Continuous Processing of Images Through User Sketched Functional Blocks

A THESIS

Submitted to the Department of Computer Engineering and Information Sciences and the Institute of Engineering and Sciences of Bilkent University in Partial Fulfillment of the Requirements for the Degree of Master of Science

By
Aydın KAYA
June, 1998

QA
76.9
K18
1998

CONTINUOUS PROCESSING OF IMAGES THROUGH USER SKETCHED FUNCTIONAL BLOCKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND

INFORMATION SCIENCES

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Aydın KAYA

June 1988

Aydın Kaya

tarafından bağışlanmıştır.

QA
76.9
K18
1988
B 1464

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Bülent ÖZGÜÇ (Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Mehmet Baray

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Neşe Yalabık

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Sciences

ABSTRACT

CONTINUOUS PROCESSING OF IMAGES THROUGH USER SKETCHED FUNCTIONAL BLOCKS

Aydın KAYA
M.S. in Computer Engineering and
Information Sciences
Supervisor: Assoc. Prof. Dr. Bülent ÖZGÜÇ
June 1988

An object oriented user interface is developed for interacting with and processing images. The software prepared for this purpose includes image processing functions as well as user friendly interaction tools both of a lower level such as menus, panels, windows and a higher level such as a schematics. The lower level utilities provide direct interface with the available image processing functions. At the higher level, the nodes of the schematics serve as image processing function instantiations and the arcs are the paths through which processed images flow. By constructing such a schematics, a complex set of operations can be applied to images continuously.

Keywords : Image-processing, user interface design, window maneger, object-oriented programming.

ÖZET

KULLANICININ ÇİZDİĞİ BLOKLARLA SÜREKLİ OLARAK GÖRÜNTÜ İŞLEME

Aydın KAYA
Bilgisayar Mühendisliği ve Enformatik Bilimleri Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Bülent ÖZGÜÇ
Haziran 1988

Görüntülerin işlemesi amacı ile nesneye yönelik bir bilgisayar, kullanıcı etkileşim sistemi geliştirilmektedir. Bu amaçla hazırlanan yazılım alçak düzeyde ve yüksek düzeyde kullanılabilen görüntü işleme foksiyonlarını ve dostça kullanıcı bilgisayar etkileşim elemanlarını içermektedir (pencereler, kontrol panoları , vs). Alçak düzeyde görüntü işleme fonksiyonları ile direk etkileşimli olarak çalışmak mümkündür. Yüksek düzeyde ise belli bir şekilde oluşturulan bir diagramın düğümleri görüntü işleme fonksiyonları yerine ve düğümler arasındaki bağlar ise işlenen görüntülerin takip edeceği yolların yerine geçmektedir. Böyle bir diagram aracılığı ile karışık bir operasyonlar kümesi sürekli bir şekilde uygulanabilmektedir.

Anahtar Kelimeler: Görüntü işleme, nesneye-yönelik programlama, çok pencereli iş düzeni, etkileşim sistemleri.

ACKNOWLEDGEMENT

I would like to thank my thesis advisor, Assoc. Prof. Bülent ÖZGÜÇ for his guidance and support during the development of this study.

I also appreciate my colleagues, Mesut Göktepe, Murat Karaorman, Ozan Azbazdar and Ahmet Coşar for their valuable discussions and comments.

I also thank my wife Beyhan Kaya for her understanding and patience during the development of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Digital image processing is the science of manipulating the digital representations of objects through a digital computer. Even if the history of this science is quite short compared to other fields, it has been used in many areas some of which are electronics, photography,and medicine.

Some people claim that transportation and data communication is in a very big competition and the winner will make the other one almost useless. If we accept this view, image processing becomes very important for the communication of people and information since it is an accepted fact that pictures or symbols convey messages better than other means. Supporting the above hypothesis there are currently many efforts to combine text and voice with images. Furthermore in a large number of software implementations, designers use symbols, icons, and images to provide a better and improved user interaction.

