Designing a Reliable Voice-Input Robot Control Language

Jorg R. Jemelka
5967 South Gallup St.
Littleton, Colorado 80120

Conventional approaches to manipulator control are based either on concepts borrowed from programming languages with data structures and primitives appropriate to the task, or on the operator's intuitive understanding, translated through the medium of a teach pendant. In the first cage, the abstract world of coordinate frames is used to guide the robot; in the second, the operator must observe the manipulator and the teach pendant simultaneously. An alternative, experimental approach uses a speech-input device for robot control, thus allowing the user to keep his or her attention fixed on the manipulator during the training session.

The program described in this article was developed by the robotics group at the University of Illinois as a prototype of a device-independent, voice-input robot control language. The development version was implemented on an Apple II microcomputer; using a Heuristics Speechplot card for voice input. This card is no longer available; however, the concepts presented here apply to any similar voice-input system.

The target robot was a Rhino XR-1, an educational robot powered by servomotors and guided by an Intel 8748-based controller. (More detailed descriptions of the Rhino XR-1 can be found in the March/April 1982, the September/October 1982, and the January/February 1983 issues of *Robotics Age.)*

A Word About the Speech Card. The Heuristics Speechplot card has a programmable vocabulary of 32 words. The operator trains the board by speaking each word to be learned. The card samples the 1000 to 4000 Hz portion of the audio spectrum and converts each word to digital form using the word's high frequency, low frequency, and overall energy components. Word length is limited to 1.25 seconds; words are separated by 0.25 seconds of silence. Once spoken, the word is associated with a string or numeric variable.

When a word is spoken, the card attempts to match the input pattern with the stored samples using a rough averaging process. When a match is found, the corresponding string or numeric value is returned to the controlling program. Accuracy with this type of device ranges from 60 to 80 percent. This rate is decreased by the types of words that can be expected in this application. Our machine, for example, might be expected to respond to commands such as left, right, up, and down. These words are short and, except for down, end abruptly (with large high-frequency components). Because of the card's small frequency sam- pie space, the error rate for these and similar words is high.

We achieved better results (70 to 90 percent accuracy) with a vocabulary tailored to the characteristics of the Speechlab board. Words borrowed from the Romance-languages tend to have a longer and better-defined sample space; therefore, left was replaced with *gauche,* right with *droit,* ahead with *avant,* and behind with *arriere.* (As a rough guide, these words are pronounced *gosh, dwah, avahn,* and *ah-ree-ai1;* respectively.) The English equivalents appear in the program displays; the operator simply speaks them in French.

Designing the Command Set. We designed commands that rotate, raise, and lower the end effector; and permit opening and closing the fingers. We also included commands to move the robot from one position to the next. They are RIGHT, LEFT, HIGHER, LOWER, AHEAD, and BEHIND. Their effects, illustrated in figure 1, are as follows:

.RIGHT and LEFT move the base as you would expect

.HIGHER moves the shoulder toward the rear of the robot, keeping the shoulder - elbow angle constant.

