Speech and Gesture Integration for a Game-Based Command and Control Environment

by

Mary Carolyn Obelnicki

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering
[M. Eng.]

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

© Massachusetts Institute of Technology 2000. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2000

Certified by .....
Alex P. Pentland
Academic Head, MIT Media Arts and Sciences Program
Toshiba Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUL 27 2000
LIBRARIES
ENG

Speech and Gesture Integration for a Game-Based Command and Control Environment

by

Mary Carolyn Obelnicki

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2000, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

A computer's ability to interpret speech and its ability to interpret gesture have both been researched independently in great depth. The challenge now is to combine these two modalities in a single system, taking advantage of the means of expression available in each domain. For this thesis we investigated two different approaches to combining speech and gesture into a multimodal system.

In the first system, *Ogg That There*, we took a high-level approach; gesture was immediately represented as the indicated object, and speech was immediately encoded as words. Interpretation is done using a simple FSM, which steps through a subject-verb-object command grammar based on the speech input. Gesture is used only to dereference deictics in the speech input. This resulted in a functional, but rigid application, which serves as a standard of comparison for future approaches.

For the second approach we investigated combining speech and gesture at a very low level. Our goal was to take advantage of correlations between speech and gesture data streams to recover from ambiguities in the corresponding data stream. We believe that integrating speech and gesture streams in symbolic form risks missing important correlations among the data. This experiment showed evidence of improved classification when speech and gesture are processed together, as opposed to the classification with either the speech or gesture data alone. It also provided insights about finding and representing low-level gesture primitives, using of phoneme probabilities as input, and synchronizing speech and gesture primitives.

Finally, this thesis discusses potential improvements to our original system, as well as additions that could be made to the system to expand the user's freedom of expression in our multimodal environment.

Thesis Supervisor: Alex P. Pentland
Title: Academic Head, MIT Media Arts and Sciences Program
Toshiba Professor of Media Arts and Sciences

Acknowledgments

First, I would like to thank my thesis advisor, Sandy Pentland, and the fearless leader of the Vismod Netrek Collective, Chris Wren, for guiding me through the many revisions of my project.

I would also like to thank my family and friends, who never let me feel alone in my struggles to finish. I thank my parents, for letting me attend MIT in the first place and my siblings, Anne and Frankie, for the rivalry that drives us to succeed.

I thank Abigail, my invaluable friend and editor, without whom my thesis would still be a mess of late night babble.

I thank Owen, who made it his personal duty to ensure that writing my thesis didn't also become a hunger strike and Julia for her empathy.

Finally, I thank Frank, who put up with me through my late-night, thesis crazed, rantings.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Independently, a computer's ability to interpret speech and gesture has been researched in great depth[11, 12, 2, 1, 25, 14] . The challenge now is to combine these two modalities in one system, taking advantage of the means of expression available in each domain. Along these veins, this thesis aims to combine speech and gesture systems developed at the Media Lab, which have shown independent success[28, 23, 20], into a new multimodal approach to identifying purposeful human action in speech and gesture data.

Beyond merely combining the independent systems, we investigated co-processing speech and gesture at a low level to take advantage of potential correlations between the data streams that may be lost when the information is abstracted to a symbolic representation.

Combining the two data streams, however, brings new issues to the forefront. These issues include underutilizing gesture data, finding the best representation of low-level speech and gesture information, and creating application-interface synergy (i.e., ensuring that the pace of the interface is well suited to the application, and the that user can communicate all necessary information).

To work toward a successful integration of speech and gesture data, we investigated two different approaches. In the first experiment, using the *Ogg That There* system,

we took a high-level approach; gesture was immediately represented as the indicated object, and speech was immediately represented as the coded words of the program. It was developed to test the feasibility of Netrek (discussed below) as our experimental test-bed and to do a first-pass determination of the interface requirements for our modified Netrek application. It was also used to create a standard of comparison for our second approach.

For the second approach we investigated combining speech and gesture at a very low level – before the representations became too symbolic. Speech and gesture were integrated at the level of phoneme probabilities and gesture primitive probabilities. This data was processed together using COGNO , a system that does automatic data clustering and alphabet discovery. Our goal here was to take advantage of low-level correlations between speech and gesture data streams, and to avoid introducing human assumptions about how speech and gesture interact into our processing.

Both of these systems were investigated using Netrek (described below) as the test-bed application.

1.2 Netrek Description

To achieve the goals stated above, this project used the the game Netrek as the arena to test our speech and gesture integration methods. The game provided a context of things for the user to talk about. Our multimodal integration methods could then be evaluated in terms of the user's success or failure in playing the game – i.e., did they win? The advantages of the Netrek platform are further explored in section 1.2.4

Netrek is a real-time Internet game of galactic domination based on the Star Trek world. During a standard game, as many as sixteen ships on two teams vie for control of the universe. Traditionally the two teams are The Federation (yellow) and The Romulans (red). The game ends when one team controls all the planets and thus the universe.

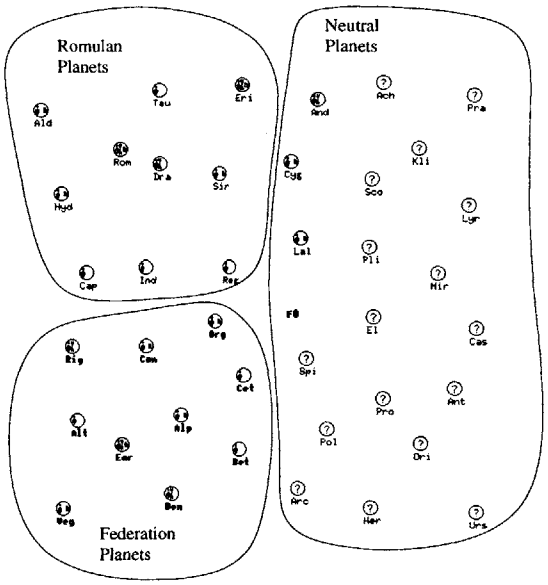

Figure 1-1: The standard Netrek universe. The planets are grouped as they are allocated at the beginning of a standard game.

1.2.1 Planets

At the beginning of a game, the planets are divided equally between the two teams. For each team, one planet is designated as the home planet. The home planet is deep in team territory and is where ships regenerate when they "die" (i.e., are destroyed by the enemy). The default home planets for the Federation and Romulan teams are Earth and Romulus respectively. (figure 1-1) A planet may also contain resources for its team members and allies. These resources are repairs, extra armies and fuel.

To take a planet, a team must overwhelm the planet with its own armies or those of its allies ("friendly armies"). This is usually done in two steps. First, the attacking team bombs the planet until a minimal number of outsider armies (three or four) remain on the planet. Then, the attacking team must transport friendly armies to the planet to overwhelm the remaining armies. The planet is taken when more friendly armies are placed on the planet than the number of outsider armies that were originally located there.

1.2.2 Ships

Players can choose from six different ships. These ships differ in characteristics like maximum speed, armament/shield strength, firepower and maximum capacity to carry armies. Choices include fast, fragile scouts; slow, sturdy battleships; and the average-at-everything, all-purpose cruiser.

During the course of the game, players participate in a number of well-defined tasks, including, dogfighting, ogging, bombing, taking planets, escorting fellow team members and defending planets.

Dogfighting is basic battle with enemy ships. Dogfighting is a means to an end, not the primary objective. It is necessary, however, because a player cannot carry armies in a ship until that ship has registered "kills" (destruction of an enemy ship). The more kills a ship has, the greater percentage of that ship's maximum army-carrying capacity the player can fill. Without kills, no armies can be transported. Dogfighting is also an important way to stop enemy ships from completing their intended tasks.

Ogging is a suicide attack against an enemy ship (assumably an important ship). Bombing is the act of attacking enemy planets to decrease the number of armies on that planet. Taking planets is the act of carrying armies from one of your planets to an enemy planet in order to conquer it.

When a ship is killed, it restarts at his or her home world, refueled and undamaged but also without kills. In other words, the ship functions as before, but cannot carry armies until it registers new kills.

Automated Players

Traditionally, each ship is controlled by a human player. However, Netrek ships can also be autonomous computer-controlled clients, called robots or bots. Most robots are excellent at low-level tasks such as targeting, navigation and ship maintenance. This makes them excellent at dogfighting.

The robots used in this thesis were originally created by Tedd Hadley. These robots have excellent targeting/dogfighting capabilities as well as the ability to per-

form a number of explicit mini-tasks. Examples of mini-tasks include "ship A ogg ship B" and "ship C bomb planet D." These tasks are based on behaviors well-defined within the Netrek environment and were programmed by Tedd Hadley using his heuristic knowledge of how these tasks are performed. While the robots can choose to perform these mini-tasks, their action-selection mechanism lacks the ability to identify strategically important tasks. As a result, robots playing against each other rarely make any sort of strategic advances, resulting in a relative stalemate.

Bots can also be directed to perform specific tasks. This coordinated strategic influence proves to be critical in turning a groups of bots into a successful team. Strategic direction easily turns the tide of the game in favor of the assisted team.

1.2.3 User Display and Collective Modifications

The standard Netrek display is made of two main windows and several smaller ones. The first large window is the tactical window in which most of the playing occurs, and in which all lower-level control commands are issued. The other large window is the galactic map, which shows the entire galaxy. There are also smaller windows that contain messages from other players and important information about other ships. The final small window allows the player to type messages to other players.

For the purposes of this project we only used the tactical display, which we modified to reflect the dimensions of the large TV used as the user display.

1.2.4 Netrek Advantages

Netrek was chosen as the backend to the multimodal investigations for a number of reasons. First, it has simple rules that are easy to learn, yet a good game involves complex strategies. Good play requires coordination between skilled team members. Also, with the use of robots, we remove the human reflex-dependent aspect of the game and leave the user to higher-level strategy concerns, which are more on pace with the type of multimodal expressions we would like to support. Second, since the standard Netrek interface requires human players to directly control his

ship and communicate through typewritten messages, a strict codification of Netrek tasks (like plays in sports) has developed to facilitate quick communication. This provides an excellent initial set of actions to base our interface on. Third, Netrek is a well-developed, well-tested, open-source Internet game. Rather than worrying about developing and testing our own game, we were able to use an existing program and focus on the objectives of this thesis. Fourth, due to its age, running Netrek is computationally inexpensive. Netrek doesn't sap resources from the more-demanding vision and speech computations. Finally, we wanted an activity that users would find interesting. Netrek maintains a strong online following in spite of its simple bitmap graphics.

1.3 System Overviews

This section presents an overview of our two approaches to speech and gesture integration.

1.3.1 Ogg That There

The *Ogg That There* system takes a high-level, symbolic approach to integrating gesture and speech. To use this interface, the user sits at a desk facing a large-screen TV and wears a noise-cancelling headset microphone. The TV in front of the user displays the Netrek tactical map. Displayed on the screen is a simplified Netrek universe of five planets. In the universe there are also six robot-controlled ships – three Romulan, three Federation. The user can direct the Federation robots to enact simple tasks. The user issues directions to the robots using simple subject-verb-object commands and pointing gestures. The general flow of information through the system is shown in figure 1-2; the components of the system are described below.

