Continuous Processing of Images through User Sketched Functional Blocks

Aydın Kaya, Bülent Özgüç*

1. Introduction

The history of image processing is relatively recent compared to other fields but it has been applied to practically every type of imagery with varying degree of success, some of which are electronics, photography, pattern recognition and medicine. Several factors encourage the future of image processing further. A major factor is the declining cost of computer hardware and the increasing availability of digitizing equipment (e.g. a TV camera interfaced to a computer)[1]. As a result it is now easy to store an image in digitized form. Another factor, promoting image processing, is the availability of better display devices and low-cost but large main memories.

Images in their natural form are pictorial that is they cannot be processed by the computer directly unless they are transformed into numeric or discrete data. Therefore an image should first be converted into numerical form. This conversion process is called digitization and it generates an image in the form of a collection of dots, called picture elements or pixels. In a colour system the value of each pixel (usually between 0 and 255) is an index to a special table, which is called the colour map look-up table. The entry of the colour map table, corresponding to a particular index contains three values and these values are in fact the intensities of three major colours, namely red, green and blue. In a gray scale device, the same idea of referencing a table could be used or the value of a pixel could be treated as an intensity. In an image the combination of all pixels form a rectangular grid or a 2-dimensional array, therefore any pixel can be addressed by specifying its address in terms of a row number and column number.

After an image is represented in numeric form, now it is possible to process it that is, a series of operations can be applied to it to obtain different forms of the original image. The operations to be applied to an image are classified according to the way that the value of output pixels are obtained. If an operation generates an output pixel whose value is obtained by using only its previous value, this is called a point operation. A local operation is the one in which the value of a pixel is changed according to its old value and the values of neighbouring pixels; this operation is sometimes called an area process. A global operation is the one in which all pixels of the image are treated in the same way. When the positions or arrangemenet of the pixels are changed this operation is called a geometric process. An operation changing the value of a pixel by comparing two or more images is called a frame process[2-4]. Some of these operations are going to be used as pictorial examples when the interface is being discussed.

2. The User Interface

The part of a system that controls the communications between the system and the user is often called the user interface. With the introduction of highly sophisticated, multi-tasking, multi-window personal computers and workstations, the science of user interface design gained a special momentum. It is of great importance to pay careful attention to the design of interactive user interfaces. A very sophisticated, fully functional software system can be totally invalidated because of poor user interfaces even in the hands of an experienced user, let alone a novice[5,6].

It is rather ironic that a new common data type, digitized pictures, is mostly processed through command language type or keyword menu type interfaces in the majority of image processing software available today. The nature of the data is pictorial and the algorithms applied to it usually follow sequences or paths resembling a network, or maybe a logic diagram. It then follows as a natural argument that a user dealing with an image processing application, should interact with the system among these lines. The basic scope of this paper is to explore such methods of interaction and suggest ways of software development for image processing applications.

Our aim is to implement an image processing system with the previously defined algorithms through the use of functional blocks connected together to form a diagram that we call a schematic. A functional block means the representation of a routine or function by a visual object in our intent. The arc connecting two blocks is the path and direction through which an image is transferred. In the development of the system

* Department of Computer Engineering and Information Sciences
Bilkent University
Ankara
Turkey

North-Holland
Computer Graphics Forum 7 (1988) 273-280

we relied on the usage of some new developments in software sciences providing for the construction of better interfaces. Most important of these are the object oriented style of programming, iconic interfacing, multiple window and multi-tasking operating systems. Even though the intention of this paper is not to explain these rather recent developments, it is still useful to give some limited details since the whole system is based on these new ideas.

2.1. Object Oriented User Interface Model

A toolkit based on the SunView*[7] window manager is expanded to form our user interface with its menus, windows, panels, many kinds of panel items, and a schematics capture system. As long as there is no need to enter text, the system is used by a pointing device like a mouse. These components have the characteristics of an object in the sense of object oriented programming style even if they may be implemented by using an ordinary language like C[8].

