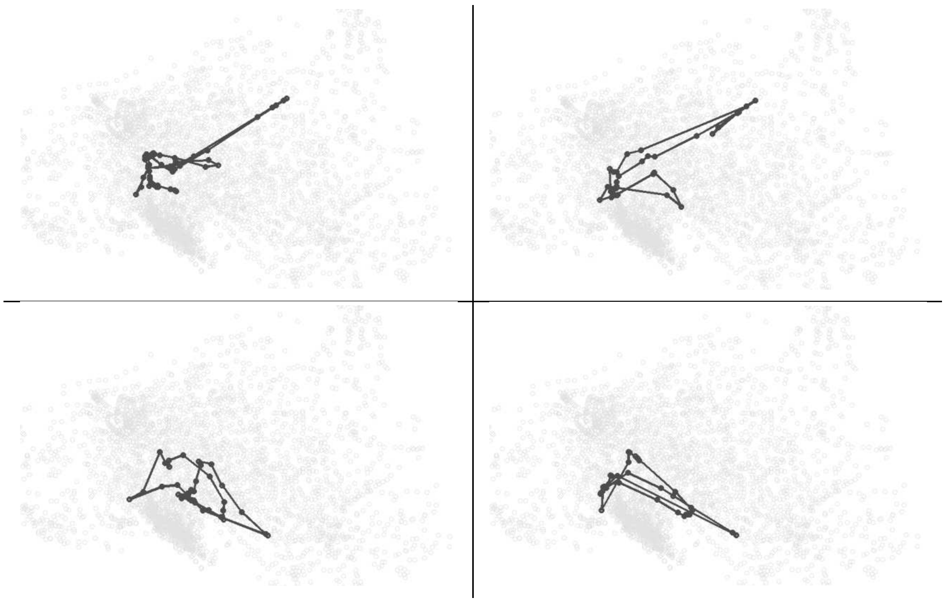

Figure 13: Contours representation of words pronunciation
Contours, representing the pronunciation of the words 'Cappuccino' (upper images) and 'Sprite' (lower images) by two different people. The contours resulted from the pronunciation of the model are on the left.

For comparison between pronunciation contours of two different people we added two additional measures, which we found helpful for this task,

- maximum distance, which is defined by

$$\min_{1\leq s\leq n} \{d(X_s^A - Y_s^B)\} \tag{21}$$

  where X_s^A and X_s^B are the parametrization of the two contours, as seen in section 3.6, after executing DWT, and $d(X, Y)$ is the Euclidean distance between points X and Y.

- summed distance, which is defined by

$$\sum_{1 \le s \le n} d(X_s^A - Y_s^B) \tag{22}$$

where X_s^A and X_s^B and $d(X,Y)$ are as defined above.

The above two measures refer only to the parametrization of the contour after processing DWT. The maximum distance measure calculate the maximum distance between two correlated point on the two contours, and the summed distance sums the Euclidean distances between the correlated points.

We discovered that the derivative distances that were defined in 3.6 and helped comparing between two contours of the same person, were not useful here. The inner structure (first and second derivatives) of the contour was less important than its points location. An example can be seen in Figure 13 where contours of the same pronounced word are shown. The point location of the two contours is similar, but their inner structure is different.

The identification rate was 44 percent in the first trial, and reached 60 percent when allowing the first two answers (out of 16) to be considered. We relate this relatively low success rate to the assumption that different people performs the transitions between phonemes differently, and therefore, correlating between the phoneme's images of two different people does not suffice for perfect identification. We base this assumption on the fact that the names that suffered from the lowest identification rate were those composed of two words ('Bloody Mary','Milk Shake','Orange Juice' and 'French Vanilla'). There especially, although pronouncing all the phonemes in the same order, different people performs the connection differently. An example of the contours calculated for the drink 'Orange Juice' can be seen in Figure 14. In this figure, each contour is divided into two parts, for the two parts of the drink's name, and the difference between the two contours can be easily seen. Considering only the single word-drinks a success rate of 69 percent is achieved, and if considering the first two answers, we reach 80 percent success rate.

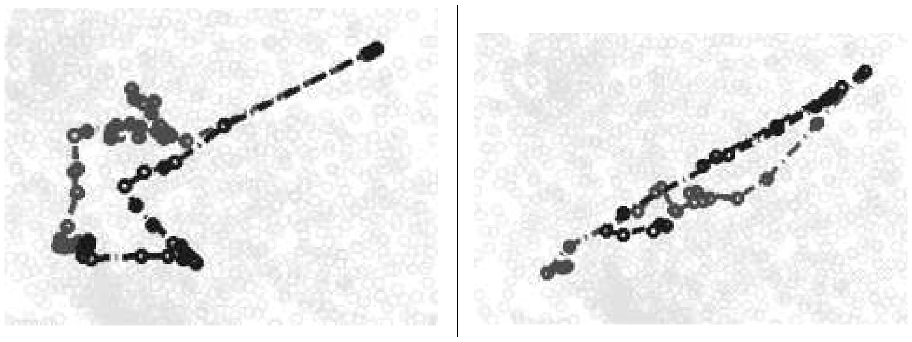

Figure 14: Contour representation of a two-word name pronunciation
Two contours, representing the pronunciation of the word 'Orange Juice' as also seen in Figure 13. The lighter part represents the first part of the drink ('Orange'), and the darker part represents the last part. The contour resulted from the pronunciation of the model is on the left.