Speech Information The speech system used in this interface takes in unprocessed speech and, after a series of processing steps, returns the symbolic representation of a pretrained word, of which there is a small group. These symbols are then taken as

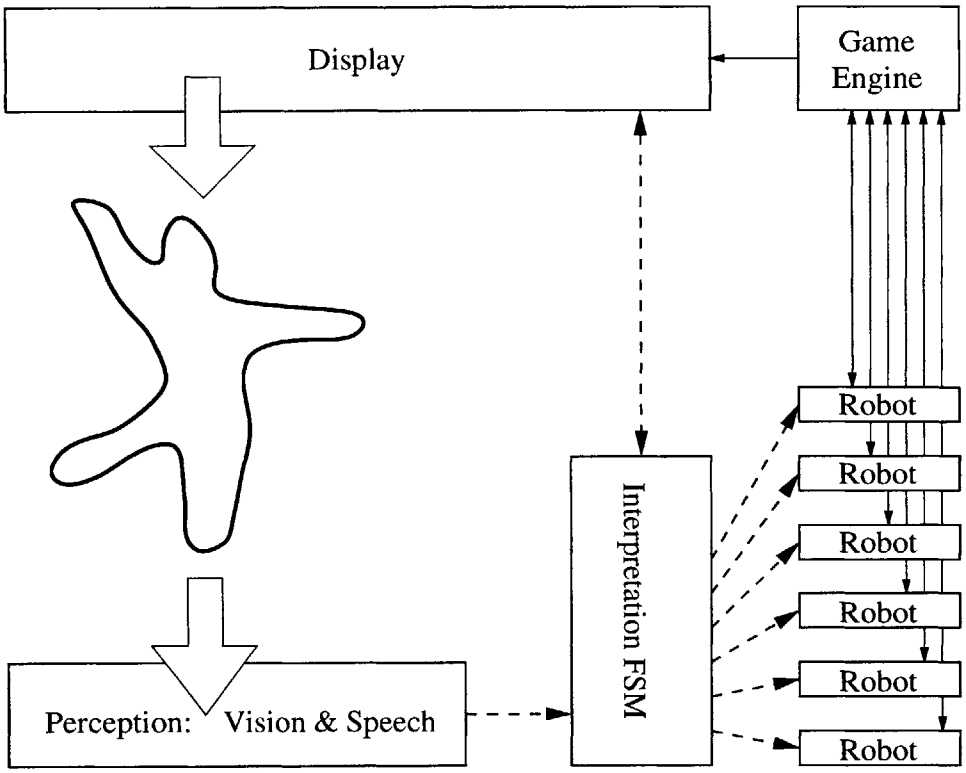

Figure 1-2: *Ogg That There* System Diagram. Arrows indicate information flow. Thin, solid lines indicate standard socket-based communications. Thick dashed lines indicate RPC based communication between our modules. Used with permission from [26]

input by the interpretation FSM.

Vision/Gesture Information Mounted on top of the TV are two cameras used to track the user's motion using the STIVE vision system. STIVE is a blob-based stero-vision system which tracks the user's hands and head at approximately 20 frames a second. At every update, the location of the user's active hand is put through a perspective dewarping to correlate the coordinates of the user's hand in real space with the corresponding screen-space coordinates. These screen-space coordinates are passed to the interpretation module, which in turn passes them onto the display to update the location of the cursor, and highlight the object closest to the cursor.

This system also is used to reference deictic indications (such as the demonstrative

pronouns this and that) to the corresponding deictic gestures. If the speech input indicates a deictic gesture, then the object highlighted by the display is used within the context of the speech command.

Display The state of the game is displayed on the rear-projection screen, faced by the user. The graphics are generated by standard Netrek client modified to allow remote access to the standard display parameters as well as the parameters related to the state of the game. If this ability had not been implemented, it would have been necessary to alter the Netrek server (Game Engine) to provide that sort of information.

Interpretation FSM The interpretation FSM is driven by input from the speech module. With each command element the user states, the interface steps through a simple FSM. The FSM enforces a subject-verb-object grammar for the user's input. With each word stated by the user, the FSM steps forward and waits for another command element that is consistent with the current state of the machine. All commands must begin with a subject (one of the Federation robots), continue with a verb, and end with an object or with a deictic and a correct pointing gesture. After a command is completed, the FMS directs the appropriate robot, to carry out the indicated command. The subject-verb-object loop continues as long as the user continues issuing commands.

1.3.2 Low-Level Data Integration

In this approach we investigated combining speech and gesture at a very low level. To study this approach we did a wizard of oz experiment through which we collected data of people interacting with a simulated Netrek system. Before interacting with the system, the set of possible robot actions was explained to the user, who was given a specific set of words he could use to express those commands. During the course of the experiment, an operator could hear the commands spoken by the user and could watch a copy of the user's tactical display with the user's gesture-controlled

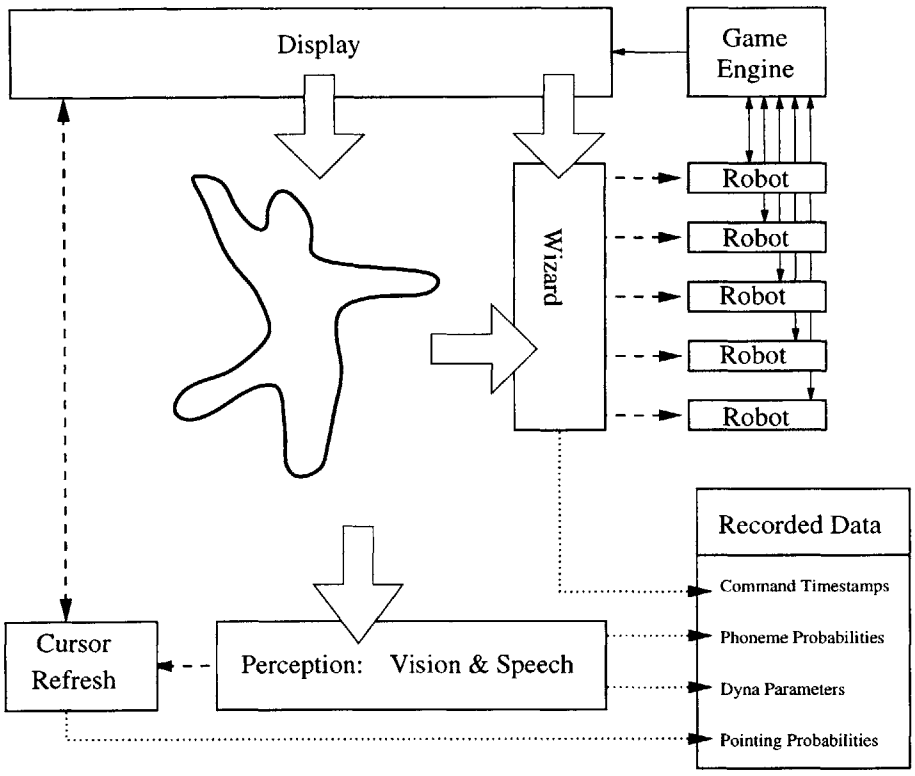

Figure 1-3: System Diagram for Data Collection. Arrow indicate information flow. Thin, solid lines indicate standard socket-based communications. Thick dashed lines indicate RPC based communication between our modules. Thick, dotted lines indicate recorded data.

cursor. Each time the user issued a command, the wizard passed the command along to the appropriate robot, simultaneously tagging the corresponding data with a label reflecting the given command.

The general flow of information collection is shown in figure 1-3. The different components used for data collection, as well as the systems used to process the data are described below.

Speech Information The method used to collect speech data is just a subsection of the processing used to produced symbolic representations of words for the *Ogg That There* system. For this investigation, the processing was stopped when the speech wave from had been processed to the level of phoneme probabilities. These results were timestamped and recorded for later off-line processing.

Gesture Information This system also tracked the location of the user's hands and head in real-time. Body tracking, however, was done using a more robust system, DYNA. DYNA is designed to tolerate temporary, full occlusions of body parts and the presence of multiple people. It is implemented using a recursive estimation framework, which uses predictions based on human dynamics and behavior to improve low-level vision processing. At each update from the vision system, the parameters calculated by DYNA, such as (x,y,z) hand position and velocity, are recorded with a timestamp.

This system also supports the idea of deictic gestures (because people always point) but in a less rigid manner than utilized in the *Ogg That There* system. At each cursor update, the probability that an object is being pointed to is calculated using a metric based on the distance of the object from the cursor. This is implemented by a small module, with the sole purpose of requesting vision system updates, doing a perspective dewarping on the active hand, passing the determined screen coordinates to the display, and then recording the calculated probabilities with a timestamp for later processing.

Data Processing After the data has been collected, it is resampled to a constant rate, 40Hz. After resampling, we ran the data through COGNO, a system for automatic clustering and primitive alphabet discovery. First, the DYNA parameters were processed alone, to find an alphabet of gesture primitives which were then used to map the raw DYNA parameters to a time-varying vector of the likelihoods for each item in the gesture alphabet. These likelihoods were then combined with the other recorded data for a second round of processing, this time looking for an alphabet of speech-gesture primitives.

1.4 Document Structure

Chapter 2 describes other work done in the area of multimodal interfaces, including statistical analysis of user's multimodal activities and an evaluation of other multimodal interfaces. Chapters 3 and 4 describe the implementation and performance

of the *Ogg That There* system and our low-level integration approach, respectively. Chapter 5 discusses potential improvements to our system as well as additions that could be made to the system to expand the user's freedom of expression in our multimodal environment.

Chapter 2

Previous Work

Much in-depth study has been made of computational interpretation of speech and of gesture, each as a stand-alone subject.[11, 12, 2, 1, 25, 14] The fruits of this research include a fairly robust set of independent speech and gesture recognizers. However, integrating these results into the next generation of multimodal systems is a remarkably young field of research. While little research exists on combining speech and gesture at a low level (as we did in this thesis), there is a body of work describing research where speech and gesture were combined at the symbolic level. People are also gathering data on simulated multimodal systems to study the exact nature of multimodal interaction. Described below are some examples of each type of research.

2.1 Work with Symbolic Integration

Although little work has been done on low-level integration of speech and gesture, quite a few systems use high-level symbolic integration of speech and gesture. Below are two examples of the types and functionality of multimodal systems implemented with symbolic integration.

Vo and Waibel [24] describes the implementation of a multimodal interface that allows users to manipulate text using a combination of speech and pen-based gestures. This interface integrates speech and gesture at a highly symbolic level, after

ambiguities in word and gesture symbol recognition have been resolved.

The interface uses eight predefined gestures and eleven pretrained words for communication. There are gestures for the ideas of select, paste, begin selection, end selection, transpose, and split line, as well as two gestures for delete. The eleven key words used are delete, move, transpose, paste, split, character, word, line, sentence, paragraph and selection. The gestures are recognized using a time delay neural network, with the activation level determining the gesture. The words are recognized using a word spotter. Speech and gesture inputs are transcribed into symbolic representations and then combined using a frame system.

It is important to note that speech and gesture data are integrated at the symbolic level, after any ambiguities in in recognition have been resolved.

Commands are recognized by filing the recognized speech and gesture symbols into slots representing various components of a plausible semantic interpretation. For instance, the user could circle a word (completing a selection gesture) and say, "please delete this word." The system would recognize "delete word" in the speech stream, and the system would recognize the circled word in the gesture stream. The two inputs are combined to identify which word to delete.

This system doesn't deal well, however, with conflicts between data streams, nor does it use the other data stream to help resolve ambiguities. For example, if the user is sloppy when circling a word, the gesture system may not know if he is circling a single word or the entire paragraph. Knowledge of whether the user said "word" or whether the user said "paragraph" would be very helpful in resolving this dilemma.

Koons and Sparrell [15] investigate the integration of speech and depictive gestures. Depictive gestures are those gestures that are closely associated with the content of speech and that complement the user's verbal descriptions. Depictive gestures can describe shape, spatial relations and movements of objects.

Koons and Sparrell implemented a system called Iconic, which allows users to describe the layout of virtual three-dimensional scenes through a free mixture of speech and depictive gestures. First, the speech data is recognized by a system that

provides a stream of recognized words and their associated time-stamps. The words are then parsed to produce a "best expression."

A pair of DataGloves are used to record the user's hand position, thus collecting the gesture data. The gesture data is tagged with pre-defined feature-level descriptions for the hand shape, orientation, movement and location. The tagged data records are then analyzed and grouped into gesture segments and stored for further processing. A high-level interpretation module then takes the symbolized speech and gesture, along with a description of the location and orientation of the objects in the virtual environment and processes all the data as a group to interpret the user's commands.

This can be best explained through the example given in the paper: The user says, "Move the chair like this," while demonstrating the desired action with a motion. The interpretation module recognizes that the user is describing a movement for "the chair." The system does a mapping between the user's hand motion and the motion of the chair, then moves the chair in the same manner.

