P. Franchi
J. Gonzalez
P. Mantey
C. Paoli
A. Parolo
J. Simmons

Design Issues and Architecture of HACIENDA, an Experimental Image Processing System

The paper provides the rationale for the architecture and design of a color image display and processing system called HACIENDA. The system was heavily influenced by one of its most important intended applications, the processing of LANDSAT data, including that to be provided by LANDSAT D. Also considered in the paper are the trade-offs involved in making the system suitable for a broader range of image processing work without unduly adding to cost or complexity.

1. Introduction

Image processing projects are underway at many IBM Scientific Centers around the world. These projects often involve the processing of LANDSAT images, for use primarily in agricultural planning. Also, the Research Division of IBM is doing extensive work in image processing and graphics. It had become obvious that a variety of display systems were being used in these projects, each with its own software and system support, so that software developed at one location was not readily usable at another. At the same time, it was noted that any meaningful distinction between image processing and graphics was disappearing. It was apparent that all these graphics and image processing projects would benefit if they had in common a powerful image display, graphics, and image processing system. The system that evolved to satisfy that need came to be called HACIENDA [1].

Design of the HACIENDA system began in 1977. The objective was to design and build a limited number of these systems for use within IBM and to support joint studies conducted primarily with universities. The display was to be configured using available IBM components, so that it could be reliably maintained in remote locations.

The design group reviewed existing display systems, requirements of the various image processing projects in IBM, and the projected requirements for the next decade. The possible features of an image processing and display system were categorized and evaluated. The basis for evaluation was heavily weighted toward processing LANDSAT images, looking forward to the data to be produced by LANDSAT D, but other applications in image processing and graphics were to be accommodated where they would not conflict with support for LANDSAT processing nor significantly contribute to cost or complexity.

In parallel with the hardware development, basic support software (HBUS) and application software (HIPS) were developed. The host environment was chosen (VM/370), software design started in late 1979, and coding in early 1980. The first HACIENDA unit was ready in January 1981, along with HBUS and HIPS. A number of the units are now installed at IBM locations throughout the world.

To prepare the reader for the issues considered later in this paper, Section 2 provides a general description of the HACIENDA system. Its three major components (display

subsystem, image processor, and graphics subsystem) are discussed in greater detail in Sections 3, 4, and 5, respectively. The supporting software is considered briefly in Section 6. A list of documents available to HACIENDA users is given in [2]. For general image processing terminology and concepts, see [3–5].

2. The HACIENDA system

HACIENDA is physically partitioned into a work station and a main logic box.

- *Work station*

The work station is that part of the system most directly in contact with users. It contains the main human-user interfaces (Fig. 1). The work station is a specially designed desk, housing a local power supply and the equipment for communication with the main logic box. On the desk top reside a high-resolution RGB monitor on which image data are displayed and a conversational terminal allowing alphanumeric communication through a standard green phosphor screen and keyboard (an IBM 3278 Model 3). The high-resolution color monitor (purchased from an OEM vendor) employs state of the art display techniques (2:1 interlace, 30 frames/second) and shadow-mask technology [6]. HACIENDA's work station may interface with other I/O units, such as light pen, joystick, graphics tablet, and image hard copy unit.

- *Main logic box*

The main logic box contains all of HACIENDA except the work station components. It includes the

- HACIENDA controller and host-system interface (in the following text, we call this the communication and control subsystem);
- Display subsystem;
- Image processing subsystem;
- Graphics subsystem; and
- Physical support for the above subsystems.

The physical partitioning has been done so that the HACIENDA work station need not be in an air-conditioned room; the work station may be placed at a convenient location within 100 meters of the air-conditioned main logic box.

- *Communication and control subsystem*

The communication and control subsystem (Fig. 2) is a modified IBM 3274 unit driven by custom microcode. It controls HACIENDA operation and interfaces it with the host computer through a standard channel attachment; it is directly attached to the HACIENDA conversational terminal and may be used to interface up to six other IBM 3270 display units to the host, independently of HACIENDA.

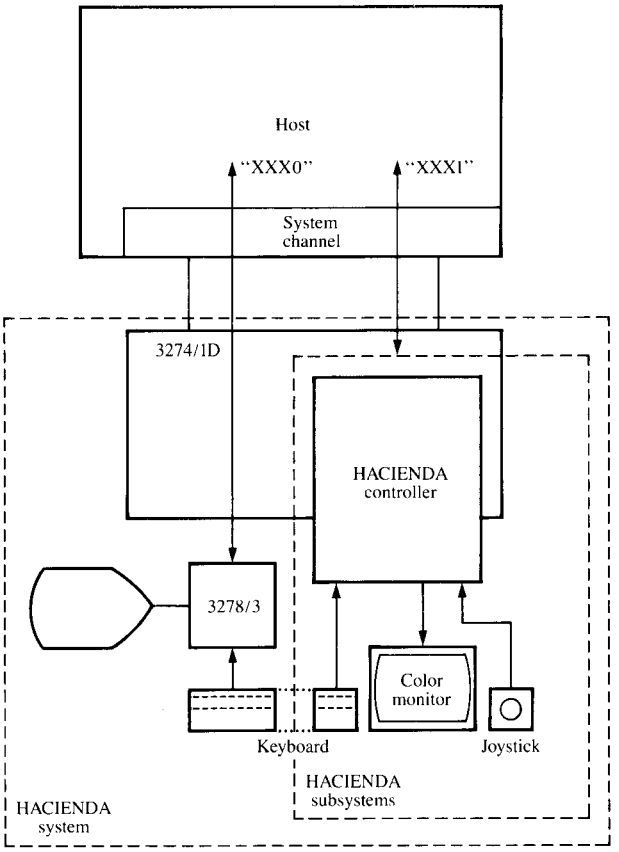

Figure 1 Host/HACIENDA/user interfaces.