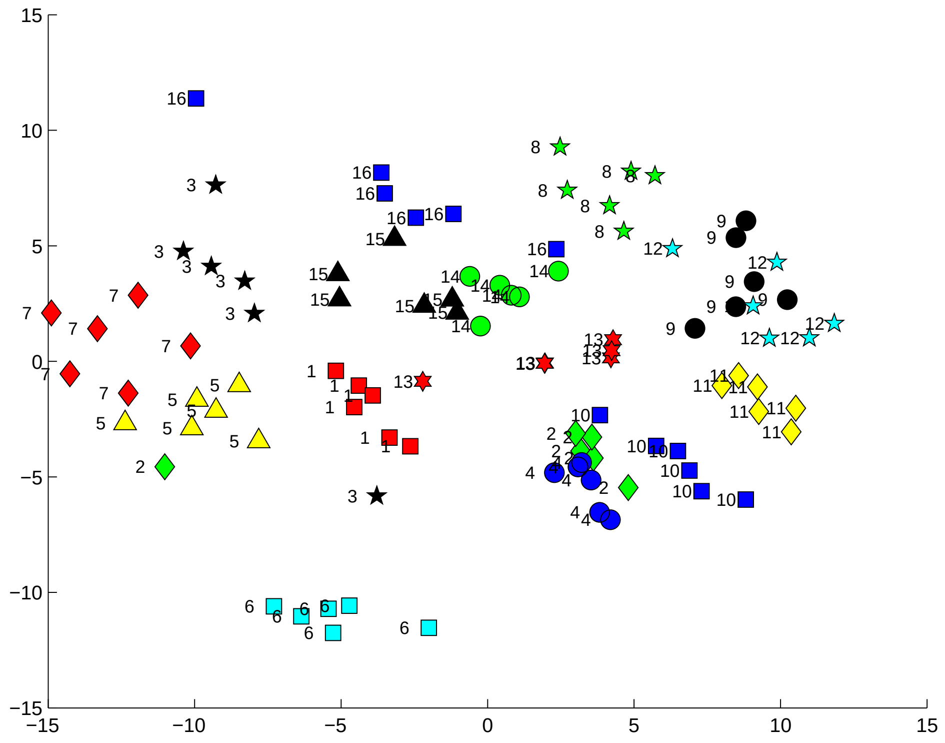

Figure 15: Choosing drink names in noisy bars
Results of mapping 16 drink names into a plane. The distance between names corresponds to the measured dissimilarity by our method. The different drink names are: 1. Bacardi, 2. Coffee, 3. Tequila, 4. Cola, 5. Martini, 6. Champagne, 7. Bloody Mary, 8. Milk Shake, 9. Orange Juice, 10. Vodka, 11. Cappuccino, 12. French Vanilla, 13. Lemonade, 14. Liqueur, 15. Sprite, 16. Sunrise.

3.8 Choosing Drink Names in Noisy Bars

Here we try to answer the following question, 'What kind of drink names should be chosen in a noisy bar, in order to easily understand by the bartender when asked for, using lipreading?'. To answer this question, we measured the distances between each two contours from the set of 96 calculated drink sequences. We received a distances matrix, and performed Classical Scaling. The first two eigenvectors captured 88% of the energy, and the first three 92%. The map presented in Figure 15, shows that the drinks: 'Bacardi', 'Martini', 'Champagne','Bloody Mary','Milk Shape','Vodka', and 'Cappuccino' have more distinct names. Putting them on the menu, possibly with Cola (but without Coffee), or French Vanilla (but without Orange Juice), might ease lipreading of the customers requests.

4 Summary

We introduced a lipreading and lip motion synthesis framework. We perceptually justified and used the JBB measure for comparison between different images, a measure that is robust under slight pose changes and varying illumination conditions.

We then flattened the visual data on a plane. This representation map, which captures the geometric structure of the data, enables us to sample the space of lip configurations by uniformly selecting points from the embedding surface. Using those selected representatives and the Dijkstra algorithm, we managed to smoothly tile between two different images, and synthesize words.

The embedding surface is then used to represent each pronounced word as a planar contour. That is, a word becomes a planar contour tracing the points on the plane at which each point represents an image. The lip reading (analysis) process was thereby reduced to comparing between planar contours. The comparison was done using an efficient dynamic programming algorithm, based on Sobolev spaces norms.

