# VISUAL SPEECH RECOGNITION BY RECURRENT NEURAL NETWORKS



GIHAD RABI

# VISUAL SPEECH RECOGNITION BY RECURRENT NEURAL NETWORKS

by

©Gihad Rabi

A thesis submitted to the

School of Graduate Studies

in partial fullfilment of the

requirements for the degree of

Master of Science

Department of Computer Science
Memorial University of Newfoundland

February 1997

St. John's Newfoundland

## Abstract

One of the major drawbacks of current acoustically-based speech recognizers is that their performance deteriorates drastically with noise. The focus of this thesis is to develop a computer system that performs speech recognition based on visual information of the speaker. The system automatically extracts visual speech features through image processing techniques that operate on facial images taken in a normally-illuminated environment. To cope with the dynamic nature of change in speech patterns with respect to time as well as the spatial variations in the individual patterns, the recognition scheme proposed in this work uses a recurrent neural network architecture. By specifying a certain behavior when the network is presented with exemplar sequences, the recurrent network is trained with no more than feed-forward complexity. The network's desired behavior is based on characterizing a given word by well-defined segments. Adaptive segmentation is employed to segment the training sequences of a given class. This technique iterates the execution of two steps. First, the sequences are segmented individually. Then, a generalized version of dynamic time warping is used to align the segments of all sequences. At each iteration, the weights of the distance functions used in the two steps are updated in a way that minimizes a segmentation error. The system has been implemented and tested on a few words and the results are satisfactory. In particular, the system has been able to distinguish between words with common segments. Moreover, it tolerates, to a great extent, variable-duration words of the same class.

## Acknowledgements

I wish to express my thanks to my supervisor Dr. Siwei Lu for his insightful guidance, constant encouragement and support. Without his contribution, it would have been impossible to give this thesis its current quality.

I would like to thank the systems support staff, and in particular Nolan White, for providing help and assistance during my research.

Special thanks are due to the staff of the Division of Educational Technology who assisted me in obtaining the data necessary for my experiments.

I am also very grateful to the administrative staff, especially Elaine Boone, who have helped in the preparation of this thesis.

I would like to acknowledge the financial support received from the School of Graduate Studies and the Department of Computer Science.

I would also like to thank my fellow graduate students and dear friends for the continuous encouragement and support.

*This thesis is dedicated to my parents*

*for their encouragement and support*

*throughout the course of my education.*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years. there has been a growing interest in automatic speech recognition. The benefits that could be brought by systems capable of understanding spoken language are great as people would be able to interface with sophisticated machines in a natural way without the hassle of pressing complicated sequences of buttons or typing commands. However. the performance of current speech recognizers is far below human ability to perceive speech. One of the major drawbacks of acoustic systems is that their performance deteriorates drastically with noise. Also. there is the problem of speaker isolation which occurs when several people talk at the same time. and it is required to identify the intended speaker. Moreover, the fact that some phonemes are very difficult to distinguish by analyzing the acoustic signal alone poses additional limitations to current approaches.

It is known that hearing-impaired people use lipreading successfully to perceive speech in the absence of acoustic information. Surprisingly, even normal hearing people utilize visual information of the speaker's face for speech perception [12]. It

has been demonstrated, through what is called the *cocktail party* effect, that with a high background noise it is easier for humans to understand speech when they watch the lips of the speaker.

The performance of machines in speech recognition could improve, too, by processing visual information. This would be particularly useful in the presence of high background noise or in the case of crosstalk. It is worth mentioning, in this context, that acoustic and visual information complement each other in speech characterization. That is, similar phonemes are often easy to distinguish visually, whereas utterances which look very similar visually can sound quite different.

The focus of this thesis is to develop a computer system that performs speech recognition based on visual information of the speaker. A speech recognition system which uses visual information alone would, probably, have limited applications since several groups of phonemes appear similar visually. Nevertheless, there is no doubt that combining such a system with an acoustic speech recognizer would result in an improved recognizer which overcomes many of the existing problems in acoustic systems, such as the low performance in noisy environments.

Neurocomputing is an attractive choice for visual speech recognition (VSR). It leads to systems that autonomously develop operational capabilities in adaptive response to an information environment that is notoriously difficult to model using conventional methods. The difficulty of VSR can be viewed in light of two main aspects of VSR which I would like to refer to as the *static aspect* and the *dynamic*

*aspect* of VSR.

1. There is a variety of speakers with different physical characteristics including the shape of lips, jaws and so on. The lips of a certain person are subject to a variety of changes in shape as well.

2. There are a lot of dialects, and there are different ways of uttering a certain word, even by the same person, on different occasions.

Artificial neural networks possess an attractive property which makes them suitable for dealing with the implications of the static aspect of VSR. This property is the ability to learn a classification task by observing only a limited number of examples. Neural networks can also discover distinguishing features in the training patterns while making weaker assumptions about the shapes of underlying distributions than those made by traditional statistical classifiers. This is particularly important if we keep in mind the lack of a comprehensive theory of lipreading. Furthermore, neural networks are known to be excellent at noise tolerance, which is an indispensable requirement for any practical system. The motivation for exploring recurrent architectures is their potential for dealing with the temporal behavior implied by the dynamic aspect of VSR.

While some previous attempts at automatic lipreading required human intervention in obtaining the visual speech signal or making the speaker wear reflective markers on his face, our system automatically extracts visual speech features through

image processing techniques that operate on facial images taken in a normally illuminated environment and without any need for reflective substances.

To properly handle the dynamic nature of change in speech patterns with respect to time as well as the spatial variations in the individual patterns, the recognition scheme proposed in this work uses recurrent neural networks. Input to the networks are sequences of low-dimensional patterns rather than matrices of pixels. This leads to a smaller architecture in size, and a shorter time for training.

## 1.1 Overview of the System

Figure 1.1 shows an overview of the system in operation. As in most connectionist approaches, the system needs to be trained on exemplar sequences. Given a set of image sequences corresponding to some word class, the system is trained as shown in Figure 1.2.

## 1.2 Organization of the Thesis

Chapter 2 surveys methods for locating the mouth in digital images, methods for extracting mouth features, techniques for mouth modeling, previous lipreading systems, and neural networks. Chapter 3 presents the method by which the speaker's mouth is automatically located in the input images. Chapter 4 describes the algorithms developed to estimate the mouth characteristics that are used to initialize a mouth *deformable template*. Chapter 5 is devoted to the application of deformable templates in extracting the shape of the mouth and tracking its movement during

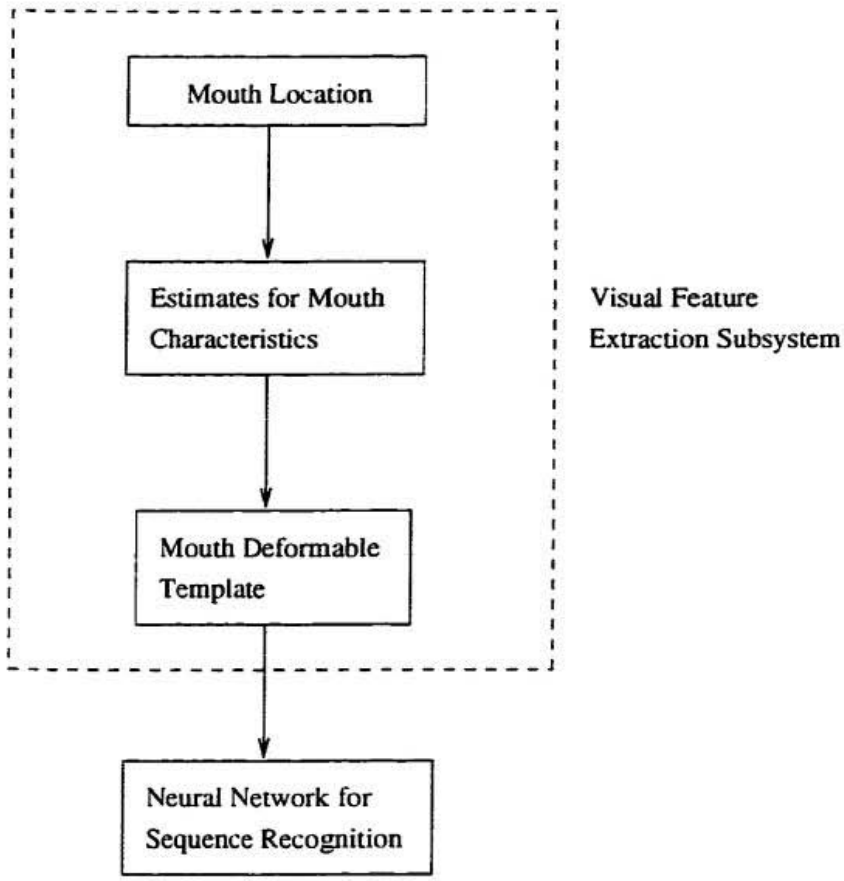


Figure 1.1: Overview of the visual speech recognition system.

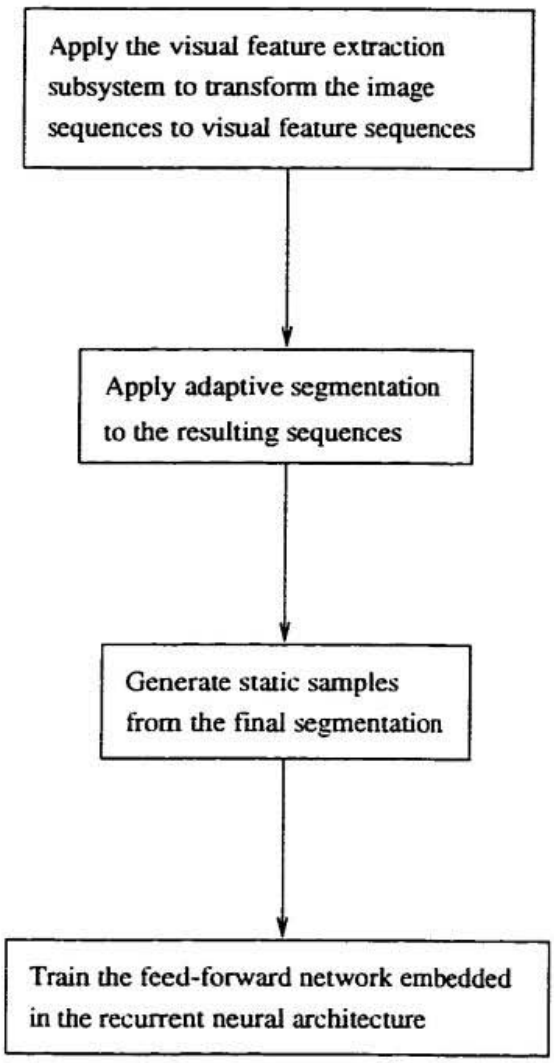


Figure 1.2: Overview of the process for training the recognition system.

speech. Chapter 6 contains a method to segment a visual speech signal, representing a word instance, into visual speech units. A technique to refine the resulting segmentation by aligning instances of the same word is described. Then, the idea of *adaptive segmentation* is introduced with a proposed implementation for the case of linear distance functions. In Chapter 7, a recurrent neural network for word recognition is presented. An efficient way for training the recurrent network based on visual word segmentation is proposed. Chapter 8 contains the main conclusions, simulation results and directions for future research.

# Chapter 2

# Survey

## 2.1 Methods to Locate the Mouth

Locating the mouth automatically has been a concern in the areas of lipreading, face recognition, and speech-assisted video processing. In general, a computer system designed for this purpose consists of two main steps. First, an input image is preprocessed and a new representation is obtained. The preprocessing step not only aims at reducing the amount of information involved in subsequent operations, but also, and perhaps more importantly, directs the search toward interesting regions of the image. The second step, basically, identifies the region containing the mouth by searching for some characteristics associated with it. The characteristics could be particular to the appearance of the mouth only, or could be derived from knowledge about the face context as well. Thus, the search may involve other facial features in addition, or instead of, the mouth.

One of the first attempts to locate facial features automatically was documented by Baron [2]. He proposed locating facial features by correlation. To locate a feature,

a set of feature masks stored in a database were correlated against each subimage of the input, and the desired location was selected at the subimage with the highest correlation. Although the reported results were good, the system is expected to have been highly sensitive to lighting conditions. This is mainly due to the fact that raw grey scale pixels were used instead of some representation that tolerates slight changes in intensity. In addition, the method is computationally expensive due to its exhaustive nature, and sensitive to scaling and rotation.

Prasad *et al.* [53] [72] implemented a system which detects a region of interest containing the mouth. They proposed two approaches: the first starts by obtaining a binary edge image, and is followed by blob detection. The blobs corresponding to the eyes and mouth can be found, next, by locating three blobs whose centroids form a triangle on which certain constraints apply. In the second approach, temporal coherence between consecutive image frames is used. This alternative approach, however, assumes that the only change in the image frames is in the lips positions. If this is not the case, then the approach is expected to fail.

Craw, Ellis and Lishman [18] described software that makes facial feature measurements. Their preprocessing is based on extracting edge information. In addition to calculating edge magnitudes by a Sobel filter, edge directions are obtained by the same way described by Kelly [35]. After preprocessing, the basic search technique used is line following. At each pixel in a central vertical line, a search for outlines of the upper and lower lips is carried on. A lip is detected if the vertical separation

between the upper and lower lips is within a reasonable range, and the extracted lip combination fits into a long thin box. One drawback of this method is that the criteria selected to identify a lip contour could be, possibly, matched by contours near to the lips such as a mustache and wrinkles. Also, this method does not make use of the relative magnitude of edges associated with the different lines in the search space.

Huang and Chen [31] employ thresholding for the preprocessing. A *scale space filter* (SSF) is used to determine the zero-crossings of the intensity histogram at different scales, and a set of thresholds is determined accordingly. After thresholding, a *rough contour estimation routine* (RCER) operates on the image. The location of the mouth comes after a sequence of estimations in which a particular feature is estimated based on a previous one.

Morphological operations [42] can also be used for preprocessing. Chow and Li [15] apply a *morphological opening residue* operation to extract all intensity valleys using a circle mask. Similarly, Chen, Graf and Wang [14] [23] perform a morphological operation to pick out areas with strong variations in intensity. The detected pixels in the result are assembled into distinct regions which are, in turn, grouped into plausible face contexts. Then, an evaluation measure is used to rank the possible face contexts. Chow and Li proposed a control strategy in which the face context with the highest rank would be verified in a later stage. However, they did not provide a good qualitative measure to indicate failure of hypothesized contexts.

Kanade [33] incorporated an interesting control strategy in his system which seeks extraction of facial features. There is a similarity between his approach and that used by Chow and Li in the concept of feedback. However, no attempt is made by Kanade to evaluate full face contexts within a single level. instead. the final face context is reached by a sequence of operations in separate levels. In each level. a procedure locates a set of features. and these are used in a subsequent level to specify the search space for another set of features. At the end of each block. its performance is evaluated and accordingly a decision is made whether to proceed to the next block or to backtrack if a failure occurs. Preprocessing consists of applying a Laplacian operator followed by thresholding. The fundamental technique used to conduct the actual search is the *integral projection* technique.

Inspired by the intuitive notion of symmetry. Reisfeld. Wolfson. and Yeshurun [57] proposed a low-level operator that performs local processing on the edges of an image. and assigns a continuous symmetry measure. consisting of magnitude and orientation. to every point. In a recent work by Intrator. Reisfeld. and Yeshurun [32]. this symmetry operator is used to preprocess the image of a human face. The candidates for the eyes and mouth are obtained from the highest peaks of the *radial symmetry map*.

Yacoob and David [74] locate the mouth as part of their comprehensive approach to labeling human face components. The input to their system is range data which has the advantage of reflecting topographic features of faces, and is, to a great extent,

insensitive to illumination conditions and projective transformations. To cope with the nonrigidity of faces, Yacoob and David propose a preprocessing stage which employs a *multistage diffusion* procedure. The diffusion procedure simulates the propagation of a number of particles among object voxels. The output is an image in which larger values correspond to more convex surface points. The connected components in this representation are identified. Then, context-based reasoning, in the form of a consistency operator, is used to label the components.

## 2.2 Methods to Extract Mouth Features

Prasad *et al.* [53] [72] examine two grey scale profiles within a region of interest (ROI) containing the mouth: the first profile is along a vertical central line and the position of the lowest valley in this profile is used to estimate the vertical position of the mouth centroid. The grey level value of the same valley is used to indicate the presence or absence of the tongue. The peaks adjacent to the mouth centroid valley correspond to the upper and lower lips and their linear separation is used to estimate the height of the mouth opening. Presence of teeth is indicated by the grey level value of the peaks. From the second grey level profile, which is along a horizontal central line, the width of the mouth opening is estimated from the linear separation between the peaks adjacent to the lowest valley in this profile.

In their procedures, Prasad *et al.* assume that the central horizontal and vertical lines always intersect with the mouth. While this assumption is reasonable for the

vertical line as the width of the mouth is about 80% of the width of the ROI (so an exact centralization is not necessary for the intersection to occur), this may not be the case for the horizontal line. Vertically, the mouth extension is less than 25% of the ROI, and since there is no guarantee that the mouth will be exactly centered in the ROI, there is a chance that the intersection will not occur. In this case or even when the line intersects exclusively with one of the lips, the estimated value obtained for the width of the mouth opening is not reliable.

Coianiz, Terrasani, and Caprile [16] use chrominance analysis to locate a set of feature points acting as constraints on the shape of the mouth. The input to their system is a fixed lower part of a person's face image. The original image is, first, preprocessed by filtering the hue component with a weight function that emphasizes red dominant regions and this is followed by thresholding with a value that can be determined empirically. Then, the spatial distribution of the pixels is analyzed to determine the center of the distribution, and its horizontal and vertical standard deviations. These measurements are used to bound the mouth in a rectangular region. Within this rectangle, the mouth vertices (corners) are localized by obtaining a binary gradient image from grey level information. The binary gradient image is projected on the horizontal axis, and the $x$-coordinates of the vertices are defined as the most external non-null points in this projection. The $y$-coordinate of a vertex is determined by examining the projection of a vertical stripe around the vertex onto the vertical axis. The distribution center of the projection is taken as the $y$-coordinate.

Once the vertices are determined, the mouth vertical symmetry axis can be correctly localized. The hue-filtered values in a thin box around this axis are projected onto the vertical axis by adding the values at each row. By analyzing this distribution, the apices, which are the points where the external and internal boundaries of the upper and lower lips intersect the vertical symmetry axis, can be determined. If the distribution has a single peak, the mouth is considered closed and the internal points merge in a single point lying on the segment joining the two corners. The external points are located at the rows having 70% of the peak value. If the mouth is open, two peaks which correspond to the upper and lower lips are expected, and the four points are obtained by cutting each peak at 70% of its value.

This color-based system has the advantage of being less sensitive to shadows than are grey level-based methods. However, there are cases that may pose a challenge to this approach. For example, existence of reddish facial hair such as a beard or mustache will, likely, confuse the system by emphasizing other facial parts in addition to the lips. In other cases, the appearance of the lips' natural color may be modified through the use of lipsticks with different colors such as pink or beige.

Chen, Graf and, Wang [14] [23] attempt to find the same features that Coianiz *et al.* extract, but their approach is different. First, the inner part of the mouth is marked using connected component analysis. The resulting connected blob is compared with a library of prototypes to make a qualitative judgement about the shape of the mouth. Possible shapes include open mouth, closed mouth, and visible

teeth. The width-to-height ratio of the blob, as well as the number of transitions from black to white in a central vertical cross section, are used in the comparison. From the previous information, the center of the mouth is estimated and several vertical cross sections through the lips are analyzed to measure the types of intensity variations that are present across the lips. The measured intensities are compared with 15 prototypes which are distinguished on the basis of the number of maxima and their relative positions, and the closest prototype is selected. Each prototype is associated with a strategy for finding the edges of the lips. The strategies were made different to handle different cases of illumination and contrast between the lips and the surrounding skin.

Rao and Mersereau [56] search for a set of points lying on the top of the upper lip and the lower lip. The vertical position of the center of the mouth is determined by examining the sum of intensity values in each row and selecting the row which has the minimum value of the distribution. The pixels in the previously selected row are analyzed to find out the horizontal extremities of the mouth. For this purpose, the average of the maximum and minimum values in this row is selected as a threshold and the leftmost and rightmost pixels with values under this threshold are defined to be the corners. Finally, three intermediate equally distanced columns are analyzed and two points in each column are identified by peak picking, one on the top of the upper lip and the other in the middle of the lower lip.

## 2.3 Methods to Model the Mouth

### 2.3.1 Active Contour Models (Snakes)

A *snake* [43] is a model to represent contours in images. It consists of a set of labeled points and an energy function designed to take minimum values when the points match some object's boundary. In general, there are three terms that contribute to the energy of a snake: internal energy, image energy, and constraints.

Properties particular to the shape of the spline obtained by linking consecutive points are controlled by the internal energy. These properties include continuity and curvature, and are usually controlled by the first and second derivatives at each point. On the other hand, the image energy attracts the snake toward features like lines and edges. The constraints represent the energy of a spring connected between a point on the contour and some point in the plane.

Kass *et al.* [43] have proposed minimizing the energy function by means of variational calculus techniques. Although the computational requirements of this approach are linear, some related problems, as pointed out by Amini *et al.* [1], are instability and a tendency for points to bunch up on strong portions of an edge. To overcome these problems, Amini *et al.* [1] have proposed a time-delayed discrete dynamic programming algorithm. This approach provides necessary and sufficient conditions for the optimality of the solution and has the advantage that hard con-

straints, in addition to the soft constraints inherited in the original formulation, can be included. However, the method is relatively slow. Williams and Shah [70] suggest a greedy algorithm which is much faster. Their approach, however, does not guarantee a global minimum, but Williams *et al.* argue that the results obtained by the greedy algorithm are comparable to those of the dynamic programming algorithm.

The original model developed by Kass *et al.* [43] can be described as knowledge-free. Bregler *et al.* [6] [7] [8] report that this kind of snake model sometimes relaxes on undesirable features. In their investigation, they show a lip-snake relaxing on the contour of the nostrils region. To improve the model, they propose analyzing a large set of possible snake shapes. The process, called *surface learning* [7], induces a low-dimensional subspace from the high-dimensional data. The internal energy, in this case, can be replaced by the nearest distance to the learned surface of legal shapes. This modified model can be categorized as a link between the active contour model and the active shape model.

### 2.3.2 Active Shape Models

*Active shape models* (ASMs)[17] are statistically-based flexible models which represent objects by sets of labeled points. Though similar to the snakes of Kass *et al.* ASMs make no heuristic assumptions about legal shape deformation. Instead, legal deformation is obtained by applying *principal components analysis* on a normalized training set. The purpose is to derive a point distribution model which describes the average shape and the main modes of variation. Another important difference is that,

rather than assuming the points should lie on strong edges and searching for such in an image, the ASM approach assumes that grey level patterns about a particular point in images of different examples will often be similar. Accordingly, principal components analysis is used to produce a statistical model of the allowable variation in one-dimensional profiles normal to the curve at each point. Luettin, Thacker, and Beet [41] modified this by concatenating the profiles of all model points to produce a global profile for each training image.

