HIPS in action: Application of HIPS in research

THOMAS R. RIEDL

New York University, New York, New York

The Human Information Processing Laboratory has developed and made available a software system called HIPS (Human Information Processing Laboratory's Image Processing System) for doing research in experimental psychology and image processing. The HIPS system offers modular flexibility and simplicity of use. Examples of how HIPS was used in carrying out research in visual information processing are provided. A description of how HIPS can be used for documentation purposes is also given.

Research in psychology has come to rely to an ever greater extent on computer hardware and software. Diminishing costs have accompanied increasing power in state-of-the-art computer technology. It has become both feasible and cost effective to satisfy a variety of special-purpose needs with general-purpose hardware. To mediate special-purpose functionality in general-purpose systems, appropriate software is essential. The features made available on a system can dramatically affect both the quality and the direction of the work undertaken on that system.

The Human Information Processing Laboratory has developed and made available a software system called HIPS (Human Information Processing Laboratory's Information Processing System) for doing image processing (Landy, Cohen, & Sperling, 1984a, 1984b). The software system was developed under the UNIX (UNIX is a trademark of Bell Labs) operating system and closely adheres to its design philosophy (Bourne, 1978; Kernighan & Mashey, 1981). HIPS is modular by design and therefore provides flexibility and simplicity of use. The hardware used in the system (see Figure 1) consists, for the most part, of standard, off-the-shelf technology. HIPS was written in the C programming language (Kernighan & Ritchie, 1978) and can be readily transported to a variety of system hardware configurations.

Below, I outline how HIPS has been utilized in the UNIX environment in conducting research in vision information processing. I trace the early phases of experimental design through the setup and refinement phases of the experiment. I also discuss how HIPS can be used for documentation purposes.

APPLICATION TO THE STUDY OF VISUAL NOISE

HIPS was used in a study of the effects of noise on the perception of dynamic images (Pavel, Sperling,

The work on image processing of American Sign Language was supported by National Science Foundation, Science and Technology to Aid the Handicapped, Grant PFR-80171189. I wish to acknowledge the assistance of George Sperling, Michael Landy, and Yoav Cohen. My mailing address is: Human Information Processing Laboratory, Department of Psychology, New York University, 6 Washington Place, New York, NY 10003.

Figure 1. System block diagram. Generated using HIPS utilities *grpoly*, *gmag*, *gshift*, *grplot*, *grstring*, and *prthlf*.

Riedl, & Vanderbeek, 1983). That study addressed the question of whether signal-to-noise (s/n) invariance holds for the perception of American Sign Language (ASL). (For a general treatment of bandwidth requirements of ASL transmission, see Sperling, 1980, 1981.) From a practical point of view, the signal-to-noise invariance would enable us to generate predictions of intelligibility of ASL when degraded by additive noise. The relationship between ASL intelligibility and noise offers a useful way to describe the information in the images and to characterize the process of human perception of ASL.

One measure that can be used to characterize a signal is its variance (Pelli, 1981). A video signal is typically made up of a number of frames of a given size that are presented at some fixed rate (given as frames per second, or fps). It is possible to determine the average luminance of each frame of the sequence. The variance is a measure of how much the luminance of the signal within a frame varies around its mean: the higher the contrast, the higher the variance. In this study, the variance is taken as a measure of the signal.

In the same way that the signal is characterized by its variance, it was assumed that additive-intensity noise could also be described. In order to test the notion of signal-to-noise invariance as it applies to ASL signals,

s/n was systematically varied. Thus, Gaussian noise distributions with specified variances were added to ASL signals of differing and known variances. The intelligibility of such "noisy" ASL signals was hypothesized to be critically dependent upon the ratio of signal variance to noise variance rather than on the absolute variances themselves. Specifically, it was anticipated that reducing the variance (i.e., the contrast) of the ASL signal would not lead to diminished intelligibility so long as the variance of the noise was reduced proportionately.

