# Face Translation: An Image-Based Approach to a Multi-Modal Communication Agent

Diploma Thesis

**Interactive Systems Laboratories**

Carnegie Mellon University, Pittsburgh, PA, USA

Universität Karlsruhe, Germany, Fakultät für Informatik

Max Ritter

Supervisors:
Professor Dr. Alex Waibel
Dr. Jie Yang
Dipl. Inform. Rainer Stiefelhagen

July 1999

Hiermit erkläre ich, daß ich diese Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Pittsburgh, den 15.7.1999

Max Ritter

# Zusammenfassung

Die vorliegende Arbeit stellt einen bildbasierten Ansatz für die Implementierung eines multi-modalen Kommunikations-Agenten vor. Dieser sogenannte Avatar basiert auf einem 2D Modell für den Kopf und auf Aufnahmen von realen, menschlichen Gesichtern. Das vorgestellte System ist in der Lage, eine audio-visuelle Ausgabe mit Hilfe einer vorher gespeicherten Datenbank und eines Eingabetexts zu erzeugen. Es benötigt minimale Trainingszeit und kann für mehrere Sprachen eingesetzt werden. Ein neues Gesicht kann mit Hilfe eines einfach zu bedienenden, GUI basierten Dialoges zu der Datenbank des Systems hinzugefügt werden. Dieser Vorgang beansprucht weniger als zehn Minuten. Das System kann beliebigen Text synthetisieren und sprechen, wobei die Lippenbewegungen mit der akustischen Sprachausgabe synchronisiert werden. Zudem kann das erzeugte Gesicht in die Richtung derjenigen Person schauen, die angesprochen werden soll. Das System bietet eine Client/Server Schnittstelle an, die eine Fernsteuerung der wesentlichen Funktionen ermöglicht und zudem Zugriff auf alle Daten erlaubt, die notwendig sind, um den Avatar auf einem anderen Rechner darzustellen.

Alle Komponenten dieses Systems werden ausführlich beschrieben. Außerdem werden verschiedene Aspekte bezüglich der Implementierung angesprochen. Der praktische Nutzen des vorgestellten Systems wird anhand von zwei Anwendungen demonstriert. Die erste Anwendung ist Face Translation (Gesichts-Übersetzung). Im Rahmen des Consortium for Speech Translation Advanced Research (C-STAR) Projektes spielt der Avatar die Rolle eines multi-modalen Übersetzungs-Agenten, welcher die audio-visuelle Ausgabe fuer ein Sprachübersetzungssystem liefert. Dies ermöglicht es zwei Leuten, die nicht dieselbe Sprache sprechen, per Telekonferenz zu kommunizieren. Beide können in ihrer Muttersprache sprechen, und beide sehen und hören ihren Gesprächspartner, bzw. den Avatar mit dem Gesicht des Gesprächspartners, in einer Sprache sprechen, die sie verstehen. Die zweite Anwendung ist multi-modales Chatten. Hierbei können mehrere Leute über das Internet miteinander kommunizieren und dabei synthetisierte Gesichter der anderen Teilnehmer sehen. Am Ende dieser Arbeit werden zukünftige Verbesserungen und Erweiterungen diskutiert.

# Abstract

This thesis presents an image-based approach to develop a multi-modal communication agent. This so-called avatar uses a 2D head model and recorded images of real human faces. The system is capable of generating audio-visual output from a pre-stored image database and text input. It requires a minimum training time, and can be applied to multiple languages. A new face can be added to the system's database through an easy-to-use GUI-based dialog, which takes less than ten minutes. Arbitrary text can be synthesized and spoken by the system with lip movements that are synchronized to the audio speech signal. The synthesized eye gaze is directed towards the message target. The system offers a client/server interface that allows remote control of the main functions and retrieval of all data necessary to show the synthesized face on a remote machine.

We describe each component of the system in detail. We also address various implementation issues and procedures. The feasibility of the presented system is demonstrated by two applications. The first is Face Translation, where the avatar serves as a multi-modal translation agent, providing audio-visual output for a speech-to-speech translation system in the context of the Consortium for Speech Translation Advanced Research (C-STAR) project. This makes it possible for two people, who do not speak the same language, to communicate over a teleconferencing system. Both can speak in their native language and both can see and hear the other person, i.e., the avatar with the other person's face, talking in a language they understand. The second application is multi-modal chat, where people can chat with each other over the Internet, seeing the other participants' synthesized faces. We conclude the thesis by discussing future work.

# Acknowledgements

This thesis was developed within the Consortium for Speech Translation Advanced Research (C-STAR) project and it would not have been possible in the presented form without support from a number of people. First of all, I would like to thank Professor Alex Waibel for the opportunity to conduct my research at the Interactive Systems Laboratories at Carnegie Mellon University in Pittsburgh. He was also the source of many ideas that formed this project. I would also like to thank: my supervisors Jie Yang and Rainer Stiefelhagen for their valuable advice; Uwe Meier and Ralph Gross for help with programming issues and other helpful discussions; Hua Yu for help with the Janus speech recognizer; Monika Woszczyna, Eric Carraux and Yue Pan for their cooperation; Ann Kim for helping me to improve my English and for her support; Naomi Shabot for proofreading my thesis and for giving me new insights into the English language. Last but not least, I want to thank my parents for their generous support during my whole education and especially during my time in Pittsburgh.

# Contents


**1 Introduction** **1**

1.1 Motivation . . . . . 1

1.2 Objective of this Thesis . . . . . 5

1.3 Related Work . . . . . 6

1.4 Overview . . . . . 7

**2 Background** **9**

2.1 Phonemes and Visemes . . . . . 9

2.2 Real Faces vs. Synthetic Faces . . . . . 11

2.3 2D vs. 3D . . . . . 12

2.4 Input Format . . . . . 13

2.5 Speech Synthesis . . . . . 14

2.6 Synchronization of Video and Audio . . . . . 14

2.7 Mouth Movement . . . . . 15

2.8 Eye Movement . . . . . 16

2.9 Head Movement . . . . . 16

2.10 Facial Expressions . . . . . 17

2.11 Detection and Tracking of Faces / Facial Features . . . . . 17

2.12 Low Bandwidth Communication . . . . . 18

2.13 Response Time . . . . . 18

**3 MaxAvatar - A Prototype System** **20**

3.1 System Overview . . . . . 21

3.1.1 Phoneme / Viseme Handling . . . . . 22

3.1.2 The Image Database . . . . . 25

3.1.3 Face Synthesis . . . . . 28

3.1.4 Movies . . . . . 35

3.1.5 Recording and Playing of Audio-Visual Data . . . . . 36

3.1.6 Face and Facial Feature Detection / Tracking . . . . . 37

3.1.7 The Client/Server Interface . . . . . 47

3.1.8 Other Modules . . . . . 49

3.2 User Registration . . . . . 50

3.2.1 Automatic Mode . . . . . 50

3.2.2 Online Interactive Mode . . . . . 52

3.2.3 Offline Interactive Mode . . . . . 55

3.3 Conversation Mode . . . . . 57

3.4 Options . . . . . 58

3.5 Compatibility . . . . . 58

3.6 Additional Experiments . . . . . 59

**4 Applications** **60**

4.1 Face Translation . . . . . 60

4.1.1 System Overview . . . . . 61

4.1.2 C-STAR II System . . . . . 63

4.1.3 Janus III . . . . . 64

4.1.4 Microphone Direction Server . . . . . 65

4.1.5 Festival . . . . . 66

4.1.6 MaxAvatar . . . . . 67

4.2 Web Avatar . . . . . 68

4.2.1 System Overview . . . . . 68

4.2.2 MaxAvatar as a Server . . . . . 69
4.2.3 Possible Extensions . . . . . 72

**5 Summary and Future Work** **73**
5.1 Summary . . . . . 73
5.2 Future Work . . . . . 74

# Chapter 1

# Introduction

## 1.1 Motivation

Communication plays a very important role in modern society. Almost every part of our life is touched by it and is thus changed by the tremendous development of communication technologies, e.g., telephone and e-mail, which allow instant communication with people all around the world. Be it companies, armies, or individuals, nowadays it is almost impossible to succeed without making efficient use of communication. And as a consequence of the greater possibilities to communicate, there are higher expectations regarding communication. This makes people look for (and find) even more efficient ways to communicate. Unfortunately, as soon as these more efficient techniques are established, the expectations increase again. Thus, besides the commonly quoted information overload, modern technology has also led to a communication overload. A few decades ago, the average person got and wrote a letter once every few weeks and used the phone infrequently or did not even own one. Nowadays, more and more people have to cope with dozens of e-mails and several phone calls each day.

The establishment of avatars will not reduce the amount of communication, but it will offer ways to communicate that are more natural for human beings than the current systems like telephone or e-mail that use only speech or only text.

In the Hindu religion, an avatar is an incarnation of a deity; hence, an embodiment or manifestation of an idea or greater reality. In the computer

terminology, an avatar is a representation of a real human being in the computer. Most commonly, it is a graphical representation, ranging from an icon over animals or fantasy creatures to realistic looking, animated images of the human being itself.

In this work, the avatar takes the form of a talking-head, and the terms avatar and talking-head will be used interchangeably.

### History of Communication

A brief review of the development of human-to-human communication will reveal why a talking-head or an avatar can help to make modern communication more natural.

The very first way to communicate was body language, including mimic and gestures, maybe combined with the utterance of some primitive noises. Much later language and speech evolved, which allowed for the conveyance of more abstract information. Again much later, written language was developed several thousand years ago. The next milestone was only reached in 1875, not even 130 years ago, when Alexander G. Bell and his assistant Thomas A. Watson built the first telephone. The second half of the twentieth century saw the rapid progress of computer technology and its application in almost every area, including communication.

From the beginning of human history until the last century, a conversation always meant a face-to-face conversation with a physically present person. The large amount of non-verbally conveyed information is lost during telephone conversations. Written language contains even less information, because it lacks intonation, hesitations, and other cues about the state of mind of the other person.

Modern technology offered possibilities for convenient global communication, and people adapted their way of communicating to the available systems. Now that these technologies are getting more and more advanced, efforts are made to adapt the communication systems to the wishes and needs of human beings and not vice versa. Steps into this direction are the use of teleconferencing systems to enhance the telephone with visual information, the use of speech input and output for computers instead of purely text-based systems, and the use of avatars to provide visual information in computer-supported communication.

### Psychology and Perception

Research about human psychology and perception confirms the value of seeing the other person's face, or at least a face representing the other person, during a conversation.

From the first year of life, the sight of a face is a powerful stimulus for humans, with or without speech [McG98]. It helps the listener to pay attention to what is being said. Furthermore, speech perception is facilitated by sight of the speaker. Humans automatically and usually subconsciously use lipreading to improve their understanding, an ability learned during infanthood. Later, the influence of visible lip-movements upon the perception of speech increases so much that under certain cirumstances, wrong lip images can result in understanding something different from what has actually been said. This is called the McGurk effect [MM76].

According to research results of Clifford Nass at Stanford University, computer personalities are "psychologically real" to users; i.e., users tend to respond psychologically to computer personalities in the same way that they respond to human personalities [MN96]. And there is evidence that computer personalities can be easily created using a minimal set of cues [NMF+95]. Also, seeing a self-image evokes more responsibility and results are perceived to be more valid and objective than when they are presented by somebody else's face [NKL98]. Research by Chris Davis and Jeesun Kim [DK98] also suggests that the perception and memory of foreign speech sounds can be enhanced by the addition of the speaker's face.

### Applications

There are a variety of applications for computer animated avatars. Their goals are to increase the comfort of a human using a computer, or to allow communication that would not be possible without the help of a computer, and still have a feeling similar or ideally equal to communicating with another human being. The most natural way of communicating is a face-to-face conversation. Nowadays, a great deal of communication is done without seeing the other person's face or even without hearing the other person's voice, via telephone, answering machines, e-mail, or other forms of text. The use of an avatar to provide visual and acoustical speech output makes communication more natural.

Avatars can be useful in human-computer interaction as well as in human-human interaction. In human-computer interaction, they can serve to give the computer a certain degree of a personality, a visible representation of the program that talks to the user. Right now, the most common use of such an interface would be to present a message or some other relevant information to the user, or to guide the user through a complex process. An example for this is the Microsoft paperclip. Although it doesn't talk, it makes the interaction more personal by having a face and using facial expressions.

In human-human interaction, there are many possible applications for avatars. They can be used for teleconferences, when only low bandwidth is available (see section 2.12). They can also be used when one wants to be seen by another person, but with a different look. It is possible to appear as a cartoon character or as another person. Maybe somebody from Europe wants to talk to somebody from Asia and therefore wants to look a little bit more Asian. In this case, the European could select his/her own face, but have the eyes replaced by Asian eyes. Or somebody gets a business call while having breakfast in his/her pyjamas. Then an existing, more business-like recording could be used.

Another application is "Face Translation." It facilitates communication between people who do not speak the same language. Speech and lip movements are translated into a language that the other one can understand. As a result, both sides can hear and see the other one talk in a language they know. "Face Translation" will be described in more detail in section 4.1.

During the last few years, chatting became a very popular way to communicate with a lot of people all around the world through the Internet. Instead of just seeing what other people type, avatars would allow a user to see faces speak. Even if this is not always desired, because a lot of people like to stay anonymous on the Internet, one can still choose to be represented by another face. An example implementation of a "Web Avatar" will be presented in section 4.2.

Another area of applications of computer animated characters is entertainment. Whole movies have been created by only using computers, e.g., "Antz" and "A Bugs Life." It is almost normal to see completely computer animated characters in a movie. TV shows also use this technique to make fun of politicians, making them say anything the writers want.

## 1.2 Objective of this Thesis

The objective of this thesis was to design and implement a front-to-end avatar system that can be integrated into the C-STAR system (see section 4.1.2) to provide audio-visual output for the speech-to-speech translation results. The avatar should consist of a face that is able to perform controlled eye and lip movements, where the lip movements should be synchronized to a speech signal. This face would represent the user of the speech-to-speech translation system. It should also be possible to use the avatar in low bandwidth teleconferencing applications.

Pursuing these goals, several design decisions were made (see chapter 2 for a more detailed description of the involved issues and how they can be handled). The most important decision was to use realistic-looking images of the face of the person that shall be represented, based on previously recorded images. Since the intention was to have a system that can easily be used by many users, this meant that a quick and convenient way to register a new user's face had to be implemented (see section 3.2).

For a realistic face and quick user registration, a 2D model of the head (see section 2.3) and animating the eyes and the lips by pasting in different recorded images (see sections 2.7 and 2.8) seemed to be the most feasible approaches. Other approaches would require either tedious manual labeling or advanced facial feature analysis techniques that were not available. Image processing techniques are used to calculate transition images between different lip images (see section 3.1.3).

Low bandwidth communication is supported in two ways. First, the images are relatively small and have only to be sent once. Afterwards, it is sufficient to send codes for the images and timing information. Second, the system is able to make the best out of any number of available eye or lip images. This makes it possible to send a minimum of images, depending on the desired output quality. The ability to handle sparse data also makes user registration very flexible.

Last but not least, a modular program structure was chosen. This facilitates future improvements and enhancements to the system.

## 1.3 Related Work

This section presents a few examples of work done by other people in the same field.

### Video Rewrite

Video Rewrite by Christoph Bregler [BCS97] creates new videos from existing footage. In the new video, persons can speak arbitrary sentences that they did not speak in the original video, similar to the famous scene with John F. Kennedy in the movie "Forest Gump."

To model a new person it is necessary to manually label 26 images. Then the training video sequence is automatically analyzed by an HMM speech recognizer to determine the phonemes and by a feature detection system using eigenpoints. The eigenpoint models are trained with the manually labeled images. Video Rewrite uses triphones to model the lip transitions. All the available triphone images are extracted from the training sequences. In the paper mentioned above, the example training sequence was 8 minutes long and provided approximately 1700 different triphones. The new soundtrack is not synthesized, but taken from an original recording. After the HMM speech recognizer has delivered phoneme and timing information for the new sentence, the best matching available triphone images are stitched into the recorded video sequence.

Although Video Rewrite can generate natural looking talking faces, it requires great effort and a long time to create such a huge triphone database. It currently only animates the lips and takes the other facial features from the original video. Moreover, the system only supports English.

In this thesis, we emphasize minimum training time and reasonable size of the database, so that a user can quickly create a database and send it through the Internet. In addition, in a language translation task, triphones might not be available in other languages. Therefore, we have to employ a different approach to handle the lip synthesis. Furthermore, besides the lip synthesis, we also synthesize eye gaze to achieve some social behavior for the synthesized face.

### Baldi

Baldi is a synthesized 3D talking-head, developed by Michael Cohen and Dominic Massaro [CM90, GGMCB94]. It uses an adaptation of Parke's original parametric model for human faces [Par75] to produce visual output that matches speech. As in MaxAvatar (see chapter 3), the speech is generated by a text-to-speech synthesizer. Since one of the goals of Baldi is to help hearing-impaired children learning to speak, the proper animation of the lips received much attention. The lip movement and coarticulation results look clearly smoother than the current results of our prototype system (see chapter 3). The synthetic head can be rotated, look in arbitrary directions, express several different emotions (see section 2.10), and speak arbitrary sentences. It is also possible to choose one of a list of available real faces that is then morphed over the 3D grid model and animated.

### Brooke and Scott

N. M. Brooke and S. D. Scott presented a two- and three-dimensional approach for audio-visual speech synthesis [BS98b] (see also section 2.3). As in our prototype system (see chapter 3), they use a data-driven approach for synthesis. To add a new face, about 20 minutes of speech have to be recorded, phonetically transcribed, and then manually processed into phonetic segments. After applying PCA methods to the images, they were used to train HMMs. These HMMs model both visual and acoustic data for each triphone. A text-to-speech synthesizer is used for phonetic transcription of the input text. The previously trained HMMs and PCA techniques allow for the creation of appropriate output images. The resulting 2D lip images are then projected onto a 3D wireframe head, which can be rotated arbitrarily.

The HMM approach seems to demand far more training data than can be obtained in our case. The 2D-3D approach, however, might be a good way to enhance our prototype system with the option of head rotation.