The functional blocks are represented in the form of icons both on the menu that they are chosen from and on the schematic in which they act as nodes. Icons are small pictures (usually they are composed by 64 by 64 pixels) and they represent the available image processing algorithms. The functional block selection menu together with a simple schematic is shown in Figure 1. The image processing algorithms these blocks perform, together with their control variables are given in Table 1.

Other building blocks of the system are the control panel and the items they contain, sliders and scroll bars, text entry windows and a set of special purpose windows serving as data templates, often referred to as a variable control window, as explained later. It is also important to note here that the term object refers to

*SunView is a registered trademark of the Sun Microsystems.

FUNCTIONAL BLOCK NAME	IMAGE PROCESSING APPLIED	CONTROL VARIABLES OR SETTINGS
AVERAGE	Enlarges an image by pixel replication or reduces an image by pixel averaging	Shrink or Expand
HISTOGRAM	Generates a histogram of the current image by counting the number of times an intensity occurs	none
CONVOL	Enhances edges by replacing pixel's value with the sum of that pixel's value and its neighbours by weighting each value by a factor (convolution kernel)	Size of the convolution kernel. The contents of the kernel
DITHER	A colour or grayscale image is reduced to be displayed on a bilevel device by adding an error term to the image intensity of each pixel before comparing it with a threshold value	none
THRESHOLD	An image is reduced to only a background and foreground colour where all values below the threshold are made background and above are made foreground	Threshold value
MEDIAN	Each pixel value is replaced with the middle value of the sorted neighbours	Size of the neighbour window
AND/OR	Two images are combined in a frame process	AND, OR, XOR, PLUS, or DIFFERENCE
POINT OP	Point opererations of the contrast and intensity enhancement	Contrast enhancement value. Intensity enhancement value
DISPLAY	Displays the image currently being processed in a window	Colour map
+/-	Reserved for another set of frame processes	none
LOAD	The input process that can be through a camera or a disk file. Currently only a disk file is read	Filename
SAVE	Output procedure for recording the current image in a file	Filename

Table 1: The Functional Blocks, Their Corresponding Algorithms and Control Variables.

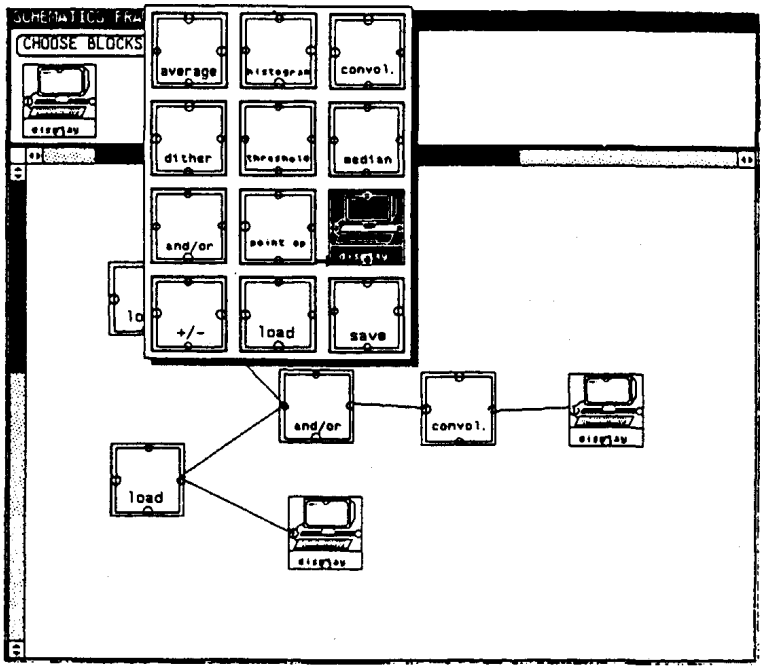

Figure 1: The Functional Block Selection Menu.