Cootes, Hill, Taylor, and Haslam [17] propose an iterative method to guide the search in ASMs. At each iteration, a region around every point is examined and the displacement required to move the point to a better location is calculated. Next, the model parameters are updated according to the previous displacements, but limits on the parameters are enforced to ensure that the shape remains similar to that of the training set.

### 2.3.3 Deformable Templates

A *deformable template* [77] [76] is a geometric model associated with an energy function that measures how well the model matches a particular object in an image. The template is made up of geometric primitives linked in a certain manner. A geometric primitive could be any curve described by a mathematical formula. This has the advantage of a compact representation in terms of the parameters of the curves involved, and gives the model the flexibility of deformation to a wide variety of shapes by changing the parameter values. Undesirable parameter configurations

can be discouraged by adding penalty terms to the energy function. To account for the rotation the object to be extracted may undergo, it is sometimes useful to have a parameter for the orientation of the model, in addition to the origin of the model's coordinate system.

The energy function can be thought of as the link between the geometric model and the image, as it enables the necessary interaction to attract the template to salient features of the image, such as edges, peaks, and valleys of the intensity. Typically, the contribution of each curve to the energy is expressed as the integral of an image potential field along the curve. Sometimes, properties of pixels in a closed region are reflected by the integral of the image field as it relates to that property over the closed region.

Finding the best parameter values that would make the template fit into the image is equivalent to optimizing the energy function. Numerical optimization techniques which don't guarantee a global optimum solution are the most commonly used for this task. They have the advantage that a good solution is generally obtained with considerably less computational and storage requirements than those of a more exhaustive method. Yuille *et al.* [77] have used *steepest descent* of the energy function in parameter space. At each iteration, every parameter is updated by the negative of the partial derivative of the function with respect to that parameter. However, the calculation of partial derivatives at every point is an exhaustive task. *Simulated annealing* can also be used in a similar way to that reported in [19]. It is based on

random sampling and no partial derivatives are involved. However, a relatively large sample size is needed. Chow and Li [15] have adapted the *downhill simplex* method proposed by Nelder and Mead [47] with some modifications which include randomizing the initial simplex configuration to ensure that a good sample is achieved over the entire search space. Coianiz *et al.* [16] have used a stochastic optimization algorithm described by Caprile and Girosi [13]. Xie *et al.* [73] have employed the L-M method of Lavenberg and Marquardt [61].

Regarding the selection of the coefficients for the energy function terms, experimentation is usually used. In the absence of a solid theoretical basis that explains the different interactions, heuristics seem to provide a practical mechanism for assigning initial values. Then, fine tuning is done by adjusting certain coefficients individually while fixing the others. The extent to which the selection of suitable coefficients may affect general performance is not well known. This issue is further complicated by uncertainties concerning other design decisions involved in the deformable template model, such as the image potential fields used, the energy function terms chosen, and the optimization technique employed.

## 2.4 Previous Speechreading Systems

Yuhas, Goldstein, Sejnowski and Jenkins [75] trained a multi-layer feedforward neural network on static images of mouth shapes for vowel recognition. The neural network was not trained to classify the images directly, but rather to estimate the *short-*

*term spectral amplitude envelope* (STSAE) of the acoustic signal. The estimated STSAE was combined with the power spectrum of the noise-degraded audio signal, and the result was presented to another neural network classifier. Several strategies for combining the two signals were explored including the average, weighted average, and $\sigma$-$\pi$ neural networks. It is clear that the applicability of this system was very limited because it was not designed to deal with temporally changing patterns.

Mase and Pentland [34] employed optical flow methods to estimate mouth opening and elongation velocities. A standard minimum distance classifier was used to match test utterances with previously-stored templates, after applying linear time warping.

Stork, Wolff and Levine [64] used a time delay neural network to recognize ten consonants from sequences of visual features. These features were detected automatically, but required that the speaker wears reflective markers around his or her mouth. Wolff, Prasad, Stork and Hennecke [72] built upon this recognizer, and replaced the requirement of the reflective substance by preprocessing algorithms that extracted features from grey level images.

Petajan [51] extracted mouth opening features from each image in a sequence of images using a simple thresholding technique, then employed linear time warping to match the extracted sequence with exemplar sequences. The linear time warping algorithm allowed only for simple dilation and contraction of time, which do not account for all natural speech variations [62]. Brooke and Petajan [10], in a different work, modified the previous system by using dynamic time warping for matching. In

both systems, there was a problem in the method used to isolate the mouth opening. It is unlikely that the threshold that worked for a particular speaker would work for others, especially with variations in skin darkness and presence or absence of facial hair.

Goldschen, Garcia and Petajan [22] described an optical speech recognizer that used information from the oral cavity shadow of a speaker's mouth. They did not mention, however, how they located and extracted the oral cavity region. During training, principal component analysis was performed on seven static oral-cavity features (area, width, height, rounding, perimeter, number of connected regions in a component frame, and number of regions in a binary image frame), their first and second derivatives with respect to time, and the magnitude of those derivatives. This resulted in thirteen features to be considered. In operation, the processing started by converting a sequence of oral cavity regions to a sequence of the thirteen features. This latter sequence was, in turn, transformed to a sequence of codevectors using the Euclidean distance. Recognition was done, next, by hidden Markov modeling.

In his visual speech recognizer, Movellan [44] took the approach of preserving the original images and letting the recognition engine discover relevant features. He used sequences of processed mouth images. These were composites of portions from different representations obtained by symmetry enforcement, temporal differentiation, subsampling and logistic thresholding. These images were modeled as mixtures of independent Gaussian distributions and the temporal dependencies were captured

with standard hidden Markov models. An obvious disadvantage in such a recognizer is the size of the input data; 300 pixels for each image in the input sequence.

Li, Dettmer and Shah [38] proposed *eigensequences* for lipreading. They used the spatiotemporal eigen decomposition, in which the set of eigenvectors spans the space of all possible sequences. Gray level values of all the pixels in all frames representing a spoken letter were put in one vector. Several of those training vectors that corresponded to a certain class were used to compute the eigenvectors of that class. Recognition was performed by computing the energy ratio when the sequence to be recognized was projected on the model eigenspace for each class. In such a scenario, a certain class was supposed to have a high energy ratio when presented with a correct instance.

Kirby, Weisser and Dangelmayr [36] coded a mouth image into a vector of $Q$ coefficients computed with respect to the set of $Q$ eigen images determined during training as the basis for the space of mouth images. A certain word of $P$ images was, accordingly, represented by a $Q \times P$ matrix. A template-matching technique based on the Euclidean distance was used to identify the words.

Bregler and Konig [8] combined both acoustic and visual data. The visual data consisted of the first ten principal components of a grey level matrix centered around the lips (eigenlips). The data was fed to a multi-layer feedforward neural network in order to estimate the probability of a certain phone, given the acoustic and visual data at each time instance. The probabilities were used by an HMM-based system to

recognize German letters. The grey level matrix coding was invariant against shifting and scaling, but not lighting. Bregler and Konig found that the first principal grey level axis represented variations in lighting. In general, there is no guarantee that the first ten principal components correspond to the ten most relevant features of visual speech dynamics.

Finn and Montgomery [21] investigated optical recognition of English consonants in a vowel-consonant-vowel (VCV) context. Twelve reflective dots were placed around the talker's mouth, and fourteen distance measurements were manually derived from the dot positions in each frame recorded at the rate of thirty per second. The resulting sequence of measurements was matched against training sequences using a weighted Euclidean distance metric.

Silsbee [62] developed a visual processor which used a modified form of *vector quantization*. Each mouth image of a sequence was mapped into the codevector which minimized a "distance" measure. The distance was computed by first finding a best alignment between the codevector and the image to be classified, then calculating the total absolute pixel-by-pixel difference between the two. However, this is not a true distance in the mathematical sense. The triangle inequality, for example, does not necessarily hold for this quantity. One of the consequences of this property was that standard vector quantization training techniques would not have been appropriate to generate the codevectors automatically. So, the codevectors were chosen by hand from the training data. The last phase of Silsbee's lipreading system was based on

hidden Markov models. The major drawback of this system was its inherent speaker dependence due to the fact that it was based on a direct match between images rather than higher level representations.

Many of the systems described here were not fully automatic. Any practical system should avoid human intervention as much as possible. Putting a reflective material on the user's face, and extracting visual features from images by hand are both clear violations of this principle.

A great deal of attention should be directed toward time handling. In many past systems, time duration was fixed and the individual frames in speech events were concatenated and the result was simply viewed as static patterns. However, lipreading is far more complex than static pattern recognition due to the fact that time plays a crucial role in speech realization. If time and space are treated equally, the operation will not be accurately represented. The dynamic nature of change in patterns with respect to time, which is the main characterization of lipreading, should be heavily emphasized.

## 2.5 Neural Networks

An artificial neural network is a collection of parallel processors connected together in the form of a directed graph, organized such that the network structure lends itself to the problem being considered [20]. Historically, much of the inspiration for the field came from the desire to produce artificial systems capable of intelligent computations

similar to those performed by the human brain.

Artificial neural networks have a great potential for parallelism, since the computations of the components are largely independent of each other. Besides the high computation rates provided by the massive parallelism, neural networks can provide a greater degree of robustness than do traditional sequential computers. One of the most attractive features of neural networks is generalization. This enables a model to function competently throughout the pattern space, even though it has learned from observing only a limited body of examples.

### 2.5.1 Multi-Layer Feed-Forward Neural Networks

A multi-layer feed-forward network can be viewed as a structure of several layers on top of each other. At the lowest level, there is an input layer. Then, there may be one or more hidden layers and, at the highest level, an output layer. The only connection allowed is the feed-forward connection from one layer to the layer immediately on top of it. This kind of architecture is especially useful for static classification tasks since it has the capability of approximating not only any continuous map arbitrarily closely, but also the derivatives of such a map.

The basic operation of each hidden or output node is to map the weighted sum of outputs from the previous layer, according to an activation function such as the logistic or Gaussian function. The importance of activation functions is that they introduce nonlinearity into the network, without which the network would not be any more powerful than a plain perceptron (linear classifier).

Typically, the *backpropagation* algorithm [58] (generalized $\delta$-rule) is used to train feed-forward networks. According to this algorithm, the network is initialized with small random weights, then all training data is presented repeatedly to the network. Weights are adjusted after every trial, in order to minimize a function of the error between the actual output produced by the network and a desired output.

One of the main problems concerning feed-forward networks trained by backpropagation is the slow convergence during training. Another problem is the lack of a solid theory to guide the user in determining the size of the network (number of hidden layers and number of nodes in each hidden layer) for a specific application.

### 2.5.2 The Hopfield Network

In this architecture, the nodes are organized as a fully-connected layer where every node receives stimulus from all others. The weight matrix is symmetric, meaning that the weights on the connections between two nodes are equal in both directions. The nodes also receive an external input. The values of the nodes at any given time define the state of the network and this state changes until a stable configuration is reached. The current state is calculated from the previous one asynchronously. That is, a node is picked randomly and its value is updated.

The convergence of the neuronal state of the Hopfield model to its stable states is based on the existence of an energy function (Liapunov function) which directs the flow in state space. Such a function depends on the current state as well as the weight matrix. To guarantee convergence, the weights must be designed such

that any update in the network's state will decrease the energy or at least keep it unchanged. In cases where the network is supposed to act as a content-addressable memory, the weight matrix is calculated by taking the outer product of each vector to be stored in the network with itself. Then, all the resulting outer products are superimposed on one other.

The original Hopfield model was binary [28] but has been extended to a continuous model by incorporating some results from neurobiology [29]. The continuous Hopfield model can be applied to optimization problems which are NP-complete [30]. In such cases, a suitable representation for the problem that corresponds to a Hopfield network should be found. Then, the network's energy function is designed in a way which reflects the constraints of the optimization problem so that the network stabilizes on a class of good solutions depending on its initial configuration.

In practice, the Hopfield network has several limitations. The associative memory has a limited capacity, uneven recall ability and recall of spurious states. For optimization problems, the approach tends to work on examples from a limited domain. Nevertheless, the Hopfield model provides an excellent demonstration of how practical problems that are tremendously difficult can be attacked by neural networks. In particular, it can be used to model some temporal phenomena. Tank and Hopfield [65] discuss tasks similar to those of recognition of words in a continuous stream of speech.

### 2.5.3 The Time Delay Neural Network (TDNN)

This architecture is a modification of the standard multi-layer feed-forward network developed to deal with patterns that are presented in parts over a period of time [67] [37]. The basic unit in the TDNN is modified by introducing time delays. The inputs to such a unit are multiplied by several sets of weights, one for each delayed input and one for the undelayed input. To train the TDNN, the backpropagation procedure is applied to patterns that are stepped through time. Each collection of TDNN units is duplicated for each one frame shift in time. The weights of the corresponding connections in the time shifted copies are constrained to be the same. This way, the network is forced to apply the same set of feature detectors to every slice of the input, which makes the abstractions learned by the network invariant under translation in time. However, the TDNN architecture is not capable of modeling words that consist of multiple phonemes.

A *multi-state time delay neural network* (MS-TDNN) [27] [24] has been proposed to extend the TDNN model to a word-level classifier. The MS-TDNN incorporates dynamic programming into its training, so that the embedded time alignment allows training with word-level external supervision. Another interesting extension of the TDNN is the *Meta-Pi* network [25], which has been designed to improve the TDNN's performance in the context of multi-speaker phoneme recognition. The Meta-Pi architecture comprises a number of TDNNs trained independently on particular speakers.

When the speech of an unknown speaker is presented, the Meta-Pi network computes its global output using a combination of the outputs of the individual sub-networks.

### 2.5.4 Recurrent Neural Networks

A recurrent neural network contains at least one unit with the special property that its current input depends on the unit's output at an earlier time. This property allows the network to keep information about past inputs for an amount of time that is not fixed *a priori*, but rather depends on its weights and the input [3]. Recurrent networks have not been used as extensively as feed-forward networks because they seem more difficult to analyze and train optimally. Nevertheless, they have important capabilities not found in feed-forward networks, including attractor dynamics and the ability to deal with temporal behavior through their own natural operation.

Several algorithms have been proposed for training recurrent networks. In the *backpropagation-through-time* algorithm [58] [68] [69], the recurrent network to be trained is unfolded into a multi-layer feed-forward network that grows by one layer on each time instance. Then, the backpropagation procedure is applied in its usual form except that the corresponding weights at each layer (or time instance) are constrained to be equal. The advantage of this algorithm is its generality in dealing with recurrent networks of any form. A major problem, however, is its growing memory requirements when given an arbitrarily long training sequence.

Another algorithm [71], called *forward propagation*, has been derived to train unconstrained neural networks in a *temporal supervised learning* task, which means

that certain of the units' output values are to match specified target values at specified times. The algorithm avoids storing the complete sequence of network activations by computing recursively and keeping in memory, during a regular forward pass, partial derivatives which indicate how each weight of the network influences each unit's activation. The strong point of this method is that it can be applied in an on-line fashion since all the computations involved can be carried out forward in time. However, it is computationally expensive. A variant of this method has been experimented with, by Williams and Zipser [71], which incorporates a technique called *teacher-forced learning*. According to this technique, the actual output of a unit is replaced by the teacher signal whenever such a value exists. Williams and Zipser [71] report that teacher-forced learning reduced radically training time for their recurrent networks.

Other training algorithms have been described [52] [3] [45] [46] [66], some of which are constrained forms of the backpropagation-through-time or forward propagation algorithms. In particular, the *recurrent backpropagation* algorithm [52] is a special case of the backpropagation-through-time when the network's input is held constant over time and the network is assumed to relax on a stable *fixpoint* [50]. The *back-propagation for sequences* (BPS) algorithm [3], also called *focused backpropagation* [45], is somewhat related to the forward propagation when the recurrent architecture is constrained to have units with a single feedback to themselves and incoming connections from the input layer.

The methods mentioned above are based on computing the gradient of an error function with respect to the weights of the network. In a recent study, Bengio *et al.* [5] conclude that training recurrent networks with such methods becomes increasingly inefficient when the temporal span of the dependencies to be learned increases.

Obviously, the existing algorithms to directly train recurrent networks suffer mainly from two problems. First, the computational complexity associated with them is usually much higher than that of feed-forward training algorithms. Second, there has not been sufficient evidence, from a theoretical point of view, that a fully recurrent network benefits from any of the recurrent algorithms. It is possible that the recurrent methods make the network settle on sub-optimal solutions that take into account short-term rather than long-term dependencies as pointed out in [4]. Olurotimi [48] presents a general framework for training recurrent networks which avoids both problems to a great extent. His framework, emphasizes the importance of retrieving the state variables of the system being modeled. If these state variables are not retrievable from the output observations, there is simply not enough information to model the system by any technique. On the other hand, if a reasonable state representation can be constructed, the weights of a fully recurrent network can be learned using an exact transformation that reveals an embedded feed-forward structure in the recurrent architecture.

# Chapter 3

# Locating the Mouth

In this chapter, a method for automatically locating a speaker's mouth is presented. The method consists of three main steps: preprocessing, locating the eyes, and locating the mouth. The input to the system is a grey-level image of a person's face. The computations involved do not aim to find the exact pixels pertaining to the mouth. Instead, a rectangular region, containing the mouth and possibly part of the nose and chin, is found.

## 3.1 Preprocessing: Edge Detection and Thresholding

Preprocessing is employed to convert the grey-level image to an edge representation. When compared to raw grey levels, edges are far less sensitive to lighting; also, they convey valuable information about boundaries between different regions.

The required task is specific to face processing, and a special-purpose detector that can be implemented efficiently is used. The following operator, basically a Laplacian operator, has been reported, in the literature [33], to work successfully in

line extraction from human faces.

| | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| -4 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| -3 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| -2 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| -1 | 1 | 1 | 1 | -4 | -4 | -4 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | -4 | -4 | -4 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | -4 | -4 | -4 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

mask $m(k,l)$, $-4 \leq k,l \leq 4$

The original grey-level image $g(i,j)$ is convolved with the above mask $m(k,l)$, $-4 \leq k,l \leq 4$, and an edge image $e(i,j)$ is produced, where each pixel in the new representation is calculated as

$$e(i,j) = \sum_{k=-4}^{4} \sum_{l=-4}^{4} g(i+l, j+k) m(k,l) \tag{3.1}$$

The effectiveness of the operator is illustrated by Figure 3.1. The binary image it produced from the original identifies the eyes well, and it seems to not be particularly sensitive to facial hair affecting outlining of the mouth segment. This operator has many advantages [33]:

1. It combines differential operation with averaging in a single step.

2. It highlights basic features of the human face such as the head outline, eyes and mouth.

Figure 3.1: Original image. and its binary edge image produced using the Laplacian operator of Equation (3.1).

3. It eliminates most irrelevant details and noise.

The purpose is to produce a binary image on which a search for the location of the eyes is performed. The strategy pursued for determining the threshold value relies on experimentation. Experiments have shown that a value of 200 for the threshold produces a binary edge image $b(i.j)$ that successfully suits the method of locating the eyes.

## 3.2 Locating the Eyes

Although not relevant to speech processing, the eyes are located so that they can be used as a reference point to locate the mouth. This process has been somewhat problematic. Kanade [33] was able to locate facial features through his pioneering work in the field. The fundamental technique used in his system was the integral projection technique. By placing a slit, be it horizontal or vertical, and analyzing the distribution of pixels along either direction, Kanade was able to locate the top of

the head, then the sides of the face. This was followed by locating the nose, mouth, chin, chin contour, cheeks, and finally the eyes. In my approach, however, there is no need to locate facial features other than the eyes.

The method used here is derived from an interesting property of the binary edge image $b(i, j)$, $0 \leq i \leq N - 1$, $0 \leq j \leq M - 1$, where $N$ is the number of rows and $M$ is the number of columns in the input image. This property allows for the simplification of the method used by Kanade. Instead of analyzing the distribution of pixels in a slit, a simple computation that maps the slit to a quantity relevant to locating the eyes is performed.

Let $h$ be a value approximately equal to the height of the eyes and the eyebrows. This value can be determined using *a priori* knowledge of the speaker's head size. The approach taken here is to fix an index representing the ratio between the height of the eyes region and the height of the head, at a value that has proven to give good approximations when applied to a variety of people. If the height of the head in the input is expected to be highly variable due to a wide range of allowed distances between the user and the camera, $h$ can be calculated by multiplying the fixed index by the height of the head. Alternatively, if the expected variation in the user's distance from the camera lies within a reasonable range, $h$ can be assigned a fixed value.

Define $\text{edgeness}_h(r)$ as the number of foreground pixels in a horizontal slit with width $M$ and height $h$ starting at row $r$. Since $b(i, j)$ is a binary edge image,

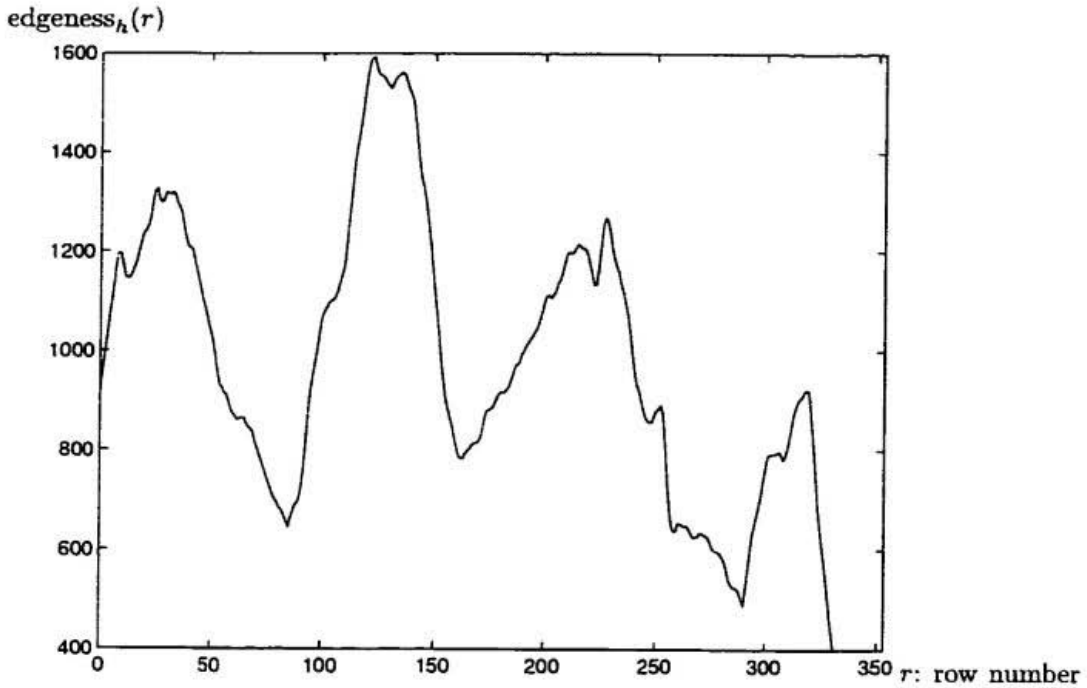


Figure 3.2: Histogram used for locating the eyes. The maximum peak occurs at the location of the eyes.

edgeness$_h(r)$ can be calculated as follows:

$$\text{edgeness}_h(r) = \sum_{i=r}^{r-h-1} \sum_{j=0}^{M-1} b(i,j)$$

Now, consider moving the slit down from the top of the image.