Later, generalization of the lipreading process was performed with promising results by exploiting the fact that the sequence of pronounced phonemes is similar to all people pronouncing the same word. This was done by first correlating between the two subject lips configuration spaces, and than comparing images of the same person only.

Our experiments suggest that exploring the geometric structure of the space of mouth images, and the contours plotted by words on this structure may provide a powerful tool for lip-reading. More generally, we show that dimensionality reduction for images can provide an efficient tool for representation of a single image or a sequence of images from the same family. It can therefor offer a way to perform synthesis and analysis of such sequences.

Several future research directions are suggested:

- Stitching the mouth image into a full face image in order to obtain a real looking synthesis of full faces.
- Execute MDS flattening on several different people's mouth images, while forcing the matching visemes (of the different people) to lie close to each other. This may result a 3-dimensional surface, that contains several layers. Each layer belongs to one person, and provides a framework for synthesis and analysis. Walking between layers may then results in synthesis of one word by several different people all together (while smoothing the transition between the different people).

Appendixes

A Affine Motion Detector Algorithm.

Kanade-Lucas tracker algorithm [20] is one of the most popular algorithms for feature tracking or image alignment. In our framework, we used the affine motion detector algorithm, which is based on the same concepts, for alignment between the taken images. For completeness, we briefly present the main ideas of this algorithm.

Let J be the first image in the sequence, to which we compare all other images. We define as reference, manually, a square window $W(X) = X$, that contains a set of coordinates $x_i = [x, y]^T$, around the nose. Let us denote this reference window as T, so that $T = J(W(X))$. We then refer to each other image I, and try to find a vector of parameters P, so that $I(W(X; P))$ is closest to T. $W(X; P)$ is a transformed version of the coordinates set X, with parameters P. The parameters are iteratively changed during the algorithm, in order to minimize the difference $(I(W(X; P)) - T)^2$. This change, as also the length of the vector P, depends on the type of transformation we allow. If we allow only translation, we define $P = [p_1, p_2]^T$ as translation parameters, and each coordinate in the set X transforms by $W(x_i; P) = [x_i + p_1, y_i + p_2]^T$. If we allow any affine transform, then $P = [p_1, p_2, p_3, p_4, p_5, p_6]^T$, and

$$W(X_i; P) = \begin{pmatrix} P_1 + (P_2 + 1)x_i + P_3 y_i \\ P_4 + P_5 x_i + (P_6 + 1)y_i \end{pmatrix}. \tag{23}$$

Each iteration we compute a transition $\triangle P$, so that $I(W(W(X; P); \triangle P)$ (the transform with parameters $\triangle P$ is executed on the coordinate set $W(X; P)$) is closer to $T(X)$. For minimizing the distance between the two sub images, we take a first order Taylor expansion about P

$$E(P) = \sum_x \left[I(W(X; P + \triangle P)) - T\right]^2 \approx \sum_x \left[I(W(X; P)) + \nabla I \frac{\partial W}{\partial P} \triangle P - T\right]^2 \tag{24}$$

where $\frac{\partial W}{\partial P}$ is the Jacobian of the defined window W, and $\nabla I = [I_x, I_y]$. We differentiate Equation (24) with respect to $\triangle P$, and get

$$\frac{\delta E(P)}{\delta P} = \sum_x \left[\nabla I \frac{\partial W}{\partial P}\right]^T \left[I(W(X; P)) + \nabla I \frac{\partial W}{\partial P} \triangle P - T\right]. \tag{25}$$

Setting this derivative to zero, and solving it, gives a set of equations, from which we can extract a close form solution for $\triangle P$.

$$\sum_x \left[\nabla I \frac{\partial W}{\partial P}\right]^T \left[\nabla I \frac{\partial W}{\partial P}\right] \cdot \triangle P = \sum_x \left[\nabla I \frac{\partial W}{\partial P}\right]^T \left[T - I(W(X; P))\right] \tag{26}$$

If we denote $(I(W(X; P)) - T(x))$ as I_t, for the Affine transform, we get the following set of equations [1]

