VISION FLASH 27

USING THE VIDISECTOR AND THE STORED PICTURE FACILITY

by

Jerome P. Lerman

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Robotics Section

June, 1972

Abstract

The stored picture facility (FAKETV) allows LISP users, and to some extent machine language users, to access a library of stored images rather than live vidisector scenes. The vidisector functions in LISP have been slightly restructured so that input from stored images or live images can be handled with no changes to the user's program. The procedure for creating stored images is also described.

Work reported herein was conducted at the Artificial Intelligence Laboratory, a Massachusetts Institute of Technology research program supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored by the Office of Naval Research under Contract Number N00014-70-A-0362-0003.

Vision flashes are informal papers intended for internal use.

This memo is located in TJ6-able form on file VIS;VF27 >.

1.0 INTRODUCTION

The stored picture facility (also called the FAKETV or FTV) gives LISP and machine language users the ability to access images that have been stored on disk or tape rather than live TV images. The FTV has been designed so that LISP programs and to some extent machine language programs can switch from stored to live pictures with no program modification. Except for some minor restrictions, input from the FAKETV can be treated exactly as input from the real vidisector. The access time per image point is commensurate. and in some cases, the FAKETV may even be faster.

'Hard' copies of images have proved invaluable in debugging programs that use vidisector input, since reproducible data allows bugs to be repeated and hunted down. A permanent library of documented scenes is available for comparing the performance of different programs. The user is freed from the vagaries of the vidisector hardware and any time-varying properties it may have.

This memo is aimed primarily at LISP users, but it contains information about the vidisector and FAKETV that may be of use to machine language programmers too.

2.0 SOME FACTS ABOUT THE VIDISECTOR

A user normally obtains intensity information about a visual scene through the image dissector camera and its associated hardware and software. For all the grubby details, consult AI Memo #178, but the casual user can probably survive with the following information:

1. The range of vidisector coordinates is 0 to 16,383. (14 bits)

2. The vidisector returns values that are inverse intensities with a range of values from 0 to 704. (1300 octal). For example, 0 is the brightest possible point and 704. is the dimmest.

3. The values obtained from the vidisector may correspond directly to the inverse intensity or to the LOG of the inverse intensity. The LOG is most often used and may be obtained in LISP by using the function NVFIX (see Section 3.0).

4. The log value is 64. x log intensity (to the base 2).

5. The ability of the vidisector to resolve points is such that it makes sense to look at every sixteenth point, forming a 1024. x 1024. point image array (for a total of about 1 million image points) instead of a 16,384. x 16,384. point image array.

6. The 'best' pictures are obtained when the confidence level, CFL, is set to three, and the dim cutoff level, DCL, is set to one. These settings will produce an image with the best

signal-to-noise ratio and the widest range of values.

7. With CFL=3 and DCL=1, the values returned by the vidisector will be in the range 200-1200 (octal). This means that vidisector values can be encoded with 9 bits of information, since 9 bits will represent 777 values.

See Section 7.0 for detailed instructions for turning the vidisector on and off.

3.0 USING THE VIDISECTOR FROM LISP

LISP has three functions that are relevant to operating the vidisector. They are:

1) NVSET
2) NVFIX
3) NVID

3.1 The NVSET Function

(NVSET NIL CFL RES DCL NIL)

where CFL = the confidence level. CFL=3 is the slowest but most accurate. CFL=0 is fastest but noisiest.

RES = the number of divisions into which each line and column is partitioned. Max is 16,384. For example, if RES=1024., this implies that the vidisector field of view is divided into a 1024. x 1024. grid.

DCL = the dim cutoff value. In most cases, DCL=1 is a good choice. See A.I. Memo 178 for more about dim cutoff.

The NVSET function ignores any arguments that are set to NIL. Thus, the first and last arguments shown above have no effect. These arguments are relics of past features that have become defunct, and are being reserved for some anticipated new features.

