# Multimodal Human-Computer Interaction

## The design and implementation of a prototype

M.J.A. André
August 1998

Department for Technical Mathematics and Computer Science
Knowledge Based Systems Group
Delft University of Technology

*Thesis committee:*
Prof. dr. H. Koppelaar
Prof. dr. ir. E.J.H. Kerckhoffs
Dr. drs. L.J.M. Rothkrantz

## Acknowledgments

This report is to be presented as my master's thesis. It is made possible with the help of the Knowledge Based Systems Group (headed by Professor H. Koppelaar) of the Department of Technical Mathematics and Computer Science of Delft University of Technology and the Center for Computer Aids for Industrial Productivity (headed by Professor J. Flanagan) at Rutgers University in New Brunswick, New Jersey.

I would like to thank Leon Rothkrantz and James Flanagan for creating the opportunity for me to study in the United States, and for supervising my work. Further thanks go to my direct supervisor Attila Medl, Ivan Maršić, and Kas Kulikowski for their valuable advice and suggestions.

Finally, I would like to thank my fellow CAIP colleagues for their help in different ways.

Maurits André, August 1998

**Abstract**

The influence of computers on people's daily lifes is increasing and the need for simpler interfaces to use computers is emerging. Current human-machine communication systems predominantly use keyboard and mouse inputs which inadequately approximate human abilities for communication. More natural communication technologies are capable of freeing computer users from the keyboard and mouse.

This work presents a prototype of a multimodal interface featuring fusion of multiple modalities for human-computer interaction. The three modalities integrated are a speech recognizer and synthesizer, a tactile glove, and a gaze tracker. The application used for this system is a collaborative whiteboard application extended to a military mission planning system. The design and implementation of the whole system and the methods applied are described and preliminary results of the real-time multimodal fusion are analyzed.

Concluded can be that this work presents a look into the future in which computers are controlled by modalities in the dimensions of sight, sound and touch. The input devices used in this project differ in accuracy, speed and efficiency. Together they form an appropriate means of communication with the system.

(This project is done at the Center for Computer Aids for Industrial Productivity (CAIP) of the Rutgers University in New Brunswick, New Jersey.)

# Contents

# Chapter 1

# Introduction

This is a report of my internship at the Center for Computer Aids for Industrial Productivity (CAIP). During my seven months stay, I conducted research at CAIP's Multimedia Information Systems Laboratory headed by Professor James L. Flanagan. This report is to be presented as my master's thesis at Delft University of Technology.

In the rest of this chapter the CAIP-center will be described and a short introduction on the project will be presented. Then, a background on the work done so far in the field of multimodal human-computer interaction is given. After that the goals and subgoals of my part of the project are defined which will result in a problem statement. Finally, an overview of the rest of this thesis is given.

## 1.1 CAIP

CAIP is short for Center for Computer Aids for Industrial Productivity. The Center was founded in 1985 and is one of the twelve highly focused Advanced Technology Centers in the United States of America. The Center is located on the Busch Campus of Rutgers University, the States University of New Jersey. The center is supported by the New Jersey Commission on Science and Technology, Rutgers University and a number of industrial members including AT&T Bell Labs, Digital Equipment, Corporation, Motorola, NEC, and Sun Microsystems.

The CAIP Center conducts forefront research in the application of computing technology and collaborates with industry and government to transfer the results of university research to societal benefit. A significant segment of CAIP's mission is to apply the technologies of high-speed scientific computing to the solution of industrial problems. From the human user's point of view, CAIP's mission is to make sophisticated computers easier to use. Current industrial applications span a broad range, and include telecommunications, manufacturing, chemical process control, software engineering, robotics, pharmaceutical evaluation, electronic circuit layout and testing, multimedia conferencing, and computer architectures. All have the common element of using high-speed computing to solve specific industrial problems.

With more than twenty distinguished faculty members, the CAIP research staff includes one Board of Governors Professor and three State of New Jersey Professors from the engineering, mathematics, and science department at Rutgers University, as well as ten full-time professional researches. Working alongside CAIP faculty and industrial researchers, top-notch graduate students actively participate in research and train for future engineering and science careers. A number of undergraduate students also participate in the center's activities.

The research staff is complemented by a highly qualified support staff of full-time technicians and programmers. In total, the Center has about 80 people from all over the world working in its offices and labs on a wide rang of equipment. The equipment includes around 100 workstations, a 512 nCUBE2S massively parallel computer, an IBM 8 node SP2 parallel computer, and lots of task specific equipment.

For more information on CAIP see [12].

### 1.1.1 Multimedia Information Systems Laboratory

The CAIP-Center contains 6 main labs as illustrated in figure 1.1. I did my research mainly in the Multimedia Information Systems Laboratory. The laboratory is headed by Professor James L. Flanagan and conducts research on human-computer interaction. Technical projects include speech recognition, speech synthesis, hands-free sound pickup, audio and video coding, distributed computing, database query by image content, and high-speed data networking. The primary focus of the laboratory is to make it easier for humans to communicate with complex computers and with each other, and to access the vast amounts of information stored online.

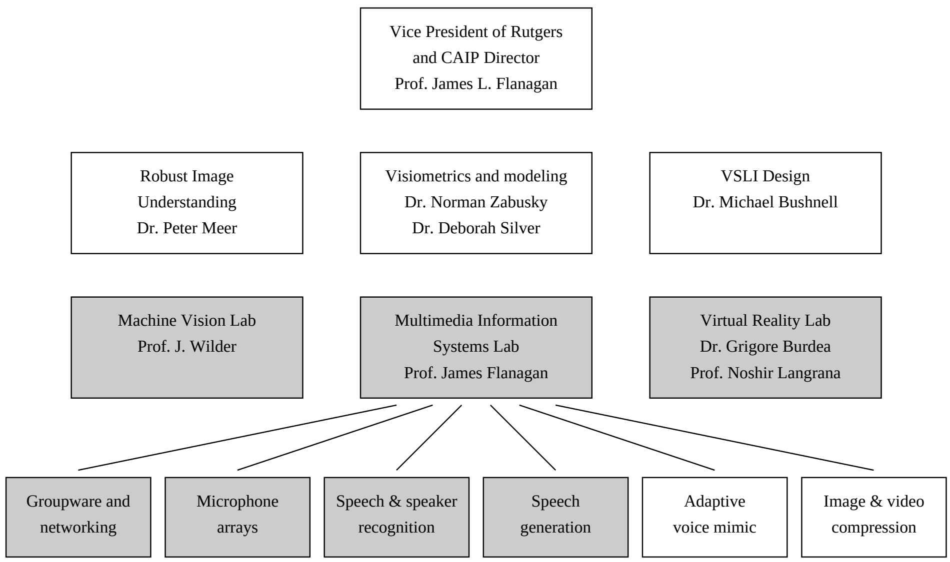


Figure 1.1: Organization of CAIP.

Since part of my project was to integrate different existing CAIP-developed technologies, I worked closely together with 4 of the Multimedia groups and with 2 other labs. (See the grey colored boxes in figure 1.1.) The Multimedia groups include Speech Recognition, Speech Generation, Microphone Arrays, and Groupware & Networking. The 2 laboratories are the Machine Vision Laboratory (Prof. J. Wilder) and the Virtualy Reality Laboratory (Dr. Grigore Burdea). My supervisor was Dr. Ivan Maršić and my direct advisor was Attila Medl.

## 1.2 Multimodal Human Computer Interaction

The way in which humans interact with computers has changed a lot over the past few decades. In the past special personnel was trained to operate the "machine". Scientist and researchers were the only people using this new machine and its computational power and speed to solve mainly mathematical problems. First, the computer was programmed by changing connectors and cables, comparable to a telephone switchboard. Then, the *stored program* principle allowed programmers to store their programs on punched cards and tapes. Usually, only one computer was available and researchers had to ask for computing time in which they were able to let the computer execute their programs. The following day the researcher would get the output of the execution. In case of an error the computer would produce wrong results or simply terminate the execution.

Nowadays, computers are used by a much wider range of people. One cannot imagine a society without computers. The ease of use of computers has improved enormously compared to the early days of computers. Mouse and keyboard are predominantly used for communication between the user and the computer.

However, computer programs and applications become more complex and more difficult to comprehend. The question arises why computers are still not easy to use and instruction for new users is necessary? One way to look at it is that current input devices approximate human abilities for communication. Researchers in the field of Human-Computer Interaction are looking for better ways to communicate with the computer in a more natural manner.

*Multimodal* interfaces can achieve more natural and effective human-computer interaction by integrating a variety of signals, or modalities, by which humans usually convey information. Especially now, when more natural communication technologies are available that are capable of freeing computer users from the keyboard and mouse, multimodal interaction is an interesting field of study. These technologies can be found in the dimensions of sight, sound, and touch.

- Sight
  Much work is done in the field of computer vision. The eyes of a user provide valuable information, such as the direction of the user's gaze. Gazetracking is a process in the user's eye is tracked in order to deduct where the user is looking at. In chapter 4 gazetracking is discussed in more detail.

- Sound
  Speech recognition systems transform spoken language into strings of words. Speech synthesis systems execute the reverse transformation: written language into spoken. Technologies in this area are described in chapter 2.

- Touch
  The hand (together with the fingers, wrist, elbow, and arm) provides a large number of degrees of freedom. Only few of them are used when operating the mouse and keyboard. Research on hand input devices try to take advantage of the hand's qualities of naturalness, dexterity, and adaptability. In chapter 3 techniques for using hand input are described.

Currently, these technologies are not sufficiently advanced to be used individually for robust human-machine communication. But in conjunction with each other promising results are obtained. They can adequately serve simultaneous multisensory information exchange, see [6] and [27].

The challenge is to properly combine these technologies to replicate the natural style of human-human communication by making the combination robust and intelligent. See [7]. This means developing a new methodology of communication and not only imitation of keyboard and mouse interfaces.

### 1.2.1 Related work

In 1980, Richard Bolt published the article "Put-That-There" (see [2]) which describes the first system that uses simultaneously voice and hand gesture inputs at a graphics interface. This system, developed at M.I.T.'s "media room", provides the user the ability to command simple shapes on large-screen graphics display by using speech combined with pointing. This enables the user to use pronouns to refer to an object by pointing at it and saying *this*.

Other researchers combine a speech recognizer and a pen input device in an interface. The pen device can be used to enter handwriting (for almost unpronounceable words), symbols, and editing gestures, such as a cross through an object to command the removal of that object. In [16] and [6] the QuickSet system is described. The application used for Quickset is a military exercise planning system, comparable to this project's application. The pen is used to place military icons on the map or to manipulate the objects. QuickSet consists of a collection of agents that communicate through the Open Agent Architecture (see [11]) developed at SRI (Stanford Research Institute).

This same architecture is used by Cheyer in his multimodal maps application, see [4]. In this application users see the map of a city and they can extract touristic information about it. For example, the user can ask for the distance between two hotels while pointing with the pen on the map.

#### Wizard-of-Oz studies

A lot of work is done in the field of *Wizard-of-Oz* studies. Especially while designing multimodal system, the studies can be useful and can help test and refine the interface. In these studies, a human *wizard*, usually using software tools, simulates the interface. Major usability problems are often uncovered with these type of simulations. (The term Wizard-of-Oz comes from the classic movie in which the wizard controls an impressive display while hidden behind a curtain.)

In [5] a Wizard-of-Oz study is presented in which both the subject and the wizard are monitored. The data collected from each can be used to evaluate and improve the system.

## 1.3 Problem statement

The objective of CAIP's research project is to establish, quantify, and evaluate techniques for designing synergistic combinations of human-machine communication modalities in the dimensions of sight, sound, and touch in collaborative multi-user environments.

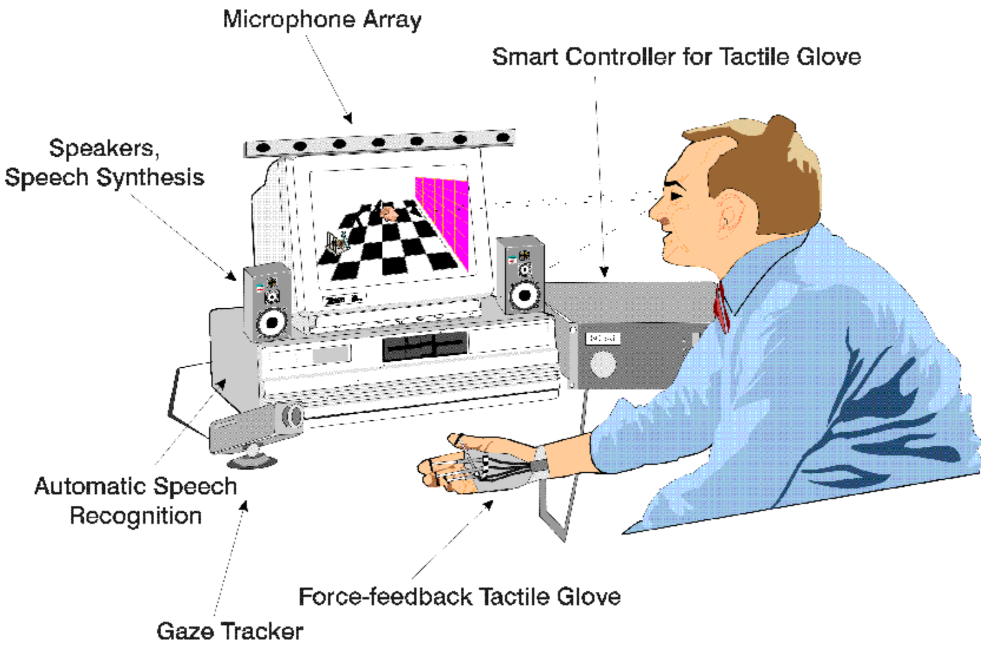


Figure 1.2: Global setup of the prototype.

I concentrate on the realization of a prototype. My goals are to design and implement a specific prototype which integrates the different input modalities. In figure 1.2 the global setup of the desired prototype is shown. Furthermore, I want to test the multimodal system and come up with conclusions and recommendations.

The process of reaching these goals can be divided into the following steps:

- Study the different technologies involved: speech recognition, speech synthesis, gazetracking, and the force-feedback tactile glove.
- Design an architecture for the integration of the modalities with the application. This includes also looking for (or designing) an appropriate multi-user application.
- Implement the application and the interface.
- Test and debug the system.
- Conduct human performance studies.

I have put the emphasis in my work more on the technical part than on the human factors (testing) part.

### 1.3.1 The example application

To fully understand the examples and conclusions of the following chapters, a short introduction on the application used for our prototype is helpful. We developed a multi-user whiteboard application in Java. This whiteboard provides a common visualization space in which simple geometric objects, like circles and rectangles, can be manipulated by multiple users. The original application provides a normal mouse based GUI (Graphical User Interface). Objects can be created, removed, resized, and moved around by clicking on certain icons and moving the mouse.

During the project, the application is extended into a military mission planning system. Objects now include military icons (for example infantry units) and a map is drawn in the background. On this map certain terrain features can be indicated, for example by issuing the command *Indicating hill nine*, and subsequently be referred to by its name, for example *Move tank seven to hill nine*.

## 1.4 Overview

Each of the following three chapters describes one of the used modalities. The order is the same as in which they were integrated into the prototype. First, speech recognition and synthesis (chapter 2), then the force-feedback tactile glove (chapter 3), and finally the gazetracker (chapter 4). In these three chapters, the techniques available are explained and the integration of specific devices into the prototype is described.

In chapter 5, the designed architecture is described. After an introduction on the test application used in this project, a number of theoretical issues such as the AI-techniques used or the importance of feedback are addressed. The technical and programming details will not be looked into until the chapter 6. In this chapter the most important classes of the source code are described. In chapter 7 the human factor studies done so far are discussed, followed by chapter 8 presenting with conclusions, recommendations, and an idea of future directions.

# Chapter 2

# Speech recognition and speech synthesis

In this chapter speech recognition in general is discussed and how it is used in our multimodal system. First, the use of speech as an input modality will be justified by giving several examples in which speech is used. Then, an introduction in the techniques and methods being used in speech recognition is given. Following is an introduction on methods for the synthesis of speech. Then, the speech recognizer and synthesizer used for the prototype is described. Finally, the problems that arose during the integration of the speech recognizer and synthesizer into the system and possible improvements are discussed.

## 2.1 Speech in applications

Speech is the most used means of communication among humans. People find speaking easy and conversation is a skill most master early in life. That makes spoken language attractive to incorporate in user interfaces. Furthermore, it is a very natural, efficient and flexible way of communicating with machines.

Following is a short list of examples in which speech can be advantageous over other means of human-computer interaction.

- In hands or eyes busy situations. While using hands and or eyes, a human can still use his/her mouth and ears for speaking and listening. For example, while carrying groceries inside the house commanding the curtains to be closed and asking the telephone answering machine who called today.
- When the user is disabled or unable to use keyboard and mouse.
- If no keyboard and/or screen is available. For example, on a small cellular telephone, instead of typing in a number, saying the name of the person you want to call.
- In situations in which natural language interaction is preferred. The user wants to use normal linguistic constructs delivered in a conversational manner.

- Telephone applications benefit a lot from speech technology since their sole means of communication with users is through the telephone audio. Currently, telephone information providers (such as travel information) are being automated.
- Discrete dictation systems exist that are capable of processing spoken language in a word processor. The user is allowed to say every word he or she wants, but the user has to leave pauses between the words.
- User interfaces can benefit from speech technology. For example, the access to certain commands can be a lot faster using speech, as opposed to using pull-down menu's.

## 2.2 Speech Recognition

Speech recognition is the process of converting an acoustic signal to text strings. Speech understanding includes one important step more, namely extracting the meaning of the utterance expressed by the speaker. This step involves parsing and analysis algorithms which are described in chapter 5. This paragraph only describes the process of mapping a speech signal to strings of words.

To clarify the theory, a number of general parameters on which speech recognizers can be categorized are described next. Then, a few issues that make the problem difficult are given. Finally, the most used methods in statistically speech recognition are explained.

### 2.2.1 Classification parameters

Several parameters exist by which speech recognition systems can be classified. Below a list of the most common parameters.

- Mode of speaking
  - An **isolated-word** (or discrete) system operates on single words at a time. Therefore it requires the speaker to pause briefly after each word. This is the simplest form of recognition to perform since the endpoints are easier to find. Also, the problem of coarticulation (the influence of one word on the other) almost disappears.
  - A **continuous** speech recognition system does not require pauses between words. Continuous speech is more difficult to handle because of a variety of effects. First, the start and end points of words are difficult to find. Another difficulty is coarticulation. The production of each phoneme (see 2.1) is influenced by its surrounding phonemes. Similarly is the production of words affected by the production of its preceding and following words. The recognition of continuous speech is also affected by the rate of speech. Fast speech tends to be harder to recognize.
- Style of speaking
  This style can range from spontaneous speech to read speech. Spontaneous speech or speech containing disfluencies is much harder to recognize than the utterance of a short command or speech read from a script.

- Speaker dependency
  The speech recognizer can be speaker independent which means that every speaker of that language can be recognized by the system without extra training. Speaker adaptive means that before actually using the system, the user needs to speak some predefined sentences to adapt the system to this speaker. The system is speaker dependent if it is trained for one particular speaker and it needs to be retrained for every other speaker.

- Vocabulary
  The vocabulary can be very small (less than 20 words, e.g. digit recognition) to very large (up to 60.000 words).

- Language Model
  The easiest language model is a finite state grammar in which all permissible words following each word are given explicitly. When approximating natural language a more general language model is specified using context-sensitive grammar.

- Transducer
  The quality of the speech signal is highly dependent on the transducer. This is the device that captures the sound of the speaker. This can vary from a normal microphone, a distant talking microphone, to a telephone.

- Signal to noise ratio (SNR)
  The SNR of the speech signal can be high (30 dB or more) to low (10 dB or less).

Concluding one can say that the most difficult type of speech recognizer is a spontaneous, continuous, large scale vocabulary, speaker-independent, free grammar speech recognizer, while the sound is captured by a telephone in a noisy environment.

### 2.2.2 Variation issues

There are several reasons why the problem of speech recognition is difficult. Following is a list of variabilities:

- Phonetic variability
  Phonemes are the building blocks of spoken language. See table 2.1. The utterance of phonemes is dependent on the context in which they appear. This is called coarticulation. For example, the phoneme [t] in the word *tree* and *three* is pronounced differently. At word boundaries contextual variation arises as well.

- Acoustic variability
  The changes in the environment (e.g. noise of computers in the background) can give acoustic differences. Also the position in the room and characteristics of the microphone give space for variability.

- Within-speaker variability
  When the speaker has a cold or is excited he or she speaks differently. Speaking rate and voice quality both change.

- Across-speaker variability
  The differences in the actual shape and size of vocal tract makes the problem hard. Also, a possibly strong dialect of the speaker can have dramatic influence. [1]

### 2.2.3 Methods and techniques

The capture of the sound signal is the first step in the speech recognition process. The next step is the signal processing phase, in which the most important features are extracted from the speech signal. This step tries to capture enough of the features so that the commonalities that define the sound can be picked out from the broad range of variations mentioned in the last paragraph. This step results in a number of vectors representing samples of the signal.

Before continuing on the next steps, the signal processing will be described.

#### Signal processing

Sound is an analog energy source. When a sound wave strikes a microphone, it is converted to an electrical current. This current is converted into a digital representation resulting in a stream of bits. The number of bits needed is dependent of two factors, the sampling rate and the quantization factor. First, the sampling rate is the number of times per second the signal is represented. For speech, a sampling rate between 8 and 16 KHz (8 to 16,000 times per second) is typical. Second, the quantization factor is a measure of precision to which the energy at each sampling point is recorded. Speech recognizers normally use 8 to 12 bits per sampling point. The storage required for one minute of speech at a sampling rate of 8 KHz with 8-bit quantization is almost half a megabyte. It is obvious that this representation is too big to manipulate.

Grouping samples together into larger blocks is a way of decreasing the size of the representation. These blocks are called frames. The whole frame can be analyzed for the appearance of speech phenomena such as a rise or drop in frequency. Grouping together 80 samples at 8 KHz results in a frame length of 10 msecs. This is a typical frame length in speech recognition systems. It is found that at this length most speech phenomena can be detected and only a few short-duration phenomena are missed.

Within each frame the most important features of the group of samples are stored. These features can be the amount of energy at each of several frequency ranges, the overall energy in a frame, and the difference from the previous frame. The features describe the pitch, volume, color, etc. of the signal.

Most speech signal processing systems include the final step of vector quantization. If a frame contains $n$ features, one can think of an $n$-dimensional space containing many points. Vector quantization divides this space into $m$ regions, labeled C1 through C$m$. Each frame can then be represented using only a single label. In case of $m$=*256*, one byte is sufficient to represent a frame. Obviously, going from a feature vector to a label causes some loss of information but applying an optimal quantization still produces an accurate result.