$$\begin{pmatrix} \sum I_x^2 & \sum xI_x^2 & \sum yI_x^2 & \sum I_xI_y & \sum xI_xI_y & \sum yI_xI_y \\ \sum xI_x^2 & \sum x^2I_x^2 & \sum xyI_x^2 & \sum xI_xI_y & \sum x^2I_xI_y & \sum xyI_xI_y \\ \sum yI_x^2 & \sum xyI_x^2 & \sum y^2I_x^2 & \sum yI_xI_y & \sum xyI_xI_y & \sum y^2I_xI_y \\ \sum I_xI_y & \sum xI_xI_y & \sum yI_xI_y & \sum I_y^2 & \sum xI_y^2 & \sum yI_y^2 \\ \sum xI_xI_y & \sum x^2I_xI_y & \sum xyI_xI_y & \sum xI_y^2 & \sum x^2I_y^2 & \sum xyI_y^2 \\ \sum yI_xI_y & \sum xyI_xI_y & \sum y^2I_xI_y & \sum yI_y^2 & \sum xyI_y^2 & \sum y^2I_y^2 \end{pmatrix} \cdot \begin{pmatrix} \triangle p_1 \\ \triangle p_2 \\ \triangle p_3 \\ \triangle p_4 \\ \triangle p_5 \\ \triangle p_6 \end{pmatrix} = - \begin{pmatrix} \sum I_xI_t \\ \sum xI_xI_t \\ \sum yI_xI_t \\ \sum I_yI_t \\ \sum xI_yI_t \\ \sum yI_yI_t \end{pmatrix} \tag{27}$$

Each iteration, the new parameters vector $\triangle P = [\triangle p_1, \triangle p_2, \triangle p_3, \triangle p_4, \triangle p_5, \triangle p_6]^T$ is recalculated, and this differential movement is added to the global movement $P \leftarrow P + \triangle P$ in the following way

$$W(X;P) \circ W(X;\triangle P) = \begin{pmatrix} 0 & 1 \\ 0 & (p_1 + p_2 \cdot x + p_3 \cdot y) \\ 0 & (p_4 + p_5 \cdot x + p_6 \cdot y) \\ 1 & 0 \\ (p_1 + p_2 \cdot x + p_3 \cdot y) & 0 \\ (p_4 + p_5 \cdot x + p_6 \cdot y) & 0 \end{pmatrix}^T \cdot \begin{pmatrix} \triangle p_1 \\ \triangle p_2 \\ \triangle p_3 \\ \triangle p_4 \\ \triangle p_5 \\ \triangle p_6 \end{pmatrix} \tag{28}$$

and so it turns that

$$P + \triangle P = \begin{pmatrix} p_1 + \triangle p_1 + p_1 \cdot \triangle p_1 + p_3 \cdot \triangle p_2 \\ p_2 + \triangle p_2 + p_2 \cdot \triangle p_1 + p_4 \cdot \triangle p_2 \\ p_3 + \triangle p_3 + p_1 \cdot \triangle p_3 + p_3 \cdot \triangle p_4 \\ p_4 + \triangle p_4 + p_2 \cdot \triangle p_3 + p_4 \cdot \triangle p_4 \\ p_5 + \triangle p_5 + p_1 \cdot \triangle p_5 + p_3 \cdot \triangle p_6 \\ p_6 + \triangle p_6 + p_2 \cdot \triangle p_6 + p_4 \cdot \triangle p_6 \end{pmatrix} \tag{29}$$

For performance considerations, it is recommended to switch between $I(W(X;P))$ and T. That is, instead of trying to fit the new image I to the reference T, we find a transformation parameters that best fit the reference T to the image I, and perform the opposite transformation on I. T his way, the 6×6 matrix in Equation 27 kept constant during all iterations, while the only changing matrix is the one on the right hand side, which depends on $I(W(X;P))$.

B Locally Linear Embedding (LLE)

Locally Linear Embedding (LLE) [18] is a flattening method designed for preserving the local structure of the data, and addressing the problem of nonlinear dimensionality reduction. The resulted embedding is optimized to preserve the local configurations of nearest neighbors, while assuming a local linear dependence between them. The 'neighborhood' definitions of each point is set by the user, and may include all points which distances from the specific point is smaller than a certain value, a fixed number of closest points, or any other reasonable decision.

The input of the conventional LLE algorithm is a set of multidimensional signals. The algorithm attempts to compute a low dimensional embedding with the property that nearby points in the high dimensional space remain nearby and similarly co-located with respect to one another in the low dimensional space. That is, the embedding is optimized to preserve the local configurations of nearest neighbors. The algorithm operates without recourse to measures of relation between faraway data points.

The LLE algorithm expects each data point and its neighbors to lie on a locally linear patch of the initial manifold or close to such one. Then, characterization of the local geometry in the neighborhood of each data point is done by linear coefficients that reconstruct the data point from its neighbors. Finally, embedding in a low dimensional space that preserves this local geometry characterization is sought. As in [18], we refer to 3 implementation stages in the algorithm:

1. Nearest neighbor search. Here we identify the nonzero elements of the weight matrix.
2. Weights calculation. The weights that reconstruct each data point from its neighbors are calculated, using a constrained least squares fits.
3. Computation of the low dimensional embedding. Here, the low dimensional embedding that best preserves the weights are found by solving an eigenvalue problem.

We now describe the implementation of each stage in the algorithm.