## 1.4 Overview

Chapter 2 describes the background needed for building an avatar. Chapter 3 presents the prototype system MaxAvatar that we have developed in the framework of this thesis. It discusses in detail all the components, their

functionalities, and their interaction. Chapter 4 introduces two existing applications in which the prototype system is used. The first application is Face Translation (see section 4.1), where the avatar is used in a spech-to-speech translation system, so that a person speaking in one language can be seen speaking the same sentence in another language, e.g., on the other side of a teleconferencing connection. The second application is a Web Avatar (see section 4.2), which demonstrates the use of an avatar for multi-modal chat over the Internet. Finally, Chapter 5 summarizes the work presented in this thesis and future work.

# Chapter 2

# Background

This section describes issues and problems related to building an avatar, with possible solutions for each of these issues. In most cases every approach has its own advantages and disadvantages, the choice depending on several constraints; e.g., the application or the context in which the avatar will be used, who is going to use it, or which hardware is available. Some decisions about the look of the avatar are simply a matter of taste.

## 2.1 Phonemes and Visemes

Phonemes and visemes provide the foundation to move the mouth of an avatar in a way that it looks as if it is speaking.

Phonemes are defined as the smallest distinguishable units of spoken language. Every spoken sentence can be written as a sequence of phonemes. Speech recognizers use dictionaries that contain all the words of their vocabulary and the corresponding phonemic transcriptions. In this work we use the Janus speech recognition and translation toolkit that was developed in the Interactive Systems Laboratories (see also section 4.1.3).

Since different languages use different sounds, different sets of phonemes are necessary to model them. Table 2.1 shows the set of phonemes used by Janus (see section 4.1.3) for the English language. Table 2.2 shows the corresponding set for the German language. The English "TH" phoneme does not exist in German. And the German umlauts do not exist in English.

| SIL | AA | AE | AH | AO | AW | AX |
|---|---|---|---|---|---|---|
| AXR | AY | B | CH | D | DH | DX |
| EH | ER | EY | F | G | HH | IH |
| IX | IY | JH | K | L | M | N |
| NG | OW | P | R | S | SH | T |
| TH | UH | UW | V | W | X | Y |
| Z | | | | | | |

Table 2.1: English phonemes used by Janus

| SIL | A | AR | AEH | AEHR | AH | AHR |
|---|---|---|---|---|---|---|
| AI | AU | B | CH | X | D | E |
| E2 | EH | EHR | ER | ER2 | EU | F |
| G | H | I | IR | IE | IHR | J |
| K | L | M | N | NG | O | OR |
| OE | OEH | ANG | OH | OHR | P | R |
| S | SCH | T | TS | TSCH | U | UR |
| UE | UEH | UEHR | UH | UHR | V | Z |

Table 2.2: German phonemes used by Janus

As phonemes are the smallest distinguishable units of spoken language, visemes are defined as the smallest distinguishable units of visible language, i.e., the different lip positions which can be seen when somebody talks. Visemes can be used to synthesize lip movement. If one has image information for every viseme, it is possible to simulate the lip movement for every utterance by creating an image sequence showing the right visemes in the right order.

Visemes are also language-dependent. For example, there is no "TH" viseme in the German language, which is not surprising, since the corresponding sound also does not exist. Different people have defined different sets of visemes even for the same language. These sets vary in the number of defined visemes. Table 2.3 shows the set of visemes used in Video Rewrite [BCS97]. The system presented in this thesis (see section 3.1) uses a slightly different approach, where visemes are not declared explicitly.

There are several ways to generate a list of visemes. One way is to do it

| Consonant classes | Vowel classes | Silence class |
|---|---|---|
| CH, JH, SH, ZH | EH | SIL |
| K, G, N, L | EY | |
| T, D, S, Z | ER | |
| P, B, M | UH | |
| F, V | AA | |
| TH, DH | AO | |
| W, R | AW | |
| HH | AY | |
| Y | UW | |
| NG | OW | |
| | OY | |
| | IY | |
| | IH | |
| | AE | |
| | AH | |

Table 2.3: Visemes used by Video Rewrite

manually; i.e., judging subjectively how similar the lip positions for different sound look and grouping those that look most similar to each other into the same viseme class. There are different methods of determining viseme classes automatically [Sti96].

There is no one-to-one mapping between phoneme and viseme boundaries. However, phoneme boundaries are currently commonly used as an approximation for viseme boundaries.

## 2.2 Real Faces vs. Synthetic Faces

One design issue for avatars is whether it should have a human face that looks as realistic as possible or a face that has been created synthetically and looks non-human or like a cartoon. The decision about this depends on the application and on the group of people who will use it. Who wants to be represented for which purpose? Should it be funny, neat, serious, or personal? It also depends on what will be feasible for the user. If the user is

not going to have a camera to record his/her face, it is not possible to take his/her face to represent him/her.

It seems that most of the recent research effort is directed towards the creation and animation of very realistic looking human faces. Although the movie industrie will definitely also be interested in using completely synthetic but realistic looking characters, it is currently more focussed on computer-animated cartoons, robots, animals, and aliens. This can be seen in movies like "A Bugs Life", "Antz", and "Star Wars Episode I". Current computer applications also usually use synthetic faces instead of trying to model a real human face. An example is the Microsoft Office paperclip that assists users in solving problems.

## 2.3 2D vs. 3D

Regarding implementation, it makes a big difference whether the avatar is designed as a 2D or a 3D model. The 2D approach [BCS97] takes regular 2D images without any information about depth and manipulates them. In order to generate views from a different angle, images have to be taken from that angle, or some sophisticated image processing techniques have to be applied to the original view. The same is true if the avatar shall show different facial expressions. Either images of these facial expressions have to be taken or the image has to be manipulated, which requires detailed knowledge about the face edge and facial feature coordinates. On the other hand, it is relatively easy to add another face when using the 2D model, especially when only a few facial features shall be animated. In the prototype system, which will be presented in section 3.1, we use a 2D model and animate only the eyes and the lips. The fastest way to add the face of a new user to this system requires only one picture of his/her face to be taken. A facial feature recognizer can automatically locate the eyes and the lips. Eye and lip images for animation can be taken from an existing face or, with a little more effort, also from the new user. Another advantage of this approach is that it is relatively easy to handle and that it does not require much computing power.

The 3D approach [HL98, KYVB98, GUAT98, Mor98] offers different advantages and disadvantages. A 3D grid model is used to model the head shape and the facial feature locations. Having the coordinates of these grid points allows the program to turn and move the head arbitrarily. The more points

the grid contains, the better the quality of the head. However, more points also result in a higher computing time for each image. Thus, a compromise has to be made between quality and speed. Facial expressions can also be modeled by moving the grid points in certain ways. The lip corners, for example, would move outwards and up to express happiness, and they would move down for a sad face. The drawback is that the face itself either has to be synthetic or a considerable amount of work is necessary to register a real face. Every point of the grid has to be mapped to the corresponding point of the face image. A very sophisticated facial feature recognizer is necessary to do this automatically. So it would most likely require manual work. Furthermore, facial expressions are different from person to person. Of course, generic movements for the grid points can be used to simulate facial expressions. However, to make the avatar smile like the real person does will require an individual movement function. Maybe this function could be learned from an example image sequence, which shows the person make the desired facial expression. Head rotation for an arbitrary angle is easier to generate with a 3D approach than with a 2D approach. However, because of occlusion there are still several images of the head necessary, taken from different angles in order to show the texture of the real head from all perspectives.

A compromise between the 2D and the 3D approach [BS98a, BS98b], which could be called a 2.5D approach, might work as follows. Starting with a 2D image, a small number of feature locations can be used as a small grid model. Head rotation can be realized by taking several 2D images of the head from different angles. Angles for which no image exists can be interpolated from the two closest existing pictures in each direction. The new feature coordinates after rotation can be calculated through the grid model. An image of the features, e.g., of the desired lip shape, can be rotated by using image processing techniques and can then be pasted at the new position.

## 2.4 Input Format

Preprocessing techniques used by an animation system depend on the input format. If the user just types the plain text, which is to be spoken, no preprocessing is needed. But maybe the text has to be sent over the Internet and the user wants it to be encrypted for privacy reasons. In this case, a decryption module has to make the text intelligible again. If the goal is

a normal spoken conversation, the user will also want to speak instead of typing the text. Hence, a speech recognizer is required. Finally, if the avatar serves as a translation agent, as in the system presented in section 4.1, a translation module is needed in addition to the speech recognizer. In many cases, these additional modules will be existing products, using their own format and protocol.

## 2.5 Speech Synthesis

The focus of this thesis is mainly on creating synthesized lip movement for existing synthesized speech, providing a visual output interface for another application (see 4.1), and integrating the avatar into this application. Since we get text from the application, we need a text-to-speech system to generate the synthesized speech. Festival [BTC98] (see also section 4.1.5) is a general multi-lingual speech synthesis system developed at the Centre for Speech Technology Research that we use as a text-to-speech synthesizer. Festival also provides a list of phonemes and the corresponding time alignment, which is all the information we need to produce lip movement that matches the acoustic speech output.

## 2.6 Synchronization of Video and Audio

For our prototype system (see section 3.1), we employ synchronization of video and audio in two ways.

When a new user wants to use the automatic mode for registration, he is asked to speak a sentence, while his/her face and speech are recorded. Time stamps are taken for the recorded images. The spoken utterance is then processed by a speech recognizer, which returns the list of phonemes for this utterance in addition to the corresponding time alignment. Comparing the image time stamps and the phoneme time alignment, one can see which image contains which phoneme and which viseme. This viseme can be extracted and stored in a database to be used for video synthesis later (see also sections 3.1.2 and 3.2.1).

Video and audio also have to be synchronized when the avatar speaks. In our prototype system, this goal is reached as follows: first, we get the synthesized

speech audio file plus the phoneme list and the corresponding time alignment from Festival; then, we use the phoneme and timing information to create appropriate images with appropriate time stamps. To make the avatar say the sentence, audio replay is started and the synthesized images are displayed according to their time stamps (see also section 3.1.4).

## 2.7 Mouth Movement

There are several ways to produce mouth movement. If a 3D model is used for the avatar, the straight-forward approach is to make the mouth change its shape by moving the grid points in an appropriate way and then reapplying the face texture.

If a 2D approach is used with one base face image, one way to make the mouth move is to manipulate the base image without the use of other image information. If the location and the shape of the lips are known, it is possible to distort them to create other lip shapes.

The advantage of the two approaches mentioned above is that they only need a minimum of recorded images as long as a detailed analysis of the used image and sophisticated image processing techniques are available.

A drawback of these approaches is that there is no image information available for the inside of the mouth. An open mouth has either to be filled with black or with artificially created images of the teeth and the tongue. The mouth movement will also not take into account individual properties of the used face.

Another 2D approach also takes only one base face image, but the mouth is animated by pasting recorded images of different lip images (see also section 2.1) into this face. This approach has the advantage that the inside of the mouth will contain realistic information, because it has been recorded. The lip images also reflect the individual lip shapes of the corresponding person. On the other hand, it is not as straight-forward as in the other two approaches how transitions between different lip images can be generated. Also, more images have to be recorded to add a new face.

In [BS98a], an approach is presented that can be used to drive 2D, 3D, or image-based animations. HMMs are trained to learn the relation between facial animation and vocal cues. Instead of phonemes or visemes, a special

representation is used that includes dynamic and contextual information. This allows to learn and reproduce realistic mouth movement including co-articulation effects.

## 2.8 Eye Movement

The approaches for mouth movement described in section 2.7 can also be applied when creating eye movement. While it is possible to generate images of an open mouth from one of a closed one, this cannot be done with eye images. To be more precise, while it is acceptable to fill the mouth opening with black, the eye opening has to be filled with a pupil and an iris on a white background to make the eye look in a certain direction. This is feasible, but it tends to look artificial.

This problem can be avoided by starting with an open eye and generating the closed eye image from there. Manipulation of the eyegaze is possible if the exact shape of the eyes and the location of the iris and the pupil are known.

The use of several images of the eyes looking in different directions and being closed completely avoids the problem of synthesizing them, unless transition images must be calculated.

## 2.9 Head Movement

The most interesting, but unfortunately also the most complicated, head movement for an avatar is rotation. Small head rotations can be used to make the head look more realistic because in a real conversation, nobody ever keeps the head completely still. Turning the head left or right also helps to address a certain person. Furthermore, it can be used for head gestures like nodding or shaking the head.

As mentioned in section 2.3, implementing head rotation requires different techniques depending on the underlying model of the head. In the case of a 3D model, all that has to be done is to calculate the view of the head points from a different perspective. A simple 2D approach with one base image does not allow head rotation. This would require the use of several base images

taken from different angles and switching to a 2.5D approach (as described in section 2.3).

## 2.10 Facial Expressions

It is a well-known fact that during face-to-face conversation a lot of information is conveyed non-verbally. An advanced avatar should be capable of facial expressions, gestures, and other types of body language. This makes the conversation more realistic and more interesting. It gives information about the mood of the person on the other side, which might not be transmitted otherwise, especially when a synthesized voice is used. Of course, there may be situations where transmission of this kind of information is not desired or not appropriate.

There are different ways to classify facial expressions. Baldi [MCB98], for example, uses five different basic expressions: happiness, sadness, anger, disgust, and fear. These basic expressions can be combined to express a variety of emotions.

Synthesis of facial expressions follows technically the same rules as synthesis of mouth movement (see section 2.7). However, it involves the whole face instead of only a limited region. It is difficult to implement such a complex process with a 2D model of the head. The structure of the face has to be known and modified so exactly at so many places that it probably leads to an implementation similar to a 3D approach.

A related problem is the analysis of facial expressions. In order to transmit this kind of information and display it with the avatar, it first has to be obtained somehow. To do this automatically is a complex and interesting field of research in itself.

## 2.11 Detection and Tracking of Faces / Facial Features

Face and facial feature detection play an important role in building a real face avatar. No matter which approach is used for the underlying head model, adding a new face and animating it requires exact knowledge of the facial

feature coordinates. Wrong coordinates can lead to disturbing results, and manual labeling of the features is usually very tedious. Thus, a reliable and precise feature detector is needed.

Individual differences between faces, different lighting conditions, and different recording equipment make detection of faces and facial features for the general case a difficult problem. Current approaches evaluate one or more of the criteria motion, color, and shape [YSMW98, SMY97a] (see also the approach implemented in the prototype system presented in section 3.1.6), or they employ neural networks [SMY97b] or hidden markov models [YSMW98, SMY97a] to deliver the desired information.

## 2.12 Low Bandwidth Communication

Although bandwidths of future networks might make it unnecessary, it is still an advantage to reduce the amount of data that has to be transmitted for communication. Avatars can be used for teleconferencing applications when only low bandwidth is available, because image information has to be transmitted only once. Afterwards, only a small amount of information has to be sent. In the case of a 3D model, only the new grid point coordinates need to be sent, or only a command to turn the head or display a different lip shape, etc. In the case of a 2D model and image manipulation, it can be enough to send information about the new lip or eye shape. When using a 2D model with pasting in of feature images, it is sufficient to send all the images once and only a code for the appropriate feature image later.

In addition to the images, it might be necessary to transmit an audio file containing the synthesized output utterance. These files are usually significantly smaller than 100 kB.

## 2.13 Response Time

If the avatar is used as a communication agent, the response time must not be too long. A conversation becomes tiring to the participants when there is always more than a few seconds of silence between the utterances. The largest amounts of time are consumed in the following stages of the communication:

- *Generating the text.* Some preprocessing may be necessary to generate the text that will be synthesized by the avatar. An example is the speech-to-speech translation application described in section 4.1, where natural speech first has to be recognized and translated.
- *Speech synthesis.* The audio file containing the output utterance has to be created.
- *Image processing.* To make the avatar speak a sentence, a sequence of images has to be synthesized. Depending on the used algorithms, this can take a considerable amount of time. Precalculation techniques (see section 3.1.3) can help to reduce this delay.
- *Data transfer.* As mentioned in section 2.12, relatively little data has to be sent. However, for a slow connection the delay can still be several seconds.

It can be expected that faster processors and an increase in bandwidth will diminish these issues.

Following are example times for the system presented in chapter 3 and the applications presented in chapter 4. Running on a 233 MHz Pentium II PC with Windows 98, Festival needs approximately one to four seconds to synthesize an utterance, depending on its length (one second for one word, four seconds for 25 words). Video synthesis on the same machine and for the same utterances takes between one and 20 seconds. However, when using precalculated images the video synthesis takes no noticeable time, no matter how long the utterance is.

# Chapter 3

# MaxAvatar - A Prototype System

MaxAvatar is a prototype for a facial animation system. It is designed to create multi-modal communication agents for different applications. We are currently using MaxAvatar in the context of a travel planning scenario, which involves speech recognition and translation (see chapter 4.1 for more details). Building an avatar is a huge task, touching many areas of research at the same time. Audio and video signals have to be recorded, processed, stored, synchronized, and replayed. Phonemes and visemes have to be retrieved, internally represented, and handled. Image processing techniques are necessary to produce reasonable output results. Efficient inter-program communication is crucial. Since we wanted an avatar that can quickly be adapted to faces of different users, we needed an easy-to-use user interface. In addition, we had to find a structure for the face database. Facial feature recognition and classification is necessary to make the process of registering a new face automatic, or at least less tedious.

In the context of this thesis, many of these issues could only be implemented in a prototypical way. However, the system and its modules have been designed in a way that should make it relatively easy to work on each topic separately as desired, and to add or exchange modules as soon as better ones are available. The main goals during development were the following:

- to build a fully working avatar that is able to speak a given text and to move its eyes to address a certain person;

- to receive the text from another application over the network;
- to allow a new user to register his/her face with acceptable effort;
- to provide a possibility to use the avatar in low-bandwidth teleconferencing;
- and to make the best out of whatever amount of image data is available.

These goals have been reached and demonstrated in the protoytpe system.

## 3.1 System Overview

MaxAvatar has been developed for Microsoft Windows and runs under Windows 95/98/NT. The first version was written completely in C, mainly because of compatibility reasons. However, as the program grew more in size and complexity and as we wanted to replace the text interface with a graphical user interface, it was converted to C++ and divided into several modules, using an object oriented approach. We used the Microsoft Foundation Classes (MFC) to add a graphical user interface. Communication with other programs is realized with socket connections. In some cases, we wrote intermediate scripts in Tcl or Perl to keep the communication more flexible.

Figure 3.1 shows the structure of MaxAvatar. The whole system consists of three parts. MaxAvatarGUI_MFC is the main program, which provides the main menu for the user. It uses the other modules and coordinates their work. The library MaxAvatarLib_MFC contains all classes (except the main program) which use MFC functionality. MaClientServer (3.1.7) uses MFC socket functions to communicate with other programs, the registration modules (3.2) allow to add new faces to the face database, and MaRecPlay (3.1.5) makes it possible to record and play synchronized audio and video signals.

