VIPS — a digital image processing algorithm development environment

D G Bailey and R M Hodgson*

The major requirements of an image processing algorithm development system are presented. VIPS, a Vax-based image processing system developed at the University of Canterbury, New Zealand, is described and discussed in terms of algorithm development. Some of the applications of VIPS are listed.

Keywords: image processing, algorithm development

Digital image processing involves using a computer to apply a sequence of mathematical operations to a numerical representation of an object[1]. The desired result may be, for example, the measurement of the length of a feature, an enhanced image for display or even a decision on whether an object meets certain specifications. The application of digital image processing techniques to any particular problem may be split into two broad phases[2]. The first phase is to determine the image processing algorithm, i.e. the sequence of mathematical operations required to achieve the desired result; the second phase is to develop appropriate hardware to implement the solution. In some applications, the system that is used to develop the algorithm may be used in the final implementation.

Algorithm development is much like carpentry. All of the various image processing operations are the tools used by the algorithm development specialist to work on the input image, which represents the raw material. The main difficulty in algorithm development is that there is little or no theory that may be used to guide the selection of the best operation from several that may be suitable[3]. In practice, the operations are chosen heuristically from a large number of possible operations. This is done by trying one operation and, if it performs unsatisfactorily, trying another until the desired result is obtained. For this reason, an image processing system used for algorithm development must be highly interactive, and have available a wide range of operations. Another of the problems often encountered in developing algorithms relates to the 'trap of the two-legged existence theorem'[4]. This is the assumption that because humans can easily perform a particular image processing task, then so can a computer vision system. In fact the only truly general-purpose image processing 'machine' available is the human being. With the current state of the art, most image processing problems are only solvable if they are sufficiently specific and restricted.

Once an algorithm has been developed to solve an image processing problem, the next step is to determine the hardware required to implement the solution. Industrial situations place a severe time restriction on image processing problems, typically of the order of one second per measurement or classification cycle. The advent of VLSI and the use of parallel processing will widen the range of industrial tasks that may be tackled by speeding up the more time expensive operations. For this reason the resultant computer vision system is usually dedicated to the specific application for which it was developed.

VIPS, standing for Vax image processing system, has found considerable application in the algorithm development phase for a number of problems. In this paper, the VIPS hardware and software are described and compared with a range of other image processing systems. Some examples are given of the applications that have been investigated using VIPS.

HARDWARE ASPECTS

A block diagram of the VIPS hardware is shown in Figure 1. The host computer system is a Vax 11/750 central processing unit. This is used to execute the software implementing the various commands that are available. Important peripheral devices are as follows.

- A terminal is part of the interactive interface between the user and the system. All image processing commands are entered and all nonpictorial results

Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA
*Department of Production Technology, Massey University, Palmerston North, New Zealand

0262-8856/88/03176-09 $03.00

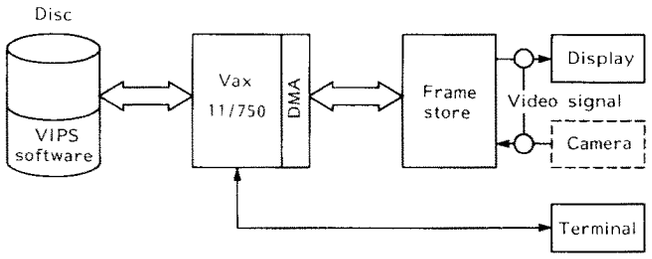

Figure 1. VIPS hardware schematic

of any operations performed are returned via the terminal.

- A DMA device (in this case a Digital Equipment DR11W) enables rapid transfer of image data between the host computer and the image capture and display subsystem.

The main component of the image capture and display subsystem is a Matrox MIP–512 image processing board. This controls the acquisition and display of images of the following resolutions: 128 × 128 pixels, 256 × 256 pixels and 512 × 512 pixels. Image input to the MIP–512 is software selectable from one of four cameras or other RS170 or RS330 standard video sources. During image capture, the input video signal is digitized to 8 bit. Image output is to a 14 inch (34.5 cm) high-resolution colour monitor, with the MIP–512 providing a false colour mapping from the image in its display buffer. The other component of the image capture and display subsystem is a slave 86/12 microprocessor which initializes the MIP–512 and controls the transfer of data between the Vax and the MIP–512.

SOFTWARE ASPECTS

VIPS was developed on the basis of experience gained in developing an earlier low-resolution system[5,6]. VIPS was developed initially to assist with research on the properties of Rank and Range filters[7,8], and was later extended to become a general facility for algorithm development.

VIPS is a command-based interactive image processing system. This means that the commands are run as procedures called from a parent program rather than directly as individual programs under the host computer operating system. Such command-based systems minimize the difficulty of maintaining the data structure (temporary images and other variables) from one command to the next, since this data structure may consist of a series of dynamic variables maintained by the parent program. A further advantage is a reduction in the time required to access each command. The disadvantages of such systems are that more program memory is required, leaving less memory for image and other data, and that the host system commands are difficult to access. The first disadvantage is only significant when high-resolution images (greater than 1024 x 1024) are being processed; this is because of the use of a virtual memory architecture on the host computer. In VIPS the second problem is overcome by providing a command that gives the user access to the host system commands.

