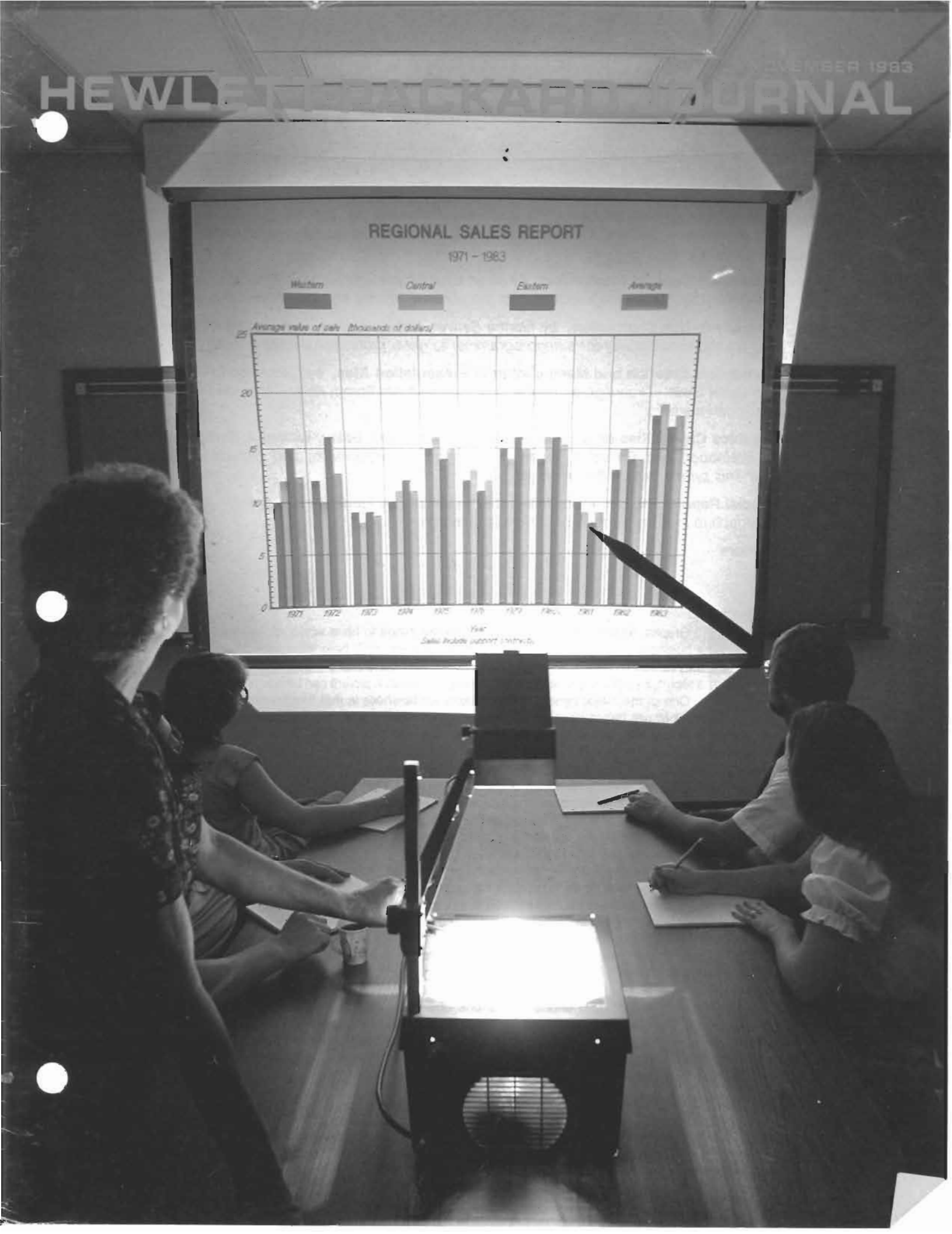
HEWLETT-PACKARD JOURNAL
DECEMBER 1983



ture of the problem was unrelated to program flow.