Iconic is an interesting system that doesn't require the user to learn a predefined set of gestures. Still, gesture doesn't influence the interpretation of speech. Furthermore, gesture is used only as an addition to speech: to add meaning in the context of the spoken command.

2.2 Statistical Analysis

The work described below endeavors to expand our knowledge of the natural integration patterns that typify people's interactions with multimodal systems. This is achieved by observing people interacting multimodally, rather than by building an interface and seeing how it works. This work provides a foundation for building better multimodal interfaces in the future based on the natural actions of people. Currently, most multimodal systems are designed using the intuition of the designer on how things *should* work, rather than on how people *actually* act.

Oviatt, DeAngeli and Kuhn [22] studied when users are most likely to compose their input multimodally, rather than unimodally. They also analyzed the main linguistic features of multimodal constructions, as well as differences from standard unimodal ones.

To analyze multimodal interaction, they observed people communicating with a simulated dynamic map system via speech and writing. Users interacted with the simulated system using speech and 2-D electronic pen input. The user's goal was to use the dynamic map system to search for a piece of real estate that satisfied a set of given constraints. With the system the user could perform such actions as circling a house on the map and asking, "Is this house in a flood zone?" – or circling and connecting two entities and asking, "How far from here to here?" In each case the system would respond textually and change the display to reflect the desired question.

According to their findings, multimodal interactions occurred most frequently during spatial location commands (e.g., "move this here," or "add this there"). Selection commands had an intermediate likelihood of being expressed multimodally. (Selection commands include actions like "label this house good" and "delete this.") In terms of linguistic features, they found that 97 percent of multimodal construction conformed to subject-verb-object grammar. Multimodal constructions also tended to be expressed in a telegraphic command language. In other words, a multimodal command might consist of the user drawing a line and instructing the system to "add dock here." An example of a unimodal command (speech alone) could be, "Add a boat dock on the west end of Reward Lake."

This research suggests that spatial domains may be ideal for developing multimodal systems. Netrek is one such spatial domain – the user must relate the location of things for a successful interaction with the game.

Mingnot, Valot, Carbonell and Robbe [10, 18] investigate differences between spontaneous speech and a tractable artificial command language composed of utterances from a restricted subset of natural language in a simulated multimodal environment. In the experiments, users communicated to the system with speech and

2-D gestures inputted through a touch screen. Subjects were asked to perform tasks involving furniture arrangement in an apartment. The first set of subjects was allowed to use spontaneous speech (say anything they wanted) to communicate with the interface. The second group of subjects used an artificial command language to communicate with the interface. Its power of expression was equivalent to the union of the semantic interpretations of all utterances used by the first group of subjects, but synonymy and ambiguity were excluded. The two groups' abilities to successfully manipulate furniture within the program were then compared to determine how the restricted command language affected the second group's ability to express themselves.

In the end, they found that linguistic constraints will be assimilated easily by users in a multimodal environment, provided that the resulting oral component of the multimodal command language is a subset of natural language that does not impose semantic restrictions on the user's expression.

This research is closely related to the systems developed in this thesis. In *Ogg That There* we let the user define his own artificial command language. In our second approach, we set an artificial command language for the user. Our results were based, in part, on a comparison of the success of these two approaches.

Bourguet and Ando [6] investigate temporal synchronization between speech and hand-pointing gestures. For their experiment, subjects completed three types of tasks

1. A "question/answer" task, where the subjects were asked questions about pictures displayed by the system, and the subject was to answer using unconstrained speech and simultaneous pointing gestures.

2. A "navigation" task, where the subjects were given a map of the Tokyo subway system and asked to explain routes to different locations.

3. A "command" task, where the subjects were asked to move graphical objects to match a given pattern. In this case, the actual editing was done by someone

monitoring the display. Users were informed that it was a simulation but asked to act as natural as possible.

In each experiment a DataGlove was used to recognize pointing gestures during the course of the data collection. The whole experiment was videotaped. The speech was then transcribed from the videotape, and the speech wave was used to timestamp each word. The words were classified into five types of expressions (nominal, deictic, verb, adverb and other). Each gesture was then associated with the expression that occurred closest to it chronologically. 96.78 percent of all gestures are shown to be synchronized with either a nominal or deictic expression. Bourguet and Ando suggest that "the choice between several hypotheses could be greatly simplified with the use of a model that indicates, in a phrase, the position of nouns and deictics."

This project relates on Bourguet and Ando's research through our alternate approaches to the co-occurrence of speech and gesture. In *Ogg That There* we depend on the co-occurrence of "that" (the deictic speech) and pointing to identify the gesture. In the second approach we are co-processing speech and gesture. We thus test Bourguet and Ando's claim that integrating speech and gesture will assist in the speech recognition process.

Chapter 3

Preliminary System: Ogg That There

This first version of an interface for Netrek is patterned after Bolt's "Put That There" system[5]. It was developed to test the feasibility of Netrek as our experimental test-bed and to do a first pass determination of the interface requirements for our modified Netrek application.

To use this interface, the user sits at a desk facing a large-screen TV and wears a noise-cancelling headset microphone. Mounted on top of the TV are two cameras used to track the user's motion using the STIVE vision system (see section 3.2.1). The TV in front of the user displays the Netrek tactical map that has been modified to fill the entire screen. (see figure 3-1) The user commands the robots using a simple Subject-Verb-Object grammar. Players can either name objects explicitly, or they can use deictic gestures and demonstrative pronouns (i.e., saying "that" while pointing to the desired item) to pinpoint the object.

For the *Ogg That There* system, the Netrek universe is limited to five planets. All five planets can be seen in the tactical display. Two planets, Romulus and Aldebaran, are Romulan. Two planets, Earth and Alpha Centuri, are Federation. The last planet, Indi, is independent. (see figure 3-2) The Federation and the Romulan teams are each made of three robot players. The robots are able to act on their own, but the human user can step in to direct the Federation robots to complete specific tasks.

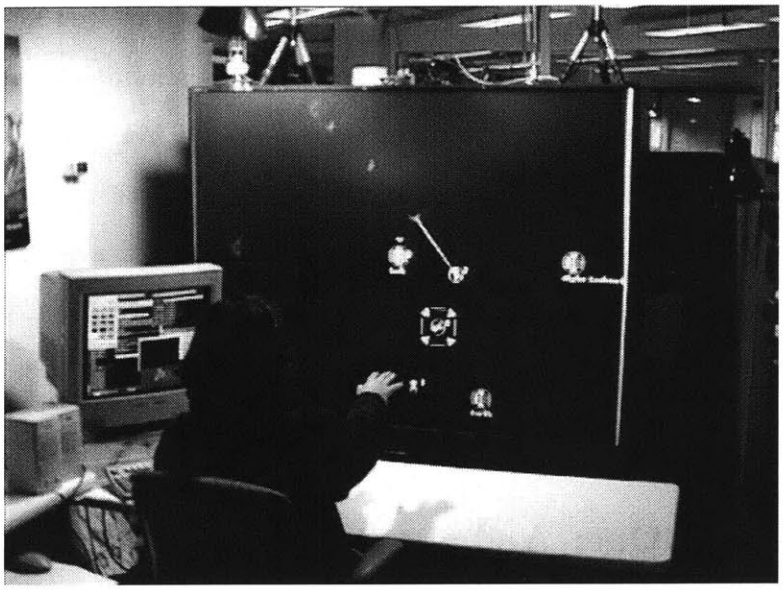

Figure 3-1: *Ogg That There* user pointing at a Federation ship

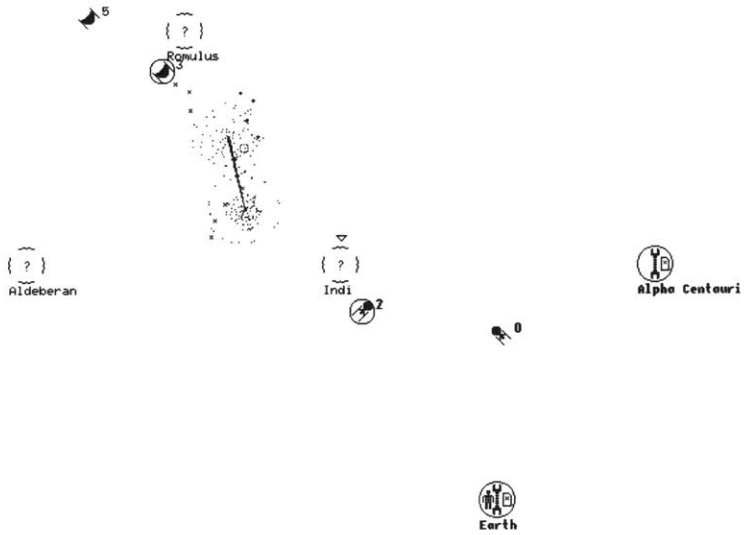

Figure 3-2: Netrek tactical map displayed to the user

3.1 Speech Information

The speech system used in this thesis was developed by Deb Roy at the MIT Media Lab as part of the CELL system[20]. The speech unit is made of two modules. The first processes the raw speech, calculates a time-varying English phoneme probability vector, and then uses the probability vector to output a most-likely phoneme string.

The second part of the system has a training mode and a testing or use mode. During the training mode, the module gathers phoneme string-label pairs. During the testing mode, the module accepts a novel input phoneme string and returns the most-likely label for that string.

3.1.1 Module One: Audio

The fundamental input to this system is an acoustic signal originating from a noise-cancelling microphone. This audio signal is sampled at 16kHz with 16-bit resolution and is converted to the RASTA-PLP spectral representation for speech[13]. This implementation uses a 20ms input window and a window step size of 10ms, resulting in a set of 12 all-pole RASTA-PLP coefficients estimated every 10ms.

The RASTA-PLP coefficients are then used as inputs to a recurrent neural network (RNN)[19]. The RNN produces a vector of 39 likelihoods: the likelihoods of the 38 English phonemes (Table 3.1) and the likelihood of silence. Since the output likelihoods are all positive and guaranteed to sum to one, they can be treated as probabilities. pipeline to this point. To train the RNN, the TIMIT database of phonetically transcribed American speech was used[21].

Finally a sequence of RNN output probabilities are run through the Viterbi algorithm to estimate the most-likely phoneme sequence. This phoneme sequence is the input for the second module.

3.1.2 Module Two: Recognizer

The second module has a training phase and a testing or use phase. During the training phase, the user indicates a label to which the successive audio should correspond.

Phoneme	Example	Phoneme	Example	Phoneme	Example
aa	caught	f	fun	q	bat (glottal stop)
ae	bat	g	go	r	ray
ah	but	hh	hay	s	say
aw	about	ih	bit	sh	shoe
ay	bite	iy	beet	sil	(silence)
b	bat	jh	joke	t	to
ch	chat	k	cat	th	thin
d	dog	l	lay	uh	book
dh	then	m	may	uw	boot
dx	dirty	n	no	v	vision
eh	bet	ow	boat	w	way
er	bird	oy	boy	y	yacht
ey	bait	p	pay	z	zoo

Table 3.1: English language phonemes encoded in the RNN. Figure used with permission from [20].

The audio is processed by the first module and the most-likely phoneme sequence is passed on to the second module.

The most-likely phoneme sequence is used to generate an hidden Markov model (HMM) for the sequence (figure 3-3). A state is assigned for each phoneme in the sequence, and the states are connected in a strict left-to-right configuration. The state transition probabilities are assigned from a context-independent set of phoneme models trained from the TIMIT training set. The user continues to create label/HMM pairs. If there are two HMMs for a label, one is chosen at random to be the central example. If there are three or more HMMs for a label, a distance metric is used to select the most central example to represent the class.

During the testing or use phase, the phoneme sequence produced by the first module is used as input for the HMMs created during the training phase. The phoneme sequence is run through the central HMM for each label. The system recognizes the phoneme sequence as an example of the label with the highest response probability. This system assumes that all utterances will correspond to a label.

