AFCRL-71-0530

AD735300

RESEARCH IN ON-LINE COMPUTATION

by

David O. Harris, James A. Howard, Roger C. Wood

University of California
Santa Barbara, California 93106

Contract No. F19620-70-C-0314
Project No. 8684

FINAL REPORT

1 July 1970 - 31 August 1971

Date of Report: 30 September 1971

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

Approved for public release; distribution unlimited.

JAN 7 1972

Contract Monitor: Hans H. Zschirnt
Data Sciences Laboratory

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
Springfield, Va. 22151

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 865
Monitored by
AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

87

AFCRL-71-0530

RESEARCH IN ON-LINE COMPUTATION

by

David O. Harris, James A. Howard, Roger C. Wood

University of California
Santa Barbara, California 93106

Contract No. F19620-70-C-0314
Project No. 8684

FINAL REPORT

1 July 1970 - 31 August 1971

Date of Report: 30 September 1971

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

Approved for public release; distribution unlimited.

Contract Monitor: Hans H. Zschirnt
Data Sciences Laboratory

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 865
Monitored by
AIR FORCE CAMBRIDGE RESEARCH LABORATORIES
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
BEDFORD, MASSACHUSETTS 01730

Program Code No. OD30

Effective Date of Contract ..... 1 July 1970

Contract Expiration Date ..... 31 August 1971

Principal Investigator and Phone No. ..... Dr. David O. Harris/ 805 961-2534

Project Scientist or Engineer and Phone No. .... Dr. Hans H. Zschirnt/ 617 861-3671

Qualified requestors may obtain additional copies from the Defense Documentation Center. All others should apply to the National Technical Information Service.

A

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
University of California
Santa Barbara, California 93106

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
RESEARCH IN ON-LINE COMPUTATION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Scientific. Final. 1 July 1970 - 31 August 1971 Approved 4 Nov. 71

5. AUTHOR(S) (First name, middle initial, last name)
David C. Harris
James A. Howard
Roger C. Wood

6. REPORT DATE
30 September 1971

7a. TOTAL NO. OF PAGES
86

7b. NO. OF REFS
30

8a. CONTRACT OR GRANT NO. ARPA ORDER NO. 865
F19628-70-C-0314

b. PROJECT, Task, Work Unit Nos.
8684 n/a n/a

c. DoD Element 611CD

d. DoD Subelement n/a

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
AFCRL-71-0530

10. DISTRIBUTION STATEMENT
A - Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES
This research was supported by the Advanced Research Projects Agency.

12. SPONSORING MILITARY ACTIVITY
Air Force Cambridge Research Laboratories (LR)
L. G. Hanscom Field
Bedford, Massachusetts 01730

13. ABSTRACT

Developments for the ARPA network include a Network Control Program, a User Telnet and providing a variety of services for network users. Development for the On-Line System included reducing core requirements, improvements to the internal scheduling algorithms, and developing a multi-line controller to provide greater latitude in system capability. The speech project continued progress in the wave function analysis/synthesis techniques, classification and recognition of phonetic information and techniques to employ for data compression.

DD FORM 1 NOV 65 1473

Unclassified
Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Network Control Program						
Telnet						
Data Reconfigure Service						
Multi-Line Controller						
High Speed Data Link						
Preprocessing Acoustic Waveforms						
Phonetic recognizer						
Steady-State Vowel Recognition						
Data Compression						

ABSTRACT

Developments for the ARPA network include a Network Control Program, a User Telnet and providing a variety of services for network users. Development for the On-Line System included reducing core requirements, improvements to the internal scheduling algorithms, and developing a multi-line controller to provide greater latitude in system capability. The speech project continued progress in the wave function analysis/synthesis techniques, classification and recognition of phonetic information and techniques to employ for data compression.

TABLE OF CONTENTS

ABSTRACT

PART I - Introduction

List of Scientists and Engineers..... i

Related Research..... i

List of Publications and Reports..... ii

PART II

Technical Findings and Major Accomplishments..... 1

A. Software..... 1

I. ARPA Network..... 1

(a) Network Control Program..... 1

(1) Software Structure..... 2

(2) Operator Control..... 3

(b) Network Access From Local Programs..... 6

(1) Assembly Language..... 7

(2) Fortran and PL/1..... 10

(3) On-Line System..... 13

(4) User Telnet..... 16

(c) Access to Local Programs From the Network..... 18

(1) On-Line System, Graphics Supported..... 19

(2) Batch Processing..... 22

(3) File Storage..... 24

(4) On-Line System, Server Telnet..... 25

(d) Network Pie-In-The-Sky..... 27

II. UCSB On-Line System..... 28

B. 360/75 On-Line System, Hardware..... 32

C. Speech Project..... 34

(a) Preprocessing of the Acoustic Waveform..... 34

(b) Wave Function Analysis/Synthesis..... 45

(c) Computer Classification and Recognition of Phonetic Information..... 48

(d) Data Compression Studies..... 72

Conclusion..... 75

PART I - Introduction

List of Scientists and Engineers Contributing to the Research

Dr. David O. Harris

Dr. James A. Howard

Dr. Roger C. Wood

Mr. Roland F. Bryan

Mr. Ronald Stoughton

Related Research

ARPA - Project 8684

Improved the On-Line System by expanding the mathematical capacity of the system adding remote job entry features and improving overall system useability and reliability. Connected the UCSB system to the ARPA network and accomplished basic network tests. This project also sponsored research related to computer communication through human speech. Research developed the speech waveform analysis/synthesis concept and basic laboratory facilities necessary for further development. Speech parameters were defined, techniques developed and research conducted to effectively analyze and synthesize the essential elements of human speech.

List of Publications and Reports Resulting from Sponsorship of the Contract

Publications

1. Carey, Bernard and Markel, John, "Simulation of a New Wave Function Analysis/Synthesis System for Speech".
2. White, John, "Surface Display in the UCSB On-Line System".
3. Carey, Bernard, "Wave-Function Speech Analysis Based on GCM Model of Acoustic Waveform" July 1970.
4. Howard, James A. and Pfeifer, Larry, "An ROM Bootstrap Loader for Small Computers", Computer Design October 1970, pp. 95-97.
5. Markel, John D. and Carey, Bernard, "Digital Voice Processing With a Wave Function Representation of Speech", Fall Joint Computer Conference, 1970, pp. 387-397.
6. Proctor, William L., "The Nanohumper, a Preprocessing Analog to Digital Converter for Speech Waveforms" October 1970.
7. White, Jim, "Specifications for Network Use of the UCSB On-Line System" October 16, 1970, NIC 5417 RFC 74.
8. White, James, "An NCP for the ARPA Network" December 21, 1970.
9. White, James E., "Network Specifications for Remote Job Entry and Remote Job Output Retrieval at UCSB" March 22, 1971 NIC 5775 RFC 105.
10. Nelson, Gary A., Pfeifer, Larry L. and Wood, Roger C., "High Speed Octave Band Digital Filtering" April 15, 1971 (accepted for publication in the IEEE Transactions On Audio and Electroacoustics; possible publication date: March 1972).
11. Krilanovich, Mark, "Network Fortran Subprograms" April 21, 1971 NIC 5831 RFC 119.
12. Krilanovich, Mark, "Network PL1 Subprograms" April 21, 1971 NIC 5832 RFC 120.
13. Krilanovich, Mark, "Network On-Line Operators" April 21, 1971 NIC 5833 RFC 121.
14. Krilanovich, Mark and White, James E., "UCSB Network Processes and Interfacing Software" April 26, 1971.
15. White, James E., "Network Specifications for UCSB's Simple-Minded File System" April 26, 1971 NIC 5834 RFC 122.

16. Howard, J.A., Nelson, G.A., and Ordung, P.F., "On-Line Computing Systems for Education at UCSB", Proc. Purdue Symposium on Applications of Computers To Elect. Engr. Education, pp. 556-563, Purdue Univ., Lafayette, Indiana, April 26 - 28, 1971.

17. Wood, R.C. and Bayne, J., "Teaching Engineering Applications of Complex Variables with an Interactive Computing System", Proc. Purdue Symposium on Applications of Computers to Elect. Engr. Education, pp. 567-572, Purdue University, Lafayette, Indiana, April 26 - 28, 1971

18. Carey, Bernard Joseph, "A Method for Automatic Time Domain Analysis of Human Speech" June 1971, PH.D. Thesis.

19. Howard, J.A., Ordung, P.F. and Wood. R.C., "On the Use of an Interactive Computing System in Engineering Education", Proc. of IEEE, Vol. 59, No. 6, pp. 969-975, June 1971.

20. White, Jim, "NCP Operator's Guide" June 4, 1971 NIC 6797.

21. Greaves, John Oliver Bernard, "An On-Line Television Computer System for the Study of the Behavior of Microorganisms" July 1971, PH.D. Thesis.

22. McAfee, J. and Presser, L., "An Algorithm for the Design of Simple Precedence Grammars" July 1971.

23. White, James E., "A User TELNET Description of an Initial Implementation" August 9, 1971 NIC 7176 RFC 206.

24. Howard, J.A., Ordung, P.F., and Wood, R.C., "On-Line Systems for Engineering Education - State of the Art", IEEE Transactions on Engineering Education, November 1971.

25. Wood, R.C. and Bruch, J.C., "Teaching Complex Variables with an Interactive Computing System", IEEE Transactions on Engineering Education, November 1971.

Reports

26. First Quarterly Report, Reporting Period: July 1, 1970 - September 30, 1970

27. Second Quarterly Report, Reporting Period: October 1, 1970 - December 31, 1970

28. Third Quarterly Report, Reporting Period: January 1, 1971 - March 31, 1971

29. Fourth Quarterly Report, Reporting Period: April 1, 1971 - June 30, 1971

30. Semianmual Technical Report, Reporting Period: July 1, 1970 - December 31, 1970

PART II

Technical Findings and Major Accomplishments

A. Software

I. ARPA Network

In this contract period, a Network Control Program was designed, implemented and modified as required to fully support the Host-Host protocol currently specified by the Network Working Group. Access to the Network has been extended to include Fortran and PL/1 programs and On-Line System users, as well as assembly-language programs. A User Telnet has been written in accordance with the protocol adopted to handle such teletype-like communication.

A number of services have been made available to Network users. Two separate specifications for access to the On-Line System have been designed and implemented, each supporting the graphic display features of the system. A remote job entry facility has been written, providing Network users with access to UCSB's batch processing facilities. A Network file system has been written, making on-line, direct-access storage available to the Network community. Finally, a teletype-oriented interface to the On-Line System has been implemented according to the Telnet protocol adopted by the Network Working Group.

All but the most recently coded of these software projects have been documented, and that documentation distributed to members of the Network community.

(a) Network Control Program

During this contract period, a Network Control Program (NCP) was designed and implemented. Supporting initially the Network Working Group's (NWG's)

Host-Host protocol specification as embodied in "Host-Host Protocol Document No. 1", it has been appropriately modified as protocol changes have been adopted by the NWG, and at this writing operates in accordance with the most recent protocol specificiation, adopted this July.

UCSB's NCP implementation has been thoroughly documented in a paper entitled "An NCP for the ARPA Network" (21 December 70, NIC 5480) and distributed to Network participants through the Network Information Center (NIC) at Stanford Research Institute in Menlo Park. Some of the material contained in that paper, and a description of more recent additions to the NCP, are briefly described in the sections which follow.

(1) Software Structure

The NCP has been implemented as a normal, batch-mode job, and is entirely self-contained and independent of all other software systems in the host, including the Operating System (OS). While in execution, the NCP causes itself to appear as if it were a part of OS. Upon initiation, the NCP instates itself as the System's Supervisor Call First Level Interrupt Handler (SVC FLIH). Applications programs issue a particular SVC to communicate with the NCP; as the System's SVC FLIH, the NCP is in a position to intercept that SVC. In a similar fashion, the NCP instates itself as the System's Input/Output (I/O) FLIH, and in this capacity intercepts I/O interrupts from the IMP Interface; hence, the NCP can initiate I/O operations to the Interface using the Start I/O (SIO) instruction. In its role as SVC FLIH, the NCP intercepts occurrences of SVC 34, the supervisor call employed internally by OS to schedule operator commands for execution. Entries made by the operator at his console which are prefixed with the character "@" are trapped by the NCP and interpreted as NCP operator commands. Thus the System's operator is afforded a convenient means of affecting the NCP's

operation in those ways necessary to make operation of the NCP feasible in a System which supports both time-sharing and batch operations, in addition to Network activity. By constructing its own Data Extent Block (DEB), the NCP can use OS's Execute Channel Program (EXCP) macro instruction to output messages to the operator's console, a device to which the NCP would not otherwise have direct access. Using EXCP, the NCP can prefix its responses to operator commands with the character "@", making them easily recognizable to the operator. Finally, by overlaying a subroutine of OS which is entered every time a task leaves the System, the NCP is able to gain control as each applications program which used the services of the NCP terminates. Hence, NCP resources which the user neglected to release, or could not release because of an unexpected abnormal termination (abend), can be reclaimed.

All modifications made to the System by the NCP are made dynamically, and are removed by the NCP before it terminates execution. Thus, the NCP's passage through the System appears exactly like that of any other job, independent of whether it terminates normally or abends.

(2) Operator Control

The NCP provides the System's operator with a comprehensive command set through which the NCP's operation can be monitored, and influenced when necessary. Such control capability was deemed necessary to make the NCP's regular insertion into and removal from the System convenient for both operator and user. A portion of the command set was designed primarily for use by system's programmers during checkout of other, Network-related software. The operator command set is briefly described in the following paragraphs. A more thorough description can be found in a paper entitled "NCP Operator's

Guide" (4 June 71, NIC 6797) distributed to Network participants through the NIC.

MENU is a command provided to aid the operator in recalling the key-words for other commands; it does nothing more than list the command set on the operator's console.

The STAT command returns a one-line summary of the state of the NCP, including the number of local sockets in use and the condition of the IMP Interface. The operator is also reminded of previously issued commands which remain in effect and restrict the NCP's operation.

LIST augments the STAT command by providing detailed information about all or a subset of local sockets. For each socket about which the NCP is queried, the following information is displayed: the name of the job using the socket, the state of the socket, and the host and socket to which it is connected.

DRAIN provides the operator with a means for gradually decreasing the level of NCP utilization in anticipation, for example, of an Initial Program Load (IPL). While a DRAIN command is in effect, the NCP permits no new Network connections to be initiated, and when the last existing connection has been terminated, so informs the operator.

STOP causes all existing connections to terminate by forcibly closing them. Those local processes whose connections were so broken are made aware of the reason for the intrusion; to the remote processes involved, it appears as though the local processes had broken the connections themselves. During the termination sequence, the NCP prohibits new connections from being initiated. When all sockets have been released, the operator is so informed.

The START command is provided to initiate Network operations after the NCP is loaded. Issuance of the command causes the IMP Interface to be initialized, a few NOPs to be transmitted to the Interface Message Processor (IMP), and a Reset control command to be sent to each known host in the Network. Thereafter, START serves to nullify a previously issued DRAIN or STOP command.

CANCEL is provided as a convenient means for terminating execution of the NCP. CANCEL causes a STOP operator command to be simulated. After a brief pause, the NCP transmits a Host-Going-Down control message to its IMP. Then, after a second short pause, the NCP disables the IMP Interface and removes itself from the System.

The LOG command enables the operator to record a comment in a log maintained by the NCP on secondary storage. The text following the keyword "LOG" is simply recorded.

TEST is the first of a series of commands designed to aid both software and hardware debugging. TEST causes all subsequent I/O interrupts generated by the IMP Interface (until the next START command) to be logged on the operator's console along with their source (i.e., input or output segment of the Interface) and status information obtained from the Channel Status Word (CSW).

To permit software development to proceed during periods when the IMP or IMP Interface is inoperative, the NCP is equipped with a mode of operation in which the functions of the hardware are simulated. Two operator commands - INT and EXT - are provided to permit dynamic switching between the NCP's normally operating (called external) mode and this special (called internal)

mode. A third command - HOLD - aids in achieving a smooth transition between modes by suspending output to the IMP Interface until a START command is issued.

RESET causes a Reset control command to be transmitted to one or more designated hosts. If and when the host acknowledges by returning a Reset Reply control command, the operator is so informed.

An operator command called FIND is provided to aid in the debugging of applications programs which use the services of the NCP. It can be employed to obtain the main storage address of any program within the System, given its name and the name of the job with which it is associated; to terminate a program by abending it; and to take a dump of the program.

Finally, the RELOAD command causes the NCP to be restarted. A new copy of the load module is obtained from secondary storage, and the NCP is reentered at its entry point. RELOAD provides a means for completely refreshing the software after a failure in the NCP, and a means for switching between two versions of the NCP, a developmental version and a production one.

(b) Network Access From Local Programs

The NCP provides any assembly-language program within the 360/75 with full access to the Network. Such access is obtained by issuing an SVC, which is intercepted and processed by the NCP. During this contract period, such access has been extended to programs written in Fortran and PL/1, and to On-Line System (OLS) users. To support Fortran and PL/1 applications, two sets of subroutines have been written and placed in the System's Fortran and PL/1 subprogram libraries, respectively, and hence can be invoked from a

program coded in either of those languages. To support OLS users, a third subsystem (i.e., in addition to those supporting the Card Oriented Language (COL) and the Mathematically Oriented Language - Single Precision Floating Point) of OLS called NET has been created and outfitted with a set of operators which provide access to the Network. Both Fortran and PL/1 subroutines and NET operators perform NCP supervisor calls for the user.

The conventions for communicating with the NCP from assembly language, Fortran, and PL/1 programs, and from OLS terminals, are briefly described in the following sections.