[1] *Speaker identification* is the process of determining which person is speaking. Across-speaker variability is essential for this problem.

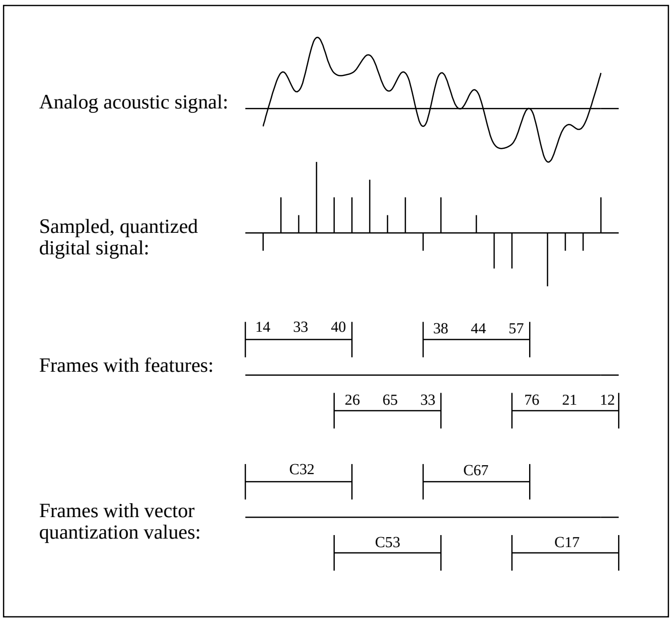


Figure 2.1: The translation of the acoustic signal into a sequence of vector quantization values.

In figure 2.1 the signal processing phases are summarized. The two lower graphs show that the frames are overlapping. This is to prevent the loss of information at frame boundaries.

Concluding one can say that signal processing wants to satisfy two goals. First, reducing the size of the representation of the speech signal. Second, come up with a representation that is likely to encode features of the signal that will be useful for word recognition.

In [22] more can be found on the theory of signal processing.

### The speech recognition model

As stated before, speech recognition is the problem of mapping a sound signal onto strings of words. A common approach is to use statistical models to solve this problem. The chance that the signal is caused by word $w$ should be maximized. Using Bayes' rule, this results in:

$$P(words|signal) = \frac{P(words)P(signal|words)}{P(signal)}$$

Given a signal, the task is to find the sequence of words that maximizes $P(words|signal)$. $P(signal)$ on the the right-hand side of the equation is a normalizing constant which can be ignored. $P(words)$ is called the language model while $P(signal|words)$ is known as the acoustic model.

### The language model

The language model gives probabilities to the occurrence of a specific sequence of words in a language. One approach to capture this is to form a grammar of the language. When the speech recognition is used as a command language or in case of a small domain specific vocabulary, a finite state or a context free grammar can be sufficient.

In case of larger vocabularies or when trying to capture natural language a statistical approach is better. In this approach, grammar restrictions are not explicitly represented by rules but implicitly by probabilities. For example, the sentence *I read a book.* has a higher probability than *Book a read I.* which has zero probability. Using the notation $w_1 \cdots w_n$ to represent a string of $n$ words, an expression for the probability of a string looks like:

$$\begin{aligned} P(w_1 \cdots w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\cdots P(w_n|w_1\cdots w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_1\cdots w_{i-1}) \end{aligned}$$

Most of the factors in this equation are complex and difficult to compute or estimate. For that reason, this formula is approximated with something simpler. Instead of looking at all preceding words, only one preceding word is taken into account. This is called a bigram model. The model is defined by the following formula:

$$\begin{aligned} P(w_1 \cdots w_n) &= P(w_1)P(w_2|w_1)P(w_3|w_2)\cdots P(w_n|w_{n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_{i-1}) \end{aligned}$$

By counting the number of times each word pair occurs in a representative corpus of strings, the probabilities can be estimated. For example, if "a" appears 10,000 times in the training corpus and it is followed by "book" a 100 times, then $P(book_i|a_{i-1}) = 0.01$.

A problem is that the training corpus does not contain all valid words of the language, so these words get an estimated probability of zero. If the speech recognizer is used to recognize natural language, this problem becomes important. Normally, a small portion of the probability distribution is set aside for these words.

One can also choose for a trigram model, in which two preceding words are taken into account. This provides values for $P(w_i|w_{i-1}w_{i-2})$. This is a more powerful language model. However, the problem is that the number of parameters increases compared to bigram models, which makes it hard to get enough training data to come up with accurate probability estimates. A compromise can be to use a model that consist of a weighted sum of the trigram, bigram, and unigram (or word frequency) models. The probability of a string is now given by:

$$P(w_1 \cdots w_n) = \prod_{i=1}^{n} (c_1 P(w_i) + c_2 P(w_i|w_{i-1}) + c_3 P(w_i|w_{i-1}w_{i-2}))$$

$$(c_1 + c_2 + c_3 = 1)$$

### The acoustic model

In written language the building blocks constructing words are called characters. In spoken language phonemes are considered the smallest atomic units to represent speech. Spoken English contains about 40 to 50 phonemes.

Table 2.1: The DARPA phonetic alphabet, listing all the phonemes used in English.

| Vowels | | Consonants B-N | | Consonants P-Z | |
|---|---|---|---|---|---|
| Phoneme | Example | Phoneme | Example | Phoneme | Example |
| [iy] | b**ea**t | [b] | **b**et | [p] | **p**et |
| [ih] | b**i**t | [ch] | **Ch**et | [r] | **r**at |
| [ey] | b**e**t | [d] | **d**ebt | [s] | **s**et |
| [ae] | b**a**t | [f] | **f**at | [sh] | **sh**oe |
| [ah] | b**u**t | [g] | **g**et | [t] | **t**en |
| [ao] | b**ough**t | [hh] | **h**at | [th] | **th**ick |
| [ow] | b**oa**t | [hv] | **h**igh | [dh] | **th**at |
| [uh] | b**oo**k | [jh] | **j**et | [dx] | bu**tt**er |
| [ux] | b**eau**ty | [k] | **k**ick | [v] | **v**et |
| [er] | B**er**t | [l] | **l**et | [w] | **w**et |
| [ay] | b**uy** | [el] | bott**le** | [wh] | **wh**ich |
| [oy] | b**oy** | [m] | **m**et | [y] | **y**et |
| [axr] | din**er** | [em] | bott**om** | [z] | **z**oo |
| [aw] | d**ow**n | [n] | **n**et | | |
| [ax] | a**bout** | [en] | butt**on** | | |
| [ix] | ros**e**s | [ng] | s**ing** | | |
| [aa] | c**o**t | [eng] | Wash**ing**ton | [-] | (*silence*) |

In table 2.1 the DARPA phonetic alphabet is shown. The pronunciation of each phoneme is dependent of the phoneme preceding and following it. That is why often not single phonemes are stored but combinations of phonemes, diphones or triphones.

In the acoustic model each word is described as a sequence of phonemes. For example, the word *book* is always pronounced with the three phonemes [b uh k]. The pronunciation of this word is specified as a Markov model. In general, a Markov model is a way of describing a process that goes through a series of states. The model describes all the possible paths and assigns a probability to each one. An important property of this Markov model is that the probability of transition from the current state to another one is only dependent of the current state.

In figure 2.2 two examples of word models are shown. The first one describes the simple word *book*. In this model only one path is possible, so all transition have probability 1.

However, when looking at complexer words, sources of phonetic variation apply. Dialect can produce variation in the utterance of the same word. For example, the verb *potato* is pronounced differently in American English as opposed to British English. Also, the problem of coarticulation arises here. Each phoneme is influenced by its preceding and following phoneme. When the speaker talks quickly this influence increases, whereas the speaker talks slowly the different phonemes can be distinguished more easily. The first *o* in *potato* can sound more like an [ah] (as in *but*) when spoken quickly.

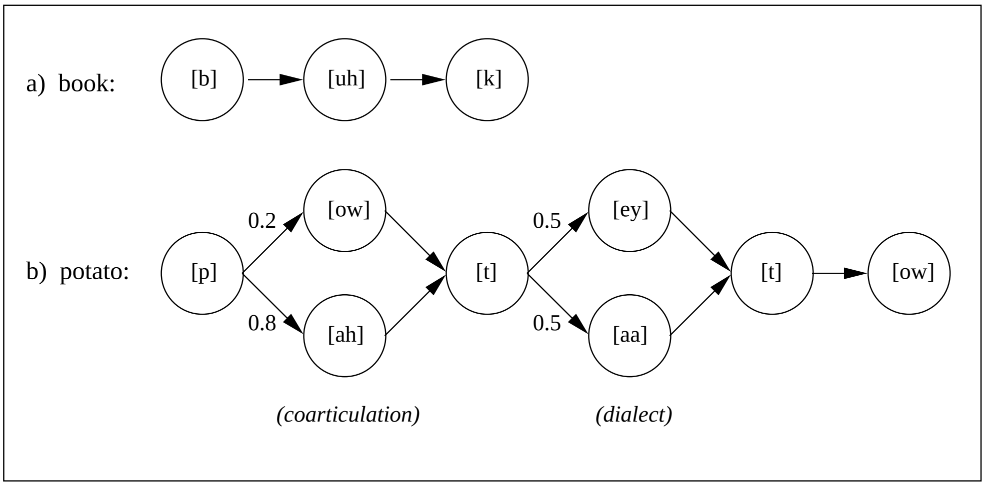


Figure 2.2: Example of a word model.

In figure 2.2 b) the word model of *potato* is shown and one can see the probabilities assigned to each transition. In the example, the speaker can speak American or British English with same probability. The chance of him or her pronouncing the first "o" clearly is set to 0.2.

The acoustic model described so far can be seen as a dictionary of word models. If a string of phonemes is given, the most probable word model can be found in the dictionary with its corresponding word. This can be seen as maximizing $P(words|phonemes)$. However, the signal processing phase does not provide a string of phonemes but a vector quantization. So, the next problem is how to maximize $P(signal|phoneme)$. A common solution is found in using hidden Markov models, or HMM's.

### Hidden Markov model

A hidden Markov model describes, just as a regular Markov model, a process that goes through a sequence of states. In a normal Markov model, the output is a sequence of state names which uniquely determines the path through the model. In a hidden Markov model, each state has a probability distribution of possible outputs, and the same output can appear in different states. The actual state of the model is hidden from the user.

Each phoneme is represented by an HMM which determines the correspondence between the phoneme and the signals. In figure 2.3 an example of an HMM for the phoneme [n] is shown. The probabilities are acquired after training which is described later. After the signal processing phase, a string of quantization values is produces. For example, [C1,C3,C5,C5,C6,C7]. Since C5 is found in the Mid as well as the End state, two paths are possible for this sequence of signals. The probability that this sequence came from the phoneme

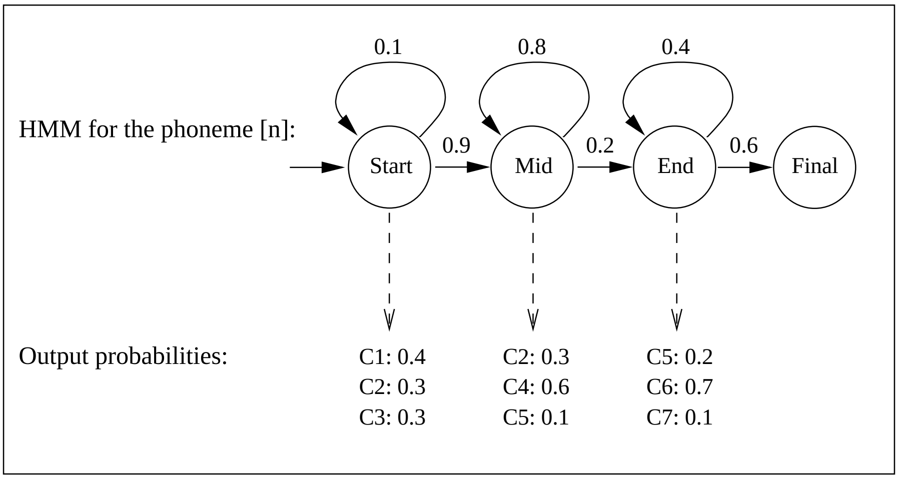


Figure 2.3: Example of an HMM for the phoneme [n]

[n] is calculated as the sum of these two paths. The probabilities of the paths are multiplied by the probability that this path generates the sequences.

$$
\begin{aligned}
&P([C1, C3, C5, C5, C6, C7] | [n]) = \\
&\quad (0.1 \times 0.9 \times 0.8 \times 0.2 \times 0.4 \times 0.6) \times (0.4 \times 0.3 \times 0.1 \times 0.1 \times 0.7 \times 0.1) + \\
&\quad (0.1 \times 0.9 \times 0.2 \times 0.4 \times 0.4 \times 0.6) \times (0.4 \times 0.3 \times 0.1 \times 0.2 \times 0.7 \times 0.1) \\
&\quad = 0.000000581
\end{aligned}
$$

Most phonemes have a duration of 50 to 100 milliseconds, which is 5 to 10 frames at 10 msec per frame. In the example, the phoneme is spoken quickly. The loops in the model allow it to represent both fast and slow speech. The multiple vector quantization values on each states represent other sources of variation. This makes the HMM a powerful model, and that is why it is used in most speech recognizers.

The three models described so far need to be combined to solve the mapping problem. The language bigram model, the word pronunciation dictionary, and the phoneme HMM can all be seen as hidden Markov models. A possible combination is to put them all into one big HMM. This is achieved by first replacing all words in the bigram model by their corresponding word pronunciation model. This results in an HMM in which each state is a phoneme. Now, replacing each phoneme state with the corresponding phoneme HMM, produces a big HMM in which each state corresponds to a distribution of vector quantization values.

### The Viterbi algorithm

The HMM described above can be used to compute $P(words|signal)$. Unfortunately, computing these probabilities is an infeasible task, since there are too many candidate words. The straightforward algorithm takes an HMM model and an output sequence and returns the most probable path through the HMM that outputs the sequence. This algorithm would be $O(M^n)$, where $M$ is the number states and $n$ is the length of the sequence.

The Viterbi algorithm offers a better solution to the problem. This algorithm makes use of the Markov property saying that the most probable path for the rest of any sequence can depend only on the state in which it starts. That means that for each state, only the most probable path that ends there needs to be tracked. The other paths can be ignored.

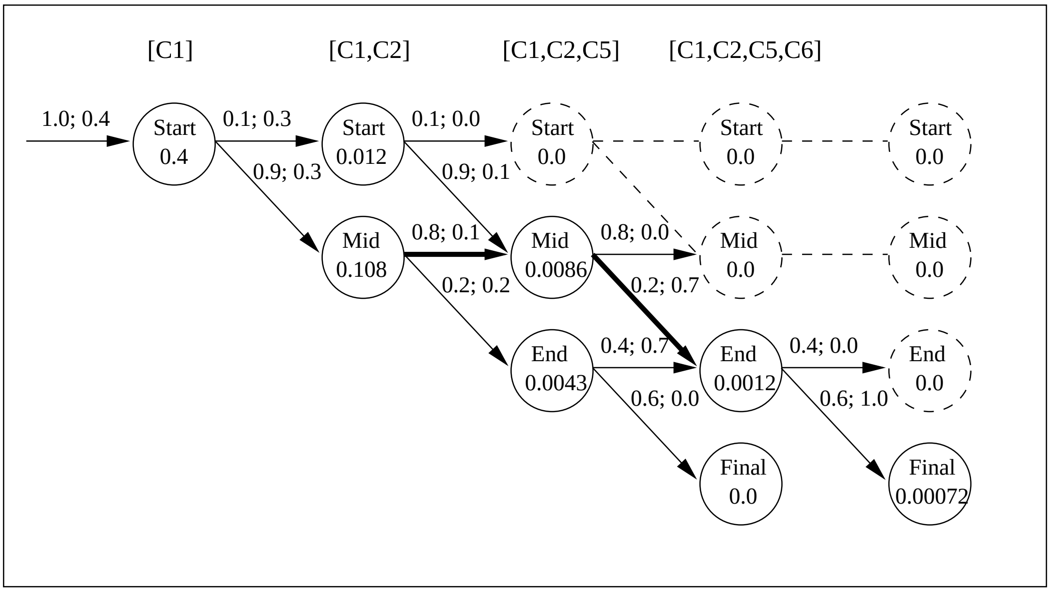


Figure 2.4: A diagram of the Viterbi algorithm computing the most probable path.

In figure 2.4 a schematic presentation of the algorithm is shown. Each column represents an iteration of the algorithm. In each state the continuations that lead to a state with an appropriate quantization value are considered. The one with the highest probability is chosen, the others need not to be remembered. In the figure this path is shown by the bold arrows.

The Viterbi algorithm is *O(bMn)*, where $b$ is the branching factor (the number of arcs out of any state). If the model is fully interconnected, then $b = M$, and the algorithm is $O(M^2n)$.

**Training the model**

Now, the only question not answered is how to train the model. The model can be trained directly from a training set of signal-words pairs. The standard algorithm for training an HMM is called Baum-Welch. This algorithm as well as other HMM algorithms are described by Rabiner in [21]. In [23] and [22] more can be found on the fundamentals of speech recognition.

## 2.2.4 Speech Recognition systems

Currently, the best speech recognizers have an accuracy ranging from about 80% to 98% depending on the situation. The exact percentage depends on the issues mentioned in paragraph 2.2.1. The best results are achieved when the microphone is of high quality, the vocabulary is small, the recognizer is speaker dependent, and the words are pronounced one by one.

## 2.3 Speech synthesis

Speech synthesis is the problem of converting a text string into spoken language. The problem is also referred to as text-to-speech (TTS) conversion. The major steps in producing speech from text are as follows:

- *Structure analysis*
  Process the input text to determine where paragraphs, sentences and other structures start and end. For most languages, punctuation and formatting data are used in this stage.
- *Text pre-processing*
  Analyze the input text for special constructs of the language, such as abbreviations, acronyms, dates, times, numbers, etc. This step translates for example, *Come at 5.30 to St. Elsewhere hospital* into *Come at five thirty to Saint Elsewhere hospital*.
- *Text-to-phoneme conversion*
  Convert each word to phonemes. See table 2.1 for the phonemes used in English.
- *Prosody analysis*
  Process the sentence structure, words and phonemes to determine appropriate prosody for the sentence. Prosody includes many feature of speech besides the sound being spoken. For example, the pitch, the timing, the pausing, the speaking rate, the emphasis on words, etc. Correct prosody is important to make the speech sound natural and for correctly conveying the meaning of a sentence.
- *Waveform production*
  Produce the actual audio waveform for each sentence. Two ways of producing this waveform using phoneme and prosody information are:
  - **Concatenation**: putting chunks of recorded human speech together. The size of the portion of speech can vary between phonemes, diphones, syllables, words, and (parts of) sentences. The longer the units of speech become, the more are needed, but the quality increases too. The choice what size to use depends on the application. For example, for an automated messaging system at a train station whole sentences can be recorded since the number of sentences is limited.
  - **Formant synthesis**: using signal processing techniques together with knowledge of how phonemes sound and how prosody affects them, to produce the waveform.

## 2.4 The Microsoft Speech recognizer and synthesizer

The current system uses a Microsoft Whisper (see [10]) speech recognizer and synthesizer engine. It is written in C++ and provides the programmer with a number of speech libraries. The programmer can relatively easy program his or her own speech application.

The language model is defined by a finite state grammar. The restricted task-specific vocabulary contains about 150 words. A part of the grammar is shown in section 5.2.2. The recognition is speaker independent, so different speakers can use the system without the need for training of the speech recognizer.

Both the speech recognizer and the speech synthesizer are connected to the multimodal system by sockets. If the recognizer recognizes a phrase, the string is sent through the socket to the system. If the phrase is not recognized, nothing is sent to the system.

The speech synthesizer is waiting for input by the system. If the system needs to give feedback, it sends an appropriate string to the speech synthesizer.

The integration of speech into the multimodal system is described in more detail in chapter 5.

### 2.4.1 Sound capture: microphone array

One group of the Multimedia Information Systems Lab at CAIP conducts research in the field of sound capture by microphone arrays. A microphone array liberates the user from body-worn or hand-held microphone equipment. This allows freedom of movement in the workplace.

The microphone used in this multimodal project is a fixed focus line microphone array that focuses on the speaker's head sitting approximately one meter from the monitor.

A CAIP-developed microphone array system is applied as a robust front-end for the speech recognizer to allow distant talking, see [17].

## 2.5 Conclusions and recommendations

The implementation of the Microsoft speech recognizer and synthesizer is reasonably easy. Sockets are used to connect both the speech recognizer and synthesizer to the system. The speech recognizer recognizes real-time and the sockets are fast enough to achieve a good response time; no delay is noticed.

The accuracy found during testing of the system is good. Most of the times the fixed focus line microphone array successfully attenuates the other sound sources and the phrase is recognized. Although an undo command is implemented it is not needed often.

If a spoken phrase is not recognized by the recognizer no string is sent to the application, although the sound of the utterance is noticed. The recognizer could send a "Not recognized" string to the system, where appropriate feedback can be issued, such as "Please repeat the last command". Currently, this is not implemented since other sound sources can invoke the system's feedback, too. For example, if a person enters the laboratory this is sometimes noticed by the speech recognizer. This would then result in a not appropriate reaction of the system. Moreover, each recognized phrase is prompted on the statusbar of the application. So, the user has visual feedback if the recognizer has not recognized the utterance.

A timestamp is a time indication for each word about when it was said. Timestamps are useful when synchronizing the different inputs. In chapter 5 the issue of timestamping is addressed in more detail. Unfortunately, the Microsoft speech recognizer does not provide the programmer timestamps. It only gives the start and end time of the recognized utterance. Since this is an important lack in the capabilities of the speech recognizer, one might consider using another one. Also, one of the design principles of the prototype is to keep it platform-independent. The Microsoft Whisper system runs exclusively under Microsoft Windows which is another reason to consider changing the speech engine.

Currently, the text-to-speech synthesizer and speech recognizer are running on two different machines. The integration of these two applications in one application seems useful. Especially, since what is said by the speech synthesizer should not be recognized by the speech recognizer.

# Chapter 3

# Hand gesture input

In this chapter modalities in the dimension of “touch” are described, in particular the force-feedback tactile glove used for the hand gesture input of the multimodal system. After an introduction, techniques for hand gesture input are addressed, followed by a number of example applications. Then the Rutgers-Master II force-feedback tactile glove is described. This chapter concludes with a conclusions and recommendations.