As there are theorems, rules, procedures etc. in other sciences so are there for image processing. The procedures or algorithms of this science are mostly suitable to be implemented by a computer. In fact there are some image processors in the market and they implement some algorithms directly in the hardware. There are also some available image processing systems in which algorithms are implemented by programs, that is in the software. In most of the systems above the algorithms are usually implemented in such a way that the sequence of applying some algorithms to some images cannot be programmed at the beginning, but instead the user has to wait for the result of one procedure so that it can be used by another procedure. This strategy is clearly distracting in some applications like creation of instances of computer animation where the instances are slightly different from each other so that they can be obtained from the previous ones by making small modifications. This strategy can be used in systems where many different images are passed through the same process or an image is passed through

many image processing procedures and obtained images are stored for animation. If the user has enough skill he or she can arrange the processing procedures in such a sequence that many different instances of images can be obtained and saved without waiting for the intermediate results.

The image processing system described here allows two different modes for using the available image processing functions. The first mode of using the system is the direct mode in which image processing functions are invoked by just pressing some buttons in a control panel. On the other hand if the system is used in the second mode, namely in the indirect mode then a flowchart like diagram should be constructed. The constructed diagram which we call a schematics is used to select the image processing functions, their execution order and their inter dependency.

The organization of the following parts of this thesis is as follows. In chapter 2 an overview of digital image processing is given. This chapter also contains information about the environment of image processing and basic terminology of raster scan display systems.

Chapter 3 classifies image processing algorithms and then describes the available image processing functions by giving examples.

Chapter 4 describes the implementation of the system in detail. In this chapter first the importance of the user interface is discussed and then using the system in the direct and indirect modes are explained by emphasizing the user interface. Internal representation of a constructed schematics and the execution strategy of a schematics are also explained.

Appendix A presents a user's manual for those interested in using the system.

Appendix B gives instance values of nodes in a particular schematics.

Finally appendix C presents the data structures in the C language.

2. OVERVIEW OF DIGITAL IMAGE PROCESSING

2.1 What is Digital Image Processing

An image is a representation of an object and digital image processing is the science of manipulation of those representations by using a computer. 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) [9]. As a result now it is easy to store an image in the digitized form. Another factor, promoting image processing, is the availability of better display devices and low-cost but large main memories.

2.2 The Environment of Digital Image Processing

The main tool of image processing is a digital computer, in addition an input device to capture and digitize an image and an output device to display the processed image are needed. Images in their natural form are pictorial, that is they cannot be processed by the computer directly unless they are transformed into numerical 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 color system the value of each pixel (usually between

0 and 255) is an index to a special table, which is called the colormap look-up table. The entry of the colormap table, corresponding to a particular index contains three values and these values are infact the intensities of three major colors, namely red,green, and blue. In a gray scale device, same idea of referencing a table could be used or the value of a pixel could be treated as an intensity. 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. Until relatively recently, image digitizing was so expensive that only a few computing centers had such a capability. Advances in technology, however, are making image digitizers less expensive and their use more spread. An important and highly versatile type of image digitizer is the so-called digitizing camera, which has a lens system and can digitize an image of any object. An example is a television camera interfaced to a computer.

After an image is represented in numerical 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 value of output pixels. 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 neighbor 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 arrangement 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 [1, 5].

After an image is processed and a desired image obtained, it is now necessary to display it. There are two basic types of display devices, permanent and volatile. Permanent displays produce a hard copy image on paper, film, or other permanent recording medium. Volatile displays produce a temporary image on a display screen and these are used commonly in interactive processing. The primary characteristics that determine the quality of a digital image display system are the size, photometric and spatial resolution, low-frequency response, and noise characteristics of the display. The size of a display system has two components. The first of these is the physical size and the second is the size of the largest digital image that the display can handle. Clearly these components should be sufficient enough for an image to be examined when it is displayed and for an image to be displayed in its complete form.

Photometric resolution means the accuracy with which the system produces correct brightness or density value in each pixel position. The capacity of the number of bits per pixel that the display can handle varies from one display to another, for example a display system that can handle 8-bit of data can be used to display 256 different intensity values [9].

For the explanation of the other characteristics, please refer to [9].