The HPDRAW design team needed a debugging tool with different capabilities. In particular, the engineers wanted to be able to examine an adjustable amount of information about program flow on a module-by-module basis. This was accomplished by creating a separate MPE** file to serve as a table for holding trace specifications.

Each module of HPDRAW contains checking code to determine if tracing is being requested. Information about routine names, parameters received, and values computed or accessed within the routine can be traced by changing the entries in the MPE file. Furthermore, tracing can be specified at the routine level as being on (trace me), off (don't trace me), directive (trace me and all routines called by me), or inquiry (trace me if my parent was traced.) By judiciously setting the options, it is possible to control the amount of tracing information that is displayed during execution. This saves time. It also allows engineers to focus on certain areas of the program flow and permits the tracing of specific variables. This debugging capability is now available to field service engineers for assistance in problem identification.

**MPE = Multiprogramming Executive, the HP 3000 Computer operating system

Acknowledgments

Phil Walden and Jim Long made HP 1000 software available to the design team. Their work was helpful in determining objectives during the development phase of HPDRAW.

Graphics Capabilities on a Laser Printer

by Tamara C. Baker, William J. Toms, James C. Bratnober, and Gerald T. Wade

TRADITIONALLY ALL TEXT DOCUMENTS have been prepared on printers and all graphics hard copy prepared on plotters. To understand why, consider that it would take 10 minutes for an HP 7221 Plotter running at full speed to plot the text of this paragraph. Plotters are designed to generate vectors (straight line segments). The HP 2680A Laser Printer, by contrast, is a raster image device similar in some ways to a black and white television CRT screen. A page of printer output is composed of small circular black dots analogous to the picture elements (pixels) on a CRT. There are 180 of these dots per inch in both the horizontal and vertical directions. Patterns of dots are grouped together into rectangular cells that form characters. A set of these characters is downloaded into the printer's memory and used to generate the text of the printed image on the page. To print a graphics figure the HP 2680A must be given a set of cells that collectively form the desired figure. This technique is fundamentally different from the vector generation approach to graphics, which more closely approximates the way an artist draws a figure. It is this difference between raster and vector generated output that has caused all documents to be printed and all graphics to be plotted until the recent release of the Interactive Formatting System/3000 (IFS/3000).

Formatting

IFS/3000 includes two major components. One is a set of intrinsics (routines) to allow programmatic control of document format. The other is IFS/2680, a program that allows interactive definition of the format of documents to be printed on the HP 2680A Laser Printer. The HP 2680A Graphics Package expands IFS/3000 by giving the user capabilities to print graphics output with programmatic

```
LPS Interpreter (A.00.00) HP36580(c) COPYRIGHT Hewlett-Packard Co. 1982

  IFS/3000 Intrinsics (A.01.00)

).help convertfigure

CONVERTFIGURE:
  Converts the specified figure into a raster image file.

  Syntax:
    .conv[ertfigure] ⟨figfilename⟩ ⟨figname⟩ ⟨rastfilename⟩ &
                     ⟨outputdev⟩ ⟨imageheight⟩ ⟨units⟩      &
                     ⟨imagerotation⟩

                     ⟨outputdev⟩      : 2680a
                     ⟨units⟩          : 0 = dots, 1 = inches, 2 = cm, 3 = mm
                     ⟨imagerotation⟩  : 0, 90, 180, 270

  Example:
  .convert figfile figure_name rastfile 2680a 3.25 1 90
```

Fig. 1. *The Help facility of the laser printing system interpreter for the HP 2680A Laser Printer provides the user with command descriptions and examples.*

intrinsics or through the laser printing system interpreter.