The basic avatar functionality is given through the classes of the library MaxAvatarLib. They are used by the main program and by MaxAvatarLib_MFC. They also use one class of MaxAvatarLib_MFC, namely MaIO (3.1.8), which provides input and output functionality through a generic interface.

MaxAvatarLib itself can be separated further in two parts. One of them consists of classes that facilitate programming: MaImage for image handling, MaParam to store and access program parameters, and MaUtil for utility functions like string operations or secure file handling. They are described in more detail in section 3.1.8. The other part builds on the first one and consists of modules with specific responsibilities during creation of the avatar. MaPhonVis (3.1.1) handles phonemes and visemes, MaDataBase (3.1.2) manages the image databases, MaSynth (3.1.3) synthesizes single faces and animates the avatar, MaMovie (3.1.4) stores recorded or synthesized image sequences, and MaImageReg (3.1.6) finds and tracks facial features.

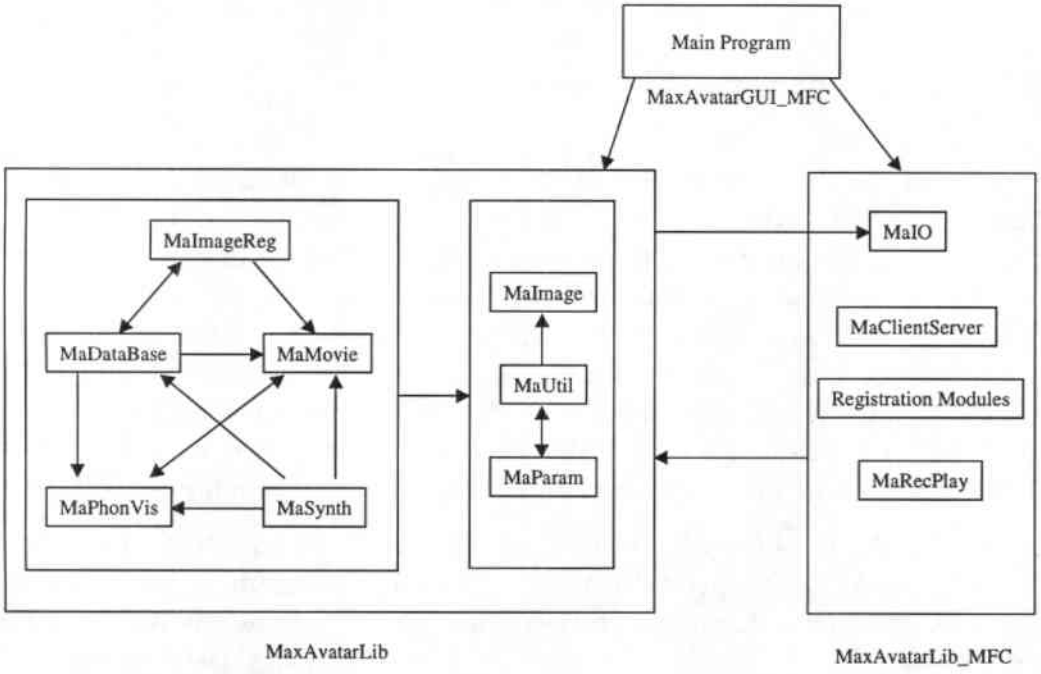


Figure 3.1: The modules of MaxAvatar and their relationships.

### 3.1.1 Phoneme / Viseme Handling

The class MaPhonVis offers functionality to handle phonemes and visemes for different languages and different naming conventions. Currently, MaPhonVis contains information about the English and the German phoneme sets of

Janus (see section 2.1), and of the English phoneme set of Festival. It also provides a conversion function between Janus and Festival phonemes.

Visemes are not handled explicitly in MaxAvatar. Instead, phoneme names are used also as viseme names; i.e., there can be as many visemes as there are phonemes for the selected language. But since user registration should be quick and easy, it will usually not be possible to get a lip image for every phoneme. To have maximum flexibility regarding user registration, MaxAvatar was implemented in a way that enables it to detect which lip images exist and then tries to produce the best output possible with these images. This is achieved by the use of a preference mapping table. The preference table contains possible alternatives for each phoneme, ordered by their similarity to this phoneme. If the exact lip image is not available, the best available alternative lip image is used. The preference table is used for every phoneme lookup. The entry for the best alternative is always the phoneme itself, so that the exact lip image will be chosen if it exists. For every phoneme, the list of alternatives includes the silence phoneme. The silence phoneme is the only phoneme for which an image is mandatory, because the system relies on the possibility to choose it in case nothing else can be found. For an open mouth image, e.g., "A", the silence viseme would of course be one of the least preferred choices.

The table was created manually with a set of viseme classes in mind. These viseme classes were also defined manually and are shown in table 3.1. The most preferred alternatives for each phoneme are other phonemes from the same viseme class. If none of these exist, the next choice are phonemes from the most similar viseme class and so forth. As mentioned above, the silence phoneme is an alternative for every phoneme. It is assumed to exist, so that the preference table will always deliver information about a lip image that can be used.

This approach provides the ability to make the best out of the existing data. As soon as more data becomes available, the synthesis can show more detailed lip movement. The goal is to have a system that allows a quick initial registration and consequently learns and becomes better while being used.

Table 3.2 and table 3.3 show examples for the mapping from phonemes to visemes. They also show how the resulting visemes depend on which visemes are available. The video sequences look reasonable in both cases. The first line in both tables shows which visemes are available in the image database.

| | |
|---|---|
| a | AA AE AH AW AX AXR AY A AR AEH AEHR AH AHR AI AU |
| e | EH ER EY E E2 EH EHR ER ER2 EU |
| i | IH IX IY I IR IE IHR |
| o | OW AO O OR OE OEH ANG OH OHR |
| u | UH UW W U UR UE UEH UEHR UH UHR |
| j | CH JH NG Y X J R |
| d | D DH DX G HH K L N T H |
| f | F V |
| s | S SH Z SCH TS TSCH |
| t | TH |
| b | B P |
| m | M |
| SIL | SIL |

Table 3.1: Implicitly used viseme classes and the phonemes they contain.

| | |
|---|---|
| Available visemes | SIL, A, B, D, E, F, I, J, M, O, S, U |
| Text | how are you today |
| Phoneme list | SIL, H, UW, AA, R, J, UW, T, ER, D, EY, SIL |
| Result visemes | SIL, D, U, A, J, J, U, D, E, D, E, SIL |

Table 3.2: Example 1 for the mapping from text to available visemes

The second line contains the text that was synthesized. The phoneme list returned by the speech synthesizer is shown in the third line. The last line contains the visemes which have been found with the help of the preference mapping table. They are an approximation to the phoneme list with available visemes, and they will be used for synthesis of the corresponding image sequence.

Although the text to be synthesized is the same in both cases, the resulting visemes are different, because there are different visemes available in the databases.

| Available visemes | SIL, A, E, F, H, I, M, O, P, R, S, T |
|---|---|
| Text | how are you today |
| Phoneme list | SIL, H, UW, AA, R, J, UW, T, ER, D, EY, SIL |
| Result visemes | SIL, H, O, A, R, R, O, T, E, T, E, SIL |

Table 3.3: Example 2 for the mapping from text to available visemes

### 3.1.2 The Image Database

MaDataBase is the database module of MaxAvatar. It is responsible for handling the recorded and the extracted face data information. Currently, this includes the base face image as well as images of the left eye, the right eye, and the lips. More face parts can be integrated into the database in the future. The database module also detects, which images are available for the currently selected face. It loads those images into memory and provides access to them for other modules, mainly the synthesizing module. When another module requests an image, a pointer to the best matching existing image is returned. For lip images, the best matching image is found with the help of the MaPhonVis module and the preference table (see 3.1.1). For eye images, the image looking in a direction closest to the requested direction is returned.

The database for an avatar face consists of the following components:

- *The base face image.* This is the background image of the avatar, into which the eye and the lip images are pasted.
- *The lip images.* During user registration, all the obtained lip images of different phonemes are cut out of the picture containing them. They are saved as small, single graphic files. Information about the user name, the feature type (lips), and the shown phoneme are encoded in the filename of each image.
- *The eye images.* As the lip images, the different eye images are retrieved from the recordings made during the user registration process and saved in single files. Their filenames contain information about the user name, the feature type (left or right eye), and the gaze direction. The handling of the gaze direction will be explained later in this section.

- *The database information file.* This file contains information about all the images in the database, i.e., which images exist, where the features in the base face image are located, and what the location of the center points is. These center points are used to position the lip images correctly in the the base face during synthesis.

The images contained in an example database are shown in figure 3.2.

The procedure for handling lip images is described in the previous section (3.1.1). The eye images are handled as follows. Regarding eye movement, five different positions are supported for each the x and the y direction. For the x direction, there are the positions left, half left, straight, half right, and right. For the y direction, there are the positions up, half up, straight, half down, and down. Combining these two sets results in 25 possible eye positions. Additionally, an eye can be in the position "closed". The left and the right eye are saved separately, and they can be controlled independently.

Above we described the core database, which is obligatory to be able to create a talking avatar with moving eyes. During creation or modification of this database, more information can be added to the database. This information is stored together with the core database:

- *The recorded image sequence.* A single file containing the images that were recorded during user registration.

- *Image time stamps.* Timing information for each image in the image sequence. The time stamps mark the time in milliseconds beginning with when the recording was started.

- *Feature coordinates.* Coordinates of regions containing the facial features. Depending on which mode of user registration was selected, these coordinates are found automatically or marked manually.

- *The recorded speech.* A wave file with the sentence that the user spoke during registration.

- *Phonemes plus time alignment.* The phonemes of the spoken sentence and their time alignment, which is used to find the images containing certain visemes.

- *Image selection information file.* When the database is edited manually, the selections are saved in this file. They are loaded when the database is to be edited again.

The programming interface of the database module allows to select and load the base face, the lips, the left eye, and the right eye from different databases. Thus, a user can create an avatar with his/her face and somebody else's lips or eyes. Or, vice versa, the user can choose to appear as somebody else while keeping some of his/her own facial features.

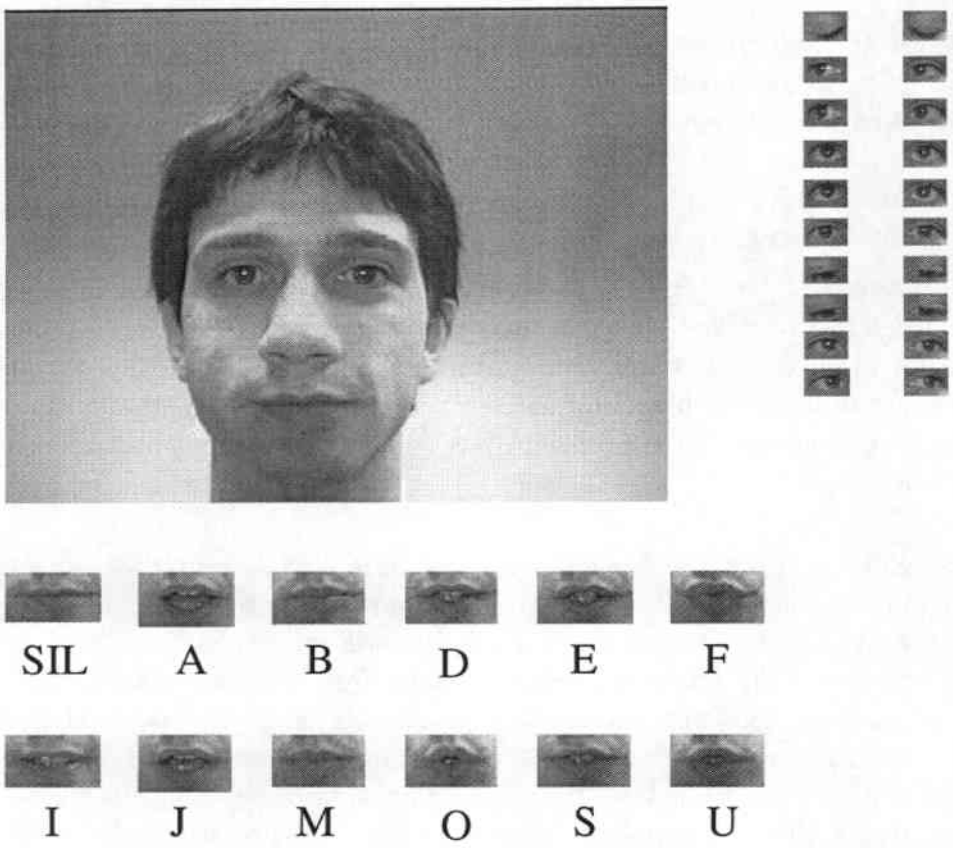


Figure 3.2: Base face, right eye, left eye and lip images from an example database.

### 3.1.3 Face Synthesis

MaSynth is the synthesizing module of MaxAvatar. It produces avatar faces that show the requested eye and lip positions. It obtains the necessary image data from the database module; i.e., calls of the synthesizing module will result in avatar images based upon the face data information loaded into the database module.

One way to get a synthesized face is to specify the x and the y positions for each of the eyes and the viseme that shall be shown by the lips. In this case, a single image will be synthesized and returned.

MaSynth also offers a function (SynthesizeMovieVideo) that allows to synthesize a face saying a whole sentence. To achieve this, we need the phoneme list, the time alignment of the phonemes and the audio file [1] of the sentence. Normally, this data will come from a speech synthesizer. It has to be stored in a MaMovie object (see 3.1.4), which is then passed to SynthesizeMovieVideo. In this function, one face is synthesized for each phoneme in the phoneme list. Afterwards, the MaMovie object is returned, now containing all the information necessary to let the avatar speak the sentence.

To synthesize a face, first the base face image and the corresponding feature locations are retrieved through the database object. For each of the eyes and for the lips the best matching available image is also retrieved from the database object. Then each of the facial feature images is stitched into the base face image at the appropriate position. The stitching process consists of three steps:

a) *Scaling.* First the facial feature image is scaled to fit the corresponding region in the base image. It is hard to distinguish automatically between cases where this step is desired or not, especially when the face and the facial features are taken from different databases. The program does not know, whether one of the users was further away from the camera than the other user, or whether one of them just has smaller eyes than the other one. The same is true for the lips. Currently, the eye images are being scaled and the lips are not being scaled.

[1] The audio file is not required for the synthesizing process. It could be added to the MaMovie object later, any time before the avatar shall speak the sentence. In our case, the wave file becomes available before the phoneme time alignment, so we add it at this point already.

This seems to produce good results. Scaling of the lip images is also a problem, because lip images for different phonemes vary much more in size than different eye images within the same database.

b) *Intensity Adaptation.* Intensity adaptation becomes necessary when the base face and the facial features are taken from different databases and when these databases were created from recordings with different lighting conditions. The implemented intensity adaptation calculates the average intensity of the destination rectangle and adjusts the average intensity of the feature image so that it is the same as in the destination rectangle.

c) *Edge Smoothing.* In the last step, a smooth transition between the original image and the new facial feature is calculated. This is done with an outer and an inner ellipse, both fitting inside the feature image. Points, which are not located in the outer ellipse, are taken from the original image. Points, which are located in the inner ellipse, are taken from the new feature image. For the rest of the points, located in the outer ellipse but not in the inner one, the result is a linear combination of the corresponding pixels in both images. The closer the point is to the outer ellipse, the bigger the weight for the pixel from the original image. And accordingly, the closer the point is to the inner ellipse, the bigger the weight for the pixel from the image of the feature.

Figure 3.3 shows the results of the stitching process. Another useful step in this process would be color adaptation, which is not implemented yet. The overall result of the above procedure looks deceivably real when applied to images from the same database. However, if the base face and the feature images stem from different databases, it often happens that the color tones are disturbingly different, so that the result looks unnatural, as shown on the right side of figure 3.3. These color differences are due to different lighting conditions, different cameras, and individual color differences between people.

The positioning of the eyes works differently than that of the lips. As described above, the new eye images are resized to fit exactly the eye region of the base face image, which is stored in the database file. Thus, it can simply be copied into the base face image at that position. The lip images are not being resized, and for different lip shapes it is not clear where exactly it has to be copied if just a rectangle is given. Therefore, the center point

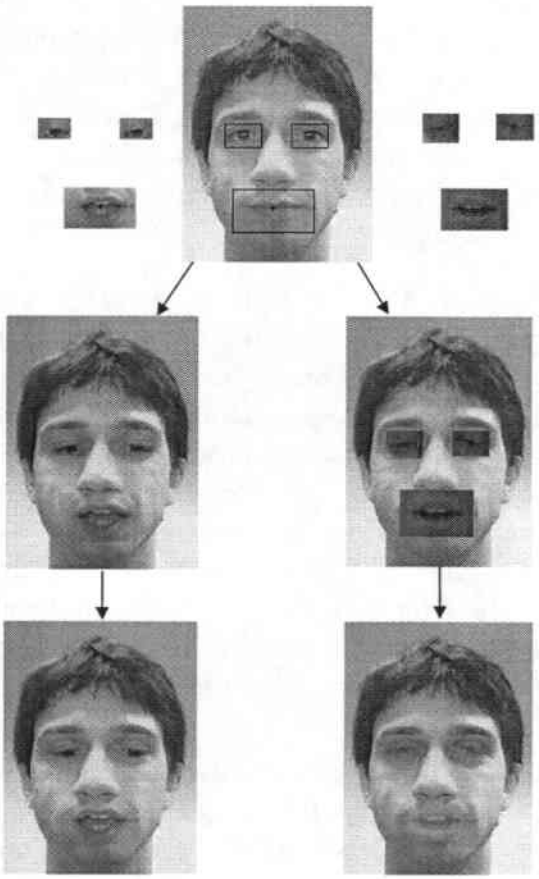

Figure 3.3: *Synthesizing of an avatar face.* Eye and lip images are stitched into a base image. First, they are only copied, then scaling, intensity adaptation and edge smoothing are applied. On the left, all images are from the same database. On the right, eyes and lips are taken from a different database than the base face.

has to be determined for the lips in the base face image and in each of the lip images in the database. The center point should be a fixed point in the face from which the lip images have been extracted. During synthesis, the new lip images are positioned in the base face image so that the lip center points match.

