MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo. No. 172

February 1969

Robot Utility Functions

Stewart Nelson

Michael Levitt

This document describes a set of routines which have been provided at both the monitor and user level to facilitate the following operations:
1) Vidissector input; 2) Pot box input; 3) Arm motion; 4) Display list generation.

This program was developed under contract with
Systems Concepts, Incorporated

1. MONITOR ROUTINES

Vidissector Control Routine(.VSCAN)

The ITS routine which is used to do the actual box scan is able to do perspective as well as nonperspective input. The system call to initiate a scan is as follows:

```
.VSCAN PTABLE
```

PTABLE is an 11_{10} word array which contains the information required for performing the scan. The format of each word of the table is described below:

0. CONTROL BITS,,VIDISSECTOR CONO
   At present only one control bit is used. This bit (4.9) determines whether or not the job should wait for the scan to complete. If the bit is Ø, no wait occurs and job is allowed to proceed and the interrupt routine handles the actual scan.
1. -COUNT,,ARRAY
   This is an AOBJN pointer as explained in the description of .VSCAN.
2. X RESOLUTION,,Y RESOLUTION
   Each 18-bit integer quantity gives the number of points to be scanned on the specified axis.
   The format of the following values is:

   INTEGER PART,,FRACTIONAL PART

3. R1
4. R2
5. C1
6. R3
7. R4
8. C2
9. P1
10. P2

. The above parameters are then used to compute the actual (X,Y) position of a scan point on the vidissector. The actual equations and method can best be explained by an ALGOL routine.

```
FOR Y1←0 STEP 1 UNTIL YRES-1 DO
BEGIN
    Y2←(2*Y1+1)/(2*YRES)
FOR X1←0 STEP 1 UNTIL XRES-1 DO
BEGIN
    Y2←(2*X1+1)/(2*XRES)
TEMP←P1*X2+P2*Y2+1
X←(R1*X2+R2*Y2+C1)/TEMP
Y←(R3*X2+R4*Y2+C2)/TEMP
ARRAY(Y1*XRES+X1)←SCAN(X,Y)
END
END
```

The .POTSET call

This call gives the user a flexible means of controlling program parameters via the input multiplexor. A maximum of 20 simultaneous connections between pots and variables is permitted. Each may be variable fixed or floating point. If fixed point, it may be an arbitrary byte within a word. The user may specify a mapping from pot values to variable values by giving an upper and lower limit. These may be inverted to give a backward mapping. Two types of control are provided, absolute and incremental. In absolute mode, the true pot position sets the value. This may be useful for positioning displayed objects with a joystick. The incremental mode permits a variable or set of variables to be changed slightly without causing a discontinuous jump in their values. The value is unchanged at connect time, but rotating the pot adds its scaled increment to the variable. Turning it down in the bottom third, or up in the top

third of the pots range causes a faster change so as to keep the control near center. The increase in gain is inhibited at low speeds to prevent drift due to noise.

The address of the call points to a table of pot connection specifications. This table consists of up to 20 blocks of 4 words, followed by a Ø word. The block format is as follows:

```
FOO/        Multiplexor channel,,control bits
FOO + 1/    Byte spec, variable address
FOO + 2/    Lower limit (value at pot = Ø)
FOO + 3/    Upper limit (value at pot = 10000)
```

Control Bits:

2.9 = delete only (if disconnect previous use of pot, no new attachment)

1.2 = absolute if ON, incremental if OFF

1.1 = floating if ON, fixed if OFF.

Byte Spec:

0 = full word; standard byte pointer format for partial word (no index or indirect allowed).