## 3.1 Introduction

Hands are a person’s primary physical connection to the world. Most everyday tasks are performed with our hands. However, when working with computers hands are used to type on a keyboard and to move the mouse. These movements can hardly be called natural for the hands. Characteristic features of the use of hands, such as dexterity and adaptability are not expressed in computer tasks.

The use of hand gesture input has a number of advantages:

- Natural
  Gesture input is a natural form of communication. If the number of gestures is limited, it is an easy-to-learn method of interacting with the computer.
- Powerful
  Hand input provides multiple degrees of freedom for each finger and for the hand. This can be used to obtain a powerful way of interacting with the system. One gesture can be used to define both a command to be executed and its parameters. For example, the user can give the command *resize* and provide the size by moving his or her hand.
- Direct
  By using gesture input, the user’s hand becomes the input device, providing direct manipulation. The user can interact with the computer by simple designation and adequate gestures.

Of course, there are also a number of disadvantages related to the use of gesture input.

- Fatigue
  The arm, wrist, hand, and fingers are all involved in making a gesture. More muscles are used as opposed to keyboard and mouse input. This means the user fatigues faster, especially if the gestural commands require a high precision over a longer period. Effects of the RSI (Repetitive Strain Injury) syndrome can occur.
- Learning
  The user needs to learn the gestures that the system recognizes. These gestures should be natural, simple, and consistent to reduce the required learning time.
- Non-intentional gestures
  A system that captures every motion of the user's hand, also captures non-intentional gestures. This means the user cannot interact gesturally with other persons or devices.
- Lack of comfort
  Currently, most hand gesture input devices consist of a glove with sensors connected through cables to a controller. The user wears this glove, being constrained in his movements by the device.

## 3.2 Techniques

In this section, techniques for gesture input are described. The "reading" of a person's hand involves tracking the position of the hand and investigating the gesture made by the hand. A classification can be made between camera based and glove based methods.

### 3.2.1 Camera based

Optical tracking of the hand uses one or more cameras to follow the user and monitor the movements he or she makes. Two methods are described next. (The first method often involves a glove, however it is classified as camera based since camera's are used.)

- Using markers
  Markers, such as infrared LED's or reflective spots, placed on the hand can be used to analyze the motion of the hand. A problem is that this methods requires a lot of processing time to analyze the different camera images and to determine the marker's position in 3D. This limits the real-time capabilities of these systems.
- Using silhouette images
  The silhouette images of the user can be used to analyze his or her movements. By using complex hardware systems using this method are able to discriminate parts of the user's body, such as hands, legs, arms, and fingers, in real-time. This way certain gestures can be deduced.

The main advantage is that these camera based systems allow the users to interact with the computer free of encumbering equipment. However, an improvement in the performance of these systems is necessary to compete with glove based systems.

### 3.2.2 Glove based

Glove based techniques use a glove mounted with a number of sensors. Two types of tracking exist:

- Magnetic tracking
  This technique uses a source element radiating a magnetic field and a small sensor that reports its position and orientation with respect to the source. Currently, several multi-source and multi-sensor systems are available that provide accurate and robust tracking of a number of points at up to 100 Hz.

- Acoustic tracking
  This techniques uses high-frequency sounds to triangulate a source within a room. Many systems send out little sounds from the source, mounted on the glove, that are received by microphones in the room.

A disadvantage of both techniques is that the glove is attached through a cable to a central device. This limits the free movements of the user.

## 3.3 Hand input in applications

Since hand input devices became commercially available, the number of applications using hand input has increased and so did the research in this area. In the next section examples of applications and research projects involved with hand input are described. Then, a number of natural gestures is listed together with their meanings.

### 3.3.1 Examples

Following is a number of examples. For more information see [26].

- Virtual reality
  The main research concerning tactile gloves can be found in the virtual reality (VR) projects. In a VR situation, a user experiences a simulated world in which he can manipulate (grab, rotate, etc.) objects using a glove. The user can move through the virtual space by pointing in the desired direction.

- Interpreting sign language
  A lot of research is conducted on the interpretation of sign language for interaction with computers and humans. These projects vary in recognizing hand signs from simple finger spelling to analysis of American Sign Language (ASL). The recognition of hand signs that involve motion is found more difficult since pattern analysis must be performed on the moving hand.

  Besides computer input and control, applications can be found in the communication of hearing impaired people with hearing people. Through the use of a speech synthesizer, sign language is transformed into spoken language, enabling a deaf person to communicate to a hearing person not knowing the sign language.

- Teleoperation
  Glove interfaces can be used to operate a device on a distance. The gestures and movements made by the user can be reproduced at the remote end.

- Animation
  Body movements of humans can be tracked for computer animation of characters. The nuances of natural motion are translated, making the animated characters seem very much alive.

### 3.3.2 Hand input language

A hand gesture can be seen as a specific formation of the fingers possibly combined with a movement or pattern of the whole hand. In table 3.1 a number of hand gestures are enumerated together with their more or less natural meaning in a virtual workspace.

Table 3.1: Table of gestures and possible meanings.

| **Gesture** | | **Meaning** |
|---|---|---|
| 1. | grab | Manipulating (translating, rotating) objects in the virtual space. |
| 2. | wave hand, after grabbing an object | Removing the select object |
| 3. | open hand | Releasing the selected object. |
| 4. | point with index finger at an object | Asking for more information on that object. |
| 5. | make a circle with thumb and index finger | O.K. |
| 6. | thumb up, closed hand | Answering a question with yes |
| 7. | thumb down, closed hand | Answering a question with no |
| 8. | curl all fingers but the thumb towards the palm of the hand | Closer |
| 9. | curl all fingers but the thumb away from the palm of the hand | Further away |
| 10. | point left with the thumb | More to the left |
| 11. | point right with the thumb | More to the right |
| 12. | wave goodbye with hand | Exit the system |

Note that some gestures are actually the same, but only the pattern made with the hand differs. For example, the gestures 6 and 7, “thumb up” and “thumb down”, only differ in orientation of the hand.

As described in the next section, both the gesture and the orientation are provided by the glove used for the prototype. However, the orientation of the wrist is only used to determine the position on the screen, it is not involved in the gesture recognition.

## 3.4 The Rutgers-Master II Force-Feedback Tactile Glove

The Rutgers Master II (RM–II) system is a portable haptic interface designed for interaction with virtual environments [3]. Its two main subsystems, shown in figure 3.1, are the *hand–master* and the *Smart Controller Interface* (SCI). The RM–II hand master can read hand gestures (fingertip positions relative to the palm) and apply forces on the fingertips corresponding to the interaction.

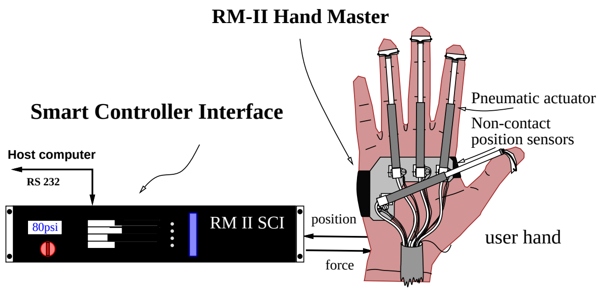


Figure 3.1: The Rutgers Master-II force-feedback tactile glove.

The hand–master unifies hand motion sensing and force display in a small, compact structure weighing 80 grams. The main structure consists of a small "L" shaped palm base on which four custom-designed linear-displacement pneumatic actuators and 8 Hall-effect sensors are mounted. The actuators are equipped with a special sensor to measure their linear displacement. A Polhemus tracker [20] mounted on the back of the hand provides wrist position/orientation.

### 3.4.1 Hand gesture module

The hand gesture module is implemented using the RM–II system described above. Hand gesture input is particularly efficient for 3D environment surfing (changing the viewpoint of virtual environment) and for virtual objects manipulation. Since the glove was used in this application in a 2D graphic environment, only gestures for object manipulation are implemented. A special gesture was defined to emulate mouse clicks. The design of the hand gesture module features natural hand gestures in an effort to create an easy-to-use interface [19]. A glove cursor is shown on the screen, see figure 3.2, to provide visual feedback to the user.

Each implemented gesture is described by specific value ranges of the linear displacement sensors. The output values of the sensors are checked against these value ranges to determine the gesture made.

The first step in designing the hand gesture module involves implementing an object selection function. In a 2D environment, the coordinates are calculated where a virtual ray along the user's index finger intersects the screen plane. The object is selected when the corresponding gesture ("pointing" the index finger) is executed. In the military mission planning system (described in chapter 5) pointing at a certain object yields information about this object which is spoken to the user by the system.

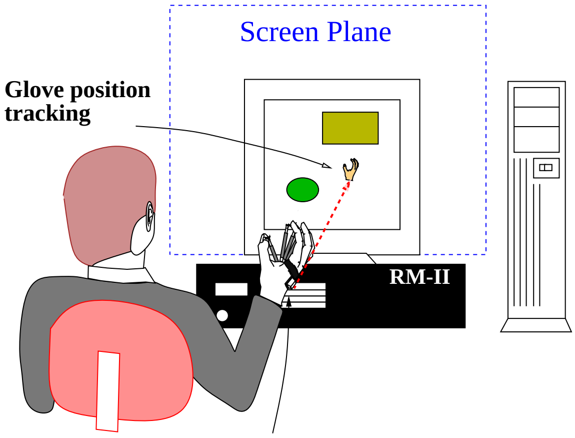


Figure 3.2: Hand gesture interaction.

Other gestures designed for object manipulation and implemented in the prototype are:

**grab** used for grabbing and moving (dragging) the selected object.

**thumb up** associated with resizing the selected object.

**open hand** corresponds to the *unselect* or *put down* command. It is also used as a reset position for hand gestures and both previous gesture commands are terminated by this gesture.

**curling the thumb** corresponds to a mouse click.

Sockets are used to connect the RM–II Smart Controller Interface to the main application. The coordinates are sent to the system that puts the glove cursor on the screen.

Table 3.2: Table of implemented gestures and their textual representations.

| **Gesture** | **Text string** |
|---|---|
| grab | `''move this with glove''` |
| open hand | `''put down''` |
| point at an object (*in drawing program*) | `''select''` |
| point at an object (*in planning system*) | `''what is this''` |
| thumb up (*only in planning system*) | `''resize this with glove''` |

The moment the user makes a gesture, this gesture is translated into a text string corresponding to the issued command. This process is similar to the translation of sign-language gestures into their spoken language representations. In table 3.2 the translation of all gestures into appropriate text strings are summarized. The string is sent to the application for further processing, resulting in the execution of the command, see chapter 5.

## 3.5 Conclusions and recommendations

The performance of the tactile glove in the prototype is real-time, no delay is noticed. The five implemented gestures are natural and easy to learn. Not many gestures should be added to keep the learning time short.

However, a number of problems are encountered:

- The problem of fatigue and lack of comfort plays a role. The user is connected to a device by cables. This contradicts the goal of using not encumbering equipment expressed by the use of a microphone array for the sound capture as described in chapter 2.

  Also, the user is limited in his use of hand gestures to communicate with humans or other devices.

- The precision required for some tasks of the military mission planning is reasonably high. This precision is difficult to achieve with the glove.

- Non-intentional gestures are interpreted as commands which can result in not desired executions of commands. An undo-command is implemented for this reason (among other reasons).

The application is in a 2D environment which makes the use of a glove less obvious than in a 3D environment. Changing the 2D into a 3D environment which is one of the objectives of the project, justifies the use of the glove.

A true multimodal system not only provides multimodal input but also multimodal output, for example, spoken output by a speech synthesizer, as described in chapter 3. Although not used in this project, the RM–II tactile glove also provides a means of output, namely haptic feedback. This feature can be used in further versions of the systems. For example in the mission planning scenario, force-feedback on the hand can be used to let the user know when he or she is moving around in a danger area.

# Chapter 4

# Gazetracking

Gazetracking, the process of following where a person is looking at, is described in this chapter. After a short introduction, an overview of the current techniques available for gazetracking is given. Then, the use of gazetrackers in applications is described, trying to give a justification for the use of the eyes in the user interface. The gazetracker used for the multimodal system is described next. A discussion presenting conclusions and recommendations ends this chapter.

## 4.1 Introduction

People use their eyes intensively every day of their lifes for a large variety of purposes, such as reading, watching television, guiding motions, gathering information, for means of non-verbal communication, etc... The actual perception process is a difficult task, consisting of sampling light structures from the ambient optic array, combining them, and integrating them with stored knowledge. In this respect, eyes are considered as *input*-organs and this is how people mainly regard them. Indeed, the eyes observe and this is their most important role. However, the eyes also provide information and can be considered as *output*-organs. The most important output is the direction the eyes are focused upon.

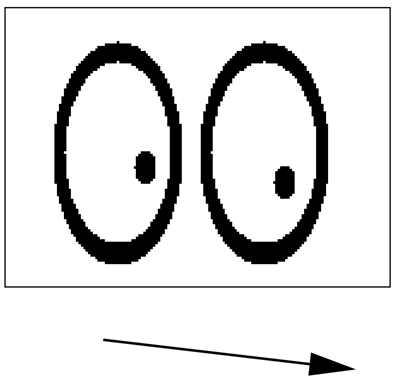

Figure 4.1: Snapshot of the Unix application *xeyes*.

In figure 4.1 a snapshot is shown of the application *xeyes*. This little “toy” application simply follows the mouse pointer by moving the pupils as though the eyes are actually looking at the pointer. One immediately sees the direction, in the figure indicated by an arrow.

Besides the direction of the gaze, other outputs are given by the eyes but they are less obvious. For example, the frequency of blinking of the eyes, and the diameter of the pupil can tell something about the person’s mental state. For example, if a person is nervous or anxious he or she tends to blink more often. Most of the times people are unaware of these eye outputs but they are part of the non-verbal communication between humans.

## 4.2 Techniques

Before looking at techniques for gazetracking, a classification of gazetrackers is given. An important distinction is made between *intrusive* and *non-intrusive* gazetrackers.

An intrusive gazetracker interferes in the normal movements of the user. It can consist of head mounted equipment, for example a special helmet or goggles. One advantage of this type of gazetracker is that it can cover an entire room since the user is wearing it. Another advantage is that it exactly tracks the light rays that actually enter the eye. A gazetracker can also be intrusive in the sense that it requires a stable head position. Obviously, the drawbacks of intrusive gazetrackers are that the user needs to wear special equipment or is not allowed to move his or her head.

A non-intrusive gazetracker tracks at a distance and allows (limited) movements of the head. The big advantage is that this type does not hamper the user much, since no equipment needs to be worn. A disadvantage is that it is hard to cover a large area with the gazetracker. The user is tracked at his or her workplace however, and the user often sits in a chair using a limited space.

Several techniques of tracking the gaze of a person are available. The most common techniques use a camera and possibly some sort of light that is shone into the eye. If light is used, it is most often infrared light since infrared light distracts the user as little as possible. Also, by using infrared light, the interference from other light sources is avoided. The camera is infrared sensitive and records the eye as well as the reflection of the light beam.

In figure 4.2 a general set-up of a desk mounted gazetracker is shown. (The gazetracker used for the multimodal prototype is of this type.)

The first difficulty is finding the eye(s). Some techniques track the face and use the reflection of a stationary light to find the eye. Other techniques require the camera to be zoomed in on the eye during the calibration phase. Then, the eye is tracked automatically.

Once an image of the eye is captured, several features can be examined in order to compute the direction of the gaze.

### 4.2.1 Image analysis

Blinking of the eyes and the diameter of the pupil are features that can be extracted reasonably easy directly from the eye-image. For example, if no light

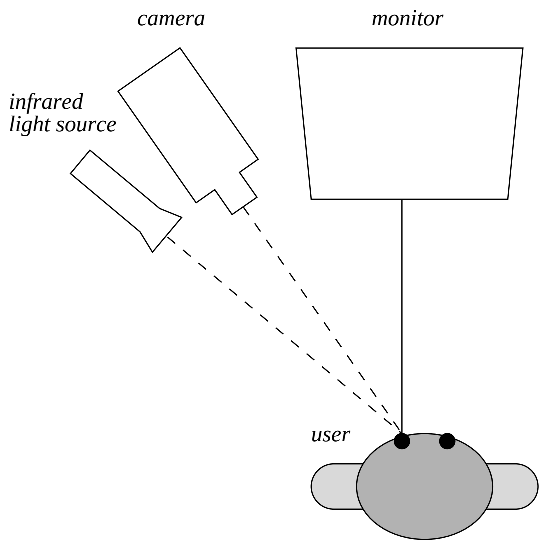


Figure 4.2: Diagram of the set-up of a camera based gazetracker.

reflection is detected by the camera for a short period, the person's eye is probably closed, indicating a blink.

A straightforward method to compute the direction of the gaze is to look at the position of the iris and pupil in the eye. This gives a good indication of the direction of the gaze. Other techniques are described next.

### Limbus tracking

The limbus is the boundary between the white area of the eye and the eye. This boundary can easily be optically detected since the iris is usually darker. The position and shape of the limbus relative to the head is used to deduct the gaze. This requires the head to be held still or the apparatus to be fixed to the user's head, so limbus tracking is intrusive. Since the eyelids cover the top and bottom of the limbus occasionally, this technique is more suitable for precise horizontal tracking than for both horizontal and vertical tracking.

### Pupil tracking

Instead of tracking the limbus, the pupil can be tracked, too. The advantage is that the pupil's border is often sharper than the border of the limbus and the pupil is less often covered by the eyelids. Thus, both horizontal and vertical tracking is possible.

### Corneal reflection and pupil tracking

The corneal reflection relative to the center of the pupil can be used to compute the direction of gaze. This method is not-intrusive, since it does not use a position relative to the position of the head. In figure 4.3 a close-up of an eye is shown. The corneal reflection and the center of the pupil are marked by little crosses.

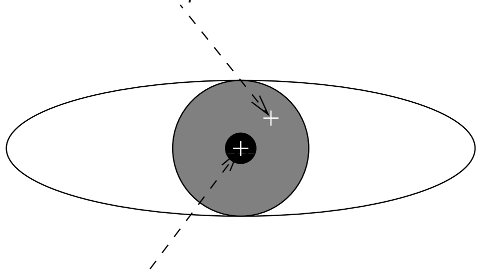


Figure 4.3: Diagram of an eye showing the two recorded points.

**Tracking using an Artificial Neural Network**

Artificial neural networks (ANN) can also be used for the computations in tracking. A digitized video image of the user's head is used and the reflection of a stationary light source can be used to find the user's eye. A smaller rectangular part of the video image is extracted and fed into an ANN. The output of the ANN is a set of display coordinates.

### 4.2.2 Other techniques

Two non-camera based techniques are still other under research in the laboratories.

1. Measuring the electric potential of the skin around the eyes.
   This technique is based on the existence of an electrostatic field that rotates along with the eye. By recording the small difference in the skin potential, the direction of the gaze can be calculated. The advantage of this technique is that no clear view of the eye is required, since the electrodes are placed on the skin. These same electrodes make this technique rather troublesome and not ideal for every day use.

2. Using a special sort of contact lens that must be worn by the user.
   In this technique, a special contact lens is used which makes it possible to measure the direction of the gaze quite accurately. The same disadvantage as for the previous technique holds however, which is why this type of gazetracking has not left the laboratories.

### 4.2.3 Calibration

All described techniques require some sort of calibration and often during the session one or more re-calibrations. During the calibration phase, the user is asked to look at a number of predefined points. These points often include the center and the corners of the screen. After calibration the gazetracker can interpolate to find the direction of the gaze.

The ANN gazetracker requires a longer calibration. The neural network needs to be trained by gathering images of the user's eye and head while the user is following a moving cursor on the screen. This needs to be done once for each different user.

## 4.3 Gazetracking in applications

In this paragraph the question "Why is gazetracking useful?" will be answered. First, a few examples are described in which gazetracking is used. Then, the issue of using gaze in a user interface is addressed.

- Eye movements are subject of research in many fields. Following the eyes of an observer while he or she is looking at a painting can be very interesting and tell a lot about the observer (and about the painting). For example, if the observer is autistic other patterns of eye movements are found.

- In Apache helicopters the technique of gazetracking is used to provide the pilots with an easy firing system. The pilot can guide the helicopter's missiles by moving his head and eyes. This way the pilot's hands are free for other tasks.

- Applications exist for paralyzed people who cannot speak nor move anything. The irony is that these people cannot move their heads so the accuracy of tracking is very high. A known application simulates a typewriter and is used for communication. The subject can look at a screen with on it a graphical version of a keyboard. Looking at a character for a specific *latency* time corresponds to typing it. This way they are able to write and they still have a means of communication. Although it is a slow process of typing a document, a journalist in New York uses this application to write his columns after he became paralyzed due to a car accident.

- Another application is known as "The Little Prince" storyteller and is described in [25]. The idea of this application is taken from the novel by Antoine de Saint Exupery with the same title. The story takes place in outer space on a planet inhabited by a little prince. The planet has several features, such as volcanoes, staircases and flowers orbiting the planet.

  The application uses a corneal/pupil reflection gazetracker. The system makes distinction between focusing on a single object or focusing on groups of objects of the same type. For example, when looking at a special volcano, information on this volcano is given, whereas looking at a group of volcanoes yields general information on volcanoes.

- More and more documents exist in digital format. One could think of an application where the eye movements of readers of a specific document are tracked. After the document has been read by a number of people, the author gets it back with parts highlighted depending on the reader's eye movements. For example, parts of the document that have a high interest by the readers can be either interesting or difficult to comprehend. This way the author gets useful information on his document.

### 4.3.1 Eyes in a user interface

Currently, mouse and keyboard are the most used means of communication with the computer. One might wonder whether the use of the user's eyes can add an extra value to the human-computer interaction. Table 4.1 shows an informal summary on how eyes can be used in a user interface.

Table 4.1: Table of eye movements and possible meanings.

| Eye movement | Meaning |
|---|---|
| 1. direction of gaze | The direction of the gaze is the point on the screen the user is looking at. This information can be used to control objects in a virtual space, as described in the next section. |
| 2. blinking of an eye | This can be interpreted as a mouse click, although the question arises whether one can impose this on a user. The active use of blinking seems a large physical burden for the user. On the other hand, the observation of blinking can provide information on the emotional state of the user as described in item 4. |
| 3. closed eyes | If the user's eyes are closed for a longer period this can indicate he or she is asleep. The system can produce auditive feedback, for example a beep, to regain the user's attention. |
| 4. eyes as part of the emotion of the face | The expression of the eyes can be used to determine the emotional state of the user. For example, a questioning look of the user can indicate he or she does not understand the situation, or needs more information concerning the object of interest. Another example: nervousness is often expressed by an increase of the frequency of blinks of the eyes. |

In the multimodal system, only the direction of the user's gaze is used, although the gazetracker used (see section 4.4) provides data on the blinks of the eyes, too. Next the use of gaze control in a user interface is described in more detail.