VIPS provides a wide selection of image processing commands and operators (see Appendix 1 for a listing). This is essential in an algorithm development environment, as the best command for a particular task is more likely to be chosen. Information on all of these commands and their associated parameters is available online through the HELP command, or by using a special HELP key on the terminal. Any errors resulting from incorrect command use are reported using the Vax standard error handling format.

It often transpires that an image processing operator that is not supplied with a development system is required in a particular application. In this case, or when new techniques are being investigated, new operations must be developed. Any system used to develop algorithms to solve image processing problems must have the ability to add new commands quickly and easily[2]. VIPS enables commands to be developed in a high-level language such as PASCAL. To minimize problems in developing and accessing new commands, VIPS provides a three-layer command structure, as illustrated in Figure 2. At the innermost layer are the core commands; these are the commands that are provided initially with the system. At the next layer are the public user commands; these commands are user developed commands that have been installed into a particular system, and are therefore available to users at that location. At the outermost layer are the private user commands which are private to individual users; it is at this level that new commands are developed. This structure enables new commands to be developed easily and used as though they are part of VIPS, allowing the command to be tested in conjunction with other VIPS commands. Users can add their own private commands which remain separate from other users' private commands. The independence of private commands from the VIPS core and public user commands speeds command development, especially when several users are working on different applications. When commands have been debugged, and are required by other users, they may be installed as public user commands.

Commands are accessed by the system through a hash addressed[9] command table. Hash addressing speeds command access by making the command table contents

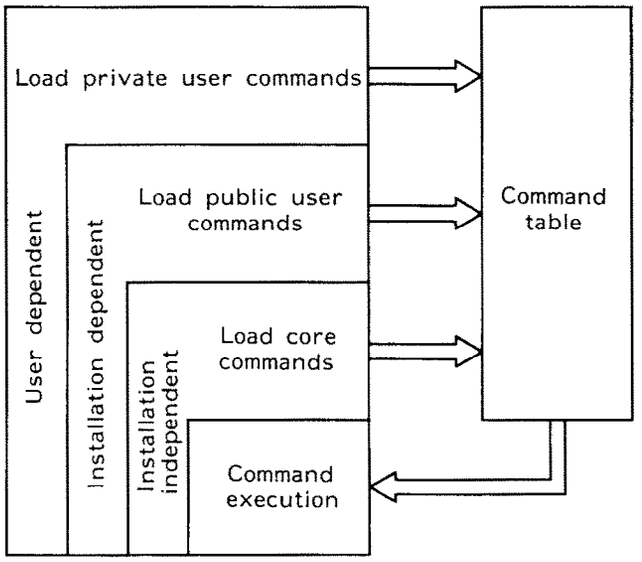

Figure 2. Three-layer command structure of VIPS

addressable. It also enables users to define command abbreviations easily by adding the definition into the hash table as a pseudocommand. This allows VIPS to be customized by providing abbreviations for commands or command lines that are frequently used. Each entry in the command table contains the address of the procedure implementing the command and information on all of the parameters used with the command. The command table is loaded at run time, when VIPS is first called. This enables VIPS to call private and public user commands without having them specifically linked to the command parsing section of the parent program. Thus when user commands are present, the user program loads the addresses and parameter information into the command table and then calls the VIPS command parsing loop. When the command is parsed, the command address saved in the command table enables the correct command to be called directly. The parameter information in the command table enables the parent program to check parameter types, and to detect whether any essential parameters have been left out. Optional values may be provided for some of the parameters if no value is specified in the command line.

A variety of variable types are provided, from general-purpose variables such as integers and real numbers to structures used more specifically for image processing such as vectors and histograms. This large variety expands the capability of VIPS command sequences by enabling interaction between commands of more than just images. Images may be of any size to allow the resolution, and hence the execution speed, to be tailored to suit the problem. The system also allows rectangular subimages of arbitrary size to be selected and operated on. One of the features of VIPS is that all user variables are accessed symbolically, rather than by location, allowing the user to concentrate on the image processing problem. In practice this means that images and other variables may be given meaningful names, making algorithms easier to develop and simplifying algorithm modification at a later date.