To print graphics on the HP 2680A Laser Printer, graphical information from a figure file must be converted to a raster image and stored in a raster image file. Then the user loads a copy of the raster image into the memory of the HP 2680A and issues a print command to place the graphics on the page. The seven new graphics intrinsics that add this capability are:

PCONVERTFIGURE converts a figure in a figure file to a raster image. Since it is much faster to print a raster image than a figure, PCONVERTFIGURE can be used to ready the graphics, possibly during off hours, for printing later.

PLOADRASTER loads an existing raster image into the HP 2680A memory. Once a raster image has been created with PCONVERTFIGURE, it must be loaded into the printer before it can be printed.

PPRINTRASTER prints a raster image that is loaded in the HP 2680A memory on the current page.

PDELETERASTER deletes a raster image from the HP 2680A memory. The HP 2680A can hold no more than 32 raster images at one time in its memory. PDELETERASTER is used to make room for more images.

PFLASHRASTER loads an existing raster image into the HP 2680A memory, prints it on the current page, and deletes the raster image from the HP 2680A memory. To make one print of a raster image, the user can combine the three steps PLOADRASTER, PPRINTRASTER, and PDELETERASTER in this intrinsic.

PPRINTFIGURE takes a figure in a figure file, converts it if necessary, then loads it and prints it. The conversion decision is based on whether there is an existing raster image of the right size and orientation, and whether the figure has been modified since the raster image's creation. All the previous intrinsics are low-level; they provide direct access to printer features. PPRINTFIGURE, on the other hand, combines low-level features with some printer memory management and conversion optimization. This intrinsic is used by TDP/3000 and HPWORD.

PFIGUREINFO returns the size, creation date, and other information about figures and raster images.

These intrinsics provide a flexible way of printing figures and the ability to store several raster images in printer memory. The HP 2680A can be configured with up to two megabytes of memory. This large amount of storage can be used to eliminate the repeated loading time of a raster image if it will be printed more than once in a job.

The laser printing system (LPS) interpreter provides a nonprogrammatic interface to most of the IFS/3000 intrinsics and gives the user access to the features of the HP 2680A through commands embedded in a text file or entered interactively on a line-by-line basis. If the interactive mode is used, the user receives an error message whenever a command is entered incorrectly. The user also has the benefit of an interactive Help facility (Fig. 1), which lists each command with an example. A description of the command and its parameters appears with the examples. These commands allow the user to control such things as the character font or form being printed, the creation and placement of raster images, or the location and orientation of printing. The user can print data in a form using named fields and can also merge text and graphics at print time. The LPS interpreter sends the text file to the HP 2680A and executes formatting commands by calling IFS/3000 intrinsics.

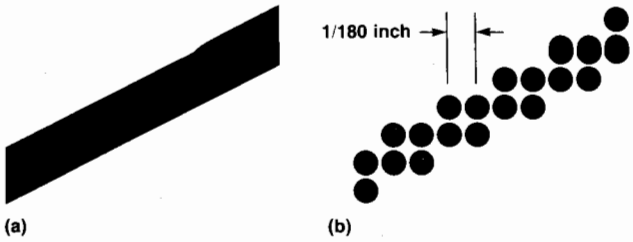


Fig. 2. *The HP 2680A Graphics Package uses a vector-to-raster conversion algorithm to simulate conventional graphic vectors (a) with printed raster images (b).*

Fig. 3. *This HPDRAW drawing contains 12,000 vectors, which would translate to 2.5 million dots if the raster image were printed on 8.5×11-inch paper.*

Plotting to the HP 2680A

Business graphics on the HP 3000 are created with DSG/3000, HPEASYCHART, and HPDRAW through a set of graphics utility routines. These routines are responsible for providing a device-independent interface to the applications programs and for driving the graphics devices. The internal representation of graphics figures consists of vector commands like MOVE, DRAW, and TEXT. To be able to plot to the HP 2680A Laser Printer all the graphics commands must be converted into a raster format.

