A Stored Picture Hacking Facility

VISION FLASH 25

by

Sidney Markowitz

Massachusetts Institute of Technology

Artificial Intelligence Laboratory

Robotics Section

JUNE 1972

Abstract

A short description of LISP functions that have been written for use with the stored picture facility. These functions allow one to display an image of a stored scene on the 340 scope, and produce graphs and histograms of intensity functions of portions of the scene.

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;VF25 >.

0.

This paper assumes some familiarity with the LISP display slave functions. The paper can be understood without this knowledge, if the reader does not worry about the definition of the term "display item" or "item number". Familiarity with the display functions is necessary for practical application of this paper. A description of the LISP display slave functions can currently be found somewhere in the file .INFO.; LISP ARCHIVE.

The functions described here can be found in VIS; DISPLY HACKS.

In the descriptions of functions, the following conventions are used:

All functions are EXPR's unless otherwise indicated.

Atoms written in capital letters are to be entered as written.

Bracketed names represent s-expressions to be chosen by the user.

Thus, the LISP function PLUS would be described as:

(PLUS [value1] [value2] ...)

1. Initialization

(SHOW [pointfile] [window] [range-sweep] [x-grain] [y-grain])

This function initializes the display slave, and displays the image of a stored scene, e.g., '(POINTS CUBE DSK VIS), on the 340.

Arguments:

[pointfile]
name of file containing stored scene

[window]
(([xll] [yll])([xur] [yur])), list of two lists indicating coordinates of lower-left and upper-right corners of scene to be displayed. Of course, no more of the scene than has been stored can be displayed. NIL means display allof the scene that is on the disk file.

[range-sweep]
If a FIXNUM value, the function initially sweeps down and across the middle of the scene, looking at the intensity of every [range-sweep]th point. The obtained values are used for scaling purposes in the display. A typical value is 5. A larger value is likely to miss a bright or dark area, while a smaller value takes more time.
If the lines through the middle of the scene are not representative of the scene as a whole, leading to a poorly scaled picture, scaling can be forced by passing a [range-sweep] equal to a two-list ([low] [high]) of lowest (brightest) value and the highest (darkest) non-dim cutoff value returned by NVID in the scene.

[x-grain]
[y-grain]
FIXNUM values. In creating the display, the function uses every [x-grain]th point in horizontal scans of every [y-grain]th line. The smaller these values, the more detail can be seen in the display, and the more flicker there is.
2 and 2 will provide all the detail one needs and perhaps a tolerable amount of flicker.
4 and 4 will give very little flicker and probably tolerable detail.
3 and 3 is a useful compromise.

Output:

After the initial rangeing sweep, the function prints a list of low and high NVID values found.

Returns:

Item number of created display (always 1).

Typical call:

Since SHOW takes a while to execute, it should be done once, and then the resulting display list dumped on the disk. The following example does just that:

```
(DUMPARRAYS (LIST (DISGORGE (SHOW '(POINTS A DSK VIS)
                                  NIL 5 3 3)))
                  '(IMAGE A DSK VIS))
```

(TVINIT [filename]) FEXPR

Assuming that

1. The stored scene is located in a file with a first name of POINTS, e.g., POINTS FOO DSK BAR
2. A previous call to SHOW has stored a display list of the image of the scene in a file with a first name of IMAGE and the rest of the names the same, e.g., IMAGE FOO DSK BAR

then this function will display the scene and initialize the stored picture facility.

Typical call: (TVINIT IMAGE A DSK VIS)

If either of the previous assumptions are not true, the following sequence of calls will achieve the same result. Assume that the point file is in FOO BAR DSK VIS and the image file is in RANDOM IMAGE DSK HACK.

```
(SSTATUS FTV FOO BAR DSK VIS)
(NVSET NIL NIL 1024. NIL NIL)
(DISINI)
(MAPCAR (FUNCTION (LAMBDA (FOO)
                          (DISGOBBLE (CAR FOO))))
        (LOADARRAYS '(RANDOM IMAGE DSK HACK)))
```

2. Picture Hacking Funtions

A. Getting a list of points.

(FIND)

displays a blinking cross, which can be moved around with space-war console number 1. Displays a point whenever the button on the console is pressed.

Returns:
a list of all such points each of the form list ([x] [y]), when the button is pressed twice (with at least 1 second pause) in succession at the same point.