both an entity in the real life, like a picture, and the term "object" in the "object oriented style of programming"[9]. The difference should be understood from the context. The object classes of the functional blocks and the panel items represent the available choices to the user. The objects have their particular properties and the user tells the system the kind of processing he wants by arranging the icons representing objects of particular functional classes in a network like diagram that we have called schematics. By pointing to the instances of various object classes in the schematics, the properties of object instances could be changed through the data template window that temporarily appears. This particular window is in fact a direct reflection of the object variables for that particular image processing instance. Each instance starts with a default set so it is not necessary to modify these variables that either control the executional variables and constants, or the sub-algorithms to be chosen. In the case of instances that are from the class of display devices, the data template window is the reflection of the image. This is indeed how intermediate images could be displayed, that is, display devices could be linked to internal nodes in the schematics and then their data templates are activated. Other visual object classes are the menus for selection or process control, and the scroll bars for moving images in whichever window they might happen to be.

In more technical terms, an object is an entity that presents a functional interface. Various details of implementation as to how the objects perform their tasks are not exposed to the user. All one has to do is to choose an object, if it belongs to a functional block class instantiate it, and if other than the default values are going to be used, set the instance variables through the data template window. Figure 2 shows the general flow of the interactive system.

When an object is chosen, it calls its associated functions in an indirect manner, that is, there is not a predefined sequence of functional calls, but the objects are active processes and are invoked by message passing paradigm when a preregistered event occurs. Figure 2 shows the general flow of the interactive system. There is however a differentiation between a user provided event and a system provided event. If the user is dealing with objects belonging to the functional classes directly, his messages are trapped by the schematics editor so that the schematics could either be constructed, edited, or the variables of the instances altered (Figure 3).

If however, a menu choice for execution is made, user events are temporarily blocked and each connected instance of a functional block executes its algorithm and passes its results to the next one. This is shown in Figure 4. The implementation of the flow and internal message passing paradigms are done through linked lists[10]. There is a simple rule checker that looks for cyclic schematics by the traversal of the list to avoid infinite loops. To notify the user of a particular stage,

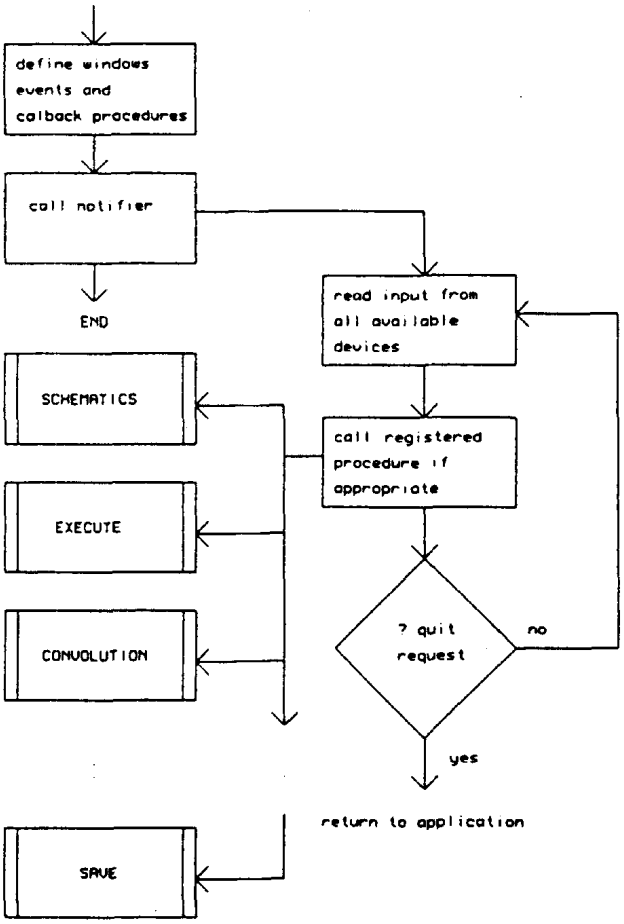

Figure 2: The General Flow of the Interactive System.