(1) Assembly Language

A set of macro instructions has been provided to aid the assembly-language programmer in invoking the NCP. Each macro instruction builds a parameter list, loads its address into a general purpose register, and issues the NCP's SVC. A macro instruction is provided for each service that can be invoked. In general, calling the macro serves only to initiate a Network operation; the programmer assumes responsibility for assuring the successful completion of each operation by (in most cases) directing an OS WAIT macro instruction to an Event Control Block (ECB) associated with the operation, which is posted by the NCP when the requested function has been performed.

The paragraphs which follow describe briefly the macro instructions provided. Detailed descriptions of all but the most recently written can be found in "An NCP for the ARPA Network". The term "identifier" denotes a five-byte field containing in its high-order byte a host address and in its low-order 32-bits the number of a socket at that host.

The three connection-establishing operations - Connect, Listen, and Accept - are invoked by variations of a macro instruction called "@OPEN", called as follows:

@OPEN ECB=address [,LSCK=identifier$_1$, WS=address [,SP=subpool] [,BS=byte-size] [,FSCK=identifier$_2$]]

where operands enclosed in brackets are optional. If both LSCK (local socket identifier) and FSCK (foreign socket identifier) are specified, the operation requested is Connect; if only LSCK is specified, the operation is Listen; and if neither is specified, it is Accept. If LSCK specifies a send socket, then BS may be specified and determines the byte size for the connection (all valid byte sizes are supported). If the operation is Listen, the NCP returns the calling, foreign socket's identifier in the workspace WS. The optional operand SP serves to place the socket in a user-defined group of sockets; the NCP permits certain operations to be applied to an entire subpool of local sockets.

The @CLOSE macro instruction whose calling sequence is simply:

@CLOSE ECB=address

serves to close on existing connection, retract a connection request, or reject a request for connection.

To transmit data through a send socket, the following macro is invoked:

@WRITE ECB=address, ADDR=buffer-address [, OFFSET=buffer-offset], BITS=buffer-length

ADDR and OFFSET specify an address, and a bit offset from that address, at which the bit string to be transmitted begins (OFFSET defaults to zero).

BITS defines the length of the bit string, and it may have any value - positive, negative, or zero. A negative or zero length implies no operation, and any length string (up to the capacity of the machine) may be sent in a single @WRITE operation. @WRITE's are issued by the user without regard to the byte size specified for the connection, or for the maximum length message transmittable through the subnet. Breaking bit strings into Network messages and properly formatting them are the sole responsibility of the NCP.

The counterpart of @WRITE for receive sockets is:

```
@READ ECB=address, ADDR=buffer-address
      [, OFFSET=buffer-offset], BITS=buffer-length
```

where the buffer designated is a receiving, rather than a source buffer. Again, no restriction is imposed upon the size of the buffer, and sole responsibility for extracting or concatenating the bit string from Network messages, and for transmitting appropriate allocations to the sender, rests with the NCP.

To transmit an interrupt over a connection, the calling sequence is:

```
@SIGNAL ECB=address
```

A macro instruction is provided for obtaining information about a local socket, and called as follows:

```
@CHECK LSCK=identifier, WS=address
```

The NCP places in the user's workspace the following information: the state of the socket, the identifier of the socket connected to it, and information about the amount of data queued by the NCP or awaited by the user.

Another variation of @CHECK:

```
@CHECK NCP
```

may be called by a user to determine whether or not the NCP is in the System. The information is returned in the condition code.

The @ID macro instruction returns the identifier of the local socket currently associated with a specified ECB:

@ID ECB=address, WS=address

@IDH returns information about a specified host. The calling sequence is:

@IDH WS=address

The workspace contains fields for both a host address and a character-string mnemonic for that host; the user specifies one and the NCP the other. A field containing the status of the host is included in the workspace, and is always filled by the NCP.

A macro instruction called @XCTL is provided to allow the user to change several parameters associated with a socket:

@XCTL ECB=address [, NEWECB=address]
[, NEWWS=address] [, NEWSP=subpool]

Finally,

@DISCARD [SP=subpool]

closes all local sockets in the designated subpool. If SP is unspecified, all sockets opened by the invoking task are closed.

(2) Fortran and PL/1

Sets of subroutines have been provided to enable Fortran and PL/1 programmers to communicate with the NCP. In general, a separate subroutine is provided for each function described in the preceding section; the subroutine contains the assembly-language instructions required to interpret the Fortran / PL/1 parameter list and issue the appropriate Network macro

instruction. The additional function of timing the operation and aborting it if not completed in a specified interval is also performed by the subroutines.

The calling sequences for the subroutines currently provided are briefly described in the following paragraphs. Complete documentation has been generated and distributed to Network participants in the form of two NWG Requests for Comments (RFC's) entitled, respectively, "Network Fortran Subprograms" and "Network PL1 Subprograms" (21 April 71, NIC 5831 and 5832). The names of PL/1 subroutines are prefixed with the character "@". The arguments in the calling sequences directly parallel those in the corresponding assembly-language macro instructions, with the exceptions that they are Fortran / PL/1 variables, constants, or expressions, rather than main storage addresses, and that a time interval is included in several cases. Completion-code's are disguised ECB's.

Connect, Listen, and Accept are invoked as follows:

CALL [@] OPEN (completion-code, time [, local-socket-identifier [, foreign-socket-identifier] , workspace])

A socket is closed with the following call:

CALL [@]CLOSE (completion-code, time)

Data is transmitted through a send socket with a subroutine call of the form:

CALL [@]WRITE (completion-code, buffer, length, time [, offset])

Data is received through a receive socket with the call:

CALL [@]READ (completion-code, buffer, length, time [, offset])

The status of a local socket is obtained by invoking:

CALL [@]CHECK (local-socket-identifier, socket-status, status-mnemonic, foreign-socket-identifier, bit-count)

At the assembly-language level, the information returned by the NCP is placed in a single workspace; here, however, each piece of information is placed in a separate Fortran / PL/1 variable.

Identifying the local socket associated with (at this level) a specified completion code variable is accomplished by invoking the following:

CALL [@] ID (completion-code-variable, local-socket-identifier)

An interrupt is transmitted as follows:

CALL [@]SIGNAL (completion-code, time)

A set of four subroutines in each language is provided to aid in applications involving communication between a Fortran / PL/1 program and an OLS terminal. Their use is not inherently restricted to that application, but their design is oriented toward it. Each operates in conjunction with a corresponding OLS operator (all of which are described in the section following this), with the Fortran / PL/1 program and OLS terminal functioning as co-routines.

A subroutine called Write to On-Line Console (WTOLC) transmits data, assumed to be a character string, for display on what is assumed to be an OLS display device. The calling sequence is:

CALL [@]WTOLC (completion-code, buffer, length, case, x-coordinate, y-coordinate, time)

The subroutine prefixes the user's character string with the OLS carriage control characters required to place the text at the indicated position within the display area. The user may specify either absolute or relative (to the last display) coordinates. "Case" permits the display of Greek (case 2) or user-defined, special characters.

A subroutine called Read from On-Line Console (RFOLC) is the counterpart of WTOLC and is applied to a receive socket. It accepts what is assumed to be a character string entered by an OLS user at his keyboard. The calling sequence is:

CALL [@]RFOLC (completion-code, buffer, length, time)

A third subroutine called Write to On-Line Console with Reply (WTOLCR) combines the functions of the previous two by causing both a message to be displayed at the user's terminal, and a response to be accepted from him. This subroutine is applied to a pair of sockets of opposite gender. The calling sequence is:

CALL [@] WTOLCR (send-completion-code, send-buffer, length, case, x-coordinate, y-coordinate, time, receive-completion-code, receive-buffer, length)

A final subroutine causes an OLS display device to be erased. Its calling sequence is:

CALL [@] ERASE (completion-code, delay-before-erase, delay-following-erase, time)

A number of the Fortran routines described above have been used by programmers at the MITRE Corporation, McLean, Virginia, in producing a Network file system which is to execute in UCSB's 360/75.

(3) On-Line System

A subsystem of OLS called NET has been created to house, among other things, a set of operators providing access to basic, NCP functions from an OLS terminal. These operators were heavily used in early, interactive Network experimentation with (in particular) the Rand Corporation in Santa Monica. All of these operators have been described in detail in an RFC entitled

"Network On-Line Operators" (21 April 71, NIC 5833) distributed to Network participants through the NIC. They are briefly described in the following paragraphs.

All of the following operators exist on Level I Real. "Q" designates a Level 0 storage location to which a socket is attached and in which the completion codes for operations involving that socket are made available to the user.

The UP operator is employed to issue a Connect, Listen, or Accept. It requires a predicate list as indicated below:

UP Q [local-socket [, foreign-socket [, host-address]]] RETURN

If both local and foreign socket numbers are specified, the operation requested is Connect ("host-address" defaults to "3"); if only a local socket number is specified, the operation is Listen; and if neither is specified, it is Accept.

The sequence for closing an existing connection, retracting a connection request, or rejecting a connection attempt is:

DWN Q

The STORE operator causes data to be transmitted through a send socket. The predicate list is:

STORE { [+/-]Q delimiter text delimiter
. Q [X] value RETURN }

where braces indicate a choice between several forms. The first form causes "text" to be transmitted as a character string. The optional "+" and "-" control the formatting of the text into logical messages. The second form causes a single byte whose value is "value" (specified in hexadecimal if "X" is present) to be transmitted.

LOAD is the counterpart of STORE and causes data to be accepted from a receive socket. Its format is:

$$\underline{\text{LOAD}} \left\{ \begin{array}{l} [+]\ Q \\ -\ Q\ \text{op-code, length}\ \underline{\text{RETURN}} \\ .\ Q\ \text{number-of-bytes}\ \underline{\text{RETURN}} \end{array} \right\}$$

The first form causes logical messages to be accepted from the socket, each preceded by an op code and length. The op code identifies the nature of the data (e.g., characters for display, keys for execution). The second form allows the user to manually specify the op code and length "n" to be applied to the next "n" bytes received over the connection. The third forms permits up to four bytes of data to be accepted and placed in the Level 0 accumulator.

The DISPLAY operator enables the user to obtain information about any local socket. It is invoked in the following manner.

$$\underline{\text{DISPLAY}} \left\{ \begin{array}{l} (\text{socket-number}\ \left\{ \begin{array}{c} \underline{\text{RETURN}} \\ \underline{\text{BACK}} \end{array} \right\} \ldots) \ldots \\ \underline{\text{RETURN}} \ldots \end{array} \right\}$$

where ellipses designate items that may be repeated any desired number of times. If the predicate list is simply RETURN, the status of all sockets attached by the user's terminal are displayed. Otherwise, information about the designated socket is displayed; successive RETURN's display successively higher-numbered sockets; successive BACK's display successively lower-numbered sockets.

The ID operator displays the number of the socket attached to a designated Level 0 storage location. It's calling sequence is:

ID Q...

DEL closes all sockets attached by the user's terminal.

REFL sends an interrupt to the foreign socket attached to the designated local socket. The key sequence:

REFL Q

transmits the interrupt.

Finally, the operators ARG and ATAN provide a convenient means for conducting bidirectional communication with another terminal or with an executing program. The calling sequences are:

ARG [,] [-] Q_1 [-] Q_2 and

ATAN delimiter [,] [-] Q_1 [-] Q_2

Both operators are applied to a pair of local sockets (designated by Q_1 and Q_2) of opposite gender. Both simultaneously monitor the receive socket and user keyboard displaying incoming data from the Network connection on the user's display device, and transmitting the user's keypushes through the send socket. ARG provides a character-at-a-time mode without local echo, and ATAN a line-at-a-time mode (with "delimiter" signalling end of line) with local echo.

(4) User Telnet

A User Telnet has been implemented within OLS to provide OLS users with access to teletype-like time-sharing systems in the Network. The program conforms to the protocol adopted by the NWG to handle such communication. UCSB's User telnet implementation is described in detail in an RFC entitled "A User Telnet - Description of an Initial Implementation" (9 August 71, NIC 7176) distributed to Network participants through the NIC. A brief description follows in succeeding paragraphs.

User Telnet resides in the NET subsystem of OLS on Level II Real under the operator LOG. The sequence for initiating a duplex Network connection to a known system is as follows:

KEYBOARD ENTRY	OLS QUERY/RESPONSE
II LOG	FOREIGN SITE NO. = (host-address)
host-address RETURN	FOREIGN SCK NO. = (socket-number)
socket-number RETURN	

"Host-address" and "socket-number" designate the system to be contacted. When the desired connection has been established, OLS will erase the user's display device and position the user to the upper left corner of the screen. From then on out, the user is effectively connected to the remote system. Keys pushed by the user are mapped into a character set (7-bit ASCII) adopted as standard by the NWG. Incoming text from the connected, remote system is mapped from that character set into a form suitable for display at the user's terminal.

The mapping adopted between ASCII and teleputer code is briefly as follows. CASE on an OLS keyboard substitutes for the CNTRL key on a teletype; hence, for example, "control-C" is struck by pressing "CASE C". An alternate means for sending control characters is provided on the upper keyboard; for example, LOG is equivalent to "CASE C". Punctuation, numerics, and upper-and lower-case alphabetics are mapped straight-forwardly into their ASCII equivalents. RETURN and BACK are mapped into carriage return and rubout, respectively. Back slash is mapped into "CASE /", left brace into "CASE ("; similar transformations are applied to a number of miscellaneous punctuation characters. Bel is simulated by displaying "<BELL>" in the upper left corner of the screen.

As a substitute for the noisy return of a teletype carriage (often a useful cue), OLS displays an underscore in the left margin of the display area.

Provision has been made for allowing the user to escape from User Telnet (by hitting a level key) and then to return to the connected remote system without penalty. Thus the user can, for example, escape from telnet and select the printer as a secondary output device, and then return to Telnet to obtain hardcopy of all succeeding dialogue with the remote system.

The key sequence "CCN RETURN" closes the user's Telnet sockets. If the sockets haven't been closed by the user before he logs off, that function is performed for him by OLS.

As an adjunct to Telnet, the user is provided with an operator for displaying the list of hosts known to the NCP. For each such host, the following information is displayed: a brief mnemonic for the host (e.g., "UTAH", "RAND", "UCSB"); its host address in decimal, hexadecimal, and octal; and the state of the host (dead to the Network or otherwise). The list is generated by pushing:

ID RETURN...

Each successive RETURN erases the display device, repositions it to the upper left corner, and displays another segment of the list, until the last hosts have been displayed.

(c) Access to Local Programs From the Network

A number of programs were written during this contract period for use by users at other Network sites. Some are interfaces to existing software systems at UCSB; others are service programs written expressly for Network users. These programs are described in the following sections.

All currently existing programs of this type are executed in the same region as the NCP. A subroutine of the NCP called the Dispatcher was written to oversee the execution of such service programs - fetch copies of them from secondary storage when requested by remote users or processes, and release the main storage they occupied when they terminate execution. Essentially, the Dispatcher is aware of a set of local sockets for which it has responsibility. With each such socket, it associates the name of a partitioned data set (PDS) and the name of a member of that PDS containing the load module for the program associated with the socket. The Dispatcher always has a Listen pending to each socket in its list. When a foreign process attempts to connect to the socket, the Dispatcher invokes the appropriate program by fetching its load module from the PDS and attaching it as a subtask, and then passes control of the socket to it. While the program is attached, the Dispatcher ceases listening on that socket; as soon as the program terminates execution, the Dispatcher re-issues its Listen. In this manner, any number of processes can be supported; a single-card change to the software adds to or deletes a socket (and hence a service) from the Dispatcher's list. The Dispatcher itself is transparent to the calling, foreign socket.

(1) On-Line System, Graphics Supported

Two processes were written during this contract period to provide Network users with full access to the On-Line System. Both included provision for transmitting curvilinear display in line, dot, dot-dot, and character modes, and the first for transmitting special character display. The first such implementation was made available to Network users shortly after the adoption by the NWG of its initial Host-Host protocol specification, and was

never used by any but local users. It was fully documented in an RFC entitled "Specifications for Network Use of the UCSB On-Line System (16 October 70, NIC 5417) distributed to Network participants. Support for this specification was terminated by UCSB this August, and replaced by generation of a new specification to which the Rand Corporation in Santa Monica has already interfaced its video-graphics system. Hence, users at Rand are able to simulate an OLS keyboard by using a light pen or Rand tablet to select squares on a pick table displayed on a television screen. An area of the screen is reserved as a simulated OLS display screen. This second implementation does not yet support special character display, but the inclusion of that feature is scheduled for the immediate future. Also, documentation of this second implementation has yet to be made available to the Network community.

A brief description of UCSB's first implementation of an interface to its On-Line System is now presented. This implementation simultaneously supported up to ten Network users. A user connected himself to OLS by a hand-shaking procedure almost identical to that which has since been adopted by the NWG as the official Initial Connection Protocol (ICP). Keypushes, represented in OLS's internal character set, were transmitted by the user on one Network connection. Software in the user's host was responsible for providing him with a means of generating those codes from his terminal. Character, curvilinear, and special-character output were returned by OLS on a second connection in the form of variable-length logical messages, each prefixed with op code and length fields. An op code was defined for each of the three classes of output. Text was transmitted in the Extended-Binary-Coded-Decimal Interchange Code (EBCDIC), with carriage control characters

embedded in the text. Curvilinear display was transmitted as lists of normalized, fixed-point x-y coordinate pairs, each coordinate in the range [0, 4095] and represented in sixteen bits. Flags indicating the display mode, and the display character if character mode, were also made available. Finally, special character display was represented in the following manner. The display area was assumed 4096 units square, and a character frame 160 units wide and 224 units high. Codes were defined for moving the beam in any of sixteen directions a distance equal to that of a move in the specified direction from the center of a 64-unit square to its perimeter. A change in position relative to the lower-right corner of the last character frame was communicated by providing the change along the x and y axes as two 12-bit signed integers.