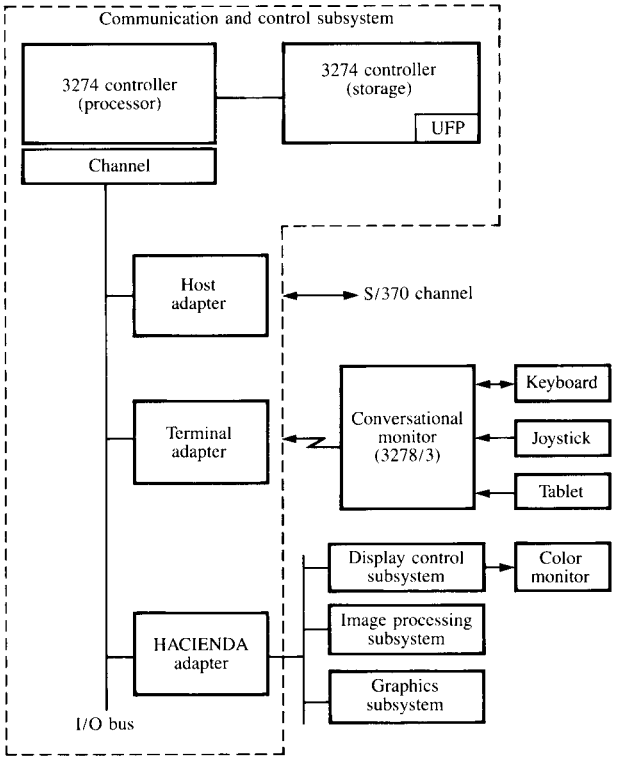

Figure 2 Communication-and-control view of the system.

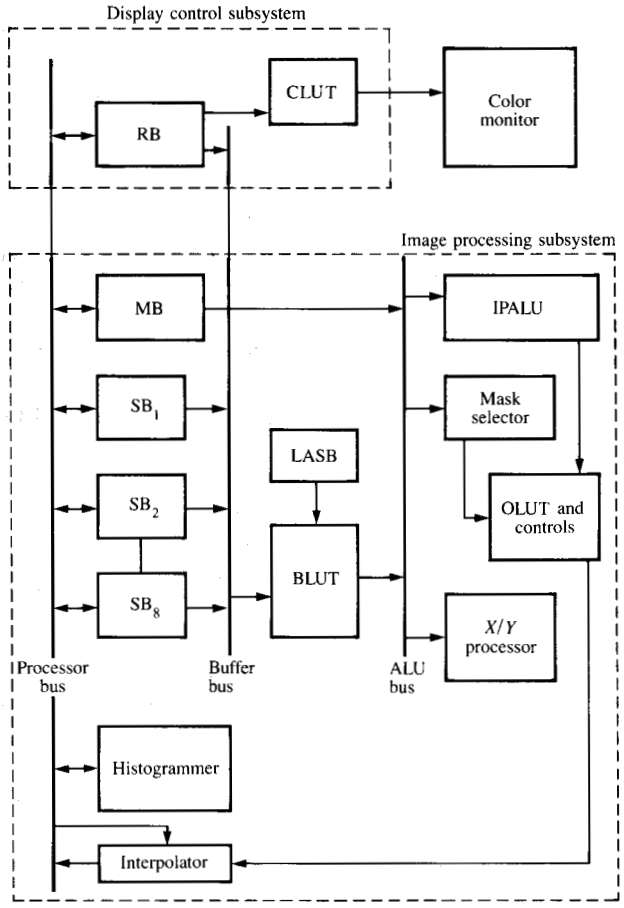

Figure 3 Data flow in the display and image processing subsystems. RB indicates a refresh buffer, MB a mask buffer, and SB a storage buffer.

HACIENDA itself, through its controller, appears to the host computer to be a peculiar IBM 3278 user terminal which can accept "structured field" commands.

This setup avoids much work otherwise needed to interface a new device to a computer and to drive it from application programs. No substantial development cost was incurred for this, as would have been the case if an optimal interface (from a HACIENDA viewpoint) were developed.

HACIENDA is usually driven through its controller by a user program running in the host. This user program is linked to a library of host routines making up the HBUS interface. Alternatively, HACIENDA may operate in a local mode. In this case, the controller runs an interpreter microprogram which allows direct execution of a HACIENDA-resident User Function Program (UFP), stored in controller random access memory. A UFP is down-loaded to HACIENDA by the host computer, where the object code is generated by using the System/370 assembler with a specially written set of macros.

HACIENDA capabilities are the same whether a host program or a UFP is in control. Both options were included, since either one may be preferred under certain usage patterns. UFP control minimizes host loading; it is optimal when a task demands heavy image processing computations and little or no general purpose processing and mass storage access. Host program control is far more flexible, but it ties HACIENDA response time to the speed with which the host can conduct I/O operations; if host load is heavy, response time can be degraded.

The controller can communicate with all HACIENDA subunits at a very low level, such as setting the sources to input buses of an ALU, controlling ALU operation, and determining the destination of the result. This level of interface, more akin to microprogramming than to normal device control, was chosen for maximum flexibility in directing available processing power to varying application goals. Specific application software may, of course, shield the user from this complexity, providing a variety of system settings already partly tailored to the application at hand.

The controller decides which HACIENDA subsystems are active at any given moment. The display subsystem is always active, and, if enabled, builds an image for display on the high-resolution color monitor, without interfering with other operations taking place in the rest of the system. However, actual display of the image may be inhibited. Activities of the image processing subsystem, the graphics subsystem, and communication with the host are mutually exclusive. The controller may start one activity when needed by request of either a host application program or the UFP running in the controller itself.