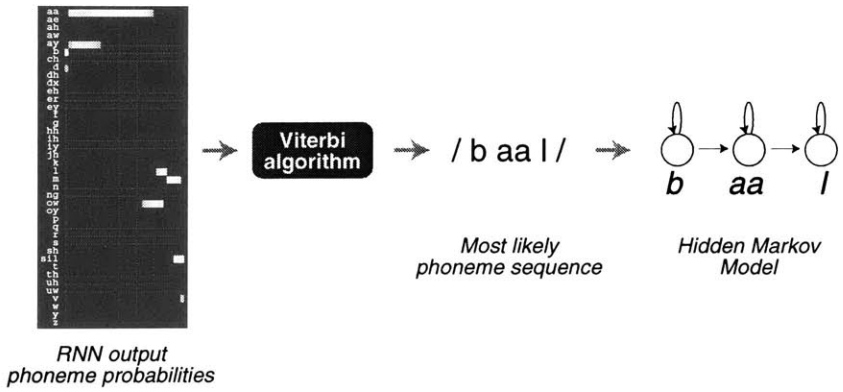

Figure 3-3: The Viterbi algorithm finds the most likely phoneme sequence for a sequence of RNN output. A left-to-right HMM is constructed by assigning a state for each phoneme in the sequence. Figure used with permission from [20].

3.2 Gesture Information

During the use of the interface, the location of the hands and head of the user are continuously tracked by the STIVE vision system (see section 3.2.1). At every update, the location of the user's active hand is put through a perspective dewarping (see section 3.2.2) to correlate the coordinates of the user's hand in real space with the corresponding screen space coordinates. The screen space coordinates are passed to the display in order to update the location of the cursor, and highlight the object closest to the input screen coordinates.

This system also is used to reference deictic indications (such as the demonstrative pronouns this and that) to the corresponding deictic gestures. If the speech input indicates a deictic gesture, then the object highlighted by the display is used within the context of the speech command.

3.2.1 STIVE

STIVE is a vision system designed to track the head and hands of a person sitting at a desk (2'x5'). (see figure 3-4). STIVE uses a wide baseline stereo vision system and statistical models of human skin color to track a person's hands in 3D in real time. This is done in two steps. First, the two-dimensional color blobs are identified

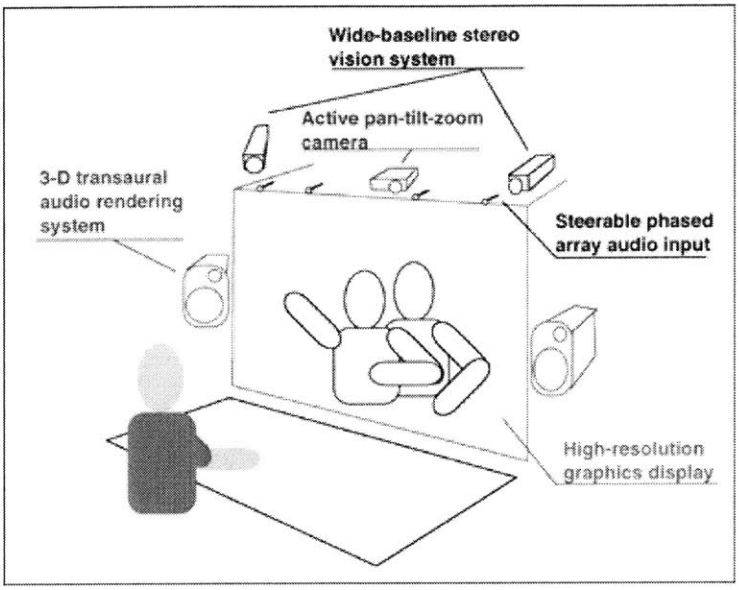

Figure 3-4: Desk set-up for STIVE

for each of the two input-cameras' images. STIVE then does a blob correspondence between the two images and uses information about the fixed position of the cameras to determine the location of the blobs (head and hands) in three dimensions. This allows the system to track simple hand gestures such as pointing.

3.2.2 Deictic Interpretation

While STIVE is useful in determining the location of the user's head and hands in 3-D relative to some scale fixed to the geometry of the STIVE setup (i.e., height off the desk), it is more difficult to tell exactly where the user believes he or she is pointing at any given time. One option is to predefine the correspondence between hand position and screen position, then expect the user to adapt to the system and ignore where he thinks he is pointing. However, this can be awkward for the user, given the high variance in pointing styles and heights. Height of the user drastically changes the user's line of sight, and thus where the user believes he is pointing, given a real-space hand position. To deal with these differences, this project determines screen coordinates by interpolation over a set of pointing examples gathered off-line.

These calibrations are examples of the user pointing to the four corners of the screen with each of his hands. Separate calibrations are maintained for each hand.

As Chris Wren explains in his thesis: "In general these four points will not form parallelograms in feature space, so linear strategies introduce unacceptable warping of the output space. *Ogg That There* employs a perspective warping to translate input features ((x, y) hand position in 3D space) to screen position:

$$\begin{bmatrix} XW \\ YW \\ W \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.1}$$

where X and Y are screen position and x and y are hand position. The parameters a, b, c, d, e, f, g and h are estimated from data. Interpolation of a novel deictic simply involves plugging the new (x, y) into equation 3.1."[26]

3.3 User Display

The state of the game is displayed on the rear-projection screen, faced by the user. The graphics for the screen are generated by the display module. The display used for this thesis is a standard Netrek client with a few modifications. First, the size of the tactical screen was altered to fit the size of the rear-projection screen. A remote procedure call (RPC) interface was also added to allow remote access to the standard display parameters.

Additional features, added to the standard client and accessed via RPC, give the user a better idea of his or her interaction with the game. One such feature is the ability to display a cursor in the tactical screen, providing the user gestural feedback. The (central module, name needed) can also issue commands to the display to highlight the object closest to the cursor (a ship or planet), so that the user can see what he has currently selected (i.e., what he is pointing to). The last feature gives the interpretation module the ability to query the display for information about the location of the different game objects, including searching for the closest object to

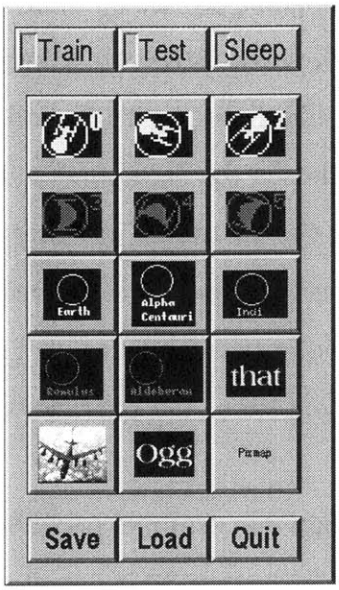

Figure 3-5: Button panel used to train the names of each of the Netrek objects

a given screen location. This feature allows the display to be used as a game state database. If this ability had not been implemented, it would have been necessary to alter the Netrek server to provide that sort of information.

3.4 Total System

To use the *Ogg That There* system, the user first trains a unique word or short phrase to correspond to each of the ships, planets and actions that exist in the context of the game. The user also trains a demonstrative pronoun, usually "that." These words are trained using a simple click-and-say interface.

The user is presented with a button panel (see figure 3-5), which consists of one button for each ship, planet and verb, and one button for the chosen demonstrative pronoun. Each button is marked with an image of the object or verb that it represents.

The user can give any single name he or she chooses to the object. For example, the planet Earth can be referred to as "Earth," "home" or "my planet," as the user chooses. Please note, however, that the user can only choose one name for each object, and he must be consistent with that name.

To train the system to recognize the user's names, the user must press the corresponding button on the panel, which is then highlighted and indicates to the system that the user's name for the highlighted button will follow. The user then says the word or short phrase he has chosen to correspond to the ship, planet, verb or demonstrative pronoun indicated by the button he just pressed. The audio is then processed by the speech system, and the phoneme string is used to create a HMM/label pair. Where a label is just a symbolic marking used to group all examples matching the same button. In this implementation, the buttons were numbered and that number was used as the label.

If the user creates more than one example per label, then a central example of the group is chosen, as described in section 3.1.2. As stated above, each user can choose whichever word or phrase he likes to name each of the ships, planets, verbs and the demonstrative pronoun as long as he is consistent.

After the words are trained, the robots are started. The robots take action without user input and begin flying around the screen. When the user decides to participate, he controls the robots with a simple subject-verb-object grammar. The subjects (e.g., the Federation ships) are F0, F1 and F2. The verbs are "bomb" and "ogg," and the objects are R3, R4 and R5 (the enemy's ships), and all of the planets, Romulus, Earth, Indi, Alpha Centauri and Aldeberan. For example, the user can command "F0 ogg R3" or "F1 bomb Aldeberan." "That" (or the user's chosen word or phrase for the idea of "that") can be used in the object slot of the grammar. If "that" is used, the referent of the corresponding deictic gesture is identified, as described in section 3.2.2. This allows commands like "F2 bomb that"(pointing at Romulus) – which then prompts the robot for F2 to bomb Romulus.

With each command element the user states, the interface steps through a simple finite state machine (FSM). (see figure 3-6) With each command element, the FSM steps forward and waits for another command element that is consistent with the current state of the machine. An example: First, the user must start the command with a word or phrase that corresponds to a subject (F0, F1 or F2). If the user says anything else, the FSM ignores it and waits for a subject. Lets say the user indicates

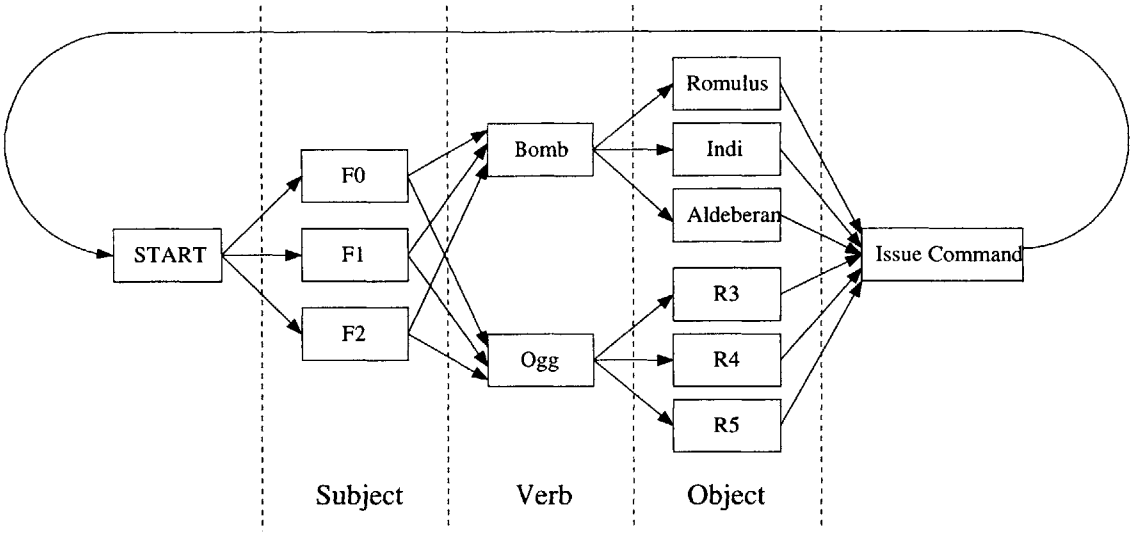

Figure 3-6: Finite State Machine for command recognition

F1. The FSM remembers that and is now waiting for a verb. Then the user indicates a bomb, so now the FSM waits for an object that makes sense in terms of the action of bombing (i.e., an enemy planet). If the user says "that" and accidently points at R3 (which might be flying really close to Romulus), the FSM ignores it – because Netrek only allows the bombing of planets. When the last part of the command is issued in an appropriate fashion, the FMS directs the appropriate robot – in this case, F1 – to carry out the indicated command – in this case, bombing Romulus.

3.5 System Performance

While the *Ogg That There* system is functional, it suffers from a number of shortcomings that prevent it from being a natural, easy-to-use, well-implemented interface. These difficulties revolve around two issues: the fact that the pace and rigidity of the interface is not well-suited to Netrek, which results in errors and frustration, and the fact that the vision system is underutilized, which results in a non-optimal system.