In that implementation, the user was interfaced to OLS at natural points in the software. Input from Network users was placed in a buffer, as if its source were a segment of the hardware interface to local users, and OLS posted via the OS post macro instruction. Output from OLS was intercepted between the terminal display subroutine of the Basic system and the I/O module. In OLS terminology, the Network software played the role of an Output Enqueue routine.

The second implementation of a graphics interface was the product of a joint experiment with Rand Corporation which was initiated in response to requests made for pairwise experiments at the Network Graphics meeting held in Boston. This implementation was carried out by establishing a set of graphic orders and writing a display encoder which builds strings of orders representing the on-line display. The set of orders are:

1)	<ERASE>	Erases the screen.
2)	<origin> <x,y>	Positions the beam at location <x,y>
3)	<vector> <x,y>	Draws a vector from current beam position to the point <x,y>
4)	<Dot> <x,y>	Positions beam at location <x,y> and displays a small dot.
5)	<shift> <Size>	Shifts character size to that specified.

All orders are 8 bits in length. Coordinate information is presented as 16 bit pairs in unsigned binary fraction format, (ie., $0 \leq x < 1$, $0 \leq y < 1$). It is assumed that characters can be displayed at any location on the display screen and the origin is in the center of the character frame. Any number of character sizes up to 256 is supported. All carriage control characters are trapped before transmission and replaced with origin commands. Beam location is recorded at our end of the connection and hence any sequence of consecutive carriage control characters can be replaced with one origin command. Any character in the output string other than those in the set of orders is assumed to be a printable character and only those characters in the 96-character ASCII set are transmitted. EBCDIC is used in place of ASCII code for convenience since both Rand and UCSB have IBM systems.

(2) Batch Processing

UCSB's batch processing facilities were made available to Network users during this contract period. Two independent processes were written to provide this service, each addressed by a standard ICP to a separate socket. One supports remote job entry (RJE) by accepting files of card images from the Network and transmitting them to UCSB's Houston Automatic Spooling Priority System (HASP) through an internal reader. The second retrieves output from remotely submitted jobs from a PDS and relays it to the Network user. These

facilities have been in use on a production basis by user groups at the Rand Corporation in Santa Monica for a number of months, and are fully documented in an RFC entitled "Network Specifications for Remote Job Entry and Remote Job Output Retrieval at UCSB" (22 March 71, NIC 5775) distributed to Network participants through the NIC. Brief descriptions are included below.

Complete card files are transmitted to UCSB through the Network, and are required to include all the usual Job Control Language (JCL) cards. In addition, to enable subsequent retrieval of what would normally be printed output, a special accounting parameter must appear on the user's job card. UCSB provides the Network user with three options for formatting his card file. The first option enables the user to transmit his file as a succession of variable length card images, each delimited by a break character of his choosing. The second option allows each card image to be preceded by a 16-bit length field. In either case, UCSB pads each record on the right or truncates it to 80 characters wherever required. The third format requires that a file be transmitted as 80-character, fixed-length records.

In retrieving output from RJE jobs, the user is provided with the following options. The user can request that the output file generated by a job be returned if currently available, or that it be transmitted as soon as it is available. He may request that a previously-made output request be cancelled. Finally, he may request that an output file be purged. In all cases, UCSB returns a completion code indicating the disposition of the user's request, accompanied by the name of the job to which the request applied. The output file itself is always transmitted as a series of variable-length logical records, each preceded by an op code and length field.

UCSB forces serial access to both of the processes described above, one of which uses an inherently serial resource - an internal reader. The greatest shortcoming of this service lies in the difficulty of providing the user with status information about his job in the presence of HASP.

(3) File Storage

During this contract period, on-line direct-access storage at UCSB was made available to Network users through a process written expressly for Network use and known as UCSB's Simple-Minded File System (SMFS). This process has been fully documented in an RFC entitled "Network Specifications for UCSB's Simple-Minded File System" (26 April 71, NIC 5834) distributed to the Network community through the NIC. A brief description of the system follows in succeeding paragraphs.

At present, one IBM 2316 disk pack with a capacity of 29M 8-bit characters and one 2314 disk drive have been dedicated to Network use. SMFS is written to support any number of on-line, dedicated drives without modification. Any number of Network users, up to some assembly-parameter maximum (currently ten) can be serviced simultaneously by SMFS.

SMFS provides storage for sequential, binary files, each characterized by a 36-character filename and two, optional, 36-character passwords, one regulating retrieval of the file, the other regulating its modification or deletion. SMFS permits filenames and passwords to be specified by the user in either EBCDIC or ASCII. Each file is stored by SMFS as a single OS data set. The more important file operations supported are described below.

SMFS regards the reservation of filename, the assignment of passwords, and the reservation of secondary storage as an operation distinct from that

of transmitting the file's contents. The operation is called file allocation and requires, in addition to the specification of filename and passwords, that an upper bound on the file's size be declared.

An operation called file update causes a bit string transmitted by the user to be logically concatenated to the current contents of the file. Boundaries between successive update operations are transparent to the user when the file is retrieved. A similar operation called file replacement causes the transmitted bit string to replace the entire contents of the file

An operation called file retrieval causes a segment of the file of specified length to be returned to the user. Successive retrieve operations return successive segments of the file. A similar operation called file space causes advancement through the file without transmission of data.

An operation called file deletion purges a file, releasing the filename and permitting SMFS to reclaim the secondary storage allocated to the file.

Finally, an operation called file rename permits re-assignment of filename.

(4) On-Line System, Server Telnet

Finally, during this contract period, an interface to the On-Line System was created to function within the constraints of the Telnet protocol adopted by the NWG to handle teletype-like communication. The nature of this protocol requires that curvilinear and special-character display be discarded, and hence only character display is made available to the user. However, no other limitation is imposed upon use of OLS through this Telnet implementation (of course, that one limitation is in itself severe). In particular, a means is provided for simulating every key on an OLS keyboard. This process is addressed as socket number 1 in accordance with Telnet protocol, and supports

simultaneously any number of Network users, up to some assembly-parameter maximum (currently five). The specifications for this service are yet to be published; a brief description follows:

OLS keyboards have more keys than there are characters in Telnet's 7-bit ASCII character set. A fairly natural correspondence can be established between elements of the character set and keys on the lower or operand keyboard. Keys on the upper or operator keyboard are represented in the following manner. Each is designated by a character string suggestive of the key's function, preceded by a semi-colon and followed by a space. For example, the SYST key is designated ";SYSTb". The character string may be specified in either upper- or lower-case and may be abbreviated to any extent desired, as long as the key indicated remains unambiguous. Hence, ";SYb" is an adequate representation of SYST. A semi-colon is represented as ";;". If a semi-colon appears anywhere within a character string representing an upper keyboard key (after at least one other character), all previous characters (excluding the semi-colon) are ignored; typing errors can thus be corrected. A carriage return anywhere within such a string has the same effect as a second semi-colon, but the initial semi-colon, too, is ignored; hence the next character entered is treated as designating a lower-keyboard key. As it becomes clear to OLS that the character string being entered does not designate a key on the upper keyboard, question marks are echoed and the erroneous characters ignored. Hitting Alt Mode or "?" anywhere within the string causes the remaining characters of the keyword to be echoed, provided the keyword is uniquely identified; otherwise, the bell is rung.

A number of pseudo-upper-keyboard keys are defined, and all of the conventions for their entry are as described in the preceding paragraph.

One such pseudo operator - "SHIFT" - causes the case of all alphabetics to be changed; "A" is translated to "a" and "a" to "A", etc., before being processed by OLS. The pseudo-operator "UNSHIFT" nullifies the effect of "SHIFT". The pseudo operators "HALFDUPLEX" and "FULLDUPLEX" disable and enable, respectively, echoing of characters entered by the user. The pseudo-operator "PREFIX" causes the character following it to become the prefix character, replacing ";". The pseudo-operator "STATE" causes the current prefix, the console mode (half- or full-duplex), and the shift state (on or off) to be displayed. Finally, the pseudo-operator "HELP" causes the entire set of operator and pseudo-operator keywords to be listed for the user.

(d) Network Pie-In-The-Sky

A number of Network-related projects are scheduled for the near future. A Data Reconfiguration Service (DRS) is currently being implemented in cooperation with the Rand Corporation. A DRS is planned for implementation on several hosts, each such host being one with substantial computing power. Essentially, the DRS is a Network process which services terminal users, communicating with them via the Telnet protocol. The DRS permits a user to define in a simple programming language, transformations to be applied to the data passing through a pair of Network connections. Character set translations, insertion of literals, and transposition of fields are among the more simple operations expressable in the language. The user is permitted to name, store, retrieve, and apply to specified connections, the programs so defined. The intent is to provide users at hosts with small amounts of computing power (Terminal Interface Processors (TIP's) are important examples of such hosts)

with a means for communicating with those specialized Network services which do not conform to one of the standard protocols supported by the users' host software.

The DRS being implemented by UCSB and Rand has three components: a compiler which reduces DRS source programs to a simpler, intermediate language, an interpreter which executes the program output by the compiler, and a front-end which interfaces to the user. The first component is being written by Rand in PL/1, the second and third by UCSB in assembly language.

UCSB's is also investigating the possibility of implementing a full-blown remote job entry service to replace the rudimentary service currently provided. Such an implementation requires the cooperation of HASP, and hence the software is more involved.

Of course, protocol changes which affect the NCP will be implemented as required. And, as higher-level protocols are adopted to standardize such services as remote job entry and file storage, existing UCSB software will be modified accordingly.

II. UCSB On-Line System

Nearly all on-line system software development was completed during the first half of the contract year as stated in the last technical report. Many features of lesser importance were implemented during the second half along with two major additions which are described below.

It has become apparent from recent conversation with interested parties that exportation of the UCSB On-Line System is constrained by two major characteristics. First, the On-Line System can, in some respects, be considered a "core hog". Whereas many installations have a main-frame

sufficient in capacity to implement our system, most are not in a position to allocate a main storage region of the magnitude necessary to support a user population equivalent to ours. To do this would mean curtailing other user services which an installation has strongly committed itself to provide. Hence, we have implemented a swapping facility which should drastically reduce the main storage requirements for a given number of users. We anticipated well in advance that this may someday become necessary and tock steps to provide for its later addition. Therefore the implementation of rollout was relatively straight forward and was accomplished in a short period of time. The On-Line System is basically a core-resident system. When a user begins a session, his data resides on a direct access volume. As he makes references to certain elements of his data base, those data elements are brought into main storage where they remain until he invokes routines to cause their return to disc. However, when contention for main storage reaches a predetermined threshold, the user occupying the most main storage is swapped back to disc and the cycle repeats itself. At this writing adequate measurements have not been made which would indicate to what extent response time is affected under certain conditions. However, experiments have been conducted using a 160K work space shared among 12 users operating under worst-case conditions for which response time was not affected by an intollerable amount.

The second modification to the On-Line System concerns the manner in which it time shares. The question often asked was how well would the UCSB On-Line System cooperate in an environment consisting of many high-priority interactive computing systems -- the answer being "not very well". This has

never been a major consideration locally since we are the sole resident of an interactive community. However, it has become quite obvious that the situation must be alleviated if the On-Line System is to be truly exportable.

We have proceeded on the premise that interactive systems should police themselves; ie., such systems should guarantee that they will not use more than a given percentage of the central processor over a small increment of time. Further, this increment must be small enough to allow all interactive systems within the community to respond quickly to simple user requests. In actuality, this is the only reasonable approach since it is very difficult to police the activity of other interactive systems running under O.S. with no knowledge of their mechanism of time sharing, particularly if they have a higher priority. Hence, the time sharing monitors were changed to measure CPU utilization during back-logged states and to assure that this utilization does not exceed a given threshold. This threshold may be set by the operator or allowed to vary according to a pre-established profile. Thus, the percentage utilization may be allowed to increase during certain hours of the day when other computing activity is light.

The On-Line System time shares with other processes by alternating its dispatching priority between its normal value (high) and zero (lowest possible priority). Setting the dispatching priority to zero has the same effect as going into the wait state in that lower priority processes may get CPU cycles. However, there is the added advantage that if no other lower priority tasks are contending for the CPU, the On-Line System will continue executing (ie., it is the only low priority process contending for processor time). The intervals of time during which the On-Line System executes in a high or

low priority state are computed as follows: Normally OLS takes a 100 millisecond time slice (task time) in the high priority state. If P denotes the maximum percentage CPU utilization for OLS and H denotes the high priority time slice, then the interval of time (L) for which OLS executes at low priority is computed as $L = \frac{(1-P)\ H}{P}$. However this formula does not take into account cycles lost to higher priority processes. To allow for this, we must subtract that amount lost from the interval we would normally execute at low priortiy. In more precise terms the algorithm may be stated as follows: Let one full cycle be defined as one interval in low priority (Ln) followed by one interval in high priority (Hn), H_o be the normal high priority time slice (100 ms), and E_n be the elapsed real time of the nth cycle. We now define

$$L_{n+1} = \frac{(1-P)H_n}{P} + Ln - En$$

If Ln+1 < 0 then Hn is extended by $\frac{P\ L_{n+1}}{1-P}$ or:

$$Hn = Hn - \left(\frac{P}{1-P}\right) Ln+1$$

$$= \frac{P\ (En-Ln)}{1-P}$$

Otherwise, if Ln+1 ≥ 0, then Hn+1 is set to Ho.

While changing the time-sharing monitors it was also a convenient time to upgrade the internal scheduling. Hence, we have replaced the previous "round robin" scheduler with a priority scheduler. This was done by simply allowing a hierarchy of "round robin" queues which are classified as follows:

Q1 - special system functions

Q2 - simple user requests

Q3 - user programs (foreground mode)

Q4 - User programs (background mode)

The relationship of priorities with batch programs is expressed as

$P(Q1) > P(Q2) > P(Q3) > P(BATCH) > P(Q4) = 0$

The significance of executing user programs in background mode is that such programs get processor time only when all other computer activity has quiesed. However, users running in this mode will not be charged for processor time. This should encourage users to run at times when computing activity is normally light.

The two preceding additions are still in the implementation phase and should become part of the prime-time system within the next month. Most development effort during the ensuing contract year will be directed towards network activities. At this writing, the only planned addition to OLS will be BUTTRAN (BUTTON FORTRAN), a mechanism for including user Fortran subroutines as OLS basic operators.

B. 360/75 On-Line System, Hardware

Maintenance and functional improvement of the UCSB IMP-HOST Interface has continued over the last year. Design improvements have been incorporated in a newly designed I.C. version of the interface for 360 operation. Several ARPA sites are interested in using the new interface, one of which is presently operational at MIT Lincoln Laboratory.

Hardware assistance in support of the UCSB On-Line System (OLS) has resulted in the development of a new Multi-Line Controller and a new version of the graphics display consol^ used on the OLS. The Multi-Line Controller (MLC) will allow the attachment of any type of user equipment to our 360 system. The MLC may find use at several ARPA sites in the near future. The new graphics terminal will allow a user to gain access to the OLS and to the ARPA Network at large by means of an acoustic coupler. Design of the new console has incorporated the newest in storage oscilloscopes and keyboards, and now provides the user with a most reliable, inexpensive, graphics (and alphanumeric) console.

The high-speed serial data link between the UCSB Network Host (IBM 360) and the speech analysis computer (SEL-810) has been designed and all purchasing has been completed. This link will be made operational in the next quarter, and will provide the hardware to link our speech system to the network. Upon completion of software development our speech system could support network users. The link will transfer 500 Kilobit/sec serial data between the two sites and will allow real-time storage of voice data at the 360.

C. Speech Project

The speech research effort at UCSB to date has been directed toward the following areas:

1. Development of a preprocessing method to filter the raw-speech string into sub-strings amenable to wave function analysis.

2. Development of a one pass wave-function analysis/synthesis system based on the cosine modulation methods that will accurately analyze and synthesize both male and female speech data.

3. Continued studies on the computer classification and recognition of phonetic information including extraction of recognition parameters from the ASCØF parameter set, single speaker recognition of (a) steady state vowels, (b) vowels embedded between two unvoiced phonemes, and (c) unvoiced phonemes, and detailed studies of the segmentation of connected voiced and unvoiced phonemes.

4. Studies of the data rate of the basic ASCØF representation and the amount of data compression possible through (a) the elimination of redundant wave function sets, (b) coarse quantization of parameter values, and (c) encoding of parameter sets.

The above topics are discussed in more detail in the subsequent paragraphs.

(a) Preprocessing of the Acoustic Waveform

As was discussed in the Semiannual Technical Report[30]* the acoustic waveform must be preprocessed in some appropriate manner prior to the analysis of the waveform into its set of wave function parameter elements. The preprocessing operation must convert the raw human speech into a form that is

*Superscripts refer to "List of Publications and Reports" at the beginning of this report.

amenable to wave function analysis and that minimizes the data rate into the wave function analysis procedure.

The above requirements can be satisfied by correct prefiltering of the acoustic waveform into contiguous bandpass frequency bands of filtered data and conversion of the filtered data to an extrema format which retains the minimum information necessary to describe the filtered human speech.

Efficient algorithms for these preprocessing steps have been developed and implemented on the IBM-1800 speech system. In addition these algorithms are presently being implemented as part of the IBM 360/75 network speech facility.

Octave Band Filtering

The filtering algorithm which has been developed[10] is a high speed octave band, time domain convolations method of filtering which provides optimum filtering characteristics and minimum data rates to the wave-function analysis system. The method is essentially a fast non-recursive digital filtering technique in which over three-fourths of the coefficients of the filter impulse response are forced to be zero by the judicious choice of filter center frequency, band width, and window. Nearly half of the remaining coefficients of the filter kernel can be discarded by taking advantage of the symmetry of the impulse response.