Mutual exclusion among these three activities seemed rather obvious when the project started. Looking back, however, we see that it imposes limitations. For instance, it would be advantageous to the graphics subsystem to use the image processing subsystem's interpolator unit (to be described later) for anti-aliasing purposes.

3. Display subsystem

The display subsystem is that part of HACIENDA which builds the image displayed on the high-resolution color monitor screen from data stored in the system (Fig. 3).

• *Refresh buffer*

The main source of data for display building is the refresh buffer, a 1024 × 1024-pixel memory. Each pixel has 12 normal bits and one overlay bit. Each pixel of the refresh buffer which is mapped onto the image at a given instant is represented thus: If the overlay bit of the pixel is on, then the corresponding pixel is of the "overlay" color, which is the same for all pixels on the screen; if the overlay bit is off, then

the 12 normal bits of the pixel determine its color. Overlay blinking is also available.

The privileged role of the overlay plane seems justified in view of the many applications which need a simple way to write labels, axes, and captions on an image without disturbing the image itself. However, a strong case could be made for a more regular architecture in which all bit planes had equal status and an appropriate lookup table was used to determine which, if any, had overlaying properties. The extra flexibility would be paid for by the extra complexity in application programs needed to build the required color lookup table (discussed more fully later); however, this could be avoided by providing a suitable "default mode" comparable to the present solution.

The refresh buffer is a flexible digital store with read-modify-write capability. Its contents, besides being used for image building, may be fed to either of the two HACIENDA main internal buses (the processor bus and the buffer bus). A new value for a pixel may be loaded from the processor bus, and then

- The new data for the pixel are mixed with the old data for the same pixel by any one of a variety of operations, including "overwrite" (new replaces old) and arithmetic and boolean operations;
- The result is selectively overwritten to the "old" data only for those bit planes which are enabled for this rewriting.

While the refresh buffer is usually accessed in a raster scan pass (just like other buffers), it can be randomly addressed when data are loaded into it by use of an image processing subsystem X/Y processor.

- *Pixel color determination*

Normal color determination is accomplished through table lookup, with the table continually accessed at video rate during image display. This kind of display architecture is becoming standard for high-quality displays [7, 8].

Table lookup in this case means that the 12 normal bits from a refresh buffer pixel are used to address a 4096-word memory, the color lookup table (CLUT), previously downloaded by the host; the word held in the CLUT at the address given by the pixel value is used to yield the pixel color.

Colors are obtained as 15-bit words, with 5 bits for each of the primary RGB components; 32 intensity steps are thus available for each component and 32 768 colors altogether, although only 4096 different ones can be active at any given time, since the CLUT only has 4096 entries.

This number of colors may appear to be an overkill, since the human eye cannot directly distinguish that many colors [9]. However, many gradually changing hues may be needed in such applications as texturing or anti-aliasing in vector-to-raster conversion.

The color determination words (CLUT entries, overlay color register, cursor color register) are actually 16 bits wide. The most significant bit is normally off, but if it is on, then the pixels which have that "color" are "blinked," i.e., flashed off and on between black and the color hue given by the other 15 bits. The blinking feature was introduced at the specific request of prospective users, despite such obvious disadvantages as the impossibility of reproducing the effect in hard copy and the possibility of operator discomfort if used too extensively.

The CLUT can be bypassed. In this case the 12-bit pixel is used directly as a data word, partitioned so as to provide 4 bits for each primary RGB component; 16 intensity steps are available for each component, 4096 different colors altogether. Blinking is off in this case.

The same effect as CLUT bypassing could obviously have been obtained with a different design of the CLUT that was built. However, the bypass may be used as the "standard" display mode in applications where multispectral data are directly assembled and displayed. In this case the CLUT can be used for a spectral distortion for some specific purpose, such as histogram equalization. CLUT bypass mode allows standard display mode to be resumed instantaneously, without disturbing the "distortion" CLUT. The spectrally altered image may thus also be recovered at once. It was decided that the frequency of use of this application deserved making a special case out of it.

- *Other display subsystem characteristics*

All or part of the refresh buffer can be mapped onto the screen image at any instant; the portion depends on the "zoom factor" (ZF), which can be 1, 2, 4, or 8. To obtain a pixel on the screen, the corresponding refresh buffer pixel is replicated ZF times in both the horizontal and vertical directions. Thus the screen may represent from 128×128 ($ZF = 8$) to 1024×1024 ($ZF = 1$) refresh buffer pixels.

The position in the refresh buffer corresponding to the screen's origin (lower left corner) is determined by the "scroll factor" (SF). Different windows in the refresh buffer may be displayed on the screen at different times. Window size depends on ZF, and window position in the refresh buffer depends on SF.

To mark a point on the screen, a cursor is available. This is a figure which, if enabled, is superimposed on the image. Cursor coordinates may be derived from a variety of sources, such as a host application program, a UFP, or the graphics

subsystem. The user may be given direct control of the cursor coordinates through joystick or graphic tablet action. When the joystick is used to control the cursor, its angular position determines cursor movement rate according to a user-defined table; this makes for maximum flexibility in this very common situation.

The cursor can take either of two shapes: a cross-hair, with screen-spanning horizontal and vertical segments crossing at the cursor coordinates, or any symbol mapped from a dedicated 64 × 64-bit matrix originally down-loaded from the host and centered at the cursor coordinates. In either case, a color from the cursor color register is shown at all points where the cursor must be displayed, while the image shows through elsewhere.

Zooming, scrolling, and cursor mixing all take place, conceptually, "on the screen"; the refresh buffer is never affected by any of this. This concept might have been extended to emulate a direct-view storage tube display's distinction between stored and refreshed strokes, which would have allowed a few extra useful application features, such as "rubber-banding" in graphics. However, it was finally decided that potential advantages were outweighed by the extra complexity.