It is observed that edgeness$_h(r)$ reaches maximum values when the eyes and the eyebrows are contained in the slit starting at $r$, as Figure 3.2 shows. This is so because the eyes region is associated with a strong variation in intensity values, making the slit of eyes distinguished from the slits of other portions. Thus, a possible approach for identifying the eyes could be accomplished by calculating edgeness$_h(r)$ for each row $r$ in the image, then searching for the maximum value in the resulting distribution.

However, the computational requirements for such an approach are rather high.

A simplification of the process can be derived from another property. The starting row of the optimal slit lies within a range of rows corresponding to slits that are also distinguished (i.e., they map to globally high values of edgeness$_h$), even though they cover the eyes partially. This suggests that a subsampling technique results in an approximated histogram, the maximum value in which lies within a sufficiently-small neighborhood of the original maximum. Therefore, the original histogram is subsampled at $0, h/2, h, 3h/2, \cdots$. That is, two consecutive samples overlap in $h/2$.

```
o ← h/2; max ← 0; s ← 0; k ← 0
while k < N/o
    sum(k) ← 0
    for i = o · k to o · k + h
        for j = 0 to M
            sum(k) ← sum(k) + b(i, j)
        end for
    end for
    if sum(k) > max then
        max ← sum(k)
        s ← k
    end if
    k ← k + 1
end while
```

Algorithm to locate the eyes

Note that the new histogram always contains a slit covering at least 75% of the eyes region, and not 50% as the overlapping factor might wrongly indicate. To show this, assume the contrary, that is, assume that the slit starting at row $l$ is the sample with maximum coverage of the eyes region, covering $\alpha < 0.75$ which corresponds to a lower part of the eyes region (a similar argument applies in the case of an upper

part). If $\alpha < 0.25$, the slit starting at $l - h$, which is also sampled, covers

$$l - (l - (1 - \alpha)h) = (1 - \alpha)h$$

which is obviously more than 75% of the upper part. If $0.25 \leq \alpha < 0.5$, the slit starting at row $l - h/2$ covers

$$l + \alpha h - (l - h/2) = (\alpha + 0.5)h$$

which is also more than 75% of the lower part. Finally, if $0.5 \leq \alpha < 0.75$, then the coverage by the slit starting at $l - h/2$ is

$$l + h/2 - (l - (1 - \alpha)h) = (1.5 - \alpha)h$$

which is at least 75% as well (upper part). In all three cases, a slit which covers at least 75% of the target region was found, contradicting the assumption that the maximum coverage is less than 75%.

The computational reduction achieved by subsampling is by $2/h$. For a typical value of $h = 30$, it requires only 6.67% of the computation that the exhaustive approach would take.

## 3.3 Determining the Region of Interest (ROI)

The organization of the human face, in terms of the basic components, is fixed. Moreover, the relative distances between these components follow a certain pattern. Using this information, and given the approximate location of the eyes obtained in the previous step, the following can be estimated:

- Dimensions of the ROI:

  The size of the region of interest is made larger than the expected size in order to account for personal variations. that may cause some deviation from the expected measurements; and the approximation made in locating the eyes due to subsampling. Consequently. the height and width of the ROI are both set to $4h$.

- Position of the ROI:

  The left-top corner of the region of interest is positioned at

  $$(r + 3h/2. M/2 - 2h)$$

  where $r$ is the starting row of the slit containing the eyes.

Note that both estimates are expressed in terms of the height of eyes region. making the system invariant to scaling. The expressions have been selected as such after experiments had shown their effectiveness in bounding the mouth entirely even when it is wide open.

By determining the ROI. the focus of attention becomes restricted to a much smaller region than the whole input image. in which more elaborate techniques can be employed to seek accurate measurements for the mouth.

This chapter has addressed automatic location of a speaker's mouth. In the next chapter we address the problem of extracting features of the mouth.

# Chapter 4

# Extracting Mouth Features

In this chapter, methods for extracting particular mouth features are described. The features include the center, width and height of the lips. The search space is restricted to the region of interest extracted in the previous chapter. Figure 4.1 shows the main mouth characteristics that the algorithms presented in this chapter estimate.

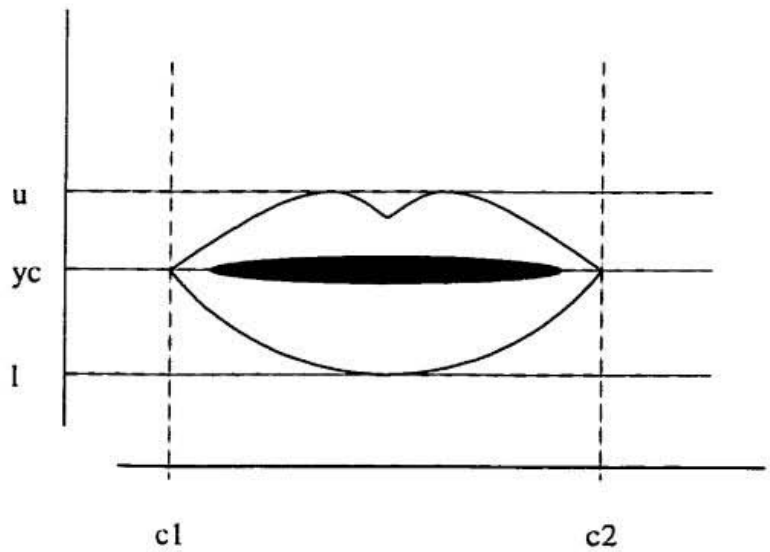


Figure 4.1: Mouth measurements.

## 4.1 Locating the Central Row and Corners of the Mouth

The central row of the mouth, i.e., the row where the upper and lower lips meet, has a strong edge presence. Accordingly, the sum of edge values along this row, within the ROI, is expected to have a maximum value. The method to locate the central row of the mouth uses the edge image $e(i, j)$ obtained by applying the Laplacian operator described in the previous chapter. An example is shown in Figure 4.2.

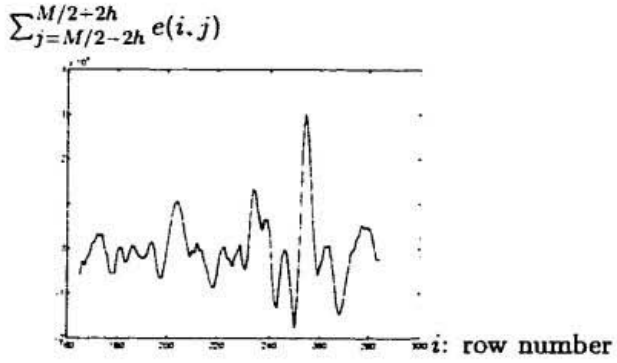


Figure 4.2: Histogram used for locating the central row of the mouth. The maximum peak corresponds to the central row.

Locating the left and right corners of the mouth enables the estimation of two important features: the central column and the width of the mouth. In this section, a method for obtaining initial estimates of the corners is described. The method assumes that the central row of the mouth has been appropriately determined.

First, the distribution of the absolute vertical gradient at each column in the central row of the mouth is analyzed. Figure 4.4 is an example of this distribution.

Let $y_c$ denote the central row of the mouth. The vertical gradient of the pixel at column $j$, row $y_c$, denoted gradient($j$), is obtained by applying the following mask, which emphasizes horizontal edges, to pixel $(y_c, j)$.

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

The corners occur near two peaks in the corresponding histogram (see Figure 4.4). These peaks are distinguished by the following characteristics:

- Define left-height($j$), the left height at column $j$, as

$$\text{left-height}(j) = |\text{gradient}(j)| - |\text{gradient}(v_1)|$$

where $v_1$ is the valley immediately preceding $j$. Let $p_1$ be such that left-height($p_1$) is maximum. The left corner $c_1$ is set at the minimum value $x$, $v_1 < x < p_1$, satisfying

$$|\text{gradient}(x)| \geq \frac{|\text{gradient}(p_1)| + |\text{gradient}(v_1)|}{2} \tag{4.1}$$

- Define right-height($j$), the right height at column $j$, as

$$\text{right-height}(j) = |\text{gradient}(j)| - |\text{gradient}(v_2)|$$

where $v_2$ is the valley immediately following $j$. Let $p_2$ be such that right-height($p_2$) is maximum. The right corner $c_2$ is set at the maximum value $y$, $p_2 < y < v_2$, satisfying

$$|\text{gradient}(y)| \geq \frac{|\text{gradient}(p_2)| + |\text{gradient}(v_2)|}{2} \tag{4.2}$$

Figure 4.3 shows a peak with its right and left heights.

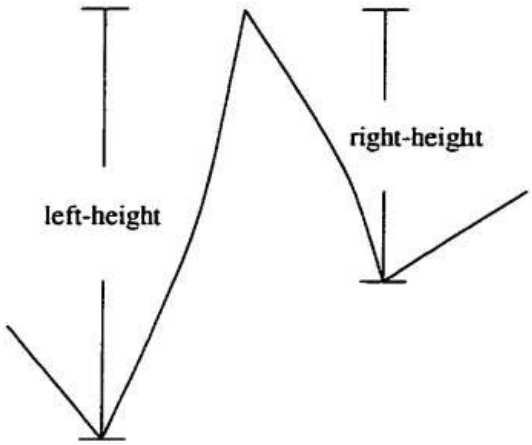


Figure 4.3: Left height and right height of a peak.

Now, initial estimates for the central column of the mouth $x_c$ and its width $w$ can be determined by

$$x_c = \frac{c_1 + c_2}{2}$$

$$w = c_2 - c_1$$

## 4.2 Locating the Upper and Lower Lips

The rows where the external borders of the upper and lower lips intersect with the vertical axis, denoted $u$ and $l$ respectively, can be identified using the previously estimated measurements. A histogram is used to sum up each row's grey levels for columns between the corners. Only rows in a small neighborhood around the center of the mouth need to be considered. It is expected that a change in intensity will occur at the maximum extremities of the lower and upper lips and this should be seen

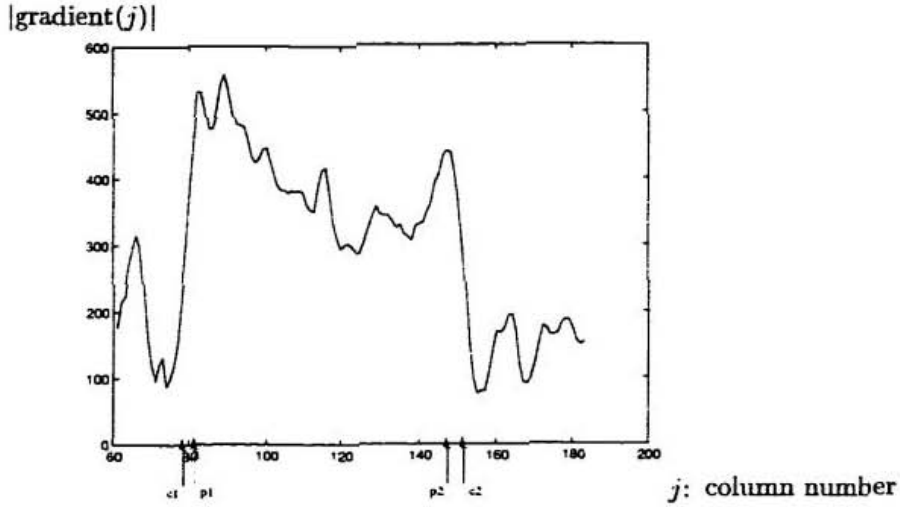


Figure 4.4: Histogram used for locating the mouth corners. First, the peaks with the maximum left and right heights ($p_1$ and $p_2$) are identified, then the corners ($c_1$ and $c_2$) are computed according to (4.1) and (4.2) (see text).

in the histogram. An example is illustrated in Figure 4.5. In fact, experiments have shown that the peak located immediately before the center of the mouth corresponds to the highest point on the upper lip and the peak after it corresponds to the lowest point on the lower lip. This estimate seems to be very good for the upper lip. However, there are cases where a peak occurs before the external outline of the lower lip because of a bright spot in the lower lip region. A method to correct this will be discussed in section 4.4.

## 4.3 Improving the Mouth Corners

The idea is to search for the best fit of two curves (corresponding to the external outline of the left half of the mouth) drawn from the left corner to the central column. The search is implemented by moving the left corner to the left until a local optimum

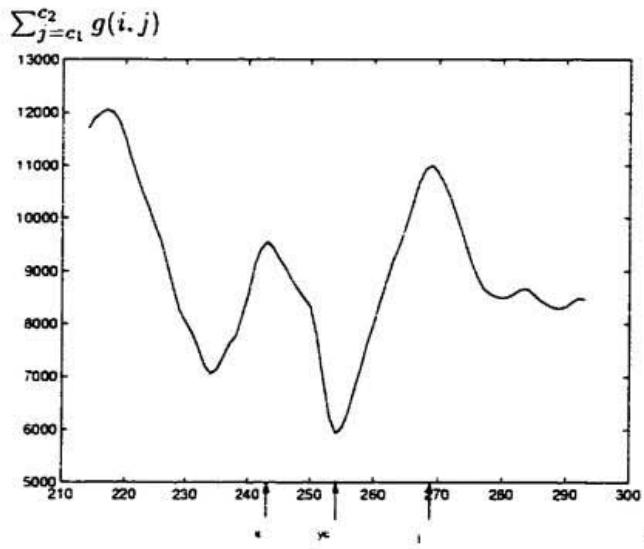


Figure 4.5: Histogram used for locating the upper and lower lips. A search for the two peaks surrounding the central row is performed.

is found for the function left-fit$(x_c, y_c, l, u, c_1)$. This function gives a measure of fitness between the left curves and the edge image, and is defined as follows:

$$\begin{aligned}\text{left-fit}(x_c, y_c, l, u, c_1) &= \sum_{x=c_1-x_c}^{0} \text{vertgrad}(y_c + y_1(x), x_c + x) \\ &\quad - \sum_{x=c_1-x_c}^{0} \text{vertgrad}(y_c + y_2(x), x_c + x),\end{aligned}$$

where $y_1(x)$ and $y_2(x)$ are the quartics given by

$$y_1(x) = (l - y_c)(1 - \frac{x^2}{(x_c - c_1)^2}) - 4(\frac{x^4}{(x_c - c_1)^4} - \frac{x^2}{(x_c - c_1)^2})$$

$$y_1(x) = (u - y_c)(1 - \frac{x^2}{(x_c - c_1)^2}) - 6(\frac{x^4}{(x_c - c_1)^4} - \frac{x^2}{(x_c - c_1)^2})$$

A similar definition applies for the right corner.

$$\begin{aligned}\text{right-fit}(x_c, y_c, l, u, c_2) &= \sum_{x=0}^{c_2-x_c} \text{vertgrad}(y_c + y_1(x), x_c + x) \\ &\quad - \sum_{x=0}^{c_2-x_c} \text{vertgrad}(y_c + y_2(x), x_c + x)\end{aligned}$$

In the definitions above, the summations have opposite signs. This is justified by the following. When applied to grey levels, vertgrad$(i, j)$ (which is the result of applying the same mask mentioned in section 4.1 to pixel $(i, j)$) emphasizes two kinds of edges:

1. Negative edges, the grey levels above which are lower than those below them. The edges on the external outline of the lower lip are of this kind.

2. Positive edges, the grey levels above which are higher than those below them. The edges on the external outline of the upper lip are of this kind.

A lower value for left-fit or right-fit means a better match between the curves and the lips.

```
while left-fit(x_c, y_c, l, u, c_1) ≥ left-fit(x_c, y_c, l, u, c_1 − 1)
    c_1 ← c_1 − 1
end while
while right-fit(x_c, y_c, l, u, c_2) ≥ right-fit(x_c, y_c, l, u, c_2 + 1)
    c_2 ← c_2 + 1
end while
```

Algorithm to improve the upper and lower lips

## 4.4 Improving the Lower Lip

As mentioned earlier, the estimate for the lower lip might not be accurate. The method used to correct this resembles the one applied to the mouth corners, but here the search is implemented by moving the lower lip down while keeping all other parameters fixed. Now, suppose that the initial estimate is accurate, then this technique will probably identify the lower lip in the wrong place. Thus, to ensure that

good initial estimates do not get corrupted, the change is not made if it goes beyond certain ranges controlled by the upper lip.

```
while left-fit(x_c, y_c, l, u, c_1) + right-fit(x_c, y_c, l, u, c_2) <
    left-fit(x_c, y_c, l + 1, u, c_1) + right-fit(x_c, y_c, l + 1, u, c_2)
        l ← l + 1
end while
while left-fit(x_c, y_c, l, u, c_1) + right-fit(x_c, y_c, l, u, c_2) ≥
    left-fit(x_c, y_c, l + 1, u, c_1) + right-fit(x_c, y_c, l + 1, u, c_2)
        l ← l + 1
end while
```

Algorithm to improve the lower lip

## 4.5 Experimentation

In this section, results of the application of the various algorithms described previously are presented. The estimated mouth measurements are compared to the actual measurements in order to qualify the accuracy of the methods. Table 4.1 contains the error of the estimated measurements when applying the algorithms to 29 facial image samples. The absolute error is calculated as the absolute difference in pixels between the estimated measurement and the actual measurement. Each row in the table summarizes the results of an individual sample. For example, the first row indicates that when the algorithms were applied to the first image sample, the estimated central row and column of the mouth and the lower lip extremity were accurate (error was 0); the estimated upper lip extremity was 2 pixels away from the actual point; and each of the estimated corners had a difference of 1 pixel from the exact value. As can be seen from Table 4.1, the accuracy of the methods is very good. In particular, the

column labeled "$y_c$" shows that the estimated central row has been accurate at all samples. Furthermore, 93.7% (163/174) of the estimated measurements have been within a range of 4 pixels from the exact ones.

Table 4.1: Error of the estimated mouth measurements obtained by applying our algorithms to 29 facial image samples

| Sample | Absolute Error (in pixels) | | | | | |
|---|---|---|---|---|---|---|
| | $y_c$ | $x_c$ | $l$ | $u$ | $c_1$ | $c_2$ |
| 1 | 0 | 0 | 0 | 2 | 1 | 1 |
| 2 | 0 | 0 | 2 | 3 | 4 | 4 |
| 3 | 0 | 1 | 1 | 2 | 3 | 0 |
| 4 | 1 | 0 | 2 | 2 | 1 | 1 |
| 5 | 0 | 2 | 0 | 2 | 8 | 4 |
| 6 | 0 | 2 | 1 | 2 | 7 | 2 |
| 7 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 0 | 2 | 7 | 2 | 6 | 2 |
| 9 | 0 | 2 | 2 | 1 | 1 | 2 |
| 10 | 0 | 0 | 2 | 1 | 3 | 4 |
| 11 | 0 | 0 | 2 | 1 | 2 | 2 |
| 12 | 0 | 1 | 2 | 0 | 1 | 0 |
| 13 | 0 | 2 | 2 | 1 | 3 | 1 |
| 14 | 0 | 1 | 1 | 3 | 1 | 3 |
| 15 | 0 | 1 | 0 | 3 | 2 | 5 |
| 16 | 0 | 0 | 7 | 4 | 4 | 4 |
| 17 | 0 | 0 | 1 | 3 | 1 | 2 |
| 18 | 0 | 0 | 0 | 4 | 1 | 1 |
| 19 | 0 | 2 | 2 | 3 | 6 | 1 |
| 20 | 0 | 0 | 7 | 8 | 0 | 0 |
| 21 | 0 | 1 | 0 | 1 | 4 | 0 |
| 22 | 1 | 2 | 1 | 8 | 2 | 2 |
| 23 | 1 | 0 | 2 | 4 | 3 | 3 |
| 24 | 0 | 0 | 1 | 2 | 3 | 3 |
| 25 | 1 | 1 | 1 | 9 | 7 | 4 |
| 26 | 0 | 0 | 1 | 1 | 2 | 2 |
| 27 | 0 | 0 | 1 | 2 | 1 | 1 |
| 28 | 1 | 0 | 1 | 1 | 1 | 1 |
| 29 | 0 | 0 | 1 | 2 | 1 | 1 |

The average and maximum errors of each estimated feature are obtained by analyzing the columns of Table 4.1, and are listed in Table 4.2. This latter table helps

in ranking the estimated variables in terms of accuracy. For example, it can be seen from Table 4.2 that the central row estimate is the most accurate one. Moreover, it is evident that all estimates are quite accurate on the average (the average error is less than 3 pixels for all estimates).

Table 4.2: Average and maximum errors of each estimated mouth measurement obtained by analyzing the samples of Table 4.1

| Measurement | Average Error (in pixels) | Maximum Error (in pixels) |
|---|---|---|
| $y_c$ | 0.1724 | 1 |
| $x_c$ | 0.6896 | 2 |
| $l$ | 1.7586 | 7 |
| $u$ | 2.6896 | 9 |
| $c_1$ | 2.7241 | 8 |
| $c_2$ | 1.9655 | 5 |

This chapter has described algorithms that estimate mouth characteristics. In the next chapter, we use these characteristics to initialize a mouth deformable template which extracts the shape of the mouth and tracks its movement during speech.