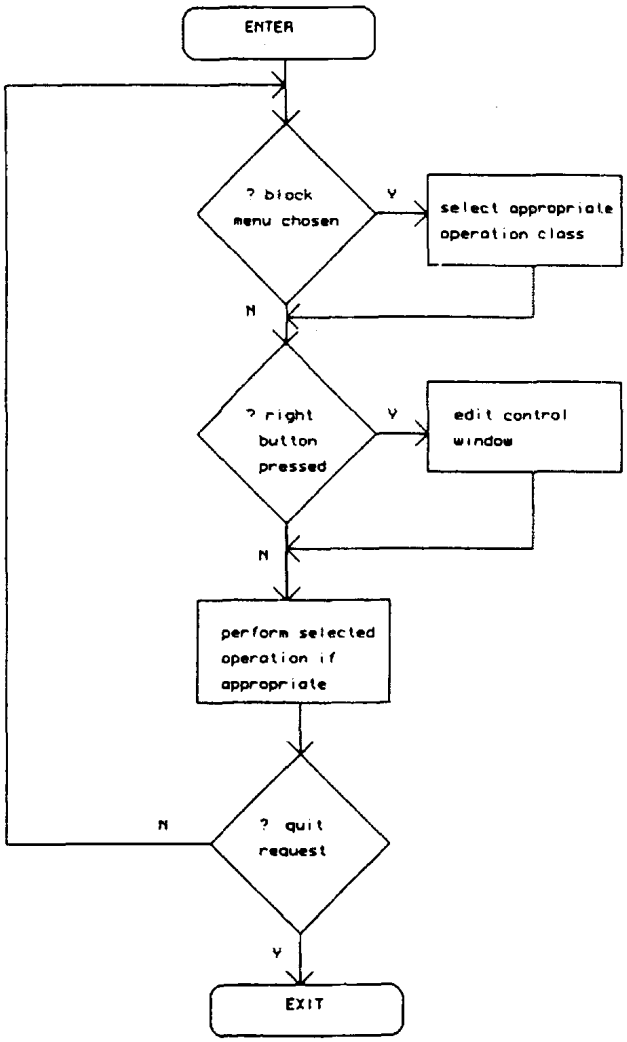

Figure 3: The Schematics Flowchart.

each functional object class instance notifies the formatter of the schematics window so that the relevant icon changes its colour when executing.

3. The Functional Blocks and Their Use

When a user wants to use an available function or routine it is enough to choose the associated icon of the object from a menu by pointing to it through a mouse. Then the user places the representation of the functional block, picked up from the menu, in a drawing window or canvas by attaching it to a mouse button and dragging it to the desired position. When the placement is completed, the user can connect the block to another, by pressing a mouse button on a pin of the start block and dragging the mouse until the rubber band line connected to it touches to a pin of the second block. In this case a link between two functional blocks will be set up both internally (a new instance of the block will be generated and its variables will be set up with the values supplied by either the user or the system itself) and externally. An external link in fact is a line between the pins of the functional block representations and it is the responsibility of the user to make the proper appearance of external links. Figure 5 shows a schematic constructed by a user to process an image.

The user continues in this manner until he forms a "flowchart-like" schematic and when he believes he has finished the construction he can start the processing by pressing a button in the control panel.

During the construction of the schematic a user can make any changes to the layout of the blocks. For example he can disconnect two blocks, remove a block from the drawing window, etc. Such changes in the schematics cause the necessary refreshing of the window and updates the internal links between the blocks by an automatically generated broadcast message.