3.5.1 Poorly Suited Pace and Rigidity

The internal FSM enforces a very rigid communication pattern: subject, verb, object, issue command. If a word is omitted or inappropriate for the phrase, the interface waits until it identifies an appropriate word and continues. *Ogg That There* eexperiences a recognition failure if a word is omitted, or if a demonstrative pronoun is used and the object pointed to is inappropriate for the command.

If the user uses a demonstrative pronoun, and the referent is inappropriate, the interface waits for an appropriate input. Pointing to the appropriate input, however, can be a challenge because all game objects of interest tend to clump together at the site of the action. Selecting one – especially the desired object – can be difficult.

The rigid grammar and lack of error recovery make *Ogg That There* a system that has difficulty keeping up with the pace of our altered Netrek game. The slowness of the interface is further exasperated by the reduction of the universe size. In a standard Netrek game, a ship regenerates a fair distance from the action. This allows for more time to issue a command as the ship travels back into the thick of it. In our reduced universe, however, ships regenerate just a few pixels away – ready to zoom back into the action. Thus, the life cycle of a ship can be less that 10 seconds. This allows little time to strategize and issue a command. Additionally, the results of your command are rarely obvious, as the ships are always moving and have such short lifespans. This lack of feedback is non-optimal for interface testing.

To further complicate things, the speed of the game and the speed of the interface are not compatible. The ships are always darting from one spot to another. A ship that is highlighted one instant may not be highlighted the next, even when the user keeps his hand in the same position. The object closest to the cursor is not the only potentially important object; all the objects relatively close to the cursor are potential referents.

In the end, the player is frustrated. The imprecision of the vision system is emphasized, although it would not have been obvious – or perhaps noticed at all – if these problems did not exist. The precision, set at 2 degrees, is a minimal amount

of error, and completely acceptable in most systems. Two degrees does not work, however, when the user attempts to select between things in close proximity to each other.

3.5.2 Underutilization of the Vision System

The STIVE system is best suited for identifying qualitative movements, rather than resolving pointing gestures with extreme precision. At the same time, the longterm goal of this project is more closely related to STIVE's current state, rather than Netrek's needs in the *Ogg That There* system. The eventual aim is to build an interface that communicates higher-level information. The next interface should therefore move away from the use of precise deictic gesture and pinpoint selection. There are input devices (such as a mouse) that are better suited to pinpoint selection than a blob-based vision system. The extra capabilites of a body-tracking system can be better utilized in a way that goes beyond just imitating a mouse.

This interface also severely underutilizes the information coming from the vision system. First, gestures are examined only after they are indicated by the speech input. A better strategy would be to use speech and gesture infomation together to determine the most likely thing the user is indicating. Looking to more than one system for primary information would also make the system more robust. The system could then recover (or at least make a good guess) when one of the two systems fails. It also could use one data stream to distinguish between ambiguities in another.

In the case of *Ogg That There* gesture information could be used to differentiate between objects with very similar-sounding names. For instance, if the user says "F ?" and is pointing at F1 while the rest of the Federation ships are on the other side of the screen, it's fairly likely that the user is talking about F1. However, the current *Ogg That There* system has no mechanism in place to make that distinction. It would make a close call guess when a concrete answer could be obtained.

3.5.3 Addressing System Preformance

In this project, we attempted to address some of these issues with a second approach to speech and gesture integration. This system uses a slightly different Netrek interaction (including a slower pace and fewer ships) and moves away from the *Ogg That There* system's approach to gestures (by doing such things as integrating speech and gesture feedback, as well as moving away from deictic gestures). The new approach is further discussed in Chapter 4.

Chapter 4

A Low-Level Approach

This chapter describes the investigation of a different approach to identifying purposeful human action in speech and gesture data. Our goal was to take advantage of correlations between speech and gesture data streams. We chose to integrate the streams at a low level – before the representations became too symbolic. High-level representations encode human assumptions about speech and gesture as separate systems. We believe that integrating speech and gesture streams in symbolic form would risk missing important correlations among the data.

The co-processing of speech and gesture also addresses the underutilization of gesture information. Through this approach, correlations and redundancies in the gesture data stream could be used to recover from recognition failures in the speech data stream, and vice versa.

4.1 Wizard of Oz Setup

To test our approach we did a wizard of oz experiment through which we collected data of people interacting with a simulated Netrek system. The data was collected using the same desk and display system as well as the same data input devices that were used for the *Ogg That There* system. For the experiment, the user played the modified Netrek game, but was given a specific goal, a list of each ship's possible actions, and a limited vocabulary.

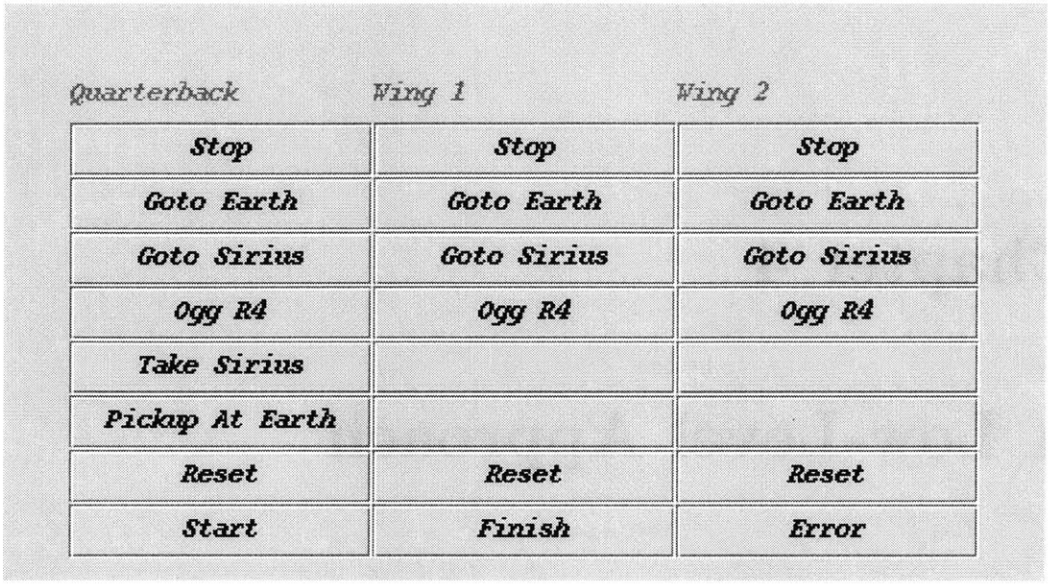

Figure 4-1: Button panel used by the Wizard to control the robots. Each button reflects a command the user is able to give the robots. The first row of command can be given to F0 (Quarterback), the middle row to F1 (Wing 1) and the last row to F2 (Wing 2).

An operator or "wizard" could hear the commands spoken by the user and watch the user's tactical display with the user's gesture-controlled cursor. Each time the user issued a command, the wizard entered the command to the robots using a traditional interface consisting of a panel of buttons (see figure 4-1). The wizard's actions simultaneously tagged the corresponding data with a label reflecting the given command. These tags and the collected data were later used for training and testing our system for identifying purposeful human action.

During the course of the experiment, we collected data reflecting the person's speech, information about the position of their hands and head, and information about what they were pointing at/near on the tactical display. The exact nature of the data collected is described in the sections below.

After the experiments were run, the data was processed using COGNO , a system that does an unsupervised clustering of data to train low-level HMMs, which are then used as an alphabet to represent the higher-level commands. This system is described in section 4.4.

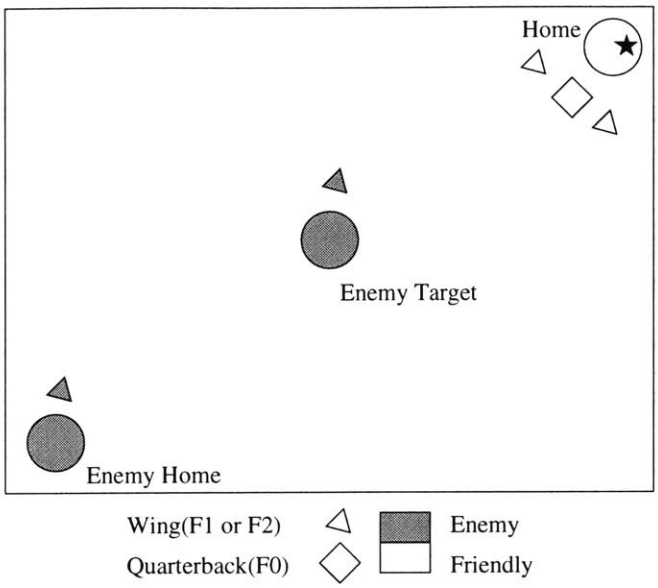

Figure 4-2: Modified Netrek Enviornment. Used with permission from [26].

4.1.1 Modifying the Netrek Environment

For this experiment, we also changed the nature of our modified Netrek game to better suit the pace of our speech and gesture integration. First, we removed the robots' action-selection mechanism. The robots no longer act on their own, but sit still until directed into action or confronted by an enemy ship. The user thus sees a clear action-reaction response to his commands. The user can also indicate specific objects much more easily because the robots aren't always in motion. This changes the pace of the game as well: Nothing happens unless the user issues a command, so the pace is reduced to the rate of command.

Second, we simplified the universe even further. The Netrek universe now contains only two Romulan planets and one Federation planet. (figure 4-2.) The Romulans have only two ships, one to defend each Romulan planet. These ships are purely for self-defense; they only take action to defend their respective planets. They will not attack the Federation ships unless provoked. The user still has control of three Federations ships, but these ships stay at Earth unless otherwise directed.

Third, we clearly defined a goal for the user by seting up a series of tasks for the

user to accomplish. The goal of the game is for the user to direct the Federation ships to take Sirius, the Romulan planet closest to Earth. Part of the user's task is to direct F0, the first Federation ship, to pick up armies on Earth and then take them to Sirius to conquer the planet. R4, however, is guarding Sirius. If the user sends F0 directly to Sirius, F0 will be killed by R4 and thus fail to take Sirius. This is where the other two Federation ships come into play. The user must use F1 and F2 to distract or kill R4 before sending in F0 to take Sirius. This task requires a strategic approach to ship coordination: R4, after being killed, quickly regenerates at the Romulan home planet (Romulus) and then flies back to its post defending Sirius. Thus, the user has a limited window of opportunity to finish his task of taking Sirius.

Finally, a few rules and constraints were modified to further simplify the game while keeping the experience challenging. The Federation ship F0 (the quarterback) can always carry armies, and Earth always has armies. (This allows the user to take an enemy planet more easily, because taking a planet requires carrying armies from a friendly planet to the enemy planet.) Sirius, the targeted Romulan planet, will never contain more armies than can be overthown with one shipment of armies from Earth. Also, each time the user successfully takes Sirius, the game returns to its original configuration, except that the ship defending Sirius, R4, becomes stronger and the task is therefore more difficult.

4.2 Speech Information

The method used to collect speech data is just a subsection of the processing described in section 3.1. The input audio is converted to the RASTA-PLP spectral representation for speech. The RASTA-PLP coefficients are used as input to the same RNN, and the time-varying English phoneme probability vector is calculated. Rather than using this vector to determine a most-likely phoneme string, as done in the *Ogg That There* system, the probability vector is recorded at each time frame, along with its time-stamp. The speech phoneme probabilities are later combined with the gesture data for processing using the COGNO architecture.

To limit the training data to a reasonable size, the user was given an artificial command language for directing the robots. The command language is a subset of natural language and does not restrict the user's semantic expression. In other words, the user's command options are not limited, but the words he could use to express those commands are limited. Robbe has found that linguistic constraints of this type are easily assimilated by the user and in some cases have a beneficial effect on the users' expression, reducing hesitation and grammatical errors[18].