# Chapter 5

# Mouth Deformable Template

## 5.1 Geometric Model

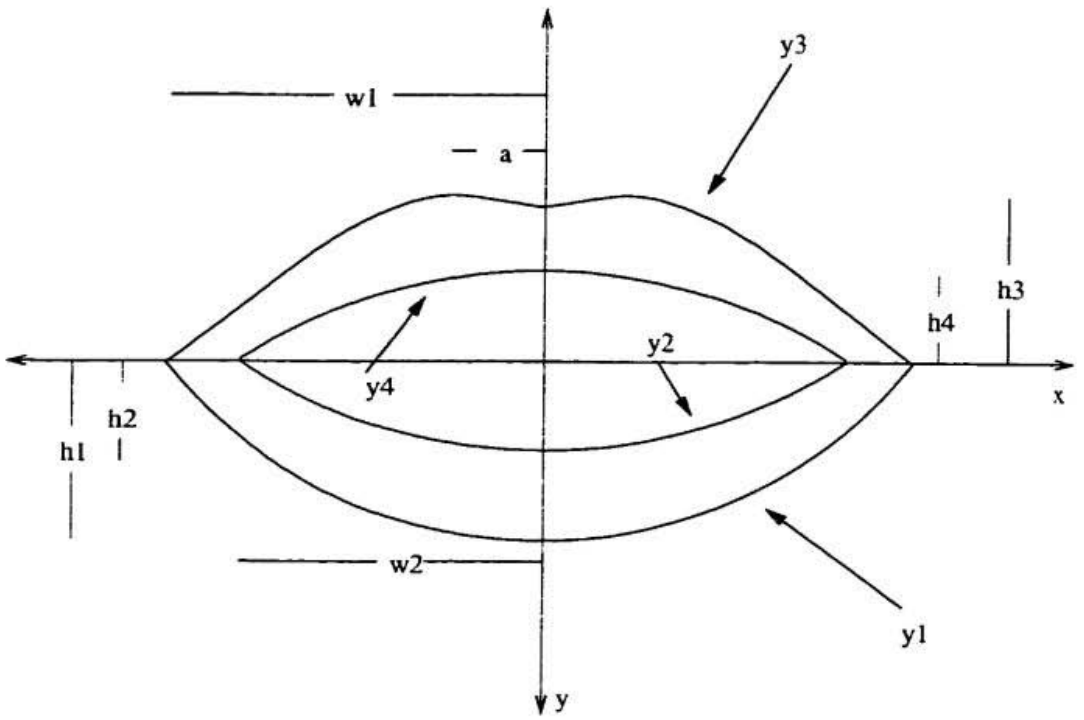


Figure 5.1: Mouth deformable template.

### 5.1.1 Geometric Primitives

The deformable template for the mouth consists of the following geometric primitives (see Figure 5.1):

1. The external outline of the lower lip is modeled as a quartic.

$$y_1(x) = h_1(1 - \frac{x^2}{w_1^2}) + 4q_1(\frac{x^4}{w_1^4} - \frac{x^2}{w_1^2}), -w_1 \leq x \leq w_1$$

2. The internal outline of the lower lip is modeled as a parabola.

$$y_2(x) = h_2(1 - \frac{x^2}{w_2^2}), -w_2 \leq x \leq w_2$$

3. The external outline of the upper lip is modeled as a quartic.

$$y_3(x) = -h_3(1 - \frac{(|x| - a)^2}{(w_1 - a)^2}) + 4q_2(\frac{(|x| - a)^4}{(w_1 - a)^4} - \frac{(|x| - a)^2}{(w_1 - a)^2}), -w_1 \leq x \leq w_1$$

4. The internal outline of the upper lip is modeled as a parabola.

$$y_4(x) = -h_4(1 - \frac{x^2}{w_2^2}), -w_2 \leq x \leq w_2$$

The points on the curves above are in reference to a coordinate system centered at (0.0). To transform the template to the image coordinate system, every point on each curve is rotated by $\theta$, which is the angle of mouth inclination in the image. then translated by $(y_c, x_c)$[1], which is the center of the mouth in the image. Figure 5.2 shows a transformed mouth template.

[1] In this thesis, a pixel in row $y$ and column $x$ is denoted $(y, x)$. To avoid any confusion, the point in the $xy$-plane that corresponds to pixel $(y, x)$ is denoted $(y, x)$ as well.

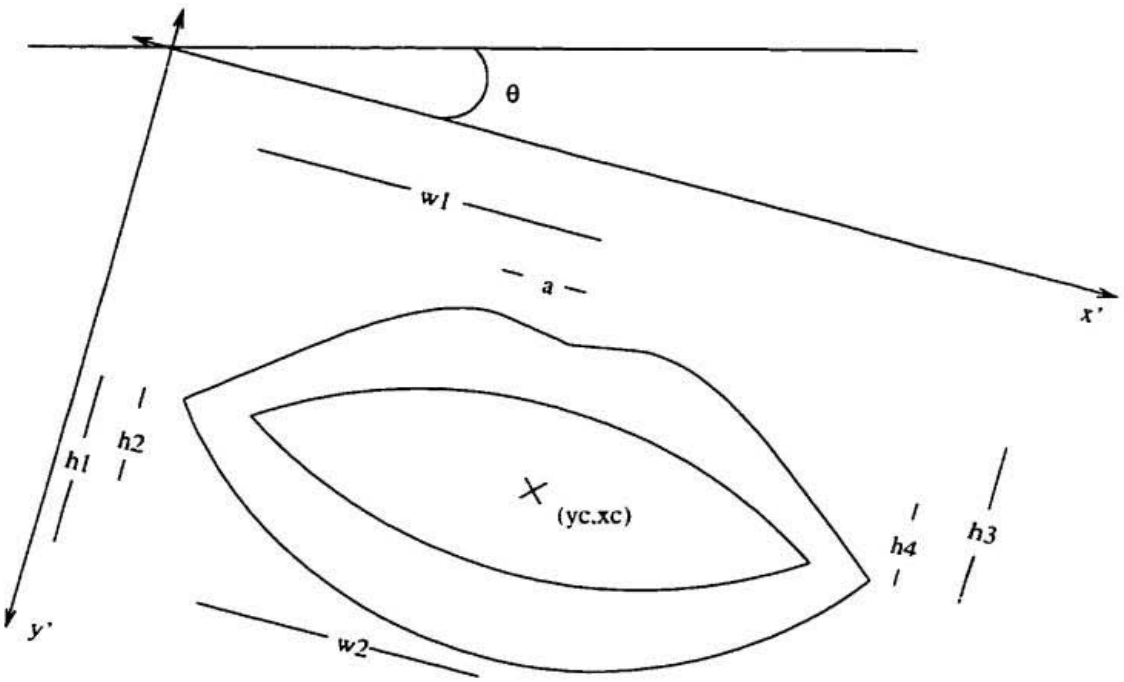


Figure 5.2: Transforming the template to the image coordinate system.

### 5.1.2 Parameters

The previous modeling scheme results in the following parameters:

1. $h_1$: height of the external outline of the lower lip.

2. $h_2$: height of the internal outline of the lower lip.

3. $h_3$: height of the external outline of the upper lip.

4. $h_4$: height of the internal outline of the upper lip.

5. $w_1$: width of the external region.

6. $a$: offset of the center of the upper quartic from the ordinate.

7. $w_2$: width of the internal region.

8. $q_1$: parameter to control how far the lower quartic deviates from a parabola.

9. $q_2$: parameter to control how far the upper quartic deviates from a parabola.

10. $x_c$: $x$-coordinate of the center of the mouth.

11. $y_c$: $y$-coordinate of the center of the external lips.

12. $\theta$: the angle of inclination.

## 5.2 Energy Function

The deformable template is associated with an energy function $E$ which gives a measure of fitness between the geometric model and the image. Consider the set

$$I = \{(y, x) : (y, x) \text{ is a lip border pixel}\}$$

The goal, at this point, is to characterize the set $I$. The deformable template approach attempts to model the set $I$ in terms of the parametrized curves contained in the geometric model. In other words, it seeks to *approximate* the set $I$ by a set $M$ such that

$$\begin{aligned} M &= C_1 \cup C_2 \cup C_3 \cup C_4, \text{ where} \\ C_1 &= \{(y', x') : y = y_1(x), -w_1 \leq x \leq w_1\}, \\ C_2 &= \{(y', x') : y = y_2(x), -w_2 \leq x \leq w_2\}, \end{aligned}$$

$$C_3 = \{(y', x') : y = y_3(x), -w_1 \leq x \leq w_1\},$$

$$\text{and } C_4 = \{(y', x') : y = y_4(x), -w_2 \leq x \leq w_2\}.$$

$(y', x')$ is the result of transforming $(y, x)$ to the image coordinate system.

$$x' = x_c + x \cdot \cos\theta - y \cdot \sin\theta$$

$$y' = y_c + x \cdot \sin\theta + y \cdot \cos\theta$$

Obviously, $M$ depends on the parameters $(h_1, h_2, h_3, h_4, w_1, a, w_2, q_1, q_2, x_c, y_c, \theta)$, and so the problem of finding the set $M$ of pixels is reduced to the problem of finding the model's parameters that would make $M$ "match" $I$.

Since the set $I$ is not known in advance, it is natural to exploit one of its known properties, which is that the pixels contained in this set have strong edge magnitudes. Consider the following quantity

$$-\frac{k_1}{|C_1|}\int_{C_1}\Phi_e(\vec{x})ds - \frac{k_2}{|C_2|}\int_{C_2}\Phi_e(\vec{x})ds - \frac{k_3}{|C_3|}\int_{C_3}\Phi_e(\vec{x})ds - \frac{k_4}{|C_4|}\int_{C_4}\Phi_e(\vec{x})ds$$

where $\Phi_e$ is an edge measure and $|C_i|$ is the length of curve $y_i$ [26]. This quantity is supposed to have a minimum value when $M$ matches $I$.

To define the edge value at pixel $(i, j)$, a $3 \times 3$ kernel $v(l, m)$, $-1 \leq l, m \leq 1$, is used.

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

$$\text{edge}(i, j) = \sum_{l=-1}^{1}\sum_{m=-1}^{1} g(i + l, j + m)v(l, m)$$

When the pixel $(i, j)$ is on the border of the lower lip, this makes edge$(i, j)$ high in magnitude, but negative in sign. On the other hand, if $(i, j)$ is on the upper lip border, edge$(i, j)$ will have a positive high value.

Given particular values for the parameters of the model, the terms of the energy function can be calculated.

$$\text{term}_1 = \frac{k_1}{w_1} \sum_{j=-w_1}^{w_1} \sum_{l=-1}^{1} \sum_{m=-1}^{1} g(i'_1 + l, j'_1 + m)v(l, m)$$

$$\text{term}_2 = \frac{k_2}{w_2} \sum_{j=-w_2}^{w_2} \sum_{l=-1}^{1} \sum_{m=-1}^{1} g(i'_2 + l, j'_2 + m)v(l, m)$$

$$\text{term}_3 = -\frac{k_3}{w_1} \sum_{j=-w_1}^{w_1} \sum_{l=-1}^{1} \sum_{m=-1}^{1} g(i'_3 + l, j'_3 + m)v(l, m)$$

$$\text{term}_4 = -\frac{k_4}{w_2} \sum_{j=-w_2}^{w_2} \sum_{l=-1}^{1} \sum_{m=-1}^{1} g(i'_4 + l, j'_4 + m)v(l, m)$$

where

$$i'_d = y_c + j \cdot \sin\theta + y_d(j) \cdot \cos\theta$$

$$j'_d = x_c + j \cdot \cos\theta - y_d(j) \cdot \sin\theta$$

$$d = 1, 2, 3, 4$$

The summations are divided by $w_1$ or $w_2$ in order to normalize the quantities. The summation in $\text{term}_3$ and $\text{term}_4$ is preceded by a minus sign because the horizontal edges on the borders of the upper lip are positive. In addition, there are penalty terms to ensure that the mouth template does not deform to illegal mouth shapes.

$$\text{term}_5 = k_5(h_1 - h_2 - k_7)^2$$

$$\text{term}_6 = k_6(h_3 - h_4 - k_8)^2$$

$\text{term}_5$ ensures that $y_1$ is always below $y_2$, and $\text{term}_6$ ensures that $y_3$ is always above $y_4$. $k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8$ are non-negative constants that can be determined empirically.

### 5.2.1 Minimizing the Energy Function

Finding the template that best matches the mouth in an image is equivalent to minimizing the energy function. Two methods have been investigated in this work.

### 5.2.2 Method 1 (Greedy Method)

```
while not stop-criteria
    for each parameter p_i
        Δp_i ←update(p_i)
    end for
    for each parameter p_i
        p_i ← p_i + Δp_i
    end for
end while
```

At each iteration, every template parameter $p_i$ is updated by one of three values: $+\text{step}_i$, $-\text{step}_i$, or 0.

If $p_i \in \{h_1, h_2, h_3, h_4, w_1, a, w_2, x_c, y_c\}$, then $\text{step}_i = 1$. This is so because these parameters are related to pixels which should have integer values. Otherwise ($p_i \in \{q_1, q_2, \theta\}$), it is preferable to assign a decimal value, such as 0.1, to $\text{step}_i$ since these variables can take real values, and it is desirable to have a value for $\text{step}_i$ that changes the shape of the template gradually during the minimization process. A value of 1 for $\text{step}_i$ when $p_i \in \{q_1, q_2, \theta\}$ could make the template change its shape drastically

in a single step. Note that $\theta$ is measured in radians rather than degrees.

$f \leftarrow E(p_1, p_2, p_3, \ldots, p_i + \text{step}_i, \ldots, p_{12}) - E(p_1, p_2, p_3, \ldots, p_i, \ldots, p_{12})$
$b \leftarrow E(p_1, p_2, p_3, \ldots, p_i, \ldots, p_{12}) - E(p_1, p_2, p_3, \ldots, p_i - \text{step}_i, \ldots, p_{12})$
if $f < 0$ then
    update $\leftarrow \text{step}_i$
else if $b > 0$ then
    update $\leftarrow -\text{step}_i$
else
    update $\leftarrow 0$
end if

In practice, the method yields acceptable results. However, theory shows that it has the potential to give erroneous results [54]. The method tries to minimize the function along the unit vectors

$$\begin{aligned} \mathbf{e}_1 &= (1.0, 0.0, \cdots, 0.0) \\ \mathbf{e}_2 &= (0.1, 0.0, \cdots, 0.0) \\ \mathbf{e}_3 &= (0.0, 1.0, \cdots, 0.0) \\ &\vdots \\ \mathbf{e}_{12} &= (0.0, 0.0, \cdots, 0.1) \end{aligned}$$

in turn. Theoretically, minimizing along a particular direction could be spoiled by the minimization along another direction.

### 5.2.3 Powell's Method

Powell's method [9] is attractive because it attempts to minimize an $n$-dimensional function without the need to explicitly compute the function's gradient. Computing the gradient requires the computation of the function's partial derivatives with

respect to each variable; a task that is not straightforward in our case because the basic terms of the energy function are not expressed analytically. Furthermore, Powell's method overcomes the potential for error in the greedy method. It consists of a mechanism for updating the set of directions (which can be the unit vectors at the beginning) as the method proceeds, attempting to devise with a non-interfering set of directions. These directions are called *conjugate directions*, and have the special property that minimization along one direction is not spoiled by subsequent minimization along another.

The basic procedure is as follows:

1. Initialize the set of directions $\mathbf{u}_i$ to the basis vectors.

$$\mathbf{u}_i = \mathbf{e}_i \qquad i = 1, \ldots, n$$

   Note that the number of such directions is equal to the number of parameters of the mouth template.

2. Repeat the following sequence of steps until the energy function stops decreasing:

   - Save the starting position as $\mathbf{P}_0$.
   - For $i = 1, \ldots, n$, move $\mathbf{P}_{i-1}$ to the minimum along direction $\mathbf{u}_i$. This involves finding a value $\lambda_{\min}$ for $\lambda$ at which the function $E(\mathbf{P}_{i-1} + \lambda \mathbf{u}_i)$ is minimum. Since the vectors $\mathbf{P}_{i-1}$ and $\mathbf{u}_i$ are fixed, $E(\mathbf{P}_{i-1} + \lambda \mathbf{u}_i)$ is

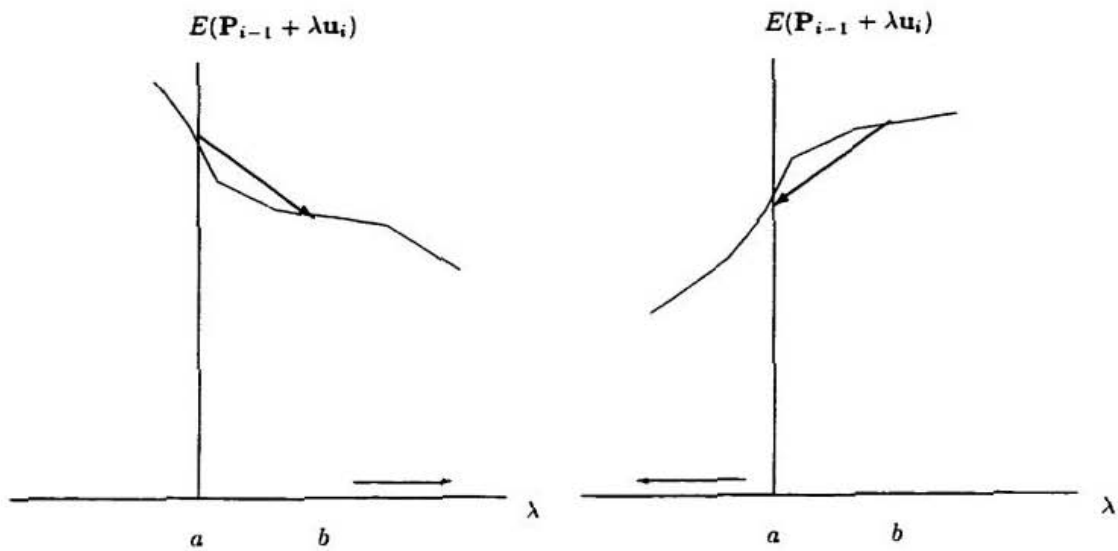


Figure 5.3: Downhill direction defined by $a$ and $b$. Search for a third point in the direction indicated by the arrow.

a function in one variable and can be minimized using a one-dimensional method. After that, set

$$\mathbf{P}_i \leftarrow \mathbf{P}_{i-1} + \lambda_{\min} \mathbf{u}_i$$

$$\Delta E_i \leftarrow E(\mathbf{P}_i) - E(\mathbf{P}_{i-1})$$

The method by which the one-dimensional function is minimized consists basically of two steps:

(a) Given initial values $a = 0$ and $b = 1$, find new points $a$, $b$, and $c$ that bracket a minimum of the function, by searching in the downhill direction defined by the function at the initial points [54] (see Figure 5.3).

By definition, $a, b, c$ are values for $\lambda$ that bracket a minimum of the

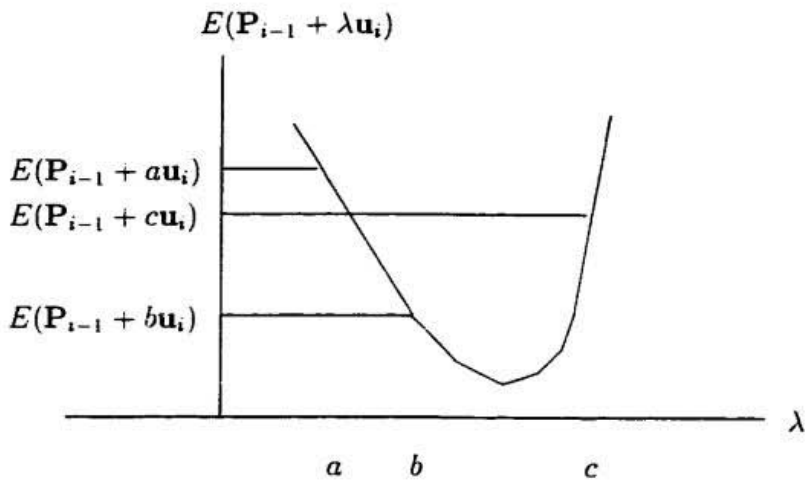


Figure 5.4: Triplet $[a, b, c]$ bracket a minimum of $E(\mathbf{P}_{i-1} + \lambda \mathbf{u}_i)$.

function $E(\mathbf{P}_{i-1} + \lambda \mathbf{u}_i)$ if $b$ is between $a$ and $c$, and $E(\mathbf{P}_{i-1} + b\mathbf{u}_i)$ is less than both $E(\mathbf{P}_{i-1} + a\mathbf{u}_i)$ and $E(\mathbf{P}_{i-1} + c\mathbf{u}_i)$ (see Figure 5.4).

(b) Use Brent's method [9] to minimize the function on the triplet $[a, b, c]$.

- For $i = 1, \ldots, n-1$, set $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$.
- Set $\mathbf{u}_n \leftarrow \mathbf{P}_n - \mathbf{P}_0$.
- Move $\mathbf{P}_n$ to the minimum along direction $\mathbf{u}_n$ and call this point $\mathbf{P}_0$. That is, find a value $\lambda_{\min}$ for $\lambda$ that minimizes the function $E(\mathbf{P}_n + \lambda \mathbf{u}_n)$. Set

$$\mathbf{P}_0 \leftarrow \mathbf{P}_n + \lambda_{\min} \mathbf{u}_n$$

The basic procedure must not be used in the form given above because it tends to produce sets of directions that become linearly dependent. When this happens, the procedure finds the minimum only over a subspace of the full $n$-dimensional case. The approach taken to fix up this problem is similar to the one reported in reference

[54]. It makes use of a heuristic scheme which tries to find a few good directions along narrow valleys instead of $n$ necessarily conjugate directions. The change to the basic procedure is the following: at each iteration, discard the old direction along which the function made its largest decrease. That is, $\mathbf{u}_i$ is discarded (substituted by the new direction $\mathbf{P}_n - \mathbf{P}_0$ which is the average direction moved after trying all $n$ possible directions) if $\Delta E_i \geq \Delta E_j$, $j = 1, 2, \ldots, n$.

## 5.3 Dividing the Minimization Process into Two Phases

The general method of deformable templates implies that all the curves making up the model are fit at the same time. However, it has been found, through experimentation, that a two-phase minimization process gives better results than those obtained by some single phase techniques. This indicates that the interaction of $w_2$ with the rest of the model does not help the template to converge. Therefore, the minimization process is divided into two phases:

1. In the first phase, the external curves are fit. The original geometric model is modified. The internal width $w_2$ is replaced by $w_1$, and the internal heights, $h_2$ and $h_4$, are replaced by $h'_2$ and $h'_4$. This simpler template has been able to capture the external borders accurately, and to give reasonable estimates for $h'_2$ and $h'_4$ (up to 3 pixels away from the actual internal heights). In this phase,

the function

$$E_1 = \text{term}_1 + \text{term}_2 + \text{term}_3 + \text{term}_4 + \text{term}_5 + \text{term}_6$$

is minimized using Powell's method.

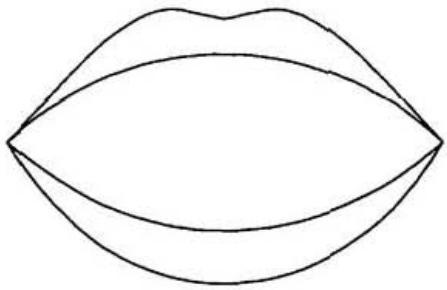

Figure 5.5: Modified mouth model for the first phase.