Today the display devices and the supporting hardware have advanced features as compared to the old type of display devices. Now it is possible to store any kind of an image obtained through a CCD (charged coupled device) like a camera in a special memory called frame buffer or frame store in a very short time. For an image to be represented without losing the spatial resolution it is necessary to have a display device that has a resolution proportional to the number of different intensities in a picture. Each pixel can be represented by a definite number of bits, usually 1 byte, to represent a definite number of light intensities or color indices. In image processing systems the display devices are usually raster scan display systems so maybe to give a short information about these devices and the terminology associated with them is necessary.

2.3 Basic Terminology of Raster Scan Display Systems

The operation of raster scan systems to draw an image on the screen is different from other systems like random scan systems. The main difference of these systems is that, the storage is used to keep the intensity information for each screen position instead of storing graphics commands in random scan systems. For raster scan systems the refresh storage is usually called a frame buffer or a refresh buffer. Each position in the frame buffer is called a picture element or pixel. Pictures on the screen are painted from frame buffer one line at a time from top to bottom. Each horizontal line is called a scan line.

Pixel positions in the frame buffer are organized as a two- dimensional array of intensity values or references, which correspond to coordinate screen positions. The number of pixel positions in the frame or display buffer is called the resolution of the frame buffer. For an acceptable image to be displayed on the screen the resolution should be about 1024 by 1024, as suggested earlier.

In a simple black and white graphics and image processing system each

screen point is either on or off, that is we need to store only 1-bit of information for each screen position or pixel in the frame buffer. For the more enhanced gray scale systems up to 24-bit information can be stored to represent many different intensities [1]. In colored display devices many combinations of three main colors (red,green and blue) can be displayed.

3. IMAGE PROCESSING ALGORITHMS

3.1 Classification of the Algorithms

Image processing algorithms can be classified in many ways. Here the classification criteria is the way of changing the pixels of a given image. As mentioned in section 2.2, if an algorithm changes a pixel's value depending only on its value, it is called a *point process*. If the algorithm changes a pixel's value based on its value and the values of the neighboring pixels it is called an *area process*. If the algorithm changes the position or the arrangement of the pixels it is called a *geometric process*. Algorithms that change pixel values based on comparing two or more images are called *frame processes* because a stored image is also called a frame [5].

In the following sections some image processing algorithms in the above classes are described and the results of applying them to three images given in figure 3.1, 3.2, and 3.3 are shown.

3.2 Point Processes

The available point processes or operations in the system are, contrast and intensity enhancement, histogram, threshold, and dither, each of which is described and related resulting images are given below.

3.2.1 Contrast and Intensity Enhancement

This operation is useful for contrast and intensity adjustment which makes the original image sharper. To increase the intensity of some pixels their old

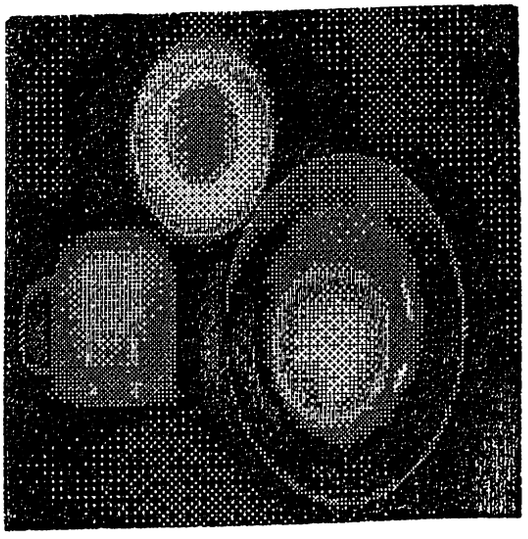

Figure 3.1: The "ORANGE"

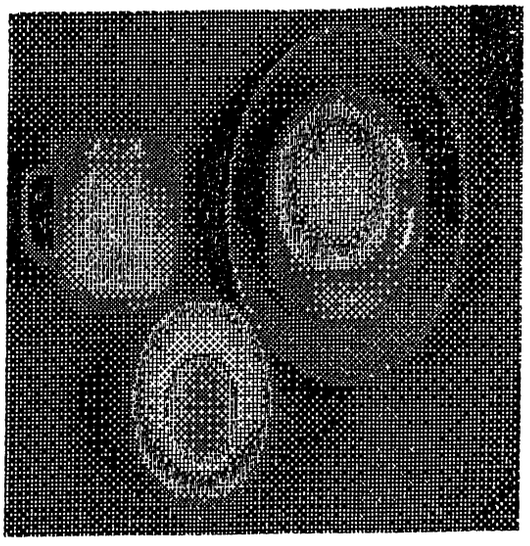

Figure 3.2: Mirror image of the "ORANGE"