1. **Nearest neighbor search**

   The neighborhood definition is defined by the user. One option is to identify a fixed number of nearest neighbors per data point. Other possibilities are to choose as neighbors all points which distances from a specific data point, in some defined metric, is smaller than a certain value. The number of neighbors does not have to be the same for all data points. This neighborhood definition expresses the assumption on the shape or size of each closely-linear patch on the manifold.

2. **Weights calculation**

   The weights W_j^i are set to minimize the following cost function

$$E(W) = \sum_i \left| X_i - \sum_j W_j^i \eta_{ij} \right|^2 . \tag{30}$$

   We denote the neighbors of a point X_i as η_{ij}, where j goes from 1 to the number of neighbors of data point X_i. We turn Equation 30 into a quadratic form by

$$E(W^i) = \left| X_i - \sum_j W_j^i \eta_{ij} \right|^2 = \left| \sum_j W_j^i (X_i - \eta_{ij}) \right|^2 = \sum_{jk} W_j^i W_k^i \cdot g_{jk}^i . \tag{31}$$

   The first identity exploit the fact the the weights of each data point sum to one, and the value g_{jk}^i is defined as:

$$g_{jk}^i = (X_i - \eta_{ij}) \cdot (X_i - \eta_{ik}) . \tag{32}$$

We denote for each data point X_i the matrix G^i as $G^i_{jk} = g^i_{jk}$, this matrix is introduced in [18] as the "local" Gram matrix. Doing so, we get a quadratic equation of W -

$$E(W^i) = (W^i)^T \cdot G^i \cdot W^i. \tag{33}$$

The reconstruction error can be minimized analytically using a Lagrange multiplier to enforce the constraint that the sum of all components of W^i equals 1. We get

$$f = (W^i)^T \cdot G^i \cdot W^i + \lambda\left((w^i)^T \cdot \mathbf{1} - 1\right) \tag{34}$$

where $\mathbf{1}$ is a vector of 1. For finding the minimum we set the derivative to zero, and solve the linear system of equations $G^iW^i = \mathbf{1}$. We then scale the weights so that they sum to 1.

In terms of the inverse Gram matrix, the optimal weights are given by

$$W_{ij} = \frac{\sum_k G^{i^{-1}}_{jk}}{\sum_{lm} G^{i^{-1}}_{lm}} \tag{35}$$

We next describe the case where only pairwise distances between each two data points are given, instead of the coordinates themselves. The Gram matrix G^i can be derived from the pairwise distances in the following way. Let S denote the symmetric square matrix that records pairwise squared distances between the point X_i and its neighbors η_{ij}, so that $S_{ij} = d(X_i, \eta_{ij})$. In order to compute the Gram matrix G^i we can assume those points are centered in the origin, so that their inner products ρ_{ij} are given exactly in terms of their pairwise squared distances S_{ij} by

$$\rho_{ij} = \frac{1}{2}\left[\left(\frac{1}{K+1}\right)\sum_{k=0}^{k}(S_{ij} + S_{kj}) - \left(\frac{1}{K+1}\right)^2 \sum_{k,l=0}^{K} S_{kl} - S_{ij}\right]. \tag{36}$$

In terms of earlier notations, $\rho_{00} = |X_i|^2$,$\rho_{0j} = X_i \cdot \eta_{ij}$ (for $j > 0$), and $\rho_{jk} = \eta_{ij} \cdot \eta_{ik}$ (for $j, k > 0$).

Finally, the Gram matrix G^i can be derived from ρ_{ij} as follows,

$$G^i_{jk} = (X_i - \eta_{ij}) \cdot (X_i \cdot \eta_{ik}) = \rho_{00} - \rho_{i0} - \rho_{0j} + \rho_{ij}. \tag{37}$$

3. **Computation of the low dimensional embedding**

Now, that the weights W_{ij} are set, the low dimensional coordinates Y_i are those that minimize the cost function

$$\Phi(Y) = \sum_i \left|Y_i - \sum_j W^i_j Y_j\right|^2 \tag{38}$$

Rewriting this function in a quadratic form yields

$$\Phi(Y) = \sum_{ij} M_{ij}\,(Y_i \cdot Y_j), \tag{39}$$

where M is a square matrix given by $M = (I - W)^T (I - W)$, $W_{ij} = W_j^i$ and I is the identity matrix, or:

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{ij}. \tag{40}$$

Translation of all the coordinates Y_i by a constant displacement does not affect the cost function in Equation 38. Therefore, we remove this translational degree of freedom by requiring the outputs to be centered on the origin,

$$\sum_i Y_i = 0. \tag{41}$$

Setting all coordinates Y_i to zero denotes a trivial minimum value of zero to the cost function in 39. M is a real symmetric, and therefore Hermitian matrix. By the Rayleitz-Ritz theorem, this minimum value will also be the smallest eigenvalue of the matrix M. We therefore dismiss the smallest eigenvalue, and its corresponding eigenvector, which consists of all equal components. We then consider the next d eigenvectors, corresponding to the next-d-smallest eigenvalues, as the d-dimensional coordinates of the new embedding, that minimize 39. From orthogonality consideration of eigenvectors, the constraint in equation 41 is kept.