2. In the second phase, $h_2$, $h_4$, and $w_2$ are fine-tuned by minimizing the function

$$E_2 = \text{term}_2 + \text{term}_4 + k_9 \tan(\frac{\pi w_2}{2w_1}) + k_{10}(h_2 - h_2')^2 + k_{11}(h_4 - h_4')^2$$

$\text{term}_2$ and $\text{term}_4$ attracts the internal curves to strong horizontal edges in the image. The term $k_9 \tan(\frac{\pi w_2}{2w_1})$ encourages $w_2$ to be less than $w_1$. The last two terms ensure that $h_2$ and $h_4$ stay relatively close to $h_2'$ and $h_4'$ respectively.

## 5.4 Tracking the Lips

Given a digitized movie of a person's face, the deformable template technique is used to model the mouth shape in each image frame.

1. A region of interest containing the mouth is located in the first frame. Then, image processing techniques are applied to roughly estimate several mouth

features such as center, width and height. These estimates are used as initial values for the deformable template parameters in the first frame.

2. Having the initial values, the deformable template can be positioned initially near the mouth. The energy of the template is minimized, and this leads to an optimum match.

3. Since there is a relatively small change between two consecutive frames, the initial values for a particular frame $i$, $i \geq 2$, are set to the parameter values of frame $i-1$. This has the advantage of saving the time that would be, otherwise, needed to apply the initial operations to every single frame.

This chapter has addressed the application of deformable templates in extracting the shape of the mouth and tracking its movement during speech. In the next chapter, we address the problem of segmenting a word into visual speech units.

# Chapter 6

# Visual Speech Segmentation

## 6.1 Introduction

In this chapter, the problem of visual speech segmentation is addressed. The method described here is applied to the training data of the speech recognizer.

The problem of visual speech segmentation is formulated as follows. Let

$$\{\mathbf{v}_i\}_0^{n-1} = \mathbf{v}_0\mathbf{v}_1\cdots\mathbf{v}_{n-1}$$

be a given sequence of vectors, representing a word visually, where $\mathbf{v}_i$ is a vector of $m$ features characterizing the mouth shape of a speaker, saying a word $w$, at time $i/r$, where $r$ is the sampling rate and assuming that the speech starts at time 0. It is required to find a natural number $l < n$, and indices $k_1, k_2, \ldots, k_{l-1}$ for $\{\mathbf{v}_i\}_0^{n-1}$ such that

$$0 = k_0 < k_1 < k_2 < \cdots < k_{l-1} \leq n-1$$

$$\begin{aligned}
\mathbf{v}_0\mathbf{v}_1\cdots\mathbf{v}_{k_1-1} &\xrightarrow{T} \mathcal{V}_0\\
\mathbf{v}_{k_1}\mathbf{v}_{k_1-1}\cdots\mathbf{v}_{k_2-1} &\xrightarrow{T} \mathcal{V}_1\\
\vdots &\quad\vdots\\
\mathbf{v}_{k_{l-1}}\mathbf{v}_{k_{l-1}-1}\cdots\mathbf{v}_{n-1} &\xrightarrow{T} \mathcal{V}_{l-1}.
\end{aligned}$$

where $T$ is a transformation, and $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_{l-1}$ are visual speech units such that

$$\begin{aligned}
\mathcal{V}_0 &\text{ is distinct from } \mathcal{V}_1\\
\mathcal{V}_1 &\text{ is distinct from } \mathcal{V}_2\\
&\vdots \qquad\qquad\qquad\quad \vdots\\
\mathcal{V}_{l-2} &\text{ is distinct from } \mathcal{V}_{l-1}.
\end{aligned}$$

$l$ is the number of segments, and $k_i$, $i = 0, 1, \ldots, l-1$ is the index of the first frame in segment $i$.

The procedures presented in this chapter operate on sequences of feature vectors derived from the parameters of the mouth deformable template. Let $W$ be the mouth external width ($w_1$) in the first frame (at time 0) of the sequence. Each frame in the sequence in mapped to a vector of the following features.

1. $h_1/W$
2. $h_2/W$
3. $h_3/W$

4. $h_4/W$

5. $w_1/W$

6. $w_2/W$

7. $a/W$

The purpose of dividing the parameters by $W$ is:

1. to make the system scale invariant.

2. and to provide a suitable (normalized) representation for the neural network that will process the data in a later stage.

An example is represented graphically in Fig. 6.1. In this figure, each horizontal line represents a feature. The lengths of the lines are proportional to the values of the features. The graph has been built by applying the mouth deformable template of the previous chapter to a sequence of facial images of a person during speech (sampled at the rate of 30 frames per second), then mapping the resulting mouth template of each image into the seven features listed above. Note that each row of horizontal lines corresponds to the shape of the mouth at a certain time.

## 6.2 Maximum Distance Method

The proposed solution to the problem posed in the previous section is based on identifying dissimilarities within the sequence to be segmented. Let $\mathbf{f}, \mathbf{f}', \mathbf{f1}, \mathbf{f2}$ be

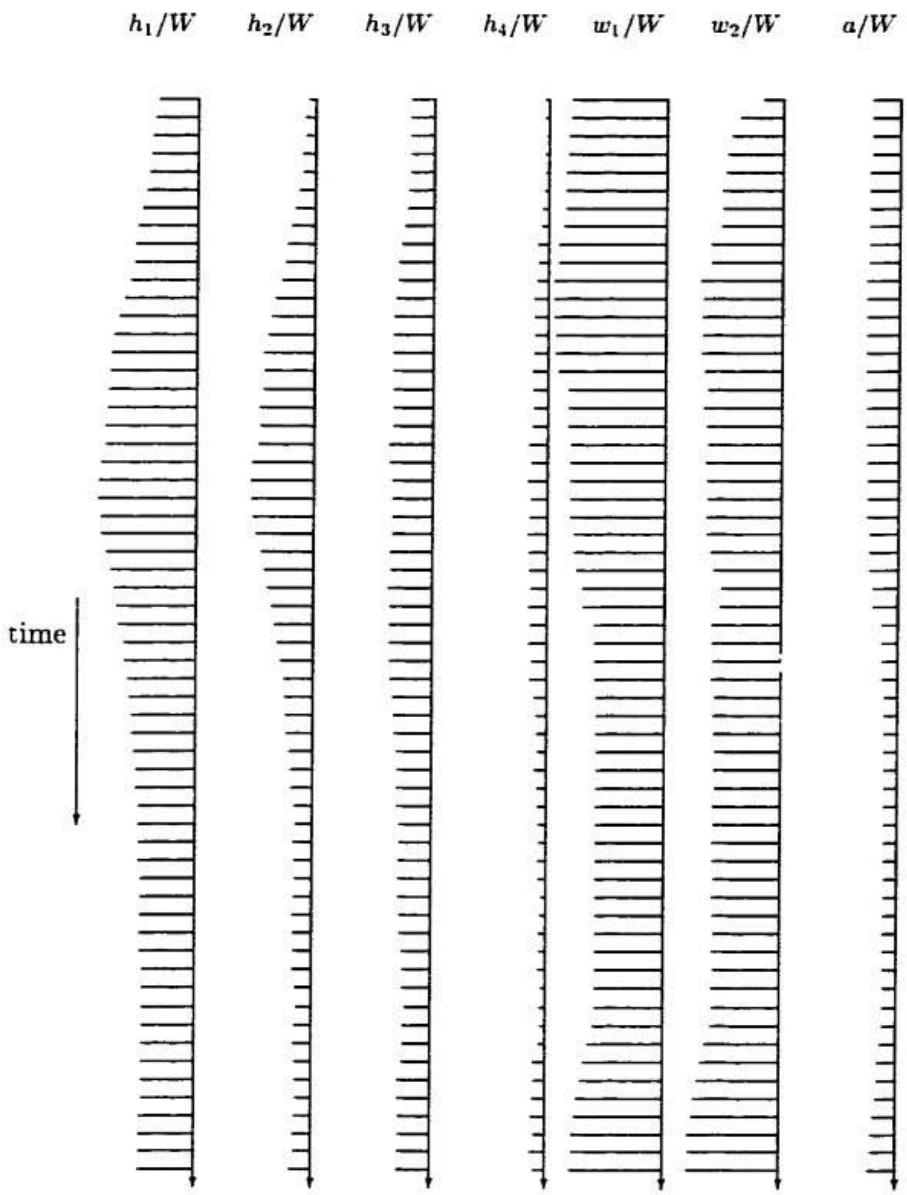


Figure 6.1: Graphical representation of a sequence of feature vectors corresponding to a person's mouth during speech.

feature vectors. Define

$$\text{dist}(\mathbf{f}, \mathbf{f}') = \sqrt{\sum_{i=0}^{m-1} (f_i - f_i')^2}$$

$$\text{dist2}(\mathbf{f}, \mathbf{f1}, \mathbf{f2}) = \min(\text{dist}(\mathbf{f}, \mathbf{f1}), \text{dist}(\mathbf{f}, \mathbf{f2}))$$

$\text{dist}(\mathbf{f}, \mathbf{f}')$ measures the distance between frames $\mathbf{f}$ and $\mathbf{f}'$. If $\mathbf{f} = \mathbf{f}'$, obviously $\text{dist}(\mathbf{f}, \mathbf{f}') = 0$. $\text{dist2}(\mathbf{f}, \mathbf{f1}, \mathbf{f2})$ measures how far frame $\mathbf{f}$ is from both frames $\mathbf{f1}$ and $\mathbf{f2}$. If $\text{dist2}(\mathbf{f}, \mathbf{f1}, \mathbf{f2}) = d$, that means that $\mathbf{f}$ has a distance of at least $d$ from $\mathbf{f1}$ and $\mathbf{f2}$.

Given a frame $\mathbf{f}$ and a set of frames $F$, it is possible to identify the most distant frame in $F$ from $\mathbf{f}$ by finding $\mathbf{f}_{\max} \in F$ according to

$$\text{dist}(\mathbf{f}, \mathbf{f}_{\max}) = \max_{\mathbf{f}' \in F} \text{dist}(\mathbf{f}, \mathbf{f}')$$

Similarly, given two frames $\mathbf{f1}$ and $\mathbf{f2}$, the most distant frame in $F$ from both $\mathbf{f1}$ and $\mathbf{f2}$ is the frame $\mathbf{f}_{\max} \in F$ such that

$$\text{dist2}(\mathbf{f}_{\max}, \mathbf{f1}, \mathbf{f2}) = \max_{\mathbf{f} \in F} \text{dist2}(\mathbf{f}, \mathbf{f1}, \mathbf{f2})$$

Upon segmenting a sequence of frames, there are two cases that have to be distinguished.

- The sequence consists of more than one segment.

  In this case, a good candidate for a segmentation point is the frame that has the maximum distance from the starting frame. The value of the maximum distance represents a *segmentation confidence* measure. The higher the value

of the maximum distance. the higher the confidence that this point is a real boundary.

- The sequence consists of only one segment.

  For this to happen, all the frames in the sequence should be "close" to each other. This situation can be identified by examining the maximum distance between the first frame and the rest of the frames. If this distance is very small, it follows that all the frames are sufficiently close to each other.

  **Claim:** In a sequence of frames $\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{s-n-1}$, suppose that $k \in V = \{s+1, s+2, \ldots, s+n-1\}$ is such that

$$\text{dist}(\mathbf{v}_s, \mathbf{v}_k) = \max_{i \in V} \text{dist}(\mathbf{v}_s, \mathbf{v}_i) = d.$$

  Then, for any $p, q \in V$

$$\text{dist}(\mathbf{v}_p, \mathbf{v}_q) \leq 2d.$$

  **Proof:** The assumption implies that

$$\text{dist}(\mathbf{v}_s, \mathbf{v}_p) \leq d$$

$$\text{dist}(\mathbf{v}_s, \mathbf{v}_q) \leq d$$

  Equivalently,

$$(v_{s,0} - v_{p,0})^2 + (v_{s,1} - v_{p,1})^2 + \cdots + (v_{s,m-1} - v_{p,m-1})^2 \leq d^2$$

$$(v_{s,0} - v_{q,0})^2 + (v_{s,1} - v_{q,1})^2 + \cdots + (v_{s,m-1} - v_{q,m-1})^2 \leq d^2$$

Adding both inequalities yields

$$\sum_{i=0}^{m-1} \left( (v_{s,i} - v_{p,i})^2 + (v_{s,i} - v_{q,i})^2 \right) \leq 2d^2 \tag{6.1}$$

$$\sum_{i=0}^{m-1} \frac{(v_{p,i} - v_{q,i})^2}{2} \leq 2d^2 \tag{6.2}$$

$$\sqrt{\sum_{i=0}^{m-1} (v_{p,i} - v_{q,i})^2} \leq 2d. \tag{6.3}$$

$$\tag{6.4}$$

That is,

$$\mathrm{dist}(\mathbf{v}_p, \mathbf{v}_q) \leq 2d \tag{6.5}$$

Inequality (6.2) follows from

$$\frac{(c-b)^2}{2} \leq (a-b)^2 + (a-c)^2$$

where $a$, $b$ and $c$ are any real numbers.□

Hence, if the segmentation confidence measure is less than a threshold $\delta$, it is guaranteed that no two frames in the sequence are at a distance greater than $2\delta$.

Consider the following algorithm that takes as input a sequence of $n$ vectors $\mathbf{v}_s \mathbf{v}_{s-1} \cdots \mathbf{v}_{s-n-1}$.

| find $k$ from the set $V = \{s+1, s+2, \ldots, s+n-1\}$ such that $\mathrm{dist}(\mathbf{v}_s, \mathbf{v}_k) = \max_{i \in V} \mathrm{dist}(\mathbf{v}_s, \mathbf{v}_i)$ |
|---|

This algorithm segments the sequence $\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{s-n-1}$ into two subsequences $h$ (internal subsequence) and $t$ such that

$$h = \mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{k-1}$$

$$t = \mathbf{v}_k\mathbf{v}_{k-1}\cdots\mathbf{v}_{n+s-1}$$

To complete the process of finding all the segments in the given sequence, the same algorithm can be applied recursively to $t$. It can also be applied to $h$. But since $\mathbf{v}_{k-1}$ and $\mathbf{v}_k$ are two consecutive visual speech frames and, thus, very close to each other, there is a high probability that applying the algorithm on $h$ will result in segmenting the subsequence at its last frame $(k-1)$. To avoid this situation, a slightly different algorithm is proposed. The algorithm takes as input an internal sequence of $n$ frames $\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{s-n-1}$ and the starting frame of the next subsequence $\mathbf{v}_{s-n}$.

find $k$ from the set $V = \{s+1, s+2, \ldots, s+n-1\}$ such that
$\text{dist2}(\mathbf{v}_k, \mathbf{v}_s, \mathbf{v}_{s-n}) = \max_{i\in V}\text{dist2}(\mathbf{v}_i, \mathbf{v}_s, \mathbf{v}_{s-n})$

Now, two recursive versions of the previous algorithms can be defined.

**Algorithm** segment($\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{s-n-1}$)
find $k$ from the set $V = \{s+1, s+2, \ldots, s+n-1\}$ such that
$\text{dist}(\mathbf{v}_s, \mathbf{v}_k) = \max_{i\in V}\text{dist}(\mathbf{v}_s, \mathbf{v}_i)$
if $\text{dist}(\mathbf{v}_s, \mathbf{v}_k) > \delta$
output $k$
segment2($\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{k-1}, \mathbf{v}_k$)
segment($\mathbf{v}_k\mathbf{v}_{k-1}\cdots\mathbf{v}_{s-n-1}$)
end if

**Algorithm** segment2($\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{s-n-1}, \mathbf{v}_{s-n}$)
find $k$ from the set $V = \{s+1, s+2, \ldots, s+n-1\}$ such that
dist2($\mathbf{v}_k, \mathbf{v}_s, \mathbf{v}_{s-n}$) = $\max_{i \in V}$ dist2($\mathbf{v}_i, \mathbf{v}_s, \mathbf{v}_{n-s}$)
if dist2($\mathbf{v}_k, \mathbf{v}_s, \mathbf{v}_{s-n}$) $> \delta$
output $k$
segment2($\mathbf{v}_s\mathbf{v}_{s-1}\cdots\mathbf{v}_{k-1}, \mathbf{v}_k$)
segment2($\mathbf{v}_k\mathbf{v}_{k-1}\cdots\mathbf{v}_{s-n-1}, \mathbf{v}_{s-n}$)
end if

Algorithm segment finds all segment boundaries with confidence greater than the threshold $\delta$ in the given sequence. segment2 does the same thing but has as input an internal subsequence. Given a sequence of frames $\mathbf{v}_0\mathbf{v}_1\cdots\mathbf{v}_{n-1}$ representing a word, all the segment boundaries ($k$'s in the problem definition) with confidence greater than the threshold $\delta$ can be found by simply calling

$$\text{segment}(\mathbf{v}_0\mathbf{v}_1\cdots\mathbf{v}_{n-1})$$

The number of segments is equal to the number of segment boundaries plus one. Figure 6.2 is a graphical representation of applying the Maximum Distance method to a word. In this figure, each feature is represented by a rectangle. The intensity of the rectangles is proportional to the value of the corresponding features. The partial results of applying the method recursively are shown at the left of the sequence (these results evolve as we go to the left direction). The final results are shown at the right of the sequence.

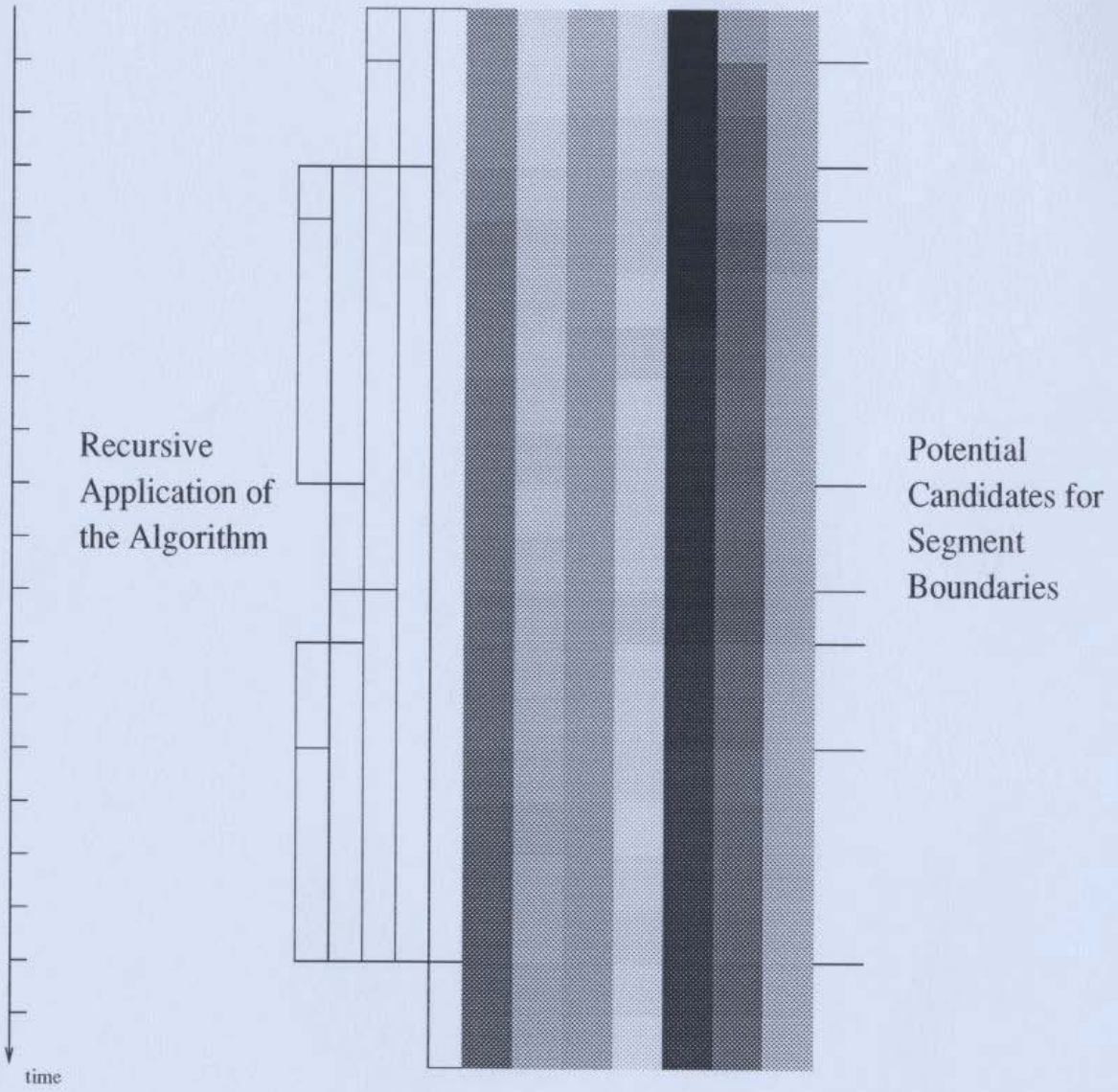


Figure 6.2: Individual segmentation by Maximum Distance method.

## 6.3 Alignment

Let $x$, $y$ and $z$ be three different samples of the same word. If these sequences are segmented individually, the result will be:

$$\underbrace{\mathbf{x}_0 \cdots}\underbrace{\mathbf{x}_{k_{x1}} \cdots}\underbrace{\mathbf{x}_{k_{x2}} \cdots} \cdots \underbrace{\mathbf{x}_{k_{xl_x}} \cdots}$$

$$\underbrace{\mathbf{y}_0 \cdots}\underbrace{\mathbf{y}_{k_{y1}} \cdots}\underbrace{\mathbf{y}_{k_{y2}} \cdots} \cdots \underbrace{\mathbf{y}_{k_{yl_y}} \cdots}$$

$$\underbrace{\mathbf{z}_0 \cdots}\underbrace{\mathbf{z}_{k_{z1}} \cdots}\underbrace{\mathbf{z}_{k_{z2}} \cdots} \cdots \underbrace{\mathbf{z}_{k_{zl_z}} \cdots}$$

The problem is that the number of segments in $x$, $y$ and $z$ might be different. Thus, alignment aims to find a segmentation for $x$, $y$ and $z$ such that the number of segments in all the sequences is equal. Moreover, and more importantly, the $i$-th segment in any of the sequences should correspond to the $i$-th segment in the others.

Given two sequences, it is possible to find such a segmentation using *dynamic time warping* (DTW) [59]. A DTW algorithm takes two sequences of frames $a = \mathbf{a}_0\mathbf{a}_1 \cdots \mathbf{a}_{n-1}$ and $b = \mathbf{b}_0\mathbf{b}_1 \cdots \mathbf{b}_{m-1}$ as input, and outputs the optimum warping path between them according to some distance measurement. The cost of the optimum warping path is also produced as an output. The algorithm measures how well the two sequences can be aligned to each other based on the distance function used.