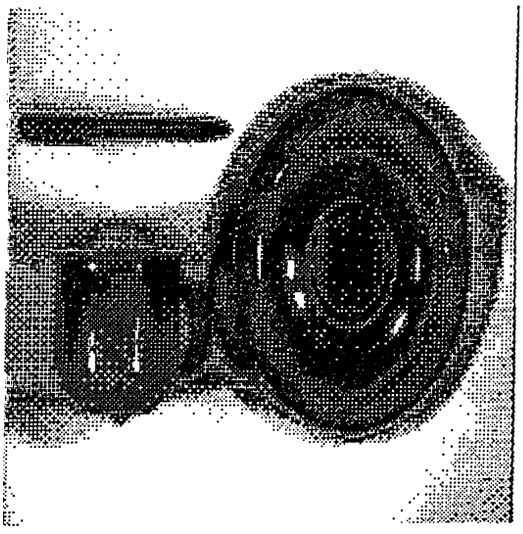

Figure 3.3: The "CUP"

values are increased by a constant value as in 1.

$$new\_value(x) = old\_value(x) + constant \tag{1}$$

As it is clear in this equation the selected pixels will be brighter or darker depending on the value of the constant added. To change the contrast of an image the only necessary thing to do is to multiply the values of the selected pixels by a constant. If we want both contrast and intensity adjustment at the same time we can use the formula 2.

$$new\_value(x) = constant_1 * old\_value(x) + constant_2 \tag{2}$$

3.2.2 Histogram

The histogram is a method of measuring an image. It is a different point process due to the fact that it does not change the value of the pixels.

The information provided by this operation is useful for image enhancement. The histogram of the picture given in figure 3.4 for example can be used to detect the distribution of the pixel values in the image. The C code of the histogram algorithm is shown in figure 3.5. The function it performs is to count the number of times each intensity level occurs. The intensity level changes from 0 to 255.

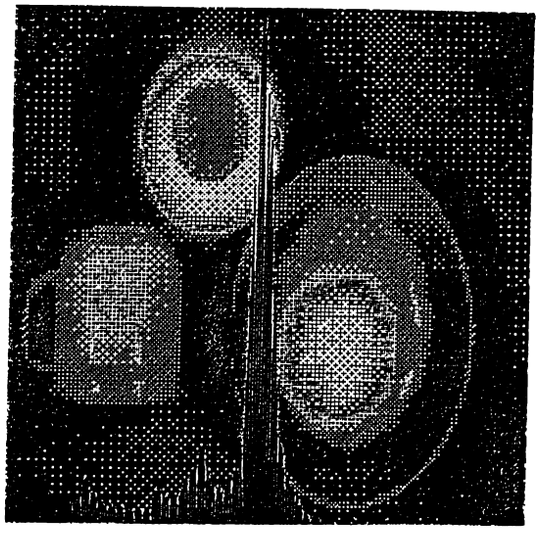

Figure 3.4: The histogram of the "ORANGE"

```
/* C partial code for histogram algorithm */
{
       int i, j, width, height, value;

       if (! pr_input) {
          printf("\n empty pixrect");
          return;
       }

       for (i=0; i < 255; i++) {
           histogram_array[i]=0;
       }

       width=pr_input->pr_size.x;
       height=pr_input->pr_size.y;
       for (i=0; i < height; i++) {
           for (j=0; j < width; j++) {
               value=(int)pr_get(pr_input, j, i);
               histogram_array[value]++;
           }
      }
```