The following program will display a 45 degree vector of 177 on the scope and permit moving its origin with pots 142 (Y) and 143(X).

```
BEG:        .POTSET PTBL
            .DSTART [-3,DTBL-1]
            .VALUE
            .VALUE
PTBL:       142,,0                      ; CHAN142,FIXED PT., INCR.
            221200,,DTBL + 1            ; Y FIELD OF PT MODE WORD
            200                         ; VECTOR GOES 177 BELOW AND LEFT
                                          OF ORIGIN
            177                         ; TOP OF SCOPE
            143,,0                      ; CHAN143,FIXT PT., INCR.
```

```
            1200,,DTBL + 1              ; X FIELD OF PT MODE WORD
            200                         ; LOWER LIMIT X
            1777                        ; UPPER LIMIT X
            0                           ; END . POTSET TABLE
DTBL:       20117                       ; TO PT. MODE, SCALE Ø INTENSITY 7
            221000,,101000              ; START AT MIDDLE OF SCOPE, TO
                                          VECTOR MODE
            777777,,3000                ; ESCAPE, INTENS, DX=DY= -177,STOP
```

The .ARMOVE call

SERVO routines have been incorporated into the ITS Monitor. These routines provide acceleration and velocity limiting, software position limit stops, conversion from joint number to multiplexor channel, and some other utility functions. The user communicates with the SERVO routines by a series of one word commands. A block of commands is transmitted to the system at one time with a non-interruptible call as follows:

```
MOVE A,[-N,,ARMBLK]
.ARMOVE A,
```

This will interpret ARMBLK through ARMBLK+N-1 as SERVO commands with each word formatted as follows:

4.9-4.4	JOINT NUMBER
4.3-3.7	COMMAND
3.6	UNUSED
3.5	INDIRECT
3.4-3.1	INDEX
2.9-1.1	ADDRESS OR OPERAND

The currently available joint addresses are:

0 - AMF SWING
1 - AMF VERTICAL
2 - AMF HORIZONTAL
3 - AMF YAW (INOPERATIVE)
4 - ALLES HAND TILT
5 - ALLES HAND GRIP
6 - ALLES HAND ROTATE
7 - ALLES HAND EXTEND
10 - AMF ROLL (INOPERATIVE)

The currently available commands are:

0 - SET DESTINATION
1 - SET VELOCITY LIMIT
2 - TEST MAGNITUDE OF POSITION ERROR
3 - TEST MAGNITUDE OF VELOCITY

The operand is computed by adding the contents of the specified INDEX register (if XR≠0) to the right half of the command. The result is masked to 18 bits. If the indirect bit is ON, the right half of the word addressed is used as the operand. No further indexing or indirecting is interpreted.

The test commands are "TRUE" if the quantity tested exceeds the operand. If any tests are "TRUE" the .ARMOVE will skip. The SERVO routines are activated by the first .ARMOVE and are turned off by a .ARMOFF or killing the job.

A .ARMOVE is illegal if (1) the arm is in use by another user, (2) the block extends above the user's memory bound, (3) an indirect reference is out of bounds, (4) an unused command code or joint number is specified, or (5) the block is over 100 words long.

2. USER LEVEL ROUTINES

Vidissector Scan Routine

In order to facilitate the nonperspective use of ITS VIDISSECTOR SCAN ROUTINE (.VSCAN), a user routine has been provided to set up the input parameters for a box scan. This routine has the following call sequence:

```
MOVEI  R,PARAM
PUSHJ  P,VSCAN
```

PARAM is the address of a word table. The format of this table and the meaning of each entry is as follows:

NOTE: All elements except the first are in floating point format.