C Classical Scaling

Classical Scaling assumes that the dissimilarities between the images are Euclidean distances (in some D-dimensional world). Based on this assumption it reveals a centered configuration of points that best preserves, subject to Frobinius norm, those distances.

In a D-dimensional Euclidean space, each point P_j is represented as a coordinates vector $[x_j^1, x_j^2, ..., x_j^k]^T$. The global configuration of these points is defined by the matrix $P = [P_1, P_2, ..., P_n]^T$. The Euclidean distance between each two points is given by

$$\begin{aligned} d_{ij}^2 &= |P_i - P_j|^2 \\ &= \textstyle\sum_{l=1}^k \left(x_i^l - x_j^l\right)^2 \\ &= \textstyle\sum_{l=1}^k (x_i^l)^2 - 2x_i^l x_j^l + (x_j^l)^2 \\ &= P_i^2 - 2P_i P_j^T + P_j^2. \end{aligned} \tag{42}$$

We denote the matrix of pairwise square Euclidean distances as D. Let J be a centering matrix $J = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, where I is the identity matrix of size $n \times n$, $\mathbf{1} = [\underbrace{1, 1, 1...1}_{n}]^T$, and Q the $n \times n$ matrix $Q_{i,j} = |P_i|^2$. We have that $QJ = 0$, and also $(QJ)^T = JQ^T = 0$. Thus,

$$\begin{aligned} JDJ &= J(Q + Q^T - 2PP^T)J \\ &= JQJ + JQ^TJ - 2JPP^TJ \\ &= 0 + 0 - 2\tilde{P}\tilde{P}^T. \end{aligned} \tag{43}$$

The coordinate $\tilde{P}$ are those of P after centering, that is $\tilde{x}_i = x_i - \frac{1}{n}\sum_{k=1}^{n} x_i^k$. The matrix $-1/2JDJ$ is a real symmetric, and therefore normal matrix, that can be diagonalised $-1/2JDJ = U\Lambda U^T$. We sort the eigenvalues, and denote

$$\Lambda_{ii}^{+} = \begin{cases} \Lambda_{ii} & \text{if } \Lambda_{ii} > 0, i < d \\ 0 & \text{otherwise} \end{cases}$$

for extracting the centered coordinated in a d-dimensional world. We than multiply $\tilde{P} = U(\Lambda^{+})^{1/2}$.

If the points were originally taken from a d-dimensional Euclidean space, only the first d eigenvalues do not vanish and contains positive values, and it is sufficient to use only the first d eigenvectors from the matrix U for constructing an exact representation. Else, using only the first d eigenvectors we have a d-dimensional approximation of the input data, that preserves part of the energy, which is equal to

$$E = \frac{\sum_{i=1}^{d} \lambda_i^2}{\sum_{i=1}^{n} \lambda_i^2}. \tag{44}$$

Where λ_i is the i-th largest eigenvalue of the matrix $-1/2JDJ$ (the i-th largest diagonal element of $\Lambda^{1/2}$).

D Least Squares Multidimensional Scaling

Least Squares MDS [2] is another flattening method that reveals the global low dimensional geometric structure of a set of points based on their pairwise dissimilarity. The structure found is the one that minimizes the stress function

$$\sigma = \sum_{i<j} w_{ij} \cdot (d_{ij} - \delta_{ij})^2 \tag{45}$$

Where d_{ij} is the Euclidean distances between points i and j in the configuration, and δ_{ij} is the dissimilarity between points i and j, given as input. w_{ij} is a weight that represents the importance of preserving the dissimilarity between points i and j as Euclidean distance.

Least Squares MDS uses a method called Iterative Majorization [2] in order to find a global structure of points that minimizes the stress.

The main idea of the majorization method is to replace iteratively the original function $f(x)$ by an auxiliary function $g(x, z)$, where z is some fixed value, that satisfies,

- $g(x, z)$ is more simple to minimize than $f(x)$.
- $f(x) < g(x, z)$.
- $f(z) = g(z, z)$.

Finding the minimum value of $f(x)$ includes choosing some initial value z^0, and defining the function $g(x, z^0)$. Then, finding the value x_{min} that minimizes that function ($g(x, z)$ is a simple, usually a quadratic function, therefore, finding its minimum is easy). From the properties defined above we get $f(x_{min}) \leq f(z)$. Then, we iteratively define z^k to be x_{min} from the previous iteration. The inequality $f(x_{min}) \leq f(z^k)$ stays valid throughout the algorithm, which stops when $f(x_{min}) - f(z^k) \leq threshold$.