VIPS is interactive, allowing the results of one operation to be examined before the next operator is chosen. A useful guide for interactive systems is that each operation should take less than 15 seconds[6]. If the operations regularly take longer than this, the time delay becomes uncomfortable for the user, and it becomes difficult for the user to maintain concentration. For an image resolution of 128 x 128, simple operations such as generating test images, adding images and measuring areas take less than two seconds. More computationally intensive operations such as filtering and Fourier transformation take 5–10 seconds. The execution time of most VIPS commands is proportional to the image size. This means that the processing time required for a 256 × 256 image will be about four times that for a 128 × 128 image. Simple operations remain within the 15 second guideline, but other operations such as filtering become slow. For this reason, all processing is performed using the minimum resolution and image size that is practical for the application. The time taken by each command also increases as the host computer becomes more heavily used, but remains within the 15 second guideline for most commands operating on images within a resolution of 128 × 128 or less.

Since the host computer is a time shared, multiuser system, VIPS provides a command timing facility. This returns the actual computer time used for each command in an algorithm. This information may then be used to locate processing bottlenecks and, in the case of time critical applications, to identify the commands in the algorithm that may require special-purpose hardware in the final implementation.

The VIPS software may be accessed at three different levels, as illustrated in Figure 3. At the lowest level, it consists of a library of image processing procedures and support utilities. This enables the procedures implementing existing commands to be called from user commands, or even incorporated in other programs. At the next level up, these procedures are available as commands which may be invoked interactively from VIPS by the user. At the highest level, the commands form the basis of an image processing language. The commands are combined together as programs which implement all or parts of image processing algorithms. Features such as looping and branching are provided to allow flexibility in the command sequence which previously required user evaluation. Loop structures also provide the means for repetitive testing of new commands and techniques. All of the looping and branching constructs necessary[10] for a language are available for use from within VIPS programs.

VIPS is written in Vax PASCAL, which contains a number of extensions to standard PASCAL. This paragraph describes the features of VIPS that make use of these nonstandard extensions of PASCAL. Vax VMS system services are used extensively for online documentation (HELP) and error handling, and to reduce the programming effort required to give information in a form similar to that of other packages that may be encountered on a Vax system. System services are also used to access device drivers for the terminal and the image capture and display subsystem. These features may be rewritten in standard PASCAL provided that appropriate drivers are available. All images used in VIPS are stored in arrays; however, standard PASCAL does not allow dynamic allocation of arrays of arbitrary size (i.e. the size is not known until run time). To over-

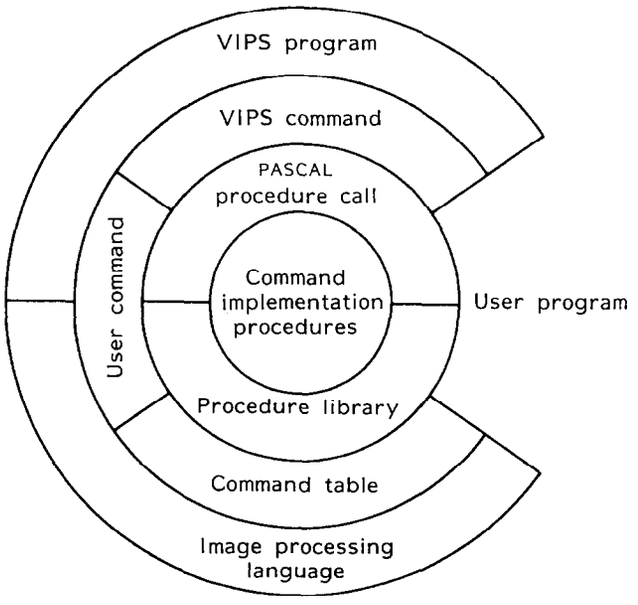

Figure 3. Different levels of access to VIPS

come this limitation, image memory is allocated in blocks of powers of two, and a descriptor of the array is maintained which describes the size and indexing of the array. Vax PASCAL allows this descriptor to be passed to an implementation procedure, and the array (arbitrarily sized) is accessed correctly from within the procedure. Because of the method used to pass images to commands, VIPS is not readily transportable to other than Vax systems. (The authors have operated VIPS on Vax 11/750 and MicroVax II systems.)

COMPARISON WITH OTHER ENVIRONMENTS

A large number of systems have been developed and are described in the literature. These range from dedicated hardware (e.g. the Clip series of processors and others[11]) to general image processing languages (such as L[12] or PIXAL[13]) or subroutine libraries (e.g. Spider[14]). These systems were all designed for different purposes, and have advantages over the other systems in accomplishing those purposes. For this reason, the comparison made here is not absolute, but rather in terms of the ability of each system to provide an environment for developing algorithms for specific image processing applications. VIPS is not compared exhaustively with each system, but systems are selected from the many available that are representative of a class in order to make a comparison of a particular feature common to a range of systems. The comparison is arranged in terms of features or concepts that are particularly important for an algorithm development environment.

The nature of algorithm development virtually requires a command-based system such as VIPS. Although subroutine libraries such as Spider[14] provide a wide range of operations, there is no convenient method of examining the effects of different operations in an interactive manner. This is also true for image processing languages, unless they also provide an interactive interface. At the other extreme are the menu-driven image processing systems such as those using personal computers (PCs) as the host (e.g. ImageLab and ImageTool[15]). These provide a very interactive means of selecting operations, but the specification of parameters is a problem for complicated operations.