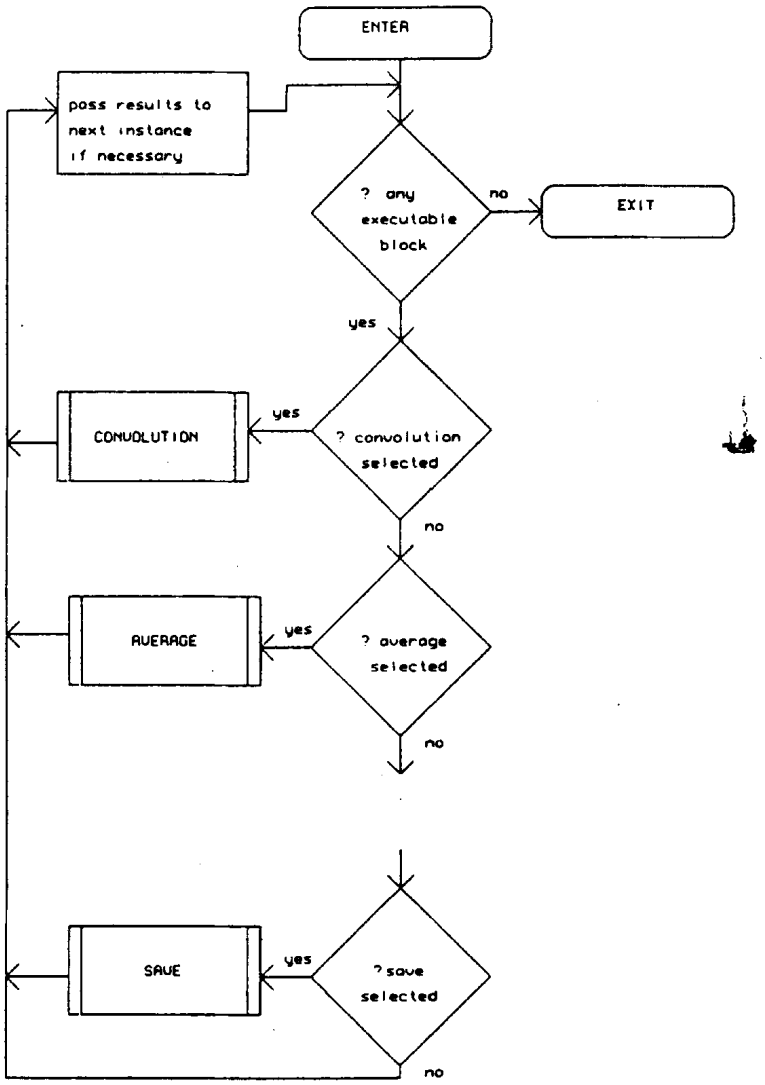

Figure 4: The Execution Flowchart of the System.

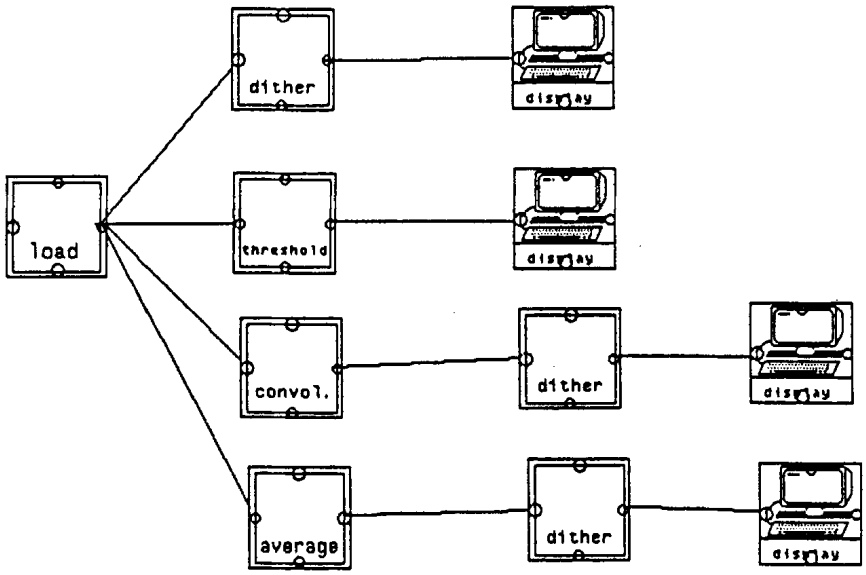

Figure 5: A Schematics to Process an Image.