### Transition Images

As described previously, for each phoneme in the synthesized utterance, one image with the best matching available lip image is synthesized. This results in abrupt transitions between the lip images, especially when two consecutive images are very different. To improve the quality of the output, we tried two approaches to make these transitions smoother. In principle, these approaches can also be applied to calculate transitions between different eye images.

The first approach to create smoother lip transitions uses Quadrilateral-to-Quadrilateral mapping, also known as four-courner mapping [Wol90]. As shown in figure 3.4, each of the lips in the images before and after the transition image has to be marked by four points; i.e., the top, the bottom, the left, and the right edges. The content of the first quadrilateral is then mapped into the second quadrilateral.

Figure 3.4: Lip transition images with Quadrilateral-to-Quadrilateral mapping. Left: the previous image. Right: the next image. Middle: the calculated transition image.

The problem of mapping one quadrilateral to another is decomposed into two problems, which are mathematically easier to handle:

a) The source quadrilateral is mapped onto a union square.

b) The union square is mapped onto the destination quadrilateral, which uses the inverse function of the first step with different parameters.

The mapping uses perspective transformations. Their general representation is

$$(x', y', w') = (u, v, w) \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{3.1.3.1}$$

where $x = x'/w'$ and $y = y'/w'$. $(u, v)$ are the coordinates of the input pixel and $(x, y)$ are the coordinates of the output pixel. $w$ is a scaling factor. The forward mapping functions are

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}} \tag{3.1.3.2}$$

$$y = \frac{y'}{w'} = \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}} \tag{3.1.3.3}$$

Without loss of generality, the transformation matrix in 3.1.3.1 can be normalized so that $a_{33} = 1$. This leaves eight degrees of freedom for a projective mapping. The eight coefficients can be determined by establishing correspondence between four points in the source and the destination image.

Although the result looks promising, it is difficult to find the corner points that yield the desired result. These are not necessarily the edges of the lips. Regarding the fact that it is possible to have different lip images with nearly the same width and height, a four point approach seems to be too simple. To successfully apply mapping techniques, more detailed knowledge about the inner and the outer edges of the lips is necessary.

The second approach uses linear combinations of two lip images to calculate a transition between them. The function takes a parameter $a \in [0..1]$ that represents the weight of the first image in the linear combination. The resulting pixel $p$ for two corresponding pixels $p_1$ in the first image and $p_2$ in the second image is

$$p = a \cdot p_1 + (1 - a) \cdot p_2 \tag{3.1.3.4}$$

A similar idea is presented in [GMO98], where all the lip images, including transition images, are calculated as a linear combination of only six recorded

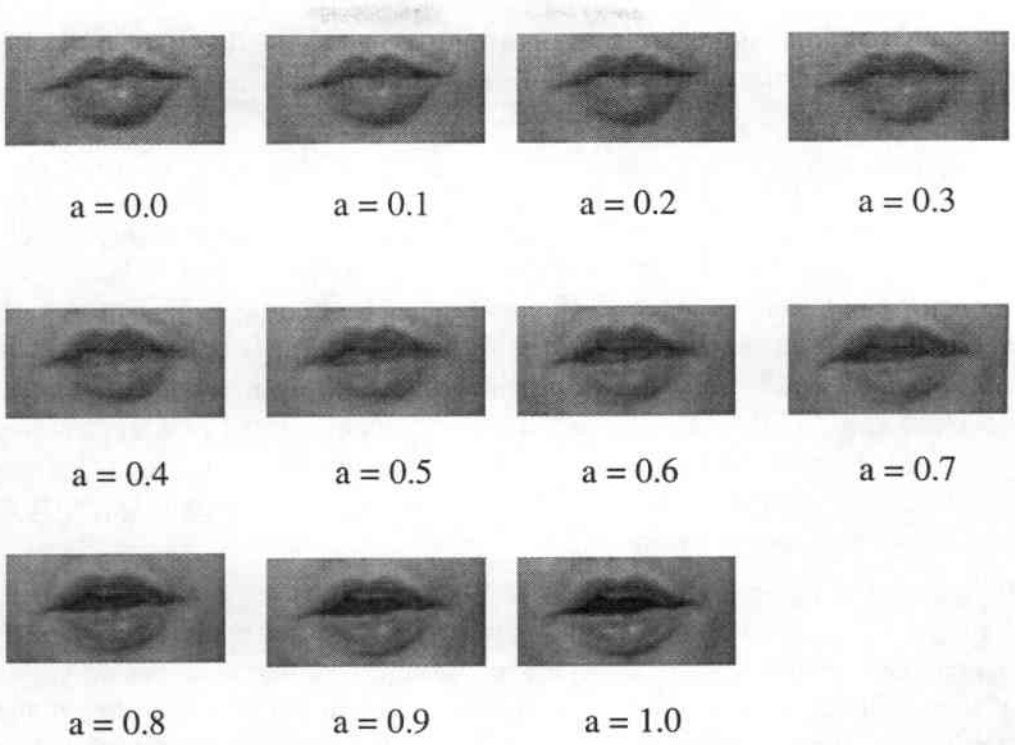


Figure 3.5: Lip transition images with linear combination.

lip images. This is also interesting regarding the possibility of adding new faces very quickly (see section 3.2).

The result for different values of $a$ is shown in figure 3.5. The most visible changes take place for $0.3 < a < 0.7$.

Although the second approach does not use any information about the lip shape, it allows for the impression of a smooth transition between two lip images.

We tested the first approach with a hand-labelled image sequence because we currently have no means of finding the appropriate quadrilaterals automatically. The second approach can be switched on or off for synthesis, and the user can choose a target frame rate that shall be reached with the additional transition images. We tested frame rates of 30, 50, and 100 images per second. The result for both approaches and for all the tested frame rates is about the same. Although single images in the sequence look like an appropriate transition between the previous and the next image, the lip movement

still does not look smooth. It seems to go to the right direction, but when looking at the result one gets the impression of too much lip movement.

### Presynthesizing

Although most of the synthesizing procedures are only performed on relatively small parts of the images, the whole process can easily take a few seconds (see section 2.13) for all the images needed to make the avatar speak a complete sentence. Together with the time consumed by speech synthesis (and maybe other steps like speech recognition or translation), this can make the system intolerably slow. When the avatar is used as a communication agent, there is hardly going to be any conversation when each sentence is followed by a period of 20 seconds of silence. An example for such an application is given in section 4.1.

Most of the time for video synthesis can be saved, although there are certain limits for it. One way to save time is to precalculate images and keep them in memory. When needed for synthesis later, they just have to be copied or, even better, only a pointer to their memory location has to be copied. MaxAvatar offers the feature to do that. Face images are precalculated for each phoneme in the database in three variations: Once each with the eyes to the left, straight, and right. After this has been done, video synthesis takes no noticeable time. Unfortunately, this approach has some limitations. The most important limitation is memory, which will usually not be big enough to allow precalculation of all the possible face images. This becomes especially true when more facial features are used, when transition images between the phonemes are desired to make the lip movement smoother, or when head rotation shall be modelled.

### Animation

Other features of the video synthesis module are the face queue and a function for random animation of the avatar. The face queue makes it possible to store a sequence of faces, i.e., for each face, the x and y positions of each eye and the desired lip image are stored. This face sequence can be accessed in FIFO (first in first out) order. The animation function generates random actions of the avatar (like blinking or looking around) by putting the appropriate face sequence data into the image queue. Each time the animation function is called, it first checks whether the face queue contains information. If it does,

the corresponding image is synthesized and displayed, and the information is dequeued. If the face queue is empty, the next action is randomly generated. Section 3.3 explains how this feature is used in the main program.

### 3.1.4 Movies

A movie object of the class MaMovie provides all the functionality to store and display an image sequence that is synchronized to an audio file. We call this a movie. Movies are used in MaxAvatar in several ways. They are used for automatic user registration (see 3.2.1) to record the video and the audio signal of the user saying a sentence. They are also used for interactive user registration (see 3.2.2 and 3.2.3) to hold a set of separately taken pictures without speech or timing information. The video synthesis module (see 3.1.3) also stores its results in a movie. The images and the additional information are stored in arrays and can be accessed through the image number. List access functions are also provided to get the next or the previous image, goto or check for the first or the last image, or to append, insert, or delete an entry.

Here is an overview over the information that can be stored in a MaMovie object:

- *Images.* All the recorded or synthesized images.
- *Time stamps.* When an image is being recorded, the time is noted when the recording is started. The time stamp for an image is the time in milliseconds after the start of recording when the image is taken.
- *Feature labels.* Another module (e.g., user registration) can take notes regarding which image contains which kind of features, i.e., where the eyes look and which viseme is visible.
- *Feature selections.* This is also intended for user registration purposes, in order to mark which image contains which feature(s) that shall be extracted and saved in the database.
- *The audio data.* The recorded or synthesized speech, which belongs to the image sequence.

There are several file formats available for synchronized audio and video data, e.g., MPEG or AVI. In the prototype system, we nevertheless decided to save the different kinds of information in separate files. There are several reasons for this. First, there is no file format, which is designed to hold all the information of a MaMovie object; e.g., feature labels and selections are usually not supported. Thus, more than one file would be necessary with existing formats. A possible solution is the development of a new file format. This might be more elegant, but it gives little actual advantages. Second, not all of the data is present and needed all the time, depending for which purpose the movie object is used. And third, for test purposes it is often handy to have direct access to data like feature labels and selections with a regular text editor. Therefore, the images, the time stamps, the feature labels and selections, and the audio data are each saved in separate files.

### 3.1.5 Recording and Playing of Audio-Visual Data

The MaRecPlay object is closely related to the MaMovie object. As the name suggests, it offers functionality to record and play synchronized audio and video data.

To help set everything up correctly for recoding, the camera image can be previewed without recording it. It is possible to choose the desired frame rates for previewing and recording separately. During previewing and recording, the actual frame rate is measured, so that a detection of a frame rate that is too low or too high is feasible. The PCs which were used to test the prototype systems have "Video For Windows" frame grabber cards installed. For recordings, a resolution of 320x240 pixels was used. The maximum frame rate at this resolution is about 15 frames per second.

Another option during recording is whether virtual memory shall be used to store recorded images. Otherwise, only available physical memory is used. The use of virtual memory significantly increases the maximum number of images which can be recorded. On the other hand, the frame rate decreases noticeably as soon as virtual memory is needed. This happens, because parts of the data have to be swapped to disc by the operating system, in order to have room for more image data in the physical memory. For user registration purposes, a frame rate of 15 frames per second is already low. A lower frame rate is not acceptable, because some of the viseme pictures will most likely

be missing. Therefore, if using Windows NT, at least 128 MB of physical memory are recommended to make sure that no virtual memory has to be used during recording.

After initialization for recording, a frame is requested from the frame grabber card. As soon as the next frame becomes available, the operating system calls a callback function in MaRecPlay, which retrieves the camera image and stores it with the current time stamp in a movie object. When this is done, the next frame is requested, and so forth.

A special feature is the split preview mode. Since we are only able to have one frame grabber card in each PC, a workaround has to be found in order to use two cameras simultaneously on one PC. One way to do this is the use of a video mixer, which is able to take input from several cameras and mix them into a single video signal. For the split preview mode, the video mixer has to deliver a signal, where the input from one camera is in the upper part of the picture, and input from another camera is in the lower part of the picture. When in split preview mode, MaRecPlay splits the camera image (in this case, the signal received from the video mixer) horizontally in two parts and displays them in two separate windows. This feature has been used to show (on one screen) a complete conversation of two parties using the avatar as a translation agent.

### 3.1.6 Face and Facial Feature Detection / Tracking

MaImageReg is the image registration module of MaxAvatar. It is responsible for finding and tracking of the face and facial features. It stores all the involved image coordinates and makes them available for other modules.

Accurate detection of facial features coordinates is crucial for good results when building an avatar with a real person's face. It is important while the database is being created. Errors in detecting the lips, for example, can result in having the image of only half the lips in the database. Precise knowledge about the feature locations is also important during the synthesis phase. The avatar's face looks disturbingly unreal when the features are only as much as a few pixels off. It is especially unnerving when the lip pictures are not put in consistently at the same location in the face. This leads to a talking mouth that is jiggling around very quickly.

Another application for very precise feature detection is the synthesis of

unavailable feature images from those, which are available. With two eye images, one with the eye looking to the left and the other one with the eye looking to the right, it is theoretically possible to calculate all the images in between. For good results, however, knowledge about the pupil position and size and about the eye edges is necessary. Similarly, new lip images could be generated from one or more lip images and exact knowledge about the lip shape.

As no reliable facial feature tracker was available, an attempt was made to build one for the prototype system. Although the results are promising, it is only a prototype with a lot of restrictions. It does not work in real time. It works well on some faces and fails on others, especially when a beard is present. Facial feature detection for the general case is a difficult problem for a computer [SMY97a, SMY97b, YSMW98]. For the presented approach, the following premises were made:

- The image to be analyzed is a frontal shot of a face.
- The face is relatively big and completely visible without occlusions.
- The face does not move quickly.

Since it was clear that no state of the art feature detector could be implemented in the framework of this thesis, care was taken to make it easy to plug in other, more advanced feature detection systems. It is possible to exchange the whole module or only the algorithms for detection of the face or single features.

Of course, regarding user registration, feature detection is only necessary to find the feature coordinates automatically. This is desirable, because one of the goals of MaxAvatar was to have a quick registration procedure for a new user, which is easy to use and as effortless as possible. This concept can be demonstrated, but often the accuracy of the results is not good enough. Therefore, an online (section 3.2.2) and an offline interactive mode (section 3.2.3) are offered as fallback mechanisms for the automatic user registration (section 3.2.1).

Supported facial features in this approch are the pupils, the eyebrows, the eye edges, and the lips. A brief description of how these features are found is given below.

Many face detectors only determine a rectangle containing the face, which is a good enough approximation for many applications. Here, an attempt was made to find the exact shape of the head for two purposes that are described in the next two paragraphs.

### Background Subtraction

When a face is recorded for user registration, the background is recorded as well and will appear again later, when the face is used for the avatar. This might not always be wanted. Imagine the example of somebody who records his/her face at home, but his/her bed is visible in the background. Later, the face database created from this recording shall be used for a business teleconference. In this case, a neutral, or at least more formal, background would be appropriate. Movie studios use a blue screen as a background when another background shall be blended in later. However, such techniques are not available for everyday use. MaImageReg offers a background subtraction function that does not rely on a blue screen background. Instead, it determines the exact outline of the head. Then the original background can be replaced with any other background.

The algorithm works as follows. Motion in the image sequence is calculated and summed up over a few frames. As stated above, one of our assumptions is that there is only little motion of the head. Experiments also show that the head never stays completely still. Assuming further, that the head is the only moving object in the image, we can take the motion information as a good estimate of the head position. Some smoothing and thresholding is applied to the motion image to eliminate noise. Using horizontal and vertical sums, the left, right, upper, and lower edge of the head are determined. Everything outside the resulting rectangle is erased. Then a hull is adapted around the remaining points, paying attention that the result is smooth and convex enough to resemble a head.

The steps of the algorithm are shown in figure 3.6.

### Face Detection

The other purpose for face detection in MaxAvatar is mainly to narrow down the search region for the facial features. This reduces the time for feature detection, because a smaller part of the image has to be processed. It also

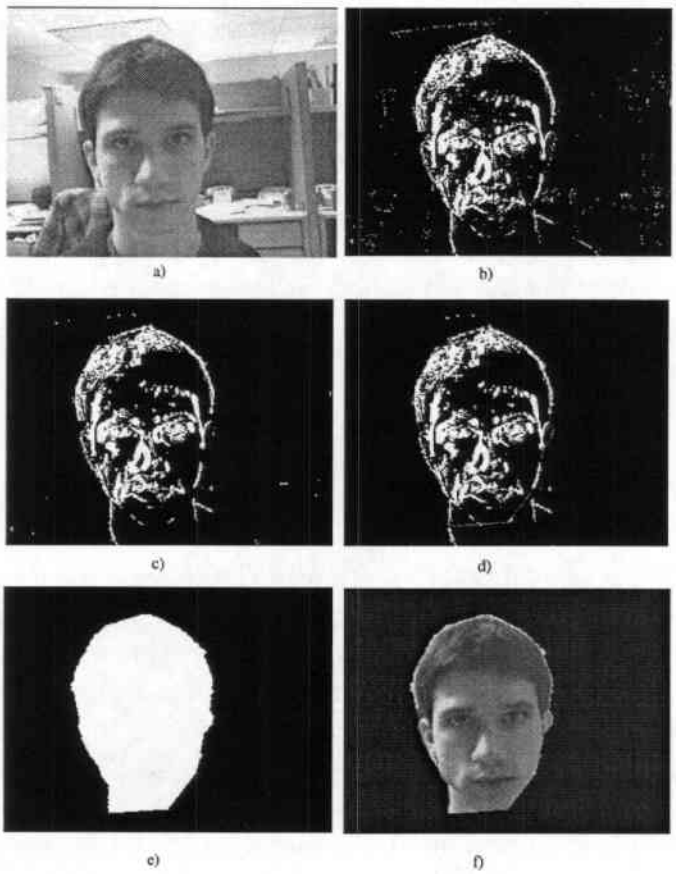


Figure 3.6: *Background subtraction.* a) the original image, b) motion over a few frames, c) applied noise filter to motion image, d) detected head outline, e) solid head mask, f) the original head image with a blue screen background

increases the probability of a correct detection. When a search is performed on the whole image, a dark spot in the background might be mistaken as a pupil, for example. In a first approach, it was tried to find the exact outline of the face, i.e., without the hair. Later, an approximating rectangle was chosen, in order to increase the speed and robustness of the face detection.

For this function we tried a different algorithm than for the background subtraction function. It consists of the following steps, which are illustrated in figure 3.7:

a) As in the background subtraction algorithm, the first step is to calculate movement in the image sequence. However, instead of several frames, only two consecutive frames are used. And the motion information is only used to get a rough idea of the face position. This is done by calculating the center point of all the pixels in the motion image. In experiments, this center point was almost always located near the center of the face.

b) A gray value image is calculated from the original RGB image, where the gray value for each pixel is essentially assigned the red value minus the green value of the original image.

c) The gray values from the previous step are clustered into two sets, which are assigned the colors black and white. This leaves the face either as big black or as a big white area. We also tried to eliminate pixels with gray values, which are most likely not of face color. This was done successfully by using a histogram, where the face color appears as a peak because it covers a big part of the image. This also had the advantage that the face could be assigned the same color (black or white) every time. However, this was not pursued further because there were often too many points eliminated within the face. The resulting holes in the face were small, but big enough to disturb the rest of the algorithm.

d) By looking at the area around the center point from step a), it is determined whether the face was assigned the color black or white in step c).

e) Starting from the center point from step a), the face is filled out solidly and everything else is erased.

f) As a final step, the resulting face blob from the previous step is smoothed with respect to some heuristics about the face shape; e.g., the face is supposed to get wider when going down from the top, stays about the same in the middle, and does not get wider in the lower part.