Stimulus-Base Generation

The study required the creation of a stimulus base, of approximately 100 sequences, each 1 to 3 sec (30 to 90 frames) in duration. Film sequences of a person signing American Sign Language were transferred to the lab's VAX computer system by using programs in the HIPS library to upload, digitize, and appropriately manipulate images originally recorded on 16-mm film. Because the ASL sequences were to be viewed in a video system, the original 16-mm film record of the signs was recorded at 30 frames/sec. These 16-mm frames were viewed by a video camera, the output of which was digitized by the system's image processor, a Grinnell GMR 270. A 16-mm Lafayette motion analyzer was used in conjunction with a JVC S100U saticon target video camera for providing video image signals to the digitizer. A utility called *tv* causes the image processor to digitize, in real time, video signals originating from a video device such as a camera or video cassette recorder (VCR). Once the video signal has been digitized, another utility, *rframe*, is invoked to read the frame from the image processor's memory into an image file on the system's digital disk. *Rframe* allows one to specify how many rows and columns are to be read from the image processor's memory. A starting row and column location within a 512 x 512 picture-element (*pixel*) matrix can likewise be specified. As such, any subarea of the digitizer input field can be read into an image file. The command

```
rframe 100 150 200 300 > picture
```

causes the system to read a frame 100 pixels high x 150 pixels wide starting at a location 200 pixels from the bottom and 300 pixels from the left of the digitizer's (512 x 512) pixel input field. The resulting image file is called *picture*. (Following UNIX conventions, the symbols "<" and ">" designate the next file named as, respectively, the input to or output of the program.) Figure 2 contains a typical ASL frame digitized and read into the system using *tv* and *rframe*. The image depicted in Figure 2 was generated using a HIPS utility called *prthlf*, which allows gray-scale images to be printed on a dot-matrix line printer.

There is a corresponding output utility, *rframe*, which allows a single frame to be written to the image processor. The image processor incorporates a digital-

Figure 2. American Sign Language frame. Digitized and uploaded using HIPS utilities *tv* and *rframe*. Halftone image printed out on dot-matrix line printer using *prthlf*.

to-analog converter (DAC), which drives its video output. *Wframe* allows one to specify where in the 512 x 512 pixel output field the frame should appear. The command line

```
wframe 100 200 < picture
```

causes the image file *picture* to be written to the image processor such that its lower left-hand corner would appear 100 pixel positions up from the bottom and 200 pixel positions from the left margin of the output field.

Another module used in the process of creating the stimulus base is *reduce*, which shrinks images in the x and y dimensions by a specified factor. The command

```
reduce 4 < picture > smallpicture
```

would cause the image *picture* to be reduced in both the x and y dimensions by a factor of 4 and written to the output image file *smallpicture*. The resulting image contains 1/16th the number of pixels of the original image; since images occupy substantial space on the system (a single 512 x 512 pixel frame occupies a quarter of a million bytes of memory space), *reduce* is often used for space-saving purposes.

Image sequences are formed by using a utility called *catframes*, which concatenates specified frames into a single image sequence file. The command

```
catframes frame1 frame2 frame3 > sequence
```

generates an image sequence file containing *frame1*, *frame2*, and *frame3*. The output image file is *sequence* and has the same number of rows and columns as the input image frame.

In summary, to create the stimulus base, individual frames are digitized using *tv* and are read, one by one, into the system using *rframe*; these individual frames are reduced in size using *reduce*; the resulting frames

are then concatenated into a single image *sequence* file using *catframes*; this series of steps is repeated for each ASL sequence making up the stimulus base. It is important to note that each of these commands, when invoked, is automatically documented and stored in an associated sequence header. This header feature of the HIPS is discussed in more detail below.

Experiment Design

The design of the experiment involved repeated modifications of the image sequences. Initially, it was essential to determine ranges of noise and contrast appropriate for the study. Programs were used that generated and added Gaussian intensity noise to the images (*gnoise*), computed the mean and variance statistics of the images (*framevar*), and varied the contrast of the images (*contrast*). The *contrast* program is a user-created program derived from existing utilities in

HIPS. The aptly named program *movie* allows downloading of image sequences from the computer's memory to the image processor for display on a standard raster-scan TV monitor. *Movie* offers a variety of control capabilities affecting the output images. A detailed discussion of how these utilities were used in the experimental design follows.

