McCULLOCH LABORATORY

VISION GROUP

FLASH #1

THE L%LINES PACKAGE

A. K. Griffith

Summary: The program (L%LINES X Y) takes feature point output from the FP%FPOINTS program (q.v.) for horizontal and vertical scans (X and Y respectively); and outputs a list consisting of two lists of line segments, represented in an obvious manner, obtained from the respective arguments. "Feature points" are points on the field of view which seem to lie along some edge in the scene. The line segments output by L%LINES are obtained by examining a set of feature points for straight chains of points.

Note: Sections marked with a star (*) are for reference; the user need only study the unstarred sections. Starred sections should be examined along with a listing of the uncompiled version of the L%LINES programs.

I. CALLING SEQUENCE

to use the L%LINES package, the user need understand only the following calling sequence:

(L%LINES X Y)

where:

X is the value of (F%FEATUREPOINTS N 0) (or (F%DFEATUREPOINTS N 0)), for some N, ("feature points" obtained from a set of horizintal scans - see the write up on the F%FPOINTS package).

Y is the value of (F%FEATUREPOINTS N 1) (or (F%DFEATUREPOINTS N 1)).

The value of (L%LINES X Y) is a list consisting of two lists obtained independently from the respective arguments, of line segment end point pairs.

A typical output might look like:

```
((((69.  450.)(33.  247.))
  ((410.  72.)(66.  123.))

      ...........

  ((23.  469.)(224.  71.)))
 (  ...............)).
```

The arguments given to L%LINES may be in "compact" or "uncompact" format. (See the F%FPOINTS Write-up.) L%LINES figures out which format the arguments are in, and acts appropriately.

Vertex information may be obtained by linking these lines by means of the L%LINKS package (q.v.).

II. PURPOSE

The input to the program consists of two sets of "feature points" obtained from horizontal and vertical intensity scans respectively. "Feature points" are locations at which the scans are adjudged to intersect with edge lines in the field scanned. From the feature points obtained from horizontal scans, the L%LINES program tries to extract straight lines segments which make angles between -45 and +45 degrees with the vertical. Similarly, from the other set of feature points, L%LINES extracts segments making angles between -45 and +45 degrees with the horizontal. A list of these two sets of lines constitutes the output of the L%LINES program.

III. AVAILABILITY

link to, or obtain the file L% >. Read it into a LISP, NLISP or DLISP containing a LAP. (q.v.). The uncompiled version requires about 1,000 words of binary program space, and about 2,400. words of free stoage. When compiled, the program requires about 1,900 words of binary program space. Additional space, both free storage and binary program, is required for the input data and intermediate values of processing. The quantity depends on the size of the arguments. The binary program space used for arguments and intermediate values is automatically reclaimed by the program, unless the function L%LINES aborts for some reason. If the latter occurrs, or if a quit is performed during the execution of L%LINES, the binary program space in use may be reclaimed by executing the function (L%RECLAIM).

IV. ERRORS AND ECCENTRICITIES

1)(BAD INPUT FORMAT) is typed by L%LINES if either argument is in an incorrect format, e.g. if it was produced by some program other than F%FPOINTS.

2)(SPACE SCARCE) is typed by L%LINES signalling a fatal space allocation disaster. If this ever happens, see me.

3)(NO CORE TYPE NON NIL TO TRY AGAIN) is typed by L%LINES as a result of attempting to allocate more binary program space when no core is available. If this happens, try to flush your other jobs, other users, etc. and try again. If enough core has been freed, usually only a few thousand words, then L%LINES will continue after you type something at it.

If any error occurrs, you should type (L%RECLAIM), as previously noted, to get back any reclaimable binary program space.

IV. *INPUT FORMAT

The first argument should be in the form:

```
((Y0 PX01 PX01 ...)
 (Y1 PX11 PX12 ...)
   .........
 (Yn PXn1 PXn2 ...)),      (4.1)
```

where Yi = iA for some A, 500.-A< Yn <500., and n>25. An element (Yi PXi1 PXi2 ...) is derived from the i-th horizontal scan across a 500. by 500. field, at a vertical distance Yi. A Pij may be in one of two forms, either a list of four numbers, or a single full word. If the "uncompact" form of a Pij is:

```
(A B C D),
```

then the corresponding full word ("compact") form is as diagrammed below:

```
(max 37777 A) B C (min 777(max 0(plus D 400)))

    14 bits   9 bits   4 bits   9 bits.
```

These four quantities refer to some point in the plane with X-coordinate B, which appears to occur at an edge of type C having amplitude D. The value of a gives a measure of the probability that there really is an edge through the point in question.

V. *SPECIAL DATA TYPES

A list of whole numbers, which is not subject to manipulation, e.g. CADR of

(10.(2043. 6724. 10000.))

is conveniently stored in consecutive words of binary program space with a suitable delimiter:

10005.	2043.
10006.	6724.
10007.	10000.
10008.	0.

and represented to LISP programs by a standard integer version of the starting address. The above list would thuus beccome:

(10. 10005.)

This has obvious advantages for saving space. A similar data type for half words would store the list:

(10.(60. 240. 163.))

as:

2703.	60.,,240.
2704.	163.,,777777

making the above list:

(10. 2703.).

L%LINES uses these data types extensively. We refer to the blocks of locations in binary program space as "full word blocks" and "half word blocks", and the LISP pointers as "full word block pointers" and "half word block pointers" respectively.

The function (L%ALLOC N) makes sure that there are at least N words of binary program space available for storing data types just mentioned, and sets a storage pointer to the current value of BPORG.