4.3 Gesture Information

This system also tracked the location of the user's hands and head in real-time. Rather than relying upon the STIVE vision system, described in section 3.2.1, this experiment used a more sophisticated system, DYNA. DYNA is based on STIVE, but more flexible and robust.

At every update from the vision system, the location of the user's active hand is put through a perspective dewarping (see section 3.2.2) to correlate the coordinates of the user's hand in real space with the corresponding screen-space coordinates. In this system, the screen-space coordinates are still passed onto the display. Unlike *Ogg That There* , however, the display is set to show only the cursor. The location of the cursor is used to calculate via a distance metric the likelihood that each of the Netrek objects is being pointed to, as described in section 4.3.2. This supports the idea of deictic gestures (because people always point) but in a less rigid manner than utilized in the *Ogg That There* system. No pinpoint selections are required.

4.3.1 Dyna

DYNA is a real-time, fully dynamic, 3-D person-tracking system. It is designed to tolorate temporary, full occlusions of body parts and the presence of multiple people.

DYNA uses a recursive estimation framework, illustrated in figure 4-3. DYNA is driven by 2-D blob features that are used to estimate the position and orientation of 3-D blobs. These features are then integrated into a fully dynamic 3-D skeletal model.

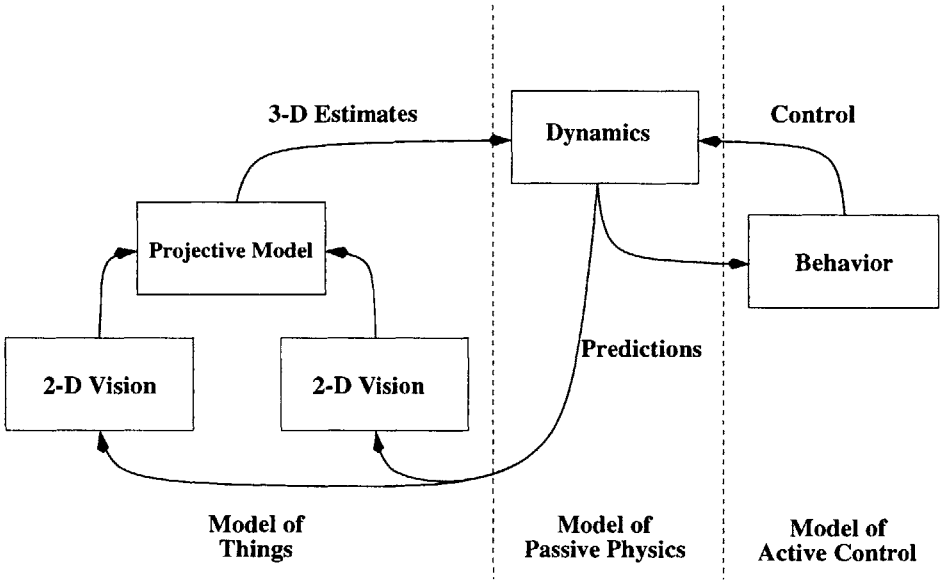

Figure 4-3: DYNA Recursive Estimation Framework. Predictive feedback from the 3-D dynamic model becomes prior knowledge for the 2-D observations process. Used with permission from [26].

The skeletal model is modeled in two parts. The dynamics module models position and velocity of the skeletal model as well as the physical constraints of the human body. Physical constraints include joint flexibility and fixed arm length. The behavior module aims to model the aspects of human motion that can not be explained solely by passive physics (i.e., purposeful human motion). Ideally, the behavior module would be an explict model of the entire human nervous and muscular systems. Because that is an unrealistic endeavor, observations of humans in action are used to estimate probability distributions over a control space.

Together, the dynamics and behavior modules are used to predict human motion. These predictions are then fed back to the 2-D vision system as prior probabilites, which in turn influence the next round of 2-D blob identifications. This provides for better, stronger, faster person-tracking. A more detailed description of the DYNA system is available in [27, 26].

At each update from the vision system, the parameters calculated by DYNA such as (x,y,z) hand position and velocity, are recorded with a timestamp. This information

is later used as input to the COGNO analysis.

4.3.2 Deictic Interpretation

In this system, we tried to avoid precise pinpoint deictic selection gestures. Instead, our goal was to develop a more natural system where the user can indicate the general direction of the desired object. To do this we developed a metric to determine the likelihood an object is being pointed at, given its distance from the cursor. The metric we used is

$$P(object) = e^{-(\frac{x}{y})^z} \tag{4.1}$$

where $P(object)$ is the probability an object is indicated, x is the distance from the object to the cursor, and y is a normalization factor that determines the approximate radius at which the probability dies off. The normalization factor, y, will always be the distance at which $P(object) = e^{-1}(about.368)$. The exponent z controls how quickly the function dies off. A large value of z therefore leads to a particularly steep, quick transition from one to zero.

For this application, y is the Netrek distance equivalent of the width of about two and a half ships, and z is set to 2. We chose z to be a small number because we wanted a gradual decay in the probability distance function, rather than a sharp cutoff. We felt that a gradual probabilistic method was better because tiny differences in distance are insignificant for determining the indicated selection. One of the problems with the *Ogg That There* system was that the tiny difference could completely change the indicated item.

The likelihood of each object is calculated whenever a new piece of information is available from the vision system. The probabilites are recorded with a timestamp and later used as input to the COGNO analysis.

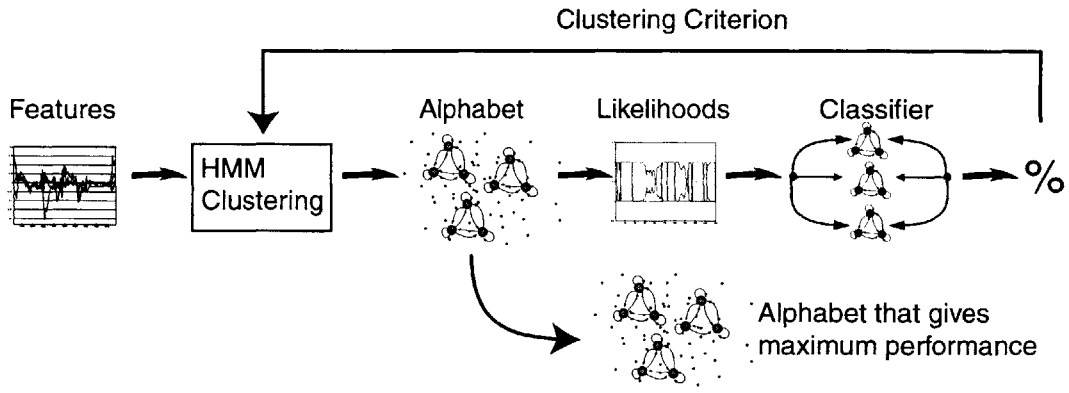

Figure 4-4: COGNO pipeline for alphabet selection.

4.4 Cogno

In this application we had a set of raw features: phoneme probabilities, DYNA output data and deictic indication probabilites. From these features we wanted to determine low-level combined speech and gesture primitives. These primitives were our alphabet, which we then combined to represent the commands of the user. In this application we used HMMs modeled on the input features to produce our alphabet. Traditional HMM training techniques, however, require the data to be segmented into examples before the parameters of the HMM can be estimated. In this application, we did not know the what the alphabet was in advance, so we could not pre-segment the data. Instead we used a different approach of automatic clustering and alphabet discovery provided by the COGNO architecture[9, 8]. An illustration of the COGNO architecture is shown in figure 4-4.

In general, COGNO generates candidate alphabets by clustering the raw features using a variation on the Segmental K-Means algorithm[17], which produces a set of HMMs to represent the alphabet. Clustering is influenced by three free parameters:

1. Number of HMMs (i.e., the number of symbols in the alphabet)
2. Number of states per HMM
3. Time scale (approximate length of a behavior)

These HMMs are then used to map the raw features to a time-varying vector of the

likelihoods for each alphabet symbol modeled by the HMMs. This likelihood vector is then used to build classifiers for the higher-level tasks (in this case, the user's Netrek commands) that are made up of a series of alphabet events. The construction of the classifiers is influenced by one free parameter, the number of states in the classifer. After construction, each classifier's performance is then evaluated. That evaluation is fed back into the HMM clustering to improve alphabet selection. This continues until the process converges on the highest-performing alphabet.

4.5 Data Processing

The data examined in this experiment were collected from two subjects. Each subject recorded three sessions, each approximately ten minutes in length. This served as the training and testing data for our investigations.

4.5.1 Data Alignment and Resampling

The speech phoneme probabilites, DYNA parameters and pointing probabilites were originally sampled as fast as the underlying processes could produce the data. This produced irregular time deltas between data points. Furthermore, the original speech sampling rate was approximately 100Hz, while both the sampling rate of DYNA parameters and the sampling rate of pointing probabilites (which is based on the DYNA rate) were about 30Hz. To combine the data, everything was resampled at 40Hz starting from a common basetime. New data points were determined by doing a simple linear interpolation between the surrounding two points.

4.5.2 Cogno Processing

After resampling, we ran the data through the COGNO architecture in various combinations. For each data set, we searched for successful alphabets using wide cariations of COGNOś free parameter values. The free parameters are: number of alphabet HMMs, number of states per alphabet HMM, timescale for the alphabet, and number

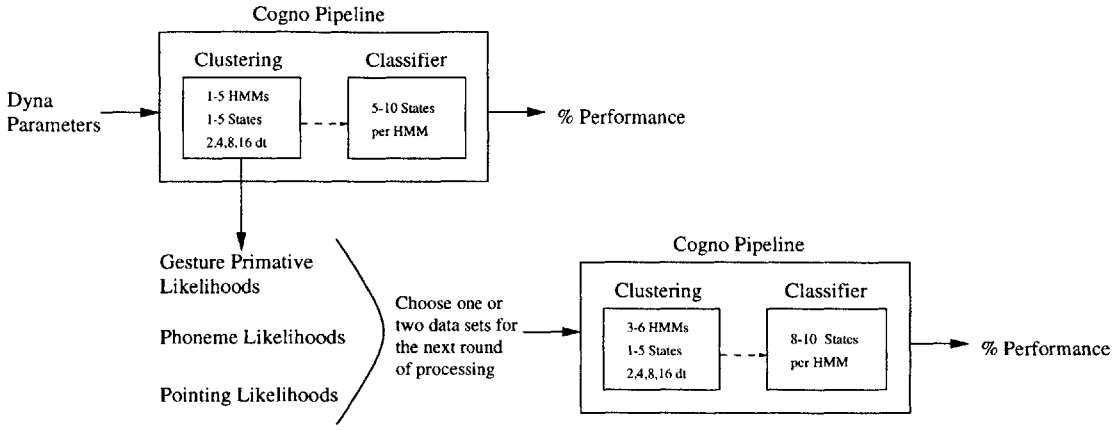

Figure 4-5: An Overview of the Data Processing Pipeline. First the DYNA parameters are processed to discover and alphabet of gesture primatives and their respective likelihood maps. The probabilities of the gesture primitive alphabet were then combined with the other sets for further COGNO processing.

of states in the high-level classifier HMMs.

First, we ran only the DYNA parameters through COGNO. Compared to the other slightly symbolic data sets (phoneme probabilities and object indication probabilities), the DYNA parameters are comparatively unprocessed. We then used the discovered alphabet as an alphabet of gesture primitives. The HMMs in the alphabet were used to map the raw DYNA parameters to a time-varying vector of the likelihoods for each item in the gesture primitive alphabet. The probabilities of the gesture primitive alphabet were then combined with the other sets (also represented as probabilities) for further COGNO processing. Figure 4-5 outlines this process. The DYNA data set was run through COGNO in all combinations of 5-10 classifier states, 1-5 low-level HMMs, 1-5 states per low-level HMM, and timescales of 2, 4, 8, and 16, for a total of 600 different investigations.