Sample shell script. Once an initial selection of parameters has been made, the next step is to produce stimulus video tapes for use in the pilot studies. UNIX allows users to write command-level programs, called shell scripts, that enable a programmer to use system resources (hardware and software) as program elements (Bourne, 1978). Figure 3 shows a typical shell script used to generate stimulus tapes for the noise study. This shell script executes a broad variety of tasks, including (note—the uppercase characters below refer to line labels in the shell script of Figure 3): (A) genera-

```

set path=(/etc/usr/ucb/bin/usr/bin/usr/hipl/usr/usrs/trieidl/bin\
/usr/usrs/trieidl/shdir)
cd/image/trimage
grerase
A  genframe 128 96 64 > greyfrm
   set tmp=('date'); set date=('echo $tmp[2] $tmp[3] ')
   set disc=0
   set c=('cat contrasts'); set sn=('cat sn')
B  set param=('cat runinfile')
C  date > runlist$date
   echo contrasts: $c >> runlist$date
   echo signal to noise ratios:$sn >> runlist$date
   set group = 1
   while($#param>0)
   set i=0
   echo GROUP: $group >> runlist$date
   while ($i<8)
D  if (!( $\epsilon$  s$param[1])) then
   set disc = 1; cd/usr/usrs/trieidl/signdisc/er
   if ( $\epsilon$  s$param[1]) goto DO
   echo s$param[1] does not exist >> /image/trimage/runlist$date
   echo s$param[1] DOES NOT EXIST; goto loop
   endif
   DO:
   echo "s$param[1] c:$c[$param[2]] sn:$sn[$param[3]]" >>\
   /image/trimage/runlist$date
   grerase
E  grstring 10 192 -s "s$param[1] c:$c[$param[2]] sn:$sn[$param[3]]"
F  contrast $c[$param[2]] < s$param[1] | seqvar -p > /image/trimage/stmp
G  gnoiset $sn[$param[3]] < /image/trimage/stmp > /image/trimage/stmp1
   cntpix < /image/trimage/stmp1 >> /image/trimage/runlist$date
H  movie 100 192 -d -c -s -b -r < /image/trimage/stmp1
I  sleep 5; betapause
J  tar cf /dev/mt12 /image/trimage/stmp1; mt eof
   loop:
   cd /image/trimage; set disc=0
   repeat 3 shift param
   @ i = $i + 1
   end
   movie 100 192 -d -s -b -r -160 < greyfrm; sleep 5; betapause
   @ group = $group + 1
   end
   cp runlist$date /usr/usrs/av
   echo LOOK AT runlist$date for NOISE STUDY tape list > msg
K  mail av < msg; rm msg
   date >> runlist$date
L  lgpr runlist$date

```

Figure 3. Sample shell script. Incorporates various HIPS and UNIX utilities. Used in the noise study and acted as the basis of a number of other shell scripts used in the course of the study.

tion of a uniform video frame; (B) reading of parameters from an input control file; (C) creation of an execution accounting file; (D) testing for existence of the parameter file; (E) writing of characters to the image processor; (F) varying of the contrast of the image sequence; (G) addition of Gaussian noise to the image sequence; (H) outputting of images to image processor and control of a video recorder; (I) delaying and stopping of the video cassette recorder; (J) archiving of images onto digital tape; (K) sending of messages to the user; (L) printing of the accounting file. A detailed discussion of how these various tasks were implemented and executed using HIPS follows.

Uniform frame generation. The *genframe* utility allows frames of uniform pixel intensity to be generated. In the command line

A) `genframe 128 96 64 > greyfrm`

the first argument, "128," specifies the pixel intensity of the frame (in an intensity range of 0 to 255), and the second and third arguments indicate the number of rows and columns, respectively, in the frame (the maximum number of rows or columns is 512). *Genframe* is self-documenting in that it creates a header for the frame generated. A typical header, as printed by program *seeheader*, is shown in Figure 4.

The header *seeheader* contains descriptive information about the image file. For example, the number of rows and columns of the image, as well as a specification of the data format of the file, are included. The byte format is used for gray-scale images, whereas float (i.e., floating point) data format is typically used for files upon which greater precision computations are to be performed. An important part of the header is the sequence history, which details the succession of HIPS and UNIX utilities and commands that were executed to produce the current image file. It is possible for this sequence history to be extracted and executed again, in whole or in part.

```
Original name: triedl
Sequence name: greyfrm
Number of frames: 1
Original data:
Number of rows: 96
Number of columns: 64
Bits per pixel: 8
Bit packing: No
Pixel format: Bytes
```

Sequence history:

```
genframe 128 96 64 "-D Tue Dec 21 23:13:30 1982"
```

No sequence description

Figure 4. Image sequence header. Contains descriptive information about the image file. The "sequence history" traces the sequence of commands that were invoked to result in the image file; this command sequence can be extracted and reinvoked, in whole or in part.

Character-string generation.

```
E) gstring 10 192 -s $sparam[1] c:$c[$sparam[2]]
   sn:$sn[$sparam[3]]
```

Grstring allows character strings to be written to the image processor and output to a monitor or recording device. Above, the first two arguments specify the position where the character strings are to appear within the output frame. The "-s" switch indicates that the following argument defines the character string to be written. Note that standard shell-variable substitution conventions and syntax can be used, for example, above, `$sparam[1]` refers to a shell variable, "param[1]"; its numeric value, designated by preceding the name with a "\$," is substituted when this line is executed by the UNIX shell command interpreter (see Thompson, 1975). *Grstring* allows the user to specify vertical character-string output, normal or reverse video background, and double-size character output.