NVSET returns as its value a list of the current vidisector parameters. If NVSET is called with all arguments set to NIL, it will have no effect on the parameter values, but will merely serve to obtain a list of their current values. For example, if

```
(NVSET NIL 3. 1024. 1. NIL)
```

is evaluated, it will return

```
(NIL 3. 1024. 1. NIL)
```

If at some later point in the program, one would like to see what the current confidence level is (without changing it), evaluate

```
(NVSET NIL NIL NIL NIL NIL)
```

which will return

```
(NIL 3. 1024. 1. NIL)
```

3.2 The NVFIX Function

```
(NVFIX  XCOR  YCOR)
```

This function will return a fixed point value that is proportional to the LOG of the inverse intensity at point

XCOR,YCOR. The LIN-LOG switch should be in LOG mode. A value of 0 is returned for points that are dim cutoff.

3.3 The NVID Function

(NVID XCOR YCOR)

This function will return a floating point value that is an inverse linear measure of light intensity at the point XCOR,YCOR. The LIN-LOG switch on the vidisector should be in LIN mode. A value of -1.0 is returned for points that are dim cutoff.

3.4 Error Messages

If the user attempts to access a vidisector point that does not exist, LISP will generate a correctable error of the FAIL-ACT type (see recent LISP documentation for explanation of error handling). To proceed from a FAIL-ACT error, one returns a list of some form to be EVAL'ed in place of the one that failed in action. If the user proceeds through the interrupt without correcting the error, the message

NON-EXISTENT VIDI POINT

will be typed.

4.0 FORMAT OF STORED IMAGES

In order to use a stored image, the image file as dumped by the program CAMERA (see Section 6.0) must reside somewhere on disk. Whenever a vidisector point is requested, the proper value must be retrieved from the disk file and returned to the user. Since a typical stored image might occupy 40 pages of core (or 40 disk tracks), it is not desirable to keep the entire image in core. On the other hand, reading from the disk is a time-consuming operation, so it would be efficient to minimize disk accesses. In order to effect a compromise, the FAKETV images are stored as a series of square regions. Each patch is 64. x 64. vidisector points, and at least one such patch resides in core. Whenever a vidisector point is accessed, the entire patch that contains it will be read into core. Thus, examining nearby vidisector points will not cause a disk access except if the points straddle the boundary between two regions. Usually, LISP maintains 4 patches of the image in core at one time, although this number can be altered (see Section 5.2). If the requested point is not contained in one of the resident patches, a new patch will be read in on top of one of the old patches.

Each image patch contains 64. x 64. = 4096. vidisector points. Since each point requires 9 bits, 4 can be packed in each 36 bit computer word. One patch, then, can be stored in 1024. words. This is equal to the size of a page of core memory and the length of a disk track. The job of selecting the proper patch of the image and reading it into core, if it is not already there, and unpacking the required 9 bits of data is handled by the

FAKETV software. This operation is transparent to the user, and he will not know when new patches are being read into or flushed from core. The 9 bits of vidisector data once unpacked, are expanded into a 36 bit word that is identical in form to the words returned by the real vidisector.

An entire vidisector frame is treated as a 1024. x 1024. grid. If the user demands higher resolution, he will get the value of the nearest grid point. No interpolation is done.

At 64. x 64. points per patch, it would require 256 disk tracks to store an entire image. It would be cumbersome to store such large files, so the CAMERA program has provisions for selecting a specified portion, or window, of the frame for recording. The first patch of each image is reserved as a header, and it contains information about where the window is located in the vidisector frame, the dim cutoff and confidence level at which the image was recorded, and a title. The title must be a LISP S-expression of less than 5000. ASCII characters. It is possible for LISP to access the information in the header for the user's enlightenment (see Section 5.3).

Stored images often will be large disk files. For this reason, the suggested method for using the FAKETV is to keep images stored on micro- or macro-tapes, read them onto disk when they are to be used, and delete them from the directory after use. Only those stored images that are being used every day should be allowed to remain on the disk. Since stored images are binary files, the TECO command E_ (E<shift O>) can be used to move them back and forth between disk and microtape. The system program DUMP is used to transfer files between disk and magtape.