Plotting to the laser printer from any of the graphics software products is a multiple-step procedure. First, all graphical commands are reduced to either MOVE or DRAW commands forming simple line segments. For instance, the three commands:

```
MOVE 0,0
DRAW 0,10
DRAW 10,10
```

generate two line segments, one from (0,0) to (0,10) and the other from (0,10) to (10,10). After all the commands are processed, the vectors are then translated into a raster image (Fig. 2) by a vector-to-raster conversion algorithm.

For each vector, the conversion algorithm must first determine which dots should be turned on (black) to represent the vector and then store the information in the bit map array that is the raster image. To give an idea of the size of the task, the HPDRAW drawing, Fig. 3, is made up of about 12,000 vectors. If it were printed on 8.5-by-11-inch paper, the raster image would have more than 2.5 million dots (total of all dots, black and white).

After the vector-to-raster conversion is done, the raster image must be partitioned into HP 2680A-specific linked character cells and saved in an MPE* disc file.

Finally, since the HP 2680A is a spooled device, the raster image must be communicated to the printer through the MPE spooler. The spool file is created and two commands are written in the file to print the image. For example, the first command might be summarized as: store this partitioned raster image in your memory as image #1. A copy of the image would follow the command. The second command would then be: print image #1 at position (x,y), a coordinate pair indicating a point on the current page.

Vector-to-Raster Conversions

Devices like the HP 2680A Laser Printer are capable of turning on any addressable point on a page, but they cannot draw lines directly like a plotter. Some means of decomposing lines into individual dots must be provided to create graphics on raster devices. Interpolation, the most obvious way to perform such decomposition, is unacceptable because it tends to leave gaps in the line and is computationally slow, since it requires floating-point calculations to determine the position of each dot.

The algorithm first used for the HP 2680A Graphics Package was originally developed by J. E. Bresenham.[1] It has the advantage of requiring only integer additions, subtractions, and arithmetic shifts (multiplications by 2). The algorithm incrementally determines the position of a point p_i in a vector, based on p_{i-1} and an error term. The octant of the vector is determined by its orientation. Octants are 45-degree areas taken from a standard Cartesian coordinate system, where octant 1 contains vectors at angles of 0 to 45 degrees, and so on, counter-clockwise around the axes (Fig. 4). A line in the first octant is plotted by stepping through raster positions from the endpoint with the smaller coordinates up to the opposite endpoint.

When stepping along a vector there are three allowable direction choices. Point p_i may be reached by a step in the X direction, the Y direction, or both directions from point p_{i-1}. But once a vector's octant location is determined, the choices are limited to two directions. A vector in the first octant may be stepped out in only increasing X or increasing X and Y directions. The two possible directions change with respect to the octant in which the vector lies. They are labeled M1 and M2; in the first octant M1 is increasing X and M2 is increasing X and Y.

During vector conversion an error term is calculated at each incremental step. The conversion algorithm attempts to minimize this term at each point on the vector by stepping in the proper direction and adjusting the term depending on the direction. For vectors in the first octant, the iterative relationship for error term E_i is as follows:

$$E_1 = 2*\Delta y - \Delta x$$

$$E_{i+1} = E_i + 2*\Delta y - 2*\Delta x \quad \text{if } E_i \geq 0 \text{ Step in M2}$$

$$= E_i + 2*\Delta y \quad \text{if } E_i < 0 \text{ Step in M1}$$

The values used in the above equations, as well as the directions M1 and M2, will change for different octant orientations.

VECTRAST, the first vector-to-raster conversion program used in the HP 2680A Graphics Package, required some special considerations when implemented on the HP 3000 Computer. The maximum-size output page initially anticipated was 11×17 inches with 180 dots per inch. This translates into roughly 6 million bits of data space to store the raster image, or about 378K words on the host computer. Since the HP 3000 Computer has a data space (stack) limit of 31,223 words, such an image could not be supported. To overcome this limitation, the strip or raster band method was used.