.LOWER moves the shoulder forward while retaining the shoulder-elbow angle

.AHEAD moves the shoulder to the front and increases the shoulder-elbow angle

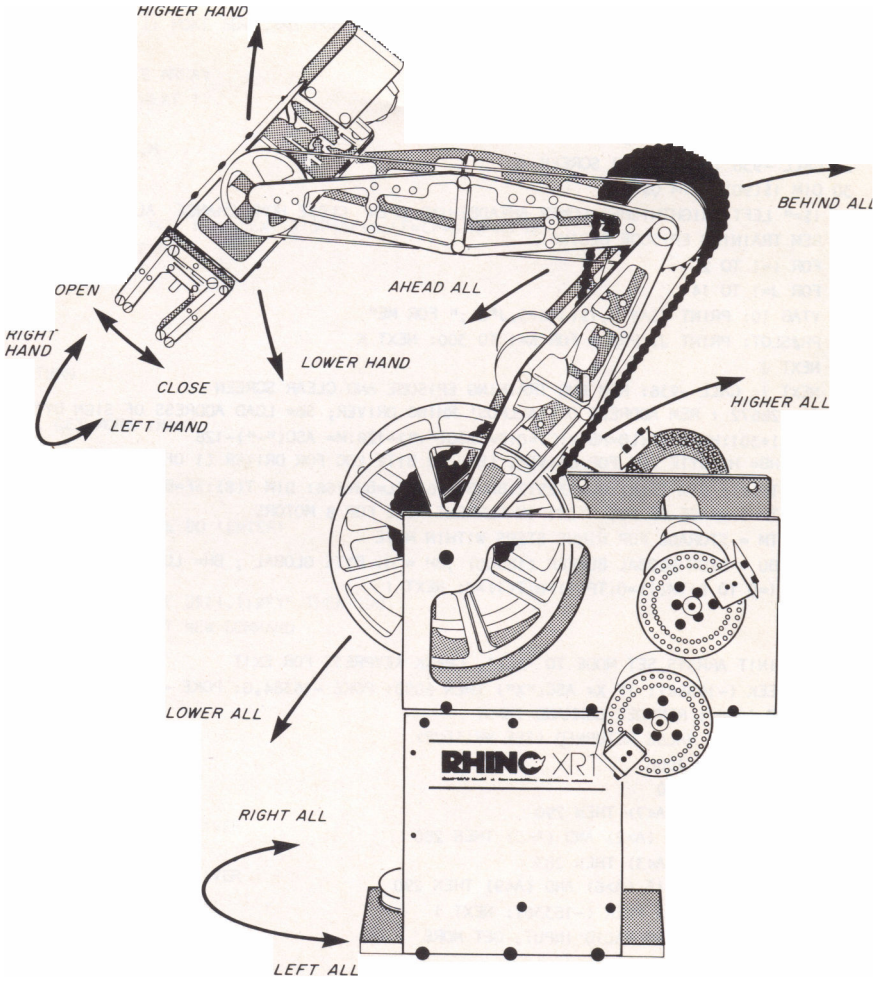
.BEHIND moves the shoulder to the ,rear and reduces the shoulder-elbow angle.

Because of the input device's low reliability, we developed a control strategy where the number of the robot's commands is limited, depending on the context, or mode, that is set. The CHANGE command alters the mode and, therefore, the currently recognized command set.

We partitioned the robot into two coordinate frames, designated ALL and HAND. The ALL coordinate frame consists of the principal axes-the base, elbow, and shoulder; the HAND frame deals with the end effector only-rotating it, and opening and closing the fingers. The commands RIGHT and LEFT result in different actions, depending on which mode is set The AHEAD command is not recognized in the HAND mode, and the OPEN and CLOSE commands are invalid in the ALL mode.

Three additional commands are valid after the CHANGE command. The ENTER command permits the entry of a move, the RUN command executes a move sequence, and the HOME command returns the robot to its starting configuration and automatically constitutes an additional move. Before the RUN or HOME command is executed, the operator must verify his or her request by typing Y and pressing the return key.

Language Summary. The control structure of this language operates on the principle of partitioning the recognized commands for the three available modes, thus decreasing the margin for error. Each mode has a limited number of options within it and, if an illegal request for a given mode is returned by the speech card, the input is



Legal commands and their effects in the ALL mode
RIGHT -moves base to the right
LEFT -moves base to the left
HIGHER -moves shoulder toward rear; maintains shoulder-elbow angle LOWER -moves shoulder forward; maintains shoulder-elbow angle AHEAD -moves shoulder toward front; increases shoulder-elbow angle BI;:HIND -moves shoulder to rear; decreases shoulder-elbow angle

Legal commands and their effects in the HAND mode RIGHT -rotates hand clockwise
LEFT -rotates hand counterclockwise
OPEN -opens fingers on hand
CLOSE -'- closes fingers on hand
HIGHER -raises hand
LOWER -lowers hand

Legal commands and their effects in the CHANGE mode ALL -switches to the ALL coordinate frame
HAND -switches to the HAND coordinate frame
ENTER -permits the entry of a move
RUN -executes the move sequence stored in memory HOME -returns the robot to its starting configuration