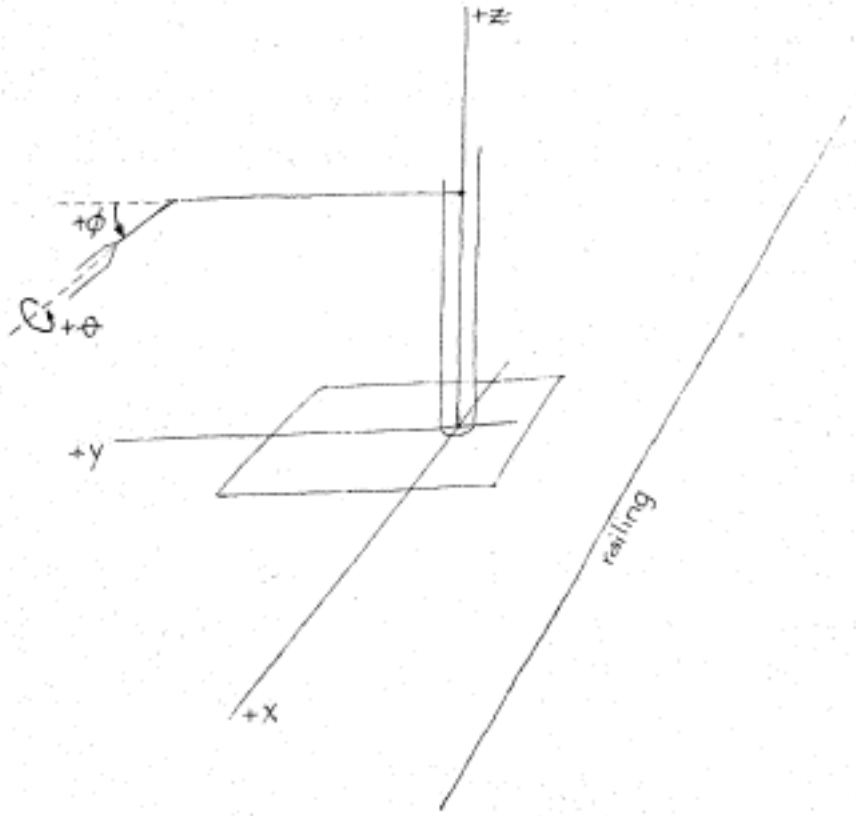
0. -SIZE,,DATA ARRAY
   This is an AOBJN pointer to the array where vidissector data should be stored. If the size specified for the array is too small to hold the requested data an error will occur and the vidissector will not be started.
1. X CENTER
2. Y CENTER
   Indicates the center position of the scan in vidissector coordinates.
3. X SIZE
4. Y SIZE
   Indicates the size of the box to be scanned in vidissector coordinates.
5. X RESOLUTION
6. Y RESOLUTION
   Gives the number of points to be scanned on each axis of the box.
7. ROTATION
   Gives the rotation of the box in radians.
8. SKEW
   Gives the skew of the box in radians.

Arm Control

As a means of simplifying user control over the arm, an interface routine is provided between the user and the system. This routine is called in the following way:

```
MOVEI R,ATABLE
PUSHJ P,WAVER
```

ATABLE is an array of new arm/hand position values. At present only 6 motions can be specified. (For purposes of simplicity, the hand is always kept pointing straight down). The user, when requesting motion, specifies the location (X,Y,Z) of the center of the gripper, as well as the orientation (rotate and tilt) of the hand and spread of the "claw." The base of the X,Y,Z coordinate system is assumed to be on the floor in the center of the swing axis. The axis are as shown below:



It should be pointed out that because of the geometry of the arm, a change in X,Y will cause a change of ϕ since an attempt is made to keep the proper orientation of the hand. If a user specifies a position outside the limit specified for the axis, the position will peg at the relevant limit. In addition, a null motion is represented by a word of all ones where the new position value is normally found. The current routine has two additional features:

1. An attempt is made to have all requested motions complete at the same time.
2. The WAVER subroutine returns to the user as soon as all motion has been started. The user can cause his program to wait for the arm motion to complete by doing a PUSHJ to the WVWAIT routine. Correspondingly, a change in tilt will cause corrections in the vertical and horizontal position so that the center of the "claw" will remain at the same point.

Display Routines

This package allows a user to build a display list which may consist of any of the following elements:

1. Control Information
   a. Intensity setting
   b. Scale setting
2. Point Display
3. Character Display
4. Vector Display.

The code is constructed in such a way that a user can either call a standard set of routines which perform the most commonly requested tasks, or else he can build his own special purpose package using the same basic primitives as does the standard package.

It should be noted that various routines use some or all of the registers W,WW,ZZ without restoring their original values. A good general rule is to not expect any of these registers to be preserved after a call to the display package. In addition, certain of the functions use other registers and this will be pointed out in the individual descriptions.

Initializing the Display Package.

The first thing that must be done is to assign a buffer to the display package and initialize its state.

This is accomplished by the following sequence:

```
MOVE W, [BUFFER START,,BUFFER END]
MOVEI WW,ERROREXIT
PUSHJ P,DSPRST
```

Register W must contain a pointer to the start of the buffer (in the left half) and a pointer to the last element of that buffer (in the right half).