The strip method of processing required dividing the

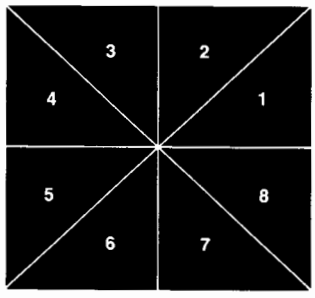


Fig. 4. *A vector can be categorized by octants. In each octant there are two allowable directions of movement between the endpoints.*

*MPE = Multiprogramming Executive, the HP 3000 Computer operating system.

raster image into strips of a size that would fit into available memory. Each strip was processed only once, then all vector portions falling within the strip were plotted. Since the input vectors were not sequenced in the same order as the strip processing, some type of processing hierarchy had to be imposed. For the first release, a scheme based on sorting the input was developed.

Vectors received from the input file were sorted according to the smallest Y component of their endpoints. These vectors were then input and plotted in the current raster band. If a vector continued out of the strip, it was placed in a list for processing in the next raster band along with the coordinates of the last point plotted in the previous band. This scheme had the advantage of being useful for any image size. The complexity of the drawing was governed by how large the active vector list grew. As the drawing of each vector on the list was completed, the space it occupied on the vector list was removed and returned to the free area.

Data space limitations caused the VECTRAST program to be run as a separate process. On the HP 3000, each process is given its own maximum 31,223-word limit of stack space. If the subsystem is not spawned separately, its data area is decreased by the amount used by its callers.

Conversion Refinements

The Bucket Brigade is the name of the conversion technique used to improve the VECTRAST program; it decreases the time spent sorting vectors and simplifies Bresenham's conversion algorithm without significant loss of accuracy in the placement of dots.

To produce an arbitrary-size raster image with limited data space, a conversion method must sort the input vectors, divide the raster image into bands, and finally convert each band, one at a time. The Bucket Brigade technique reduces the time spent in vector sorting without increasing the time spent in conversion. The method is based on the observation that input vectors need to be sorted into separate bands, but not sorted within each band. The Bucket Brigade eliminates the full sort of input vectors, replacing it with a specialized two-phase bucket sort.

In phase 1 (Fig. 5a) main memory is allocated (it will be used differently for the two phases of the process) and divided into n buckets, where n is the total number of raster bands in the final image. As each vector is read from the input file, it is placed into the appropriate bucket. Long vectors are broken into subvectors that lie entirely in one raster band and the subvectors are then added to the appropriate buckets. When a bucket is filled with vectors, it is written to a temporary scratch file and its memory space is emptied so the bucket can be refilled. After all the vectors have been read, any partially filled buckets are flushed to the disc file.

Phase 2 redefines the use of main memory (Fig. 5b). It allocates room for only one bucket of vectors plus one raster band of dots. The raster band is first cleared and the bucket file is read into the bucket area. The vectors in any bucket for this raster band are then converted and recorded in the raster image area. There is no carryover of long vectors since they have already been broken into subvectors that fit entirely into this band. When all the buckets for this raster band have been processed the complete band is written to the output file. The raster band and bucket are cleared and the bucket file is scanned for a bucket of vectors in the next raster band. This process is repeated until all the raster bands have been processed.

This specialized bucket sort uses less CPU time than a full sort since vectors in a bucket can be in any order. Furthermore, the number of disc accesses are reduced by transferring complete buckets of vectors rather than the individual vectors themselves. If a bucket can hold 400 vectors then the number of disc accesses is reduced by up