The filter kernel r(nT) which is employed in the filtering process is given by

$$r(nt) = B\, w(nT)\, \frac{\sin\frac{\pi n}{4}}{\frac{\pi n}{4}} \cos\frac{\pi n}{2} = -N \leq n \leq N \qquad (1)$$

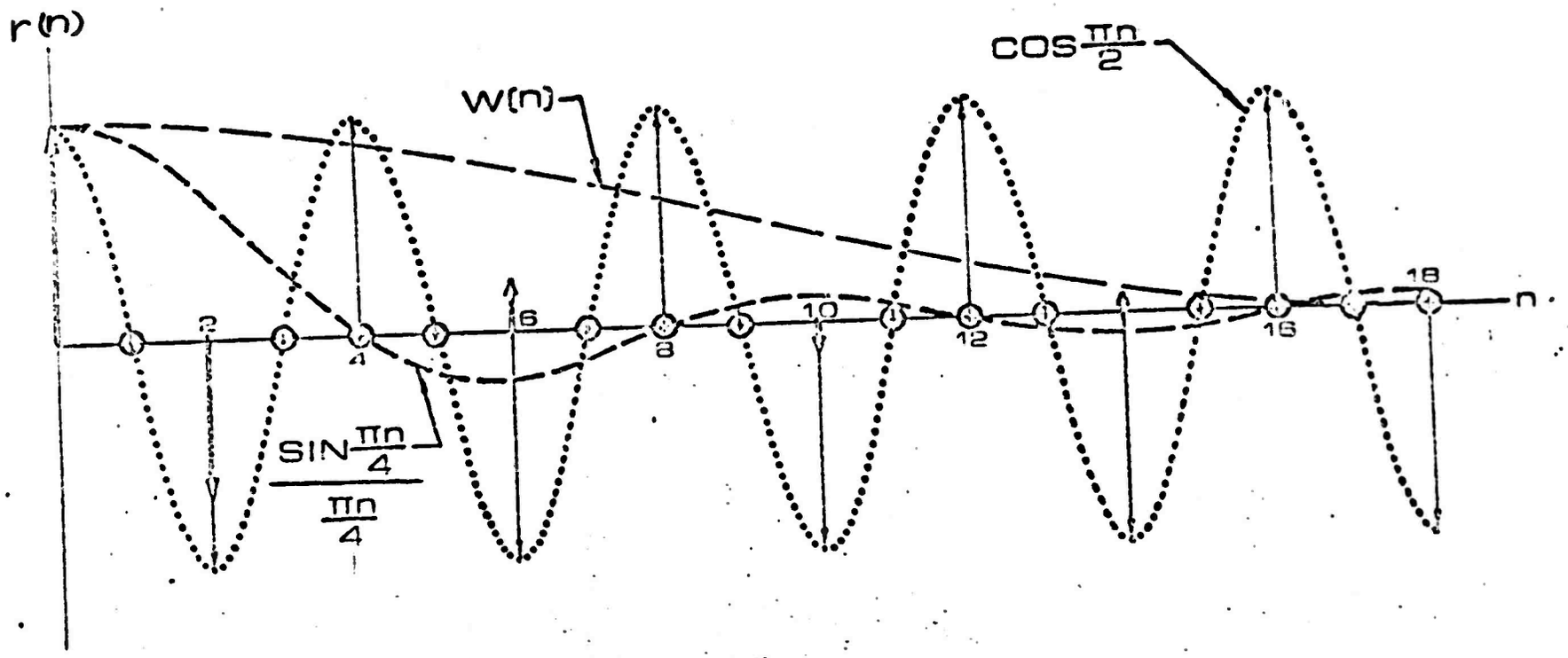


FIGURE 1 "Minimum Coefficients" Kernel for N = 18. There are five non-zero coefficients to represent a 37 coefficient impulse response.

Note that Eq. (1) represents an ideal bandpass filter modulated by a window function w(nT). B is a gain normalization factor and w(nT) a window function described by a Gaussian times a Hanning window as shown in Eq. (2)

$$w(nT) = \left[\exp\left[-\tfrac{1}{2}\left(\frac{knT}{\tau}\right)^2\right]\right]\left[1+\cos\frac{\pi nT}{\tau}\right]\cdot\tfrac{1}{2} \qquad k\geq 0 \tag{2}$$

The parameter τ in Eq. (2) is the time width of the window. In order that the end values of the window coincide with sample points a value of τ=NT is employed. The resulting window is independent of the sampling interval T as indicated in Eq. (3).

$$w(n) = \exp\left[-\tfrac{1}{2}\left(\frac{kn}{N}\right)^2\right]\left[1+\cos\left(\frac{\pi n}{N}\right)\right] \quad k\geq 0 \tag{3}$$

When N=2+4i (i=0,1,2,. . .) the zero end-points of the window will coincide with non-zero values of the kernel. This maximizes the effective time width of the window while further reducing the number of non-zero coefficients required to represent the kernel.

Figure 1 shows the "minimum coefficients" kernel r(nT) for N=18. Note that only five non-zero coefficients, indicated by the heavy arrows in Figure 1, are required to represent a 37 coefficient impluse response.

If a signal sampled at a sampling frequency f_s is bandlimited at $f_m = f_s/4$ the resulting data string [x(j)] can be filtered into octave bands using a single minimum-coefficients kernel. The steps required to accomplish this (illustrated as magnitude frequency spectra in Figure 2) are as follows:

a. Perform a minimum-coefficients convolution on the input data string $[x(j)]$. The output sequence $[y(j)]$ is an octave band. (Figure 2c)

b. If the next lower octave band is desired, go to (c), otherwise terminate.

c. Subtract the filter output $[y(j)]$ point-by-point from the original data sequence to obtain the "residue data", which is bandlimited at $f_m \leftarrow f_m/2$. (Figure 2d)

d. Decimate the "residue data" by throwing out every other point. This new sequence $[x'(j)]$ has half as many elements as the "original data" and is effectively sampled at $f_s \leftarrow f_s/2$.

e. Replace $[x(j)]$ with $[x'(j)]$ and go to (a).

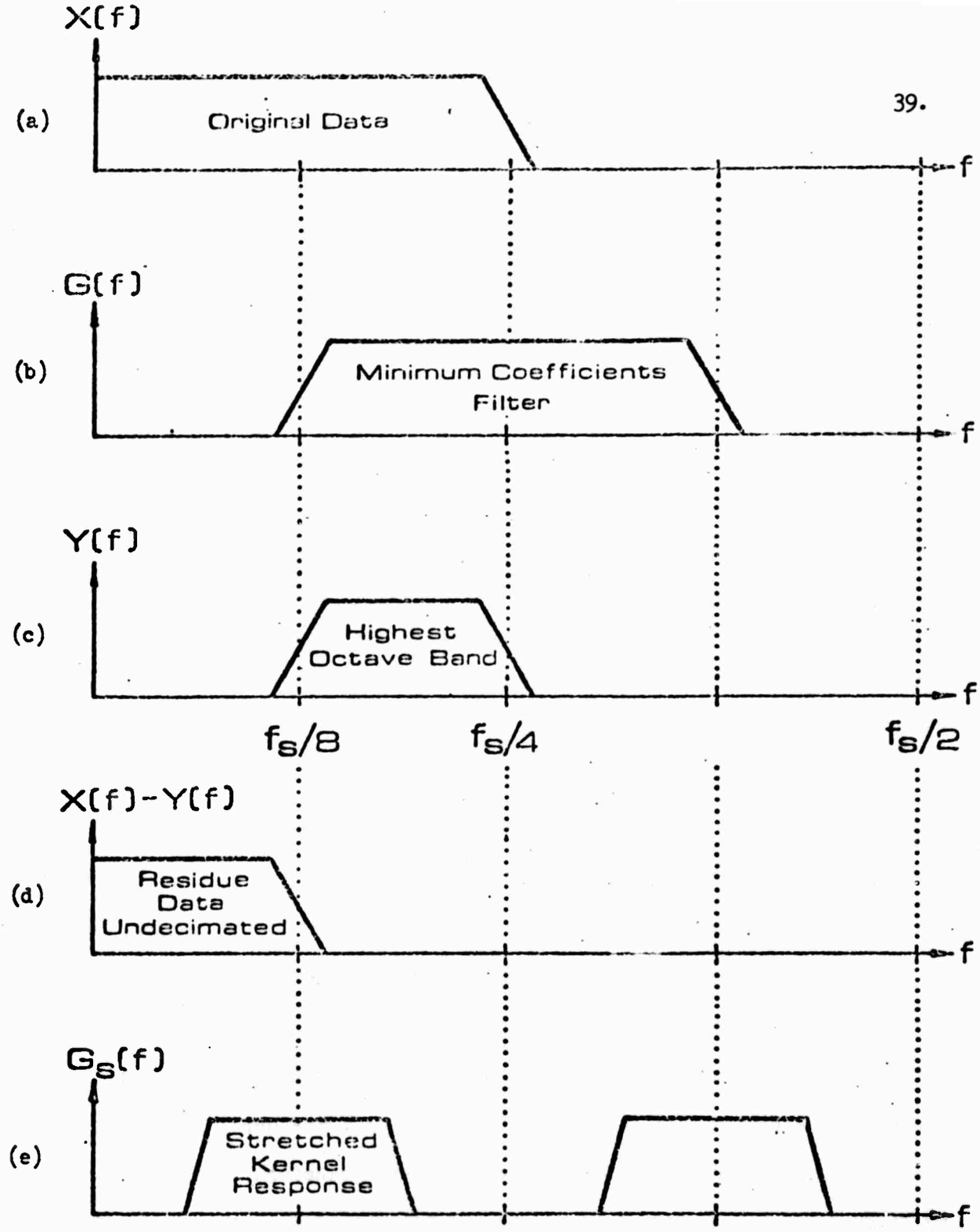
One advantage of the above method is that the time required to filter out each successive octave band is one half the time of the previous band. (See (d) above). The total time to filter the "original data" into M bands is thus

$$t_M = t_h \left[1 + \tfrac{1}{2} + \tfrac{1}{4} + \ldots + (\tfrac{1}{2})^{M-1}\right]$$
$$= 2t_h \left[1 - \frac{1}{2^M}\right] \qquad (4)$$

where t_h is the time to filter out the highest octave band. However two possible limitations of the approach are:

a. The data strings for successive bands are <u>not</u> the same length and hence cannot be directly summed.

b. All bands are effectively sampled at twice the Nyguist frequency which may not provide sufficient data resolution.



NOT REPRODUCIBLE

FIGURE 2 (a) Original data bandlimited at $f_s/4$.
(b) Response of an M-C filter.
(c) Result of filtering original data with an M-C filter.
(d) Residue data resulting from subtraction of (c) from (a) before decimation.
(e) Response of an SMC filter stretched once, i.e. K = 1. Note that the unwanted spectral repetition occurs where the undecimated residue-data is essentially zero.

These problems can be obviated by employing a "stretched" minimum coefficients kernel coupled with data decimation. In a "stretched" kernel all non-zero values are unchanged, but are located as if Figure 1 were printed on a rubber sheet and then stretched uniformly to the right until the end point, r(N), is located as though $N \leftarrow 2^K N$ (K=0,1,2,3). Thus if all octave bands must have the same number of elements as the original data a stretched minimum coefficient kernel with K=0 is used to remove the highest octave band. The undecimated residue data are then obtained. The kernel is then stretched (K=1) and the data filtered again to remove the next octave band. The procedure is repeated for K=2,3,... until all desired bands have been removed. Figure 2c shows the response of a stretched minimum coefficient filter for the case K=1.

Filtering by FFT methods has a significent advantage over full-kernel convolution for all but very small kernel sizes. It will now be shown that the class of filters described here are very competitive with FFT convolution for many practical applications. The one factor which is the largest contributor in determining filtering time, no matter which approach is taken, is the number of filter coefficients. In time domain convolution the run time increases linearly with the kernel size. With FFT convolution the size of the kernel determines the number of points in the transform. Table I shows the optimum relationship between kernel size and transform size, as provided by Helms.* If the number of kernel points becomes sufficiently large one may

*Helms, Howard D., "Fast Fourier Transform Method of Computing Difference Equations and Simulating Filters", IEEE Transactions on Audio and Electroacoustics, Vol. AU-15, No. 2, June 1967.

become memory bound and be forced to use a sub-optimum size transform. To avoid discontinuities in doing repetative FFT convolutions, each transform must overlap the previous transform by the number of points in the kernel. The entire filtering process is then completed by repeating the FFT approximately 2M/N-L+1 times; where M equals the length of the data list, N equals the number of points in the transform and L equals the number of kernel points.

Table II gives comparative convolution times for filtering 16384 data points using four different techniques. Programs were run in FORTRAN H on an IBM System 360/75. The FFT was implemented with Sande's* algorithm of Radix 2, and assuming all real data, performed a 2^N point transform on 2^{N+1} data points.

The number of kernel points used in column 1 of Table II was chosen to be close to the upper end of each range of those shown in column 1 of Table I, therefore simulating worst-case conditions for both methods of convolution. The results in Table II are also plotted in Figure 3. From this plot it can be seen that with respect to fast FFT convolution the classical full-kernel convolution loses its speed advantage at about 30 kernel points. Half-kernel convolution, being about 1.5 times faster than using a full-kernel, extends its speed advantage up to about 60 kernel points. And finally, the minimum-coefficients filter has speed advantage over the FFT convolution up to about 300 kernel points.

*Gentleman, W. M. and Sande, G., "Fast Fourier Transforms - For Fun and Profit", Proceedings, Fall Joint Computer Conference, 1966, pp. 563-578.

TABLE I

L_{min} through L_{max}	n	2^n point transform
11 - 17	6	64
18 - 29	7	128
30 - 52	8	256
53 - 94	9	512
95 - 171	10	1024
172 - 310	11	2048
310 - 575	12	4096
575 - 1050	13	8192
1050 - 2000	14	16384

TABLE I Relationship between kernel size and transform size for FFT convolution, from Helms [6].

NOT REPRODUCIBLE

TABLE II

NUMBER POINTS IN KERNEL	NUMBER REAL POINTS IN FFT	TIME FOR 1 FFT (SEC)	NUMBER TIMES REPEAT FFT	TOTAL FFT TIME (SEC)	TIME FOR FAST CONV. (SEC)	TIME FOR HALF KERNEL(SEC)	TIME FOR FULL CONV. (SEC)
13	64	.013	630	8.19	.64	2.65	3.87
29	128	.03	328	9.84	1.42	5.68	8.58
45	256	.07	154	10.78	2.21	8.84	13.3
93	512	.16	78	12.48	4.57	18.2	28.
165	1024	.37	38	14.06	8.11	32.4	48.9
309	2048	.82	18	14.76	15.2	60.8	91.9
573	4096	1.8	10	18.	28.1	112.4	169.9
1045	8192	3.9	4	15.6	51.3	205.	309.
1992	16384	8.55	2	17.1	97.9	391.	591.

TABLE II Filtering times using 4 different types of convolution on 16384 real data points. Computations were performed on an IBM 360/75 in FORTRAN H.

NOT REPRODUCIBLE

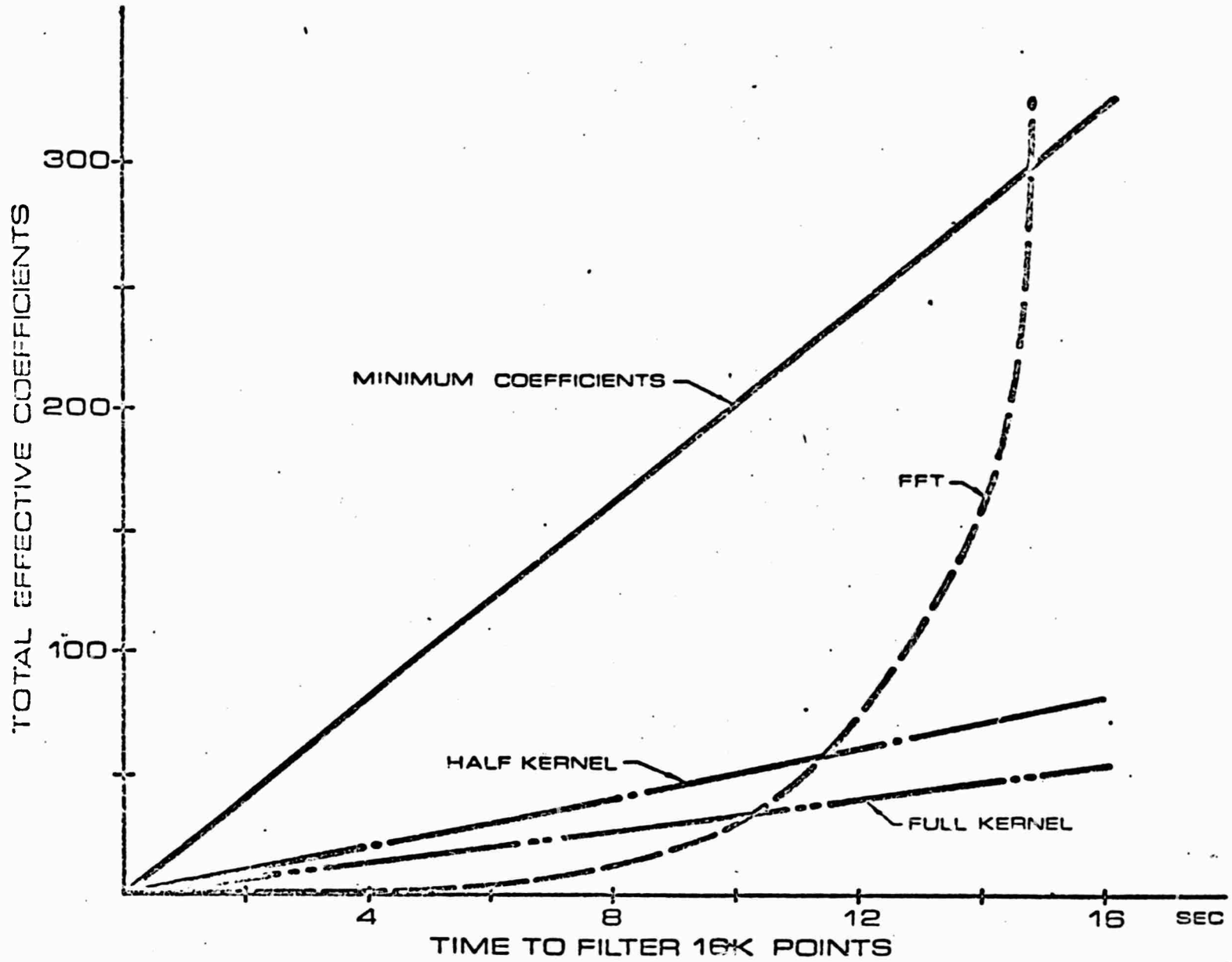


FIGURE 3 Filtering time comparisons programmed in FORTRAN H and executed on an IBM 360/75 using real variables.