Most command-based systems, including menu-driven systems, allow users to define a macro consisting of a sequence of commands, but often few, if any, branching constructs are available. This is especially true of early systems such as Susie[16] or Ucips[5,6]. At the other extreme are languages where loops and branches are provided. In this sense, VIPS could be likened to an interpreted language similar to BASIC. Systems based around FORTH, LISP or PROLOG have similar properties (e.g. Provision[17]). Commands may be entered interactively while an algorithm is being developed, and commands may be combined into an image processing program where the branching constructs provided by VIPS allow considerable flexibility. The effectiveness of this type of system in making it easy for nonspecialists to develop algorithms is demonstrated each year by final-year Bachelor of Engineering project students working on image processing projects at the University of Canterbury, New Zealand.

Almost all image processing systems that have been developed recently have available a wide range of operations, from low-level image-to-image operations to those using higher-level data structures. The most notable exceptions to this are the PC-based menu-driven systems. Such systems usually provide only image-to-image operations.

When no suitable operations are available for a particular task, and a new operation must be developed, a single-level software architecture is most desirable[2]. VIPS, like most other command-based systems, has two levels. While ideas for new operations may be tested at the command level, the overhead in accessing individual pixels makes this too inefficient to be practical. However, the structure provided by VIPS allows users to develop programs in a high-level language and add them into the system with reasonable ease. Other operations may be accessed directly within a new operation by procedure calls. The use of PASCAL as the implementation language is an advantage since most new users have already had some exposure to the language.

There are three ways commonly used to access images: file-oriented, frame-buffer-oriented and memory-oriented systems. File-oriented image access is generally employed in applications where very large images are commonly used, such as remote sensing (e.g. Vicar[1]). Such systems usually operate in batch mode because of the large image size. Systems which store images in frame buffers (such as Ucips[5]) either have a small range of fixed image sizes or use windowing schemes to operate on a subimage. There is only a small number of images, and these are accessed by number rather than by name. Memory-oriented systems such as VIPS are much more flexible in the size, number and naming of images and other variables.

VIPS also has available a wide range of variable types. This allows a flexibility normally only obtainable in image processing when using either an image processing language or a conventional high-level language with a subroutine library. This feature is shared by few other command-based systems.

To make the system easier to use, full online documentation is essential. This is particularly important in an algorithm development environment, where there may be a large number of commands available. VIPS provides complete online documentation for all of the available commands and variable types. The information is cross referenced so that, if one operation is not successful for a particular task, related operations may be found quickly. Few systems give more than a brief description of commands. A series of online tutorials and example algorithms is also provided for users who are new to VIPS or image processing. VIPS also provides a context sensitive HELP key which may be used to get information on command parameters. This feature serves a similar purpose to prompting for missing parameters, a method often used by other command-based systems.

Most other systems (e.g. Susie[16] or Provision[17]) use abbreviated command names. Users have to learn many of these abbreviations to make proper use of the system. VIPS uses full meaningful names, which are easier to learn. Users are able to define their own abbreviations (or even redefine the command names) to suit their own tastes if they wish.

VIPS, like many other command-based systems, uses a general-purpose time shared computer system as its host rather than dedicated high-speed hardware (see Duff[11] for examples). Speed is not necessary for developing algorithms provided that the system is interactive, whereas flexibility is essential. VIPS is unsuitable for the implementation of the final algorithms in all but special cases where speed is not important, such as in small batch runs typically used for research.

APPLICATIONS

There are currently three VIPS systems installed in New Zealand. These are at the University of Canterbury, where the system was initially developed, at the Wool Research Organisation of New Zealand (WRONZ) and at the Forest Research Institute (FRI).

At the University of Canterbury, one of the major applications of VIPS is in teaching image processing techniques, where it is used as part of a Master of Engineering course in image processing[18]. In conjunction with a series of lectures, the students are given a typical industrial image processing problem for which they must develop a suitable algorithm.

An important use to which VIPS is ideally suited is the development of new commands and techniques required when investigating new areas of image processing. VIPS has been used in a number of such areas, including an investigation into the properties of Rank and Range filters[7,8], and the development of algorithms for growth ring tracking and defect detection in optical and X-ray images of timber[20].

The most important area of application of VIPS is as an algorithm development tool. Because of the interactive nature of algorithm development, VIPS provides a near ideal environment. Some of the applications that VIPS has been used to investigate over the past three years include

- measurement of the area within growth rings of trees[3]
- quality grading of kiwifruit to be exported[3]
- classification of cell types in a stained wool fibre[21]
- statistical texture measures for carpet wear assessment[19]
- determination of preservative penetration into timber
- detection of shives in paper handsheets
- measurement of parameters of wood pulp fibre cross-sections[22,23]
- defect detection in sawn timber[19,20].