4. Image processing subsystem

The image processing subsystem provides storage for image data. When enabled by the controller, it can transfer and manipulate those data (Fig. 3). An image processing "cycle" is an event in which up to one million pixels are taken from some buffers, processed by the image processor, possibly interpolated and/or histogrammed, and finally returned to some other buffers.

Storage buffers

The architecture allows up to eight scratch pad buffers; they are 1024 × 1024 memories with 8 bits per pixel, called band or storage buffers. Two to six storage buffers can be physically present inside the main logic box in the present implementation; space limitations forbid the use of all the eight storage buffers allowed for in the architecture.

Like the refresh buffer, storage buffers can provide or accept data to or from the processor bus, which provides communication between all of HACIENDA's subsystems. They can store image data, results or partial results of operations, or data or programs for the graphics subsystem.

All special refresh buffer features discussed earlier would also be desirable for scratch pad memories: direct addressing, read-modify-write capability through a local ALU, selective bit plane overwriting. However, it was decided that the added cost would not be justified by the added usefulness, and moreover, cheaper memory was a better investment at this state of technology than a smaller quantity of sophisticated feature-laden storage.

When participating in image processing operations, all buffers (refresh and storage buffers) are addressed in a raster scan pass within a "processing window," which is any rectangular subset of the buffer. Windows are independently set for each buffer and may include different amounts of data and/or have different origins and aspect ratios for each buffer.

A more radical break with respect to existing practice would have been to provide an "undifferentiated storage resource," an $N_1 \times N_2$ store with N_3-bit words, allowing arbitrary rectangular subsets of this to be used for processing and display. Keeping the maximum at around the present 7.5 megabytes, the store could have been a 2048 × 2048 memory with 16-bit words. Display window and processing windows would have taken nonoverlapping slices of this as determined by the user.

However appealing this scheme may be, with its greater flexibility, it does not appear to be cost-effective at the present time. It allows for no modularity in storage growth, as can be done with the present scheme by the addition of storage buffers. It also presents the programmer with serious storage allocation problems and may give practically insurmountable bandwidth problems if several windows fall inside a single physical memory device. Therefore, no further consideration was given to this scheme.

Buses and lookup tables

The processor bus is HACIENDA's main communication highway. All storage buffers and the refresh buffer can load or unload to or from it. The controller and the graphics subsystem use it for high-speed, cycle-stealing, bulk I/O; the image processor returns its results via the processor bus.

Buffers involved as data sources in an image processing cycle (there may be up to 9) feed their contents to the buffer bus. From there, the 8-bit or 12-bit data are transferred to the 16-bit ALU bus with one of three possible protocols:

- Data are padded with zeros on the most significant bits,
- Data are sign-extended to 16 bits, or
- Data are used to address one of several lookup tables (ILUT) input to the image processor.

The buffer-bus-to-ALU-bus protocol may differ for each source buffer, since data on the buffer bus are tagged according to their source.

Each ILUT is a 256-entry store with 16-bit words. Lookup table operations can be thought of as implementations of any

desired function of one variable; the variable is used as an address and the corresponding result is fetched from that address. Lookup tables are usually down-loaded from the host before actual operation.

Eight ILUTs are available; each storage buffer may use any one of them, and each ILUT may be used by more than one storage buffer. The storage buffer to ILUT mapping is accomplished through a look-aside buffer (LASB) which associates one ILUT with each storage buffer. The LASB is usually down-loaded by the host before actual operation.

This scheme is far more flexible than one in which each storage buffer has a fixed associated ILUT, as originally envisaged. If the same ILUT transformation is to be applied to several storage buffers, the scheme allows for ILUT sharing. If HACIENDA is to operate in a local mode, the host can pre-load all ILUTs with several useful transformation functions; a UFP can easily switch storage buffers between pre-loaded ILUTs as needed, since the mass of data to be actually exchanged is minimal (just the contents of one LASB word).

- *X/Y processor*

Data on the ALU bus can be processed by either the X/Y processor (XYP) or the image processing ALU (IPALU).

The X/Y processor treats ALU bus data as addresses for random access to the refresh buffer. This overrides normal raster scanning and is only available when the refresh buffer is used as the destination for output of an image processing cycle. Two storage buffers (possibly modified by ILUTs) may be designated as the source of X and Y coordinates; they select the destination pixel in the refresh buffer, either in a direct address mode or as an offset from the refresh buffer pixel addressed immediately before.

A randomly addressable mode would also be useful when data are read from the refresh buffer, and we originally planned to implement this. However, the bandwidth of the refresh buffer is already overtaxed as things stand, and it cannot support the extra load.

- *Image processing ALU*

The IPALU operates on N corresponding pixels of input data, one from each source buffer, to produce one resulting pixel of output. For each pixel, operation is as follows.

First, the IPALU accumulator is loaded from a constant register. Then, the contents of the accumulator are operated on N times with an incoming data pixel, intermediate results being stored back into the accumulator. Incoming data may arrive in any order, which restricts the IPALU to symmetrical operations, such as addition and boolean operations. This is not a limitation; subtraction, for example, may be performed by adding one unchanged value (the minuend) to the other one on which a negation operation is performed through ILUT processing (the subtrahend).

An IPALU operation that was not considered for inclusion at design time now appears desirable. This is a function that would select the higher (MAX) of the words applied to it; an obvious dual to this is a MIN function. Both were rejected originally as not being generally useful processing primitives. However, some image processing algorithms (such as simple nonlinear filters) rely on these operations. Since, in the present architecture, they are not supported in a pipelined image processing cycle (as is, for example, addition), they must be provided by software at the expense of several image processing cycles.

The final result left in the accumulator when the N input pixels have been processed may be used as is (to 16-bit precision) or be further processed by another LUT operation.

- *OLUT and contour filling*

The output lookup table (OLUT) may be applied at the IPALU's output. It has 4096 entries of 16 bits each, and is thus addressed by the least significant 12 bits of the final result of the IPALU for each pixel.