The DTW algorithm operates by calculating a matrix $g(i, j)$, $1 \leq i \leq n$, $1 \leq j \leq m$. The interpretation of $g$ is that

$g(i, j)$ is the cost of the optimum warping path between subsequences $\mathbf{a}_0\mathbf{a}_1 \cdots \mathbf{a}_{i-1}$ and $\mathbf{b}_0\mathbf{b}_1 \cdots \mathbf{b}_{j-1}$

Therefore, the required output will be stored in $g(n, m)$. To calculate $g(i, j)$, there are two cases:

1. $i = 1$ and $j = 1$ (initial case)

$$g(1, 1) = 2\text{dist}(\mathbf{a}_0, \mathbf{b}_0) \tag{6.6}$$

2. $i \neq 1$ or $j \neq 1$

   In calculating $g(i, j)$ in general, there are three subpaths to consider:

   (a) The subpath formed by linking the optimum warping path between $\mathbf{a}_0\mathbf{a}_1 \cdots \mathbf{a}_{i-1}$ and $\mathbf{b}_0\mathbf{b}_1 \cdots \mathbf{b}_{j-2}$ to the alignment of frames $\mathbf{a}_{i-1}$ and $\mathbf{b}_{j-1}$.

   (b) The subpath formed by linking the optimum warping path between $\mathbf{a}_0\mathbf{a}_1 \cdots \mathbf{a}_{i-2}$ and $\mathbf{b}_0\mathbf{b}_1 \cdots \mathbf{b}_{j-2}$ to the alignment of frames $\mathbf{a}_{i-1}$ and $\mathbf{b}_{j-1}$.

   (c) The subpath formed by linking the optimum warping path between $\mathbf{a}_0\mathbf{a}_1 \cdots \mathbf{a}_{i-2}$ and $\mathbf{b}_0\mathbf{b}_1 \cdots \mathbf{b}_{j-1}$ to the alignment of frames $\mathbf{a}_{i-1}$ and $\mathbf{b}_{j-1}$.

   Figure 6.3 illustrates this situation. It shows the three options available at a certain point while constructing the warping path between two sequences. Clearly, the optimum cost in this case is the cost of the subpath that has the

minimum cost among the three subpaths.

$$g(i,j) = \min \begin{cases} g(i,j-1) + \text{dist}(\mathbf{a}_{i-1}, \mathbf{b}_{j-1}) \\ g(i-1,j-1) + 2\text{dist}(\mathbf{a}_{i-1}, \mathbf{b}_{j-1}) \\ g(i-1,j) + \text{dist}(\mathbf{a}_{i-1}, \mathbf{b}_{j-1}) \end{cases} \tag{6.7}$$

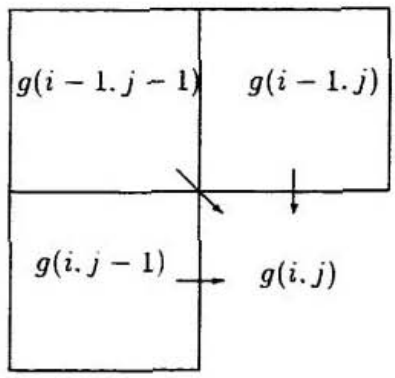


Figure 6.3: Calculating the optimum warping path

The coefficients of dist() in (6.5) and (6.6) are called the *weighting coefficients*. The specific values given to these coefficients make the algorithm symmetric. This implies that the result of aligning $a$ to $b$ will be the same as that of aligning $b$ to $a$.

When calculating the first row of $g$ ($g(1,j)$, $1 \leq j \leq m$), the elements in the 0-th row are used. Since there are no real subpaths coming from such elements that need to be considered, they should be initialized to a large value in order to prevent them from being selected as the optimum choice. A similar argument applies to the 0-th column.

$$g(0,j) = \infty, \qquad 1 \leq j \leq m$$

$$g(i,0) = \infty, \qquad 1 \leq i \leq n$$

The DTW algorithm uses a parameter $r$ called the window length. This parameter limits the optimum path to not go beyond a region around the diagonal of matrix $g$. Therefore, only the elements in this region of $g$ need to be calculated. The limited region can be expressed by

$$g(i,j), \qquad 1 \leq j \leq m \text{ and } j - r \leq i \leq j + r$$

The purpose of such a constraint is to prevent an excessive timing difference between the frames of the two sequences. In order for the constrained set of points to include $g(n, m)$, it is necessary that

$$r \geq n - m$$

is satisfied.

The optimum warping path can be obtained by recording the choice made at each calculation, then tracing the sequence of choices from the final output $g(n, m)$ back to the initial point $g(1, 1)$.

```
for j = 1 to m
    g(0, j) ← ∞
end for
for i = 1 to n
    g(i, 0) ← ∞
end for
g(1, 1) ← 2dist(a_0, b_0))
for j = 1 to m
    for i=max(1, j − r) to min(n, j + r)
        if i ≠ 1 or j ≠ 1
            g(i, j) ← min(
                g(i, j − 1) + dist(a_{i−1}, b_{j−1}),
                g(i − 1, j − 1) + 2dist(a_{i−1}, b_{j−1}),
                g(i − 1, j) + dist(a_{i−1}, b_{j−1}))
        end if
    end for
end for
cost ← g(n, m)
```

The alignment problem can make use of DTW. However, the usual DTW algorithm needs to be modified to suit this problem. In the modified version, the algorithm takes as input the segment boundaries in two sequences. The optimum warping path is used to derive a unified segmentation scheme for both sequences.

To convert the path produced by the algorithm to a unified segmentation:

1. A vector of ordered pairs, called the *correspondence vector*, is obtained by following the warping path. The first component in an ordered pair represents a frame in sequence $a$, and the second component is the corresponding frame in $b$.

2. Starting from the end of the correspondence vector, which represents the last

frames in both $a$ and $b$, the following is repeated until the beginning of the correspondence vector is reached.

(a) If the two adjacent pairs, at the current point, are different in both components; record a segmentation boundary.

(b) Decrement the current point index.

The problem, now, is how to generalize this alignment method to more than two sequences. A possible way to generalize the method could be to select a reference sequence and align it to all other sequences. But even then, this might lead to a different segmentation for the reference sequence in every alignment. To obtain a unique segmentation, the following procedure is proposed.

```
select a reference sequence x
segment x
for each sequence s ≠ x
    segment s
    align s to x
    update the segmentations of x and s according
        to the result of the previous alignment
end for
for each sequence s ≠ x
    align s to x {this time the segmentation of x is fixed}
end for
```

Now, two questions arise. First, how to select the reference sequence $x$? Second, in which order should the other sequences be aligned to $x$? In all probability, such decisions will affect the final alignments. Hence, there is a need to rank the sequences according to some criteria. The criteria selected for ranking a certain sequence is

based on how well the sequence aligns to the rest of the sequences. Recall that the DTW algorithm returns the minimum cost at which sequences $x$ and $s$ can be aligned to each other cost$(x, s)$. To get an overall cost for a particular sequence $x$, the average cost over all other sequences is taken

$$\text{rank}(x) = \frac{\sum_{s \neq x} \text{cost}(x, s)}{N - 1}$$

where $N$ is the number of sequences. Figure 6.4 shows the result of aligning 4 instances of the same word using the method described above.

## 6.4 Adaptive Segmentation

Two distance functions have been used in the segmentation method described above. First, the distance function for individual segmentation and second, the distance function for dynamic time warping. The Euclidean distance was selected for both purposes.

There is no reason why the Euclidean distance is better than a weighted distance. Using a weighted distance could not have been possible before because there was no *a priori* knowledge about the segments of a particular word. But now that there is a segmentation (initial segmentation), this knowledge can be used to derive proper weights.

This principle leads to an iterative method that starts by segmentation and alignment using the Euclidean distance, which is equivalent to a weighted distance with equal weights. At each iteration, an error is computed based on the previous seg-

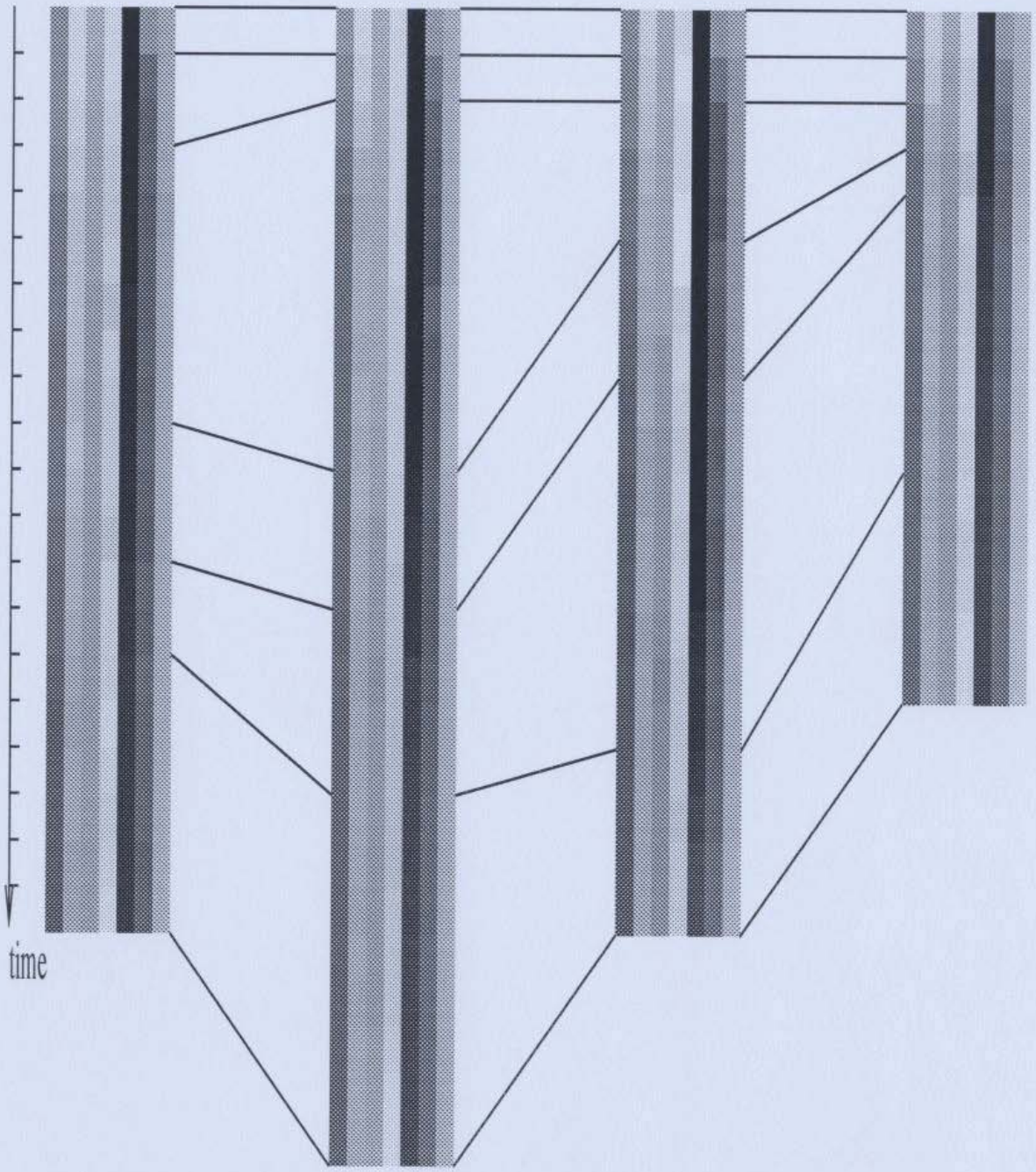


Figure 6.4: Alignment between segments in samples of the same word.

mentation. The weights are adjusted in such a way that reduces the error. Then, segmentation and alignment are repeated using the new weighted distances.

### 6.4.1 Weights for Individual Segmentation Distance

Let $\mathbf{w}$ be the set of weights of the distance function used in the individual segmentation. Two tables are formed.

- Similarity table for $\mathbf{w}$:

  Each entry is a vector computed as

  $$\mathbf{w} * (\mathbf{x} - \mathbf{y})$$

  where $\mathbf{x}$ and $\mathbf{y}$ are two frames belonging to the same segment, and $*$ denotes "multiplication" of vectors element by element. For each segment containing more than one frame, $\mathbf{x}$ is selected as the first frame, and $\mathbf{y}$ as the most distant frame from $\mathbf{x}$ in the same segment.

- Difference table for $\mathbf{w}$:

  Each entry is a vector computed as

  $$\mathbf{w} * (\mathbf{x}' - \mathbf{y}')$$

  where $\mathbf{x}'$ and $\mathbf{y}'$ are two frames belonging to adjacent segments. For each two adjacent segments, $\mathbf{x}'$ is selected as the first frame in the first segment, and $\mathbf{y}'$ as the first frame in the second segment.

The average entry $\mathbf{a} = [a_0 \quad a_1 \quad a_2 \cdots a_{m-1}]$ in the similarity table is computed, then the average element $A$ in that entry is taken

$$A = \frac{1}{m} \sum_{i=0}^{m-1} a_i$$

The error derived from the similarity table is expressed as

$$\text{error}_1 = \frac{1}{d} \sum_{i=0}^{m-1} \Gamma^2(A - a_i)$$

where

$$\Gamma(\alpha) = \begin{cases} \alpha, & \alpha < 0 \\ 0, & \text{otherwise} \end{cases}$$

and $d$ is the number of elements in $\mathbf{a}$ that are greater than $A$. The weight updates based on this error are as follows.

$$\Delta w_i = \eta \Gamma(A - a_i)$$

In other words, the weights of the features that have a large (greater than $A$) contribution to the total distance in frames of the same sequence are reduced by their deviation from the average contribution. The justification is that such features are not so relevant in segmentation given the fact that they recorded a large contribution for frames in the same sequence.

Similarly, the average entry $\mathbf{b} = [b_0 \quad b_1 \quad b_2 \cdots b_{m-1}]$ in the difference table is computed, as well as the average element $B$ in that entry

$$B = \frac{1}{m} \sum_{i=0}^{m-1} b_i$$

The error derived from the difference table is expressed as

$$\text{error}_2 = \frac{1}{e}\sum_{i=0}^{m-1}\Gamma^2(b_i - B)$$

where $e$ is the number of elements in $\mathbf{b}$ that are less than $B$. The weight updates based on this error are:

$$\Delta w_i = -\eta\Gamma(b_i - B)$$

That is, the weights of the features that have a small contribution to the total distance in frames of distinct segments are enforced. The purpose is to emphasize the relevance of such features in distinguishing between frames of different segments.

### 6.4.2 Weights for DTW Distance

To adjust the weights of the DTW distance $\mathbf{z}$, similarity and difference tables are formed.

- Similarity table for $\mathbf{z}$:

  Each entry is a vector computed as

  $$\mathbf{z} * (\mathbf{x} - \mathbf{y})$$

  where $\mathbf{x}$ is the starting frame in a segment belonging to the reference sample, and $\mathbf{y}$ is the starting frame in the corresponding segment of another sample.

- Difference table for $\mathbf{z}$:

  Each entry is a vector computed as

  $$\mathbf{z} * (\mathbf{x}' - \mathbf{y}')$$

where $\mathbf{x}'$ is the starting frame in a segment belonging to the reference sample, and $\mathbf{y}'$ is the starting frame in a segment adjacent to the corresponding segment of another sample.

The error derived from the similarity table for $\mathbf{z}$ is calculated as

$$\text{error}_3 = \frac{1}{p}\sum_{i=0}^{m-1}\Gamma^2(C - c_i)$$

where $\mathbf{c} = [c_0 \;\; c_1 \;\; c_2 \cdots c_{m-1}]$ is the average entry in the similarity table for $\mathbf{z}$.

$$C = \frac{1}{m}\sum_{i=0}^{m-1} c_i$$

and $p$ is the number of elements in $\mathbf{c}$ that are greater than $C$. The weight updates according to this error are

$$\Delta z_i = \eta\Gamma(C - c_i)$$

The error derived from the difference table for $\mathbf{z}$ is calculated as

$$\text{error}_4 = \frac{1}{q}\sum_{i=0}^{m-1}\Gamma^2(f_i - F)$$

where $\mathbf{f} = [f_0 \;\; f_1 \;\; f_2 \cdots f_{m-1}]$ is the average entry in the difference table for $\mathbf{z}$.

$$F = \frac{1}{m}\sum_{i=0}^{m-1} f_i$$

and $q$ is the number of elements in $\mathbf{f}$ that are less than $F$. The weight updates according to this error are

$$\Delta z_i = -\eta\Gamma(f_i - F)$$

The overall error is calculated as

$$\text{error}_1 + \text{error}_2 + \text{error}_3 + \text{error}_4$$

In this chapter, we have addressed the problem of segmenting a word into visual speech units. The next chapter addresses the problem of visual word recognition using recurrent neural networks.

# Chapter 7

# Visual Speech Recognition

## 7.1 Neural Network Architecture

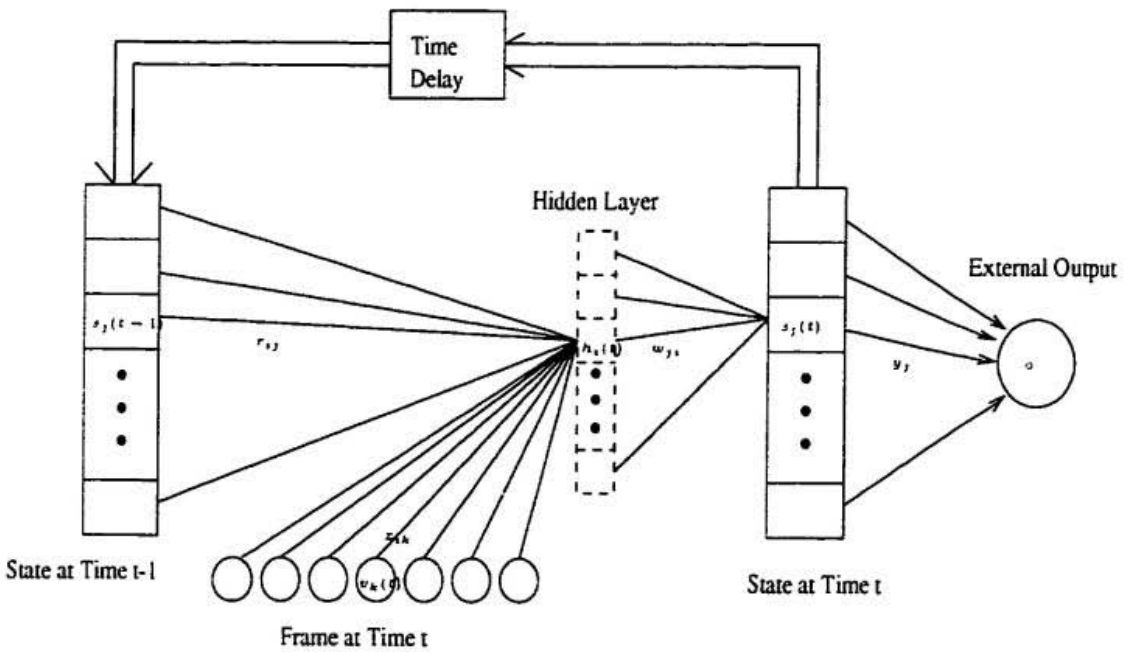


Figure 7.1: Recurrent neural network for visual speech recognition.

### 7.1.1 Neural Network Computation

The input frames are presented to the input layer sequentially, one at a time. The input frame at time $t$ is denoted $\mathbf{v}(t)$, and the $k$-th unit in the input layer is denoted $v_k(t)$.

The units of the hidden layer are computed according to

$$h_i(t) = f\left(\sum_k x_{ik} v_k(t) + \sum_j r_{ij} s_j(t-1)\right)$$

where $\mathbf{s}(t-1)$ is the time-delayed state vector, $x_{ik}$ is the weight on the connection from the $k$-th input unit to the $i$-th hidden unit, $r_{ij}$ is the weight on the connection from the $j$-th unit in the time-delayed state vector to the $i$-th hidden unit, and $f$ is a sigmoidal function.

$$f(\alpha) = \frac{1}{1+e^{-\alpha}}.$$

The connections from the time-delayed state layer to the hidden layer are called recurrent connections. These connections make the network different from an ordinary multi-layer feed-forward neural network. The purpose of such connections is to provide a context for each input pattern so that the network can capture not only the spatial characteristics of the individual patterns, but also the dynamic change of patterns with time. Before presenting the first input frame to the network, the time-delayed state vector is assumed to have no activity, that is, all units in that layer are zero.

To compute the current state vector $\mathbf{s}(t)$, the following is used

$$s_j(t) = f\left(\sum_i w_{ji} h_i(t)\right)$$

where $w_{ji}$ is the weight on the connection from the $i$-th hidden unit to the $j$-th state unit.

The external output is considered only when all input frames have been presented to the network, and is calculated as

$$o = f\left(\sum_j y_j s_j(t)\right)$$

where $y_j$ is the weight on the connection from the $j$-th state unit to the external output unit. The recurrent neural network is shown is Figure 7.1.

### 7.1.2 Desired Activity on State Vector

Given a correct sequence of frames

$$\mathbf{v}_0\mathbf{v}_1\mathbf{v}_2\cdots\mathbf{v}_{n-1}$$

which can be segmented at $k_1, k_2, k_3, \ldots, k_{l-1}$, the desired activity on the state vector is formulated as follows:

> The frames belonging to the same segment should cause the network to form the same pattern on the state vector.

That is, when presenting the frames of the first segment (segment 0), the state should be at each time the same, call it $s_1$. When presenting the frames of the second segment, the state should be $s_2$ and so on. After presenting all the frames

to the network, the external output is inspected. It is designed to recognize the last state which corresponds to the last segment.

In general, $s_i$ and $s_j$, where $i < j$, should be different from each other. Otherwise, the order segment $i-1$ preceding segment $j-1$ could never be enforced. That is because $s_i$ and $s_j$ would correspond to frames of both segments. Even if the two segments are spatially identical, it should be emphasized that the corresponding states must be made different. The correspondence between segments and states is not based on the spatial characteristics of the segments alone. It is also based on the order of the segments. This results in state $s_j$ being characterized by:

- The spatial characteristics of the individual components corresponding to that state.
- The context provided by the recurrent connections (preceded by state $s_{j-1}$).

There is an obvious recursion in the previous characterization, the main purpose of which is to ensure that the segments follow the specific order dictated by the nature of the word to be recognized.

**State Encoding**

The following encoding scheme will be used for the states:

$$s_0 \rightarrow 000\cdots00$$

$$s_1 \rightarrow 100\cdots00$$

$$s_2 \rightarrow 010\cdots00$$

$$\vdots$$

$$s_l \rightarrow 000\cdots10$$

$$s_\infty \rightarrow 000\cdots01$$

The state variable contains at least $l+1$ bits (any number of trailing zeros should not make a difference). $s_0$ encodes the initial state, and $s_\infty$ encodes an undefined state which indicates failure to recognize the sequence. In this scheme, each unit on the state layer corresponds to a particular state. Note that, from a theoretical viewpoint, one set of state variables is as good as any other.