The octave filtering band technique has been implemented on the 1800 speech system (2 μsec memory, 16 bit word) using assembly language, fixed point arithmetic, and kernels with from seven to eleven non-zero coefficients. Both the stretched-kernel (for K = 0,1,2,3,4) and the data-decimation techniques are available as options and the kernel currently being used has seven non-zero coefficients (N = 26). Because of the small number of arithmetic operations required, the accumulated truncation errors appear to be insignificant. No in depth error analysis has been attempted; however, when using the seven coefficient kernel, a DC signal is attenuated 80 db if double precision additions are performed and 70 db when single precision additions are used. A ten percent speed advantage is achieved using single precision arithmetic.

In our application human speech, bandlimited at 3200 Hz, sampled at 12.8 KHz, and stored in 8192 words of core memory is filtered into five octave bands in the following typical times:

Frequency Bands in Hz.	Filter Time (sec.)
1600 - 3200	2.06
800 - 1600	.99
400 - 1600	.49
200 - 400	.23
100 - 200	.12
Total Time	3.89

The 8K data list represents .64 seconds of speech data or about one-sixth the filtering time.

For many applications where octave bands are suitable, the stretched minimum coefficient family of digital filters can provide filtering algorithms which are both easy to implement and faster than equivalent FFT convolution. Moreover, because of their inherently short lag times and the straight forward arithmetic involved these methods are amenable to real time filtering using mini-machines or special purpose hardware.

(b) Wave Function Analysis/Synthesis

As indicated in the Semiannual Technical Report two one-pass wave function analysis/synthesis systems based on Cosine Modulation Methods have been developed. These systems are the Gaussian Cosine Modulated (GCM) Analyzer/Synthesizer and the Hanning Cosine Modulated (HCM) Analyzer/Synthesizer respectively.

A member of the GCM family of wave-functions can be described by the relation

$$U(t)_{GCM}=Ae^{-\left[\frac{\pi}{s}(t-c)\right]^2}\cos\ (2\pi F_o(t-c)-\phi) \qquad (5)$$

which is a Gaussian envelope of amplitude A, centered at time C, and with spread S, multiplied by a cosine wave of frequency F_o and phase ϕ with respect to C. An HCM wave function

$$U(t)_{HCM}=\frac{A}{2}\left[1+\cos\frac{2\pi}{s}\ (t-c)\right]\cos\ 2\pi F_o(t-c)-\phi) \qquad (6)$$

is a Hanning window with parameters A, S, and C multiplied by a cosine wave.

Both the GCM and HCM analysis algorithms have been implemented on the 1800 speech system. Both analyzers employ table look-up procedures and the coding has been optimized so that a direct one pass parameter calculation based on extrema data provides quite high speed analysis. The analysis

process consists of generating five lists of ASCØF parameters which completely specify the time behavior of five bands (e.g. the octave bands 100-200, 200-400, 400-800, 800-1600, and 1600-3200 Hz respectively) of filtered speech. From a comparative viewpoint both the GCM and HCM analyzers require about the same amount of time to analyze one second of speech (e.g. 3 seconds per second of speech on the 1800 system). The major difference between the two analysis systems is that the HCM analyzer generates on the average about five percent more ASCØF parameters than the GCM analyzer.

Both the GCM and HCM synthesis algorithms have been implemented on the 1800 speech system. The purpose of the synthesis systems has been to reconstruct speech waveforms from the respective GCM and HCM ASCØF parameter lists to make an auditory and/or visual (on a CRT) comparison with the original recording of the acoustic waveform to varify that the ASCØF lists accurately define the time domain behavior of the original acoustic waveform. Experimental studies over the last year involving a large number of speech samples have varified that both the HCM and GCM models will accurately represent the time domain behavior of human speech without any unacceptable errors in the analysis processes. Both visual observation of the time demain structure of synthetic speech versus the original recording coupled with aural comparisons of synthetic and original versions have shown excellent agreement in all cases between the two. Similar comparisons between HCM and GCM synthetic speech have also indicated excellent agreement. This is not to say that a perfect match exists between human speech and GCM or HCM wave function based synthetic speech. The data do not always precisely fit the models chosen, so that only a good approximation is being made at times. But, in almost all samples studied, these differences were apparent only to the eye and not the ear.

In summary, two wave function analysis/synthesis systems have been developed, implemented, and tested on the 1800 speech system which essentially completes this aspect of the speech project. The GCM analysis/synthesis algorithms have been described in an earlier ARPA report.(*) Both the HCM and GCM analysis/synthesis procedures are also completely described by Carey.(18)

(c) Computer Classification and Recognition of Phonetic Information

The research program in the classification and recognition of phonetic information has solidly established that recognition parameters can be extracted from the ASCØF parameter set and used to perform reliable recognition of speech. This has been demonstrated by the following accomplishments:

1. Development of a segmentation procedure which reliably partitions the input speech data into basic voiced-unvoiced segments.
2. Utilization of the segmentation algorithm in the formulation of a single speaker vowel recognizer for recognizing steady state vowels and vowels embedded between two unvoiced phonemes using ASCØF data.
3. Extension of the segmentation algorithm for voiced-unvoiced phonemes to the detection of unvoiced phonemes.
4. Preliminary development of an unvoiced phoneme recognizer.
5. Extension of the segmentation algorithms of (1.) and (3.) to the segmentation of voiced phonemes.

Voiced-Unvoiced Phoneme Segmentation and Vowel Recognition

The voiced-unvoiced segmentation algorithm and vowel-recognizer have been described in detail in the Semianmual Technical Report.(30) The single-speaker

*Final Report, June 30, 1970

vowel recognizer is a table driven recognizer which uses information supplied by the segmenter for the voiced segment to identify the 10 vowels and two diphthongs. Because of the preliminary nature of the segmentation, the vowels must be spoken in the context of two unvoiced consonants (e.g. vowel embedded between the two consonants). The information for the development and testing of the recognizer was obtained from a data base of 240 recordings. Each vowel was recorded 20 times, at various pitches, between various consonants, by a single speaker. Seven basic features (frequency in the upper three bands and energy in all 4 bands) were extracted from each recording and filed. Using these parameters, a series of two-dimensional maps were constructed to provide the appropriate vowel clustering and separation. This information was then employed to structure the vowel recognizer.

The confusion matrix resulting from the testing of the recognizer for the set of 240 recordings is depicted in Figure 5. For the 240 recordings there are only six cases where a vowel was missrecognized (e.g. 3 cases where /e/ was classified as /I/, two cases where /æ/ was classified as /ʌ/, and a single case where /ɛ/ was classified as /æ/). This gives a recognition score of 97.4 percent correct identification for the 240 recordings.

The two /æ,ʌ/ errors occurred because the best separations between /æ,ʌ/ that could be obtained had two cases of overlap of /æ/ into the /ʌ/ cluster. Similarly, the single error in which /ɛ/ was classified as /æ/ was because the /æ/ cluster contained one /e/. The problem of separation between /I,e/ was not resolved in feature space nor was a consistent differentiating characteristic found in the complex of wavefunction parameters. Since these two vowels overlapped in all frequency and energy maps, advantage

INTENDED VOWEL	RESPONSE											
	i	I	u	ɑ	ɔ	æ	ʊ	ʌ	o	e	ɝ	ɛ
i	20											
I		20										
u			20									
ɑ				20								
ɔ					20							
æ						18		2				
ʊ							20					
ʌ								20				
o									20			
e		3								17		
ɝ											20	
ɛ						1						19

Figure 5

Confusion matrix for single speaker vowel recognizer

was taken of the fact that /e/ is a diphthong which the segmenter breaks into two voiced segments. Thus the appearance of two voiced segments was used to separate /e/ from /I/. This rather crude decision element had the highest error rate by failing to break three out of twenty of /e/ into two segments.

The vowel recognition score for a single speaker was very encouraging. It reflected the belief that meaningful features could be extracted from the ASCØF set. The pitch synchronous frequency and energy estimators were condensed into representative frequency and energy elements with a sufficient accuracy that, in most cases, the examination of one or two of these elements enabled the discrimination between vowels or groups of vowels. Recognition of multiple speakers, if possible with this technique, may require an entirely new set of crossplots for each speaker or a simple adjustment of the existing decision elements. In either case, a scheme for automatic table generation for a known number of pattern classes would be desirable. Then, once a speaker had a table for his voice, the mere placement of his table in the recognizer would adapt the recognition system to his voice. These factors are being explored in order to adapt the single speaker recognizer to multiple speakers.

The 17-node vowel recognition tree is fairly efficient, having only six more nodes than the absolute minimum. In the worst case, eight nodes must be traversed to reach a terminal branch. A time measurement was made starting from the time the parameters were made available to the time when the entire word was processed. The spoken word was "shot" with a vowel duration of approximately 200 msec. This vowel required processing of eight nodes, and total run time was 114 msec.

Unvoiced Phoneme Segmentation and Recognition

The unvoiced phonemes of general American English are: /s/ see, /f/ fee, /θ/ thin, /ʃ/ she, /tʃ/ chin, /h/ he, /p/ pee, /t/ tee, /k/ key. All share one common property, aperiodicity, and all except /h/ have energy peaks above 1200 Hz. Ideally the unvoiced phonemes can be separated into two mutually exclusive classes, stoplike and fricativelike respectively. The /p/, /t/, and /k/ can exist in three different forms; 1) as a plosive in the initial position (pan, tan, can), 2) as a stop in the final position (complete closure and sound termination, e.g. map), 3) as a stop-silence-burst in the medial or final position (complete closure followed by pressure release). The /tʃ/ can exist as a (1) or (3), for if it occured as a stop without release, it would sound like a /t/, e.g. "match" would become "mat". The presence of silence preceeding the burst can generally be expected, except when the tongue, at the termination of the previous phoneme, is already in the position of the stoplike phoneme (e.g. the /n,t/ of "enter").

The fricativelike phonemes are /s/, /f/, /θ/, /ʃ/, and /h/. These are characterized as low-energy noise-like sounds, usually of longer duration than the stoplike phonemes.

Segmentation of Unvoiced Phonemes

Segmentation of the unvoiced phonemes involved the extension of the voiced-unvoiced segmentation algorithm to the unvoiced case. This was accomplished by modifying the segmenter to detect silence in the unvoiced segment.

The detection of silence serves as a means of separation of unvoiced phonemes in a significant number of words. Examples of this are the /s/ and

/p/ of "spin" or /f/ and /t/ of "lift". The /s/ followed by a /p/, /t/, or /k/ is probably the most commonly occuring sequence of unvoiced phonemes. Other words, such as "sticks" or "cliffs" contain coupled unvoiced phonemes, the majority of these words being plural nouns. The existence of a /p,s/ or /t,s/ pair can be detected by the preceeding silence but further separation of coupled unvoiced pairs has not yet been attempted in this study.

For any speech input which has been filtered into fixed bands, silence is defined as the lack of wave function activity for at least 30 msec in all bands, over the same time interval. The test for silence begins in Band 5, at the starting point of the unvoiced segment. The successive values of the ASCOF parameter

$$\Delta C_j = C_{j+1} - C_j \qquad j = 1,2,\ldots, n-1$$

(n = number of wavefunctions in the unvoiced segment)

are compared with 30 msec. If ΔC_j is greater than 30 msec, the time C_j is passed to the next lower band and the test continued from that point in that band. This time-transfer is performed on all five bands and then repeated again, with the second pass beginning at the largest C_j from the first pass. After the second pass, the resulting C_j is the time of the last wavefunction, over all five bands, which occurred prior to the quiet zone. Two passes are necessary to avoid the false detection of silence in low-energy signals such as in /f/ and /θ/. A simplified example of this is shown in Figure 6. Each "x" signifies a wavefunction and the figure illustrates a possible wavefunction sequence for the /f,t/ of the word "raft". The five bands are shown twice to simulate two passes.

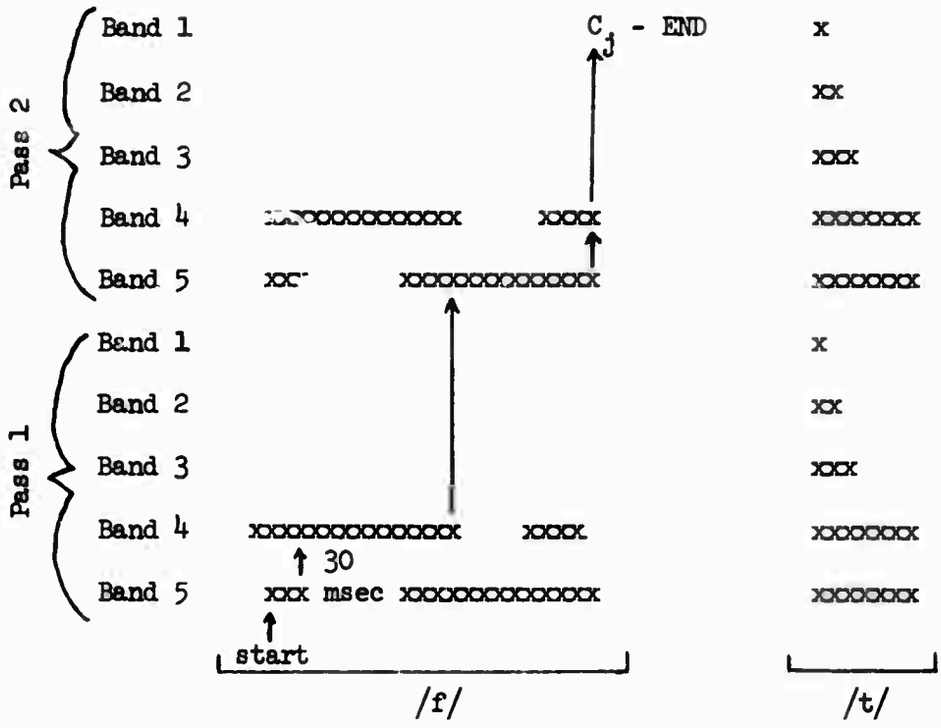


Figure 6 Two pass time transfer which might occur in the detection of silence in the /f,t/ of the word "raft".

The silence associated with a /p/, /t/, /k/, /tʃ/ in the medial or final position does not extend over the entire interval between the end of the voiced segment and the beginning of the burst. At the termination of the voiced segment the signal is still damping out for a short time before total silence exists. An example of this is shown in Figure 7 for the word "thick".

The segment immediately following the vowel is actually classified as "unvoiced" and the segment following that is the quiet zone. If the phoneme after the vowel is a /p/, /t/, /k/, or /tʃ/, this unvoiced segment can be considered as "null", for it contains little or no phonemic recognition information. However, in the general case, the unvoiced phonemes /s/, /ʃ/,

/f/, and /θ/ could also occur in this same position (/h/ is restricted to the initial position). Therefore, the null segment is treated as a phoneme, and a recognition step is performed to determine if the first unvoiced interval after the voiced segment is null or one of the other four possible phonemes. Besides the limitation on the position of /h/, the only other constraint placed on the unvoiced phonemes is that the /p/, /t/, and /k/ in the final position must be released, for the recognition of these three will be based upon the data in the burst segment.