#### Gaze control

Replacing the mouse by a gaze control can have an advantage in speed. Normally, the eye first fixates on a location before the hand moves the mouse to this location, or in short: "the eye is there before the hand". When a user wants to click on an icon at the screen several actions take place. The mouse is grabbed, the user looks at the specific icon, the user moves the mouse pointer to the location and presses the mouse button. These actions often overlap in time, but a gaze control only needs the time to look at the icon. So, the time needed to move the arm to the mouse and to move the mouse to the icon are saved.

However, a number of disadvantages of gaze control as opposed to mouse control exist:

- There is no analogy to the mouse-button. Clicking and dragging objects is not possible.
- The mouse can be put aside whereas the gaze control is always there. Intentional and non-intentional control is hard to distinguish. A control to engage and disengage the intentional manipulatory looking seems necessary.
- The human eye does not move calm and controlled like the mouse can be moved. The eyes jump around from spot to spot. Fixing the eyes on a specific point for a longer period feels unnatural and might be a strain.

The first two problems can be solved using either a latency time (as described in the typewriter example above) or a manual button. However, this button contradicts the *hands free* advantage of gaze control. A better solution can be found by introducing other modalities, such as speech and hand input, as is done in this project.

The third problem is more difficult but special filters can reduce this problem.

## 4.4 The ISCAN gazetracker

The gazetracker used for the prototype is a desk mounted, non-intrusive type of gazetracker. It is an ISCAN RK-726, (see [14]) with a similar set-up as shown in figure 4.2. The camera is pointed at the user's left eye. The technique is based on the location of the centroid of the pupil relative to the corneal reflection. The frequency at which this is done is 60 Hz. The infrared light illuminator is mounted on the camera and the whole device is put below the monitor.

An eye-monitor is part of the system, showing the eye as illustrated in figure 4.3. The gazetracker is connected to a Pentium computer. It provides not only the coordinates of where the user is looking at, but also the pupil's diameter and the blinking of the eye. It is possible to store gaze patterns and to analyze them after the session.

Currently, the coordinates of the gaze are sent to the multimodal system after some preprocessing, which means translating the coordinates to the appropriate screen location. Sockets are used for the connection between the gazetracker's computer and the prototype. A sight cursor (circle and cross) is drawn on the screen to give the user feedback on the accuracy of the gazetracking. The gaze coordinates are used for further processing by the fusion agent. This will be described in the next chapter.

## 4.5 Conclusions and recommendations

The use of a gazetracker is found to be useful. In the application (see chapter 5) a little sight cursor is used to represent the user's eye on the screen. Although sockets are used for the connection, no delay is noticed. The process of tracking the eye, determining the location on the screen, and drawing the cursor on the screen, is real-time.

The speed at which certain commands can be issued is impressive. For example, while looking at an object the user can say *Delete this*, and the object is removed. In chapter 8 a more detailed discussion on the use of gazetracking in the developed prototype is presented.

The accuracy is the most critical aspect of gazetracking. Being one or two degrees off track is annoying and can result in wrong commands. The user is allowed to move the head but the degree of freedom is limited to about 20 cm. The problem of free movement of the head and the accuracy of the tracking are obviously dependent of each other. Following are two simple solutions:

- Integration of face-tracker
  A face-tracker uses a camera to track the face of a user. CAIP has developed its own face-tracker. The integration of a face-tracker and a gaze-tracker seems promising. The face-tracker determines where the face is, so the gaze-tracker only needs to find the eye within a smaller area. An easy way of integration would be the following. Calibrate both trackers and perform the same movements on the camera of the gazetracker as performed on the camera of the face-tracker.

- Self re-calibration
  Sometimes the gazetracker is inaccurate and by re-calibrating it the gaze-tracker is accurate again. During experiments it was found that calibration only needs to take 5 to 10 seconds for an experienced user. This is a reasonably short period so it is not a burden for the user, nor for the execution of the task if the gazetracker is calibrated once or twice during a session. However, currently it is impossible for the user to calibrate the gazetracker by him- or herself. There is always the need for an operator of the gaze-tracker. It would be useful if the user could calibrate the gaze-tracker alone. This could be done by automating the process of calibration. For example, by issuing the command *Calibrate* the system outputs *Start calibration* and the user needs to look at the eye cursor while it appears subsequently at the defined calibration points.

Of these two solutions, the first one wants to increase the accuracy, whereas the second provides an easier way to deal with inaccuracy.

Another solution that deals with the inaccuracy, without solving it, is a different interpretation of the output of the gazetracker. Instead of using the coordinates of the gaze one can use the raw gaze tracking data at a higher semantic level and determine the user's *object of interest*.

# Chapter 5

# Architecture

## 5.1 Introduction

In this chapter the architecture of the system will be discussed. To get a better insight in the design and the examples used in explanations, the application the system was built upon is described first. The type of application and the possible commands are given. Then, the actual design will be discussed. This includes a description of the used artificial intelligence techniques. Following is an elaboration about the spoken feedback the system generates. Finally, an number of multimodal issues are addressed, such as referring to objects and ambiguity.

## 5.2 The application

When looking for an application several criteria were taken into account. The application had to be multi-user, so multiple users can use it simultaneously. It had to be written in Java (see 6.1.1), the main language at the laboratory. Furthermore, the application had to be simple, though effective and visually attractive. The main idea was to build a simple prototype that could serve as a basis for complexer systems. In the end, a whiteboard application is chosen.

The whiteboard serves as a common visualization space which allows creating and manipulating objects. Objects can be lines, rectangles and circles. These can be moved around in a space which is shared by the different users. Several user can join a session, however only one glove and one gazetracker are available. Thus, the application still supports normal mouse input for the other users.

In figure 5.1 one can see a screendump of the application. The application still supports normal mouse input. By clicking on the buttons on the toolbar the user can select a function. The user can create colored objects, select, move, resize, and delete them. By extending the normal mouse interface with 3 modalities, the graphics and layout of the application has changed. The user is given two extra pointing devices, a glove and a gaze tracker (see 5.5.3 for an elaboration). The glove is represented by an icon resembling a hand. The user's 'eye' is represented by a sight icon. Down on the screen, on the status bar, the phrase recognized by the speech recognizer is printed. The possible feedback

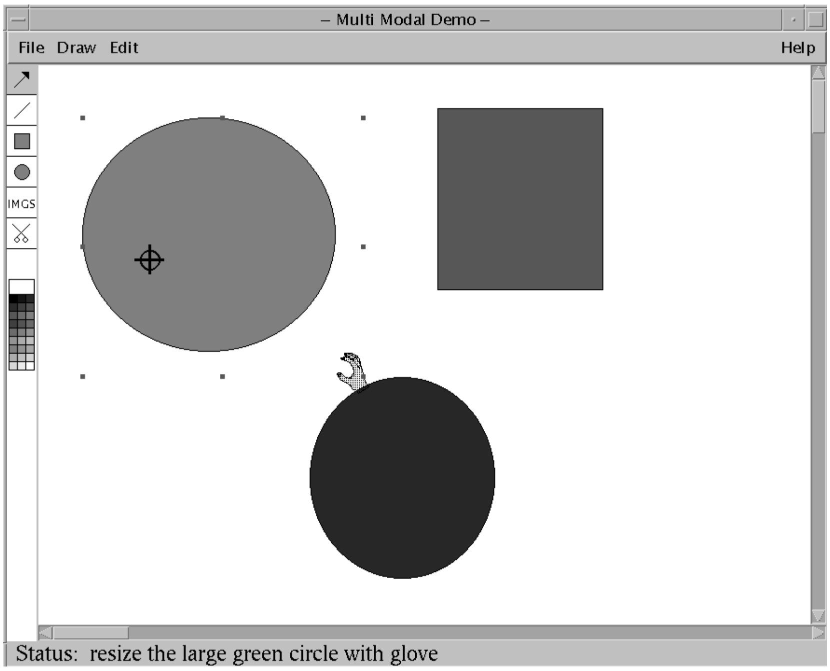


Figure 5.1: Manipulation of colored objects.

(see 5.4) given by the system is printed on the status bar, too. When no voice output is wanted the user can still read the system's feedback.

Half-way through the project the application is extended to serve as the basis for a military mission planning system. Functionality was added to the whiteboard. Figure 5.2 shows a screendump of the system. Objects now include military icons (such as tanks and helicopters) and the background contains a military map. On this map certain terrain features (e.g. hill, road) can be indicated by the users. For example, the utterance *Indicate hill nine with glove* results in activating a recording program which fills in appropriate map-tiles with information while the user simultaneously leads the glove along a certain region. The command *Finished recording* deactivates the recording program. The information is stored in memory, so the user can refer to that region. For example, *Move tank four to hill nine.* More on map representation can be found in [24].

### 5.2.1 The commands

In this paragraph the possible commands are described. Following an enumeration of the most important commands:

- create
  This command is used to create objects. In case of geometric figures, the command needs several parameters: type (line, circle or rectangle), color, and the two corner positions defining the size and position. When using military icons, a type (tank, helicopter) and identifier (one or two digits)

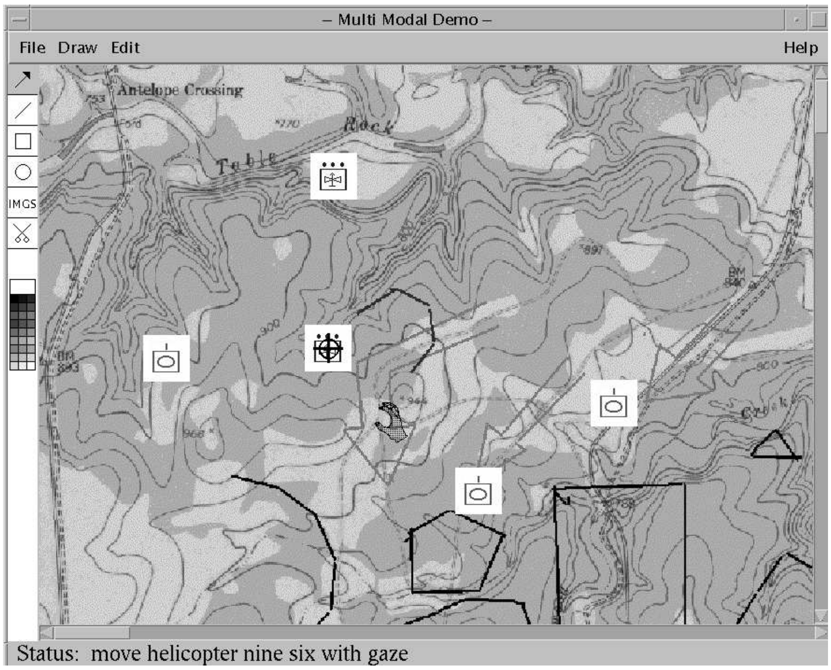


Figure 5.2: Manipulation of military icons.

is sufficient. A location can be given as a parameter, too. If a location is not added the user has to put down the icon on the desired location using either glove or gaze.

- select
  The selection of objects can be done by using a location or by describing the object:
  - the user looks at an object while saying *Select this*, or the user simply grabs the object with the glove.
  - the user can describe the object, for example *Select the large green rectangle*.
- unselect
  This command unselects the currently used object. It is also used to put down the object while moving it around.
- move
  For the movement of an object roughly two parameters are necessary. First, the object meant for the movement. This can be obtained through location or description (see select). Then, the destination of the object. This can either be obtained by pointing, looking or describing, using the map representation.

In table 5.1 one can see a matrix with in the first column the different possible commands. In the next columns the different input devices are given, and possible combinations. Each command with an X in a specific cell can be issued using the corresponding modalities.

Table 5.1: Table of commands.

| **Command** | **Speech** | **Glove** | **Sp. & Glove** | **Sp. & Gaze** |
|---|---|---|---|---|
| Create | | | X | X |
| Select | X | X | X | X |
| Unselect | X | X | | |
| Move | X | X | X | X |
| Delete | X | | X | X |
| Resize (in drawing program) | | X | X | X |
| Indicate (in planning system) | | | X | X |
| Identify (in planning system) | | X | X | X |

Certain commands can be issued using only one modality, either speech or glove.

By using *only speech*, one can select, delete and move objects. In case of a movement of an object, the user needs to describe the destination using the map representation.

By using *only glove*, the user can select, move and resize objects.

All commands can be issued using a combination of modalities. Note the importance of speech which is used almost always.

### 5.2.2 The command language

When using a speech recognizer, a grammar is essential, see chapter 2. In the test application, the initial task-specific vocabulary is small and contains about 150 words. Implemented is a simple context-free grammar. To get a better insight in the applications and to understand the described examples, part of the grammar is shown below. The following code fragment uses a common syntax in which:

- capitalized identifiers are non-terminals
- parenthesized identifiers are terminal symbols
- alternatives are separated by the symbol |
- a non-terminal between rectangular brackets [ ] is optional
- PHRASE is the start symbol

```
PHRASE          = META_PHRASE | CMD_PHRASE | NOUN_PHRASE |
                  PREP_PHRASE

META_PHRASE     = "forget it" | "leave it" | "end session"

CMD_PHRASE      = CMD_VERB [ NOUN_PHRASE ] [ PREP_PHRASE ]
                  [PREP_PHRASE ]

CMD_VERB        = "move" | "relocate" | "remove" | "delete" |
```

```
                  "resize" | "draw" | "create" | "place" | "put"

NOUN_PHRASE     = [ ARTICLE ] [ MODIFIER ] OBJ_TYPE |
                  "it" | "this"

ARTICLE         = "a" | "an" | "the"

MODIFIER        = COLOR [ SIZE ] | SIZE [ COLOR ] | "new"

COLOR           = "red" | "green" | "blue" | "black"

SIZE            = "short" | "small" | "big" | "large" | "long"

OBJ_TYPE        = "rectangle" | "box" | "circle" | "oval" |
                  "line" | ICON

ICON            = "tank" ID | "helicopter" ID

ID              = NUMBER [ NUMBER ]

NUMBER          = "one" | "two" | "three" | "four" | "five" |
                  "six" | "seven" | "eight" | "nine" | "zero"

PREP_PHRASE     = [ PREPOSITION ] POSITION_SPEC

PREPOSITION     = "to" | "from" | "at" | "through" |
                  "passing through" | "till" | "up to"

POSITION_SPEC   = "here" | "there" | "this position" |
                  "this point" | TERRAIN_FEATURE

TERRAIN_FEATURE = "hill" ID | "camp" ID
```

## 5.3 The design

When designing the system, one goal was to maintain a degree of flexibility. This means that the application can be changed easily and that new input devices are easy to add to the system. Latter is illustrated in the incremental implementation of the system itself. The three different modalities are added sequentially. The speech recognizer followed by the tactile glove and finally the gaze tracker.

In figure 5.3 the architecture is shown. Each of the three modalities has a processing agent who communicates with the fusion agent. Spoken language is chosen as the common means of communication. Each modality module sends a text string, possibly combined with an XY coordinate pair, to the fusion agent. Gesture information provided by the tactile glove is translated into appropriate written text by the gesture recognition module. Note that this process is similar to the translation of sign-language gestures into their spoken language representations.

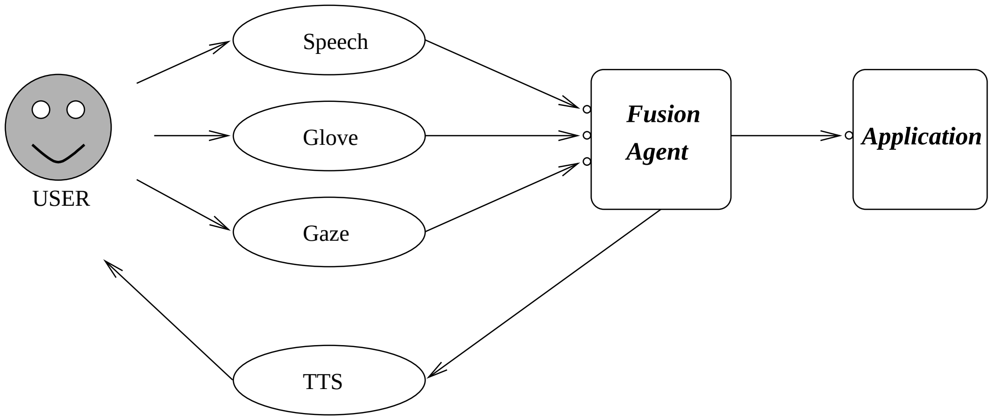


Figure 5.3: Global architecture of the system.

The fusion agent fuses the different inputs and produces either a command or feedback. The command needs to be executed by the application, while the feedback is processed through a text-to-speech synthesizer to produce spoken feedback. The multimodal fusion approach in the agent is a variation of the slot-filler method. This method is well-known in the artificial intelligence, see [28] and [1] Information necessary for command or language understanding is often encoded in structures called frames. First, the theory of frames is given and then the fusion agent is discussed.

### 5.3.1 Knowledge representation: Frames

In its most abstract formulation, a frame is simply a cluster of facts and objects that describe some typical situation. Frames have the property of inheritance which means that a frame can be a child of another frame, incorporating all its features. In the graphical representation inheritance is shown by the arrows pointing towards the father frame. (See figure 5.4). The principal objects in a frame are assigned names, called *slots*. In this application frames contain information about the commands given by multiple modalities.

Figure 5.4 shows the frame structure of the commands of our application. Each possible command is defined in a corresponding command frame. Different commands need different parameters, stored in the frame's slots. For example, the *Delete* command only needs a object id to determine which object must be removed. The *Create* command also needs parameters concerning the type of object, location and size.

#### Generic properties

Pre-set values in a frame, such as the value of the slot *Confirmation*, are called generic properties. In this case it means that the designer decides for every command whether it requires a confirmation or not. As can be seen in the diagram, the *Exit* command needs to be confirmed. (The generic slots are are filled in uppercase in the diagram.)

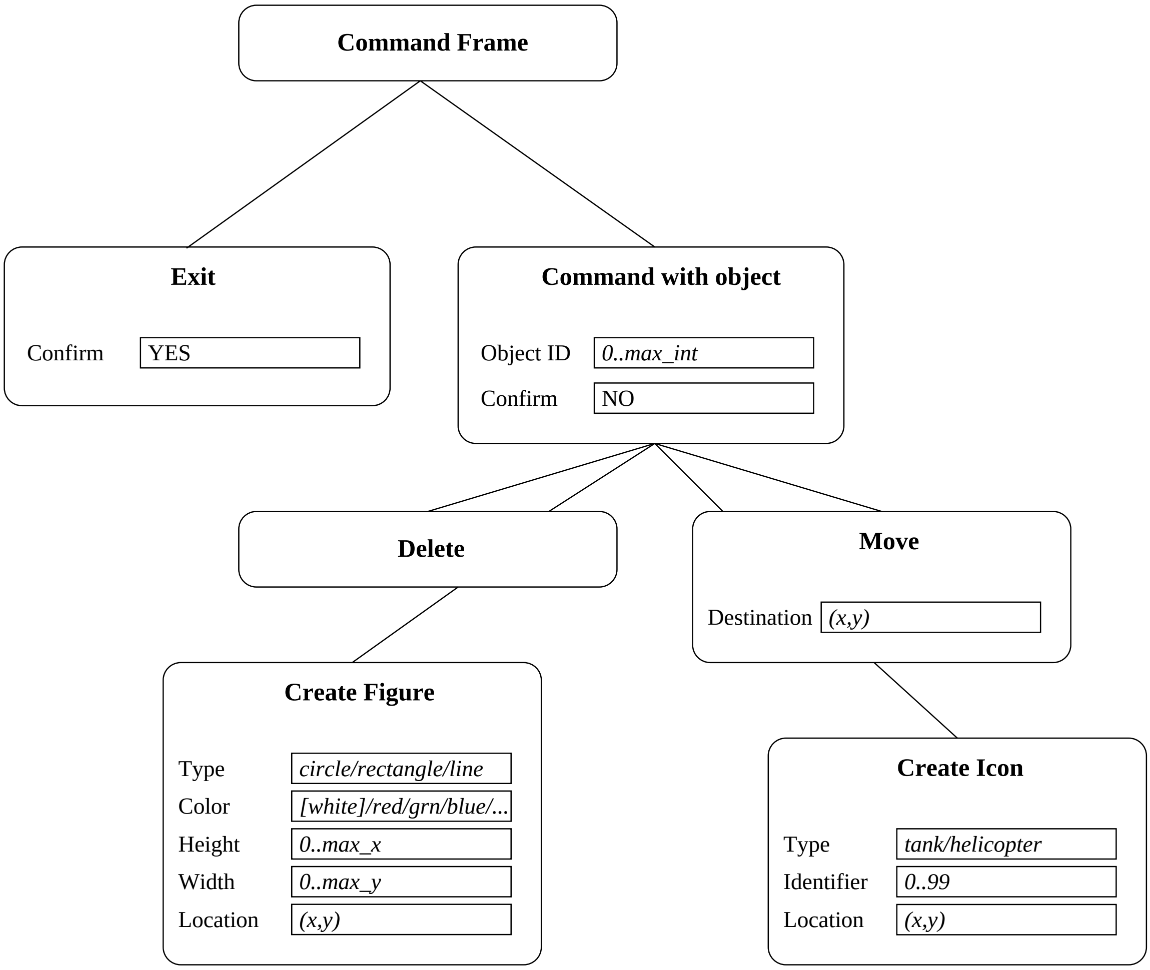


Figure 5.4: Example of Command Frame structure.