## 7.2 Training

Two types of training are commonly seen: *supervised* and *unsupervised*. In supervised training, there is a "teacher" who "tells" the network what the correct output for a certain input is. In unsupervised training, the network is autonomous. It finds out about some of the properties of the data set, and learns to reflect these properties in its output. For the network used in this thesis, we chose supervised training since we knew the correct output for each training sequence.

Suppose the recurrent neural network (RNN) described previously has trained to recognize a word $w$ of the form

$$\underbrace{\mathbf{v}_0\mathbf{v}_1\cdots\mathbf{v}_{k_1-1}}_{\text{segment 0}}\underbrace{\mathbf{v}_{k_1}\mathbf{v}_{k_1+1}\cdots\mathbf{v}_{k_2-1}}_{\text{segment 1}}\cdots\underbrace{\mathbf{v}_{k_{l-1}}\mathbf{v}_{k_{l-1}+1}\cdots\mathbf{v}_{n-1}}_{\text{segment } l-1}$$

then, on a new instance $w'$ (of the same class of $w$)

$$\underbrace{\mathbf{v}'_0\mathbf{v}'_1\cdots\mathbf{v}'_{k'_1-1}}_{\text{segment 0}}\underbrace{\mathbf{v}'_{k'_1}\mathbf{v}'_{k'_1-1}\cdots\mathbf{v}'_{k'_2-1}}_{\text{segment 1}}\cdots\underbrace{\mathbf{v}'_{k'_{l-1}}\mathbf{v}'_{k'_{l-1}-1}\cdots\mathbf{v}'_{n'-1}}_{\text{segment } l-1}$$

where

$$\begin{aligned}
\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k_1-1}, \mathbf{v}'_0, \mathbf{v}'_1, \ldots, \mathbf{v}'_{k'_1-1} &\in \mathcal{V}_0 \\
\mathbf{v}_{k_1}, \mathbf{v}_{k_1-1}, \ldots, \mathbf{v}_{k_2-1}, \mathbf{v}'_{k'_1}, \mathbf{v}'_{k'_1-1}, \ldots, \mathbf{v}'_{k'_2-1} &\in \mathcal{V}_1 \\
&\vdots \quad \vdots \\
\mathbf{v}_{k_{l-1}}, \mathbf{v}_{k_{l-1}-1}, \ldots, \mathbf{v}_{n-1}, \mathbf{v}'_{k'_{l-1}}, \mathbf{v}'_{k'_{l-1}+1}, \ldots, \mathbf{v}'_{n'-1} &\in \mathcal{V}_{l-1}
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{V}_0 &\text{ is distinct from } \mathcal{V}_1 \\
\mathcal{V}_1 &\text{ is distinct from } \mathcal{V}_2 \\
&\vdots \qquad\qquad\quad \vdots \\
\mathcal{V}_{l-2} &\text{ is distinct from } \mathcal{V}_{l-1}
\end{aligned}$$

it is required that the RNN behaves as follows:

- Just before taking the first input, the current state is in initial configuration $s_0$.
- On inputs $\mathbf{v}'_0, \mathbf{v}'_1, \ldots, \mathbf{v}'_{k'_1-1}$, the RNN should have $s_1$ on the state layer.
- On inputs $\mathbf{v}'_{k'_1}, \mathbf{v}'_{k'_1+1}, \ldots, \mathbf{v}'_{k'_2-1}$ (frames of the segment 1), the state should be

$s_2$.

$\vdots$

- On inputs $\mathbf{v}'_{k'_{l-1}}, \mathbf{v}'_{k'_{l-1}-1}, \ldots, \mathbf{v}'_{n'-1}$, the state should be $s_l$.

- When all the frames are presented and the state vector is in $s_l$, the external output which is connected to the state layer by feed-forward connections must have a high value indicating that $w'$ is recognized.

Therefore, the RNN needs to learn from the sample word $w$ the following transformation:

$$R : (\mathbf{v}(t), s(t-1)) \rightarrow s(t)$$

such that

$$\left\{\begin{array}{ll} R(\mathbf{v}_0, s_0) & = s_1 \\ R(\mathbf{v}_1, s_1) & = s_1 \\ \vdots & \\ R(\mathbf{v}_{k_1-1}, s_1) & = s_1 \end{array}\right. \left\{\begin{array}{ll} R(\mathbf{v}_{k_1}, s_1) & = s_2 \\ R(\mathbf{v}_{k_1-1}, s_2) & = s_2 \\ \vdots & \\ R(\mathbf{v}_{k_2-1}, s_2) & = s_2 \end{array}\right. \cdots \left\{\begin{array}{ll} R(\mathbf{v}_{k_{l-1}}, s_{l-1}) & = s_l \\ R(\mathbf{v}_{k_{l-1}-1}, s_l) & = s_l \\ \vdots & \\ R(\mathbf{v}_{n-1}, s_l) & = s_l \end{array}\right.$$

More generally,

$$R(\mathbf{v}, s_i) = \left\{\begin{array}{ll} s_1 & \text{if } i = 0 \text{ and } \mathbf{v} \in \mathcal{V}_0 \\ s_i & \text{if } i < 0 \leq l \text{ and } \mathbf{v} \in \mathcal{V}_{i-1} \\ s_{i-1} & \text{if } i < 0 < l \text{ and } \mathbf{v} \in \mathcal{V}_i \\ s_\infty & \text{otherwise} \end{array}\right. \tag{7.1}$$

The last line is to ensure that the RNN does not recognize wrong sequences. Note that it is not necessary for

$$k_1 = k'_1, k_2 = k'_2, \ldots, k_{l-1} = k'_{l-1}, \text{ and, } n = n'$$

to hold. This makes the system tolerant to variable-length segments, which enables it to handle realistic situations where the time taken to speak a certain word varies from instance to instance even by the same speaker. Furthermore, it is almost impossible to have the duration of a certain segment exactly the same in different instances of a given word.

In summary,

1. A dynamical (abstract) system with time-varying input and state is strongly believed to have the potential to handle important aspects of word recognition. The behavior of the system can be described by

$$x(t) = y(x(t-1), \text{input}(t)), \qquad t_0 < t \leq t_f$$

2. There are time-delayed samples (to be used as training data) of the system which include state variables expected to appear during the system's operation

| Input | State |
|---|---|
| $\text{input}(t_1)$ | $x(t_1)$ |
| $\text{input}(t_2)$ | $x(t_2)$ |
| $\vdots$ | $\vdots$ |
| $\text{input}(t_f)$ | $x(t_f)$ |

These are obtained by specifying a desired behavior, where the specification is based on characterizing a word by certain segments appearing in a specific order.

Given points 1 and 2, and according to a proposition by Olurotimi [48], if an RNN is to learn the dynamics of the system $x(t)$ using the available training data, it does

not need more than feed-forward complexity. That is, it is only required to train the feed-forward network embedded in the recurrent architecture by conventional methods (back-propagation). The RNN will learn its required dynamic behavior as much as the feed-forward network learns its static task, namely, the transformation $R : \mathcal{V} \times \mathcal{S} \rightarrow \mathcal{S}$, where $\mathcal{V}$ is the set of input patterns and $S$ is the set of states.

The feed-forward neural network embedded in the recurrent architecture is obtained by the following steps:

1. Remove the external output unit and its connections since this part of the network performs only a static pattern classification task that can be left out at this stage without affecting the training of other parts.

2. Remove the time-delay connections.

3. Consider the time-delayed state vector as part of the input in the embedded network.

4. Consider the state vector as the output layer in the embedded architecture.

Figure 7.2 demonstrates the process.

### 7.2.1 Training Data for the Embedded Feed-Forward Network

Equation (7.1) is used as the basis for obtaining the training samples for the embedded network. Given training sequences for the RNN containing instances of the

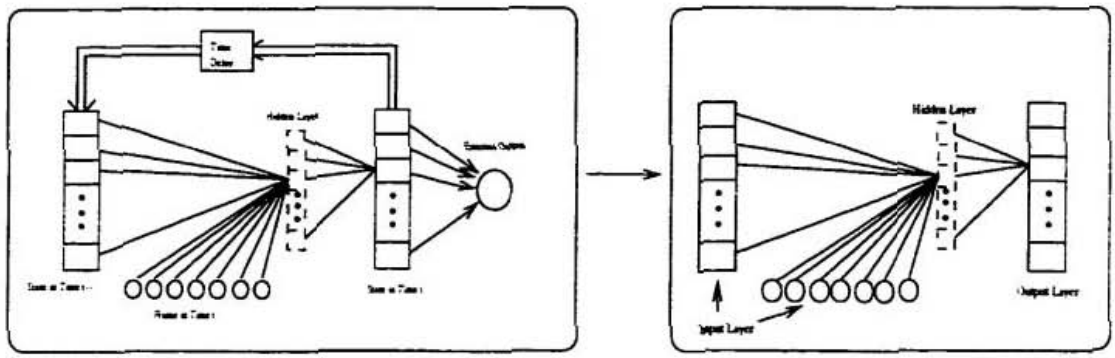

Figure 7.2: Obtaining the feed-forward neural network embedded in the RNN.

word $w$ that is supposed to be recognized by the network as well as instances of other words. an algorithm has been designed to generate the training samples for the embedded feed-forward network.

Each sample generated by the algorithm is an input-output pair. The input has two parts: state and (speech) frame. The generated samples contain all possible states. That is, there are samples containing $s_0$ in their state part, others containing $s_1$, and so on.

- For samples containing $s_0$ as part of the input, there are two possible static outputs:

  1. $s_1$: if the frame part of the sample belongs to segment 0 of word $w$.

  2. $s_\infty$: otherwise.

  Therefore, there should be samples of the form

$$\text{input} = s_0, \mathbf{u}$$

$$\text{output} = s_1$$

where **u** is a frame belonging to segment 0 of $w$. The pseudo code used to produce such samples is

```
for each sample sequence s belonging to word w
    for each frame u in segment 0 of s
        output(s_0, u, s_1)
    end for
end for
```

Also, there should be samples of the form

$$\text{input} = s_0, \mathbf{v}$$

$$\text{output} = s_\infty$$

where **v** is a frame belonging to a segment that is spatially different from segment 0 of $w$. The issue of obtaining such frames will be discussed later.

- For samples containing $s_k$, $k = 1, 2, 3, \ldots, l-1$, there are three possible outputs:

  1. $s_k$: if the associated frame belongs to segment $k-1$ of word $w$.

  2. $s_{k-1}$: if the associated frame belongs to segment $k$ of word $w$.

  3. $s_\infty$: otherwise.

- For samples containing $s_l$ in the input part, there are two possible outputs:

  1. $s_l$: if the associated frame belongs to segment $l-1$ of word $w$.

  2. $s_\infty$: otherwise.

- Finally, for samples containing $s_\infty$, there is only one possible output which is $s_\infty$.

### 7.2.2 Obtaining frames for the undefined state $s_\infty$

As shown earlier, the embedded network should be trained to produce $s_\infty$ if the frame present in the input is not consistent with the input state. This includes the following cases:

1. The state is $s_0$ and the frame belongs to a segment that is spatially different from segment 0.

2. The state is $s_k$, $k = 1, 2, 3, \ldots, l-1$, and the frame belongs to a segment that is spatially different from segments $k$ and $k-1$.

3. The state is $s_l$ and the frame belongs to a segment that is spatially different from segment $l-1$.

It is very important to identify such spatially different segments as failure to do so could probably lead to problems in training. For example, if a certain segment $i$ was assumed wrongly to be spatially different from segment 0, the static sample

$$\text{input} = s_0, \mathbf{v}$$

$$\text{output} = s_\infty$$

where $\mathbf{v}$ belongs to segment $i$; could be included in the training set. However, the training set contains, as well, samples of the form

$$\text{input} = s_0, \mathbf{u}$$

$$\text{output} = s_1$$

where $\mathbf{u}$ belongs to segment 0. Since the inputs of the two samples are practically of the same class while the outputs $s_1$ and $s_\infty$ are different, this situation violates the definition of a mapping, and if allowed will certainly confuse the training process.

Even if the spatial characteristics of segment $i$ are slightly different from those of segment 0, the frames of segment $i$ should not be used to generate samples that have $s_0$ in the input state and $s_\infty$ in the output. Those frames should not be used because, after training, unseen frames that supposedly belong to segment 0 will be possibly viewed by the network to be close to both segments 0 and $i$. If the training set included such samples, this would increase the chance that the network enters undefined state configurations for sequences that are acceptably close to those used in training. Therefore, when generating the static samples that contain a particular input state and map to $s_\infty$, it is necessary to select the most distant segments from the one that corresponds to that state. One way to achieve this is by developing a difference measurement between segments, and selecting the segments with the highest difference values. To implement such difference measurement, some of the concepts discussed in the previous chapter will be employed.

Assume that there are $n$ samples of segment $a$ taken from words belonging to class $c_1$, and $m$ samples of segment $b$ taken from words belonging to class $c_2$. Let $\text{dist}_1$ be the weighted distance function used for aligning the segments of the words belonging to class $c_1$, and $\text{dist}_2$ be the weighted distance function used for aligning the segments of the words belonging to class $c_2$. First, each segment is mapped to a representative frame by averaging all the frames in the segment. This results in two sets of frames $U$ and $V$ containing the representative frames for sample segments of kinds $a$ and $b$ respectively. Second, a new distance function, dist, is derived by averaging the weights of the two distance functions $\text{dist}_1$ and $\text{dist}_2$. Accordingly, for any two frames $\mathbf{u} \in U$ and $\mathbf{v} \in V$, the distance between them (and also between the two segment samples being represented) is calculated as

$$\text{dist}(\mathbf{u}, \mathbf{v}) = \frac{\text{dist}_1(\mathbf{u}, \mathbf{v}) + \text{dist}_2(\mathbf{u}, \mathbf{v})}{2}$$

To get an overall difference measurement between the two kinds of segments, the distance between all possible $nm$ frame pairs $\mathbf{u} \in U$ and $\mathbf{v} \in V$, is calculated. Then the average of the pairwise distances is taken

$$\frac{1}{nm} \sum_{\mathbf{u} \in U} \sum_{\mathbf{v} \in V} \text{dist}(\mathbf{u}, \mathbf{v}).$$

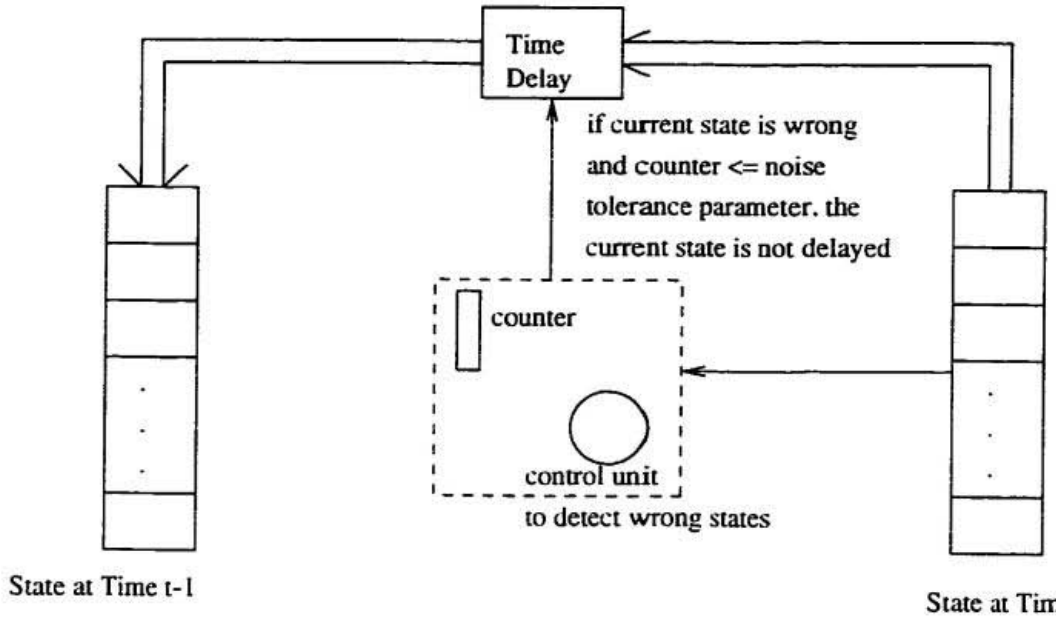


Figure 7.3: Noise handling in the recurrent networks used for visual speech recognition.

## 7.3 Noise Handling

In the original model described in subsection 7.1.1, if a correct sequence contains some noise, this will cause the state to become $s_\infty$ when the network reads the first noisy frame, and stay $s_\infty$ until the end of the sequence. This situation is not desirable since it does not tolerate any noise. What makes the model intolerant to noise is the fact that the current state is delayed every time to be part of the input to the hidden layer in the next time instance.

To make the model noise-tolerant, a mechanism to monitor the current state is incorporated by having a control unit that indicates whether the current state will cause the sequence to be rejected when time-delayed. If this is the case, the

current state is ignored and an internal counter is incremented. The same scenario is allowed to happen for subsequent frames as long as the value of the counter does not exceed the value of a parameter called the *noise tolerance* parameter. If the network recovers from the wrong configuration before exceeding the noise tolerance parameter, the counter is reset and the current state is time-delayed. Otherwise, the sequence contains more noise than what can be tolerated and is considered wrong. Figure 7.3 demonstrates the mechanism for noise handling. Note that the noise tolerance parameter is the maximum number of consecutive noisy frames that are allowed to exist in a correct sequence. If this parameter is set to zero, not a single noisy frame will be allowed to occur in a correct sequence, and the network will be noise-intolerant.

## 7.4 External Output

The external output unit indicates whether the sequence is correct or not. To determine this, the output unit is inspected after all the input frames have been fed to the network. To calculate the external output, the state layer is copied into a Maximum Detection Subnetwork (MDS) which makes the maximum unit one and the rest of the units zero. This can be implemented by a Hopfield network performing a *winner-take-all* competition in which every unit enforces itself and inhibits the others. The final state unit in the MDS is, then, copied into the external output unit. This way, any sequence causing the network to have a maximum output on the final state unit,

at the end, will be recognized as a correct sequence. Sequences causing a maximum output on any other state will be considered wrong. The computation of the external output is illustrated in Figure 7.4.

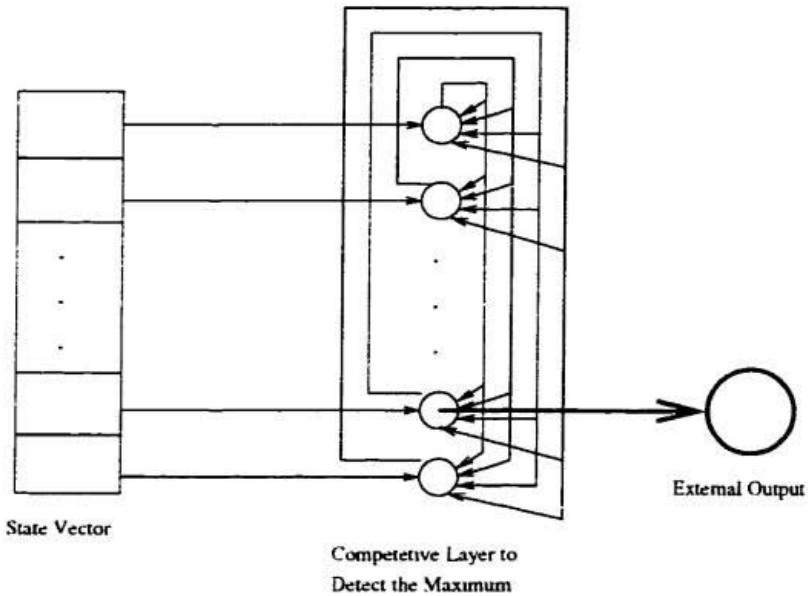


Figure 7.4: Computation of the external output in the recurrent networks used for visual speech recognition.

# Chapter 8

# Experiments, Results and Conclusion

## 8.1 Experiments and Results

In this thesis, a computer system for visual speech recognition has been presented. The input to the system is a sequence of digital images containing the face of a person during speech. In the first phase of the system's operation, time-varying visual speech patterns are obtained from the sequence of images. Through a number of algorithms, that have been developed, main characteristics of the mouth are estimated. These estimates are used to initialize a deformable template model. An energy function has been designed to measure how well the template's geometric primitives match the lips' outlines in the image. Due to the relatively high dimensionality of the energy function, seeking an exact solution for its minimization problem is not practical. Using a numerical optimization technique, a good solution is obtained with considerably less computational and storage requirements than that of an exhaustive method. Taking advantage of the relatively small change between consecutive

frames, the system sets as initial position of each subsequent frame the best fit of the preceding one, thus, eliminating the need to apply the initial operations to every single frame. A recurrent neural network architecture has been proposed to classify the spatio-temporal pattern obtained in the first phase. In this network, recurrent connections are made between the hidden layer and the state layer so that a context can be combined with the input patterns which are fed to the network one at a time. Training the recurrent network is accomplished by training the feed-forward network embedded in the recurrent architecture. To derive static training samples for the feed-forward network, a certain behavior is specified when the network is presented with sample sequences. The specification is based on characterizing a given word by a sequence of segments appearing in a certain order, where each segment is a variable-length set of frames that represent a visual speech cue. Adaptive segmentation is employed to segment the training sequences of a given class. This method iterates the execution of two steps. First, the sample sequences are segmented individually by an algorithm that has been developed. Then, a generalized version of dynamic time warping is used to align the segments of all sequences belonging to the same class. At each iteration, the weights of the distance functions used in the previous two steps are updated adaptively in a way that minimizes a segmentation error.

The system has been implemented using the C language and simulated on a Sun Sparc workstation under the Unix operating system. To test the performance of the mouth extraction subsystem, the technique has been applied to 15 people. Figure

8.1 shows the result of some samples.