We will now find the auxiliary function $g(x,z)$, and this will complete this method description.

Lets, first develop the expression for the stress function

$$\begin{aligned}\sigma &= \textstyle\sum_{i<j} w_{ij}\cdot(d_{ij}-\delta_{ij}(X))^2\\ &= \textstyle\sum_{i<j} w_{ij}\delta_{ij}^2 + \sum_{i<j} w_{ij}d_{ij}^2(X) - 2\cdot\sum_{i<j} w_{ij}\delta_{ij}d_{ij}(X)\\ &= \eta_\delta^2 + \eta^2(X) - 2\rho(X)\end{aligned} \tag{46}$$

The stress function contains three parts, and we will analyze each of them separately. The first part $\eta_\delta^2 = \sum_{i<j} w_{ij}\delta_{ij}^2$ is a constant that depends only on the dissimilarity matrix that was given as input. For analyzing the second component $\eta^2(X) = \sum_{i<j} w_{ij}d_{ij}^2(X)$, lets first explore the Euclidean distance between points i and j in a given configuration X, in an m-dimensional Euclidean space,

$$\begin{aligned}d_{ij}^2 &= \textstyle\sum_{t=1}^m (X_{it}-X_{jt})^2\\ &= \textstyle\sum_{t=1}^m X_t'(e_i-e_j)(e_i-e_j)'X_t\\ &= \textstyle\sum_{t=1}^m X_tA_{ij}X_t\\ &= trX'A_{ij}X\end{aligned} \tag{47}$$

Where X_t is the t-th column of the configuration matrix X, e_i is the e-th column of the identity matrix I, and A_{ij} is simply the matrix with $a_{ii} = a_{jj} = 1$, and $a_{ij} = a_{ji} = -1$, and all other elements are zero. The weighted sum will therefore be

$$\begin{aligned}w_{ij}d_{ij}^2 &= w_{ij}trX'A_{ij}X\\ &= trX'w_{ij}A_{ij}X,\end{aligned} \tag{48}$$

and summing over all $i<j$ gives

$$\begin{aligned}\eta^2(X) &= \textstyle\sum_{i<j} w_{ij}d_{ij}^2(X)\\ &= trX'\left(\textstyle\sum_{i<j} w_{ij}A_{ij}\right)X\\ &= trX'VX.\end{aligned} \tag{49}$$

Where

$$V_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j\\ \sum_j w_{ij} & \text{if } i = j\end{cases}$$

The above expression is equal to $\eta^2(X)$ (at all points) and quadratic in X, and therefore is a good majorization function for $\eta^2(X)$.

We next find a majorization function for the third expression $-\rho(X) = -\sum_{i<j} w_{ij}\delta_{ij}d_{ij}(X)$. We start by using the Cauchy-Schwarz inequality,

$$\sum_{t=1}^m p_aq_a \leq \left(\sum_{t=1}^m p_t^2\right)^{\frac{1}{2}}\left(\sum_{t=1}^m q_t^2\right)^{\frac{1}{2}}. \tag{50}$$

If we substitute p_t by $(X_{it} - X_{jt})$, and q_t by $(Z_{it} - Z_{jt})$ we get:

$$\begin{array}{rcl} \sum_{t=1}^{m}(X_{it}-X_{jt})(Z_{it}-Z_{jt}) & \leq & (\sum_{a=1}^{m}(X_{it}-X_{jt})^2)^{\frac{1}{2}}(\sum_{a=1}^{m}(Z_{it}-Z_{jt})^2)^{\frac{1}{2}} \\ & = & d_{ij}(X)d_{ij}(Z). \end{array} \tag{51}$$

From the above, we can get an expression for $d_{ij}(X)$,

$$-d_{ij}(X) \leq -\frac{\sum_{t=1}^{m}(X_{it}-X_{jt})(Z_{it}-Z_{jt})}{d_{ij}(Z)} \tag{52}$$

With equality if $Z = X$. If $d_{ij}(Z) = 0$ for a particular i and j, than $d_{ij}(X)$ is undefined. In those cases we will replace the fraction in the right hand side of Equation (52) with zero, while it still holds that $-d_{ij}(X) \leq 0$. We now sum over all $i < j$ and get the (linear) majorization function for the third term,

$$\begin{array}{rcl} -\rho(X) & = & -\sum_{i<j} w_{ij}\delta_{ij}d_{ij}(X) \\ & \leq & -trX'\left(\frac{w_{ij}\delta_{ij}}{d_{ij}(Z)}A_{ij}\right)Z \\ & = & -trX'B(Z)Z. \end{array} \tag{53}$$

Where,

$$\begin{array}{l} B_{ij} = \begin{cases} -\frac{w_{ij}\delta_{ij}}{d_{ij}(Z)} & \text{if } i \neq j \text{ and } d_{ij} \neq 0 \\ 0 & \text{if } i \neq j \text{ and } d_{ij} = 0, \end{cases} \\ B_{ii} = \sum_{j=1, j\neq i}^{n} B_{ij}. \end{array}$$

The auxiliary function $g(x, z)$ for the stress function is therefore

$$g(x,z) = \sum_{i<j} w_{ij}\delta_{ij}^2 + tr(X'VX) + tr(X'B(Z)Z), \tag{54}$$

where B and V are as defined before.