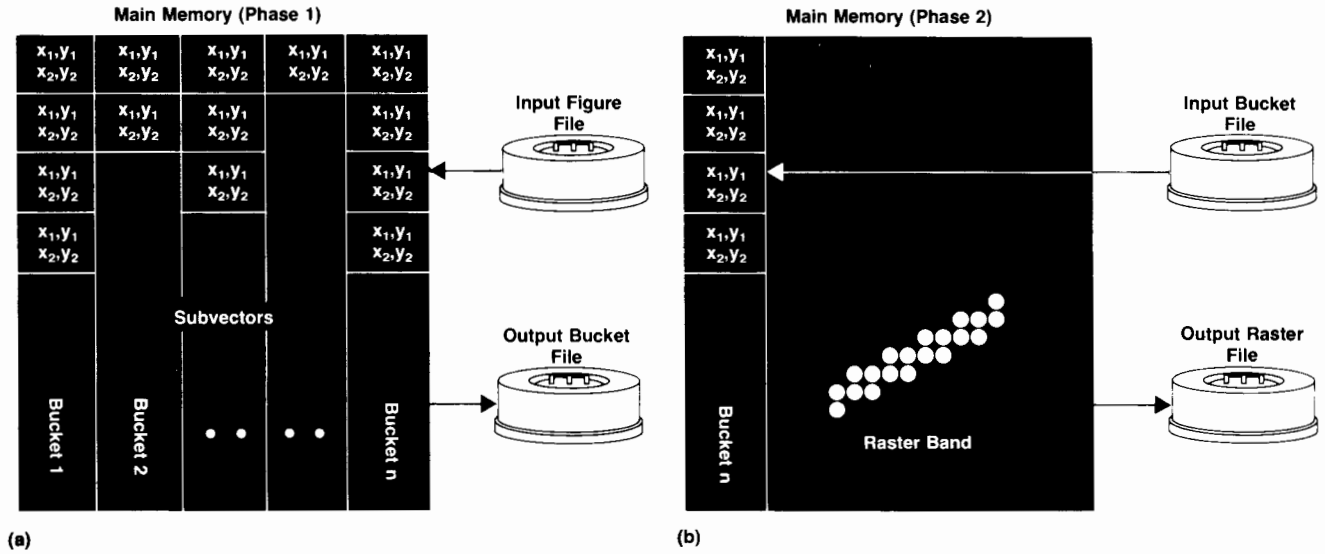


Fig. 5. *Memory use during Bucket Brigade vector-to-raster conversion. (a) Phase 1 fills memory buckets with subvectors that fall entirely with each raster band. (b) Phase 2 converts and records all the subvector buckets associated with a given raster band and then writes the completed band to an output file.*