The function L%FPSTORE takes a list in the form described in section IV, and returns a list of the form:

```
((Y1 P1)
 (Y2 P2)
 .......
 (Yn Pn)),
```

where the Pi's are full word block pointers which point to blocks in which are stored the full word equivalents of the corresponding set of feature point lists.

VI. *PRINCIPAL SUB FUNCTIONS

the main program, L%LINES, takes each of the two arguments and first applies L%SFILL. The result of this is to fill arrays L%SLOPES and L%DENS with data structures composed mainly of full and half word pointers. The program L%SA acts on the values in this array, and its value, processed by L%ENDS consists in one of the sets of lines output by L%LINES. The other set of lines is similarly generated from the other argument, except that x and y are interchanged in the resulting segment end points.

(L%SFILL N) takes as argument 0 or 1 respectively corresponding to its use of horizontal-scan feature point information (the first argument of L%LINES), or vertical-scan feature point information. It performs an L%PROJ (cf. below) on the feature points for slope arguments ranging between the values of L%SLLO and L%SLHI (usually 0 and 500.) in increments of L%SLINC units (normally 5). The normal values of L%SLLO L%SLHI and L%SLINC result in angular projection of the feature point raster at angles between -45 and +45 degrees with the vertical at intervals of about one degree. For a given value of SL, (L%PROJ SL)returns a list of x-intercepts of the lines extracted from the feature point raster which make an angle corresponding to SL with the vertical. (The intercepts are actually augmented by 250. To be positive). More exactly, the output of L%PROJ is, e.g.:

```
((10. 280.)(53. 720.) ...)
```

which indicates that there exists a line wihich intersects the line y=250. at an x co ordinate of 30.(=280.-250.) and which has a relative strength of 10. Similarly, there appears to exist a line at the given angle which has the described intercept of 470. and strength of 53. The threshold on these strength values is given by:

```
(PLUS L%P2(TIMES L%P1 N))
```

where N is the number of scan lines. The strength value is approximately twice the sum of weights of the feature points in a narrow band (about 4 units in width) with indicated slope and intercept. A weight for a given point is:

```
(ADD1(MAX 3(*QUO M 16.)))
```

where M is the "probability" denoted by A in the feature point explanation given in section IV. For each value of SL, each element of the output of (L%PROJ SL) is operated on by a function L%CHOP, which extracts the entire set of feature points lying within a narrow band of slope and intercepts given respectively by the argument of L%PROJ and by the output of L%PROJ; and cuts it into "segments" consisting of strings of adjacent feature points. These segments look like this:

(density sl int ptr) (6.1)

where "density" is related to the number of feature points in the segment, and the "ptr" is a half-word binary program space pointer pointing to a block of half words which are the absolute addresses where the feature points of the segment are stored. The "density" is thresholded at a value given by:

(PLUS L%P4(TIMES L%P3 N))

where N is the number of scen lines taken. The "density" value is approximately one half of the sum of weights of feature points in a segment, where the "weights" are as defined as before. The segments are extracted from a region on the feature point raster with the given slope and intercept, and with a width given by the value of L%LW. These segments are stored in an array L%SLOPES by slope, and they are indexed by density in another array L%DENS. ((L%SLOPES I) contains simply the value of (L%CHOP (TIMES 10. I)SW), for SW=0 or SW=1 depending on whether the first or second argument of L%LINES is being processed. (L%DENS I) contains a list of the slopes (divided by ten) of the L%CHOP-produced segments having density I (See 6.1 to recall "density".)

L%SA acts on the segments in the array L%SLOPES by associating sets of these elments together into groups having a large number of common points. The value of L%SA is a list of these groups; here a group is a list of associated segments with the one with highest density first. This grouping is necessitated by the existence of redundant segments among the outputs of L%CHOP for various arguments, for example if a set of points lies along a line with slope 250. It may give rise to segments in the outputs of (L%CHOP 250.), (L%CHOP 260.) and (L%CHOP 270.).

L%SA considers the various segments put in the array L%SLOPES by L%SFILL, in decreasing order of density. It does this by accessing through the array L%DENS in top-down order. When applied to a given segment, L%SB first removes that segment from the list containing it in the array L%SLOPES. Then the lists in L%SLOPES are searched for other segments which have more than a certain fraction of their members (the fraction being the value of L%SB2) in the original segment. This search is limited by some obvious heuristics, which are parametrized by L%SB1 and L%SB3 (L%SB1 gives a bound on the range of intercepts explored, and L%SB3 given a bound on the range of slopes.) These latter segments are grouped with the original and also removed from the array L%SLOPES. Since segments are considered in decreasing order of density, all of the latter segments have a smaller density than the former (the particular argument of L%SB) else they would have already been removed from the array L%SLOPES. The sort of "generalized subset" relation used for grouping was preferred to some simpler sort of transitive grouping relation; and accounts for the necessity of the L%DENS array for indexing in order to consider segments in the described order.

The function L%ENDS takes as argument a "group" as output by L%SA, and turns it into a line segment. This is done by taking the slope, intercept and limits of the most dense line in the group, and constructing a set of end points. The output format is:

((X1 Y1)(X2 Y2)).

VII. *USEFUL AUXILIARY FUNCTIONS

(L%GETFPSCAN X) takes as argument an element of the form:

(Yi Pi)

where pi is a full word block pointer, and returns an element of the form given by the menbers of (4.1).

(L%GETSLICE x) takes as argument a half-word block pointer, and returns a list of feature points like the Pij of (4.1), expressed in the "uncompact" form.