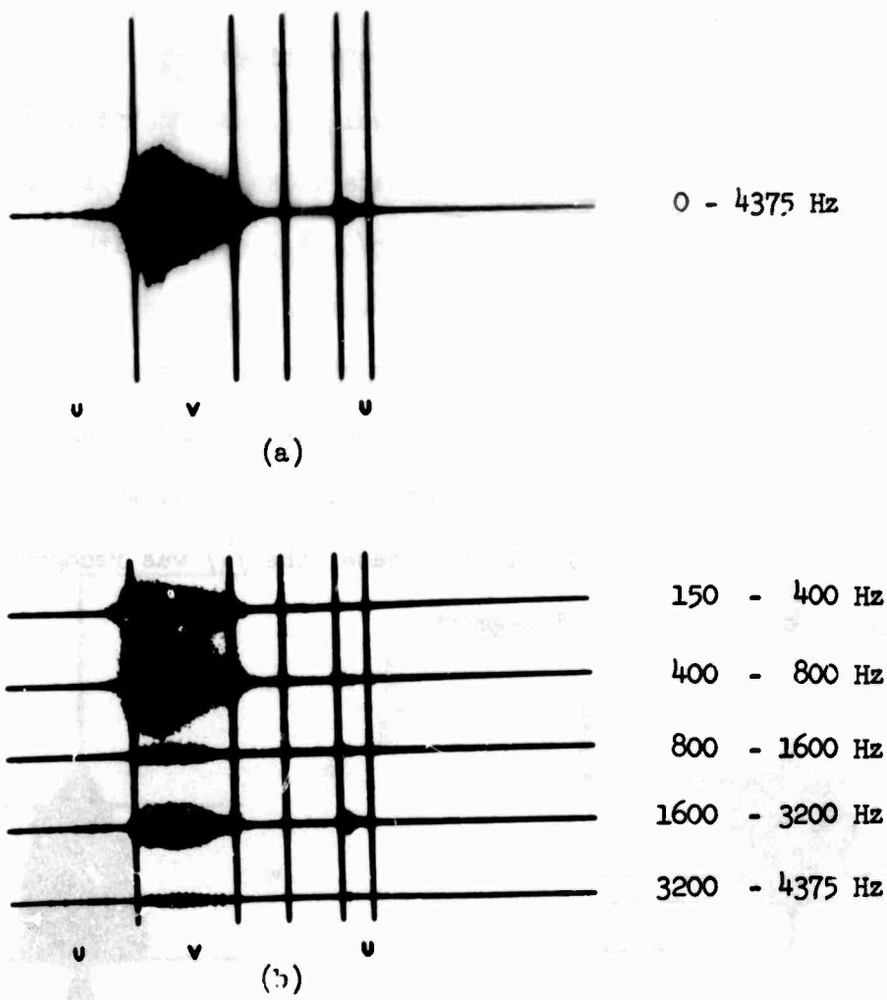


Figure 7 Marking of unvoiced phonemes (U) and voiced segments (V) for the word "thick".

Figures 8,9, and 10 depict typical results of the unvoiced phoneme segmentation process.

Figure 8 is a very simple example of the detection of silence at the end of the word "chief". In this case, each marked segment corresponds to one phoneme. Figure 8b illustrates the degree of energy in each band for the /tʃ/ and /f/. Especially note the very small signal in the upper two bands for the /f/. This same low energy signal can be seen back in Figure 7 for the /θ/of "thick". This is due to the fact that the energy peaks for these two phonemes are located well above the 4375 Hz cutoff of Band 5.

Figure 9 shows how the phonemes /s/ and /t/ of "sticks" are separated by silence. The voiced segment contains only vowel /I/, and the final segment contains the two phonemes /k,s/. The acoustic data looks like an /s/ but the preceeding silence indicates the presence of a /p/, /t/, or /k/ with the /s/.

Figure 10 illustrates the location of the unvoiced phonemes in the word "splashed". The /s/ characteristically shows its strongest energy concentration in Band 5, while the /ʃ/ has its energy components in bands 4 and 5. The /p/ is short duration with its largest component in Band 1, while the /t/ concentrates mostly in Bands 4 and 5. In this case, the /ʃ/ was recognized as an unvoiced phoneme and not a null segment.

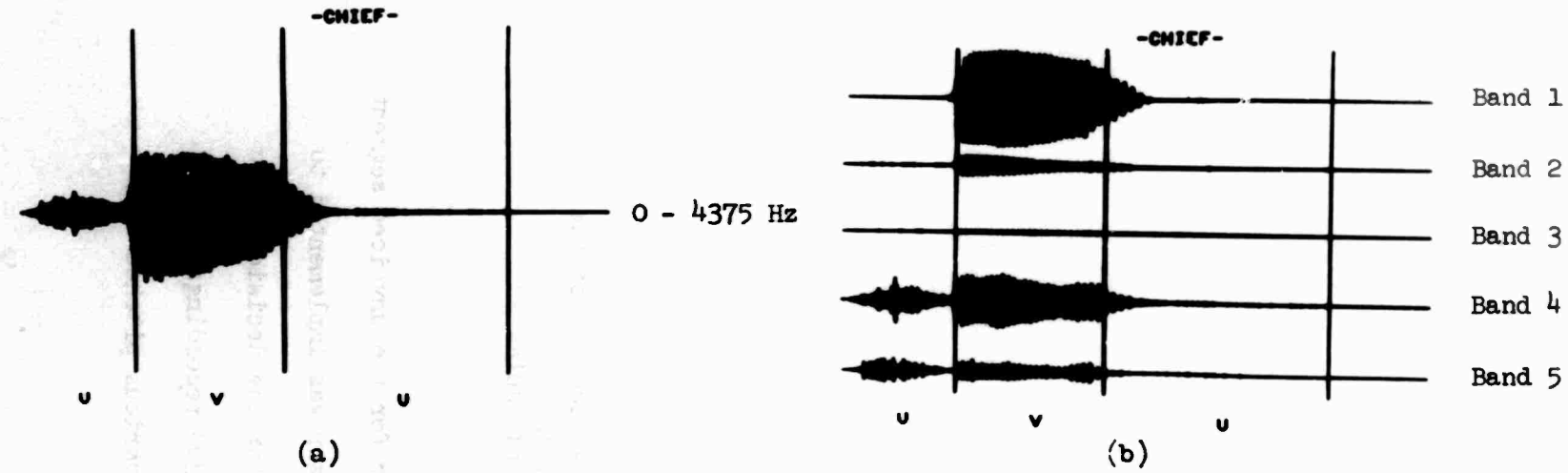


Figure 8 Markers indicating the location of the unvoiced phonemes (U) and the voiced segment (V) in the word "chief".

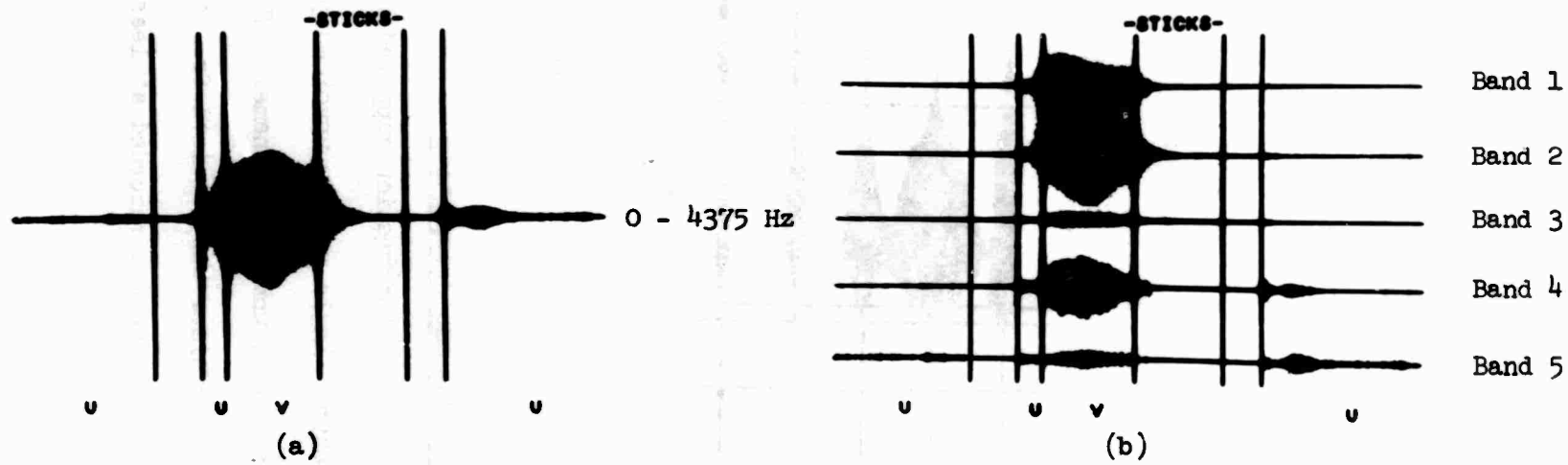


Figure 9 Markers indicating the location of the unvoiced phonemes (U) and the voiced segment (V) for the word "sticks". The final unvoiced interval actually contains the phoneme pair /k,s/.

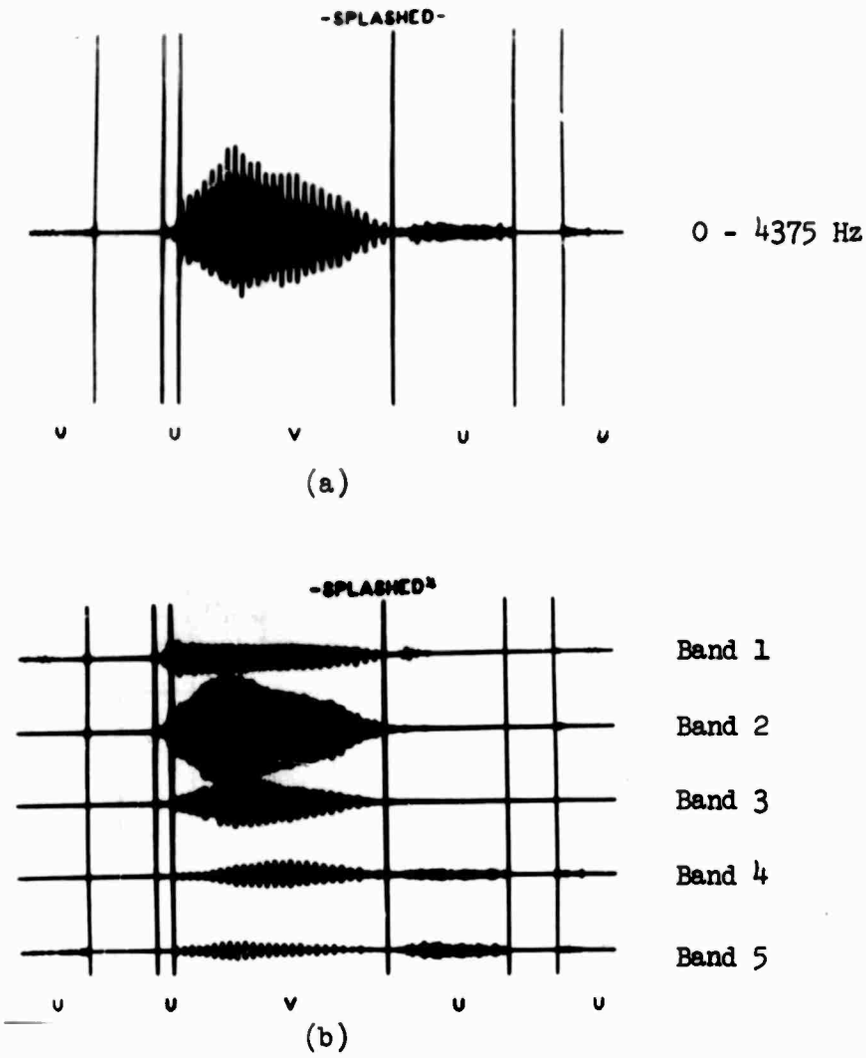


Figure 10 Markers indicating the location of the unvoiced phonemes (U) and the voiced segment (V) in the word "splashed".

Unvoiced Phoneme Recognition

Using the information supplied by the segmenter for the unvoiced segment a prototype single speaker unvoiced phoneme recognizer was implemented to identify the 9 unvoiced phonemes. The information for the decision tree in the recognizer was obtained from a data base of 300 recordings. Each unvoiced phoneme was recorded at least 20 times at various pitches by a

single speaker. A feature list like the one depicted in Figure 11 is extracted for each unvoiced segment of a word and filed. Using these features a series of two-dimensional maps were constructed to provide the appropriate unvoiced phoneme clustering and separation.

Phoneme Class (vowel, fricative, stop, nasal, etc.)
General Segment ID (Voiced, Unvoiced)
Displacement into feature list
Segment starting time
Segment stop time
Band 1 Energy
Band 2 Energy
Band 3 Energy
Band 4 Energy
Band 5 Energy
Average Band 1 Weighted Frequency
Average Band 2 Weighted Frequency
Average Band 3 Weighted Frequency
Average Band 4 Weighted Frequency
Average Band 5 Weighted Frequency

Figure 11 Format of feature list generated for each segment.

Typical examples of the two-dimensional clustering are shown in Figures 12, 13, and 14. Figure 12 shows the decision element for separating the unvoiced phoneme /k/ (K) from the unvoiced phonemes /ʃ/ and /tʃ/ (X and C respectively). A second example is illustrated in Figure 13 where the unvoiced phonemes /t/ (T) and /s/ (S) exhibit separation. Figure 14 illustrates a third example of two-dimensional clustering where the unvoiced

phoneme /p/ (P) has been separated from the phoneme /t/ (T).

The decision tree being employed for the prototype unvoiced phoneme recognizer is shown in Figure 15. All decision elements are either one or two demensional, based on cluster formations similar to those of Figures 12 - 14. Note that the tree contains 26 nodes and Figures 12 - 14 correspond to nodes 7, 15, and 20 respectively.

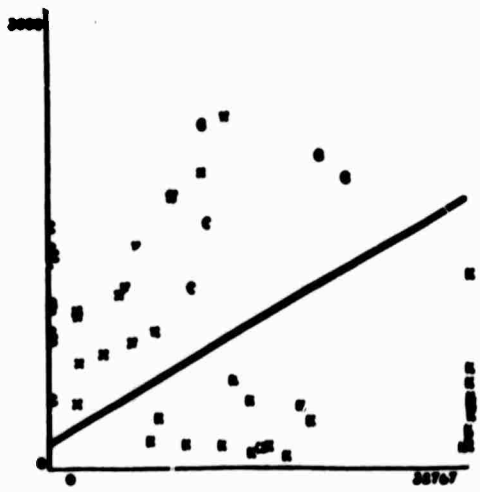


Figure 12 photo — Separation of unvoiced phoneme /k/ (K) from /ʃ/ (X) and /tʃ/ (C) using Band 5 energy and Band 1 energy.

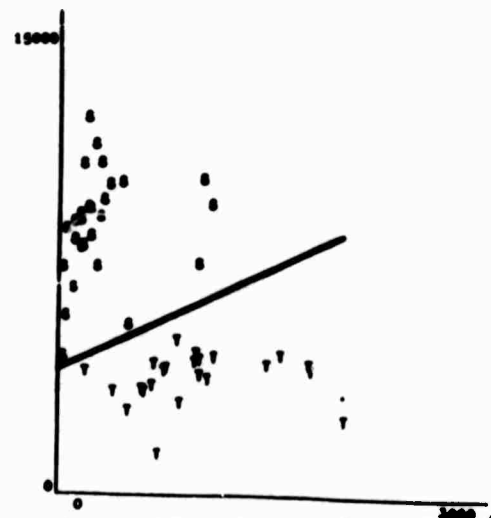


Figure 13 photo — Separation of /t/ (T) from /s/ (S) using time duration of unvoiced segment and Band 4 energy.

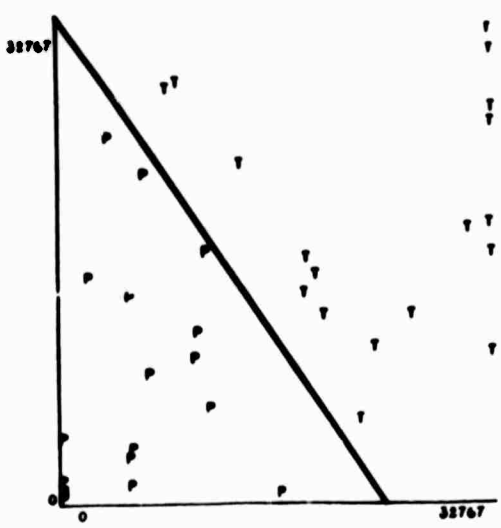


Figure 14 photo — Separation of /p/ (P) from /t/ (T) using Band 4 energy and Band 5 energy.