Figure 4 lists the VIPS program for the detection of blemishes on kiwifruit, and shows typical images at various stages throughout the processing.

Full details of the VIPS hardware and software are available to educational institutions at a modest cost. For further information, contact the second author.

CONCLUSIONS

VIPS provides an excellent environment for both command and algorithm development, the key features being

- the general-purpose interactive nature of the system
- the large selection of commands available
- the ease with which new commands may be added.

Other features which are important are: the availability of looping and branching constructs for use within command sequences; the wide range of variable types; and complete crossreferenced online documentation.

Once the algorithm has been developed, VIPS may be used to implement it for small-scale laboratory runs where speed is not critical. Where time is important, the information obtained from VIPS is valuable for developing a dedicated system.

ACKNOWLEDGEMENTS

During the initial development of VIPS, D G Bailey held a University Grants Committee postgraduate scholarship. The assistance of A Earl and R Cox in constructing the version 1 hardware is also acknowledged. Version 2 of VIPS was developed at the Wool Research Organisation of New Zealand (WRONZ), and version 3 at the Pulp and Paper Research Organisation of New Zealand (PAPRO). The New Zealand Kiwifruit Authority provided grants for the kiwifruit grading project.

REFERENCES

1 **Castleman, K R** *Digital image processing* Prentice-Hall, Englewood Cliffs, NJ, USA (1979)
2 **Brumfitt, P J** 'Environments for image processing algorithm development' *Image Vision Comput.* Vol 2 No 4 (November 1984) pp 198–203
3 **Bailey, D G** 'Hardware and software developments for applied digital image processing' *PhD thesis* University of Canterbury, Christchurch, New Zealand (1985)
4 **Hunt, B R** 'Digital image processing' *Adv. Electron. Electron Phys.* Vol 60 (1983) pp 161–221
5 **Cady, F M and Hodgson, R M** 'Microprocessor based interactive image processing system' *IEE Proc. E* Vol 127 (1980) pp 197–202
6 **Cady, F M, Hodgson, R M, Pairman, D, Rodgers, M A and Atkinson, G J** 'Interactive image processing software for a microprocessor' *IEE Proc. E* Vol 128 (1981) pp 165–171
7 **Hodgson, R M, Bailey, D G, Naylor, M J, Ng, A L M and McNeill, S J** 'Properties, implementations and applications of rank filters' *Image Vision Comput.* Vol 3 No 1 (February 1985) pp 3–14
8 **Bailey, D G and Hodgson, R M** 'Range filters: local intensity subrange filters and their properties' *Image Vision Comput.* Vol 3 No 3 (August 1985) pp 99–110
9 **Knuth, D E** *The art of computer programming, Vol 3* Addison-Wesley, Reading, MA, USA (1973) p 506
10 **Wirth, N** 'On the composition of well structured programs' *ACM Comput. Surv.* Vol 6 (1974) pp 247–259
11 **Duff, M J B** 'Special hardware for pattern processing' *Proc. 6th Int. Conf. Pattern Recognition* (1982) p 368

a

```
PROGRAM
  SUBC kiwifruit 20                          ! Removal of background
  EXPAND kiwifruit                           ! Intensity range normalization
  FILTER RANK kiwifruit kiwi2 5              ! Median prefilter to remove hairs
  LET kiwi3 = kiwi2
  HULL kiwi3                                 ! Convex hull across rows
  TRANSPOSE kiwi3
  HULL kiwi3                                 ! Convex hull of columns to give model
  SUB kiwi3 kiwi2                            ! Compare fruit with dynamic model
  FILTER RANK kiwi3 kiwi2 5
  EXTREME kiwi2,,maximum                     ! Locate maximum intensity
  IF maximum > 50
    OUT "Kiwifruit has a point defect"/LINE
  ELSE
    THRESHOLD kiwi2 24                       ! Locate area defects
    AREA kiwi2,,defect_area
    IF defect_area > sq_cm                   ! Compare with 1 cm square limit
      OUT "Kiwifruit has an area defect"/LINE
      ELSE
      OUT "Kiwifruit is acceptable"/LINE
    END
  END
END
```