5.0 USING THE STORED PICTURE FACILITY IN LISP

5.1 Selecting the FAKETV

When LISP is first loaded, it is initialized for live vidisector input. In order to cause LISP to get vidisector data from a stored image, one must evaluate

```
(SSTATUS FTV FILNM1 FILNM2 DEV USR)
```

where FILNM1 FILNM2 is the name of the stored image file created by the CAMERA program (e.g. POINTS CUBE), DEV is the device where the file is expected to reside, and USR is the user directory to search for the specified file name.

As its value, the SSTATUS will return

```
( (XLL  YLL) (XUR  YUR) FILNM1 FILNM2 DEV USR)
```

where

XLL = the X coordinate of the lower-left corner of the window
YLL = the Y coordinate of the lower-left corner
XUR = the X coordinate of the upper-right corner of the window
YUR = the Y coordinate of the upper-right corner
FILNM1 FILNM2 DEV USR = the file specification of the currently opened FAKETV image. If the specified image file can not be

found, LISP will cause an error of the FAIL-ACT type.

The window coordinates are scaled to the current resolution as set by an NVSET, e.g. if RES = 1024. and XLL = 100., then if RES is changed to 2048., XLL will be returned as 200.

To cause LISP to revert back to real vidisector input, evaluate:

```
(SSTATUS FTV)
```

At any time, the state of the vidisector (whether live or stored input) can be ascertained by evaluating

```
(STATUS FTV) Note: Status not SStatus
```

If the value is NIL, then input is live. If the value is non-NIL, it will be the window coordinates and file specification of the currently active stored image as described above for SSTATUS.

5.2 Setting the Number of Patches Residing in Core

At any time, it is possible to change the number of patches that will reside in core. By default, there will be 4 sub-images resident. This may be changed to any number between 1 and 200 by

```
(SSTATUS FTVSIZE NUMBER)
```

where NUMBER = the desired number of resident patches.

5.3 Reading The Image Title

In order to read the S-expression that may reside in the header of the stored image, one can evaluate

```
                (STATUS FTVTITLE)
```

This will return NIL if there is no image title, or it will return a complete S-expression if one exists. For example, if the image header contains the expression

```
        (SETQ CALIBRATION '( 1.0    2.0    3.0    4.0
                             5.0    6.0    7.0    8.0
                             9.0   10.0   11.0   12.0
                            13.0   14.0   15.0   16.0) )
```

then

```
                (EVAL (STATUS FTVTITLE))
```

will cause the variable CALIBRATION to be set to the specified list.

5.4 Restrictions and Cautions About Using the FAKETV

Once a stored image has been selected, programs may use the vidisector functions in the normal manner with the following exceptions and cautions.

1) NVSETing the resolution to values higher than 1024. will be of limited value. since the stored images are recorded on a 1024. by 1024. grid, higher resolution will cause the FAKETV to select the nearest point on the grid. For example, if the resolution was set to 2048., then (NVFIX 0 0) and (NVFIX 1 0) will return the exact same value.

2) NVSETing the confidence level when using stored images

has no effect. NVSETing the DCL will cause the values returned by the FAKETV to change just as if they were inputted from the real TV.

3) NVFIX or NVID will not return a value that indicates that a dim cutoff has occurred. Instead, the darkest possible value (as determined by the current DCL and CFL) will be returned.

4) If the vidisector value of a point outside the window of the stored image is requested, LISP will generate a FAIL-ACT type of correctable error. The user can supply a function to handle such errors by returning a value such as the dim cutoff value, or a marker value such as -1, instead of accepting the error. If the user does not correct the error, the message

NON-EXISTENT VIDI POINT

will be typed.

5.5 Some FAKETV File Name Conventions

For normal usage of the FAKETV, the following file naming conventions are suggested:

1) A stored picture is called POINTS xxxxxx, where xxxxxx is a user supplied name, e.g. POINTS CUBE.

2) A line drawing made from a stored picture called POINTS xxxxxx is called LINES xxxxxx, e.g. LINES CUBE.