Table I. A summary of the robotcontrol language command words. Note that the RIGHT, LEFT, HIGHER, and LOWER commands have different effects, depending on the mode that the system is in.

```
10 POKE 74,124: POKE 204,124: POKE 75,21: POKE 205,21:SLOT=4
11 REM SET LOMEM TO 5500
12 PRINT "" ;"BLOAD RESET ,A$9200"
13 PRINT "";"BLOAD ALLMOS ,A$9000"
14 REM LOAD ASSEMBLY LANGUAGE DRIVERS
15 REM HIMEM = -32513
20 PR#SLOT: PRINT: PR#O
 16 REM INIT VOICE BOARD
25 CALL -936 : REM CLEAR SCREEN
30 DIM I$(90) : DIM Q$(5)
35 1$=" LEFT RIGHTHIGHER LOWER AHEADBEHIND (PEN CLOSE HOME CHANGE ALL HAND RUN ENTER "
36 REM TRAINING EPISODE BEGINS
40 FOR I=1 TO 2
50 FOR J=1 TO 14
60 V TAB 10: PRINT "SAY" ; I$( J*6-5,J*6) ;" FOR ME"
70 PR#SLOT: PRINT J: PR#O: F~ K=1 TO 300: NEXT K
90 NEXT J
100 NEXT I: CALL -936: REM END TRAINING EPISODE AND CLEAR SCREEN
105 SI=-28672 : REM ADDRESS OF 'ALLMOS' RHINO DRIVER; SB= LOAD ADDRESS OF SIGN BYTE
110 SB=SI+301:HB=SB+8:LB=HB+8:CT=0:P= ASC("+")-128:M= ASC("-")-128
115 REM HB= HI BYTE LOC FOR DRIVER, LB = LO BYTE LOC FOR DRIVER (1 OF 8)
120 DIM TP(8): DIM TM(8):BU=-32512:BH=BU+768:BL=BH+768: DIM T(8):SF=BU:HF=8H:LF=BL:CNT=BL+767 125 REM TP=ST~AGE OF POS
STEPS IN CURRENT MOVE FOR 8 MOTORS
126 REM TM = STORAGE FOR MINUS STEPS WITHIN MOVE
127 REM BU = SIGN GLOBAL BUFFER ($8000) ;BH = HI BYTE GLOBAL; BH= LO
130 FOR 1=0 TO 8:TM(I)=0:TP(I)=0:T(I)=0: NEXT I
150 MD=1

155 REM INIT ARRAYS,SET MODE TO 'ALL', CHECK KEYPRESS FOR EXIT
190 X= PEEK (-16384): IF X= ASC("X") THEN 6050: POKE -16384,0: POKE -16384+16,0
195 GOSUB 3000 : REM GET CO~AND INPUT
196 REM PROCESS VALUE RETURNED (SEE WRITEUP)
200 IF A=10 THEN 4000
210 IF A=9 THEN 1100
220 IF (MD=I) AND (A<7) THEN 290
230 IF MD=2 THEN IF (A>2) AND (A<5) THEN 280
250 IF (MD=2) AND (A<3) THEN 285
260 IF (MD=2) THEN IF (A>6) AND (A<9) THEN 290
265 FOR 1=1 To 10:X= PEEK (-16336): NEXT I
270. GOTO 190: REM NO VALID INPUT, GET ~E
280 A=A+7: GOT0290
285 A=A+I1
290 REM SET UP RETURNED VALUE ACCORD I NG TO MODE AND JUMP TO APPROPR I ATE ROUT I NE
295 GOTO (A+2)*100
300 FOR 1=0 TO 7: POKE SB+I,P: POKE HB+I,o: POKE LB+I,o: NEXT I: POKE LB+2,(5*16):TP(2)=TP(2)+50: CALL -28672: OOT0190
400 FOR 1=0 TO 7: POKE SB+I,P: POKE HB+I,O: POKE LB+I,o: NEXT I: POKE SB+2,M: POKE LB+2,BO:TM(2)=TM(2)-50: CALL -28672: OOTO 190 450 REM
300= 'LEFT' (MOVE BASE +50) ; 400= 'RIGHT' (MOVE BASE -50)
500 FOR I=0 TO 7: POKE SB+I,P: POKE HB+I,o: POKE LB+I,O: NEXT I: POKE SB+3,M: POKE LB+3,80:TM(3)=TM(3)-50:
550 POKE SB+4,P: POKE LB+4,80:TP(4)=TP(4)+50: CALL -28671: OOTO 190
575 REM 400 TO 450 = 'HIGHER' I. E. SHOULDER GETS +50, ELBOW GETS -50 STEPS
600 FOR I=0 TO 7: POKE SB+I,P: POKE HB+I,O: POKE LB+I,O: NEXT 1: POKE SB+3,P: POKE LB+3,80:TP(3)=TP(3)+50
650 POKE SB+4,M: POKE LB+4,80:TM(4)=TM(4)-50: CALL -26671: OOTO 190
675 REM 600 To 650 = lLOWER', SHOULDER GETS -50; ELBOW GETS +50 STEPS
700 FOR I=0 To 7: POKE SB+I,P: POKE HB+I,O: POKE LB+I,o: NEXT I: POKE SB+3,P: POKE LB+3,80:TP(3)=TP(3)+50
750 POKE SB+4,P: POKE LB+4,80:TP(4)=TP(4)+50: CALL -28671: OOTO 190
775 REM 'AHEAD' SHOULEDR GETS +50, ELBOW GETS +50 STEPS
800 FOR I=0 To 7: POKE SB+I,P: POKE HB+I,O: POKE LB+I,O: NEXT 1: POKE SB+3,M: POKE LB+3,80:TM(3)=TM(3)-50
850 POKE SB+4,M: POKE LB+4,80:TM(4)=TM(4)-50: CALL -28671: OOTO 190
875 REM 'BEHIND' SHOULDER AND ELBOW GET -50 STEPS, UPDATE TM ARRAYS