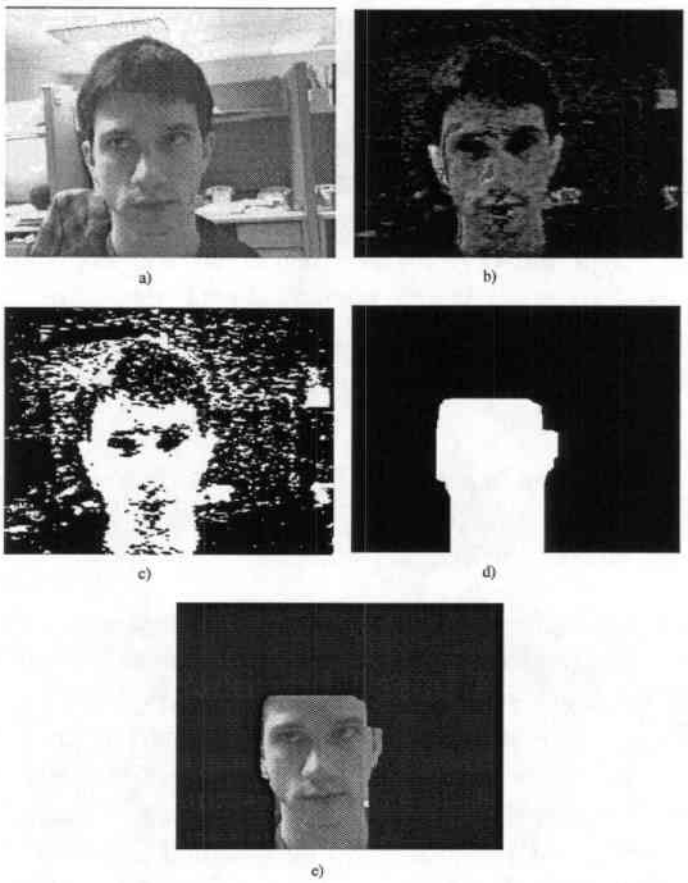


Figure 3.7: *Face detection.* a) the original image, b) face color, c) clustered face color, d) solid face mask, e) the original face image with a blue screen background

### Feature Detection

The feature detection algorithms described in this paragraph rely on information about face location and size. They also should be called in a certain order, as will be explained. For instance, the search region for the lips is

determined from the face size and from the position of the pupils, which therefore must be found before the lips.

Figure 3.8 shows the results of the feature detection algorithms presented below. The following things are detected and marked: a rectangle containing the face, a circle marking the pupil (and maybe the iris), a curve denoting the edge of the upper eyelid, a small rectangle on the lower edge of the eyebrows, a rectangle containing the lip area, and a point indicating the center of the lips.

Figure 3.8: Original face image and detected features.

### Pupils

The first feature that is looked for is the right pupil. With the information about the face location, the search region for the right pupil is estimated to be found in the upper left part of the face image. When the right pupil has been found, the search region for the left pupil can be better estimated.

To classify a spot as a pupil, several criteria are checked. Most of all, the spot has to be dark. Whenever a cluster of dark pixels is discovered, it is considered as a possible pupil. "Dark" means it is either the darkest spot found so far, or it is almost as dark as the best candidate found so far. In addition to darkness, the region around the spot is examined to see whether it can possibly be an eye. For that, the deviation in that area should be high, and the area surrounding the pupil should be brighter than the pupil itself. This helps distinguishing the pupil from the eyebrows or hair, which might also be very dark.

### Eyebrows

With the pupils as a starting point, the next feature looked for are the eyebrows. They are not used for feature extraction during user registration, but used as an upper boundary for estimation of the eye region.

In an estimated region above the pupils, the eyebrow search function looks for a dark stripe with a transition to a brighter stripe on the lower edge. This transition is found by edge detection. A candidate is considered to be better than a previously found one, when it is either significantly darker, or the transition from dark to bright is significantly steeper.

### Eye edges

Finding the exact eye edges is a difficult problem for several reasons:

- The pupil can take various positions within the eye. Thus, knowing the pupil location does not tell too much about the eye edges, especially not about the left and the right edges.
- Varying lighting conditions make it hard to find general rules. With a light source from above, the upper edge of the eye will be relatively dark because of shadows from the eyebrow region. With light from below, the upper eye edge will be bright. Similar problems occur for the left and the right edges. Side lighting can cause a shadow from the nose, or for one eye to be completely bright and the other one completely in shadows.
- There are also great individual differences between the eyes of different people. They differ in size, color and shape (e.g., caucasian versus asian shaped eyes), and the distance between the upper edge of the eye and the eyebrow varies. The opening angle varies as well. Sometimes the whole pupil is visible, sometimes only a small part of it. When the opening angle is small, the visible part of the eyeball is sometimes very dark because of shadows. This makes it almost impossible to tell the pupil and the eyeball apart, and thus it becomes also impossible to tell where the eye is looking.
- Another difficult problem is the case of closed eyes. Obviously, the presented approach to find the pupils first must fail when the eyes

are closed. This case must be handled separately. One way to do that would be to look at the (hopefully correct) eye positions of the previous frames. Assuming that the face did not move too much, an indicator for a closed eye would be brightness where there used to be darkness in the eye region indicates a closed eye. A lot of movement in the eye region is another indicator, although quick pupil movements can produce a similar effect.

In MaImageReg, the following steps are taken to determine the eye edges. First, the algorithm looks for the upper and the lower edges. More specifically, it tries to find the first dark pixel, which does not belong to the pupil. This dark pixel stems from the eyelashes or a small shadow on the eyeball from the eyelids. There is usually an entire line of these dark pixels, especially on the lower edge of the upper eyelid. The algorithm tries to trace this line to both sides, starting at the x position of the pupil. The left and the right eye corners are marked where this line ends in a dark spot, which is where the upper and the lower eyelid merge. This algorithm works well in most cases. Problems seem to arise mainly when the dark line or the end spot are not very distinctive, or when the eyes are not open wide enough so that only a little part of the pupil is visible.

### Eye Detection with Correlation

We tested another approach to find the eyes was tested, using correlation. The correlation of one or more sample eye images was calculated for every possible position in the estimated search region and for different sizes of the sample images. The position with the highest correlation was taken as the eye position in the image. The results were promising, although overall not better than those of the approach described above. When the eyes were found, the positioning was not as exact. Under poor lighting conditions, the result was sometimes completely off, where in the other approach, at least the pupil was detected correctly.

Correlation alone does not seem to be a solution for the feature detection problem. It also can become pretty time consuming, particularly when samples from different types of eyes in different sizes looking in different directions are tested, because of the huge number of tests which have to be performed. However, correlation might be helpful when used to complement other techniques or when the search area is very small.

### Lips

With the help of the eye locations and the face size, a rough estimate is made for where to look for the lips. The first thing looked for is the horizontal center line of the lips. This is done by searching for the minimum sum over a few horizontal lines. Around this line, a more accurate estimate of the lip region is made. The rest of the algorithm works only on this region.

Two new images are created. One gets the result of the derivation in the x direction. The other one gets the result of the second derivation in the y direction, which is used to find the upper and the lower lip boundaries. Furthermore, the horizontal center line of the lips is determined more accurately by looking for the darkest points near the previously found line. The right edge is found by starting in the middle of the horizontal line and then following it to the right. The place of the steepest transition in the x derivation image during this process is marked as the right lip edge. The left lip edge is found analogously. The derivation images and the detected boundaries are shown for a sample image in figure 3.9.

After smoothing the upper and lower lip boundaries, their extreme points are taken as the upper and the lower edges of the smallest rectangle completely containing the lips.

### Constraints and Smoothing

Many of the algorithms described in this section use certain constraints or magic numbers, which have been selected by trial and error. Through fine-tuning of these values, the results can still be improved. In the future, some of them should also be determined automatically for the individual case, to make the algorithms work more generally.

MaImageReg stores all the feature coordinates that have been calculated for an image sequence. This provides the opportunity for sophisticated smoothing techniques to filter out wrong values. MaImageReg also supports different smoothing modes, so that new techniques can easily be implemented and tried. So far, only averaging between the current and the previous value is available.

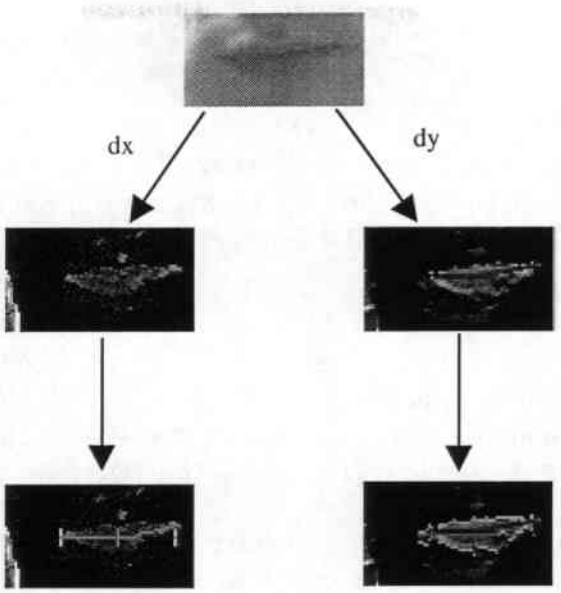


Figure 3.9: Top: the original lip image. Below: Derivation of the lip image in x direction (on the left side) and in y direction (on the right side) plus the detected lip boundaries.

**Template Tracking**

Another surprisingly effective algorithm in MaImageReg is the template tracking algorithm. If it is given a rectangle containing a feature, this feature can be tracked with an acceptable frame rate, depending on the size of the rectangle and on the machine used. The algorithm is similar to correlation. It calculates the difference between the image rectangle in the previous frame and the image rectangle in the current frame. This is done at different positions in a configurable area and with configurable accuracy.

Template tracking is a much easier task than feature detection. Nevertheless, it can be very useful. In MaxAvatar, it is used to make the on- and offline interactive user registration more convenient (see 3.2.2 and 3.2.3).

### 3.1.7 The Client/Server Interface

MaClientServer handles the client/server connections between MaxAvatar and other programs. A list of these connections and a short description of

the communication purpose is given below. For more details see the corresponding sections, which are also mentioned below.

MaxAvatar acts for some of the connections as a client and for others as a server. In all cases it is possible to select the port number through the graphical user interface. In the case of MaxAvatar being the client, the hostname of the server can also be selected.

- *Festival.* MaxAvatar connects to the Festival text-to-speech system as a client through a TCL script. The text to be synthesized is sent to Festival, which returns the synthesized wave form plus the corresponding list of phonemes and their time alignment. This information is used to synthesize the video part of the avatar (see also section 2.5).

- *Janus.* The Janus speech recognizer is used during automatic user registration. MaxAvatar connects as a client through an intermediate TCL script. It sends a recorded audio file, where the user was asked to say a given sentence. A forced alignment is performed on that sentence, and MaxAvatar gets back the corresponding list of phonemes and their time alignment. This information is used to retrieve viseme images from the video recording for the face database (see also section 3.2.1).

- *C-STAR blackboard.* MaxAvatar connects to the C-STAR blackboard as a client through an intermediate TCL script. This script receives text and some other information from the C-STAR blackboard. It extracts the text and sends it to MaxAvatar, which synthesizes and displays the avatar speaking that text (see also section 4.1).

- *Microphone direction server.* The microphone direction server is connected to a microphone array or some other means to find out the relative direction of a person speaking. It provides this direction information to clients as MaxAvatar. MaxAvatar can make use of it by turning the eyes of the avatar so that it looks at the speaker. It can also address the person who spoke last when saying a sentence, e.g., to answer a question (see also sections 3.3 and 4.1).

- *Image server.* MaxAvatar can be used not only as a front-end for audio-visual output, but also as a back-end to deliver all the information necessary to display the avatar in another application (see also section 4.2).

- *File server.* MaxAvatar also provides the basic functionality of a file server. This complements the image server mentioned above that only sends references to images. Should there be no other way to access those, they can be downloaded through the file server (see also section 4.2).

### 3.1.8 Other Modules

#### MaIO

The MaIO module plays a special role in MaxAvatar. It encapsulates system-dependent input and output functionality as far as is possible. The current system uses the Microsoft Foundation Classes for the graphical user interface, audio and video recording and playback, socket connections, and timers. Thus, it currently only runs under Microsoft Windows 95/98/NT (see also section 3.5). The basic functionality of MaxAvatar, without user registration, should be relatively easy to port to another operating system by replacing the input and output functions in MaIO. These functions are consistently used in the whole program.

#### MaImage

The MaImage class provides functions to display (through MaIO), load, save, modify, and evaluate images. It uses the C image library, developed in the Interactive Systems Laboratories, for the underlying datastructures. It also offers additional functionality, which is used by many of the other modules.

#### MaParam

MaParam allows convenient handling of program parameters. In a first approach, the parameters were stored in the Windows registry. This had the disadvantages of being platform-dependent, requiring a lot of manual adaptation for individual preferences each time the program was installed on a different machine, and not providing easy access for test purposes. As a result, the current version uses a parameter file, which is platform-independent and easily accessible with a regular text editor. It can be modified with or without MaxAvatar. MaParam supports comments in the parameter file, as well as relative paths. The use of relative paths makes the installation of

MaxAvatar very easy. To add a new parameter, a line with the parameter name and its value have to be added to the parameter file. It can then be accessed through its name by every module of MaxAvatar.

#### MaUtil

MaUtil is a collection of general purpose functions that did not fit into any of the modules with more specific responsibilities. Examples of MaUtil functions include routines for safe file handling, audio format conversions, and modification of strings.

## 3.2 User Registration

An important aspect during the development of MaxAvatar was the registration of a new user. The goal was to give every user the opportunity to add his/her own face to the face database in a quick and easy process and then use it as a face for the avatar.

MaxAvatar offers three different modes for user registration. The first is the automatic mode (3.2.1), which requires very little effort for the user, but also does not allow for corrections. The second mode is the online interactive mode (3.2.2). Here, the user is prompted for more specific actions to take a series of single pictures, and it is possible to modify the results. The third mode is the offline interactive mode (3.2.3), which can be used to manually correct the results from automatic mode, or to interactively create an avatar face from an arbitrary image sequence. It also offers some features to make this process easier.

### 3.2.1 Automatic Mode

When choosing automatic registration, the user is first asked for his/her name. If a face with this name already exists, the user can either enter a different name or overwrite the old database. For test purposes it is also possible to proceed and reuse an existing recording, an existing phoneme time alignment, or feature coordinates previously found.

If a face with this name does not exist yet or if the user chooses to overwrite the old one, some instructions for the recording are displayed. Before the

recording is started, a live preview of the camera image is shown. This helps the user to position his/her head correctly; i.e., in a way that the head is completely visible and large in the picture. Pressing the *Enter* key starts the recording.

The user is asked to speak the sentence "How far is it from the airport?" The phoneme list for this sentence is "SIL HH AW F AA R IH Z IX TD F R AO M DH IY EH R P AO R TD SIL." It is a relatively short sentence, containing a set of visemes, which is sufficient to create resonable looking mouth movement. It is possible to expand the database of viseme images later, if desired.

After saying that sentence, the user is supposed to look to the left, to the right, up, down, and to blink. This will provide image data to animate the eyes of the avatar. Finally, pressing the *Enter* key again stops the recording.

The rest of the procedure is fully automatic. After saving the recording to disc, the feature detection algorithms (described in 3.1.6) are used to get the coordinates of the eyes and the lips.

The audio recording and the known text are sent to the Janus recognizer. Janus uses forced alignment to compute the time stamps of the phonemes in the acoustic signal. With these phoneme time stamps and the video time stamps from the camera recording, the system can determine which image in the recorded sequence provides which viseme. For each of the available visemes, the part of the image containing the lips (determined by the feature positions which have been calculated previously) is extracted, labeled, and saved in the database. This is described in more detail in section 3.1.2.

Typical durations of phonemes are between 10 and 200 milliseconds. Using the method described above to locate certain visemes in an image sequence might cause one or both of two possible problems, depending on the frame rate during recording. For high frame rates or long phonemes, more than one image frame will be associated with one phoneme. Some of those frames might contain the desired viseme, while the others only show a transition state from the previous or to the next viseme. The problem is how to find the right image frame. It has not been solved in the prototype system yet. Considered approaches include the use of correlation with an existing set of lip images, and the use of a lip shape recognizer, which is not yet available.

The eye images can be retrieved from the recorded image sequence by only evaluating the feature coordinates. The eye gaze can be determined from

the relative position of the pupils and the eye edges. A separate detector is necessary to find the image containing closed eyes. When the image and the eye positions are known, the eye images can also be extracted, labeled and saved in the database.

Unfortunately, in the current system the detection of the left and right eye edges and of closed eyes is not advanced enough to produce a usable database of eye images automatically. This is an important point for future work.

### 3.2.2 Online Interactive Mode

When choosing online interactive registration, the user is first asked for his name. If a face with this name already exists, the user can either enter a different name or edit the existing recording. The format of the recording for online interactive registration is different from the format for automatic registration. For automatic registration, a whole movie (see 3.1.4) is recorded of the user saying the sentence and moving the eyes. For online interactive registration, only a number of single snapshots are taken (which are nevertheless internally stored in a movie object as well).

A screenshot of the dialog window is shown in figure 3.10. This section describes which features this dialog offers.

During online interactive registration, it is possible to switch back and forth between two different modes. One is the camera preview mode and the other is the editing mode. In preview mode, the user can position his/her head in the camera image, mark the eyes and the lips, and take a snapshot whenever he/she is ready. In editing mode, the taken snapshots can be reviewed and the feature marks can be repositioned. It is also possible to take a snapshot again if it turns out that the old one is not good enough.