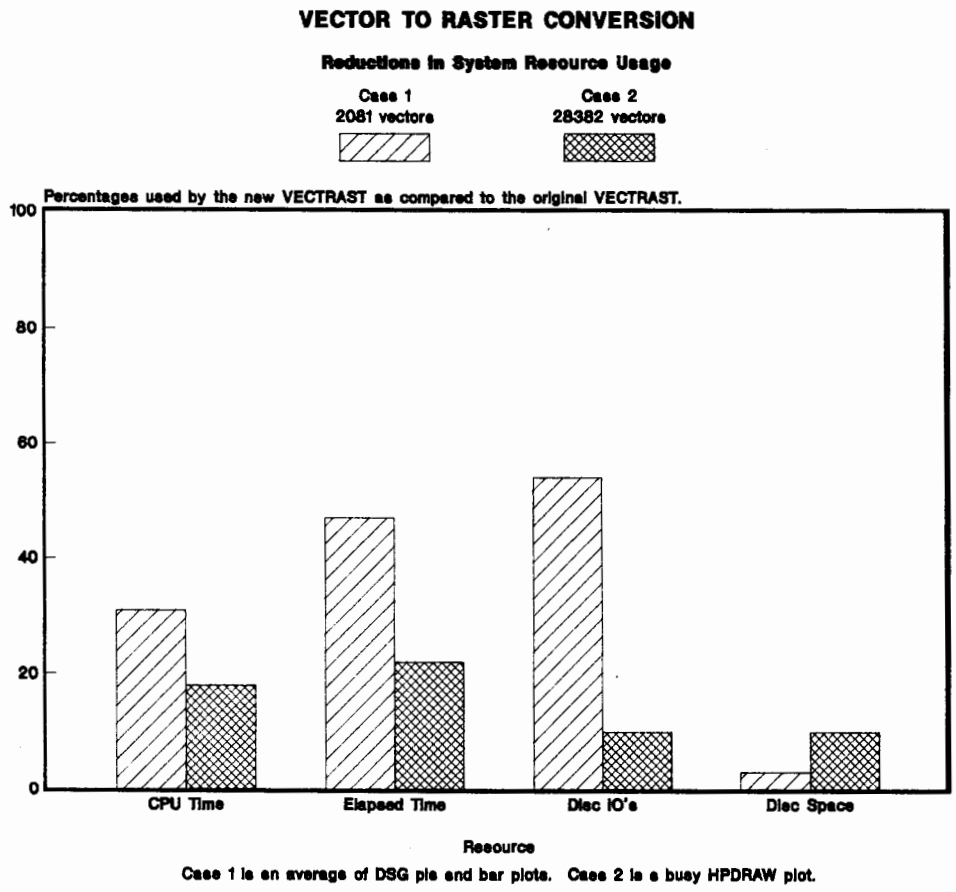


Fig. 6. *Benchmark comparisons illustrating the improved performance of the new VECTRAST using the Bucket Brigade technique over the early VECTRAST.*

to 400:1. Usually a full sort involves several stages so each vector is handled more than once. Consequently, the reduction can be even greater since buckets are written only once. A full sort requires large index files and several scratch files. Since the bucket file is the only necessary scratch file, disc space requirements are reduced. The process of subvectoring during the sort process frees memory space used for remembering long vectors that reside in more than one raster band. This main memory can then be used to make the buckets and raster bands larger. The overall result is a further reduction in disc I/O without increasing CPU time.

Two benchmark tests were run using the original VECTRAST program for vector-to-raster conversion as a standard. One was a simple bar chart which involved about 2,000 vectors. The second case was a large schematic which contained 28,000 vectors. The Bucket Brigade improvements in the use of CPU time, total elapsed time, disc I/Os and disc space are best summarized by Fig. 6. The amounts of each resource used by the original program are set to 100% for both tests and Bucket Brigade performance is expressed as a percentage of the original's. The original program uses the full presort of vectors technique. Note that as the number of vectors increases, the benefits of the Bucket Brigade technique become more pronounced.

The application program tool SAMPLER/3000 revealed that a cosine function was being called each time a diagonal vector's corresponding raster width was calculated. Development engineers realized that by substituting simple arithmetic statements for this trigonometric routine each vector could be processed faster.

The problem became how to determine which dots must be turned on to approximate a vector of an arbitrary width without using trigonometric functions. The solution involves a vector category scheme. A vector can be specified by its endpoints, (x_1,y_1) and (x_2,y_2), and its width. The width of the vector should be approximately half on each side of an imaginary line that connects the endpoints. Consider four cases:

1. The vector is horizontal ($y_1=y_2$) and can be plotted as a series of horizontal lines from $y=y_1-(\text{width}/2)$ to $y=y_1+(\text{width}/2)$. Width is expressed as a purely vertical dimension spanning x_1 to x_2 (Fig. 7a).
2. The vector is vertical ($x_1=x_2$) and can be plotted as a series of vertical lines from $x=x_1-(\text{width}/2)$ to $x=x_1+(\text{width}/2)$. Width is expressed as a purely horizontal dimension spanning y_1 to y_2 (Fig. 7b).
3. The vector is primarily horizontal ($|x_2-x_1|>|(y_2-y_1|$). This is similiar to the purely horizontal case where width should be plotted vertically. To do this on a diagonal line, x is stepped from x_1 to x_2, one dot at a time. At each value of x, y is calculated based on the vector endpoints. Then dots at width/2 are plotted above and below this value (Fig. 7c). The equations used are:

The slope of a line is $m=(y_2-y_1)/x_2-x_1)$
The Y intercept is $b=y_1-(m*x_1)$
For any x, $y=m*x+b$

4. The vector is primarily vertical ($|x_2-x_1|\leq|y_2-y_1|$). This is similiar to the purely vertical case where width should be plotted horizontally. Values of y are stepped from y_1 to y_2, one dot at a time. At each value of y, x is calculated, and the points at width/2 to the left and right of this value are plotted (Fig. 7d). The equations used are:

   The slope of a line is $m=(y_2-y_1)/(x_2-x_1)$
   The Y intercept is $b=y_1-(m*x_1)$
   For any y, $x=(y-b)/m$

This technique eliminates all calls to trigonometric functions and their associated overhead. Only simple arithmetic calculations are used. Bresenham's original algorithm achieves greater accuracy, limited by the dot or pixel resolution. This simpler algorithm produces comparable results but uses significantly less CPU time.

The benchmark results in Fig. 6 were produced using the arithmetically simplified replacement of the Bresenham algorithm. The new VECTRAST on the HP 3000 Computer can handle conversion of approximately two billion vectors into a raster image of over 95 million pixels. On the HP 2680A Laser Printer this translates into a plot 11 inches wide and over 22 feet long. It does this using only 40,000 bytes of main memory.

Algorithm refinements and the Bucket Brigade technique were incorporated into a new VECTRAST program that extends a system's range of image processing and makes vector-to-raster conversions with a significant reduction in all system resource use with no loss of functionality.

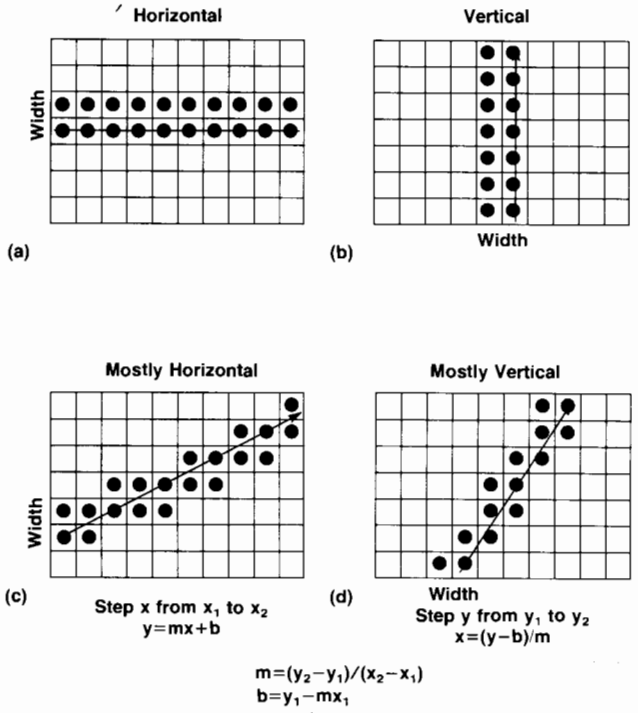


Fig. 7. *Vector width calculations are simplified by identifying directions: a) horizontal b) vertical c) primarily horizontal d) primarily vertical.*

Acknowledgments

Major contributions to the HP 2680A Graphics package were made by Jim Stratton. The project was a coordinated effort of several software project teams. We had almost daily contact with other groups from the Information Systems lab in Cupertino and the TDP/3000 project in Pinewood, England. By far the most outside communication occurred with our colleagues in the Boise Division where the HP 2680A Laser Printer is manufactured. New firmware was developed and tested in Boise to allow graphics to be printed on the laser printer. They also wrote the initial vector-to-raster conversion program based on Bresenham's algorithm.

Reference

1. J.E. Bresenham, "Algorithm for Computer Control of a Digital Plotter," IBM Systems Journal, Vol. 4, no. 1, 1965.