Alternatively, the OLUT may allow selective processing according to a "contour filling" operation. A section of the graphics subsystem, described in more detail below, produces for each pixel one bit which is 0 for pixels outside a pre-drawn contour and 1 for pixels inside it. This bit may optionally be used as the most significant one in the OLUT address, with the IPALU result giving only the 11 least significant bits. This allows different output functions for pixels inside and outside the pre-drawn contour [10].

As an example of the power of lookup table processing, consider approximate multiplication of pixel values. One ILUT is loaded with the logarithm of input values (addresses); the OLUT is loaded with the antilogarithm of input values (addresses). After this initialization operation, multiplication can be performed by adding in the IPALU the logarithms of the factors and taking the antilogarithm of the result as the final product. The speed of this operation, after initial lookup table loading, is the same as for normal addition.

- *Interpolator and histogrammer*

Results from the IPALU (and possibly the OLUT) are fed back to the processor bus via an interpolator unit. This unit works as a programmable digital filter [11] with 12 stages (8 forward and 4 feedback) and a magnification factor up to 32. Its coefficients are down-loaded by the host, as is the chosen magnification factor.

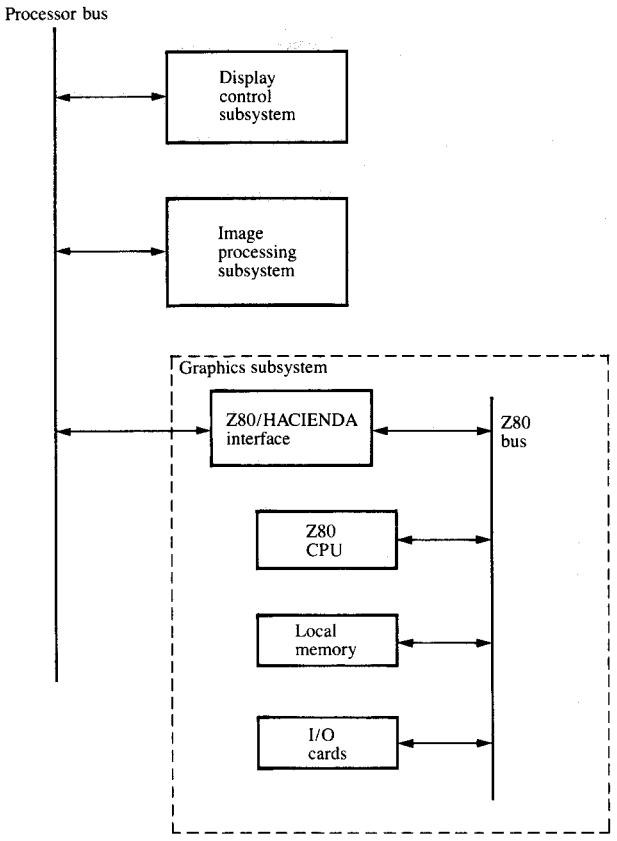

Figure 4 The graphics subsystem.

The interpolator is a one-dimensional unit and in principle cannot be applied to two-dimensional spatial filtering problems, unless the kernel of the 2D filter happens to be decomposable. However, 2D FIR filtering is possible using the interpolator in several image processing cycles (as many cycles as the number of kernel rows or columns, whichever is smaller) [12]. On the other hand, providing to a filtering unit the 3×3- or 5×5-pixel matrix surrounding every pixel to be operated on would require a major overhaul of the HACIENDA architecture, which is based on raster scan processing.

Once back on the processor bus, the results may be loaded into any or all buffers, except those that are producing data in the same image processing cycle. Image processing results, or any other data flowing on the processor bus, may also be fed to another special purpose hardware unit that may be considered a part of the image processing subsystem, the histogrammer.

The histogrammer unit has 4096 32-bit registers that may count in various ways according to the data flowing on the processor bus. This allows statistics to be gathered about data being loaded, unloaded, or processed. Histogrammer registers may be set up and read by the controller, and its exact operation programmed.

This "on-the-fly" histogrammer is not common in previous architectures. We believe it is quite useful, given the importance of statistics gathering in classification and pattern recognition tasks.

5. Graphics subsystem

The graphics subsystem functions described here reflect mainly the architecture of HACIENDA as planned in the design phase, with some desirable features that experience has suggested. Due to cost and time constraints, the actual system offers only a subset of these functions. The improved function and performance of the planned architecture were based primarily on a dedicated microprocessor for graphics. The main function of the graphics subsystem is to generate graphics in the form of characters, vectors, and other primitives. It interprets an IBM 3277 Graphics Attachment data stream [13]. Some extra primitives are offered for direct curve generation. The main improvement, however, is a new subroutine-like "sub-data-stream" concept. Together with scaling, translation, and rotation settings, it allows for great flexibility in handling replicated subsets in one or more drawings (mechanical parts, symbols, logos, etc.).

These functions are performed by a dedicated Zilog microprocessor (Z80) interfaced with the HACIENDA system so as to be able to control all of its operations when enabled by the controller. With respect to controller operation, the only limitation of the graphics subsystem is that it cannot start its own operation nor an image processing cycle, nor can it communicate with the host (Fig. 4).

Dedicated hardware would offer a higher processing bandwidth, but we judged that it would not be worth the additional development and production cost, especially since it would also offer less flexibility.

- *Graphics interpreter and other operations*

Graphic operation of the subsystem is driven by a resident graphics interpreter. In addition to the above mentioned primitives, it is meant to give the user ample room for expansion by allowing extra primitives to be added as Z80 machine code subroutines.

Another important function of the graphics interpreter is memory management. The storage buffers may hold data and programs for the graphics subsystem. Storage buffers are 1M byte each; they cannot be randomly accessed but must be raster scanned. Thus they have the characteristics of a peculiar "mass memory" to the Z80. The graphics inter-

preter handles them as such, providing the "peripheral drivers" for them that an operating system would supply if it were present.