The user is prompted separately for every picture to be taken. In this first attempt, 14 different pictures are supposed to be taken: six for the eyes (straight, left, right, up, down, and closed) and eight for the lips (A, E, I, O, M, F, R, S). The silence viseme and the base face image are taken from the snapshot for the straight eyes. The kind of snapshot to be taken next is displayed in the dialog window. The user can also choose to be prompted by a voice telling him/her what to do next. This makes it easier to concentrate on the camera image. When the user is ready, clicking on a button or pressing a key is enough to take the snapshot.

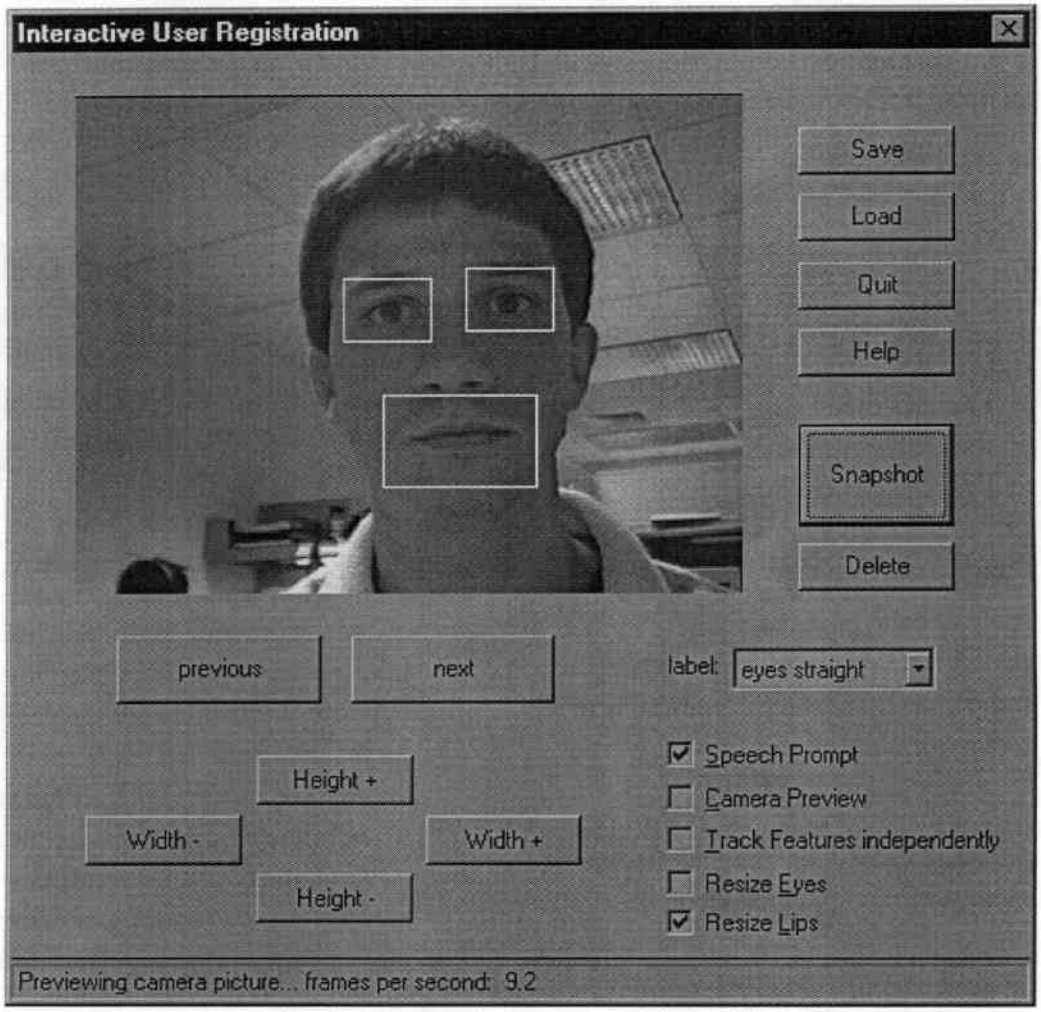


Figure 3.10: Screenshot of the dialog for online interactive user registration.

### Marking and Resizing Features

In order to create the database from the taken pictures, the positions of the eyes and the lips have to be known. Interactive registration requires the user to mark those positions at least once. This can be done by simply clicking in the image. A click with the left mouse button in the upper half of the image is interpreted as the position for the right eye (which appears on the left side of the camera image). Accordingly, a click with the right mouse button in the upper half of the image is interpreted as the position for the left eye. Clicking in the lower half of the image with either mouse button will mark the lips.

Each click is supposed to be made at the center of a feature. Then a default size for the eyes or the lips is applied to mark a rectangle around this center point. This size can be changed in case the rectangle is too big or too small.

Marking and resizing of the feature regions is possible in camera preview mode, where they will be tracked as explained below, and in the editing mode to correct them if necessary.

### Tracking Features

To make it unnecessary to click on each feature in each image, feature tracking (as described at the end of 3.1.6) is used to follow the features once they have been marked in camera preview mode. Two different tracking modes are available.

In the first tracking mode, the eyes and the lips are tracked independently from each other. This offers maximum flexibility to follow the features in spite of head tilt or rotation. The biggest disadvantage is that the used tracking algorithm cannot cope with closing and reopening of the eyes. When the eyes are closed, it usually follows the eyelashes instead of remaining at the same position. When the eyes are opened again, the algorithm often fails to find the correct position again and tracks a lower region instead.

The second tracking mode tracks only the lips and assumes that the eyes stay in the same position relative to the lips. In this mode, clicking for a new lip position also relocates the eyes accordingly. This approach has several advantages. It is faster because only one rectangle has to be tracked instead of three. It also completely avoids the problem of how to cope with closed eyes. Finally, it is more convenient to use because only one mouseclick is

necessary to update the feature positions when the tracker loses them. On the other hand, this tracking mode cannot produce exact results when the head is tilted or turned because these movements change the relative position of the lips and the eyes.

Both tracking modes can only handle slow movements of the head. However, this depends on the frame rate provided by the used frame grabber and on the computing power of the machine on which the program runs.

At any point, the pictures taken so far plus the feature marks can be saved to disc. An image database will be created from all the information which is already available. Previously saved data can also be loaded anytime to continue working on it.

### 3.2.3 Offline Interactive Mode

The offline interactive user registration can be used to correct the results of the automatic registration. It can also be used to create a new avatar face from an arbitrary image sequence, provided this sequence contains the necessary image information about eyes and lips.

A screenshot of the dialog window is shown in figure 3.11. This section describes, which features this dialog offers.

Each image in the sequence can be viewed separately. The dialog window display the number of the image in the sequence, the total number of images in the sequence, which parts of the images are currently selected for storage in the database, what the x and y positions of the eyes are, and which viseme is visible.

For the offline interactive registration, not as many assumptions have been made as have for the online interactive registration. Therefore, marking of the facial features is a little bit more complicated. First, the user has to select the position of which feature shall be edited. Possible selections are the whole face, the right eye, the left eye, and the lips. In the image, the feature selected for editing is displayed in red. The other features are displayed in white. When over the image area, the cursor changes to a cross to allow precise positioning. Clicking with the left mouse button marks one corner of the rectangle. The next click marks a second corner, so that the rectangle is defined. Clicking with the right mouse button within the rectangle marks a

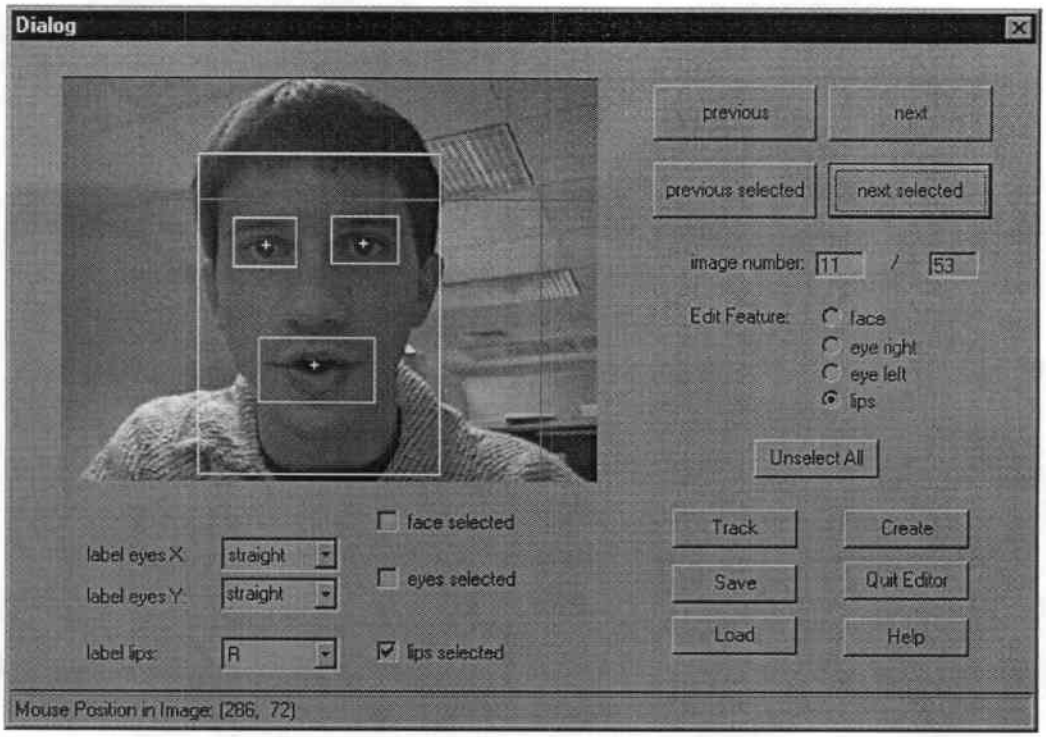


Figure 3.11: Screenshot of the dialog for offline interactive user registration / editing of a database.

special point. In the case of an eye, this point marks the pupil. In the case of the lips, it marks the lip center. Currently, marking the pupil position is not necessary since the gaze direction has to be entered manually anyway.

When the eyes and the lips have been marked in one image, they can be tracked through the rest of the image sequence. For this purpose, the tracking algorithm (see 3.1.6) is used. Unlike in the dialog for online interactive registration, the eyes and the lips are always tracked independently. Should the tracking algorithm fail at a certain point, the positions can be corrected in the corresponding image and tracking can be resumed from there.

The user has to select manually, which eye and lip images for the database shall be taken from which image in the sequence. The dialog offers checkboxes to make this selection. One image must be selected as the base face for the avatar. When selecting the eyes, their x and y direction (or that they are closed) must be chosen. Accordingly, for each selected lips image, the correct

phoneme has to be chosen. This can also be done for images, where the features are not selected for the database, but it is not necessary.

For the feature that is selected for editing the position, it is possible to jump to the next image in the sequence, where this feature is selected for database storage. This allows to easily check whether something (an eye position or a viseme) is missing or selected more than once. For instance, when the lips are selected for position editing, it is possible to click through all the lip images currently selected to be viseme images for the database.

All the information about positions, selections, and feature labels can be saved or reloaded any time. When everything is done, the database can be created. All images that were in the database before are deleted to avoid inconsistencies.

## 3.3 Conversation Mode

In conversation mode, MaxAvatar displays the avatar face according to the currently selected databases for the base face, the eyes and the lips. These selections can be changed while the face is displayed. This mode is used in the context of an application described in section 4.1.

As long as nothing happens, the avatar is animated randomly. It looks around and blinks once in a while. If the connection to the server, which is supposed to deliver text to speak, is not established yet, MaxAvatar constantly tries to make this connection in the background.

When the connection exists and text is received from the server, a movie of the avatar speaking that text is immediately synthesized and displayed.

If MaxAvatar gets connection to the microphone direction server (see section 4.1.4), the avatar looks into the corresponding direction when somebody speaks. After a few seconds, it starts looking around again. When the next sentence is synthesized, it is synthesized with eyes that look in the direction where somebody was speaking most recently. The text received by the avatar is likely to be a responese to what was last said, so the person who spoke last should be addressed.

Another feature for conversation mode, is "stretching". When "stretching" is turned on, the avatar is displayed with the biggest size possible without distorting the aspect ratio inside the main window.

## 3.4 Options

This section gives an overview over the options that allow to configure various aspects of MaxAvatar. They are automatically saved in the parameter file (see 3.1.8) and loaded again when the program is started the next time.

- *Face settings.* The base face image, the database for the right eye, the left eye and the lips can be chosen independently.
- *Conversation mode settings.* The speed of the eye movements can be changed, and also the probability for eye movements, which is evaluated each time the animation function is called. There is also an option to display the received microphone direction information graphically. It can be selected whether the synthesized movie shall be played immediately or not. It may be useful to disable playing when MaxAvatar is used as a back-end image server, as in the example presented in 4.2.
- *Hostnames and port numbers.* The hostnames and port numbers can be configured for the connections mentioned in the section about the MaClientServer module (3.1.7).
- *Frame rates.* The frame rates for camera previewing and recording are configurable.
- *Default sizes for lips and eyes.* These sizes are used for online interactive user registration (3.2.2) and can also be modified there.
- *Console output.* A console window can be switched on and off. It can be very useful for test output.

## 3.5 Compatibility

Currently, MaxAvatar runs only under Microsoft Windows 95/98/NT. This is mostly due to the use of the Microsoft Foundation Classes (MFC) for the graphical user interface, and to the use of Windows or PC specific multimedia functionality.

It will be a considerable amount of work to make the whole system run on another platform. To make this task easier, MaxAvatar has been split into three parts:

- *MaxAvatarLib.* This library contains all the modules that do not use MFC. It should compile on almost any platform without big problems. All the input and output (except file I/O) is done through MaIO, which is part of MaxAvatarLib_MFC. Thus, MaIO has to be changed to meet the needs of the other platform.

- *MaxAvatarLib_MFC.* This library contains all the modules that use MFC. They deal with windows, dialogs, windows specific socket functions, and multimedia.

- *MaxAvatarGUI_MFC.* This is the main program, also using MFC. It uses and coordinates the other modules, and it contains the main menu.

In order to use MaxAvatar on another platform, the latter two parts have to be modified or reimplemented. With the help of state-of-the-art development tools, rebuilding the graphical user interface should be relatively easy. However, porting of the user registration will require considerable effort, since implementing of audio and video handling and synchronization may be quite different (or not possible at all) on another operating system.

## 3.6 Additional Experiments

The first steps towards a talking-head were experiments with a morphing program under Unix, written in Tcl/Tk. It allows to manually mark lines in two faces. With the help of these lines, one face can seamlessly be morphed into the other face. The major disadvantage was the lack of a program to determine these lines automatically, which makes the registration of a new face a tedious manual task.

Another experiment was the control of a turnable head with the input of a gazetracker [SMY97a]. By detecting the facial features, the gazetracker is able to estimate the head rotation in x, y, and z direction within certain angles. We used these angles to determine the corresponding view of the artificial head, so that it was moving as the person in front of the camera. Head rotation is currently not part of MaxAvatar (see section 3.1), but it will play an important role in future extensions.

# Chapter 4

# Applications

In this chapter, two existing applications for MaxAvatar (see chapter 3) are presented. In the first application, MaxAvatar provides audio-visual speech output as a front-end for a teleconferencing scenario involving speech recognition and translation. In the second application, MaxAvatar plays the role of a back-end server for an Internet chat applet.

## 4.1 Face Translation

More often than ever before in history, people with the background of different cultures and languages want to communicate with each other. This phenomenon is both caused and followed by more efficient means of transportation, the globalization of economy and media, international politics, tourism, and modern forms of communication like the telephone, teleconferencing systems and, last but not least, the Internet. The most straightforward way to overcome language barriers in communication is of course to learn to speak and understand the other language. However, in today's fast paced society and business, this is often not feasible. Another solution is translating or interpreting. But translators and interpreters are highly specialized people, which means they are expensive and not always available.

In order to have cheap and instant translation for everybody, efforts are made to employ computers for this purpose. Speech translation is a very complex problem and it has only partly been solved. Nevertheless, there are already

speech translation systems available. One of them is the C-STAR II system, presented in section 4.1.2. It is not only capable of translating text from one language into another, but it also provides the possibility of complete communication using spoken language. This could be compared to a telephone with a built-in interpreter.

Face translation expands this idea from audio communication to audio-visual communication. Using the avatar system as described in chapter 3, the face of the communication peer can be seen with lip movements that are synchronized to the translated speech. Thus, everybody can not only hear, but also see the other one talk in his/her own language.

The C-STAR system uses the limited domain of a travel planning scenario to demonstrate the usefulness of speech translation. Several languages are integrated into the system and it is designed to be easily expandable with additional languages. Here, we will use the example of a travel agent, who only speaks German, and a client, who only speaks English. The client wants to book a flight to Germany and can speak to the travel agent with the help of the C-Star system. Through the avatar, the client can also see the travel agent talking in English. In the current setup, translation and avatar output is only done from German to English. In principle, however, the setup can easily be extended to allow translation in both directions.

### 4.1.1 System Overview

Figure 4.1 gives an overview over the Face Translation system and its components. The travel agent and the clients do not speak the same language. Therefore, the travel agent's speech is recognized and translated by the Janus III system (see 4.1.3). The resulting text is sent to MaxAvatar, which in turn sends it to the text-to-speech synthesizer Festival (see 4.1.5). Festival returns the speech wave form, the phonemes it contains and their time alignment. With this data and with the information it got from the microphone direction server (see 4.1.4) about which of the clients spoke last, MaxAvatar retrieves the appropriate images from the image database and synthesizes the video part of the avatar.

The result is a movie where the avatar looks at one of the clients (or just straight, if nobody spoke) and says what the travel agent said, but in the clients' language. This movie is automatically played at the clients' side.