Of the 600 different gesture primitive alphabets generated, we chose a few of the best-performing alphabets for use in the second stage of processing. In the second stage, we tried these different data combinations as inputs to the COGNO architecture:

- gesture primitive alphabet (alone) – a.k.a., gestures
- phoneme probabilites (alone) – a.k.a., speech

- object indication probabilities (alone) – a.k.a., pointing
- gestures and speech
- gestures and pointing
- speech and pointing

Each of these data sets was then run through COGNO in all combinations of 8-10 classifier states, 3-6 low-level HMMs, 1-5 states per low-level HMM, and timescales of 2, 4, 8, and 16, for a total of 240 different investigations per data combination.

To adequately train a classifier, the dimensionality of the input vector must be a reasonable size. Initial results suggested that running all three data sets together would result in an output space too large for training, unless orders of magnitude more training data were collected. We therefore trained on only one data set or two data sets at a time.

The performance of each of these data sets is further discussed in the next section.

4.6 Results and System Performance

The best alphabet produced by the first round of COGNO processing had 42.0 percent success in classifying the commands. Best guessing would have scored 28.7 percent success. While 42 percent is not remarkable compared to best guessing, the yield is about what should be predicted for the first round of processing. Poor performance is to be expected since gestures alone cannot always distinguish between the commands. For example, F1 and F2 are positioned near each other on Earth, and R4 guards Sirius. Therefore, "F1 goto Sirius" and "F2 ogg R4" can look very similar. In this first round of processing we were not looking to solve the classification problem, we were only looking to discover a gesture primitive alphabet to be used in the second round of COGNO processing.

The results of the second round of COGNO processing (using the best result from the first round) were, however, disappointing. None of the second-round experiments

showed significantly better classifying preformance than the performance achieved by the first round of processing. In fact, the only data set to do better was the combination of phoneme probabilities and the gesture primitive alphabet (the alphabet with 42.0 percent performance obtained from the first round of COGNO processing). Processing the joint speech and gesture data with COGNO parameters (five alphabet HMMs, five states per alphabet HMM, a timescale of eight frames, and eight states per high-level HMM) showed 45.8 percent success in classifying the commands. All other data combinations showed similar or slightly worse performance than the first round of COGNO processing.

Despite these disappointing results, the processed data does show evidence of improved classification when speech and gesture are processed together, as opposed to processing each separately. As an example, compare the most successful speech and gesture classifier (45.8 percent, described above) with the performance of the classifiers obtained by processing the phoneme probabilities or the gesture primitive alphabets alone with the same COGNO settings. (The COGNO settings in this case were five alphabet HMMs, five states per alphabet HMM, a timescale of eight frames, and eight states per high-level HMM). Processed together, the phoneme and gesture primitive probabilities produced a classification performance success rate of 45.8 percent. If that success were dependent on speech (or gesture) alone, you would expect the the classifiers produced by processing the speech (or gesture) data to show similar performance. Processed alone with the same COGNO parameters, the phoneme probabilites obtained 17.6 percent performance. The gesture primitive probabilities had a performance of 16.1 percent. As shown in the confusion matricies (tables 4.2, 4.1, 4.3), speech and gesture alone create different types of misclassifications. Speech show a bias towards the commands "F2 Ogg R4" and "F0 Pickup Armies," while gesture shows a bias towards the command "F0 Take Sirius." By combining speech and gesture, some of these misclassifications are corrected, leading to substantial improvements. This suggests that low-level speech and gesture co-processing has promise if some of the difficulties can be alleviated.

We believe our difficulties stem from three areas: the high dimensionality of the

	F0 Goto Sirius	F1 Goto Sirius	F2 Goto Sirius	F0 Ogg R4	F1 Ogg R4	F2 Ogg R4	F0 Take Sirius	F0 Pickup Armies
F0 Goto Sirius	**0**	0	0	0	0	0	0	0
F1 Goto Sirius	0	**0**	0	0	0	0	100	0
F2 Goto Sirius	0	0	**0**	0	0	0	100	0
F0 Ogg R4	0	0	0	**0**	0	0	100	0
F1 Ogg R4	0	0	0	0	**4**	0	96	0
F2 Ogg R4	0	0	0	0	40	**0**	60	0
F0 Take Sirius	0	0	0	0	1	0	**99**	0
F0 Pickup Armies	0	0	0	0	10	0	90	**0**

[p]

Table 4.1: Gesture: The confusion matrix for COGNO run on the primitive guesture alphabet. Performance = 16.1% (Some unused commands were removed.)

	F0 Goto Sirius	F1 Goto Sirius	F2 Goto Sirius	F0 Ogg R4	F1 Ogg R4	F2 Ogg R4	F0 Take Sirius	F0 Pickup Armies
F0 Goto Sirius	**0**	0	0	0	0	0	0	0
F1 Goto Sirius	0	**0**	0	0	0	100	0	0
F2 Goto Sirius	0	0	**0**	0	0	35	0	63
F0 Ogg R4	0	0	0	**0**	0	60	17	21
F1 Ogg R4	0	0	0	0	**0**	62	0	39
F2 Ogg R4	0	0	0	0	0	**32**	0	68
F0 Take Sirius	0	0	0	0	0	33	**14**	53
F0 Pickup Armies	0	0	0	0	0	59	11	**29**

Table 4.2: Speech: The confusion matrix for COGNO run on the phoneme probabilities. Performance = 17.6% (Some unused commands were removed.)

	F0 Goto Sirius	F1 Goto Sirius	F2 Goto Sirius	F0 Ogg R4	F1 Ogg R4	F2 Ogg R4	F0 Take Sirius	F0 Pickup Armies
F0 Goto Sirius	**0**	0	0	0	0	0	0	0
F1 Goto Sirius	0	**0**	0	0	80	0	0	20
F2 Goto Sirius	0	0	**0**	0	63	0	0	37
F0 Ogg R4	0	0	0	**0**	100	0	0	0
F1 Ogg R4	0	0	0	0	**96**	4	0	0
F2 Ogg R4	0	0	9	0	44	**41**	0	5
F0 Take Sirius	0	0	20	0	72	2	**0**	7
F0 Pickup Armies	0	0	0	0	44	10	0	**45**

Table 4.3: Speech and Gesture: The confusion matrix for COGNO run on the primitive guesture alphabet and phomene probabilities. Performance = 45.8% (Some unused commands were removed.)

phoneme probability vector, discovering a good low-level representation of gesture, and the different rates at which speech and gesture primitives occur.

The phoneme probability is an input vector is of dimensionality 39. Such a large dimensionality requires a copious amount of data for accurate training. We hoped that using multiple input streams would help distinguish between subtle ambiguities in the speech data (thus requiring less training data). In most cases, however, including the speech data added too much complexity to the system. This hurt the performance of the training: The other variable alone performed better. In our few successes, I suspect the training found a correlation between a command and a characteristic phoneme, allowing the system to focus on that phoneme for classification.

Our second problem has to do with finding an optimal gesture primitive alphabet. In this experiment, the selective pressure for COGNO processing was to identify the correct command. However, the gesture alphabet that provides the best classification of the commands is not necessarily a good representation of gesture primitives. The first round of COGNO processing showed better performance on the higer times scales (8, 16), and with more complicated HMMs(5 or 6 states per low level HMM) This suggests that these alphabets are more complex and thus less likely to represent the gesture primitives we are looking for. Complex sysbols with longer durations are also non-optimal for integration with speech primitives, already an obstacle to high performance.

Speech and gesture occur at different rates. Speech events occur at a much higher frequency than gesture events. Before doing this experiment, however, the comparitive rates between speech and gesture primitives were not known. We found that in the most successful gesture alphabets, events occured at a much slower rate than phoneme probabilities. While executing just one gesture primitive, ten or more phonemes could occur. This difference in rates also exasperates the problem of speech-gesture syncronization. A small difference in the timing of a gesture (with respect to gesture time) would mean the gesture occurs with a completely different phoneme. Figure 4-6 shows a comparison between the rate of phoneme activation probabilites and a likelihood trace of one item in the gesture alphabet that had the best classi-

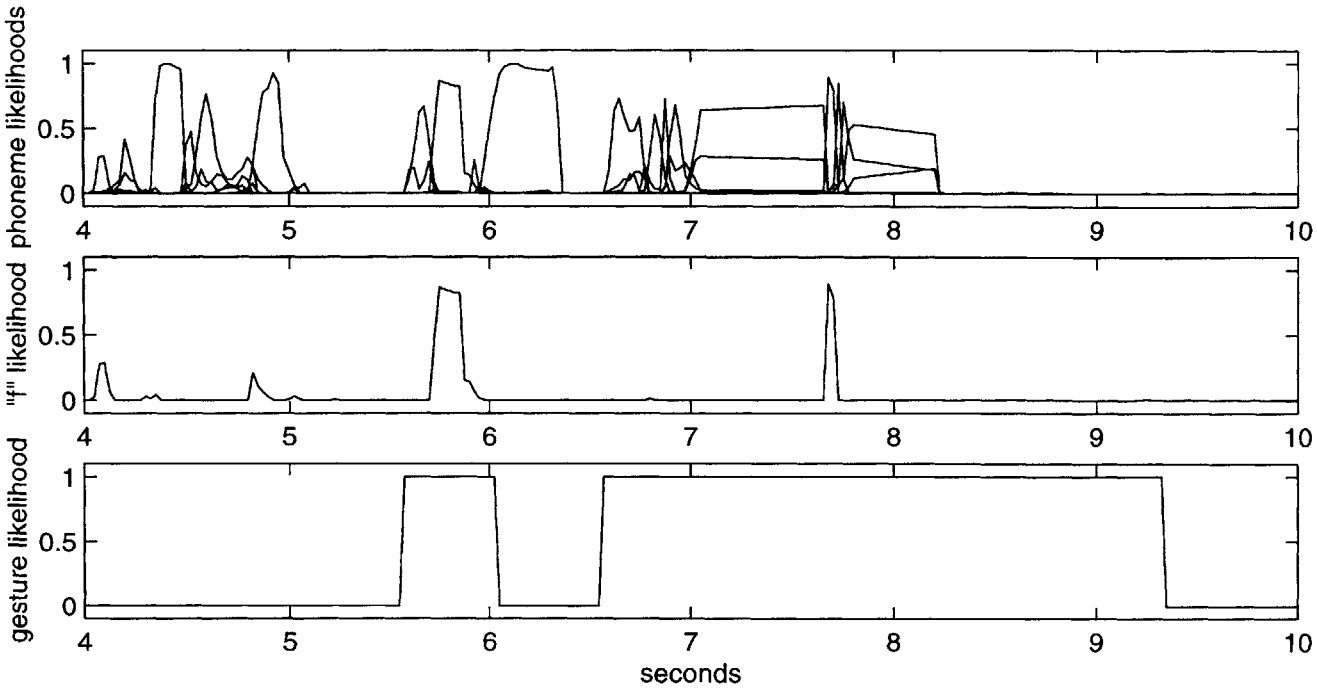

Figure 4-6: These graphs show a set of co-occuring speech and gesture data. The top graph shows the likelihoods of a selected group of phonemes over time. The middle graph shows the likelihood of the phoneme "f" over time. These show rates of about 10 phonemes per seconds. In comparison, the bottom graph shows the time-varying likelihood of one item in the low-level gesture alphabet. We can we that the gesture durations (0.5 - 3.0 seconds) are much longer than the duration of the phonemes.

fication performance, for one segment of collected data. The simple straightforward HMM's used in this experiment are not well-suited to integrating data with these types of data relationships.

Possible solutions to these problems presented here are discusssed in Section 5.2, Improvements and Future Work.

Chapter 5

Conclusions

For this thesis we investigated two different approaches to combining speech and gesture into a multimodal system. For the *Ogg That There* system, we took a high-level approach; gesture was immediately represented as the indicated object, and speech was immediately represented as the coded words of the program. This resulted in a functional, but rigid application. While not an ideal final interface, *Ogg That There* was an important developmental step. *Ogg That There* led to key insights on the interface requirements for further applications by testing the feasibility of Netrek as an experimental test-bed. It also provided a standard of comparison for future approaches.