• *Contour filling*

In the main graphic mode, directly driven by the graphics intepreter, the graphics subsystem function is graphic translation. It converts vectors, characters, and curves to sets of pixels to be overwritten or merged with images in the refresh buffer and/or storage buffers. A particular operation interacts heavily with image processing: the contour filling operation.

The main hardware component for contour filling is a 1024 × 1024 dedicated store, the mask buffer. Each mask buffer pixel is a 2-bit word holding two flags indicating whether the pixel is on the contour or off it and whether it is a point where a horizontal ray intersects the contour. These values are determined for each pixel by the graphics subsystem during vector-to-raster conversion (i.e., contour drawing) and merged with previous values through a read-modify-write machine coupled to the mask buffer.

During a subsequent image processing cycle, another finite state machine raster scans the mask buffer, using the flags to give image processing logic a single bit per pixel denoting whether it is inside or outside the contour.

The parity flag algorithm adopted for contour filling is particularly appropriate to the raster scan environment where the image processing is done. It maximizes usefulness of the two mask buffer bits by allowing filling of any curve whatsoever, with or without such peculiarities as self-tangency and self-intersection.The contour itself may also optionally be considered inside or outside the curve. Finally, storage buffers may play the role of mass memories in this case too, allowing several partly drawn curves to coexist and/or be merged.

Particular attention was devoted to the contour filling features because of their usefulness in image processing and especially in raster graphics applications.

6. System software support

HACIENDA is locally attached to a System/370 (or to IBM 3081, 303X, 4331, or 4341) channel via 3274 controller logic and microcode. The application program controls the HACIENDA system by sending/receiving outbound/inbound 3270 data streams [14].

• *Design issues*

Two approaches were possible for supporting HACIENDA: the alphanumeric terminal (3278) and the HACIENDA subsystems could be treated as a single device with escape code imbedded in the data stream for accessing the HACIENDA subsystems; conversely, two independent devices could be defined with two device addresses, one for the terminal and one for the HACIENDA subsystems. One approach leads to different implementation of a significant portion of 3274 interface microcode than the other. In principle a scheme in which the definition of either mode is made at device initialization time would also have been possible. However, time and resource constraints led us to implement only one of the two alternatives. The two-device scheme was eventually chosen as offering easier maintainability of the HACIENDA-unique software and better isolation from the host operating system and 3274 microcode evolution.

The second decision was which host operating system should be selected for HACIENDA application programming. The decision was in favor of VM/370, because this was the system preferred, almost universally, by those expected to receive HACIENDA. This decision, along with the two-device approach, made it quite obvious that VM/370 should be left to support the display terminal as a standard virtual machine [15] console and to support the HACIENDA subsystem "device" under CMS as a special device, dedicated to the user's virtual machine. The two devices are declared as two 3278s at system generation time.

A third and more difficult decision was where to put the borderline between the so-called "basic" and "application" software. In other words, the HACIENDA Basic User Software (HBUS) should be easy to use for application programmers (be user-friendly) but, at the same time, be sufficiently low-level to give access to all the capabilities of the machine (be complete) with appropriate performance.

Completeness was chosen as the first criterion. As a matter of fact the HBUS interface allows the application programmer to activate and make use of all the functions of HACIENDA, without any restriction.

The friendliness target was pursued by designing a user's function library built on top of the HBUS interface (see Fig. 5). This library (HFUL) provides the user with a simplified view of HACIENDA, by higher level routines, where the most commonly used sequences of HBUS calls are coded. The application program can intermix calls to the library with HBUS interface routines.

The HACIENDA Function User Library (HFUL) is expected to expand, hosting the contributions of the HACIENDA user community. The first set of routines was designed having in mind the requirements of the Hacienda Image Processing System (HIPS). HIPS is an application subsystem which, even though primarily designed for analy-

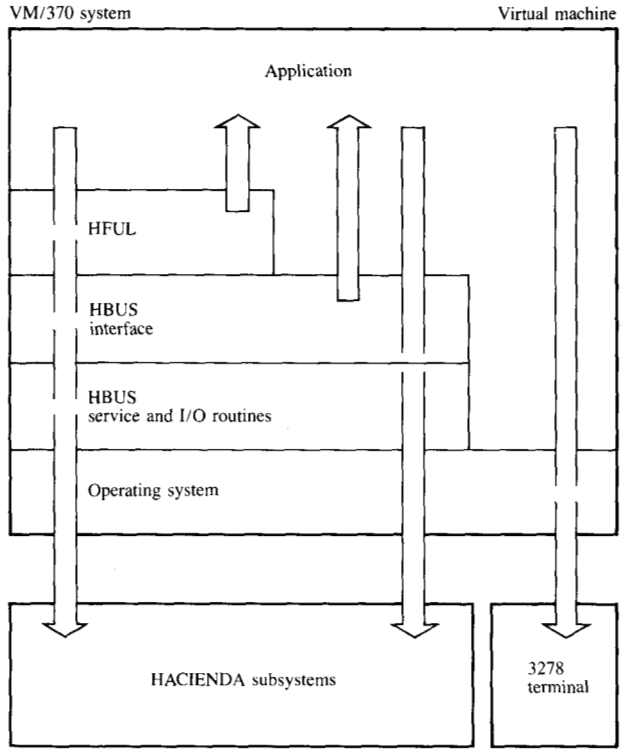

Figure 5 The system software layers.

sis of multispectral image data such as LANDSAT, can be used with all types of images [16].