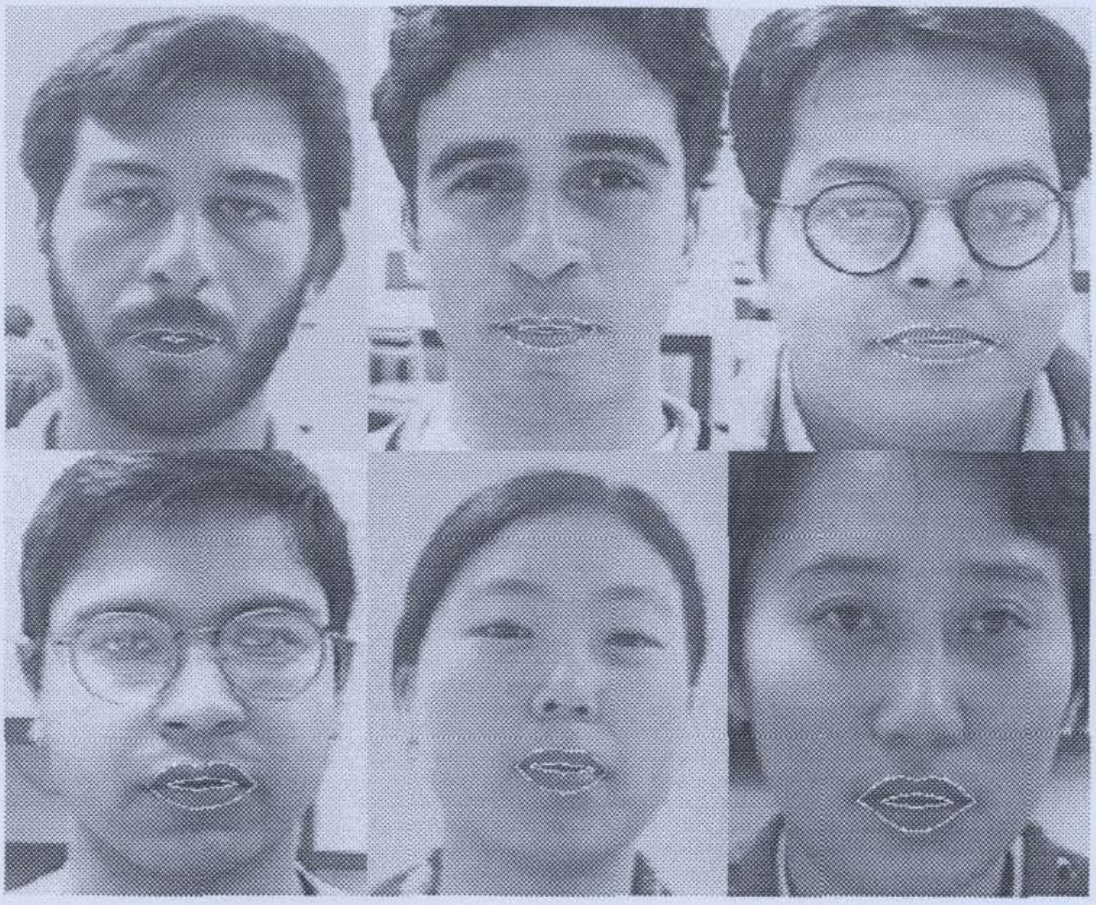

Figure 8.1: Deformable template applied to images.

As can be seen, the method tolerates some rotation and facial tilt. The grey scale values of the pixels in the lips region varied from person to person, and the program was still able to get a good match. Also, the presence of facial hair did not seem to have affected the performance of the program. It should be noted that when matching natural curves with parametrized curves, slight details might be missed. However, the final parameter values are, to a great extent, accurate. This is suitable for the task of visual speech recognition, since the task uses only the dynamic change

of those parameters over time. Because the method of locating the mouth depends on locating the eyes, the system is limited to cases where the speaker's eyes can be seen. If the speaker's eyes are obscured by wearing a pair of sunglasses, for example, the system will not be expected to locate the mouth appropriately because it will probably fail in locating the eyes.

To test the complete system, experiments have been carried out to recognize 5 words: yes, no, down, right, left. A person was recorded while uttering each word 6 times using an 8 mm Sony Handycam. The recordings were copied to a Beta format tape, then digitized and sampled at the rate of 30 frames per second in one of two methods. In the first method, the Beta tape was played back one frame at a time using a Beta ACE editor. The ACE editor produced an NTSC signal that was captured and saved to disk by a Sun Videopix capture board. In the second method, the Beta tape was played by a Beta ACE editor and the output was sent to a *personal animation recorder*. This recorder consisted of two specialized hardware cards that were put in a Pentium computer system. One of the cards was a Live Video card with a time base corrector. The Live Video card sent its output to a PAR board (video compression board) that stored the video on a hard drive in compressed format. The personal animation recorder system had software to retrieve the video and store it as a sequence of individual frames in jpg format. A training data set consisting of 4 instances of each word class was used to train 5 recurrent units corresponding to the 5 word classes. The training sequences were segmented using the adaptive

Table 8.1: Results of implementing the visual speech recognition system to recognize 5 word classes (adaptive segmentation was used in this implementation)

| Word | Training Sequences | Static Samples | Convergence error | | iteration | Performance on Training Set | Performance on Test Set |
|---|---|---|---|---|---|---|---|
| yes | 4 | 498 | 0.0273 | at | 1000 | 19/20 | 10/10 |
| no | 4 | 376 | 0.0100 | at | 726 | 20/20 | 10/10 |
| down | 4 | 654 | 0.0099 | at | 842 | 20/20 | 10/10 |
| right | 4 | 570 | 0.0095 | at | 614 | 20/20 | 10/10 |
| left | 4 | 532 | 0.0240 | at | 1000 | 20/20 | 10/10 |

segmentation technique proposed in chapter 6. Accordingly, static samples were generated to train the feed-forward networks corresponding to the recurrent units as described in chapter 7. Each feed-forward network was trained independently for 1000 iterations using the backpropagation algorithm described in subsection 2.5.1. Training was stopped in any of the networks whenever the system error for that network was reduced to 0.01. After training, the system was tested on a data set consisting of the sequences that were not used for training (2 instances for each word). Table 8.1 summarizes the training process and the classification results for the system. There was only one case where a misclassification happened. One of the 'right' instances was confused to be 'yes', which caused the performance of the 'yes' unit on the training set to be 19/20. Due to the relatively small size of the training data, the 'yes' unit was not able to capture all the differences between the two words. There were no other kinds of confusions. In particular, the recognition scheme was able to distinguish between 'down' and 'right' even though they have common starting parts (/da/ /ra/). Also, 'down' and 'no' were distinguishable despite the fact that

Table 8.2: Results of implementing the visual speech recognition system to recognize 5 word classes (adaptive segmentation was not used in this implementation)

| Word | Training Sequences | Static Samples | Convergence error | | iteration | Performance on Training Set | Performance on Test Set |
|---|---|---|---|---|---|---|---|
| yes | 4 | 554 | 0.3924 | at | 1000 | 16/20 | 8/10 |
| no | 4 | 444 | 0.0153 | at | 1000 | 20/20 | 10/10 |
| down | 4 | 694 | 0.0117 | at | 1000 | 20/20 | 8/10 |
| right | 4 | 646 | 1.4941 | at | 1000 | 16/20 | 8/10 |
| left | 4 | 628 | 0.0170 | at | 1000 | 18/20 | 9/10 |

most segments in 'no' do exist in 'down'.

To show the effect of the adaptive segmentation technique proposed in this thesis, another system was built in exactly the same way as the first one, except that the adaptive segmentation of chapter 6 was not used. The results are summarized in Table 8.2.

The convergence of the second system was generally slower than before, indicating that adaptive segmentation provides an easier task to learn. Furthermore, the classification results were much better in the first system that used adaptive segmentation. It is worth mentioning here that the computational time required to train the tasks of the first system, which used adaptive segmentation, was less than the time required by the tasks of the second system. These results demonstrate that segmentation plays a crucial role in visual speech recognition, and the method of adaptive segmentation when applied to the training sequences leads to a better system.

## 8.2 Contributions

1. Knowledge about the spatial organization of the human face has been used to develop a heuristic that limits the search space of the mouth location effectively. The heuristic is based on characterizing the mouth by its relative location with respect to the eyes rather than local details of the human mouth which are very sensitive to distance, orientation and illumination; furthermore, these can vary from person to person. The eyes are located by applying a simple version of slit analysis to the Laplacian of the facial image.

2. Several algorithms proposed here have been applied to estimate local measurements of the mouth, including the center, left and right corners, and upper and lower lips. These algorithms analyze the relative magnitude of the grey levels associated with horizontal and vertical lines in the region of interest. The algorithms provide a robust means of positioning an initial mouth template prior to the application of the deformable template. This is known to be crucial for the technique to succeed in extracting the shape of the mouth.

3. The two-phase minimization algorithm proposed in this thesis, which uses a modified version of Powell's method, has been shown to be suitable for minimizing the energy function of the mouth deformable template, so that the system is able to track the speaker's mouth during speech.

4. The approach taken to model the mouth in terms of key parameters not only enables the extraction of its shape, but also provides a compact description of it for classification.

5. The fact that there is a relatively small change in the visual speech signal makes the task of visual word segmentation difficult. This thesis presents a robust method to identify potential candidates for segment boundaries.

6. In the project, the dynamic time warping algorithm has been generalized to take several sequences as input instead of two. The generalized algorithm selects the segment boundaries for words of the same class by finding warping paths that minimize a weighted distance measure.

7. The idea of adaptive segmentation has been introduced. Given an initial segmentation of a set of words belonging to a certain class, properties of the segments are learned and reflected in the weights of the distance functions used for segmentation and alignment such that a better segmentation will be produced if the adjusted distance functions are used.

8. A recurrent neural network architecture has been proposed for visual word recognition. The recurrent connections between the state and the hidden layers provide a context with each speech pattern so that the network is capable of capturing not only the spatial characteristics of the individual patterns, but also the dynamic change of the patterns with time.

9. The proposed architecture for recognition allows for a natural way of handling variable-length words by taking the input patterns one at a time. This eliminates the need to alter the input sequences to make them of a fixed length.

10. By specifying a desired behavior based on word segmentation, the recurrent network is trained with no more than feed-forward complexity. Theoretically, this approach guarantees that the dynamical behavior is learned as long as there is an unambiguous training set representing the state variables of the system in the form of static samples.

## 8.3 Directions for Future Work

### 8.3.1 Use of Other Articulators

In this thesis, the shape of the lips has been used for speech recognition. Perhaps the movement of the lips is the most valuable source of information, but it is not the only one. For example, the appearance of the teeth and tongue may assist in automatic lipreading as it is believed to do in human lipreading. A system that would incorporate those articulators should contain a subsystem which automatically analyzes the interior of the mouth, searching for such objects as the teeth and tongue. A suitable modeling technique for the teeth and tongue would also be necessary.

### 8.3.2 Clustering Segments

When static samples for training are being generated, determining the most distant segments from a particular segment is significant in preserving the network's ability

to tolerate spatial variations in unseen sequences. Our system uses a linear distance function for the task. This might not be the best way to do it. Clustering all the segments based on nonlinear associations to determine the difference between segments might provide an improved system, for example.

### 8.3.3 Nonlinear Classifiers for Segmentation

The idea of adaptive segmentation is implemented using linear functions. In general, linear functions have limited classification capabilities. Therefore, it is believed that the adaptive segmentation technique will improve if it uses nonlinear functions. A suitable way to implement this could be by neural network classifiers.

### 8.3.4 Extension to Multi-Speakers

It is desirable to extend the system to handle multiple speakers. We believe that this will add a new dimension to the problem. For example, because people are different in the physical characteristics of their mouths, it is possible that the geometric features representing mouth shapes of two different speakers at the same viseme be different. To handle this properly, it might be necessary to incorporate additional features characterizing the speaker with each frame. Another implication of having multiple speakers is the alignment between training sequences of different speakers. The distance function used for that would have to be designed in a way that accounts for the differences between different speech cues but not for the differences emerging from the physical characteristics of the speakers.

# Bibliography


[1] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9), 855-867, 1990.

[2] R. J. Baron, "Mechanisms of human facial recognition." *International Journal of Man-Machine Studies*, 15(2), 137-178, 1981.

[3] Y. Bengio, "Artificial neural networks and their application to sequence recognition." Ph.D. Thesis, McGill University, 1991.

[4] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Transactions on Neural Networks*, 3(2), 252-259, 1992.

[5] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult." *IEEE Transactions on Neural Networks*, 5(2), 157-166, 1994.

[6] C. Bregler, and Y. Konig, " 'Eigenlips' for robust speech recognition." *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Adelaide, Australia, 1994.

[7] C. Bregler, and S. M. Omohundro, "Surface learning with applications to lipreading," in: J. D. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann Publishers, San Francisco, CA, 43-50, 1994.

[8] C. Bregler, S. M. Omohundro, and Y. Konig, "A hybrid approach to bimodal speech recognition." *28th Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 1994.

[9] R. P. Brent, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973.

[10] N. M. Brooke, and E. D. Petajan, "Seeing speech: Investigations into the synthesis and recognition of visible speech movements using automatic image processing and computer graphics," *Proceedings of the Interanational Conference on Speech Input and Output: Techniques and Applications*, 104-109, March 1986.

[11] V. Bruce, and M. Burton, "Computer recognition of faces," in: A. W. Young, and H. D. Ellis (eds.), *Handbook of Research on Face Processing*, Elsevier Science Publishers B. V. (North Holland), 487-506, 1989.

[12] R. Campbell, "Lipreading," in: A. W. Young, and H. D. Ellis (eds.), *Handbook of Research on Face Processing*, Elsevier Science Publishers B. V. (North Holland), 187-205, 1989.

[13] B. Caprile, and F. Girosi, "A nondeterministic minimization algorithm," AI Memo 1254, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, September 1990.

[14] T. Chen, H. P. Graf, and K. Wang, "Speech assisted video processing: Interpolation and low-bitrate coding," *28th Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 1994.

[15] G. Chow, and X. Li, "Towards a system for automatic facial feature detection," *Pattern Recognition*, 26(12), 1739-1755, 1993.

[16] T. Coianiz, L. Torresani, and B. Caprile, "2D deformable models for visual speech analysis," in: D. G. Stork (ed.), *NATO ASI Speechreading by Man and Machine: Models, Systems, and Application*, Springer Verlag, 1995.

[17] T. F. Cootes, A. Hill, C. J. Taylor, and J. Haslam, "Use of active shape models for locating structures in medical images," *Image and Vision Computing*, 12(6), 355-365, 1994.

[18] I. Craw, H. Ellis, and J. R. Lishman, "Automatic extraction of face-features," *Pattern Recognition Letters*, 5(2), 183-187, 1987.

[19] I. Craw, D. Tock, and A. Bennett, "Finding face features," *Second European Conference on Computer Vision*, Santa Margherita Ligure, Italy, 92-96, May 1992.

[20] J. A. Freeman, D. M. Skapura, *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley Publishing Company, Inc., 1992.

[21] K. E. Finn, and A. A. Montgomery, "Automatic optically-based recognition of speech," *Pattern Recognition Letters*, 8(3), 159-164, 1988.

[22] A. J. Goldschen, O. N. Garcia, and E. Petajan, "Continuous optical automatic speech recognition by lipreading," *28th Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 1994.

[23] H. P. Graf, T. Chen, E. Petajan, and E. Cosatto, "Locating faces and facial parts," *International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 41-46, June 1995.

[24] P. Haffner, M. Franzini, and A. Waibel, "Integrating time alignment and neural networks for high performance continuous speech recognition," *International Conference on Acoustics, Speech, and Signal Processing*, New York, NY, 105-108, 1991.

[25] J. B. Hampshire, and A. H. Waibel, "The Meta-Pi network: Building distributed knowledge representations for robust pattern recognition," Technical Report CMU-CS-89-166, Carnegie Mellon University, August 1989.

[26] M. E. Hennecke, K. V. Prasad, and D. G. Stork, "Using deformable templates to infer visual speech dynamics," *28th Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 578-582, 1994.

[27] H. Hild, and A. Waibel, "Multi-speaker/speaker-independent architectures for the multi-state time delay neural network," *International Conference on Acoustics, Speech, and Signal Processing*, April 1993.

[28] J. J. Hopfield, "Neural networks and physical systems with emergent collective abilities," *Proceedings of the National Academy of Science*, 79(8), 2254-2558, 1982.

[29] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Science*, 81(10), 3088-3092, 1984.

[30] J. J. Hopfield, and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, 52, 141-152, 1985.

[31] C.-L. Huang, and C.-W. Chen, "Human facial feature extraction for face interpretation and recognition," *Pattern Recognition*, 25(12), 1435-1444, 1992.

[32] N. Intrator, D. Reisfeld, Y. Yeshurun, "Face recognition using a hybrid supervised/unsupervised neural network," *Pattern Recognition Letters*, 17(1), 67-76, 1996.

[33] T. Kanade, *Computer Recognition of Human Faces*, Birkhauser Verlag, Basel and Stuttgart, 1977.

[34] M. Kass, A. Witkin, and D. Terzopoulos. "Snakes: active contour models." *International Journal of Computer Vision*, 1(4), 321-331, 1988.

[35] M. D. Kelly, "Edge detection in pictures by computer using planning," in: B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Edinburgh University Press, Edinburgh, 397-409, 1971.

[36] M. Kirby, F. Weisser, and G. Dangelmayr, "A model problem in the representation of digital image sequences." *Pattern Recognition*, 26(1), 63-73, 1993.

[37] K. J. Lang, and G. E. Hinton, "A time-delay neural network architecture for speech recognition," Technical Report CMU-CS-88-152, Carnegie Mellon University, December 1988.

[38] N. Li, S. Dettmer, and M. Shah, "Lipreading using eigensequences," *International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 30-34, June 1995.

[39] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, 4(2), 4-22, 1987.

[40] R. P. Lippmann, "Review of neural networks for speech recognition," *Neural Computation*, 1(1), 1-38, 1989.

[41] J. Luettin, N. A. Thacker, S. W. Beet, "Active shape models for visual speech feature extraction," in: D. G. Stork (ed.), *NATO ASI Speechreading by Man and Machine: Models, Systems, and Application*, Springer Verlag, 1995.

[42] P. Maragos, "Tutorial on advances in morphological image processing and analysis," *Optical Engineering*, 27(6), 623-632, 1987.

[43] K. Mase, and A. Pentland, "Lip reading: Automatic visual recognition of spoken words," *Image Understanding and Machine Vision*, Optical Society of America, Washington, DC, 124-127, 1989.

[44] J. R. Movellan, "Visual speech recognition with stochastic networks," in: G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge, MA, 1995.

[45] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," *Complex Systems*, 3, 349-381, 1989.

[46] K. S. Narendra, and K. Parthasarathy, "Gradient methods for the opimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, 2(2), 252-262, 1991.

[47] J. A. Nelder, and R. Mead. "A simple method for function minimization." *Computer Journal*. 7(4),308-313. 1965.

[48] O. Olurotimi. "Recurrent neural network training with feedforward complexity." *IEEE Transactions on Neural Networks*. 5(2), 185-197. 1994.

[49] Y.-H. Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Publishing Company, Inc., 1989.

[50] B. A. Pearlmutter. "Dynamic recurrent neural networks." Technical Report CMU-CS-88-191. Carnegie Mellon University. December 1990.

[51] E. D. Petajan. "Automatic lipreading to enhance speech recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. San Francisco, CA, 40-47, 1985.

[52] F. J. Pineda. "Generalization of back-propagation to recurrent neural networks." *Physical Review Letters*, 59(19), 2229-2232, 1987.

[53] K. V. Prasad, D. G. Stork, and G. J. Wolff. "Preprocessing video images for neural learning of lipreading," Technical Report CRC-TR-93-26. Ricoh Californina Research Center. September 1993.

[54] W. H. Press, S. A. Teukolski, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press. 1992.

[55] Gihad Rabi and Siwei Lu. "Visual speech recognition by recurrent neural networks." to appear in *The 10th Annual Canadian Conference on Electrical and Computer Engineering*. St. John's. NF. Canada. May 1997.

[56] R. R. Rao. and R. M. Mersereau. "Lip modeling for visual speech recognition." *28th Annual Asilomar Conference on Signals, Systems, and Computers*. Pacific Grove, CA, 1994.

[57] D. Reisfeld, H. Wolfson, and Y. Yeshurun, "Detection of interest points using symmetry," *Third International Conference on Computer Vision*, Osaka, Japan. 62-65, December 1990.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in: D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed Processing*, Vol. 1. MIT Press, Cambridge, MA. 318-362. 1986.

[59] H. Sakoe, and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1), 1978.

[60] A. Samal, and P. A. Iyengar. "Automatic recognition and analysis of human faces and facial expressions: A survey." *Pattern Recognition*, 25(1), 65-77, 1992.

[61] L. E. Scales, *Introduction to Non-linear Optimization*, Macmillan Publishers Ltd, London, 1985.

[62] P. L. Silsbee, *Computer Lipreading for Improved Accuracy in Automatic Speech Recognition*, Ph.D. Dissertation, The University of Texas at Austin, May 1993.

[63] P. L. Silsbee. "Sensory integration in audiovisual automatic speech recognition." *28th Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 1994.

[64] D. G. Stork, G. Wolff, and E. Levine, "Neural network lipreading system for improved speech recognition," *International Joint Conference on Neural Networks*, Vol. 2, Baltimore, MD, 285-295, 1992.

[65] D. W. Tank, and J. J. Hopfield, "Neural computation by concentrating information in time." *Proceedings of the National Academy of Sciences*, 84(7), 1896-1900, 1987.

[66] N. B. Toomarian, and J. Barhen, "Learning a trajectory using adjoint functions and teacher forcing." *Neural Networks*, 5(3), 473-484, 1992.

[67] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3), 328-339, 1989.

[68] P. J. Werbos, "Backpropagation through time: What it does and how to do it." *Proceedings of the IEEE*, 78(10), 1550-1560, 1990.

[69] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, 1(4), 339-356, 1988.

[70] D. J. Williams, and M. Shah, "A fast algorithm for active contours," *Third International Conference on Computer Vision*, Osaka, Japan, 592-595, 1990.

[71] R. J. Williams, and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, 1(2), 270-280, 1989.

[72] G. J. Wolff, K. V. Prasad, D. G. Stork, and M. Hennecke, "Lipreading by neural networks: Visual preprocessing, learning and sensory integration," in: J. D. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in the Neural Information Processing Systems 6*, Morgan Kaufmann Publishers, San Francisco, CA, 1027-1034, 1994.

[73] X. Xie, R. Sudhakar, and H. Zhuang, "On improving eye feature extraction using deformable templates," *Pattern Recognition*, 27(6), 791-799, 1994.

[74] Y. Yacoob, and L. S. Davis, "Labeling of human face components from range data," *CVGIP: Image Understanding*, 60(2), 168-178, 1994.

[75] B. P. Yuhas, M. H. Goldstein, T. J. Sejnowski, and R. E. Jenkins, "Neural network models of sensory integration for improved vowel recognition," *Proceedings of the IEEE*, 78(10), 1658-1668, 1990.

[76] A. L. Yuille, "Deformable templates for face recognition," *Journal of Cognitive Neuroscience*, 3(1), 59-70, 1991.

[77] A. L. Yuille, D. S. Cohen, and P. W. Hallinan, "Facial feature extraction by deformable templates," Technical Report 88-2, Harvard Robotics Laboratory, 1988.