3) A file for intensity modulated display of a stored image (see Vision Flash 25) is called IMAGE xxxxxx, e.g. IMAGE

CUBE.

6.0 HOW TO CREATE A STORED IMAGE

CAMERA is an easy to use program for creating image files for the stored picture facility. It interacts with the user by asking a series of questions which can usually be answered with a Y (for YES) or N (for NO). Typing |G will cause the program to start over, and typing |X causes it to kill itself.

1) To load and start the program, type:

```
CAMERA|K
```

2) The program will begin by asking if the PDP10 (the time- sharing system) is to handle the vidisecting. If the PDP6 is operational, the answer to this question should be no. This will cause the slave program to be automatically loaded into the PDP6. Loading is completed when the next question is typed. The user must make sure the PDP6 is running by doing the following:

Press the STOP switch
Press the I/O RESET switch
Set the address switches to 6000 (octal)
Press START

If the address switches are now set to zero (all of them down), the register lights should be blinking regularly. This indicates that the slave is alive and well.

3) After determing which processor is to do the vidisecting, the program asks for the file name of the image to be dumped. It should have the following familiar form:

FILNM1 FILNM2 DEV: USER;

by default, DEV will be DSK and USER will be VIS.

Dumping is allowed on the following devices:

DSK: UTn:

4) The program will now ask if a window should be computed. If the answer is no, go to (5). If the answer is yes, the line finder (Horn-Binford variety) will be activated. A coarse line drawing will be computed and the program will attempt to put a window around the objects of interest. The line drawing and window will be displayed on the 340. When the window is completed, the program will ask for approval. If the window is satisfactory, type Y and proceed to (6). If the window is unsatisfactory (a likely occurrence), type N. This will enable the old pot box to control the position of the window. Experiment to determine which pots affect the left, right, top and bottom margin of the window. When the window is correctly positioned as shown by the display, type Y and go to (6). The pots are now disconnected so don't worry about messing up the window position.

5) If the line finder isn't used to compute a window, the program will ask if the entire frame is to be scanned. If this is desired, type Y and go to (6). Think twice about scanning an entire vidisector frame since it will take a long time (about 45

minutes) and will consume 256 blocks of disk space. If only part of the frame is wanted, type N. The program will now ask for the lower left and upper right coordinates of the window. Type the desired values (from 0 to 1023.) followed by a space. Note that numbers must be followed by a decimal point to be interpreted to the base 10. Otherwise, they will be considered as octal numbers.

6) When the window decision has been made, the program will type out the name of the file being dumped, the window parameters, the confidence level and dim cutoff level.

7) Next, the user is asked if an image title is desired. If the answer is no, go to (8). If a title is desired, the user must type an S-expression suitable for input to LISP. The title will be terminated when the parentheses balance. Since the header S-expression may be read and evaluated by LISP, the user should be cautious about inserting periods that may cause dot context errors, or macro characters (such as semi-colon) that may make the string unreadable.

It is often useful to include the calibration vector with a stored image file. This vector, consisting of sixteen entries, relates the origin of the vidisector frame of reference to the arm's frame of reference. It also contains the coefficients of the equation that describes the table plane. The calibration vector, if it exists, resides on disk as a file called

```
HORN TRANS VIS; DSK:
```

If the user wishes to include it in the image header, he may do so by inserting a number sign character, #, anywhere in the title string. It will be replaced by the sixteen numbers as read from the disk file. The user must make sure that the HORN TRANS file contains a valid calibration vector.

For example, typing

```
        ( (COMMENT THIS IS A TEXTURED CUBE)
         (CALIBRATION #) )
```

will cause the following to appear as the header string:

```
  ((COMMENT THIS IS A TEXTURED CUBE)
   (CALIBRATION   -79.77    408.2        -57.54      8615.0
                 -279.76      0.22       281.44     13001.8
                   -0.14603   1.602E-2    -0.10723     16.0
                    1.0E-3    1.2E-2       1.0          8.32 ))
```

8) After the title has been typed, the program will type

```
        SCANNING
```

to indicate that the vidisector is in operation. When dumping is completed, the total number of blocks that have been written will be typed. The name of another image file will be requested and the procedure will start over at step (3). To kill the program, type |X.