In our second approach we investigated combining speech and gesture at a very low level. By changing to a low-level integration procedure, we aimed at eliminating excessive (and possibly detrimental) human assumptions about how speech and gesture are correlated. This experiment showed evidence of improved classification when speech and gesture are processed together, as opposed to the classification with either the speech or gesture data alone. It also provided insights about finding and representing low-level gesture primitives, using of phoneme probabilities as input to the COGNO architecture, and synchronizing speech and gesture primitives.

In the course of this research, we encountered difficulties – some related to our specific application, but most reflective of problems faced in the development of any multimodal system. These difficulties are discussed in section 5.1. Section 5.2 dis-

cusses potential improvements to the system, some of which are research projects unto themselves. Section 5.3 presents possible uses for the results of a good speech-gesture representation and integration.

5.1 Difficulties

5.1.1 Feature Representation

One of the challenges faced in this project was determining the best representation of speech and gesture information. For the *Ogg That There* system, we looked for meaning in the user's actions after the input features had been processed to a very symbolic level. Symbolic representation has many benefits. Symbols have clearly defined meanings, and are easy to combine into more meaningful constructs. At the symbolic level it is also very easy to develop and define constraints on allowable constructs, simplifying the search for possible meanings. However, to create symbolic representations of a person's actions, you must abstract the user's actions into predefined symbols. This ignores much of the information about what the user is actually doing and misses potential correlations between lower-level aspects of the data.

Using the data from our wizard of oz experiment, we looked at this problem from the opposite direction. Using the COGNO system we endeavored to discover an alphabet based on speech and gesture primitives. In the process, we hit two stumbling blocks.

First, low-level information has a much higher dimensionality than a symbolic representation of the same thing. This makes it difficult to manage. In the case of our speech data, the input vector was so large that we needed orders of magnitude more training data if we were to properly identify and train the low-level alphabet HMMs.

Second, the exact character of low-level gesture alphabet is unknown. We found that the alphabet that provides the optimal performance is not necessarily the alphabet best suited to further processing. The gesture alphabets we found were optimized

for best representing an entire command and therefore were inclined to be complex and act over the larger time scales we investigated. These are not the characteristics we are looking for in a low-level gesture primitive alphabet. Unfortunately, it's unclear what the best selective pressures would be to discover a good set of gesture primitives.

Creating representations of speech and gesture that are both descriptive and manageable continues to be a challenge.

5.1.2 Co-processing Speech and Gesture

As we saw with the *Ogg That There* system, lack of feedback between speech and gesture leads to a non-optimal interface. The obvious next step is to co-process speech and gesture, but implementing this step leads to difficulties. First, speech and gesture have different time scales. A good sampling of speech needs to take place at about 100Hz. Gestures, however, easily can be tracked at 30 frames a second and usually can be tracked with less. Many phonemes can be uttered in the course of one gesture. Furthermore, there isn't a guaranteed sequential correspondence between speech and gesture. For example, a person may to point to a place and say, "I'm going there." He may point first, and then say the statement; he may say the statement, and then point; or he may point and say the statement simultaneously. All of these are valid with the same meaning, but our sequential integration of speech and gesture would treat them as different commands.

Both of these problems suggest that the best strategy would use a more complex approach to speech and gesture co-processing than the straightforward integration attempted in Chapter 4.

5.1.3 Creating Application-Interface Synergy

Another challenge tackled in this thesis is creating an interface that allows the user to easily express what he needs to communicate in a timely manner with respect to the application. The *Ogg That There* system was inferior in this respect. We attempted

to correct this problem by changing both the application and the modes of expression for our second system.

In this project we also were held back because our application was unable to completely support the type of multimodal expression we wanted to implement. Specifically, we wanted to implement an interface that supported more complex directions to the robot, and thus more complex gestures to express those directions. Our current version of the robot is only able to support simple direct-action commands, however.

5.2 Improvements and Future Work

5.2.1 A Different Speech Model

Using a different parameterization of speech is one way to avoid the problems associated with using phoneme probabilities. Tony Jebara, of the MIT Media Lab, is currently developing such an alternative. His system uses principle componet analysis[3, 16] to take advantage of correlations between the different phoneme probabilities and thus reduce the dimensionality of the representation to ten parameters. This method has recognition rates above 90 percent of the original speech data. Using this parameterization could significantly reduce the amount of data needed as input to COGNO, thus making the search for speech and gesture primitives more manageable.

5.2.2 Gesture Primitives

Ideally we would like to use COGNO to discover a set of HMMs that well represent an alphabet of generic gesture primitives. We could then use those HMMs for the first stage of processing of the DYNA data, rather than trying to discover gesture primitives from our command sample data.

One approach to finding an alphabet of gesture primitives is to collect a fully labeled example set of generic gestures. Examples could include people pointing to arbitrary objects, people moving around at their desk, or people tracing out shapes in the air, among other tasks. Running COGNO on these gestures should create a low-

level alphabet which better represents the desired generic gesture primitives. Two factors contribute to the higher performance. First, COGNO is dealing with a simpler set of gestures. This should make it more straightforward to cluster primitives within the example set. Second, the selective pressure is to best identify the gesture – not to identify a more complex command. Hopefully, this simpler selective pressure will avoid accidental encoding of higher-level ideas in our gesture primitive alphabet.

5.2.3 Coupled HMMs for Speech and Gesture Integration

Speech and gesture co-occur, but components of a speech and gesture communication don't necessarily occur in a predictable sequence. We can look at the gestures as a series of events, and the speech as a series of events, but how these two co-occurring sequences will align with each other is not guaranteed to be identical to any previous co-occurrence.

Using coupled HMMs could address this problem. Coupled HMMs are designed to model multiple interacting processes where there are causal influences between the processes, but no strict serialization of the processes' events. Coupled HMMs have been successfully used to model two-handed actions[7]. Integrating the data streams representing the motion of independent hands involved in the same action is a similar problem to integrating speech and gesture data streams involved in multimodal communication.

5.3 Long-Term Avenues

After some of the underlying problems have been solved, we can then build upon our basic system to develop a more interesting multimodal environment. A couple interesting avenues of research are integrating new and different types of data streams, and expanding the user's freedom of expression.

5.3.1 Integrating New Data Streams

Galvanic skin response and pitch tracking of speech are two examples of data streams that would be interesting to integrate. These could be used to monitor the user's stress level, which in turn could influence the behavior of the robots. For instance, the more anxious the user is, the faster the ships move.

More gesture dimensions could also serve as sources of input data. Rather than relying solely on the location of the user's hands and head, we could use the angular pitch of the user's torso, the angle of the head relative to the torso, and even biometrically measured muscle tension to control the robots. A user might tense their muscles to increase the speed of attacking ships, or tilt their head to change the view of the game area.

5.3.2 Expanding Game Complexity and the User's Range of Expression

Implementing support for more interesting and complex commands is one way to enrich the user's experience with our multimodal system. This will lead to a more exciting game with more options for control of the robots, thus allowing new and different strategies. This also creates a richer research environment, with more interesting and complex gestures and speech patterns to interpret.

Along these lines, work is already under way to reimplement the Netrek robots using Bruce Blumberg's reactive behavior architecture[4]. Specifically, users will not only command the proximate behavior of the ships, but will also provide strategy-level commands. For example, rather than just telling a robot to "goto Sirius," one might tell it what path to take, which enemies to avoid along the way, what other opportunities would be more important than going to Sirius, and what their relative priorities are.

We can also work toward providing a more natural and adaptive interface for the user. One possible solution is a system that incorporates on-line learning of the user's commands. For instance, we could start with a limited set of pre-learned simple

commands that provide limited functionality (such as "F1 ogg R4"). The user could use these commands, but also inject natural speech and gesture into the system. The user would do (gesture/speak) as they like, naming objects and gesturing things as they would normally – without attempting to fit the interface's predefined commands. Based on the context of the user's actions – and with the aid of corrective feedback from the user, the interface would eventually learn the correlations between the user's speech/gesture and commands.

This supports the idea of more complicated user-defined constructs. With this system, the user could create a name for a group of ships in order to refer to them collectively, or the user could develop a gesture that represents one of his favorite strategies of attack. In general, this tackles the challenge of adapting the interface to the user, rather than expecting the user to adapt to the interface.

Bibliography

[1] Ali Azarbayejani and Alex Pentland. Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features. In *Proceedings of 13th ICPR*, Vienna, Austria, August 1996. IEEE Computer Society Press.

[2] M. Bates, R. Bobrow, P. Fung, R. Ingria, F. Kubala, J. Makhoul, L. Nguyen, R. Schwartz, and D. Stallard. The bbn/harc spoken language understanding system. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 111 –114, 1993.

[3] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In *Proceeding of the Workshop on Motion of Nonrigid and Articulated Objects*. IEEE Computer Society, 1994.

[4] B. Blumberg. Action-selection in hamsterdam: Lessons from ethology. In *The Proceedings of the 3rd International Conference on the Simulation of Adaptive Behavior*, Brighton, August 1994.

[5] R. A. Bolt. 'put-that-there': Voice and gesture at the graphics interface. In *Computer Graphics Proceedings, SIGGRAPH 1980,*, volume 14, pages 262–70, July 1980.

[6] Marie-Luce Bourguet and Akio Ando. Synchronization of speech and hand gestures during multimodal human-computer interaction. *CHI 98, pg 241-242*, 1998.

[7] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden markov models for complex action recognition. In *Computer Vision and Pattern Recognition*, pages 994–9, Los Alamitos, CA, USA, June 1997. IEEE.

[8] Brian P. Clarkson Christopher R. Wren and Alex P. Pentland. Understanding purposeful human motion. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[9] Brian Clarkson and Alex Pentland. Unsupervised clustering of ambulatory audio and video. In *ICASSP'99*, 1999.

[10] Claude Valot Cristophe Mignot and Noelle Carbonell. An experimental study of future natural multimodal human-computer interaction. *INTERCHI'93*, 1993.

[11] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communications. *Communications of the Association for Computing Machinery*, 30(11):964–972, 1987.

[12] S. Furui. Unsupervised speaker adaptation method based on hierarchical spectral clustering. In *Proceedings of ICASSP*, pages 286–289, 1989.

[13] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing*, 1994.

[14] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. European Conference on Computer Vision*, pages 343–356, Cambridge, UK, 1996.

[15] David B. Koons and Carlton J. Sparrell. Iconic: Speech and depictive gesture at the human-machine interface. *CHI 94*, 1994.

[16] Penio S. Penev and Joseph J. Atick. Local feature analysis: A general statistical theory for object representation. *Network-Computation in Neural Systems*, 7(3):477–500, 1996.

[17] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[18] Sandrine Robbe. An empirical study of speech and gesture interaction: Towards the definition of ergonomic design guidelines. *CHI 98, pg 349-350*, 1998.

[19] T. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Trans. Neural Networks*, 1994.

[20] Deb Roy. *Learning Words from Sights and Sounds: A Computational Model.* PhD thesis, Massachusetts Institute of Technology, 1999.

[21] S. Seneff and V. Zue. Transcription and alignment of the timit database. In *Proceedings of the Second Symposium on Advanced Man-Machine Interface through Spoken Language*, November 1988.

[22] Antonella DeAngeli Sharon Oviatt and Karen Kuhn. Integration and synchronization of input modes during multimodal human-computer interaction. *CHI 97, pg 415-422*, 1997.

[23] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Proceedings of International Symposium on Computer Vision*, Coral Gables, FL, USA, 1995. IEEE Computer Society Press.

[24] Minh Tue Vo and Alex Waibel. A multimodal human-computer interface: Combination of gesture and speech recognition. *InterCHI 93, pg 69-70*, 1993.

[25] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.

[26] Christopher R. Wren. *Understanding Expressive Action.* PhD thesis, Massachusetts Institute of Technology, 2000.

[27] Christopher R. Wren and Alex P. Pentland. Dynamic models of human motion (long version). Technical Report 415, MIT Media Laboratory Perceptual Computing Group, 1997. http://www.media.mit.edu/vismod/.

[28] Christopher R. Wren and Alex P. Pentland. Dynamic models of human motion. In *Proceedings of FG'98*, Nara, Japan, April 1998. IEEE.