Contrast manipulation.

```
F) contrast $c[$sparam[2]] < $sparam[1] | seqvar
   -p > /image/trimage/stmp
```

As indicated previously, *contrast* is an example of a "customized" program derived from existing HIPS programs. Sections from the C program source code of the utilities *addseq* and *framevar* were extracted and combined with some new program code to create *contrast*. For the purposes of the noise study, it was important that contrast manipulations be performed by varying pixel intensities around the mean intensity of the image sequence. Accordingly, the *contrast* program first computes the global mean intensity of the image sequence (i.e., the mean of all pixels of all frames) and then scales intensities around this mean. Above, the argument passed to *contrast* specifies by what factor the image contrast is to be modified. The program allows contrasts to be either increased or decreased. Note once again that shell-variable substitutions can be made.

Addition of Gaussian noise.

```
G) gnoiset $sn[$sparam[3]] < /image/trimage/stmp
   > /image/trimage/stmp1
```

Gnoiset is another illustration of a customized utility. It is a modification of the HIPS utility *gnoise*. This program serves the purpose of adding specific amounts of Gaussian intensity noise to an image sequence. The program allows a signal-to-noise-ratio parameter to be specified as an input argument. The program first computes the intensity mean and variance of the input image sequence. It then generates noise frames in which intensity values are distributed in a Gaussian fashion (Gonzalez & Wintz, 1977; Rosenfeld & Kak, 1976). The Gaussian distributions (one for each input frame) are generated according to the specified signal-to-noise ratio and the computed variance of the input image

sequence. According to this method, a signal-to-noise ratio of 1 means that the intensity variance of the Gaussian "noise" distribution is equal to that of the ASL image "signal." The Gaussian noise frames, each having a mean of 0, are algebraically added, pixel by pixel, to the input image frames. The resulting output image sequence is one in which the Gaussian noise is dynamic. It was images of this type, with differing signal-to-noise ratios and contrasts, that were shown to subjects in the experimental runs.

Video Output. *Movie*, as indicated above, is perhaps the most often used HIPS utility. It allows image sequences to be downloaded from the system's memory to the image processor and output in standard video signal form (RS-170). As such, any piece of video equipment that can accept standard video signals can be driven by the system. Output devices typically used in the system are a Sony SLO-323 VCR and a Conrac video monitor. In the command line

```
H) movie 100 192 -d -c -s -b -r </image/trimage/stmpl
```

a number of option switches that produce various effects are specified: "d" causes a double-size image to be output; "c" generates and outputs a visual warning, "Cue," which precedes the ASL image sequence; "s" results in a single output of the image sequence (if not specified, the sequence would be output repeatedly in a looping fashion); "b" indicates that a blank frame should be output as the last frame of the sequence (this prevents the last frame of the ASL sequence from being left on the screen); "r" causes a "record" control pulse to be sent to the recording device just prior to the output of the sequence, causing the VCR to begin recording at the beginning of the image sequence.

Another noteworthy feature of *movie*, as implemented on the VAX system, is that video output can be synchronized to standard video sync pulses. The system is capable of generating real-time video image sequences that can be output to a standard monitor or VCR.

Equipment control.

```
I) sleep 5; betapause
```

Betapause is a hardware control utility. It "toggles" a control line, which in the configuration used causes a VCR to be stopped or started. As it appears in this shell script, its purpose is to stop the VCR previously started by the *movie* program ("r" switch). ("Sleep 5" is a UNIX command that causes the system to delay 5 sec before stopping the recorder.)

A shell script of the sort described is typically easy to debug and modify, if necessary. In fact, before the study was completed, no fewer than eight different versions of this script had been generated, each derived from previous versions, with appropriate modifications to suit the requirements of the study at each phase.

Documentation

Once the pilot studies had been run and the results collected, final adjustments were made to the parameters and programs for the generation of the stimulus video tapes to be used in the experiment. The necessary shell scripts were written (again, based on earlier scripts), and the final stimulus tapes were generated. The experiment was then run using a Sony SLO-323 video recorder and Koyo TMC-9M TV monitor (the monitor had been calibrated using HIPS). Score sheets were created on the system and used for data collection. Data were tallied and entered into the system for analysis.