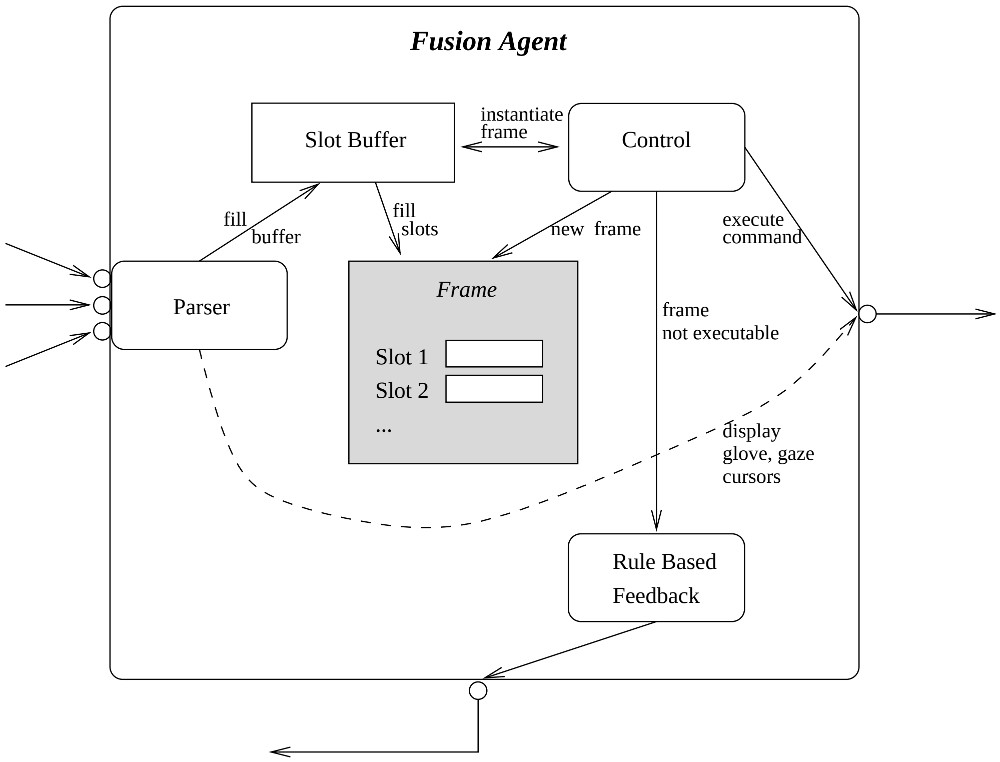


Figure 5.5: The Fusion Agent.

These pre-set values can be dependent of the skill of the user, or rely on a user model if user. For example, an expert user of the system might find each confirmation annoying while a beginner finds it very useful. Currently, this is not implemented in the system.

### Default values

Each slot can contain a default value which is assumed if no other value is known. In the diagram, the default values are shown between brackets. For example, if a circle is to be created and no color is given, white is the default color.

### Frame functions

Related to each command frame is a execution function, that is able to perform the execution of the command in the application. This function uses the slots of the frame to provide the appropriate parameters.

For more theory on frames see [8].

### 5.3.2 Fusion Agent

The fusion agent plays a central role in the architecture. In figure 5.5 a schematic representation of the fusion agent is shown. The function of the fusion agent is to fuse the multiple inputs and to start up a dialogue with the user. If sufficient information is available, the appropriate command will be executed. Otherwise, the fusion agent will come up appropriate feedback, which will be spoken to the user.

Each module of the fusion agent is described next.

#### Parser

First the parser parses the different inputs given by the three modalities. The main function of the parser is to install new slots in the buffer and assign values. The parser uses a keyword approach combined with some grammar constraints. See 5.2.2 for the grammar of the speech recognizer which more or less equals the parser's grammar.

#### Slot buffer

An important component of the fusion agent is the slot buffer. It stores the incoming values for all possible slots defined by the command vocabulary. This is essential because present information is often used in the future. Consider this example: the user selects an object by saying *Select the green circle.* Next, he or she can refer to this object using the word *it.* For example, *Move it.* If the object from the previous utterance is stored in a slot buffer, the the *Move it* command has sufficient information about what object to move. In other words, the slot buffer contains memory of past operations.

#### Rule Based Feedback

In section 5.4 the possible feedback of the system is described. The feedback generator is based on rules. A rule contains of a set of conditions and a conclusion or action. If the conditions are satisfied the actions of the rule are taken. Following are three example rules.

This rule produces feedback when the object described by the user does not exist.

```
IF
   ((command = MOVE) OR (command = DELETE) OR
    (command = SELECT) OR (command = RESIZE))
   AND (not exist (object))
THEN
   feedback (object 'does not exist.')
```

In a military mission planning scenario each object must have a unique identifier. This means objects with the same names must be prevented. The following rule takes care of that.

```
IF (command = CREATE) AND (objsort = MILITARY) AND
   (objtype = x) AND (objid   = y)
```

```
THEN
   feedback (x y 'already exists.')
```

The command to delete all objects of a specific type needs to be used carefully. This rule asks for confirmation.

```
IF
   (command = DELETE_ALL) AND
   (objtype = x)
THEN
   feedback ('are you sure you want to delete all' (multiple(x))
```

### Slot filling and command execution

The fusion agent parses the utterance and fills the corresponding slots in the slot buffer. Possible tactile glove positions or gaze positions are read when appropriate.

For example, the utterance *From here to here create a red rectangle*, causes the following slots to be filled in the slot buffer. The positions of the two opposite corners of the object, the object's type, the object's color and the operation or command type. If only the gesture recognition module is providing information, the parser relies only on that input.

In the current design, complementarity and concurrency problems are handled on a first-come first-serve basis. The system executes a command as soon as all necessary information is available. The information may be provided entirely or partly by one of the three modalities.

The instantiation of a particular command frame is done by analysis of information in the slot buffer. The slot buffer is monitored by the control module to determine if the command slot is filled. If it is not filled, the system will wait for more input information provided by any modality. If the command slot is filled, the corresponding command frame is instantiated. Next, the fusion agent examines whether sufficient information is available in the slot buffer to fill each predefined slot of that particular frame. If essential information is missing or ambiguity arises the system initiate feedback. When enough slots of the frame are filled the command is executed by the application.

To achieve maximum flexibility, the design assures that a particular frame contains a minimum number of slots that are necessary to unambiguously execute a command. For example, if there is a *blue rectangle* and a *red circle* in the workspace, it is sufficient to say *Delete the rectangle* because the *delete* frame only contains the minimum necessary information, i.e. the *object ID* slot. The utterance *Delete the large blue rectangle* generates the same command because it obviously contains the minimum information needed for the intended action.

## 5.4 Feedback

Feedback by the system is essential to establish a bi-directional interaction between user and system. Feedback can be spoken or printed. The goal of making the human-computer interaction resemble human-human interaction is satisfied more when the output by the system is also in a natural, human way. Spoken language is the main communication channel in human-human interaction.

That means one wants not only spoken input by the user by means of a speech recognizer, but also spoken output by the system by means of a speech synthesizer.

The system provides both spoken and printed feedback. The feedback is always printed and it can be spoken if the user wants so. If privacy is an issue, it is better not to have the computer to speak out load. On the other hand, even a little bit of spoken output can enable eyes-free interaction and can provide the user with the sense of having a conversational partner rather than speaking to an inanimate object.

The conversational feedback implemented in the system can be categorized as follows:

- Confirmation
  Certain dangerous or 'big' commands might need confirmation before execution. For example:
  User: *Exit the system*
  System: *Are you sure?*
  or
  User: *Delete all tanks*
  System: *Are you sure you want to delete all tanks?*

- Information retrieval
  It is very natural to ask the system a question and to get a spoken answer. For example:
  User: *What is this?* (While pointing/looking at an object)
  System: *This is tank seven*

- Missing data
  When the user does not finish his command the system asks for the necessary information. For example:
  User: *Create a tank*
  System: *Need to specify identifier for a tank*

- Semantic error
  To resolve semantic errors feedback can be used. For example, in our application each tank must have a unique identifier.
  User: *Create tank seven*
  System: *Tank seven already exists*

If the utterance is not recognized by the speech recognizer no feedback is given. This is not implemented as the user would get a notice (something like *Please, repeat your command.*) every time he or she tells something meant for other people, instead of for the system. During demonstrations of the system, the user often makes remarks to spectators.

## 5.5 Multimodal issues

In this paragraph a number of problems and issues are discussed concerned with the design of the multimodal system. First, the references to objects are discussed followed by an explanation on the timestamping of words in a sentence Then, the problem concerned with using two pointing devices is addressed. Next, the way ambiguity is solved in the prototype is described.

### 5.5.1 Referring to objects

Since most commands involve one (or more) object, the selection of an object is an essential issue. The new modalities add extra ways in which to describe an object as opposed to the clicking of the mouse in normal GUI driven applications. In our prototype, objects can be referred to in two distinct ways. The user can point at an object using either gaze or glove (see 5.5.3), or the user can describe an object.

Following, an evaluation of how an object can be selected:

- Clicking of the mouse.
  This was the only manner of selection supported by the original application and it is still available for the user.

- Describing in speech.
  For example consider the object described in the command *Select the small red circle*. Now every object needs to be checked for these three features, the shape being a circle, the color being red and the size being smaller than all other red circles. What description is sufficient is also dependent on the current state of the whiteboard. If there has been only one red circle created, the description *the circle* is enough, though *the small red circle* will work as well.

- Using anaphora.
  The user is also able to use a reference to a previously used object. If the user just has finished resizing a rectangle and he or she wants to move it, it is sufficient to say *Move it*.

- Using the glove.
  The user can simple select the wanted object by “taking” it with the glove.

- Using gaze and a pronoun.
  The user can say *move this* while looking at an object. The coordinates of the gaze will be taken as the position where to look for an object. The corresponding object will be selected and moved.

- Using the glove and a pronoun.
  The user can say *move this* while clicking with the glove at a certain object. Then this object will be subject to the movement.

#### Referring to objects using speech

As mentioned above, the user can refer to an object by describing it. Now, a more detailed discussion on this description is given.

In a military mission planning system each object (icon) is named uniquely by its type and an identifier. For example, only one *tank seven* exists. However, in a drawing application objects do not have a unique name. In our system geometric objects can be described using the object's type, color and size, and combinations of these two features. Size can only be used in comparison to other objects. Implemented is the concept of smaller and larger than. For example, in case of three circles the user can describe the smallest and the largest of the three. Thus, the "medium"-sized circle can not be described directly. But, if the three circles have (two or more) different colors, the color can be used as distinctive feature.

Referring relatively to other objects is useful as well, for example, *Select the green circle left of the big red circle.* Although it may look simple, implementing relative references leads to several problems. Not only does the number of objects in a sentence increase, they also have different meanings. The used keyword spotting parser is not well suited for this, although it is possible. Also, though concepts as *left* and *right* are relatively easy to define in a 2D situation, concepts like *next to* and *below* in a 3D situation involve a lot more natural understanding of the image. See [18] for more details.

In the current system, referring relatively to other objects is not possible.

### 5.5.2 Timestamps

To synchronize the spoken input with the deictic inputs timestamping of the words of an utterance is useful. By timestamping each word the system can deduce what location is meant with the speech input. The locations of the glove and gaze cursors need to be stored. The fusion agent can determine at what time, during the utterance of what word, what location is pointed at with gaze and glove.

Consider the following example:

```
Create a   green rectangle from here to here
  t1   t2   t3        t4     t5   t6  t7  t8
  |    |    |         |      |    |   |   |
  |    |    |         |      |    |   |   |
 xy1  xy2  xy3       xy4    xy5  xy6 xy7 xy8
```

Now the system knows at what location to draw the green rectangle with the desired size. The need for a (simulated) mouse click now disappears. This is especially useful since the modality gaze does not provide the user with a click ability.

Unfortunately, the Microsoft speech recognizer does not provide timestamping of the words, it only gives start- and endtime of the sentence. This is the reason why the glove has a gesture simulating a 'mouse click'.

In the next paragraph the issue of timestamping plays an important role as well.

### 5.5.3 Two pointing devices

The project's goal to integrate a tactile glove and a gaze tracker means that the system possesses two devices to point with. Actually, the system supports three pointing devices because the mouse can still be used as well. In practice it is found that the mouse is not used by the user if he or she is wearing the tactile glove.

A problem is the question how to decide which modality is meant by the user at a certain point during execution. This problem arises at two moments:

- When the user is pointing at something, either with gaze or glove, while giving a command. For example, *Delete this*. The problem is how to decide whether the user means the object he or she is looking at with gaze or pointing at with glove.
- When the user wants to use gaze or glove for a movement. For example, *Move tank seven*. Does the user want to move tank seven by gaze or glove?

A straightforward solution could be to demand from the user to explicitly tell the system which modality is meant. For example: *Move tank seven with gaze* or *Move tank seven with glove*. This is implemented in the prototype. An extension to this solution would be to set a preferred modality which the user can set. Then, if the user omits *with gaze* or *with glove* the system automatically chooses the preferred modality.

Still, this solution is not optimal and seems only practical for the second type of the problem. When a user looks at a certain object and wants it to be deleted, it should be sufficient to say the command: *Delete this*, instead of, for example: *Delete this, what I am looking at*.

One could also try to solve the problem by adding a user model to the system. This model keeps track of all actions by a specific user. In the beginning the user has to tell the system which modality is meant. After a while the system can use this information to (statistically) foresee which modality this user means at a certain point during execution.

Two facts were taken into account during the design:

- Early testing of the prototype proved that when users want to point at or grab an object with the glove they tend to look at it as well. On the other hand when a user looks at an object the glove is not necessarily nearby.
- The speech recognizer used in the prototype does not provide timestamps (see 5.5.2) which means that the system cannot deduce where to user was pointing at (either gaze or glove) while speaking. Therefore, a mouse click can be simulated by the glove (see 3.4.1).

This results in the following choices for the final design.

- If the user says *with gaze* or *with glove* the choice is obvious.
- If the user clicks with the glove (or mouse), that location is taken.
- Finally, the location pointed at by the preferred modality (usually gaze) is taken.

### 5.5.4 Ambiguity

Ambiguity arises when a command can be interpreted in more than one way. Especially, when natural language is used this problem can occur. In our prototype the language is a command language, but since speech is not the only

Table 5.2: Table of ambiguities and the system's solutions. Sp = speech, Gl = glove and Ga = gaze.

| Modality | Example | Decision |
|---|---|---|
| 1. Sp | The user says *Create tank*. | Feedback is generated: *Need to specify identifier for a tank* |
| 2. Sp, Ga | The user says *Move tank seven to hill nine*, while looking at hill two. | Tank seven is moved to hill nine. |
| 3. Sp, Gl (point) | The user says *Move tank seven to hill nine*, while pointing at hill three. | Tank seven is moved to hill nine. |
| 4. Sp, Ga, Gl | The user says *Move tank seven to hill nine*, while looking at hill two and pointing at hill three. | Tank seven is moved to hill nine. |
| 5. Sp, Ga, Gl | The user says *Move tank seven here*, while looking at hill two and pointing at hill three. | If the user clicked with the glove at hill three, the tank is moved to hill three, otherwise it is moved to hill two. |
| 6. Sp, Gl (gesture) | The user says *Move tank seven to hill nine*, while identifying *tank seven* with the glove. | Both commands are executed in order of which command was given first. If glove precedes speech, the system will first to the identification and then move the tank. If speech precedes glove, the system will move the tank and probably de the identification wrong, since the tank has disappeared from its original position where it was identified. |
| 7. Sp, Gl | The user is grabbing an object and dragging it around and says *Put down*, without changing the gesture. | The object will be put down on the place where it was said, and the dragging of the object is stopped. The user needs to restart the grab gesture to continue dragging. |

modality, the problem of ambiguity can still arise. In table 5.2 occurrences of ambiguity are summarized and the decision made by our system is given.

Concluding one can say that the system often decides in favor of speech. This is logical since a user is expected to mean *hill nine* if he or she says *hill nine*. The system also decides more in favor of gaze than glove. This decision is based on the following: if users point at point X, they tend to look at point X too, whereas, if users look at point X, they do not necessarily look at the same point.

### 5.5.5 Redundancy

Redundancy arises when the same command is expressed in different ways, or in our multimodal system "One modality helps the other". Following is a number of examples.

- The user is looking at the same object as he or she is grabbing with the glove.
- The user says *Move tank seven to hill nine* while looking from tank seven to hill nine, back and forth.
- The user says *Create tank seven here* while looking and pointing at the same position.
- The user make the "put down" gesture while saying *Put down*.

In our prototype, redundancy is not handled in a special manner and obviously does not produce problems.

Redundancy and ambiguity are opposite of each other. Where ambiguity can be a reason for problems, redundancy is seen as a big advantage of multimodal input. It should be used better in future work as described in chapter 8.

# Chapter 6

# Implementation

In this chapter the actual implementation of the prototype is presented. First, the programming languages used to write the prototype in are described. Then, the way the different devices are connected to the system is discussed. Subsequently, the important classes are explained followed by conclusions in the end.

## 6.1 Implementation languages

Integrating different input devices into one system involves also the integration of different platforms and languages. The main language used for this project is Java which is described in the next section. CORBA is chosen as the framework for the collaborative environment in which the application runs. A short introduction on CORBA follows in section 6.1.2. Finally, some parts that deal with low-level aspects are written in C.

### 6.1.1 Java

Java is used as the main language in this project. Java is a language developed at Sun Microsystems by a team headed by James Gosling in 1993.

Originally, it was designed to serve as a small language for consumer devices, such as cable TV switchboxes, and therefore it had to be able to run on different platforms. It was decided that Java would generate intermediate code (called byte-code) to be run by an interpreter. In the same period, the world wide web was growing bigger and bigger and the Java developers realized that Java could be used to make websites more interesting. Now, Java applets can be run by most web browsers, enabling the programs to be written only once, instead of rewriting the program for every other platform. (See [9] on the Internet for more information).

Besides the language being platform independent, it has other advantages. It is an object oriented language based on C++, so easy to learn for C++ programmers, but several things are different.

- Garbage Collector
  Java contains a garbage collector that manages the memory automatically. Objects that are no longer being referenced are removed from memory.

- No pointers
  Since Java has this garbage collector, the error prone pointers of C++ are not required. Pointers can lead to inefficient memory use, or memory "leaks" or just memory loss due to improper memory overwrite.
- Single inheritance
  In C++ it is possible to derive one class from several classes, called multiple inheritance. This feature can lead to complex programming. In Java a class can only be the subclass of one superclass.
- Less bugs
  Java features strong typing, no unsafe construct, no architecture dependent construct, and concurrency (multi threading) making it more bug-free than C++.

The main reason why this language is chosen can be found in the platform independency of Java. The prototype is developed on a SUN Ultra Sparc and it has been ported successfully and easily on a laptop pc. For reasons of physical portability, the glove and gazetracker were not used for the demonstrations on this laptop but these can be connected easily as well through the use of sockets, see 6.2.

### 6.1.2 CORBA

In 1989 the Object Management Group was formed (OMG) to help reduce complexity, lower the cost, and speed up the introduction of software application. OMG's mission is to promote the theory and practice of object technology for the development of distributed computing systems. OMG does this through the introduction of an architectural framework (OMA) with supporting detailed interface specifications. Implementations are the domain of vendors, end-users, and those developing products and projects to solve a particular computing or business problem.

The CORBA specification was introduced by OMG in 1991. It defines the Interface Definition Language (IDL) and the Application Programming Interfaces (API) that enable client/server object interaction within a specific implementation of an Object Request Broker (ORB). CORBA allows applications to communicate with one another no matter where they are located or who has designed them in what language.

More information on CORBA and OMG can be found on the Internet, see [13].

In our implementation, the JavaIDL of JavaSoft is used. The `JavaEd.idl` file contains the IDL specification of the program, defining the interfaces of the different packages. It shows which objects with which attributes are distributed among the users. Then several users can log on to the same session and manipulate object on the shared workspace.

## 6.2 Socket connections

The connections between the different input modalities and the fusion agent brought up a problem. In Java (version 1.1) the support for low level programming is limited. This is some of the price a language pays for being platform independent. In our case it was very difficult to access the com-ports of the computer. Sockets, on the other hand, are very well supported in Java. So, little programs written in C read in the com-ports and put the data from the gazetracker and tactile glove each on a socket. These sockets are read in again by the application. (These programs are called *Client_Glove* and *Client_Gaze*). The speech recognizer runs on a pc and is connected to the application through a socket connection as well.

It is found that the use of these Internet connections is easy, and although maybe not the nicest solution, it has proven to be fast enough.

See also [15] where the use of socket connections for a multimodal system is described.

## 6.3 Java classes

The code is divided in two main packages called `JavaSM` and `JavaEd`.

`JavaSM` contains only one class called `SessionManagerServer` and needs to be executed by the user at the start of a new session of the application. This class takes care of distribution of events to other participants of the session. The events, such as creation or movement of objects, are handled and distributed over the various clients.

The package `JavaEd` contains the more interesting classes. Several functionalities are implemented in this package. To make it easier and more reusable for future developers, the following subpackages are made within the package JavaEd:

- `JavaEd.Ed`, containing code about the actual drawing program.
- `JavaEd.Fig`, containing code about the figures (rectangles, circles, lines and icons).
- `JavaEd.Op`, containing code about the different operators that can be used on objects.
- `JavaEd.Map`, containing code about the map representation.

Figure 6.1 shows the structure of the (important) classes of the `JavaEd` package.

Now, the most important classes of both the package `JavaEd` are described.

### 6.3.1 JavaDraw

This file contains the main application, so in this file one finds, as the Java-syntax describes, a main function.

Only one `JavaDraw` object is created and it requires 3 parameters at startup. First the hostname of the computer on which the SessionManagerServer is running. The other two parameters are a name and a number that together identify the specific user.

The `JavaDraw` contains several objects which together form the whole application. The `Drawarea` object is included which is the whiteboard application. `JavaDraw`'s member objects also include all objects of the multimodal interface. So, the three modalities, `SpeechInput`, `GloveInput`, and `GazeInput` and

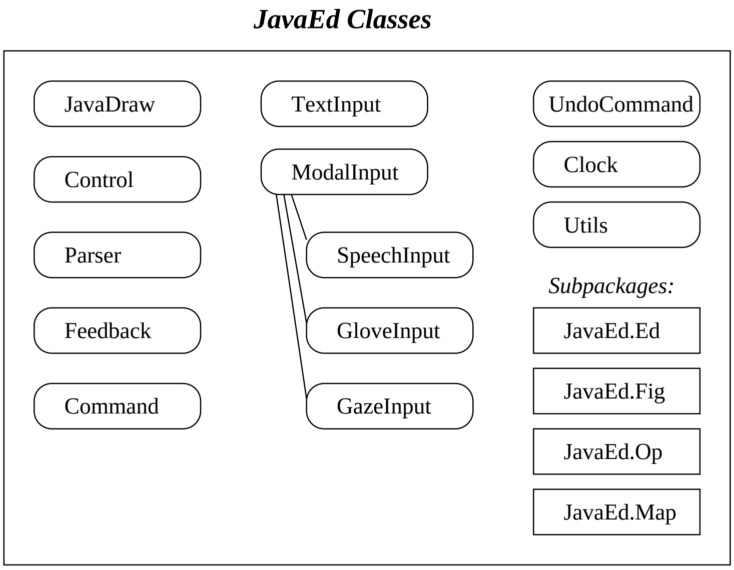


Figure 6.1: The JavaEd package.

the objects of the Fusion Agent as described in 5.3.2 are included. These are `Control`, `Parser` (which includes the Slot Buffer), and `Feedback`.