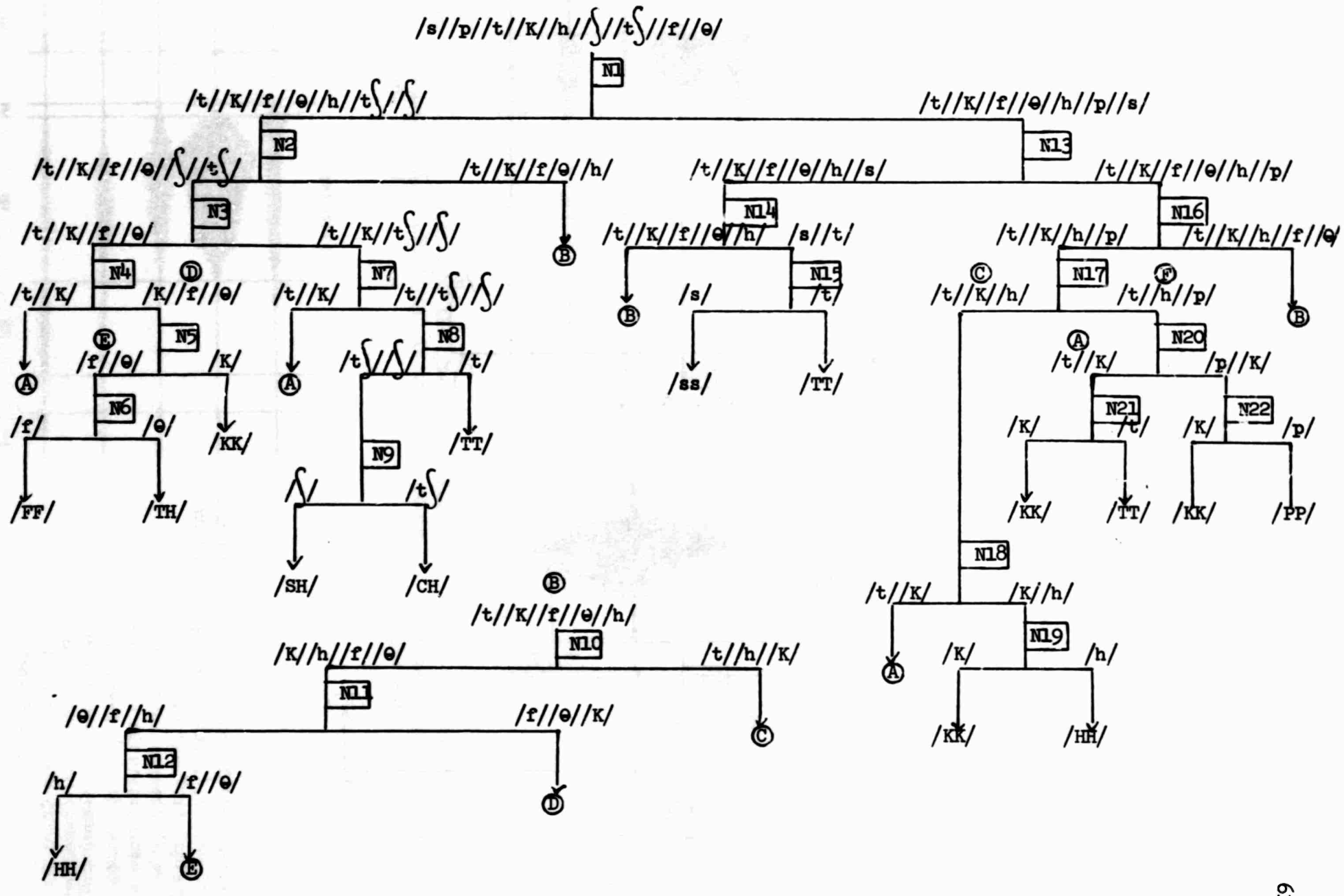
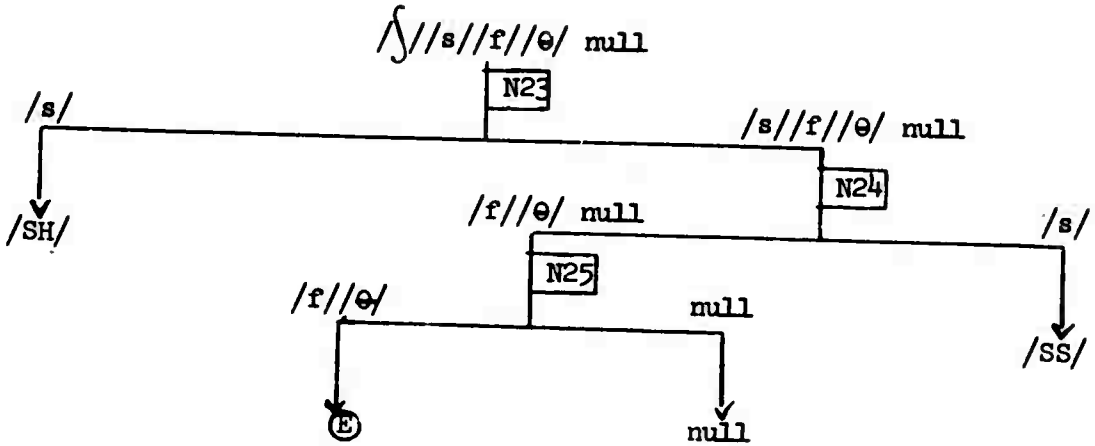
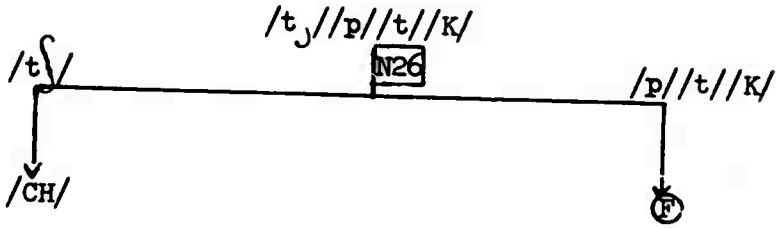


Figure 15 Decision Tree for Single Speaker Unvoiced Phoneme Recognizer


/ʃ//s//f//θ/ null
N23
/s/
/s//f//θ/ null
N24
/SH/
/f//θ/ null
/s/
N25
/f//θ/
null
/SS/
E
null


/tʃ//p//t//K/
N26
/tʃ/
/p//t//K/
/CH/
F

Some initial recognition results obtained from the recognizer are shown in Figures 16 - 18. Figure 16 illustrates the correct recognition of the two unvoiced phonemes /θ/(TH) and /k/ (KK) in the word "thick". In addition the vowel /I/ (II) has also been correctly recognized since the vowel recognizer has been included as part of the unvoiced phoneme recognizer. Figure 17 shows correct identification of the phonemes /k/ (KK), /I/ (II) and /k/ (KK) in the word "kick". Figure 18 shows correct recognition of the unvoiced phonemes /s,p,ʃ,t/ in the word "splashed". Note that the voiced segment /l,æ/ has been missrecognized as the vowel /U/ (UH). This is due to the fact that the segmentation program has no provisions for segmenting connected voiced phonemes.

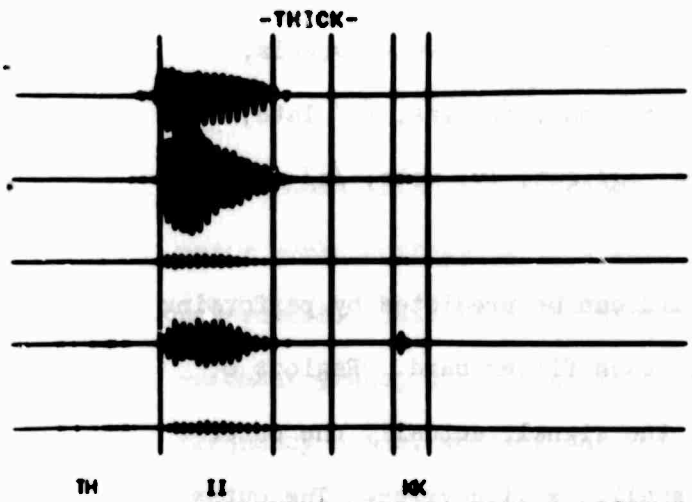


Figure 16 photo Correct recognition of the phoneme string /θ,I,k/ (TH,II,KK) in the word "thick".

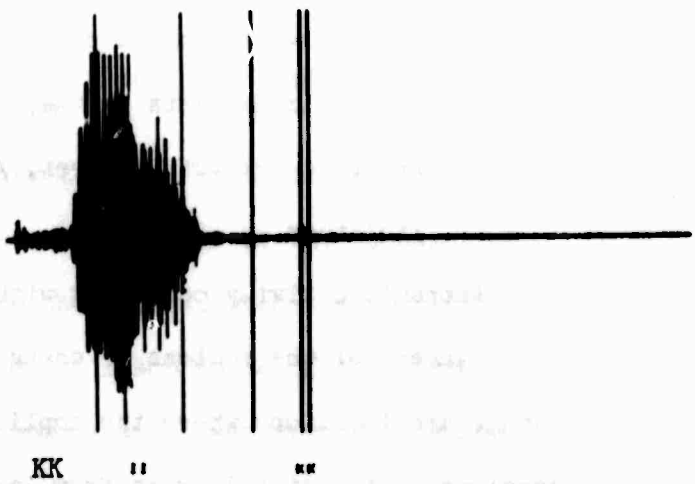


Figure 17 photo Correct recognition of the phonemes /k,I,k/ (KK,II,KK) in the word "kick".

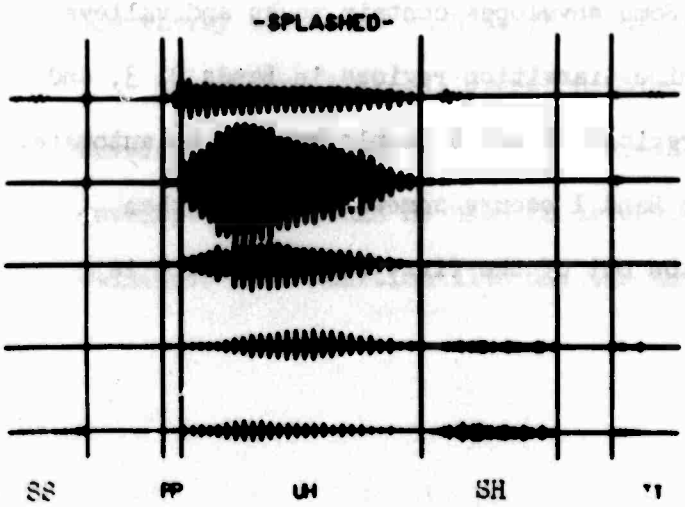


Figure 18 photo Recognition of the unvoiced phonemes /s,p,ʃ,t/ (SS,PP, SH,TT) in the word "splashed". The voiced segment /l,æ/ has been missrecognized as the vowel /U/ (UH) since no provisions for segmenting and recognizing connected voiced phonemes exist in the recognizer.

Segmentation of Voiced Phonemes

Voiced phoneme segmentation basically involved the extension of the unvoiced phoneme segmentation algorithm to isolate the clusters of voiced phonemes existing in the voiced segment generated by the segmenter. Clusters of voiced phonemes can exist as much longer strings than the unvoiced phonemes. For this reason, the phonemic coupling can become very complex and the boundaries very subtle. Therefore, the following preliminary constraints were established with regards to segmentation:

1) The voiced portion of any input word must not contain vowel-pairs or consonant-pairs.
2) The allowable voiced phoneme set consists of the twelve vowels, and the consonants /m/ me, /n/ no, /n/ song, /y/ yes, /l/ late, /r/ row, /w/ wet, /b/ bet, /d/ done, /g/ get, /v/ vote, /z/ zoo, /th/ them.

Phonemic activity occurring within a word can be predicted by performing an alignment of the regions of change within each filter band. Regions of change are locations where the amplitude of the signal, actually the outer envelope of the signal, goes from large to small, or vice versa. The outer envelope of each filter band during a voiced segment is generated during the pitch-synchronous data reduction step. The envelopes for the first four bands of "yellow" are shown in Figure 19. Some envelopes contain peaks and valleys and it is not too difficult to determine transition regions in Bands 2, 3, and 4. However, the alignment of these regions is not a simple process to automate. The transition into the first peak in Band 1 occurs somewhat earlier than those of Bands 2 and 3. The transition out of the first peak of Band 4 is

occurring at the peaks of Bands 2 and 3. Examples such as these can be found in a great number of words, making the job of alignment non-trivial.

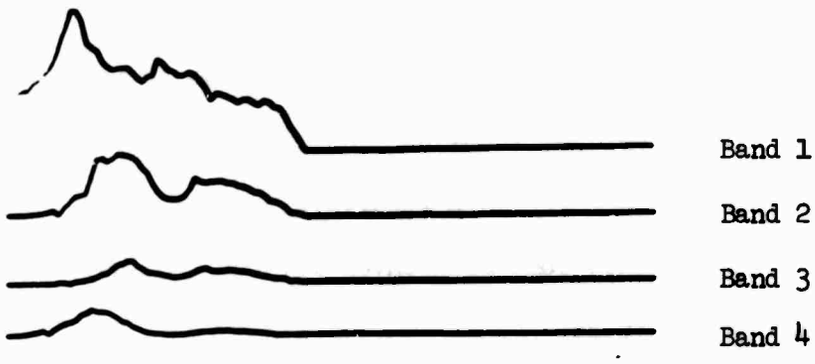


Figure 19 Envelopes of the first four bands of the word "yellow".

A most interesting feature of the envelopes is that, in most cases, the major peaks correspond to vowels and the valleys occur at the location of non-vowels. This is to be expected since the vowels, with their multiple resonant peaks, have significant energy in at least three of the first four frequency bands, whereas the voiced consonants are lower energy phonemes. Since most of the formant movement is taking place in the region covered by Bands 2, 3, and 4 the peak-valley phenomena is most obvious in these bands. The Band 1 envelope does not exhibit much change unless a vowel is next to a low energy consonant such as /b/, /d/, or /g/.

To eliminate the alignment problem associated with multiple bands, the envelopes from Bands 2, 3, and 4 were added together to form a single composite envelope which could be used to determine the transitions between peaks and valleys. This addition also has the advantage of accentuating the difference

between the peaks and valleys. Figure 20 shows some examples of the 3-band composite envelopes for the words "yellow", "leaving", and "alone". Recalling constraint number (1), it should be noted that vowel clusters or voiced consonant clusters occur as one large peak or one wide valley.

To reduce the detection of false peaks and valleys due to the sometimes uneven contour of this new envelope, it is subjected to a smoothing process. This is done by a program which performs a recursive 5-point smoothing of the composite envelope. The smoothed composite envelopes for "yellow", "leaving", and "alone" are shown in Figure 21.

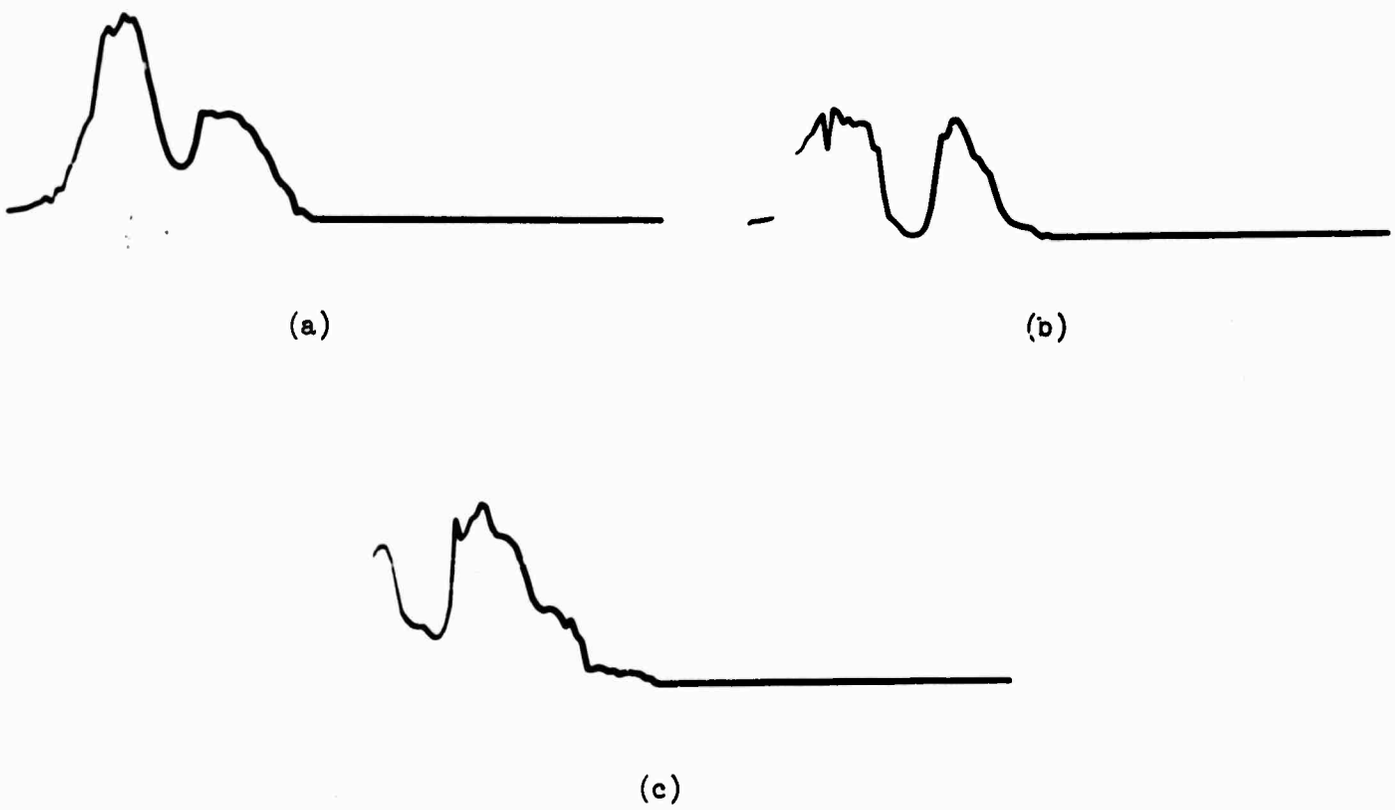


Figure 20 Composite envelope resulting from the summation of the individual envelopes from Bands 2, 3, and 4 for the words (a) "yellow", (b) "leaving", (c) "alone".