b

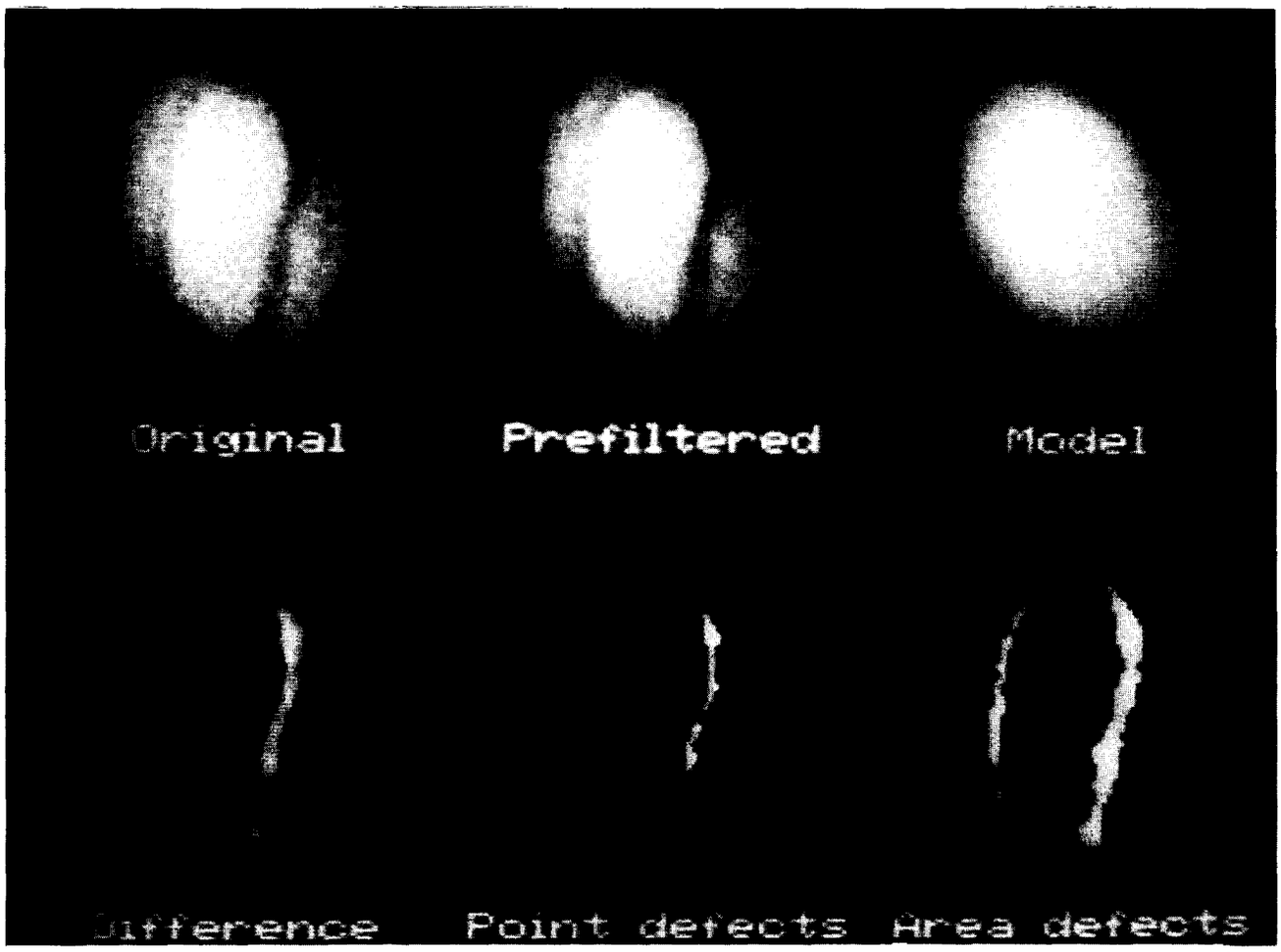

Figure 4. Use of VIPS to detect defects in kiwifruit: **a**, *the VIPS program used;* **b**, *typical images obtained during processing of a kiwifruit with a water stain defect (top row shows original image, image after filtering to remove hairs and the model generated; bottom row shows defect regions, point defects and area defects)*

12 **Radhakrishnan, T, Barrera, R, Guzman, A and Jinich, A** 'Design of a high level language (L) for image processing' in **Duff, M J B and Levialdi, S (eds)** *Languages and architectures for image processing* Academic Press, London, UK (1981) pp 25–40

13 **Levialdi, S, Maggiolo-Schettini, A, Napoli, M, Tortora, G and Uccella, G** 'On the design and implementation of PIXAL, a language for image processing' in **Duff, M J B and Levialdi, S (eds)** *Languages and architectures for image processing* Academic Press, London, UK (1981) pp 89–98