- *HBUS routines*

All the routines of HBUS are invoked via the standard IBM call interface and as such can be used by image processing applications written in Assembler, PL/I, FORTRAN, Pascal, etc. Two types of routines exist: interface routines, available to the user, and internal routines, for service and I/O operations. Three types of interface routines exist:

- Control routines: used to initialize hardware and software, make available to the application status and asynchronous interruptions, handle errors, etc.
- Define routines: used to set up and prepare the HACIENDA logical subunits for subsequent execution commands.
- Operate routines: used to command the execution of the HACIENDA hardware functions.

Typical examples of operate routines are: load buffers, start image processor cycles, set zoom and scroll, load interpolator coefficients, load/unload histogrammer registers, execute UFP.

Examples of define routines are: define load/unload mode for buffers, define windows, define histogrammer operation mode.

- *UFP support*

Software for supporting local operations (or UFP mode) is based on the following tools, which help the programmer to build, load, and debug user function programs:

- A macro-language to be used in conjunction with a subset of System/370 assembler for coding UFPs. It offers pseudo-instructions such as MOVE, DOWHILE, RETURN, etc.
- A second library to help the programmer to prepare data structures (structured fields) for activating the HACIENDA functions. Names and calling lists are the same as the corresponding HBUS routines, in order to facilitate programming.
- In addition to the above libraries (FPUL), a set of HBUS routines is provided to Load/Unload UFP object code to/from HACIENDA controller storage, Execute and Resume UFPs.
- A controller resident interpreter is provided for execution and interactive debugging of UFP code.

Programming of UFP based on FPUL may appear to be a complex task compared to host programming based on HBUS. However, as already mentioned in Section 2, the key issue for UFP is not functionality but performance. This fact can make the increased programming complexity of the machine, in UFP mode, more acceptable.

7. System performance

The HACIENDA controller provides information to, and accepts information from, the host at an instantaneous byte rate established by the channel or controller, whichever is slower. The instantaneous data transfer rate for write operations is a maximum of 0.65M bytes per second and for read operations is a maximum of 0.4M bytes per second. The transfer rate between controller storage and HACIENDA buffers is of the same order of magnitude. It is independent of write or read operations but varies according to the pixel format (16, 8, or 1 bit per pixel) and the destination buffer(s): refresh, storage, or lookup tables.

The image processing loop is fully asynchronous and employs high-speed processing hardware. Accordingly, it is currently limited by memory access time. An IPALU cycle involving several storage buffers with one million pixels being processed from each takes 3.4 seconds, including possible simple interpolation, contour filling, and histogramming, which add no time to the operation. The time rises to up to 5.4 seconds if the refresh buffer is involved (this includes possible read-modify-write operation on the RB).

The time may rise again if interpolation algorithms are very complex; each million multiplications needed for interpolating adds 0.2 second.

If fewer than one million pixels are involved, time needed decreases almost linearly with the number of pixels. To appreciate the extent to which memory access contributes to processing time, consider that a one-million-pixel operation with "zero-access-time" memory but other hardware unchanged would take 0.2 second for each buffer involved.

The graphics subsystem is limited in speed by the Z80 microprocessor's operations themselves. It takes from 15 to 35 microseconds to plot one pixel, depending on vector characteristics such as length and orientation. This time includes the flag-setting computations for the contour-filling first pass algorithm.

8. Concluding remarks

Although the initial idea of HACIENDA originated more than five years ago, its architecture is still current [17]. In addition to HIPS, other software for remote sensing applications has been developed (DIMAPS-II), which has revealed more clearly other advantages as well as some limitations of HACIENDA [18]. Image processing algorithms specifically designed for HACIENDA are under development [12]. Applications are also in progress in areas quite apart from the original scope, such as seismic data processing for the oil industry and interactive systems for model analysis. Other application possibilities appear quite promising.

Intensive use of HACIENDA throughout the world is bringing to IBM a valuable amount of experience in a variety of image processing applications.

Acknowledgments

The successful completion of the HACIENDA project is the result of the synergetic effort of many people from several countries, who at various levels—management, technical, administration, and support—made their contributions. The authors of this paper constitute only a limited sample of those people. While it is not possible to recognize all contributions from the various IBM functions directly involved in the development of HACIENDA, the authors wish to express particular appreciation to L. M. Branscomb, who has encouraged and supported this project since the very beginning, and to R. Aguilar, whose enthusiastic support made this project a reality.

References and notes

1. The IBM 7350 Image Processing System, which was announced in April 1982, is a successor to HACIENDA and was derived from experience gained with that system. The two systems provide very similar capabilities and software interfaces.
2. P. Franchi, "HACIENDA: A System for Color Image Display and Processing," *Technical Report G513-3586,* IBM Scientific Center, Rome, Italy, December 1981.
3. W. Pratt, *Digital Image Processing,* John Wiley & Sons, Inc., New York, 1978.
4. R. Bernstein, "Digital Image Processing of Earth Observation Sensor Data," *IBM J. Res. Develop.* **20,** 1, 40–57 (January 1976).
5. E. Hall, *Computer Image Processing and Recognition,* Academic Press, Inc., New York, 1979.
6. S. Sherr, *Electronic Displays,* John Wiley & Sons, Inc., New York, 1979.
7. J. Adams and R. Wallis, "New Concepts in Display Technology," *Computer* **10,** 8, 61–69 (August 1977).
8. H. C. Andrews, "Digital Image Processing," *IEEE Spectrum* **16,** 4, 38–49 (April 1979).
9. R. Luccio, "Psicologia della Percezione del Colore," Note da Seminari e Conferenze, Elaborazione del Colore, June 1981, available from IBM Scientific Center, Rome, Italy.
10. The announced IBM 7350 is planned with a 65 536-entry OLUT. In this machine the addresses of OLUT entries are generated either by the 16 bits of the IPALU output or its 15 least significant bits.
11. A. Oppenheim and R. Schafer, *Digital Signal Processing,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975.
12. F. Ramirez, "Local Operations at HACIENDA," *Internal Report,* February 1982, available from the IBM Scientific Center, Madrid, Spain.
13. *IBM 3277 Graphic Attachment RPQ* (7H0284) *General Information Manual,* Order No. GA33-3039, available through IBM branch offices.
14. *IBM 3270 Data Stream Programmer's Reference,* Order No. GA23-0059, available through IBM branch offices.
15. *IBM Virtual Machine/System Product: Introduction,* Order No. GC18-6200, available through IBM branch offices.
16. "HACIENDA Image Processing System: General Information Manual," *Technical Report F-018,* IBM Scientific Center, Paris, France, November 1981.
17. C. Reader and L. Hubble, "Trends in Image Display Systems," *Proc. IEEE* **69,** 606–614 (May 1981).
18. J. V. Dave and R. Porinelli, "DIMAPS-II: DIMAPS for HACIENDA," *Technical Report,* IBM Scientific Center, Palo Alto, CA.