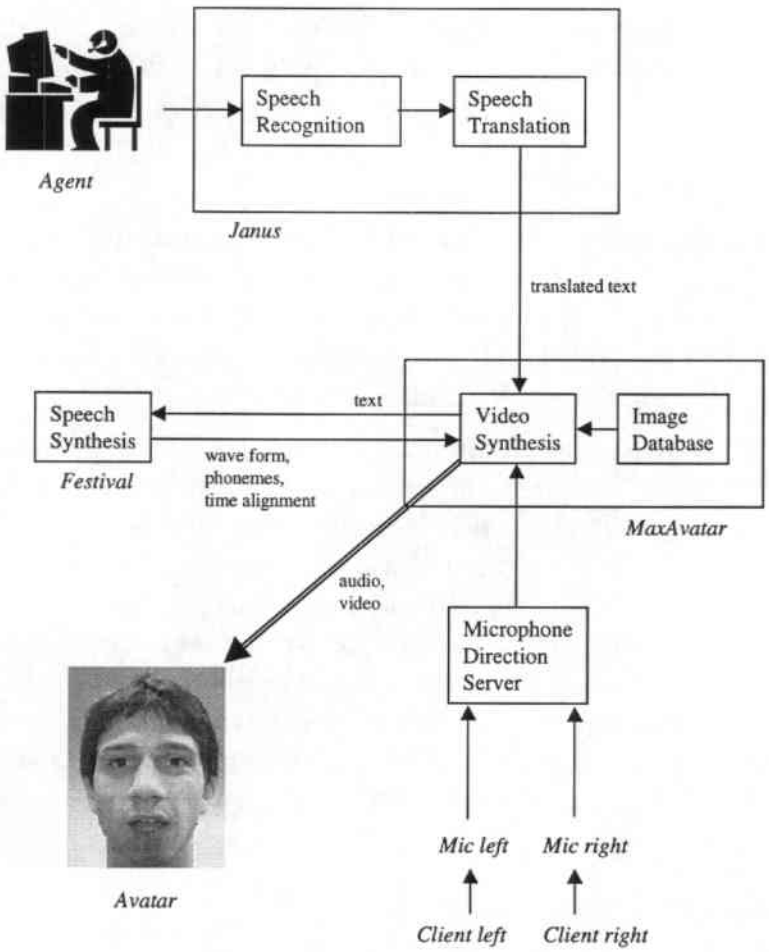


Figure 4.1: The components of Face Translation.

The travel agent and the clients can exchange information through a web browser. For example, the agent can send a calendar page, information about train or flight schedules, hotels and cities. The clients are able to mark preferred dates on the calendar, browse through pictures and panorama views of the location they plan to visit, etc. The avatar can either be displayed in a separate window, or as a part of the communication web page.

### 4.1.2 C-STAR II System

This thesis has been done in the framework of the C-STAR II project, where C-STAR stands for "Consortium for Speech Translation Advanced Research." The goal of this consortium is to cooperate in the research on speech-to-speech translation [Wai96, LGLW98, Wos98, WBG+98].

The former consortium (C-STAR) successfully demonstrated international automatic interpreting telephony in a series of bilingual experiments in January, 1993 using Japanese, English, and German.

These tests were successful, but required a limited way of speaking (e.g., grammatically correct sentences and clear speaking style, etc.). The C-STAR II research aims to reduce such limitations as much as possible, i.e., C-STAR II is aiming at more robust speech interpreting systems which can recognize ordinary spontaneous speech, translate it into other languages, and generate synthesized speech. Moreover, C-STAR II focuses on multi-lingual translation in larger-scale tasks.

The C-STAR II consortium consists of six partners and several affiliate groups from ten countries. The partners in C-STAR II are: ATR, Japan; ISL, University of Karlsruhe, Germany; ISL, Carnegie Mellon University, USA; ETRI, Korea; IRST, Italy; CLIPS-GETA, France.

All C-STAR II partners commit to build a full end-to-end system, each handling their own language as input and producing at least one output language. The affiliates perform relevant basic research, participate in workshops, and can optionally build individual components in cooperation with one or more of the C-STAR II partners.

Since the six main partners are responsible for one input language each, the C-STAR II setup allows recognition and translation for a minimum of six languages: English, German, French, Japanese, Korean, and Italian.

For the development of all systems, a common task is defined to facilitate the combination of individual systems in a prototype translation system. This prototype system shows the application of speech-to-speech translation in the domain of travel planning, which contains a rich structure of sub-domains, e.g., hotel reservation, transportation, sightseeing, and scheduling.

#### The Interchange Format

C-STAR uses an interlingua for machine translation of spoken travel planning dialogs, which is called interchange format (IF). Its purpose is to facilitate translation between many language pairs with minimal effort. Sites that wish to use IF supply an analyzer that produces IF from sentences in the their own language, and a generator that takes IF as input and produces sentences in their own language. Translation from one language to another can be accomplished by taking the analyzer from the first and the generator of the second language. This makes it relatively easy to add new languages to the system, both as an input and as an output language.

The interlingua approach abstracts from different ways of expressing the same thing or culturally dependent phrases. It recognizes and translates the underlying meaning or intent of an utterance. Utterances which have been analyzed and translated into IF can be distributed to all connected partners. Thus, analysis has only to be performed once, and every site can generate the translation in the desired destination language directly from IF.

### 4.1.3 Janus III

The Janus speech translation system [Wai96, OAM+92, GSB+95, LGLW98, WBG+98] was developed in the Interactive Systems Laboratories at Carnegie Mellon University and University of Karlsruhe. It is written in C and offers a Tcl/Tk interface. Janus translates spoken language, much like a human interpreter. It operates on a number of limited domains such as appointment scheduling, hotel reservation, or travel planning. The Janus project is aiming to make human-to-human communication across language barriers easier. Unlike a human interpreter, Janus can also access databases to automatically provide additional information such as train schedules or city maps to the user. All systems are designed to work without a conventional keyboard. Whenever speech input is insufficient, other modalities such as handwriting and gesture recognition can be used to create a user friendly interface.

To provide translation for conversational speech, the system has to handle fragmentary, errorful and disfluent language and heavily coarticulated and noisy speech. Instead of literal translation it has to provide useful interpretation of a user's intent. For the discourse domain of human-to-human appointment scheduling negotiations a vocabulary size of 3000 to 5000 words

was observed, depending on the language. Perplexities in this task range between 30 and 70. The system runs in less than two times real time.

Janus III was designed for multi-party, multi-lingual conversations between travel agents and their clients. It uses an interlingua-based approach (see 4.1.2) and consists basically of three modules: speech recognition, analysis, and generation. The analyzer and the generator are language-independent, i.e., they consist of a general processor that can load language-specific information. Therefore, it is relatively easy to add new languages. Janus III uses semantic grammars, which provide accurate translations for limited domains. For these limited domains, they can be developed relatively fast, but expanding them to cover new domains requires a lot of effort. To cope with this problem, modular grammars and common libraries are used.

There are multiple grammars for the different sub-domains of the travel planning task. The Soup parser [Gav98, GW98] is used to analyze input using these multiple grammars concurrently. Soup is a stochastic, chart-based, top-down parser, specially designed for parsing spoken language utterances with very large, multi-domain semantic grammars in real time. It is written in C++.

Figure 4.2 shows how text is translated from one language into another with the Janus system. The speech recognizer sends an N-best list of speech hypotheses as text in the source language to the Soup parser, which analyzes it. The Parser-to-IF mapper converts the output from Soup into the interchange format. From there, the IF-to-Gen mapper and the Phoenix generator produce the appropriate text in the target language. This text is synthesized to speech by a text-to-speech synthesizer like Festival (see 4.1.5). The synthesizer delivers an audio file, the phoneme list, and its time alignment to the avatar program, which uses this information to produce a face, which speaks the text in the target language.

### 4.1.4 Microphone Direction Server

The microphone direction server helps to detect the message target, i.e., who should be addressed by the avatar. This is accomplished by evaluating the input from several microphones, which can be thought of as a virtual microphone array.

For the demo setup, we currently use only two microphones, which are located

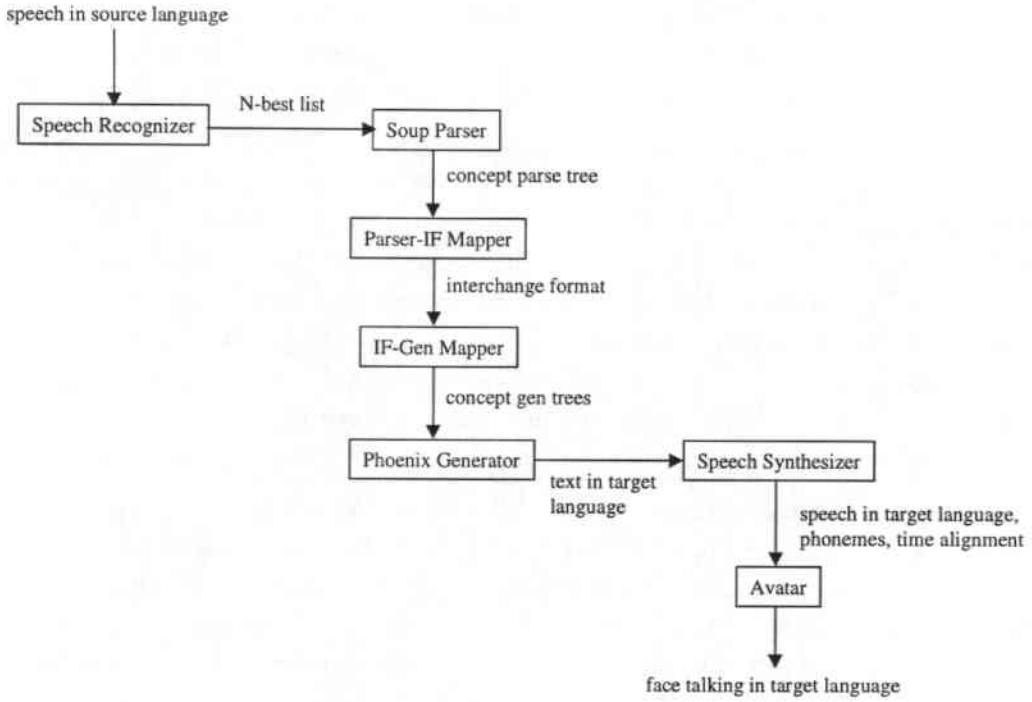


Figure 4.2: The Janus III machine translation system plus the avatar for audio-visual output.

on a conference table. The microphone direction server informs the avatar program which of the microphones gets the input. Since the location of the microphones is known, the avatar can look in the appropriate direction immediately to show attention, or later when it speaks a sentence to address the correct person.

The program used as a microphone direction server is also capable of learning to distinguish the voices of several people. During a conversation, it can then recognize who is speaking. This information is currently not used by the avatar. However, in the future it might be useful to address certain persons by their name.

### 4.1.5 Festival

Festival is a general multi-lingual speech synthesis system developed at the Centre for Speech Technology Research (CSTR) at the University of Edin-

burgh. It offers a full text-to-speech system with various APIs, as well an environment for development and research of speech synthesis techniques. It is written in C++ with a Scheme-based command interpreter for general control and uses the Edinburgh Speech Tools for low level architecture. Festival is multi-lingual (currently English (British and American), Spanish, and Welsh), although English is the most advanced. It contains several externally configurable language-independent modules including phonesets, lexicons, letter-to-sound rules, tokenizing, part of speech tagging, intonation, and duration. The built-in waveform synthesizer is diphone-based. The use of other waveform synthesizers, for example MBROLA [Dut97], is supported. There are also various voices which can be selected for synthesis output.

Festival also offers a client/server interface, which we use to initiate speech synthesis and collect the results as soon as they are available. The results that are needed to synthesize the lip movement of the avatar are the synthesized wave form, the corresponding phonemes, and their time alignment.

### 4.1.6 MaxAvatar

The last component of the Face Translation application is MaxAvatar, which was presented in chapter 3.1. It provides audio-visual output for the translated speech. MaxAvatar communicates with the other modules described in this chapter, in order to create an avatar that speaks the translated text and looks into the right direction.

The other components can be connected and disconnected dynamically. As long as there is no input about the microphone direction, the avatar looks straight when it talks. When nothing is happening, it looks around a little bit (see also section 3.3). Without connection to the Janus system, text can still be entered manually. Of course, if no text-to-speech synthesizer is accessible, no new text can be synthesized.

Figure 4.3 shows an example of how an image sequence is generated for the translated text. One of the clients asks a question in English. The agent answers in German. The answer is translated, and an avatar image sequence is generated, also taking into account which of the clients asked the question.

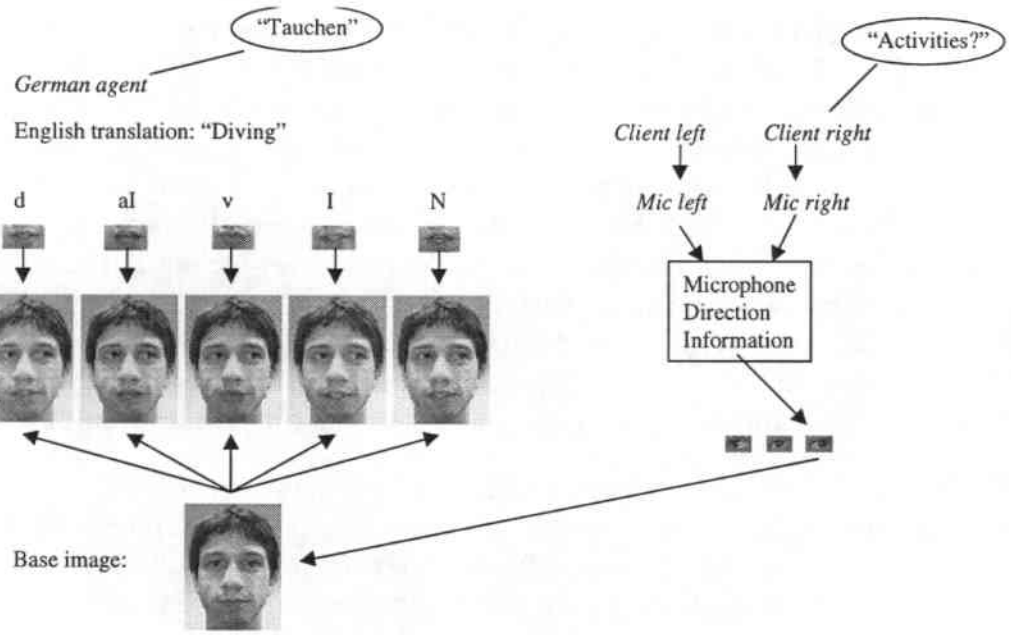


Figure 4.3: An example for Face Translation.

## 4.2 Web Avatar

Internet chat has become a very popular application. People enjoy connecting on this new world-spanning medium, meeting new people virtually and discussing various topics. Chat usually works with plain text as input and output. The Web Avatar introduces the idea of how the chat interface can become multi-modal.

### 4.2.1 System Overview

Figure 4.4 shows the components of the Web Avatar. In the current version, several users can download the chat client applet from the http server and start it in their web browsers. The applet then connects to the chat server, the image information server, and the file server. The chat server is responsible for broadcasting the text that one user types to all the other users. The image information server and the file server (see section 4.2.2) provide all the information necessary for the chat client to display the sentence.

Through the interface of the applet, a user can select a face from the face

database of MaxAvatar. Text can be entered, which will then be synthesized as an avatar movie and sent to the other users, where it is displayed automatically. This means that every user still has to type to input text. But the messages of other people are spoken by a talking-head, whose appearance can be chosen individually. Currently, users choose the face they want to see talking in their browser, which is sufficient for communication between two users. In a future version, the user should instead be able to choose the face that appears in the browsers of all the other connected users when he/she sent a message.

The images for the avatar face are relatively small in size (typically less than 50 kB) and have to be transmitted only once. Afterwards, only the information for lip synchronization has to be sent; i.e., which images have to be displayed when. The synthesized sound file has to be sent for each sentence. Currently, the wave format is used, which has two disadvantages. The files are relatively big (typically between 20 and 100 kB), and playing them requires a system-dependent plug-in for the browser. We plan to switch to sun audio files, which are smaller and better supported for this purpose.

### 4.2.2 MaxAvatar as a Server

In the Face Translation application described in section 4.1, MaxAvatar functions as a front-end module and gets its data from other components of the system. In the case of the Web Avatar application, MaxAvatar plays the opposite role. It generates the avatar in the background and delivers the results to the client applets, which are responsible for displaying it.

#### Image Server

This server enables other programs to control MaxAvatar remotely. They can request information about available faces in the database, about available images for these faces, and about which images to use at what time to display a certain sentence. The client can also select the face settings for the base face, the eyes, and the lips. The client can tell MaxAvatar to synthesize a certain text or to start or stop waiting for text from the C-STAR server.

Following is a list of supported requests and messages plus brief descriptions:

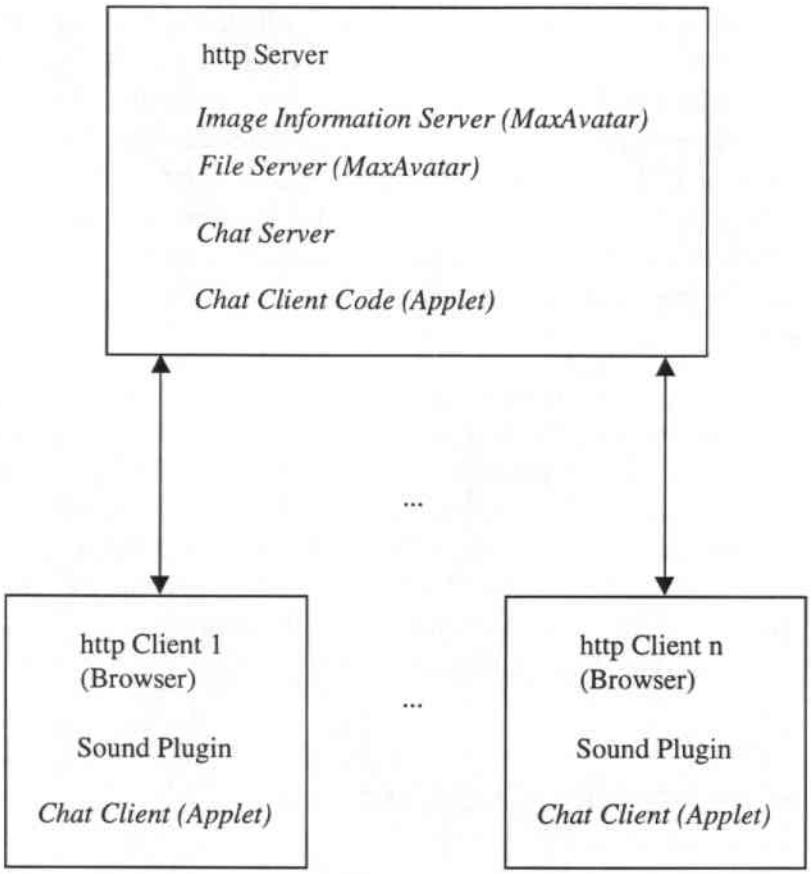


Figure 4.4: Components of the Web Avatar and their relationships.

- *init:* This tells MaxAvatar to load the images required for video synthesis into memory, according to the current settings.