### 6.3.2 Drawarea

This object contains the graphical part of the application. It takes care of the drawing of rectangles, circles, military icons and so on. It also draws both the glove and the gaze cursor. It provides various access functions like `Move` for dragging, `MoveTo` for moving an object to a specific location, `Resize` for resizing, etcetera. The current status of the geometric figures and military icons is stored here.

### 6.3.3 ModalInput

This is an abstract class providing general functions for the connection between a modality and the main program. The three integrated modalities are all subclasses of this class. In particular, it handles the socket connections.

### 6.3.4 SpeechInput

This class serves as an intermediate between the Microsoft Speech recognizer (SR) and the interface of the application. At initialization it opens a socket which number corresponds with the socket opened by the SR. The SR then sends its local time over the socket so both clocks can be synchronized. For this purpose, a static class called `Clock` is implemented.

After synchronization this object stays in a loop and waits for output by the SR. The SR sends the recognized phrase over the socket and the start- and endtime of the utterance. This object reads in the phrase and the start- and endtime. Then, it performs some processing on the phrase which is of type string. It adds to each word the position of the mouse cursor, gaze cursor and glove cursor. Currently the SR does not provide timestamps for each word. It only gives the begin and end time of the utterance. To solve this, an array of mouse-clicks or "glove-clicks" is maintained in which each mouse event is stored. The SpeechInput object scans the array and takes the x,y-coordinates corresponding to deictic words like "this" or "here". In the near future a self-developed SR will be used which provides the timestamps so this 'work-around' can be avoided.

### 6.3.5 TextInput

For testing purposes, a class called `TextInput` which is similar to `SpeechInput` only it takes the input from a little textfield window instead of from the socket. After processing the input, this object sends its output to the parser. This way, the Speech recognizer can be simulated and no voice input is required.

### 6.3.6 GloveInput

This class is similar to `SpeechInput`. The tactile glove driver called *Client_Glove* is written in C and takes care of the tactile glove input. It sends x,y-coordinates over a socket which this class handles.

The x,y-coordinates are relative to the upper left corner of the screen. In this class the coordinates of the application window are subtracted from the input. Furthermore, the coordinates are multiplied by a constant factor (0.4 was empirically found) and averaged over 3 values. This is done to make the movements of the glove smoother and more natural. After this preprocessing the coordinates are sent to the drawarea. This drawarea draws at the corresponding position a so called glove cursor.

The user can also make a number of meaningful gestures as described in chapter 3. These gestures are translated into strings by the driver and send over the same socket. This object reads in the gesture text and sends it to the parser.

### 6.3.7 GazeInput

This class is similar to `GloveInput` in that it reads in two coordinates and sends these to the drawarea after preprocessing. This preprocessing includes multiplication by some factors as well as averaging over 4 values. The reason again was to make the movements smoother and more natural. However, the eye movements are still jumpy and another filter could be useful in future implementations.

### 6.3.8 Command

This class describes the command frames as described in 5.3.1. Each command describes its own frame with specific slots.

### 6.3.9 Parser

The parser contains a slot buffer and its main task is to fill the slots of this buffer. The parser uses a hybrid keyword / grammar approach. This means that it looks for certain grammar constructs, such as noun phrase or verb phrase, but this is done in a keyword spotting manner. After parsing the sentence (coming from `SpeechInput` or from `GloveInput`) the appropriate slots are filled. This object is monitored by the `Control` object described next.

### 6.3.10 Control

This object monitors the slot buffer and instantiates new command frames. Such a frame is an object of the class `Command`.

The `Control` object also handles the command frames. It has access to the methods provided by `Drawarea`. If the command refers to an existing object (for example *Move tank seven*) the `Control` object looks whether the object exists. If it does, the corresponding action is executed. If not, the `Feedback` object, described next, takes care of appropriate feedback.

### 6.3.11 Feedback

As described above, this object generates appropriate feedback for the user and sends this to the Microsoft Text-To-Speech (TTS) engine. It is rule based, as described in 5.3.2.

At construction it opens a socket connection with the computer which runs the TTS.

### 6.3.12 Map

In the future, the application will serve as a military mission planning system. To anticipate on this, a simple map representation has been implemented. In this object, the different terrain features are stored. The user can use the command *Indicate hill nine* and the program will store the location indicated by the user in this object. A tile representation of the map is used. The map is divided into several tiles, each tile containing one or more terrain features. All the tiles together represent the whole map. The terrain features are stored two-fold:

- Each tile maintains a list of its terrain features.
- Each terrain feature holds a list of the tiles it consists of.

Besides, one list is constructed containing the different features with attributes. So, the tiles only use references to the features. By storing the features twice the access time is reduced. When referring to a certain location the location meant can be found immediately. On the other hand, the terrain features of a certain location can be found directly, too.

### 6.3.13 UndoCommand

This class resembles the `Command` class. Each time an object of the class `Command` is executed, an `UndoCommand` object is constructed containing exactly the "opposite" command. For example, if a circle is created, an `UndoCommand` object is instantiated which defines the removal of this circle. An array of constant length (defined in the static class Utils of these objects is stored in case the user wants to undo several previously issued commands. When the array is full, the “oldest” object is discarded.

### 6.3.14 Utils

In order to make the code readable and flexible, well named constants for fixed values are used. In this static class, all these constants are stored together with a number of useful functions. For example, several conversion functions are put in this class.

## 6.4 How to run

In this section, explained is how to run the application. The Whiteboard has to be run as a stand-alone application. It could not be made to run as an applet due to the security constraints on applets. For example, an applet can connect only to the server from which it was downloaded.

One can compile all files in the WhiteBoard/java directory or use the pre-compiled files in the directory WhiteBoard/classes. To run the whiteboard, follow these steps:

1. Go to the Java classes directory:
   % cd /WhiteBoard/classes/
   and add this directory's path to your CLASSPATH environment variable.

2. Run the session manager (server):
   % java JavaSM.SessionManagerServer [-v] [portnumber]
   “portnumber” is the server port number and is optional. If it is not given, the SessionManagerServer uses a default port number. Once the SessionManagerServer runs, it will print the actual port number. This number is needed when connecting client whiteboards.

3. Run the whiteboard:
   % java JavaEd.JavaDraw <sessnMgr_URL> <ed_name> <ed_portnum>
   for example:
   % java JavaEd.JavaDraw biko.rutgers.edu:12121/javaSM ed1 7009
   sessnMgr_URL should be given in the following format:
   machinename:portnumber/SMname where machinename and portnumber are determined in step 2 above. SMname is the object name. ed_name and ed_portnum are any string name and a port number for this editor. Now, users can run any number of whiteboards on different machines repeating Step 3. This establishes a collaborative session.

   The following steps are needed to connect the different modalities. The order in which this is done is arbitrary but it is advised to connect the speech recognizer last. In that case the recognition process is not disturbed by noise while calibrating the gaze tracker.

4. Connect the tactile glove:
   % cd /WhiteBoard/classes/JavaEd/
   % Client_Glove

5. Connect the gazetracker:
   `% cd /WhiteBoard/classes/JavaEd/`
   `% Client_Gaze`

6. Connect the speech recognizer:
   Currently, a Microsoft speech recognizer is used running on a PC. Start up the following program:
   `D:\stimulate\SR\WinSR.exe`
   Click on OK to connect and click Start to start recognition.

Now the system is running supporting the three modalities. Modalities can be connected and disconnected at any time without causing the program to halt.

## 6.5 Conclusions

In this section conclusions concerning the implementation of the prototype are given. A few important issues are addressed.

First, the response time is the time which the program takes to execute the commands. This time should be short in order not to annoy the user and prevent him or her from repeating the command. Then, the robustness of a system tells whether it is protected against strange situations, and how often it “hangs”, meaning terminates the execution. Following are two issues that are more concerned with the program code itself. First, something is said about the bugs in the systems, followed by its flexibility. A system is flexible, if it is easy to change parts of the system and to adapt it to other situations.

#### Response time

The response time is good. All commands are executed well within a second after the command has been issued. One can say the system works real-time. Sometimes, due to a lot of network traffic, responses might take longer.

#### Robustness

Although the prototype has been developed in a short time period, it is reasonably robust. If something goes wrong with one of the modalities, the session can still continue. For example, if a wire of the glove gets disconnected, the user can continue work, and later on reconnect the glove.

#### Bugs

During development and testing (as described in chapter 8), a lot of bugs have been removed. Still, a totally bug-free program is never achieved as is the same in this project.

#### Flexibility

The implementation is reasonably flexible. After integrating speech, it was easy to integrate the glove gestures, and then the integration of the gazetracker was accomplished fast, too. So, the system has a flexible design if changing modalities is concerned.

Applying another application to the multimodal interface should influence only the application, parser, and feedback module. One attempt has been made to change the application. This went reasonably fast since the program was not very complex. However, if more complex application are used, it probably needs a bit more work.

# Chapter 7

# Testing and Human Factors Studies

In this chapter the testing of the system is discussed. This includes testing the program as well as performing usability tests. First, a photo of the prototype is shown, followed by a description of the demonstrations of the system given at CAIP. Then, the types of tests used to test the program are discussed. Then, it is explained why usability tests are required followed by a description of the video tape made together with military personnel. Finally, the prepared human factors studies are presented.

## 7.1 Introduction

Figure 7.1 shows a photo of the multimodal prototype. One can see all its components, the microphone array on top of the monitor, the glove worn by the user, the Polhemus sensor above the microphone array, and the gazetracker below the monitor. See figure 1.2 for a schematic picture which closely resembles this photo.

The picture of figure 7.1 was taken during one of CAIP's demonstrations for colleagues or company members of CAIP. During these demonstrations, I was always the user so I got well acquainted with the system. Being the programmer of the interface, this had the big advantage that the user (tester) and writer were the same person, so errors and bugs could be removed fast.

The usability of the system in terms of efficiency, ease to learn, etc., were not tested sufficiently by me while doing the demonstrations. That is one of the reasons why military personnel was asked to cooperate and to perform a demonstration which was video taped, see 7.3.1. Besides this, a human factors study was prepared, see 7.3.2.

Now, the testing of the program is described and in the next section the usability tests are addressed.

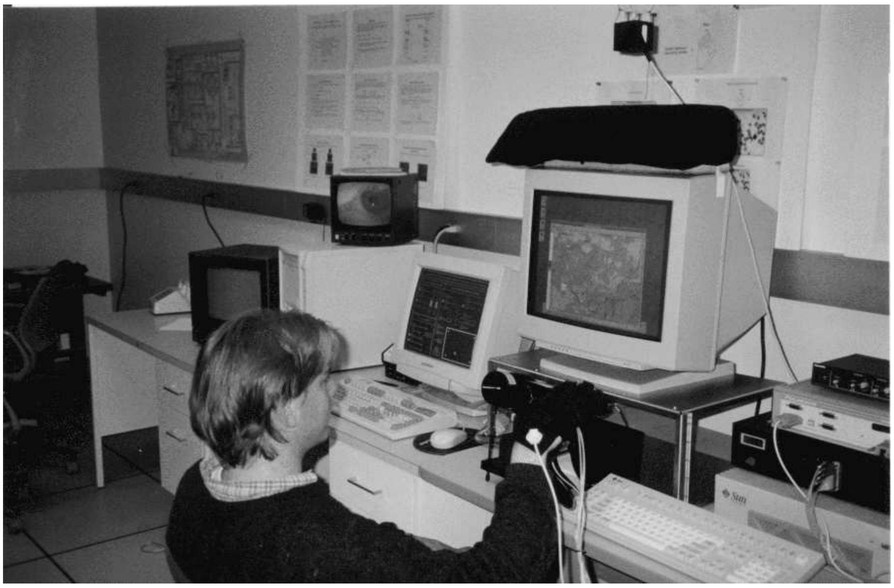

Figure 7.1: A photo of the multimodal system.

## 7.2 Testing of the program

As described above, a number of bugs are encountered and removed after demonstrations. Besides this, test sessions are done continuously.

To test the parser of the fusion agent, keyboard input is available besides speech input. This way possible errors of the speech recognizer can be avoided while testing the program.

The following classification of tests can be made:

#### Module testing

Each module of the system is tested before it is integrated into the rest of the system. For example, this was done for each modality before it was added to the prototype.

#### Integration testing

After the module has been tested alone, it is tested in the whole system. Also, if a command or new functionality is added, the system undergoes new tests in which these commands are tried in combination with other commands.

#### Configuration testing

One reason for using Java is to improve the portability of the system. Being developed on a Unix machine, it has been ported successfully and easily to a laptop pc. The program code did not have to be re-compiled, except for one class (`Utils`) containing several parameters, such as the names of the computers, that needed to be changed.

#### Time testing

Response times are tested for each command. One can say, all commands are executed within a second. Since Internet connections are used, response times may increase a bit if the network traffic is high, but even then, the response time is short enough.

#### Recovery testing

Working with several computers and devices means that the chances of failure of one of them increases. Each modality can be connected and disconnected to the main system whenever wanted. This means, for example if the computer of the gazetracker "hangs" it can simply be disconnected, restarted, and subsequently be reconnected. These tests were performed successfully for each device.

## 7.3 Usability tests

Testing the program is necessary to check the technical parts of the system. Usability tests are required to achieve acceptance by the user. These tests provide information about the changes in interaction involved in using the multimodal interface. Possible improvements can be obtained by performing these tests.

A usability test includes the use of the system by a group of persons, measuring features (such as speed and accuracy) while they use the system, and afterwards asking them questions about the use of the system. The first usability test conducted for this system can be seen in the demonstration made with the help of military personnel, as described next. Following is a description of the preparation of a human factors study.

### 7.3.1 Demonstration with military personnel

In cooperation with people of the army national guard training and training technology battle lab of Fort Dix, New Jersey, a demonstration was video taped to illustrate the usefulness of the multimodal and collaborative aspects of the system. This was the first time, people outside the development environment of the system used the multimodal interface.

The military mission planning scenario used for this demonstration (shown in appendix D) involves two players, a task force commander and a task force operations officer. The commander resides at a remote location using a normal mouse input graphical user interface whereas the operations officer uses the multimodal interface. A video conferencing tool was incorporated so both participants could see and hear each other.

At the beginning of the scenario, the task force commander commands the task force operations officer to execute a mission called "Op. Plan Right Guard". After authentification, the operations officer create three icons, called *Knight 06*, *Black 6*, and *Rook 16*, and he indicates three terrain features, called *Hill 968*, *Landing zone Camelot*, and *Checkpoint 1*. Then, these icons are move around on the map, by the operations officer using the different modalities and by the commander using normal mouse input. They do this in collaborative manner, supported by the video conferencing tool.

The command language is adapted to the military jargon. For example, the command *Put down* to release an icon on a certain location is changed into *Occupy*. This involves changes in the grammar of the speech recognizer and the grammar of the parser. These changes were easily made in the system, showing a good degree of flexibility of the system.

The conclusions concerning the usability of the system obtained by this demonstration are described in chapter 8.

### 7.3.2 Human Factors studies

The human factors studies of a system describe the usability of that system, or how well a system is used by its users. Its outcome can be used to improve the interface and to adapt it to the users' wishes and expectations.

Due to time problems (among others) the human factors study is prepared but not performed. Following is a description of how the study would look like.

A number of subjects is asked to perform several tasks in different ways while an observer observes the subject. Time measurements are made by the observer using a stop watch while the system provides log files containing time measurements, too. These time measurements show the start- and endtime of a command as well as the start- and endtime of the execution of the command, providing information about the length of the command, response time, and execution time.

During the session, the observer needs to fill in a number of tables. After the session, the subject is given a questionnaire he or she is asked to fill in. One can see that objective measurements combined with subjective opinions are obtained this way.

The tasks are divided in *Drawing* tasks and *Military mission* tasks, to keep it clear which part of the application is used, for both the user and the observer. In the beginning the user is asked to perform a task using a specific modality. After a while, when the user is more acquainted with the system, he or she is free to make a choice what modality to use. This way, one can determine the user's preference for a certain modality, and the relationship between which modality is preferred for what sort of command.

The whole test procedure for one subject can be divided in three main phases, each containing several steps:

1. **Preparation**
   - (a) Short introduction and explanation of the system and its capabilities and commands.
   - (b) Starting the program.
   - (c) Sitting in the proper position.
   - (d) Calibrating the gazetracker.
   - (e) Wearing the glove.
   - (f) Starting the speech recognizer.

2. **Test**
    (a) Drawing test; includes creating, moving, resizing, and deleting geometric figures.
    (b) Military mission test; includes creating, moving, identifying, and deleting military icons.
    (c) Choice test; now the user is free to choose which modalities he or she wants to use.
3. **Questionnaire**

In appendix A the tasks for the user are given, in appendix B the tables the observer needs to fill in are shown, and in appendix C the questionnaire to be filled in by the subject is given.

By performing these test sessions several differences can be observed.

- The differences between the two applications: the drawing program and the military mission planning system.
- The differences between the modalities in terms of usability, speed, accuracy, etc.
- The differences between the commands.

Very interesting are the relationships between the factors described above. For example, a certain command can be issued more by a certain modality, or one application is more appropriate for multimodal use than another. In the next chapter these relationships are addressed.

# Chapter 8

# Results and conclusions

In this chapter the results are presented and recommendations are given for the future. First, the question whether my goals and subgoals as stated in the first chapter are achieved, is answered. Then, conclusions on the use of the multimodal system are given, followed by an idea of future work.

## 8.1 Results

Following is a list of the goals of my work as defined in chapter 1, together with the obtained results.

- *Study the three modalities*
  I studied the three modalities and described each of them in one chapter. In chapter 2 both speech recognition and speech synthesis are described. Hand gesture input is addressed in chapter 3 as well as the Rutgers tactile glove. Finally, the use of eyes in a user interface, by means of a gazetracker is discussed in chapter 4.

- *Design an architecture and search for an application*
  I designed the architecture as presented in chapter 5. The whiteboard application was chosen for this prototype. Part of this Java application already existed but I had to add extra functionality. For example, I added an undo-command, the map representation and the use of military icons.

- *Implement the multimodal system*
  I implemented the system, meaning the gradual integration of the three modalities. First, speech was integrated, followed by the tactile glove and the gazetracker. A photo of the system can be seen in figure 7.1.

- *Test and debug the system*
  In chapter 7 the testing and debugging of the program is described. A lot of bugs are removed from the program and the system works good, but a totally bug-free program can never be claimed.

- *Conduct human performance studies*
  Together with military personnel a demonstration was video taped and

human factors studies are prepared. The human factors studies are prepared and described in the previous chapter, however they are not yet performed. Preliminary results are presented in this chapter.

## 8.2 Conclusions

The following conclusions are preliminary in the sense that they are made from my own experience and from the experiences gained during the demonstration with the military personnel. A better understanding can be achieved by performing the human factors study as described above.

First, the necessary instruction before a user can use the system is described, followed by conclusions concerning the application. Subsequently, all commands are discussed. Then, the features (such as speed and accuracy) of the different modalities are compared. Then, general conclusion are presented concerning the relationship between the type of command and the modality.

### 8.2.1 Instruction

Before being able to operate the multimodal interface, the user needs an instruction in which he or she is told about the capabilities of the application. The uses of the different modalities need to be learned.

For the speech recognizer that involves getting familiar with the command language. The language uses a normal Subject-Verb-Object-Location order, in which the subject is mostly skipped. It is a small, domain specific language, containing synonyms for a number of commands (for example, *delete* and *remove*). It is found that this language is easy to learn after a short practice and while operating the system.

The gestures need to be practiced to be able to use hand gesture input by the tactile glove. The number of gestures is limited which makes it easy for the user to remember the gestures. Moreover, most gestures are natural requiring a short instruction.

Finally, the user needs to get acquainted with the gazetracker. The first time it is a strange experience, since everywhere you look a cursor is following you. A short instruction is necessary for the calibration of the gazetracker and the user needs to be assisted for this.

### 8.2.2 Application

One main drawing program is used for this project and it is extended with features and functionality of a military mission planning system. The application is found to be effective to show the use of a multimodal interface. After a short introduction, users get familiar with the application and its commands.

The application is robust and easy to extend with extra features. However, changing the application into a real fully supported military mission planning system involves a lot more and one might consider to write a new application from scratch for this purpose.

### 8.2.3 The different commands

In this section the relationship between the sort of command and the most appropriate modality for it is investigated. Following is a description of each command, if necessary distinguished per application. The options a user has to issue the command are illustrated by examples. Then, these options are compared.

Since the *Put down* command which releases an object, is part of a number of other commands, it is discussed first.

#### Releasing an object

Command: Put down
Synonyms: Unselect, Place down
Examples:

1. Speech: *Put down*, and the object is released.
2. Glove: making the release gesture (open hand), and the object is released.

This command is almost always part of another command preceding it. It is found that the *Put down* command is issued using the same modality as is used for the preceding command. For example, if the glove is used for moving it is also used for the *Put down* command.

#### Creating a geometric figure

Command: Create
Synonyms: Draw
Examples:

1. Speech: *From here to here create a red rectangle*, while the glove points at the two corner positions of the rectangle, determining it size and location.
2. Speech: *Create a red rectangle with glove*, while the upper left corner of the rectangle is the point being looked at and the rectangle is resized by gaze immediately. The command *Put down* stops the resizing, and the size is determined.
3. Speech: *Create a red rectangle with glove*, while the upper left corner of the rectangle is the point being looked at and the rectangle is resized by glove immediately. The command *Put down* stops the resizing, and the size is determined.
4. Speech: *Create a red rectangle*, while the upper left corner of the rectangle is the point being looked at and the rectangle is resized by the preferred modality.

All options have their advantages and drawbacks. The first one does not require further commands, and a reasonable degree of precision can be achieved, however when the size is wrong, a new command needs to be issued. The second option, is useful if the desired size of the object is not known in advance. Resizing an object with gaze (option 3) is a strange experience in the beginning. The object is stretched everywhere you look. Option 4 results in option 2 or 3. In most cases, option 1 or 2 is used.

**Creating a military icon**

Command: Create
Synonyms: none
Examples:

1. Speech: *Create tank seven*, and the corresponding icon is being moved by the preferred modality (usually gaze) and is released after issuing *Put down*.

2. Speech: *Create tank seven here*, while clicking with the glove at a certain position. The icon is placed on that position.

3. Speech: *Create tank seven here*, while looking at a certain position. The icon is placed on that position.

The difference between creating a geometric figure and a military icon is that an icon has a predefined size, so only a location needs to be provided. Option 1 is preferred if the location of the icon is not known in advance. Since the size of the icon is not big, moving it with gaze in not a problem. Option 2 and 3 are preferred if the location is known.