Received March 26, 1982; revised October 14, 1982

Paolo Franchi *IBM Italy, Via Giorgione, 129, 00147 Rome, Italy.* Dr. Franchi received his degree in physics from the University of Milan in 1964. During the period 1964 to 1968, he carried out postdoctoral work at the Institute of Physical Sciences at the University of Milan, studying propagation of electromagnetic waves, plasma physics, and modeling of multiparticle systems. In 1969 he joined the IBM Scientific Center in Pisa, Italy. In 1971, while on assignment at the Grenoble, France, Scientific Center, he worked on virtual storage and time-sharing systems. He returned to Pisa in 1972 and started design and development work in data communication. From 1973 to 1974 at the Cambridge, Massachusetts, Scientific Center, and until 1977 at the Pisa Scientific Center, he worked on developing the packet-switched computer network of Italian universities. Dr. Franchi moved to the Rome Scientific Center in 1978 and since then has been working on HACIENDA. In 1979 to 1980 he managed the basic software development. Currently he is manager of image processing activities at the Scientific Center.

Jorge Gonzalez *IBM Spain, Paseo de la Castellana, 4, Madrid 1, Spain.* Dr. Gonzalez received a degree in telecommunication engineering from the Polytechnic University of Madrid in 1972 and an M.S. in electrical engineering from the Carnegie-Mellon

University, Pittsburgh, Pennsylvania, in 1973. In 1981 he obtained the doctoral degree from the Polytechnic University of Madrid in the area of digital filtering. In 1974 he joined the IBM Madrid Scientific Center, where he has been involved in image processing, digital filtering, signal coding, and image processor architecture. He contributed to requirement definition and design of the HACIENDA image processing subsystem. Dr. Gonzalez is a member of the Institute of Electrical and Electronics Engineers.

Patrick Mantey *IBM Research Division, 5600 Cottle Road, San Jose, California 95193.* Dr. Mantey received his Ph.D. in electrical engineering from Stanford University in 1965. He was a member of the Stanford faculty for several years, and continues to lecture at Stanford in electrical engineering since joining IBM in San Jose in 1967. His B.S. and M.S. are also in electrical engineering, from the University of Notre Dame and the University of Wisconsin, respectively. Dr. Mantey has published numerous papers in the areas of system theory, digital filtering, control and information processing, data base and graphics systems and applications, and decision support systems. He organized and managed a pioneering project in geographic data systems, and his group made significant developments in the area of interactive graphics involving relational data bases and color raster displays. He has been issued several patents in the areas of digital filtering, computer graphics, and image processing. At the request of Dr. Lewis Branscomb, IBM Vice President and Chief Scientist, he organized and chaired the task force that developed the HACIENDA architecture, carrying it to its transfer to IBM Italy. He is currently manager of the Experimental Systems Department, computer science, in the San Jose Research laboratory, continuing work in interactive applications involving image, data base, text, and documents, as well as VLSI design.

Carlo Paoli *IBM Italy, Via S. Maria 67, 56100 Pisa, Italy.* Mr. Paoli attended the Istituto Tècnico Industriale Galileo Galilei, majoring in electronic engineering. From 1967 to 1969, he worked as system programmer at CNR Computing Center in Pisa. He joined IBM in 1970 at the Pisa Scientific Center. From 1972 to 1977 he carried out design and development work for the Italian packet-switched computer network, a joint project among IBM, CNR, and a number of universities (RPCNET). At Boca Raton, Florida, in early 1979, he participated in programming RPCNET architecture on the System/1. Since 1979, he has been working on HACIENDA. He participated in the design and development of the system support software (HBUS). Mr. Paoli is currently a research staff member at the Scientific Center in Pisa, working in image editing and processing applications.

Albertao Parolo *IBM Italy, Via Lecco, 61, 20059 Vimercate, Italy.* Dr. Parolo received his degree in electronic engineering in 1978 from the Politècnico di Milano University, where he was involved in the development of the ground tracking station for the SIRIO Italian experimental communication satellite. He joined IBM in 1979 and since then has been with the HACIENDA development team as a hardware logic designer. Dr. Parolo is a member of the Computer Society and the Institute of Electrical and Electronics Engineers.

John Simmons *IBM United Kingdom Laboratories Ltd., Hursley House, Hursley Park, Winchester, Hampshire S021 2JN, England.* Dr. Simmons is a member of the graphics design group at the Hursley laboratory and is currently working on the design of future graphics products. He received his B.Sc. and Ph.D. in physics at the University of Birmingham, England, and worked for many years in pattern recognition techniques as applied to the evaluation of bubble chamber data in the field of high-energy physics. He joined IBM at the Hursley laboratory in 1974 and has been associated with terminal systems since that time. Following the initial design of the HACIENDA image processing system, while a member of the Special Engineering Department, Dr. Simmons spent two years at the Vimercate plant in Italy as main system architect.