- *getFaceList:* A list of all the available faces for the avatar is returned.

- *getImageRef:* Information about all the available images according to the current face settings is sent back to the client. For each available image a URL is sent. This URL can be used to retrieve the actual file through the file server (see next paragraph).

- *showInit:* Sends back information about what to show as an inital face.

- *show:* After a "show" request, a "show" message is sent, providing information about how to play a synthesized sentence. The corresponding synthesized wave file is copied to a temporary file in case the sentence, including sound, is to be displayed later.

- *dispose:* Tells MaxAvatar that a certain wave file that has been stored for later use is not needed any more. It will be deleted.

- *cstarServer:* This allows to change the hostname and the port number of the C-STAR server.

- *startListeningCstarServer:* Makes MaxAvatar go into "conversation mode". That means, text received from the C-STAR system is sent to Festival to synthesize speech, and then a sequence of images with synchronized lip movement is generated. As soon as the synthesis is completed, a "show" message is returned.

- *stopListeningCstarServer:* Makes MaxAvatar go out of "conversation mode". Even if more text is received from the C-STAR system, it will not be synthesized, and no "show" messages will be sent.

- *synthesizeText:* Text can be sent to MaxAvatar. It will be synthesized and a "show" message will be returned with the information about how to display that text.

- *selectFace:* This lets the client select separately from which of the available faces the base face image, the eyes, and the lips shall be taken when the avatar is synthesized.

### File Server

On a separate port, MaxAvatar also provides the functionality of a file server. This enables clients to download the image and audio files needed to display the avatar. The image server described in the previous paragraph only provides the URLs to these files. If an http server is running on the same machine as MaxAvatar, the client can use this http server and the URLs to retrieve the files. Otherwise, it can be done through MaxAvatar's file server.

When the requested URL does not exist, an error code is returned. Otherwise the filesize, the filename, and the file contents are sent over the socket.

### 4.2.3 Possible Extensions

Right now, the web avatar can be used for chat over the Internet with text input on one side and audio-visual output in the same language on the other side. This application can be extended in several ways.

One possible extension would be the use of a microphone for speech input. The speech input can be analyzed on a phoneme level (phonemes and time alignment) to be able to generate an avatar video synchronized to the original utterance. The result would be low bandwidth teleconferencing through a web browser, where people can hear each other's original voices but see a synthesized version of the face.

A next step could be to completely recognize the spoken input sentence and to use the resulting text to synthesize it again for speech output. This way, people would not only be able to choose a face for the avatar, but also a voice if the used speech synthesizer allows to do so.

Finally, instead of speech recognition, speech-to-speech translation could be performed. This would lead to a face translation (see 4.1) application available to everybody on the Internet. Everybody with a microphone and access to the Internet could communicate with people from all over the world, speaking and hearing his/her own language.

All these extensions would allow low bandwidth teleconferencing through web browsers without even having to use a camera. Only a microphone is necessary. The additional use of a camera would yield further possibilities as for instance registration of a new user's face while he/she is using the system, i.e., without an explicit registration process.

# Chapter 5

# Summary and Future Work

## 5.1 Summary

A complete front-to-end talking-head system (MaxAvatar) was designed, implemented, and presented in this thesis. The system has the following features:

- *2D head model.* The underlying head model uses a 2D approach. This minimizes the effort for adding a new face, but also limits the possibilities for head movement.
- *Real human face.* The avatar's face can be chosen from a database of recorded real human faces or a new one can be added. This serves the goal of computer-supported human-to-human communication.
- *Lip animation.* Lip movement can be synthesized so that it matches a speech utterance. This utterance can be created from arbitrary text input. The lip movement is generated by stitching appropriate lip images into the base face and displaying the resulting image sequence according to timing information. Approaches for smooth transitions between different lip images have been tested, but the results leave room for improvement (see sections 3.1.3 and 5.2).
- *Eye animation.* Controlled eye movements can also be generated by stitching appropriate eye images into the base face. This enables the avatar to address certain persons by looking in their direction.

- *Quick and easy-to-use user registration.* A new user can add his/her face to the database following an easy-to-use GUI-based process. Three different ways of user registration have been implemented (see section 3.2). The concept of a fully automatic user registration has been demonstrated, but it does not work well enough yet because the feature detection module is not reliable enough. However, as the system is currently designed, it takes less than ten minutes to add a face with one of the other two presented methods.

- *Client/server interface.* MaxAvatar can receive text to be synthesized and information about the speaker's location through a client/server interface. In addition, clients can remotely connect and control much of MaxAvatar's functionality. Clients themselves can also request all the information necessary to display the avatar.

- *Modular program structure.* An object-oriented, modular program structure has been chosen to facilitate future improvements and extensions.

The features described above have been demonstrated by using MaxAvatar for audio-visual output in a speech-to-speech translation application (Face Translation, see section 4.1) and in an Internet chat application (Web Avatar, see section 4.2).

Most of the current talking-head research projects focus on perfecting certain aspects of a talking-head, e.g., coarticulation or facial expressions. The goal of this thesis, however, was to show the feasibility of a complete avatar system that can easily be personalized and used by everybody for human-to-human communication. This goal has been reached. Optimizing the single components or integrating better or additional components is the next step.

## 5.2 Future Work

The system presented in section 3 is a prototype system in many respects. Instead of trying to optimize single components, much effort was directed towards the goal of having a complete front-to-end avatar with a convenient graphical user interface and a structure that facilitates changes. This section lists possible and desirable extensions and improvements for future versions.

### Viseme Selection

For user registration, a way has to be found to select exactly the images that contain the viseme images. As mentioned in section 3.2.1, this could be accomplished through correlation with an existing set of lip images or the use of a lip shape recognizer.

For synthesis, the selection of the best lip images for the image sequence can be improved significantly. One way to do this is the use of context information. Currently, only the corresponding phoneme is regarded in choosing the viseme image. The correct image strongly depends on previous and following phonemes, so they should be taken into account, as well. A step in this direction would be the use of trisemes, which would take into account the current, the previous, and the next phoneme when selecting an image. However, there are cases when trisemes are also insufficient. For example, in the word "enclave" it is not enough to know that the K phoneme is followed by an L phoneme to select the right lip image. In the word "enclose" the K phoneme has the same preceeding and succeeding phonemes, but the viseme should be different. It depends on the next vowel rather than on the next phoneme.

Furthermore, viseme boundaries are not necessarily the same as phoneme boundaries, although these are a common approximation. Building a realistic viseme synthesis might require letting go of relying on phonemic transcriptions and creating separate dictionaries to map words into sequences of visemes.

### Viseme Transitions

Another point to be improved is the transition between different lip images. Right now, by default, only one lip image is used for each phoneme in the synthesized speech. This results in coarse transitions, especially when consecutive images are very different. More naturally looking results could be produced by calculating transition images between different lip images. MaxAvatar contains experimental functions for this purpose (see section 3.1.3). Their results are promising, but not yet satisfying. One of them relies on precise knowledge about the lip position and shape in each lip image. At the moment, we have no possibility of generating this information automatically, and hand labeling would require too much effort. Further work in this direction would require either a different approach to calculate these images

or a way to find exact information about the lips automatically. The other approach for lip transitions is significantly simpler with similar results as the first one. However, it is not clear how much the results of this approach can be improved.

### Head Movements

The current system only supports one base image for the face; i.e., the face itself does not change or move. Only the eyes and the lips are animated. An advanced version of the avatar should also be able to move the head in a realistic, more lively way. Most of all, the head should be able to turn left and right, which would also enhance the ability to address a certain person in a conversation. However, this will require to move away from the 2D approach towards a 3D or 2.5D approach, as described in section 2.3.

### Facial Expressions

To make the output look more realistic, the avatar should also be capable of several facial expressions. A first approach towards this goal could be the use of more lip and eye images, e.g., showing a smile. In the long run, modeling facial expressions requires manipulations of almost all parts of the face. The problem of encoding these movements and acquiring the corresponding images will have to be solved. Another problem in this context is how to determine when the avatar shall have which facial expression. If it shall reflect the facial expression of the person who is represented by the avatar, this person's facial expression has to be recognized first. This is a complex and interesting field of research in itself.

### Speed

With the exception of the option to precalculate face images (see section 3.1.3), MaxAvatar has not been optimized for speed. However, for real-world applications like communication, speed is critical. A significant amount of processing time can be saved by optimizing the graphics algorithms used or by using techniques like precalculation of images. Nevertheless, there will be a point where a compromise has to be made between quality and speed. For example, when smooth transitions between lip images, head movements, or different facial expressions shall be calculated, it is not feasible to precal-

culate all the possible face images and store them in main memory for quick access. On the other hand, newer and faster processors will provide greater speed for image calculations.

### Automatic User Registration

A first approach for automatic user registration has been implemented (see section 3.2.1). To make it better for everyday use, two aspects have to be improved. First, a more stable and precise automatic feature detection is necessary to reliably locate facial features in the recorded images. Second, the classification for lips and eyes has to be improved so that the correct images are chosen for the database.

With a fully functioning automatic user registration, further options become available. If it is feasible to have a camera constantly taking pictures of the user while he/she is working, the image database for his/her face could be built and improved immediately without any effort on his/her part and without any explicit registering process.

### Feature Detection and Modeling

As mentioned above, feature detection and modeling is a very important part of building an avatar that is convenient to use. It allows automatic extraction of facial feature images, as well as synthesis and manipulation of output images. Since even a single image in the database that is only a few pixels off can lead to disturbing results, it is crucial to have a very reliable and precise feature detection module. The currently implemented version leaves a lot of room for improvement.

### System Independence

It would be good to have an avatar that runs on every platform. This would make the program available for a wider range of users and applications. As explained at the beginning of section 3.1 and in sections 3.1.8 and 3.5, Max-Avatar currently only runs under Microsoft Windows. The possibility of porting it to other systems had been taken into account during the design phase. However, the complete graphical user interface would have to be rewritten. Also, system-independent support for multi-media functionality,

# List of Figures


3.1 The modules of MaxAvatar and their relationships. . . . . 22

3.2 Base face, right eye, left eye and lip images from an example database. . . . . 27

3.3 *Synthesizing of an avatar face.* Eye and lip images are stitched into a base image. First, they are only copied, then scaling, intensity adaptation and edge smoothing are applied. On the left, all images are from the same database. On the right, eyes and lips are taken from a different database than the base face. 30

3.4 Lip transition images with Quadrilateral-to-Quadrilateral mapping. Left: the previous image. Right: the next image. Middle: the calculated transition image. . . . . 31

3.5 Lip transition images with linear combination. . . . . 33

3.6 *Background subtraction.* a) the original image, b) motion over a few frames, c) applied noise filter to motion image, d) detected head outline, e) solid head mask, f) the original head image with a blue screen background . . . . . 40

3.7 *Face detection.* a) the original image, b) face color, c) clustered face color, d) solid face mask, e) the original face image with a blue screen background . . . . . 42

3.8 Original face image and detected features. . . . . 43

3.9 Top: the original lip image. Below: Derivation of the lip image in x direction (on the left side) and in y direction (on the right side) plus the detected lip boundaries. . . . . 47

3.10 Screenshot of the dialog for online interactive user registration. 53

3.11 Screenshot of the dialog for offline interactive user registration / editing of a database. . . . . 56

4.1 The components of Face Translation. . . . . 62

4.2 The Janus III machine translation system plus the avatar for audio-visual output. . . . . 66

4.3 An example for Face Translation. . . . . 68

4.4 Components of the Web Avatar and their relationships. . . . 70

# List of Tables

2.1 English phonemes used by Janus . . . . . 10
2.2 German phonemes used by Janus . . . . . 10
2.3 Visemes used by Video Rewrite . . . . . 11

3.1 Implicitly used viseme classes and the phonemes they contain. 24
3.2 Example 1 for the mapping from text to available visemes . . 24
3.3 Example 2 for the mapping from text to available visemes . . 25

# Bibliography


[BCS97] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video Rewrite: Driving Visual Speech with Audio. In *Computer Graphics Proceedings, Annual Conference Series*, volume 8, pages 353–360, August 1997.

[BS98a] Matthew Brand and Ken Shab. Voice-driven animation, 1998.

[BS98b] N. M. Brooke and S. D. Scott. Two- and three-dimensional audio-visual speech synthesis. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 6, pages 213–218, December 1998.

[BTC98] Alan W. Black, Paul Taylor, and Richard Caley. Festival. `http://www.cstr.ed.ac.uk/projects/festival.html`, 1998. The Centre for Speech Technology Research (CSTR) at the University of Edinburgh.

[CM90] M. Cohen and D. Massaro. Synthesis of visible speech. *Behavior Research Methods: Instruments and Computers*, 4(22):260–263, 1990.

[DK98] Chris Davis and Jeesun Kim. Repeating and remembering foreign language words: Does seeing help? In *Proceeding of the CHI Conference, Los Angeles, CA (CHI 98)*, 1998.

[Dut97] Thierry Dutoit. Mbrola. `http://tcts.fpms.ac.be/synthesis/mbrola.html`, 1997. The Circuit Theory and Signal Processing (TCTS) Lab of the Faculte Polytechnique de Mons in Belgium.

[Gav98] Marsal Gavaldà. The soup home page. `http://www.is.cs.cmu.edu/ISL.speech.parsing.soup.html`, June 1998.

[GGMCB94] B. Le Goff, T. Guiard-Marigny, M. Cohen, and C. Benoit. Real-time analysis-synthesis and intelligibility of talking faces. In *Proceedings of the Second ESCA/IEEE Workshop on Speech Synthesis, New Paltz, New York, USA*, volume 3, September 1994.

[GMO98] Lei Gao, Yasushiro Mukaigawa, and Yuichi Ohta. Synthesis of facial images with lip motion from several real views. IEEE, 1998.

[GSB+95] P. Geutner, B. Suhm, F. D. Buø, T. Kemp, A. Lavie, L. Mayfield, A. E. McNair, I. Rogina, T. Sloboda, W. Ward, M. Woszczcyna, and A. Waibel. Integrating different learning approaches into a multilingual spoken translation system. In *Workshop on New Approaches to Learning for Natural Language Processing*, Montreal, Canada, August 1995. International Joint Conference on Artificial Intelligence.

[GUAT98] Francisco M. Galanes, Jack Unverferth, Levent Arslan, and David Talkin. Generation of lip-synched synthetic faces from phonetically clustered face movement data. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 4, pages 191–194, December 1998.

[GW98] Marsal Gavaldà and Alex Waibel. Growing semantic grammars. In *Proceedings of COLING/ACL-98*, 1998.

[HL98] Asa Hällgren and Bertil Lyberg. Visual speech synthesis with concatenative speech. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 3, pages 181–183, December 1998.

[KYVB98] Takaaki Kuratate, Hani Yehia, and Eric Vatikiotis-Bateson. Kinematics-based synthesis of realistic talking faces. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 6, pages 185–190, December 1998.

[LGLW98] Lori Levin, Donna Gates, Alon Lavie, and Alex Waibel. An interlingua based on domain actions for machine translation of task-oriented dialogues. In *Proceedings of the ICSLP 98*, Sydney, Australia, 30th November – 4th December 1998.

[MCB98] Dominic W. Massaro, Michael Cohen, and Jonas Beskow. Visible speech synthesis. NSF Workshop on Speech Synthesis August 6–7, `http://mambo.ucsc.edu/psl/dwm/nsfspeech/index.htm`, 1998. Perceptual Science Laboratory (PSL) at the University of California - Santa Cruz.

[McG98] Harry McGurk. Developmental psychology and the vision of speech. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 18, pages 3–20, 1998.

[MM76] H. McGurk and J. MacDonald. Hearing lips and seeing voices. *Nature*, 3(264):746–748, 1976.

[MN96] Youngme Moon and Clifford Nass. How "real" are computer personalities? psychological responses to personality types in human-computer interaction, 1996.

[Mor98] Shigeo Morishima. Real-time talking head driven by voice and its application. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 5, pages 195–199, December 1998.

[NKL98] Clifford Nass, Eun-Young Kim, and Eun-Ju Lee. When your face is the interface: An experimental comparison of interacting with one's own face or someone else's face. In *Proceeding of the CHI Conference, Los Angeles, CA (CHI 98)*, 1998.

[NMF+95] Clifford Nass, Youngme Moon, B. J. Fogg, Byron Reeves, and D. Christopher Dryer. Can computer personalities be human personalities? In *International Journal of Human-Computer Studies*, 1995.

[OAM+92] L. Osterholtz, C. Augustine, A. McNair, I. Rogina, H. Saito, T. Sloboda, J. Tebelskis, A. Waibel, and M. Woszczyna. Testing generality in janus: A multi-lingual speech translation system. In *Proceedings of ICASSP*. IEEE, 1992.

[Par75] F. I. Parke. Parametrized models for facial animation. *IEEE Computer Graphics and Applications*, 8(2):61–68, 1975.

[SMY97a] Rainer Stiefelhagen, Uwe Meier, and Jie Yang. Real-time Lip-Tracking for Lipreading. In *Proceedings of Eurospeech 97*, volume 4, 1997.

[SMY97b] Rainer Stiefelhagen, Uwe Meier, and Jie Yang. Tracking Eyes and Monitoring Eye Gaze. In *Proceedings of the Workshop on Perceptual User Interfaces (PUI97)*, volume 3, 1997.

[Sti96] Rainer Stiefelhagen. Automatische bestimmung von visemen für das maschinelle lippenlesen. Studienarbeit, February 1996.

[Wai96] Alex Waibel. Interactive translation of conversational speech. *Computer*, 7(29), July 1996.

[WBG+98] Monika Woszczyna, Matthew Broadhead, Donna Gates, Marsal Gavaldà, Alon Lavie, Lori Levin, and Alex Waibel. A modular approach to spoken language translation for large domains. In *Proceedings of the 1998 Conference of the Association for Machine Translation in the Americas (AMTA-98)*, 1998.

[Wol90] George Wolberg. *Digital Image Warping*, chapter 3, pages 52–56. Washington IEEE Computer Society Press, 1990.

[Wos98] Monika Woszczyna. The C-STAR II Homepage. `http://www.is.cs.cmu.edu/cstar/`, 1998.

[YSMW98] Jie Yang, Rainer Stiefelhagen, Uwe Meier, and Alex Waibel. Real-time Face and Facial Feature Tracking and Applications. In *Proceedings of Auditory-Visual Speech Processing (AVSP 98)*, volume 8, pages 79–84, December 1998.