**Moving to a predefined location**

Command: Move
Synonyms: Relocate
Examples:

1. Speech: *Move tank seven to hill nine*, and the tank is moved to the corresponding location. (Instead of *tank seven* one can also read *the red rectangle* in these examples.)

2. Speech: *Move tank seven here*, while clicking with the glove at a certain position. The icon is placed on that position.

3. Speech: *Move tank seven here*, while looking at a certain position. The icon is placed on that position.

If the map representation is used, the first option is preferable since exact locations with a meaning can be used. Option 2 or 3 are used depending on the user's preference for gaze or glove. In most cases though, gaze is preferred since it is faster: "the eye is there before the hand".

**Moving an object**

Command: Move
Synonyms: Relocate
Examples:

1. Glove: grabbing the object and moving it around, releasing it by issuing the *Put down* command. In this case, this is usually done by glove, too.

2. Speech: *Move tank seven with glove*, and the tank is moved wherever the glove points. The *Put down* command is used to release the object. (Instead of *tank seven* one can read also *the red rectangle* in these examples.)

3. Speech: *Move tank seven with gaze*, and the tank is moved wherever the user looks at. The *Put down* command is used to release the object.

4. Speech: *Move tank seven*, and the tank is moved by the preferred modality. The *Put down* command is used to release the object.

Grabbing the object with the glove seems the most natural way to do this task. Option 2 seems a bit laborious compared to option 1. Option 3 is used more for small objects than for big objects, because it is difficult for the user to look at the location where the object needs to be placed, without having the big object obstructing his view. Option 4 results in option 2 or 3.

**Resizing a geometric figure**

Command: Resize
Synonyms: none
Examples:

1. Glove: making the resize gesture and when the wanted size is obtained, releasing the figure by issuing the *Put down* command.

2. Speech: *Resize the red rectangle with glove*, and the rectangle is resized by gloved. The *Put down* command terminates the resizing.

3. Speech: *Resize the red rectangle with gaze*, and the rectangle is resized by gaze. The *Put down* command terminates the resizing.

4. Speech: *Resize the red rectangle*, and the rectangle is resized by the preferred modality. The *Put down* command terminates the resizing.

If the user prefers using the glove, option 1 is chosen. Option 2 is again laborious and option 3 is appropriate for small figures. Option 4 results in 2 or 3.

**Identifying a military icon**

Command: Identify
Synonyms: What is this
Examples:

1. Speech: *Identify* while looking at the desired object.

2. Speech: *Identify* while clicking with the glove at the desired object.

3. Glove: making the identify gesture at an object.

Option 1 is fast and easy while option 2 is slower because the hand needs to be move where the user is looking at. The choice between option 1 and 3 depends on the user's preference.

**Deleting a figure or icon**

Command: Delete
Synonyms: Remove
Examples:

1. Speech: *Delete tank seven*, and tank seven is removed.
2. Speech: *Delete this* while looking at the desired object and it is removed.
3. Speech: *Delete this* while clicking with the glove at the desired object and it is removed.

If the desired object is hidden behind other objects or difficult to see, option 1 is preferred. Option 2 is faster than option 3.

### 8.2.4 Modality vs. type of operation

If the commands are considered in conjunction with the modalities, three main types of operations can be distinguished: pointing, selecting, and moving. Pointing means specifying a certain location. Selecting means specifying a certain object for the command. Moving means using a modality to move an object to a certain location.

If one wants to compare the modalities, three features are important:

- Accuracy: how precise is the modality.
- Speed: how fast is the modality.
- Learning: how easy is it to learn to use the modality.

In table 8.1 the modalities are compared per type of operation in terms of these three features.

Table 8.1: Comparison of modalities per type of operation.

| **Modality** | **Accuracy** | **Speed** | **Learning** |
|---|---|---|---|
| **Pointing** | | | |
| (Speech) | | | |
| Gaze | – | ++ | ++ |
| Glove | – | + | + |
| Mouse | ++ | – | + |
| **Selecting** | | | |
| Speech | ++ | ++ | – |
| Gaze | + | ++ | ++ |
| Glove | + | + | + |
| Mouse | ++ | – | + |
| **Moving** | | | |
| Speech | ++ | ++ | – |
| Gaze | – | + | – |
| Glove | + | ++ | ++ |
| Mouse | + | + | + |

As can be seen in the first row, speech is not used in the pointing operation. Mouse is still the most accurate device for pointing. However, both gaze and glove are faster. Gaze, since the eye looks at the location before the hand is there, and glove, since grasping another device is avoided. All three modalities do not require much learning.

Selecting an object can be done accurately by all modalities, however mouse and speech are the most accurate. Speech and gaze are the fastest, gaze for the same reason as described above. Selecting by speech does require learning of the command language.

Moving an object to some location is best done with speech, if the location can be referred to by a name. Using glove is fast, natural, and does not require learning.

## 8.3 Future Work

In this section a few directions for future work are given and a new design is proposed. First, recommendations are given for each modality, followed by a number of general remarks and a new design.

### Speech

To synchronize the different inputs, timestamping the words of a sentence is essential. A speech recognizer should be used that provides these timestamps. The speech recognizer and the synthesizer currently run on two different machines. These two should be integrated on one machine.

The command language is small and domain specific. The bigger the language becomes, the harder it is to learn it. To reduce the amount of learning time, a more natural grammar can be better.

### Glove

The glove is not very precise and accurate for this kind of application. Changing to 3D could improve its usefulness.

The RM–II tactile glove features haptic feedback, which is not used in this prototype. In the future one might consider using this feature establishing not only multimodal input, but also multimodal output.

Lack of comfort is still a problem with the glove. The user is connected through cables to a central device, limiting him in his movements. If the performance of camera based techniques improves, changing to such a technique can be an option.

### Gaze

Gazetracking is fast but needs to be accurate to be useful. Improvements can be made by integrating a face tracker and by enabling the user to self-calibrate the device.

If gazetracking is always accurate, the need for a gaze cursor disappears, improving the naturalness of the system.

### Help files

Instruction given by experienced users is required for new users. Help files and better documentation can reduce this need for instruction.

**Conversational interaction**

Currently, the system feedbacks to the user if something is wrong, information is missing, or confirmation is required. However, more attention could be given to this conversational interaction in order to achieve real bi-directional interaction.

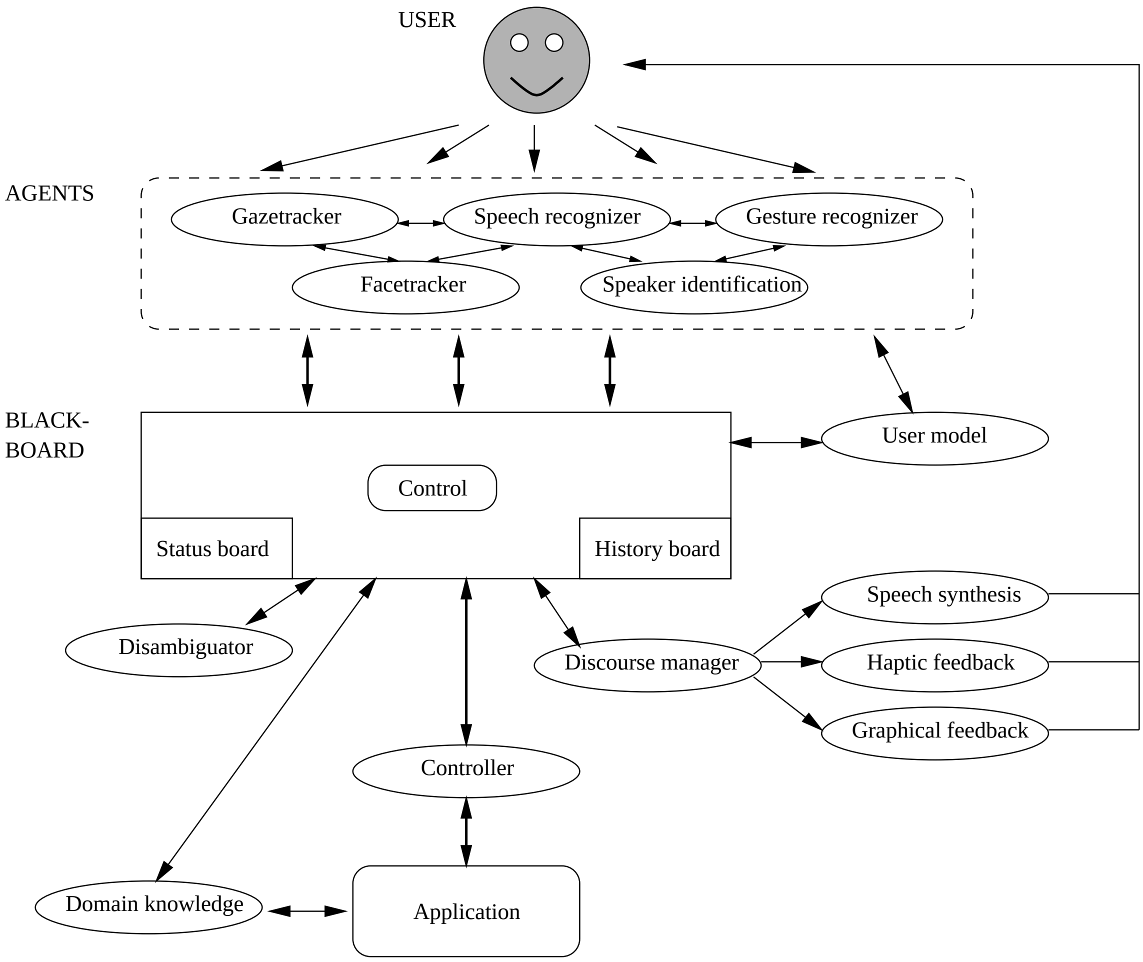


Figure 8.1: An idea of a new design.

## 8.3.1 New design

Although the design used in this prototype works well, it still resembles the mouse interface. Afterwards I think, one needs to forget all concepts of a normal mouse input graphical user interface when designing a multimodal interface. One should not remake the mouse input with the use of other devices but look for a totally different approach. The whole user should be observed and investigated to deduce his or her intentions. The user should not need to adapt to the system, but the system should adapt to the user. Natural language should be used instead of a command language that needs to be learned by the user.

The current design is simple in the sense that is one directional, it a sort of a pipe from input to output. The input is fused, a decision is made, and a command is executed or feedback is generated. A cyclic approach in which the separate agents communicate with each other can result in real bi-directional interaction.

In figure 8.1 a new design is proposed. In this design a number of independent agents process the different inputs by the user. Five input agents are shown: gazetracker, speech recognizer, gesture recognizer, facetracker, and a speaker identification agent. These agents communicate with each other and with a blackboard architecture by means of a meta language, for example an inter agent language like KQML (Knowledge Query and Manipulation Language). The agents communicate to improve each other's capabilities. A few examples:

- The facetracker, gazetracker, speech recognizer, and gesture recognizer can all help the speaker identification agent to determine who is using the system.
- The facetracker can give the gazetracker information about the position of the eyes.
- The gazetracker can provide information to the speech recognizer about the grammar to be recognized. For example, if the user's attention is on the video window showing the other user(s) the user probably wants to communicate with them instead of with the application. The speech recognizer can take advantage of this knowledge.
- The agents together provide information to the user model which retrieves and store the preferences and other features of this specific user.

The input agents communicate also with the blackboard. They put information on the blackboard and read in appropriate information. For example, the current status can be that the answer to a confirmation is required. This indicates the speech recognizer to expect a *yes* or a *no*.

The blackboard is controlled by a control agent and contains a status board and a history board. The status board contains information about the current status of the system whereas the history board holds information about the past.

The disambiguation agent resolves ambiguities between the different inputs and assign probabilities to them. A discourse manager takes care of the conversational interaction. It generates feedback and chooses the appropriate means of communication. This can be a possible combination of speech, haptic, and graphical feedback. The application's domain specific knowledge is stored in a knowledge base and is updated by the application.

A controller agent makes the final decision whether a command is ready for execution. If so, it issues the appropriate command to the application.

## 8.4 Conclusion

A look ahead in the future of human-computer interaction has been shown by this project. Although the three modalities each still have their drawbacks, together they can serve as a natural and robust means of communication with a computer. In the future, as speech recognition will become better, speech synthesis will sound more natural, gazetracking will be more accurate and less cumbersome, and the use of hand gesture input will be free of worn equipment, more can be expected from multimodal human-computer interaction.

# Bibliography


[1] J. Allen. *Natural Language Understanding.* The Benjamin/Cummings Publ. Co., California, 1994.

[2] R. Bolt. Put-that-there: Voice and gesturei at the graphics interface. *Computer Graphics*, 14(3):262–270, 1980.

[3] G. Burdea. *Force and Touch Feedback for Virtual Reality.* John Wiley & Sons, New York, 1996.

[4] A. Cheyer and L. Julia. Multimodal maps: An agent-based approach. *Proceedings of the International Conference on Cooperative Multimodal Communication*, 1995.

[5] A. Cheyer, L. Julia, and J-C. Martin. A unified framework for constructing multimodal experiments and applications. *Proceedings of the Second International Conference on Cooperative Multimodal Communication*, pages 63–69, 1998.

[6] P.R. Cohen, L. Chen, J. Clow, M. Johnston, D. McGee J. Pittman, and I. Smith. Quickset: A multimodal interface for distributed interactive simulation. *Proceedings of the UIST'96 demonstration session*, 1996.

[7] J. Flanagan and I. Marsic. Issues in measuring benefits of multimodal interfaces. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97)*, 1997.

[8] R. Frost. *Introduction to Knowledge Based Systems.* Collins Professional and Technical Books, London, 1986.

[9] http://java.sun.com.

[10] http://research.microsoft.com/srg.

[11] http://www.ai.sri.com/ooa.

[12] http://www.caip.rutgers.edu.

[13] http://www.omg.org.

[14] ISCAN Inc., Burlington, MA. *Operation instructions*, August 1994.

[15] X. Jing, J. Yang, M.T. Vo, and A. Waibel. Java front-end for web-based multimodal human-computer interaction. *Workshop on Perceptual User Interfaces*, 1997.

[16] M. Johnston, P.R. Cohen, D. McGee, S.L. Oviatt, J. Pittman, and I. Smith. Unification-based multimodal integration. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 1997.

[17] Q. Lin, C.W. Che, D.S. Yuk, and J.L. Flanagan. Robust distant talking speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 21–24, 1996.

[18] P. Olivier. Language and space: Coordinating the visual and verbal domains. *Proceedings of the Second International Conference on Cooperative Multimodal Communication*, pages 181–183, 1998.

[19] V.I. Pavlovic, R. Sharma, and T.S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(7), 1997.

[20] Polhemus Inc., Colchester, VT. *Fastrak User's Manual*, 1993.

[21] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 7(2):21–24, 1989.

[22] L.R. Rabiner and B.H. Juang, editors. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[23] S. Russel and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 1995.

[24] H. Samet and A. Soffer. Marco: Map retrieval by content. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(8):783–798, 1996.

[25] I. Starker and R.A. Bolt. A gaze-responsive self-disclosing display. *Proceedings of CHI*, pages 3–9, 1990.

[26] D.J. Sturman and S. Zeltzer. A survey of glove-based input. *IEEE Computerr Graphics and Applications*, pages 30–39, january 1993.

[27] A. Waibel, M.T. Vo, P. Duchnowski, and S. Manke. Multimodal interfaces. *Artifical Intelligence Review*, 10(3-4), 1995.

[28] P.H. Winston. *Artifical Intelligence, 3rd edition*. Addision-Wesley Publ. Co., 1992.

# Appendix A

# Task tables

Table A.1: Table of drawing tasks for the user.

| **Modality** | **Task** |
|---|---|
| mouse | Create a big red rectangle at the upper left corner |
| mouse | Create a small blue circle at the lower right corner |
| sp & gl | Create a big red circle at the upper right corner |
| sp & ga | Create a small blue rectangle at the lower left corner |
| mouse | Indicate a hill 9 |
| gl | Indicate a checkpoint 1 |
| mouse | Move the small circle to hill 9 |
| mouse | Move the big circle to hill 9 to cover the small circle |
| mouse | Recover the small circle |
| sp | Move the small rectangle to hill 9 |
| sp | Move the big rectangle to hill 9 to cover the small rectangle |
| sp | Move the small rectangle to checkpoint 1 |
| sp & gl | Move the big circle to cover the small rectangle |
| sp & gl | Recover the small rectangle |
| sp & ga | Move the small circle into the big rectangle |
| sp & ga | Recover the small circle |
| gl | Move the big rectangle to cover the small rectangle |
| gl | Move the big rectangle to the upper left corner |
| mouse | Resize the small rectangle to the size of the big one |
| sp & gl | Resize its size to the original size |
| sp & ga | Resize the small rectangle to the size of the big one |
| gl | Resize its size to the original size |
| mouse | Delete the big rectangle |
| sp | Delete the big circle |
| sp & ga | Delete the small rectangle |
| sp & gl | Delete the small circle |

Table A.2: Table of military mission tasks for the user.

| Modality | Task |
|---|---|
| mouse | Create a tank 1 at the upper left corner |
| mouse | Create a helicopter 1 at the lower right corner |
| sp & gl | Create a tank 2 at the upper right corner |
| sp & ga | Create a helicopter 2 at the lower left corner |
| mouse | Indicate a hill 9 |
| gl | Indicate a checkpoint 1 |
| mouse | Move tank 1 to hill 9 |
| mouse | Move tank 2 to hill 9 to cover tank 1 |
| mouse | Recover tank 1 |
| sp | Move helicopter 1 to hill 9 |
| sp | Move helicopter 2 to hill 9 to cover the helicopter 1 |
| sp | Move helicopter 1 to checkpoint 1 |
| sp & gl | Move tank 1 to cover helicopter 1 |
| sp & gl | Recover helicopter 1 |
| sp & ga | Move helicopter 2 to cover tank 2 |
| sp & ga | Recover tank 2 |
| gl | Move tank 1 to cover tank 2 |
| gl | Move tank 1 to the upper left corner |
| sp & gl | Identify tank 1 |
| sp & ga | Identify tank 2 |
| gl | Identify helicopter 1 |
| mouse | Delete tank 1 |
| sp | Delete tank 2 |
| sp & ga | Delete helicopter 1 |
| sp & gl | Delete helicopter 2 |

Table A.3: Table of tasks where the user can choose the modalities.

| Modality | Task |
|---|---|
| free choice | Create tank 7 |
| | Create a red circle |
| | Move tank 7 to hill 9 |
| | Move tank 7 to the right upper corner |
| | Resize the red circle |
| | Identify tank 7 |
| | Delete the red circle |
| | Delete tank 7 |

# Appendix B

# Observer tables

Table B.1: Table of time measurements.

| **Task** | **Time** |
|---|---|
| Initiating the program | |
| Sitting in proper position | |
| Calibrating the gazetracker | |
| Wearing the glove | |
| Starting the speech recognizer | |

Table B.2: Table of drawing test.

| Command | Modality | Object | Time | # tries | Fault by user | | | Fault by system | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | **Sp** | **Gl** | **Ga** | **Sp** | **Gl** | **Ga** |
| Create | mouse | red rectangle | | | | | | | | |
| | | blue circle | | | | | | | | |
| | sp & gl | red circle | | | | | | | | |
| | sp & ga | blue rect. | | | | | | | | |
| Indicate | sp & m | hill 9 | | | | | | | | |
| | sp & gl | checkpoint 1 | | | | | | | | |
| Move | mouse | small circle | | | | | | | | |
| | | big circle | | | | | | | | |
| | | big circle | | | | | | | | |
| | | small circle | | | | | | | | |
| | sp | small rect. | | | | | | | | |
| | | big rect. | | | | | | | | |
| | | small rect. | | | | | | | | |
| | sp & gl | big circle | | | | | | | | |
| | | big circle | | | | | | | | |
| | sp & ga | small circle | | | | | | | | |
| | | small circle | | | | | | | | |
| | gl | big rect. | | | | | | | | |
| | | big rect. | | | | | | | | |
| Resize | mouse | small rect. | | | | | | | | |
| | sp & gl | small rect. | | | | | | | | |
| | sp & ga | small rect. | | | | | | | | |
| | gl | small rect. | | | | | | | | |
| Delete | mouse | big rect. | | | | | | | | |
| | sp | big circle | | | | | | | | |
| | sp & ga | small rect. | | | | | | | | |
| | sp & gl | small circle | | | | | | | | |

Table B.3: Table of military mission test.

| **Command** | **Modality** | **Object** | **Time** | **# tries** | **Fault by user** | | | **Fault by system** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | **Sp** | **Gl** | **Ga** | **Sp** | **Gl** | **Ga** |
| Create | mouse | tank 1 | | | | | | | | |
| | | helicopter 1 | | | | | | | | |
| | sp & gl | tank 2 | | | | | | | | |
| | sp & ga | helicopter 2 | | | | | | | | |
| Move | mouse | tank 1 | | | | | | | | |
| | | tank 2 | | | | | | | | |
| | | tank 2 | | | | | | | | |
| | | tank 1 | | | | | | | | |
| | sp | helicopter 1 | | | | | | | | |
| | | helicopter 2 | | | | | | | | |
| | | helicopter 1 | | | | | | | | |
| | sp & gl | tank 1 | | | | | | | | |
| | | tank 1 | | | | | | | | |
| | sp & ga | helicopter 2 | | | | | | | | |
| | | helicopter 2 | | | | | | | | |
| | gl | tank 1 | | | | | | | | |
| | | tank 1 | | | | | | | | |
| Identify | sp & gl | tank 1 | | | | | | | | |
| | sp & ga | tank 2 | | | | | | | | |
| | gl | helicopter 1 | | | | | | | | |
| Delete | mouse | tank 1 | | | | | | | | |
| | sp | tank 2 | | | | | | | | |
| | sp & ga | helicopter 1 | | | | | | | | |
| | sp & gl | helicopter 2 | | | | | | | | |

Table B.4: Table of free tasks.

| **Command** | **Modality** | **Object** | **Time** | **# tries** | **Fault by user** | | | **Fault by system** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | **Sp** | **Gl** | **Ga** | **Sp** | **Gl** | **Ga** |
| Create | | tank 7 | | | | | | | | |
| | | red circle | | | | | | | | |
| Move | | tank 7 | | | | | | | | |
| | | tank 7 | | | | | | | | |
| Resize | | red circle | | | | | | | | |
| Identify | | tank 7 | | | | | | | | |
| Delete | | tank 7 | | | | | | | | |
| | | red circle | | | | | | | | |

# Appendix C

# Questionnaire

Please give score for the following feature, the range is from 1 to 5.
1 = hard to learn / hard to use / slow / low accuracy
5 = easy to learn / easy to use / fast / high accuracy

Table C.1: Table of scores per feature.

| **Modalities** | **Learning** | **Convenience** | **Speed** | **Accuracy** |
|---|---|---|---|---|
| Mouse | | | | |
| Speech | | | | |
| Glove | | | | |
| Gaze | | | | |
| Sp & mouse | | | | |
| Sp & gl | | | | |
| Sp & ga | | | | |

Place and X in the box which best describes your feeling toward the statement:

a. I was treated with respect and had full control over the system.
[ ] do not agree [ ] somewhat agree [ ] agree [ ] strongly agree

b. It is not difficult to remember all commands after a short training period (like this time).
[ ] do not agree [ ] somewhat agree [ ] agree [ ] strongly agree

c. It is not difficult to remember all gestures after a short training period.
[ ] do not agree [ ] somewhat agree [ ] agree [ ] strongly agree

d. I am satisfied with the performance of the system.
[ ] do not agree [ ] somewhat agree [ ] agree [ ] strongly agree

e. I feel most fault were caused by
[ ] speech recognizer. [ ] glove. [ ] gazetracker. [ ] program.

f. I find the following modality the most easy to use.
[ ] speech recognizer [ ] glove [ ] gazetracker [ ] mouse

g. I find the following modality the most difficult to use.
[ ] speech recognizer [ ] glove [ ] gazetracker [ ] mouse

h. I find the following modality the most powerful.
[ ] speech recognizer [ ] glove [ ] gazetracker [ ] mouse

Please write down any other remarks below:

# Appendix D

# Military mission planning scenario

The task force commander resides at the remote side using a normal mouse graphical user interface and is called the "remote" while the task force operations officer uses the multimodal interface and is called the "local".

| **User** | **Modality** | **Task** |
|---|---|---|
| Remote | | Commands local to execute "Op. plan Right Guard" |
| Local | Sp & ga | Create knight 06 |
| Local | Sp & ga | Create black 6 |
| Local | Sp & ga | Create rook 16 |
| Local | Sp & gl | Indicate hill 968 |
| Local | Sp & gl | Indicate landing zone Camelot |
| Local | Sp & gl | Indicate checkpoint 1 |
| Local | Sp | Move knight 06 to hill 968 |
| Local | Sp | Identify |
| Remote | | Tells local to be prepared to defend checkpoint 1 |
| Local | Gl | Move black six to checkpoint 1 |
| Remote | | Tells local that he will provide rook 16 to support |
| Remote | Mouse | Move rook 16 to checkpoint 1 |
| Local | Sp | Move knight 06 to landing zone Camelot |
| Local | Sp | Identify |
| Local | Sp | Delete black 6 |
| Local | Sp & ga | Delete rook 16 |
| Local | Sp & ga | Delete knight 06 |

# Appendix E

# Publication

At the Conference on Cooperative Multimodal Communication in Tilburg (The Netherlands), I presented the following paper at 29th of January 1998. This paper was accepted in the end of December at which time the gazetracker had not been integrated yet. Hence, gaze is not mentioned in the following document.