We next summarize the whole process of minimizing the stress function

1. Set initial configuration of points $X^{[0]}$.
2. Compute σ_r^0.
3. Set $K = 0$.
4. $K = K + 1$.
5. Update the configuration $X^{[K]}$ by minimizing the majorization function from Equation (54).
6. Compute σ_r^K
7. if $\sigma_r^{K-1} - \sigma_r^K < \textit{threshold}$ or K has reached the pre-defined maximum number of iterations, return $X^{[K]}$. Otherwise, return to step 4.

The initial configuration is critical, because the stress function has a lot of local minima. In our application, we used the solution of the classical scaling for the initial configuration. Malone, Tarazaga and Trosset [21] suggest to multiply this solution by a scalar t_2,

$$t_2 = \left(\frac{\langle D_2(X), \Delta_2 \rangle_F}{\|D_2(X)\|_F^2} \right)^{1/2} \tag{55}$$

where $D_2(X)$ is the squared Euclidean distances matrix of a configuration X, Δ_2 is the squared dissimilarity matrix and $\langle\rangle_F$ and $\|\|_F$ are the Frobenious inner product and norm, respectively. This modification did not result in any improvement of the final stress in our application.

References

[1] J. R. Bergen, P. J. Burt, R. Hingorani, and S. Peleg. A three-frame algorithm for estimating two component image motion. *IEEE Trans on PAMI*, 14(9), 1992.

[2] I. Borg and P. Groenen. *Modern Multidimensional Scaling - Theory and Applications.* Springer-Verlag New York, Inc, 1997.

[3] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. *Computer Graphics*, 31:353–360, August 1997.

[4] C. Bregler, H. Hild, S. Manke, and A. Waibel. Improving connected letter recognition by lipreading. *in Proc. IEEE Int. Conf. on ASSP*, pages 557–560, 1993.

[5] C. Bregler and S. M. Omohundro. Surface learning with applications to lipreading. In *NIPS*, volume 6, pages 43–50, 1994.

[6] C. Bregler, S. M. Omohundro, M. Covell, M. Slaney, S. Ahmad, D. A. Forsyth, and J. A. Feldman. Probabilistic models of verbal and body gestures. *Computer Vision in Man-Machine Interfaces (R. Cipolla and A. Pentland eds), Cambridge University Press*, 1998.

[7] J. B. Tenenbaum V. de Silva and J. C. Langford. A global geometric frame-work for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[8] D. L. Donoho and C. E. Grimes. Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. In *Proceedings of the National Academy of Arts and Sciences*, volume 100.

[9] P. Duchnowski, M. Hunke, D. Bsching, U. Meier, and A. Waibel. Toward movement-invariant automatic lipreading and speech recognition. In *Proc. ICASSP '95*, pages 109–112, 1995.

[10] C. G. Fisher. Confusions among visually perceived consonants. *Journal of Speech and Hearing Research*, (15):474–482, 1968.

[11] D. W. Jacobs, P. N. Belhumeur, and R. Basri. Comparing images under variable illumination. In *Proc. of CVPR*, pages 610–617, 1998.

[12] G. A. Kalberer and L. Van Gool. Lip animation based on observed 3d speech dynamics. In S. El-Hakim and A. Gruen, editors, *Proc. of SPIE*, volume 4309, pages 16–25, January 2001.

[13] N. Li, S. Dettmer, and M. Shah. Visually recognizing speech using eigensequences. In *Motion-Based Recognition*, pages 345–371. Klwer Academic Publishing, 1997.

[14] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[15] J. Luettin. *Visual Speech And Speaker Recognition.* PhD thesis, University of Sheffield, May 1997.

[16] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, December 2000.

[17] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. ASSP*, ASSP-26:43–49, February 1978.

[18] L. K. Saul and S. T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, pages 119–155, 2003.

[19] E. L. Schwartz A. Shaw and E. Wolfson. A numerical solution to the generalized mapmaker's problem: Flattening nonconvex polyhedral surfaces. *IEEE Trans on PAMI*, 11(9):1005–1008, September 1989.

[20] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[21] S. W. Malone P. Tarazaga M. W. Trosset. Optimal dilations for metric multidimensional scaling. In *Proceedings of the Statistical Computing Section*, 2000.

[22] P. Vanroose, G. A. Kalberer, P. Wambacq, and L. Van Gool. From speech to 3D face animation. *Procs. of BSIT*, 2002.