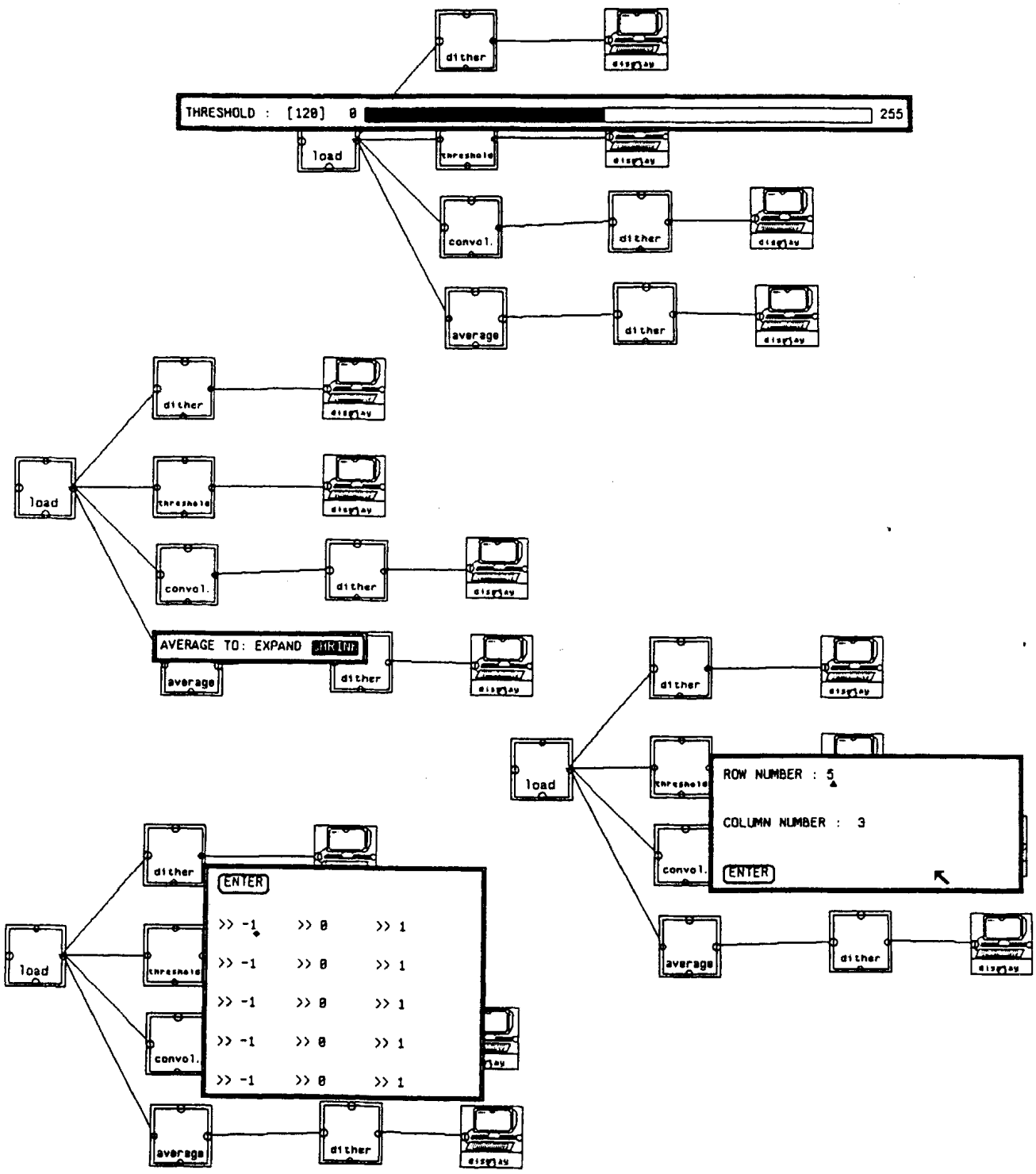

Figure 6: The Usage of the Data Template Windows.

The processing or the execution of the schematics is the interpretation of the functional blocks and the internal links between them. Until all the blocks in the schematics are exhausted, the appropriate functions are invoked through a message passing paradigm and the necessary parts of the obtained results are passed to the next functional block if one exists.

During the placement of blocks on the canvas (drawing data template window of the schematic function) the user is free to change the settings of an object and he does this by pressing