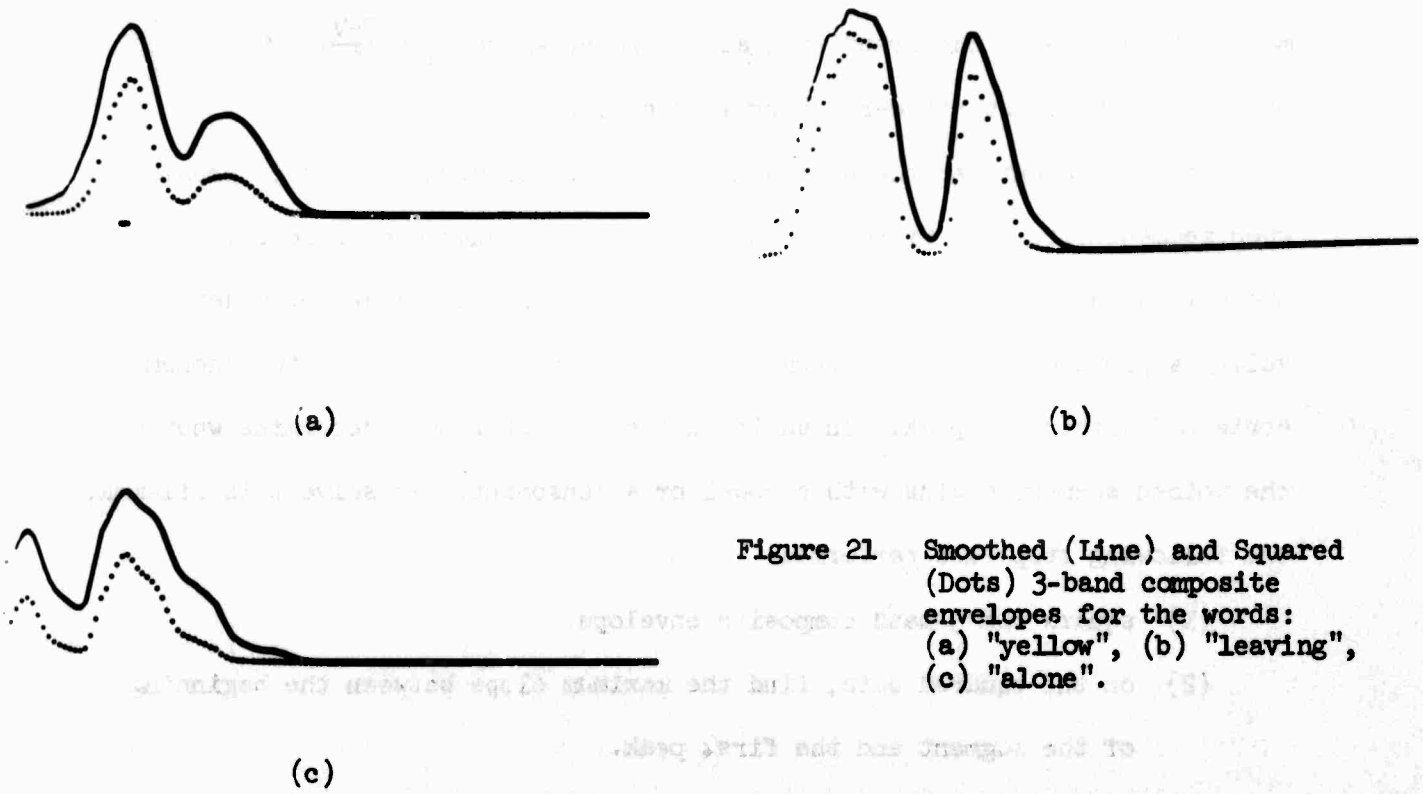


Figure 21 Smoothed (Line) and Squared (Dots) 3-band composite envelopes for the words: (a) "yellow", (b) "leaving", (c) "alone".

Based upon this single envelope, phoneme boundaries are determined to be somewhere along the transitions between peaks and valleys. A four-point check is used in the determination of a peak or valley. If $x_j > x_{j-1}$ and $x_j > x_{j+1}$ and $x_{j+1} > x_{j+2}$, then a peak of amplitude $P = x_j$ exists. If $x_j < x_{j-1}$ and $x_j < x_{j+1}$ and $x_{j+1} < x_{j+2}$, then a valley of amplitude $V = x_j$ exists.

In order for an envelope transition to qualify as a phoneme boundary, the distance between a valley and the preceeding peak, or a peak and the preceeding valley must be sufficiently large. This distance is a function of the data itself and depends upon whether the transition is peak-to-valley or valley-to-peak. If the current valley, V, is less than .5P (previous peak), then a boundary marker is placed on the transition at the point where $x_j = \frac{P+3V}{4}$. If the current peak, P, is greater than 1.25V (previous valley), then a boundary

marker is placed on the transition at the point where $x_j = \frac{P+V}{2}$. The values of the various constants were determined empirically.

Voiced segments at the beginning and ending of words introduce problems when segmentation is performed using the composite envelope approach. If the voiced segment at the beginning of a word is not preceeded by a definite valley a problem arises in determining whether there are one or two phonemes contained within the peak. In addition the segmenter must determine whether the voiced segment begins with a vowel or a consonant. To solve this dilemma, the following steps are performed:

(1) square the 3-band composite envelope.

(2) on the squared data, find the maximum slope between the beginning of the segment and the first peak.

(3) project that slope down to the time axis.

(4) if the time where the projection crosses the axis is greater than the starting time of the segment, then the segment begins with a consonant.

(5) if the time where the projection crosses the axis is less than the starting time of the segment, then the segment begins with a vowel.

When it is determined that a segment begins with a consonant, the half-way point between the beginning of the segment and the first peak of the 3-band composite (not squared) is designated as the boundary between the consonant and the vowel.

The reasoning behind this method of projection is the consonants, with their lower energy in Bands 2, 3, and 4, will cause the composite envelope (after squaring) to start out at a low amplitude and rise rather steeply to

the first peak. The vowels, on the other hand, will start out at a higher amplitude and therefore have a more shallow slope. The effects of squaring the 3-band composite envelope were included in Figure 21.

The end of the voiced segment poses the same problem as the beginning. Does the segment end with a vowel or a consonant? The same projection technique is used at the end, but with slight modification. The steps are:

(1) on the 3-band composite envelope (not squared) find the maximum slope between the end of the segment and the half-way point to the last peak.

(2) project the slope to the time axis.

(3) if the time of the axis intercept is less than the time of the 3rd-to-the-last pitch period, the segment ends with a consonant.

(4) if the time of the axis intercept is greater than the time of the 3rd-to-the-last pitch period, the segment ends with a vowel.

When it has been determined that a segment ends with a consonant, the boundary marker is placed at the point of maximum slope from step 1. In the ending conditions, the unsquared envelope is used because, at the end of a word, the vowels tend to taper off slowly in energy and therefore do not stop abruptly.

The following four figures demonstrate the effectiveness of the segmentation for a male speaker. The voiced phonemic segements are identified with a "V" for vowellike or a "VC" for voiced consonant. Unvoiced phonemes are designated by a "U". The vertical lines on the figures are not to be thought of as exact phoneme boundaries, but rather as approximate locations of transition regions.

Figure 22 illustrates how the /d/, /m/, /n/ and final /n/ of "dimension", by a male speaker, have been correctly classified as voiced consonants, and the /I/, /ɛ/, and /ʌ/ as vowellike. The /ʃ/ is also correctly classified as unvoiced. Figure 23 shows the correct voiced phoneme segmentation for the word "leaving". Figure 24 pictures the vowel/consonant separation for the word "yellow", by a male speaker. Figure 25 shows that, for the string of six coupled phonemes in the word "redeemer", the phoneme boundary markers are correctly located and the phonemes properly classified.

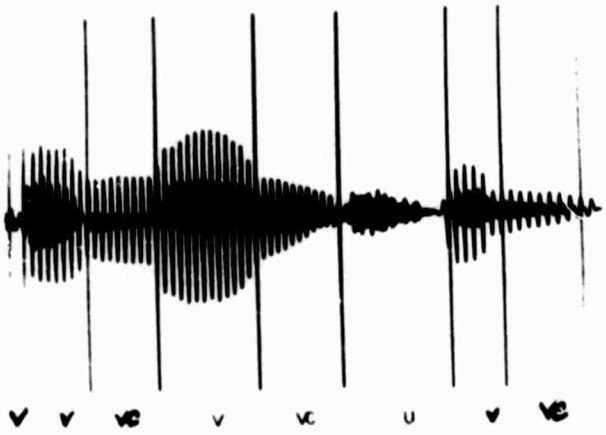


Figure 22 Separation of the vowellike sounds (V) from the voiced consonant sounds (VC) for the word "dimension" by a male speaker.

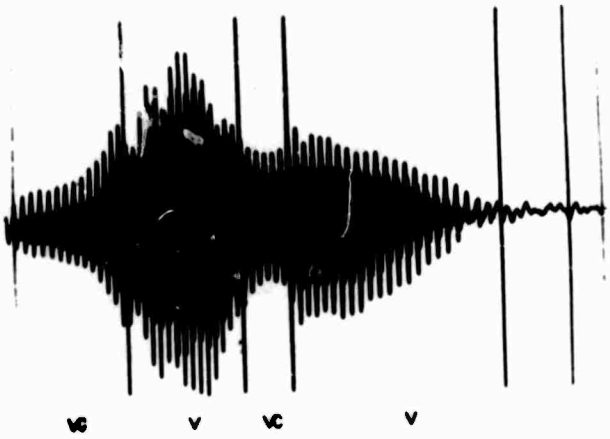


Figure 23 Separation of the vowellike sounds (V) from the voiced consonant sounds (VC) in the word "yellow", by a male speaker.

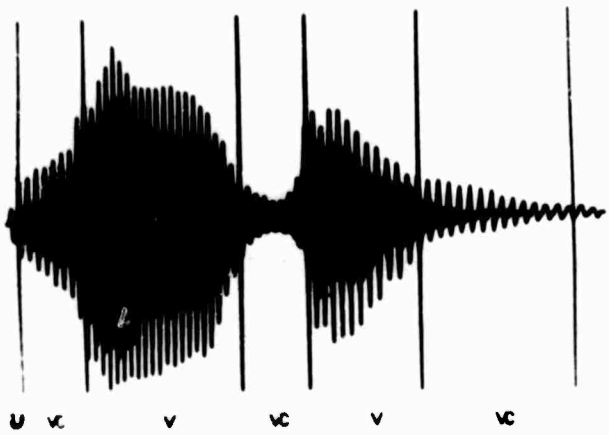


Figure 24 Separation of the vowellike sounds (V) from the voiced consonants (VC) in the word "leaving", by a male speaker.

Figure 25 Separation of the vowellike sounds (V) from the voiced consonant sounds (VC) in the word "redeemer", by a male speaker.

The segmenter described in this section represents a preliminary version of the final segmenter that will be used in a single-speaker phoneme recognizer. The segmenter is currently being tested on a wide spectrum of words in order to measure its effectiveness and to modify it where necessary. A data base of 157 words has been created up to this point in time. The basic features of frequency and energy in each frequency band have been extracted and entered into appropriate files for the voiced and unvoiced segment of each of these words. These features will be utilized to structure the single speaker phoneme recognizer when the segmenter has been finalized. Note that the major effort required in the recognizer will be the recognition of the voiced phonemes since algorithms have already been established for vowel and unvoiced phoneme recognition.

(d) Data Compression Studies

The end objective to data compression of speech is that all utterances or sounds still be intelligible. This criterion provides a necessary and sufficient test for all ideas, and is quantitatively measurable.

The basic speech waveform is bandlimited to .1 - 4.5 KHz., and the five sub-bands employed are .1 - .4, .4 - .8, .8 - 1.6, 1.6 - 3.2, 3.2 - 4.5 KHz.

Three general areas of study have been emphasized in the compression studies completed to date. The first is the tabulation of the data rate of the "highest fidelity" ASCØF representation. This involves making the best possible fit to the speech waveform, in the time domain, with Gaussian wave-functions and determining the required bit rate without further compression. The bit rate is estimated by computing the number of wave-functions required and assuming an average value of eight bits for each of the five parameters. For example: the word "Unite" [420 msec long, filtered into five sub-bands] was found to require 826 wave-functions. The bit rate is found by:

$$\text{bit rate} = 826\ \text{(wave-functions)} \times \frac{5\ \text{(parameters)}}{\text{(wave-function)}} \times \frac{8\ \text{(bits)}}{\text{(parameter)}} \times \frac{1}{420\ \text{(msec)}}$$

$$= 78{,}666\ \text{bits/sec}$$

Studies have shown bit rates for full words on the order of 45 - 100,000 bits/sec for the raw ASCØF stream. These results are somewhat higher than estimated previously; this is the direct result of including an additional band (3.2 - 4.5 KHz) which was found to be necessary for fricatives. The above bit rates improve somewhat for phrases, as the ASCØF representation automatically ceases for any quiet period, regardless of how short.

The second general area of study has been an attempt to determine how many of these wave-functions are superfluous from a perceptual standpoint. The elimination of redundant parameter sets is extremely simple to implement (given a criterion for redundancy), and seemed the logical place to begin. Two techniques are employed to reduce the ASCØF stream to a minimum level while still retaining the ability to reconstruct a satisfactory replica of the sound based only on the wave functions remaining.

A sort program has been implemented to select the "principal" (the amplitude of a wave-function being a local maximum or within 80% of one of its neighbors) wave-functions in the five frequency bands. In the lower two bands, most of the wave-functions qualify as principals; however, there is considerable redundancy in the higher three bands. The second technique discards ASCØF sets with an amplitude less than a (normalized) threshold. One procedure normalizes with respect to the largest amplitude in the band and sets a threshold test. This allows a consideration of the possible differences in amplitudes from band to band. Another procedure is to normalize with respect to the largest amplitude in a moving time window and employ a threshold test. For a group of test words, the percentage of ASCØF sets remaining after executing the preceding routines ranged from 13.7 - 30%.

The third general area of study has been the encoding of the remaining ASCØF sets. Assumed in the bit rate calculations for the original raw data were 8 bits per parameter. With the final criterion being intelligibility, an attempt is being made to determine the minimum number of bits necessary for each parameter. A summary of results obtained to date are as follows:

For amplitude, 3 bits are being used to quantize into 8 levels. Non-uniform quantization, with an emphasis on the lower values of amplitude was found to be more acceptable than equal spacing between levels.

For spread, 0 bits are now being used. It was found acceptable to use one value for spread in each of the five bands. The values being used are 8.1, 6.5, 2.5, 2.5, 1.8 msec. These values remain the same for all words.

For centers, ΔC and not C is quantized using 8 bits. A coarse-fine technique is used with the fine quantization (0 - 12.8 msec) accurate to the nearest .1 msec and the course quantization (12.8 msec - 128 msec) accurate to within 1 msec.

For phase, 1 bit is being used. It was found that quantizing phase often resulted in a "scratchy" quality. This was determined to be a function of the interdependence of phase and center location. The remedy was to quantize phase, and then by holding the time value of the maximum extrema constant, calculate a new corresponding center location. This eliminated any "scratchiness" by matching phase and center. The center location was found to adjust up to 1.5 msec for some words.

For frequency, it was found necessary to use 1 bit for the lower three bands and 2 bits for the upper two bands.

Results of test words used appear in Tables 1 and 2. Recognition at the reduced levels are quite good, however no formal testing has, to date, been done.

Studies are presently being conducted into possible repeating of ASCØF parameters for a voiced region. Preliminary results are encouraging and a further reduction of 500 - 1200 bits/sec per word would seem possible. Procedures for the final testing of intelligibility are currently under consideration.

Conclusion

Research accomplished has been in consonance with the tasks/objectives assigned. The research will form the foundation to continue research in On-Line computing and human speech as a mode of communicating with a computer. This research will be conducted under contract DAHC04-71-C-0043.

Submitted by:

David O. Harris
Principal Investigator

James A. Howard
Principal Investigator

Roger C. Wood
Principal Investigator

	Original bits/sec	Sort bits/sec	Normal bits/sec	Window bits/sec	% ASCOF Remaining	Encoding ASCOF bits/sec
Unite	78,666 b/s N=826	35,809 b/s N=376	24,381 b/s N=256	15,529 b/s N=163	19.7%	5114 b/s
Quarter	99,809 b/s N=1048	42,952 b/s N=451	35,333 b/s N=371	13,619 b/s N=143	13.7%	4392 b/s
Sheet	109,258 b/s N=1147	46,095 b/s N=484	36,000 b/s N=378	20,953 b/s N=220	19.1%	6915 b/s
Point	63,619 b/s N=668	31,523 b/s N=331	19,714 b/s N=207	13,619 b/s N=143	21.4%	4392 b/s
Division	92,380 b/s N=970	43,904 b/s N=461	37,809 b/s N=397	18,857 b/s N=198	20.4%	6142 b/s
Number	71,714 b/s N=753	33,619 b/s N=353	22,191 b/s N=233	14,953 b/s N=156	20.5%	4758 b/s
One	64,000 b/s N=672	30,285 b/s N=318	18,952 b/s N=199	12,191 b/s N=128	19.1%	3943 b/s
Two	88,000 b/s N=924	42,666 b/s N=448	35,333 b/s N=371	14,762 b/s N=155	16.8%	4700 b/s
Five	71,428 b/s N=750	32,000 b/s N=336	24,000 b/s N=252	19,334 b/s N=203	27 %	6251 b/s
Six	80,952 b/s N=850	34,285 b/s N=360	22,476 b/s N=236	14,191 b/s N=149	17.6%	4610 b/s
Seven	68,857 b/s N=723	31,523 b/s N=331	21,238 b/s N=223	15,810 b/s N=166	23 %	5060 b/s

TABLE 1

	Original bits/sec	Sort bits/sec	Normal bits/sec	Window bits/sec	% ASCOF Remaining	Encoding ASCOF bits/sec
Eight	86,190 b/s N=905	12,523 b/s N=331	35,047 b/s N=368	21,143 b/s N=222	24.5%	7094 b/s
Nine	71,714 b/s N=753	32,571 b/s N=342	22,571 b/s N=237	18,476 b/s N=194	25.8%	5821 b/s
Memory	72,285 b/s N=759	39,809 b/s N=418	26,286 b/s N=276	20,953 b/s N=220	29 %	6915 b/s
Byte	45,238 b/s N=475	20,761 b/s N=218	15,904 b/s N=167	13,809 b/s N=144	30 %	4394 b/s
Reader	72,571 b/s N=762	36,571 b/s N=384	29,905 b/s N=314	15,905 b/s N=167	22 %	5204 b/s
Mickey	47,714 b/s N=501	25,142 b/s N=264	17,619 b/s N=185	9429 b/s N=99	20 %	2913 b/s
Mouse	69,523 b/s N=730	30,285 b/s N=318	24,381 b/s N=256	21,715 b/s N=228	30 %	6820 b/s

Table 2