Register WW must contain an address to be transferred to if the buffer should become full while generating the display list.

Upon returning from DSPRST, the "DISPLAY" will be in parameter mode, the scale will be Ø, and the size will be 7. At this point the user is ready to generate displays.

Display Modes

The current package is capable of putting the display in any one of four modes. This can be accomplished by a PUSHJ to an appropriate routine.

The modes and the proper routine used to place the display in that mode are as follows:

	Mode	Mode Setting Routine
1.	Parameter	MODPRM
2.	Point	MODPNT
3.	Character	MODCHR
4.	Vector	MODVEC

If the "DISPLAY" was already in the mode requested, nothing happens. In all other cases, any necessary clean up is performed before the mode is actually changed.

Parameter Mode Operations MODPRM

Two routines are provided for the express purpose of adjusting these quantities. It should also be noted that these routines make sure that the "DISPLAY" is in parameter mode.

Intensity. The calling sequence for an intensity change is:

```
MOVEI W, NEW INTENSITY
PUSHJ P,DSPBRT
```

Scale. The calling sequence for a scale change is:

```
MOVEI W, NEWSCALE-4
PUSHJ P,DSPSCL
```

For example:

if (W) = 20_8

the new scale is 1.

It should be pointed out that if the new scale or intensity is the same as the current value, the operation is a no-op. In addition, if the "DISPLAY" is currently in parameter mode, no additional half-words will be added to the display list.

Point Mode

In order to allow the same routines to be used for both positioning as well as displaying, the register Z controls itensifications: If (Z) = 0,

the display is positioned to the coordinates requested, but no intensification occurs. However, when (Z) = 2000, the point is displayed. NOTE: Any other value in Z will cause randomness.

The possible entry points and their meaning are as follows:

DSPFLT — The desired position is found in X and Y as floating point values. Registers A, B, C, are altered and not restored.

DSPPNT — Desired position is found in B and C as fixed point numbers.

DSPPTU — Same as above, except display is assumed to be in point mode. This entry is useful (and also slightly more efficient) when displaying a sequence of points.

NOTE: All values used by the routines mentioned above are truncated to 10 bits.

Character Mode

Two routines are provided for displaying characters:

DSPCHR — Displays a character at the current X,Y position. The character to be displayed is found in W.

DSPTXT — Displays a text string, produced by the ASCIZ pseudo-op starting at the most recent coordinates. A byte pointer to the string should be in W.

DSPMSG — Positions the display to the location specified in B and C (as fixed point coordinate values) and then displays a message as described in DSPTXT.

Both the DSPTXT and DSPMSG routines destroy the current contents of register A.

Vector Display

The entry point for displaying a vector is DSPVEC. The registers B and C contain the values ΔX and ΔY, respectively, with the current position of the display being used as the starting point on the vector.

Registers A,B, and C are destroyed by this routine.

Putting Arbitrary Commands into the Display

Although great care should be exercised in putting commands directly into the buffer, this facility is available through the routine DSPPUT.

This is accomplished by putting 18-bit value to be placed in the buffer into register W and calling DSPPUT. NOTE: Since no check is made on the input at this point, it is up to the user to make sure the display is in the proper mode.

Terminating the Display List

When a desired display list has been completely generated it must be terminated so that it can be displayed on the scope. This is accomplished by the routine DSPSTT. No arguments are transmitted, but a BLKO pointer to the current display list is returned in register W. This pointer can be passed to the system display processor (.DSTART) directly or it can be added to a list of BLKO pointers and then passed to the system as a part of the list.

*CORRECTION TO PAGE 5

Before The .ARMOVE call, after the program on Page 5 read:

Note that the program does not need to be running to move the line on the scope. The .POTSET call is illegal if either 1) the user tries to connect a pot already in use by another user, 2) the table is more than twenty blocks long, 3) twenty pots are already connected, or 4) the address exceeds the user's memory bound. A pot is disconnected when 1) the user disconnects it with a .POTSET, 2) the user reduces his memory bound below the address the pot controls, or 3) the job is killed.