a mouse button on the representation of the object on the canvas. In this case a data template window for the selected functional block is opened and the user makes the necessary updates related to the object class instance. Figure 6 shows the usage of data template windows to make changes on the settings of an object instance. Table 1 shows the values that can be set for the functional

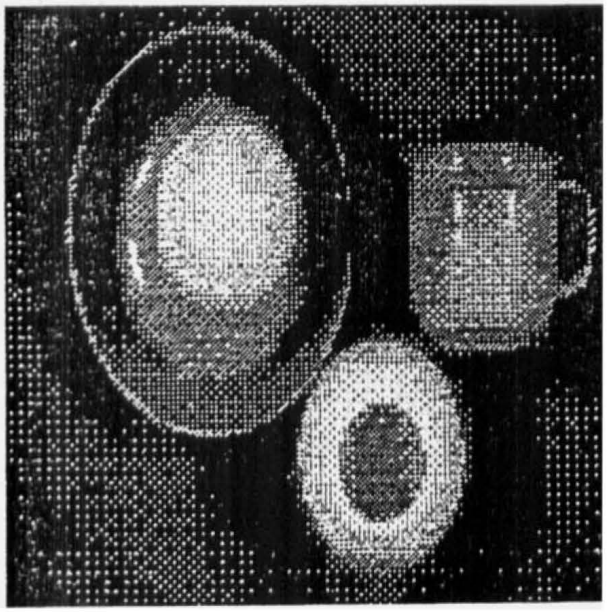

Figure 7: The "Orange" has been Dithered.

Figure 8: The Threshold of the "Orange".

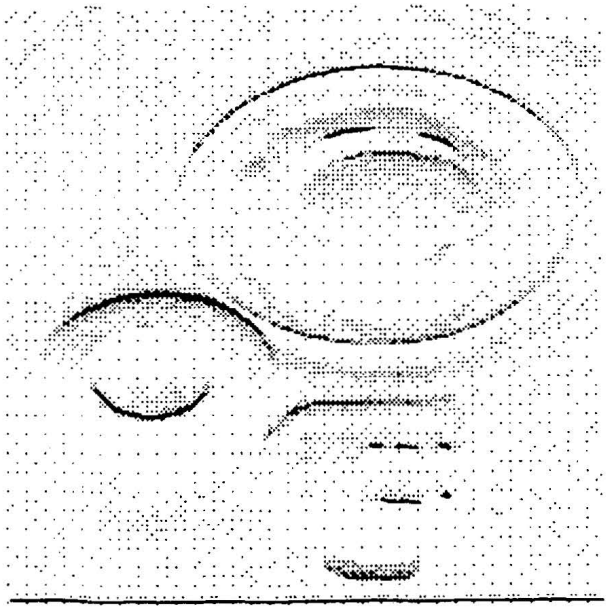

Figure 9: The "Orange" has been Convolved to Amplify the Vertical Lines.