900 FOR I=0 TO 7: POKE SB+I,P: POKE HB+I,O: POKE LB+I,O: NEXT 1: POKE SB+7,M: POKE LB+7,80:TM(7)=TM(7)-50: CALL -28672: OOTO 190 925 REM
'OPEN' FINGERS
1000 FOR I=0 TO 7: POKE SB+I,P: POKE HB+I,O: POKE LB+I,O: NEXT I: POKE SB+7,P: POKE LB+7,80:TP(7)=TP(7)+50: CALL -28672: OOTO 190 1050
REM 'CLOSE' FINGERS
1100 V TAB 20: TAB 1: PRINT "HOME1 ";: INPUT Q$: IF Q$="" THEN 1150: IF Q$(1,1)I"Y" THEN 1150
1105 REM KEYBOARD VERIFICATION FOR 'HOME'
1110 FOR I=0 TO 7: POKE BU+(CT*8)+I, ASC("*")-128: POKE BH+(CT*8)+1,0: POKE BL+(CT*8)+1,0:CT=CT+1
1115 SF=SF+8:HF=HF+8:LF=LF+8: POKE CNT,CT
1120 CALL -28160
1150 V TAB 20: TAB I: PRINT "
1190 OOTO 190
1195 REM END OF 'HOME' ROUTINE
1200 SN=P:TP(5)=TP(5)+50
1225 REM HAND 'HIGHER' ROUTINE, VALID ONLY IN MODE 2
1250 FOR I=0 TO 7: POKE SB+I,P: POKE HB+I,o: POKE LB+I,O: NEXT I: POKE SB+5,SN: POKE LB+5,80: CALL -28672: OOTO 190
1300 SN=M:TM(5)=TM(5)-50: GOTO 1250
```

```
 1350 REM HAND 'LOWER' ROUTINE SET SIGN TO -, UPDATES HAND ASSEMBLY ARRAY LOC.
 1400 SN=P:TP(6)=TP(6)+50
 1425 REM 'LEFT' FOR HAND ASSEMBLY, VALID ONLY IN MODE 2, UPDATE ARRAY
 1450 FOR I=0 TO 7: POKE SB+I,P: POKE HB+I,o: POKE LB+I,O: NEXT I: POKE SB+6,SN: POKE LB+6,60: CALL -26672: GOTO 190
 1500 SN=M:TM(6)=TM(6)-50: GOTO 1450
 1525 REM 'RIGHT' FOR HAND ASSEMBLY, VALID ONLY IN MODE 2, UPDATE ARRAY
2000 V TAB (10): FOR I=0 TO 6:T( I)=TM( I)+TP( I):TM( I)=O:TP(I)=O: NEXT I
2005 REM 'ENTER' COMMAND: ADD +/- TM AND TP ARRAY ELTS
2010 FOR I=0 TO 7: POKE SF+I,P: IF SGN (T( 1)=-1 THEN POKE SF+I,M
2012 REM UPDATE GLOBAL SIGN BUFFER
2015 X= ABS (T( I)):Y=X MOD 100:X=X/100:
2020 POKE HF+I, ((X/10)*16) + (X MOD 10): POKE LF+I, ((Y/10)*16)+(Y MOD 10): NEXT I:SF=SF+6:LF=LF+6:HF=HF+6
2025 REM CONVERT MOVES TO BCD AND STORE I N H I AND LO GLOBAL BUFFERS
2030 CT=CT+I: POKE CNT,CT: GOTO 190
3000 V TAB 7: TAB 1: PRINT "YOUR COMMAND ";
3005 REM COMMAND ENTRY ROUTINE
3010 IN#SLOT: INPUT A: IN#O: PR#O: TAB I
3020 V TAB 5: PRINT "YOU SAID ";I$(A*6-5,A*6);" ?": RETURN
4000 PRINT "": GOSUB 3000: IF A<11 THEN 4000
4005 REM 'CHANGE' COMMAND ALL KINEMATICS INVALID HERE, GET MORE INPUT
4010 IF A=11 THEN MD=1 : REM SET 'ALL' MODE
4020 IF A=12 THEN MD=2 : SET 'HAND' MODE
4030 IF A=13 THEN 5000 : REM GOTO 'RUN' ROUTINE
4032 IF A=14 THEN IF CT<93 THEN 2000: REM IF LESS THAN 93 MOVES THE DO 'ENTER'
4035 A=0 : REM SET COMMAND RETURNED VAL TO 0
4040 GOTO 190
5000 V TAB 20: TAB 1: PRINT "RUN "; : INPUT Q$: IF Q$="" THEN 190: IF Q$(1,1)="Y" THEN 190
5004 REM VERIFY 'RUN' COMMAND WITH KEYBOARD CONFIRMATION IF 'N' GET NEW COMMAND
5005 POKE -16364+16,0: POKE -16364,0
5010 FOR I=BU TO (SF-6) STEP 6: REM FROM BEGIN OF GLOBAL TO END
5020 FOR J=O TO 7:X= PEEK( I+J): IF X= ASC("*")-126 THEN 6000
5021 REM IF ,*, IN GLOBAL THE D0 'HOME' ROUTINE
5025 POKE SB+J,X
5026 REM TAKE SIGN FROM GLOBAL AND PUT INTO 'ALLMOS' MOTOR DRIVER
5030 X= PEEK (1+766+J): POKE HB+J,X
5031 REM TAKE HI MOTOR BYTE FROM GLOBAL AND PUT INTO 'ALLMOS' MOTOR DRIVER
5035 X= PEEK ( 1+766*2+J): POKE LB+J,X
5036 REM TAKE LO MOTOR BYTE FROM GLOBAL AND PUT INTO 'ALLMOS' MOTOR DRIVER
5040 NEXT J: CALL -26672:
5041 REM ONE MOVE TRANSFERRED, CALL DRIVER
5045 NEXT I: REM DO FOR ALL MOVES IN GLOBAL
5050 A= PEEK (-16364): IF A<126 THEN 5010
5055 REM IF KEYPRESS THEN EXIT ELSE 00 ALL MOVES AGAIN
5060 MD=1: POKE -16364+16,0: POKE -16364,0: GOTO 190
6000 CALL -26160: REM CALL 'HOME' ROUTINE
6010 GOTO 5045 : REM RETURN FOR NEXT MOVE IN GLOBAL
6050 END
```