# Integration of Speech and Gesture for Multimodal Human-Computer Interaction

M. Andre[1,2], V. G. Popescu[1], A. Shaikh[1], A. Medl[1], I. Marsic[1], C. Kulikowski[1], J. Flanagan[1]

[1] Rutgers University, Piscataway NJ 08854-8088, USA
[2] Delft University of Technology, POB 356, 2600 AJ Delft, The Netherlands

**Abstract.** This paper presents a multimodal interface featuring fusion of multiple modalities for natural human-computer interaction. The architecture of the interface and the methods applied are described, and the results of the real-time multimodal fusion are analyzed. The research in progress concerning a mission planning scenario is discussed and other possible future directions are also presented.
**Keywords:** Multimodal interfaces, speech recognition, microphone-array, force-feedback tactile glove, military maps.

## 1 Introduction

Current human/machine communication systems predominantly use keyboard and mouse inputs that inadequately approximate human abilities for communication. More natural communication technologies such as speech, sight and touch, are capable of freeing computer users from the keyboard and mouse. Although they are not sufficiently advanced to be used individually for robust human/machine communication, they have adequately advanced to serve simultaneous multisensory information exchange, see Bolt (1980), Cohen *et al.*(1996b) and Waibel *et al.*(1995c). The challenge is to properly combine these technologies to replicate the natural style of human/human communication by making the combination robust and intelligent, see Flanagan and Marsic (1997a).

## 2 Problem Statement

The objective of this research is to establish, quantify, and evaluate techniques for designing synergistic combinations of human-machine communication modalities in the dimensions of sight, sound and touch in collaborative multiuser environments.

The CAIP Center's goal is to design a multimodal human-computer interaction system with the following characteristics and components:

- force-feedback tactile input and gesture recognition
- automatic speech recognition and text-to-speech conversion
- gaze tracking
- language understanding and fusion of multimodal information
- intelligent agents for conversational interaction and feedback
- applications for collaborative mission planning and design problems

Although the system is in an early stage of development, results concerning tactile information processing, robust gesture recognition, natural language understanding and multimodal fusion are promising.

The present study combines real-time speech input with asynchronous gesture input provided by the CAIP Center's Rutgers-Master II force-feedback tactile glove, Burdea (1996a)

Real-life problems of collaborative mission planning are being tested as potential applications of the multimodal interface. Ongoing research concerning multimodal manipulation of icons on military maps is reported. First, we briefly describe the functional system components. Then, the methods applied for language processing and sensory fusion are introduced and research progress is outlined.

## 3 System Components

### 3.1 RM–II Force-Feedback Tactile Glove

The Rutgers Master II (RM–II) system is a portable haptic interface designed for interaction with virtual environments, Gomez (1995b). Its two main subsystems, shown in Fig. 1, are the *hand–master* and the *Smart Controller Interface* (SCI).

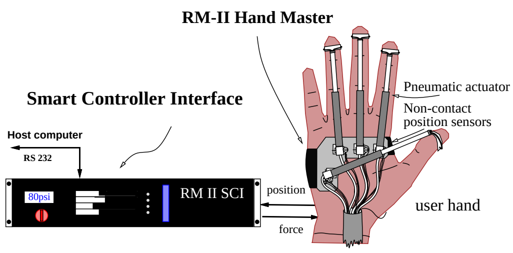


**Fig. 1.** The Rutgers Master-II force-feedback tactile glove.

The RM–II hand master can read hand gestures (fingertip positions relative to the palm) and apply forces on the fingertips corresponding to the interaction.

The hand-master unifies hand motion sensing and force display in a small, compact structue weighing 80 grams. The main structue consists of a small "L"-shaped palm base on which four custom-designed linear-displacement pneumatic actuators and 8 Hall-effect sensors are mounted. The actuators are equipped with a special sensor to measure their linear displacement. A 6D Polhemus tracker, Polhemus (1993), mounted on the back of the hand provides wrist position/orientation.

### 3.2 Hand gesture module

The hand gesture module is implemented using the RM–II system described above. Hand gesture input is particularly efficient for two kinds of tasks: 3D environment surfing (changing the viewpoint of virtual environment) and virtual objects manipulation. Since the glove was used in this application in a 2D graphic environment, only gestures for object manipulation were implemented. In addition, a special gesture was defined to emulate mouse clicks (to maintain compatibility with current GUI's). The design of the hand gesture module features only natural hand gestures in an effort to create an easy-to-use interface, see Pavlovic (1997b).

The first step in designing the hand gesture module involves implementing an object *selection function*. In a 2D environment, the coordinates are calculated where a virtual ray along the user's index finger intersects the screen plane and outputed when the corresponding gesture ("pointing") is executed. The selection point is further checked to see if it is inside any object in the environment. Other gestures designed for object manipulation are:

**grab** : used for grabbing and moving the selected object;
**thumb up** : associated with resizing the selected object;
**open hand** : corresponds to the 'unselect' or 'drop object' command; it is also used as a reset position for hand gestures;

**curling the thumb** : corresponds to mouse clicks.

Once a hand gesture is recognized, the corresponding string is generated and sent to the parser as indicated in Fig 3. In addition, a separate stream is continuously sending hand-pointing screen coordinates, as illustrated in Fig. 2.

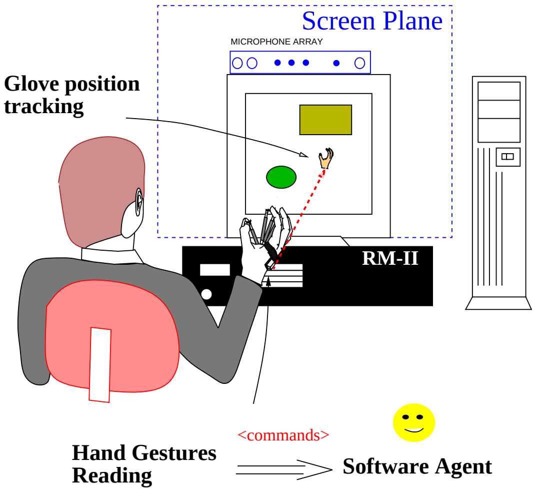


**Fig. 2.** Hand gesture interaction and glove display.

### 3.3 Speech Recognizer

The current system uses a Microsoft Whisper (1997d) speech recognizer engine with a finite-state grammar and a restricted task-specific vocabulary. The recognition is speaker-independent.

Once the speech recognizer has recognized a phrase, it sends the text to the parser together with two time-stamps. Although the current recognizer does not provide time-stamps after each word, it does provide the time at the start and end of each utterance. In future applications – to achieve temporal synchronization of the modalities –, time-stamps after every word of the utterance will be necessary to exactly determine where the user is pointing at (with tactile glove or mouse) while speaking.

Furthermore, the Whisper system exclusively runs under Microsoft Windows and is not portable to different platforms. Therefore, a CAIP-developed recognizer will be applied to solve these problems, see Lin (1996c).

### 3.4 Microphone Array

CAIP's microphone array technology liberates the user from body–worn or hand–held microphone equipment, permitting freedom of movement in the workplace. The current fixed focus line microphone array focuses on the speaker's head sitting approximately 3 feet from the monitor. Other sound sources are successfully attenuated.

A CAIP-developed microphone array system is applied as a robust front-end for the speech recognizer to allow distant talking, see Lin (1996c).

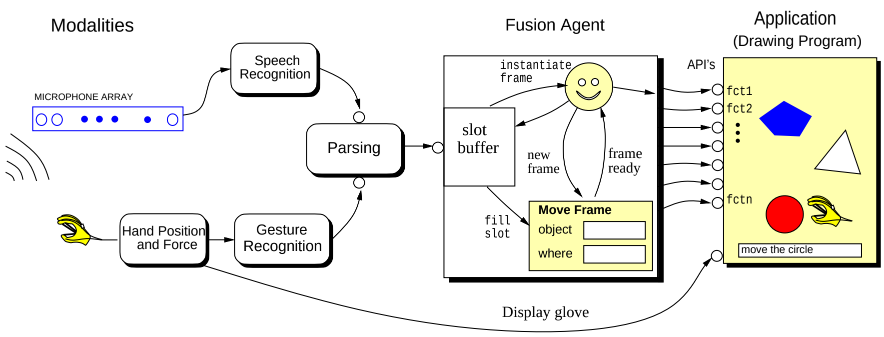


**Fig. 3.** Integration of tactile glove and speech input in a collaborative application.

## 4 Language Processing and Sensory Fusion

### 4.1 Parser

The first step in the understanding of multimodal commands involves parsing of the sensory inputs. In our system, the parser communicates with each modality and the fusion agent as illustrated in Fig 3. The reason for communicating through the parser is that we have chosen spoken language as the common means of communication. Gesture information provided by the tactile glove is first translated into written text by the gesture recognition module, and forwarded for further analysis to the parser. Note that this process is similar to the translation of sign-language gestures into their spoken language representations.

In the test application, the initial task-specific vocabulary is small and contains about 150 words. We implemented a simple context-free grammar as shown below. The following code fragment uses a common syntax in which:

- capitalized identifiers are non-terminals
- parenthesized identifiers are terminal symbols
- alternatives are separated by the symbol |
- a non-terminal between rectangular brackets [ ] is optional
- PHRASE is the start symbol

```
PHRASE          = META_PHRASE | CMD_PHRASE | NOUN_PHRASE | PREP_PHRASE

META_PHRASE     = "forget it" | "leave it" | "end"

CMD_PHRASE      = CMD_VERB [ NOUN_PHRASE ] [ PREP_PHRASE ] [ PREP_PHRASE ]

CMD_VERB        = "move" | "relocate" | "remove" | "delete" | "resize"
                | "draw" | "create" | "place" | "put"

NOUN_PHRASE     = [ ARTICLE ] [ MODIFIER ] OBJ_TYPE | "it" | "this"

ARTICLE         = "a" | "an" | "the"

MODIFIER        = COLOR [ SIZE ] | SIZE [ COLOR ] | "new"

COLOR           = "red" | "green" | "blue" | "black"
```

```
SIZE            = "short" | "small" | "big" | "large" | "long"

OBJ_TYPE        = "rectangle" | "box" | "circle" | "oval" | "line" | ICON

ICON            = "tank" ID | "helicopter" ID

ID              = NUMBER [ NUMBER ]

NUMBER          = "one" | "two" | "three" | "four" | "five" | "six"
                | "seven" | "eight" | "nine" | "zero"

PREP_PHRASE     = [ PREPOSITION ] POSITION_SPEC

PREPOSITION     = "to" | "from" | "at" | "through" | "passing through"
                | "till" | "up to"

POSITION_SPEC   = "here" | "there" | "this position" | "this point"
                | TERRAIN_FEATURE

TERRAIN_FEATURE = "hill" ID | "camp" ID
```

As mentioned above, synchronization of speech and gesture is essential in future developments. Currently, glove positions needed for the application are indicated by pointing gestures of the tactile glove and forwarded to the parser as position specifiers.

### 4.2 Multimodal Fusion

Our approach to multimodal fusion is a variation of the slot-filler method which is well-known in artificial intelligence, see Winston (1992) or Allen (1994). See also Cheyer (1995a). Information necessary for command or language understanding is often encoded in structures called *frames*. In its most abstract formulation, a frame is simply a cluster of facts and objects that describe some typical object or situation. In our application, frames contain information about commands given by multiple modalities, such as tactile glove and speech inputs. A similar architecture is described in Shaikh (1997c).

The principal objects in a frame are assigned names, called *slots*. For example, the command frame "delete" contains only one slot called "object ID," while the frame "create" should contain slots called "object type," "color," etc. The slots can be viewed as functions that take values to instantiate the actual frame. Thus a particular instance of a frame represents a structured set of knowledge.

Our approach to multimodal fusion is illustrated in Fig. 3. We are using a predefined set of frames corresponding to the possible commands in the current application. A slot writer-reader method which installs and retrieves slot values has been implemented as part of the fusion agent module. It uses the keyword approach combined with some grammar constraints.

*Slot Buffer* An important component of the fusion agent is the slot buffer. It stores the incoming values for all possible slots defined by the command vocabulary. This is essential because present information is often used in the future. Consider this example: the user selects an object by saying "Select the green circle." Next, he can say "Move it." If the object from the previous utterance is stored in a slot buffer, then the "Move it" command has sufficient information about what to move. In other words, the slot buffer contains memory of past operations.

*Slot Filling and Command Execution* The parser fills the slots in the slot buffer that are designated in the utterance using the slot-writer method and reads the tactile glove positions and gestures when appropriate.

For example, the utterance "*From here to here create a red rectangle*," causes the following slots to be filled in the slot-buffer: the positions of the two opposite corners of the object, the object's type, the object's color, and the operation or command type.

On the other hand, if only the gesture recognition module is providing information, the parser relies only on that input. In the current design, complementarity and concurrency problems are handled on a first-come first-serve basis. The system executes a command as soon as all necessary information is available. The information may be provided entirely or partly by either modality.

The instantiation of a particular command frame is done by analysis of information in the slot buffer. A demon monitors the slot buffer to determine if the command slot is filled. If it is not filled, the system will wait for more input information provided by any modality.

If the command slot is filled, the demon will instantiate the corresponding command frame and examine if there is sufficient information in the slot-buffer to fill each predefined slot of that particular frame. The demon must of course wait until all slots are filled. Then the command will be executed through the application programming interface (API).

To achieve maximum flexibility, the design assures that a particular frame contains the minimum number of slots that are necessary to unambiguously execute a command. For example, if there is a *blue rectangle* and a *red circle* in the workspace, it is sufficient to say "*delete the rectangle*" because the "*delete*" frame only contains the minimum necessary information i.e., the "*object ID*" slot. The utterance "*delete the large blue rectangle*" generates the same command because it obviously contains the minimum information needed for the intended action.

If the command slot of the slot buffer is not filled by the parser, then the system will wait for more input information provided by either modality.

## 5 Results and Discussion

Our current testbed uses a 150-word vocabulary with a finite grammar and a force-feedback tactile glove. The speech recognition works adequately for distant talking due to the microphone array. The fusion agent with the slot-buffer performs well even for asynchronous inputs from modalities. The command execution in the collaborative drawing program operates without substantial delays when tested in our laboratory. However, its current functionality is limited to manipulating (creating, moving, resizing, deleting, etc.) colored geometric objects and icons on topographical maps.

Fig. 4 illustrates how colored objects can be manipulated using simultaneous speech and tactile inputs. Fig. 5 shows a military mission planning map with the icon of a helicopter being moved by a command generated by the gesture recognition module alone. Here, "helicopter 96" is selected by the selection gesture and moved by the "*grab-move*" gesture.

## 6 Research in Progress

We intend to integrate a gaze-tracker to select and manipulate objects, adding further flexibility to the sensory input.

We are also proposing to investigate the possibility of a gesture recognition agent design to interpret possible commands given by gaze sequences.

An intelligent information agent is being designed to answer queries and provide information to the user about the actual state of the workspace (e.g., contents, positions and types of icons, history of operations, etc.). This will be a crucial element of a mission planning system which could be applied to disaster relief or rescue operations.

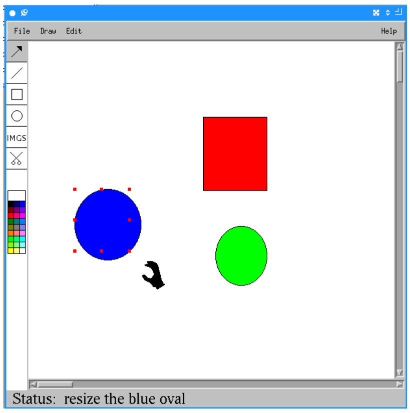


**Fig. 4.** Manipulation of colored objects.

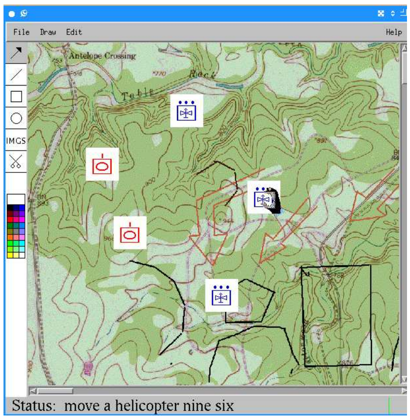


**Fig. 5.** Manipulation of military icons.

Text-to-synthetic speech answerback to the user will include notification about unrecognized phrases and unexecutable operations as well as acknowledgment of commands.

A mission planning scenario is being designed to evaluate the multimodal collaborative system. Geographical information is represented in map-tile format similar to that of the MARCO system (1996d). In order to achieve interactivity, the user should be able to indicate self-defined regions with the tactile glove pointer and speech. For example, the utterance "now I am indicating camp number six" results in activating a recording program which fills in appropriate map-tiles with information while the user simultaneously leads the pointer along the region to be defined as "camp number six." The utterance "finish recording" deactivates the recording program. The information is stored in memory, so the user can refer to that region (e.g., "move tank four six to camp number six").

A structure for knowledge representation is being designed to include a description of **a)** the map-content (i.e., topographical or man-made features) which is expected to remain invariant during a problem-solving session, **b)** the icons expected within the session, including their attributes and hierarchical decomposition into subicons, **c)** rules constraining the expected and allowed behavior of icons within the map-context.

For more information, see: http://www.caip.rutgers.edu/multimedia/multimodal/

## 7 Acknowledgment

Components of this research are supported by NSF Contract No.IRI-9618854, DARPA Contract No.N66001-96-C-8510, and by the Rutgers Center for Computer Aids for Industrial Productivity (CAIP).

## References

Allen, J. (1994) Natural Language Understanding. The Benjamin/Cummings Publ. Co., California,

Bolt, R. (1980) Put-That-There: Voice and Gesture at the graphics interface, Computer Graphics, Vol. 14-3, pp. 262-270.

Burdea, G. (1996) Force and Touch Feedback for Virtual Reality. John Wiley & Sons, New York.

Cheyer, A. and Julia, L. (1995) Multimodal Maps: An Agent-based Approach. International Conference on Cooperative Multimodal Communication (CMC/95), Eindhoven, The Netherlands.

Cohen, P.R., Chen, L., Clow, J., Johnston, M., McGee, D., Pittman, J., and I. Smith (1996) Quickset: A Multimodal Interface for Distributed Interactive Simulation. Proceedings of the UIST'96 demonstration session, Seattle.

Flanagan, J., Marsic, I. (1997) Issues in Measuring the Benefits of Multimodal Interfaces. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97), Munich, Germany.

Gomez, D., Burdea, G., Langrana N. (1995) Modeling of the RUTGERS MASTER–II haptic display, Proc. ASME WAM, DSC-Vol. 57-2, pp. 727–734,

Lin, Q., Che, C.-W., Yuk, D.-S. and Flanagan, J. L. (1996) Robust Distant Talking Speech Recognition. Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'96), pp.21-24, Atlanta, GA.

Pavlovic, V.I., Sharma, R. and Huang, T.S. (1997) Visual interpretation of hand gestures for human-computer interaction: A review. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.19, No.7

Polhemus, Inc. (1993) Fastrak User's Manual, Colchester, VT.

Samet, H. and Soffer, A. (1996) MARCO: MAp Retrieval by COntent, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 18, No.8., pp. 783-798.

Shaikh, A., Juth, S., Medl, A., Marsic, I., Kulikowski, C. and Flanagan, J.L. (1997) An Architecture for Fusion of Multimodal Information, Proc. Workshop on Perceptual User Interfaces, Banff, Alberta, Canada.

Whisper Speech Recognizer by Microsoft Corp., www.research.microsoft.com/research/srg/whisper.htm

Waibel, A., Vo, M.T., Duchnowski, P., Manke, S. (1995) Multimodal Interfaces. Artificial Intelligence Review, Vol.10, No.3-4.

Winston, P.H. (1992) Artificial Intelligence, 3rd edition. Addison-Wesley Publ. Co.