9) All pictures are dumped with the dim cutoff level, DCL, set to 1, and the confidence level, CFL, set to three. These parameters can not be changed in the program, since any other values will cause CAMERA to produce poor results.

7.0 OPERATING THE VIDISECTOR

CAUTION 1: Before stepping in front of the vidisector, put the lens cover on. Bright objects such as eyeglasses, belt buckles etc. can damage the photocathode.
CAUTION 2: Don't cycle the power on-off switch. Wait a minute or so between changing state to allow the high voltages to stabilize.

7.1 Turning on the Vidisector

1. Move the vidisector to the desired position and lock the dolly. Standard position may be obtained when the dolly is aligned with the marks on the floor and the height and camera angle are adjusted so that the yellow tape marks on the dolly frame are aligned.

2. Turn on 400 Hz generator using switch in the circuit breaker box. Turn off the overhead flourescent lights in order to minimize 60 Hz noise.

3. Turn on sunguns using the switches on their cords.

4. Turn on the vidisector by moving the switch marked RESET-DESTRUCT to DESTRUCT.

5. Make sure LOG-LIN switch is in LOG position if using CAMERA or LISP's NVFIX function.

6. Set up the desired scene while the vidisector warms

up. Before removing the lens cover from the camera, make sure no shiny objects are within the field of view.

7. The pushbutton switch marked RASTER ON switches the vidisector between MONITOR mode and random access COMPUTER mode. Put it in MONITOR mode and use the 3 rightmost pointed brown knobs to bring a small portion of the scene (an edge for example) into view.

8. Use the toggle switch marked FOCUS on the end of a cable attached to the vidisector to optimize focus and the switch marked IRIS to set the iris opening.

9. Switch back into COMPUTER mode.

10. The vidisector is now ready to scan the scene

7.2 Turning Off the Vidisector

1. Replace the lens cover.

2. Turn off the vidisector power by switching the RESET-DESTRUCT switch to RESET.

3. Turn off the sunguns, and then turn off the 400 Hz power. It is important to remember to turn off each individual sungun, since if they are on when the 400 Hz power is switched on, they may be burned out by the startup surge.

APPENDIX I

THE STORED PICTURE LIBRARY (as of 6/15/72)

All images are stored on DSK: VIS; unless otherwise noted

POINTS A

A single rectangular white brick lying down. The object is made of slightly translucent painted plastic.

POINTS B

Two disjoint white bricks lying down. The left brick is painted plastic (same as in POINTS A) and the right brick is sanded plaster.

POINTS C

A scene consisting of three white bricks. One brick is standing up, and is partially obscured by a lying down brick. This brick supports another brick which lies on top of it without obscuring the standing brick. A complicated scene that may contain some shadows.

POINTS ARCH1

An arch made of two standing bricks that support a third brick. The top brick overhangs the left support slightly.

POINTS WOOD1

A complicated scene consisting of two disjoint rectangular stacks of wooden objects similar to SOMA cubes. There is a lot of visible wood grain.

PICTUR STRIPE

A single white rectangular brick standing up. The sides are covered with stripes made by a Flair pen. Each face has stripes that are in a different direction.

POINTS TCUBE

A single standing rectangular white brick. The faces of the brick are covered with short (about 1/4 inch) line segments made with a blue flair pen. The line segments are in random directions and have uniform intensity over each face. No line segments cross the boundaries between faces.

APPENDIX II

USING THE FAKETV WITH MIDAS CODE

A collection of MIDAS routines for accesing FAKETV images can be obtained by inserting the file

FAKETV INSERT DSK: VIS;

into machine language programs. The routines contained in this package will allow .VSCANs and calls to the TVC or NVID devices to be simulated. Also, routines are available for accessing the information contained in the image header. The routines are extensively commented. For assistance, see Jerome Lerman.