Listing I. The control program ROBOVOX. This program. written in Apple Integer BASIC. provides the means for obtaining and evaluating the spoken commands returned by the Heuristics Speechplot card. It uses two assembly-language routines. which control the Rhino XR-I robot.

ignored and another is requested. The chances of a recognition error occurring in the options within any mode is smaller than the possibility of error between mode subsets, thus enhancing overall reliability. Table 1 contains a summary of the language's commands.

Program Analysis. The control program (listing 1), written in Integer BASIC, uses two assembly-language routines to actually drive the robot. Each command moves the joint motors corresponding to that command 50 encoder steps. Each ENTER command updates the global move buffer (an area of the Apple's memory where the recorded moves are stored) with the total number of steps executed by the active motors since the previous ENTER command. The RUN command executes all the moves stored in the global buffer by repeatedly calling the ALLMOS assembly language driver and performing all moves in turn until the buffer is empty. The stored move sequence is executed continuously or until any key is pressed.

The BASIC program, called ROBOVOX, requires that the operator train the system by reciting each command word twice. Two independent samples of each command are thus available for matching against future jnput. The resulting digitized spectral sample is then stored. The training session occurs in lines 40-100.

The subroutine at line 3000 obtains the input command and displays the recognized command on the screen. The program stops if the operator types X at any time during command entry. If the speech card cannot recognize the word, it returns a value greater than the number of commands (14) and requests another command.

The commands are decoded in lines 200-270. A returned value of 10, corresponding to the CHANGE command,