(LINE [endpoints] [numpoints])

Arguments:

[endpoints]

list of two points in same format as returned by (FIND)

[numpoints]

FIXNUM value

Returns:

List of [numpoints] points on line between the [endpoints]

(CIRCLE [center] [radius] [numpoints])

Arguments:

[center]

list of list (x y) of coordinates of center point, in same format as returned by (FIND).

[radius]
[numpoints]

FIXNUM values

Returns:
List of [numpoints] points of circle around [center] with radius [radius].

(AREA [points] [num1] [num2])

Arguments:

[points]
list of coordinates of three points, in same format as returned by (FIND). For purposes of this description call them point1, point2, and point3.

[num1]
[num2]
FIXNUM values

Returns:
List of points in area contained by a parallelogram with sides point1-point2 and point2-point3. Side point1-point2 will contain [num1] points and side point2-point3 will contain [num2] points, so the returned list will contain (* [num1] [num2]) points.

B. Getting a function of intensity of a list of points.

(POINTOUT [itemname] [points] [predicate]) FEXPR

Optionally displays list of points, and returns list of application of [predicate] to those points (e.g., intensity at those points).

Arguments:

[itemname]
If NIL indicates that the points are not to be displayed.
Otherwise, it is an atom to be SETQ'd to the number of the display item created by this function. Note that since this atom is not EVAL'd, it does not require a value before the function call.

[points]
list of points in same format as return of LINE, CIRCLE, and AREA. This argument is EVAL'd.

[predicate]
(optional) Function of two FIXNUM arguments (x y) to be applied to each point. If [predicate] not specified, defaults to:

```
'(LAMBDA (X Y) (- 1024. (NVFIX X Y)))
```

which returns intensity at point (X Y).
This argument is EVAL'd.

Returns:
List of results of applying [predicate] to each point, with side effect of displaying the points on the scope. The following will do the same thing without displaying the points:

```
(MAPCAR '(LAMBDA (POINT)([predicate] (FIX (PLUS 0.5 (CAR POINT)))
                                     (FIX (PLUS 0.5 (CADR POINT)))))
        [points])
```

or just plain

```
(MAPCAR '(LAMBDA (POINT)(APPLY '([predicate] POINT)) [points])
```

if [points] has only FIXNUM values.

(HISTOGRAM [points] [range][pred])

Arguments:

[points]
list of points in same format as returned by LINE, CIRCLE, AREA.

[range]
([low] [high]), two-list indicating range of values in [points] list.

[pred]
(optional) Function of two FIXNUM arguments (x y) to be applied to each point in [points]. If [pred] not specified, defaults to:
'(LAMBDA (X Y) (- 1024. (NVFIX X Y)))

Returns:
List of two-lists ([value] [numpoints]). [value]s are the integers sequentially from [low] to [high]. [numpoints] is the number of points for which [pred] returns a value] within 0.5 of [value].

NOTE:
For use with vidisector values, the return of this function should probably be smoothed with a moving average type function, to eliminate noise effects.

3. Graphing Results.

The function (GRAPH) is located in COM:GRAPH > and is documented in DSK:SID;GRMEM >. It is a generalized graph display and plotting routine.

Typical call:
To graph the intensity along a line of a scene, specifying the line with the space war console, say:

```
(GRAPH (POINTOUT ITEM (LINE (FIND) 40.))
       '((0 0)(300. 300.)) NIL NIL)
```

This leads to a pretty picture. The meaning of the various arguments will be found in the graph routine memo.

4. Random

(DRAWLINES) FEXPR
can be used for drawing outlines of figures for later plotting. It displays a blinking cross which can be moved around with the space-war console. Points are marked by pressing the console button, and successive points are joined by straight lines. Pressing the button twice at the same point will cause no line to be drawn from there to the next indicated point. Pressing the button three times in succession at the same point will cause the function to return the display item number of the lines.

(DRAWLINES [points])
takes a list of 4-lists of the form ([x1] [y1] [x2] [y2]) ([x3] [y3] [x4] [y4]) ...) each element of which specifies the endpoints of a line to be drawn. Thus, the following example will display linefinder output that has been stored in file LINES FOO DSK VIS.

```
(UREAD LINES FOO DSK VIS)
((LAMBDA(↑Q) (DRAWLINES (READ))) T)
```