Graphic and text documentation. Since the HIPS system was implemented under UNIX (Berkeley Beta-4.1), there are available a variety of software packages to perform other useful functions (Thompson, 1975). For example, a program called *v550* enables previewing of output on a graphics terminal before committing to hard copy. There is also a utility that allows contour plotting with hidden line capability. Text can be produced with UNIX word processing programs (*nroff* and *troff*).

HIPS-generated documentation. It is possible to use a variety of HIPS utilities to generate diagrams and illustrations. An example of a HIPS-generated block diagram is shown in Figure 1. This diagram was produced by using utilities (*grpoly*, *gmag*, and *gshift*) that generate and manipulate three-dimensional vector representations of polygons, which can be plotted and displayed (using *grplot*) on the image processor. The utility *grstring* allows alphanumeric character labels to be generated and integrated into previously *grplotted* frames. These utilities, along with *prthlf*, were used to produce the block diagram in Figure 1. This is an example of how the high degree of modularity of HIPS allows a broad variety of tasks to be performed with relative ease.

CONCLUSION

HIPS running under UNIX effectively performs the variety of tasks typically encountered in the course of doing psychological research in image processing. It is a system whose functional modularity allows novel applications to be programmed straightforwardly and with a minimum of additional programming effort. HIPS has been implemented on a hardware system that uses only standard, reasonably priced, off-the-shelf technology, underscoring its potential applicability to systems and uses.

REFERENCES

- BOURNE, R. S. (1978). An introduction to the Unix Shell. *Bell System Technical Journal*, 57, 2797-2822.
- GONZALEZ, R. C., & WINTZ, P. (1977). *Digital image processing*. Reading, MA: Addison-Wesley.

- KERNIGHAN, B. W., & MASHEY, J. R. (1981, April). The Unix programming environment. *Computer*, pp. 12-22.
- KERNIGHAN, B. W., & RITCHIE, D. M. (1978). *The C programming language*. Englewood Cliffs, NJ: Prentice-Hall.
- LANDY, M. S., COHEN, Y., & SPERLING, G. (1984a). A Unix-based image processing system. *Computer Vision, Graphics, and Image Processing*, **25**, 331-347.
- LANDY, M. S., COHEN, Y., & SPERLING, G. (1984b). HIPS: Image processing under UNIX. Software and applications. *Behavior Research Methods, Instruments, & Computers*, **16**, 199-216.
- PAVEL, M., SPERLING, G., RIEDL, T., & VANDERBEEK, A. (1983). *The effect of signal-to-noise ratio on the perception of American Sign Language*. Manuscript in preparation.
- PELLI, D. G. (1981). *Effects of visual noise* (pp. 159-166). Unpublished doctoral dissertation, Cambridge University, Cambridge, England.
- ROSENFELD, A., & KAK, A. C. (1976). *Digital picture processing* (pp. 162-168). New York: Academic Press.
- SPERLING, G. (1980). Bandwidth requirements for video transmission of American Sign Language and finger spelling. *Science*, **210**, 797-799.
- SPERLING, G. (1981). Video transmission of American Sign Language and finger spelling: Present and projected bandwidth requirements. *IEEE Transactions on Communications*, **COM-29**(12), 1993-2002.
- THOMPSON, K. (1975). *The Unix command language, structured programming—Infotech state of the art report* (pp. 375-384). Berkshire, England: Infotech International.

Appendix
A List of the Programs in HIPS Mentioned in This Article (For a
More Complete Listing, See Landy, Cohen, & Sperling, 1984b)

Program	Synopsis
<i>betapause</i>	Control (start/stop) VCR.
<i>catframes</i>	Concatenate separately stored frames into a sequence.
<i>framevar</i>	Compute a gray-level mean and variance.
<i>genframe</i>	Generate a homogeneous byte-formatted image.
<i>gmag</i>	Globally scale a vector plot.
<i>gnoise</i>	Gaussian noise generation.
<i>gpoly</i>	Generate a vector plot of a polygon.
<i>grplot</i>	Plot vector images to image processor.
<i>grstring</i>	Generate and display alphanumeric characters.
<i>movie</i>	Display an image sequence in real time.
<i>prthlf</i>	Print halftone images to line printer.
<i>reduce</i>	Reduce image size using pixel averaging.
<i>rframe</i>	Read a frame from the image processor.
<i>seeheader</i>	Print out image header information in a readable format.
<i>tvc</i>	Digitize video images.
<i>wframe</i>	Write a frame on the image processor.