14 **Tamura, H, Sakane, S, Tomita, F, Yokoya, N, Kaneko, M and Sakaua, K** 'Design and implementation of SPIDER — a transportable image processing software package' *Comput. Vision, Graphics, Image Process.* Vol 23 (1983) pp 273–294
15 *Imagelab and Imagetool model 100 user manuals* Werner Frei Associates, Santa Monica, CA, USA (1986)
16 **Batchelor, B G, Brumfitt, P J and Smith, B V D** 'Command language for interactive image analysis' *IEE Proc. E* Vol 127 (1980) pp 203–218
17 **Batchelor, B G** 'Merging the AUTOVIEW image processing language with PROLOG' *Image Vision Comput.* Vol 4 No 4 (November 1986) pp 189–196
18 **Hodgson, R M** 'First course in digital image processing' *Int. J. Elec. Eng. Educ.* (to be submitted)
19 **Lee Hok Siew, Hodgson, R M and Wood, E J** 'Texture measures for carpet wear assessment' *IEEE Trans. Pattern Anal. Mach. Intell.* Vol 10 No 1 (1988) pp 92–105
20 **Ling, Y P** 'Defect assessment in sawn timber' *MEng report* University of Canterbury, Christchurch, New Zealand (1986)
21 **Orwin, D F G and Bailey, D G** 'The measurement of wool cortical cell proportions' in *Proc. Workshop Applications of Mathematics and Physics in the Wool Industry* Lincoln College, Canterbury, New Zealand (1988) pp 330–337
22 **Kibblewhite, R P and Bailey, D G** 'Measurement of fibre cross section dimensions using image processing' *Appita* (to be published)
23 **Bailey, D G and Kibblewhite, R P** 'Automated measurement of pulp fibre cross sections' (in preparation)

APPENDIX 1: VIPS ROUTINES

All routines listed below in upper case are VIPS commands, while those in lower case are callable procedures. Almost all of the VIPS commands and functionals may also be called as procedures.

Functionals

%COLUMN	Returns the column part of a vector
%DISTANCE	Returns the length of a vector
%HISTOGRAM	Returns the value of a selected point in a histogram
%INDEX	Returns a value from the specified position in a list
%INTEGER	Converts a real number into an integer
%LENGTH	Returns the length of a list or string
%REAL	Converts an integer into a real number
%ROW	Returns the row part of a vector
%SIZE	Returns the size of an image
%SQRT	Returns the square root of a number
%STRING	Converts an entity into a string
%TRANSPOSE	Returns the transpose of a vector
%TYPE	Returns the type of a variable as a string
%VALUE	Returns the value of a selected point in an image

Chain code manipulation

CHAIN AREA	Returns the area surrounded by a chain
CHAIN BRANCHES	Returns the number of separate branches of a chain
CHAIN CODE	Extracts all boundary chain codes from a binary image
CHAIN CHULL	Returns the convex hull of a loop
CHAIN DRAW	Redraws the chain in an image
CHAIN EXTRACT	Extracts a single chain fom a set of chains
CHAIN LENGTH	Returns the number of elements in a chain
CHAIN LOOP	Determines whether the chain is a loop or a line
CHAIN MOMENT	Calculates moments of a loop
CHAIN PERIMETER	Calculates the corrected length of a chain
CHAIN RECTANGLE	Determines the minimum area enclosing a rectangle
CHAIN SIZE	Returns the extent of a chain

Data conversion

CONVERT	Converts from one image type to another
swap_fortran	Swaps row column information in a FORTRAN descriptor
upcase	Converts a VARYING OF CHAR into upper case
varying_of_char	Converts a FORTRAN string to VARYING OF CHAR

Data extraction

AREA	Calculates the area of an object in pixels
BLOB	Counts the number of independent blobs in an image
EXTREME	Finds the minimum and maximum pixel values
FLASH	Displays an image with a flashing cursor
HIST GET	Obtains the histogram of an image
HIST DISPLAY	Displays a histogram on the screen
PROFILE	Obtains an intensity line profile of an image
SLICE	Slices through the intensities of an image
STATISTICS	Obtains the mean and SD of an intensity range

Data input and output

FILE	Opens a file for data output
INQUIRE	Initializes variables interactively
LOAD	Loads one or more variables from a file

OUT	Outputs data to the terminal
SAVE	Saves one or more variables in a file
WRITE	Outputs data to the data file

Display routines

CAPTURE	Captures an image onto the display
CLEAR	Clears all or part of the display
DISPLAY	Displays an image
DOWNLOAD	Downloads a user display routine
FLASH	Displays an image with a flashing cursor
GET	Gets an image from the display
HIST DISPLAY	Displays a histogram
ROAM	Modifies the hardware display characteristics
SET DISPLAY	Initializes and sets the display to be used
SET LUT	Selects and initializes a hardware lookup table
SLICE	Slices through the intensities of an image
assign__channel	Assigns a channel to the display
clear__error	Clears the error LED on the display
deassign__channel	Deassigns a channel to the display
display__abort	Exit handler for user display commands
end__point__display	Terminates the display of a series of points
init__point__display	Initializes the display of a series of points
send__control__block	Sends a control block to the display
transfer__block	Sends or receives a block of data from the display

Filters

FILTER ENHANCE	A rank-based edge enhancement filter
FILTER LINEAR	Linear 3 × 3 convolution filter
FILTER MOMENT	Moment-based filter using a 3 × 3 square window
FILTER RANGE	A rank-based edge detection filter
FILTER RANK	Rank filters an image
FILTER SOBEL	A nonlinear local edge detection filter
FILTER TRIM	A trimmed linear filter using a 3 × 3 square window
SET MASK	Initializes a convolution mask
SHRINK	Shrinks (or expands) a binary image