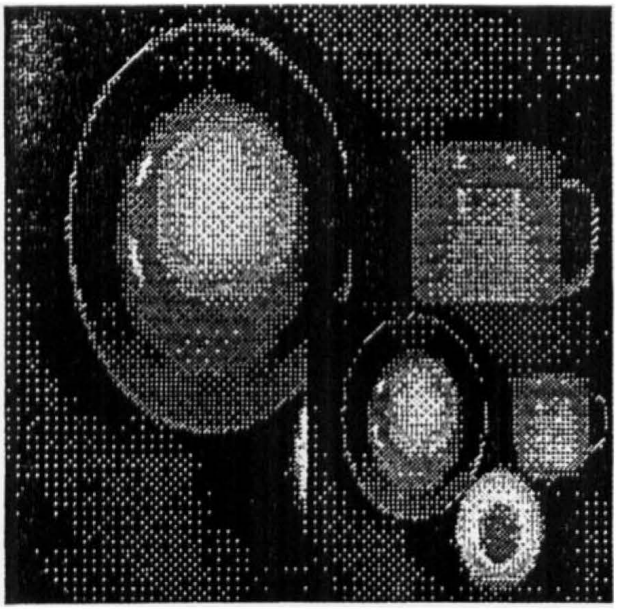

Figure 10: The "Orange" has been Shrunk.

block instances. These windows are used, in the example, to set a threshold value of 120, to set the average to shrink, and to set the convolution kernel size and then enter its values. The results of the four display devices are shown in Figures 7 to 10.

4. Conclusion

The system and style of interface described presents ideas for interacting with image data. Many tools, standards, and methods have been developed for interacting with textual and graphical data. Image data still waits for its share of standards both in data communication and in storage-retrieval techniques and formats, let alone in interaction principles. The approach suggested seems suitable since it represents the actual events taking place during image manipulation procedures. This is not much different, say, to the insertion of special commands and characters in text formatting.

Images displayed in some special device data template windows can be edited manually by operations such as cut and paste, colour map manipulation, painting, etc. This is an ongoing part of the project.

With new entries to the VLSI market, some image processing functions are implemented in hardware. This increases the speed of many such functions, and perhaps the approach described in this paper could be run in real time where the input device is a CCD camera and each functional block is a reference to a chip on a board. Whatever the environment may be, the use of highly interactive and user friendly pictorial interfaces is believed to be a feasible solution in computer imaging applications.

References

1. Kenneth R. Castleman, *Digital Image Processing*, Prentice Hall Inc., Englewood Cliffs, N. J. (1979).
2. Donald Hearn and M. Pauline Baker, *Computer Graphics*, Prentice Hall (1986).
3. David F. Rogers, *Procedural Elements For Computer Graphics*, Mc Graw Hill (1985).
4. Benjamin M. Dawson, "Introduction to Image Processing Algoritms," *BYTE*, pp. 169-186 (March 1987).
5. Bülent Özgüç, "Thoughts On User Interface Design For Multi Window Environments," pp. 477-488 in *Second International Symposium on Computer and Information Sciences*, Istanbul (1987).
6. William M. Newman, Robert F. Sproull, *Principles of Interactive Computer Graphics*, McGraw Hill (1979).
7. Sun Microsystems, Inc., *Sun View Programmer's Guide*, Mountain View, California, 1986.
8. Owen M. Densmore, David S. H. Rosenthal, "A User-Interface Toolkit in Object-Oriented POSTSCRIPT," *Computer Graphics Forum* **6**, pp. 171-180 (1987).
9. O. M. Nierstrasz, "What Is Object In Object-Oriented Programming," pp. 1-13 in *The Proceedings of the CERN*, Renesse, The Netherlands (1986).
10. Jonathan P. Jacky and Ira J. Kalet, "An Object-Oriented Programming Discipline For Standard Pascal," *Communications of the ACM* **30**(9), pp. 772-776 (September 1987).