Fourier transformation

FFT	Fast Fourier transforms an image

Image generation and retrieval

LOAD	Loads one or more variables from a file
NOISE	Generates a noise image with specified statistics
SAVE	Saves one or more variables in a file
TEST	Generates a test image

Image manipulation

CHULL	Convex hull of blobs in a binary image
COPY	Copies all or part of one image into another
DISTANCE	Applies a distance transform to a binary image
EXTEND	Extends the edges of an image
FILL	Fills a region to a uniform intensity
HULL	Convex hull of intensities across the rows of an image
ROTATE	Rotates an image by 90°
TEXT	Displays text in an image or on the display
THIN	Thins an image to its skeletal form
TRANSPOSE	Transposes an image (swaps the rows and columns)
ZOOM	Magnifies or reduces an image

Line drawing routines

ARCANGLE	Plots a circular arc subtending a given angle
ARCPOINT	Plots a circular arc between two points
CIRCLE	Plots a circle of a given radius
DRAW	Draws a series of line segments
LINE	Plots a line between two points
PLOT	Plots a point in an image
RAY	Plots a line from a start point in a given direction

Point operators on a single image

ADDC	Adds a constant to an image
CLIP	Clips the intensities of an image at specified limits
DIVC	Divides an image by a constant
EXPAND	Linearly expands the intensity range of an image
HIST EQUAL	Performs histogram equalization on an image
HIST SHAPE	Performs arbitrary histogram shaping on an image
INVERT	Inverts the intensities of an image
LOOKUP	Translates intensities in an image via a lookup table
MULTC	Multiplies an image by a constant
SUBC	Subtracts a constant from an image
THRESHOLD	Thresholds an image at the specified intensities

Point operators on two images

ADD	Adds two images with wrap-around or saturation

AND	Logical AND of two images
DIV	Calculates the ratio of two images
OR	Logical OR of two images
MULT	Multiplies two images
SUB	Subtracts two images with wraparound or saturation
XOR	Logical exclusive OR of two images

Program commands

EDIT	Creates or modifies a VIPS program
ELSE	Optional part of an IF command
END	Marks the end of a PROGRAM, IF, FOR or WHILE
EXIT	Exits from the current program, or all program levels
FOR	A loop command using a loop variable
IF	A branch command selects one of two command sequences
ON	Specifies the action to take on errors or ^Cs
PROGRAM	Indicates the start of a VIPS program
REPEAT	A conditional loop command with the test at the end
RUN	Runs a VIPS program
UNTIL	The test command at the end of a REPEAT loop
WITH	Repeats a loop once for each value provided
WHILE	A conditional loop command with the test at the start

Variable manipulation primitives

change-image compatibility	Changes the size of a VIPS image Checks if two images are of the same size
copy__program	Copies a program from one program variable to another
chain__add	Adds two chains together
chain__copy	Copies one chain into another
chain__delete	Deletes a chain
image__copy	Copies an image from one image variable to another
list__add	Adds two lists together
list__copy	Copies one list into another
list__delete	Deletes a list

VIPS control procedures

debug__handler	Provides traceback information for user commands
execute	Parses and executes a VIPS command line
find__command	Locates a command in the command table
handler	Error handler used by VIPS
initialise	Initializes VIPS
load__command	Loads a user command into the VIPS command table
load__variable	Loads a user variable into the VIPS variable table
main__loop	Calls the VIPS program
match__variable	Obtains a variable name from a 'wildcard' operation
parse__variable	Parses a variable from its name
private__help	Informs VIPS of help libraries for private commands
queue__ctrlc__ast	Control C handler used by VIPS

Miscellaneous VIPS commands

$	Executes a DCL command as a subprocess
CONTINUE	Continues after a set time or when a key is pressed
DECLARE	Declares a VIPS variable
DEFAULTS	Lists the default VIPS values
DEFINE	Defines symbols which may be used as commands
DELETE	Disposes of VIPS variables
EXIT	Exits from VIPS
HELP	Provides information on all of the VIPS commands
INFO	Provides general information on modifications etc.
LET	Assigns one VIPS variable to another
PSEUDO	False colour selection
SET	Enables and disables various controls
SHOW	Displays current VIPS variables

Miscellaneous utilities

angle	Calculates the angle to a point from the origin
call	Calls a procedure with the parameters provided
check__var	Checks if a string could be a VIPS variable name
dispose__temp__image	Deletes a temporary image variable
extract	Extracts an entity from a string
get__address	Obtains the address of a variable or procedure
get__image__type	Returns the type code of an image
get__temp__image	Creates a temporary image variable
obtain__type	From a type code returns a string
option	Checks on option string for a particular option
phase	Calculates the phase of a complex number
read__prompt	A prompted read from the terminal
sdesc	Provides a descriptor to a string variable
str	Returns a dynamic string pointer
vect	Returns a vector from two integers