Figure 3.5: C code for the histogram algorithm

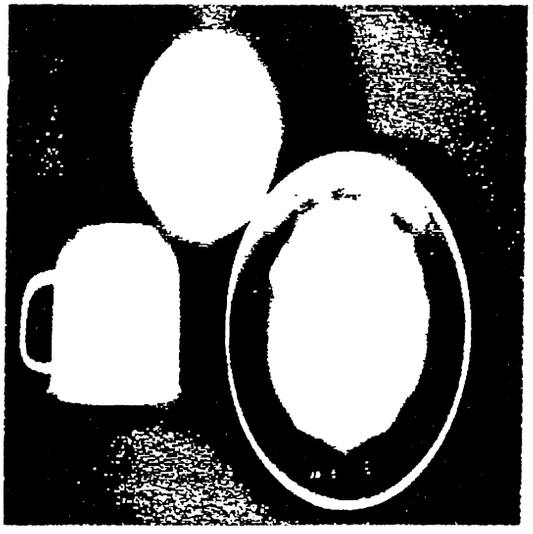

Figure 3.6: The threshold of the "ORANGE" with a threshold value 60

3.2.3 Threshold

Threshold is a global operation since it scans the values of all pixels. The pixel values are either changed to the lowest intensity (black) or to the highest intensity (white) depending on the value of a given constant and the value of the pixel. The constant used to compare to the values of the pixels are known as the threshold value which can range from 0 to 255. When the value of a pixel is less than the threshold value it is changed to 0 otherwise it is changed to 255.

The result of applying this operation is shown in figure 3.6 and the algorithm is presented in figure 3.7.

The threshold value is usually chosen to be the mid value of the intensities. The algorithm aims at increasing the visual resolution but also causes the fine details to be lost due to the large relative errors in displayed intensity for each pixel(e.g. if threshold value is 128 then there is no difference between the values 0 and 127 since both values will be changed to 0 in the final display). A technique developed by Floyd and Steinberg [2] distributes the large relative error to surrounding pixels to improve the details.

3.2.4 Dither

Dithering algorithm is a technique to increase the visual resolution without reducing the spatial resolution. This technique is used to display an image

```
/* C partial code for the threshold algorithm */
{
       int i,j, width, height, value, result;

       if (! pr_input) {
          printf("\n empty pixrect");
          return;
       }

       width=pr_input->pr_size.x;
       height=pr_input->pr_size.y;
       for (i=0; i < height; i++) {
           for (j=0; j < width; j++) {
               value=(int)pr_get(pr_input, j, i);
               if(value > threshold_value)
                 pr_put(pr_input, j, i, FOREGROUND_COLOR);
               else
                 pr_put(pr_input, j, i, BACKGROUND_COLOR);
           }
        }
}
```

Figure 3.7: C partial code of the threshold algorithm

on a bilevel device and attempts to introduce a random error into the image. This error is added to the image intensity of each pixel before comparing with the selected threshold value. Adding a completely random error does not yield an optimum result. However, an optimum additive error pattern that minimizes pattern texture effects exists. The smallest dither pattern or matrix is 2 by 2. An optimum 2 by 2 matrix is

$$D_2 = \begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix} \tag{3}$$

Large dither matrices are obtained by using the relation

$$D_n = \begin{pmatrix} 4D_{n/2} & 4D_{n/2} + 2U_{n/2} \\ 4D_{n/2} + 3U_{n/2} & 4D_{n/2} + U_{n/2} \end{pmatrix} \tag{4}$$

where n is the order of the dither matrix and it should be equal to integer powers of 2 starting from 2 and U is the matrix of sizes $n/2$ by $n/2$ whose elements are all equal to one.

The algorithm compares the value of a pixel say in position x, y with the value of the element in position i,j where

$i = x \ mod \ n, j = y \ mod \ n$ (n is the order of the matrix)

and if the value of the pixel in position x,y is equal to p(x,y) and it is greater or equal to the value of D(i,j) then the output pixel's value becomes 1 otherwise its value is zero. Figure 3.8 shows a picture obtained by using the dither algorithm, and the C partial code for the algorithm is given in figure 3.9.

3.3 Area Processes

An area process uses neighborhood information to change the values of pixels. They are typically used for spatial filtering (such as filtering out repeated elements) and changing an image's structure. They can sharpen the the image's appearance besides removing noise, blurring or smoothing the image.

The available area processes in the system are convolution and median, which are described in the following sections.

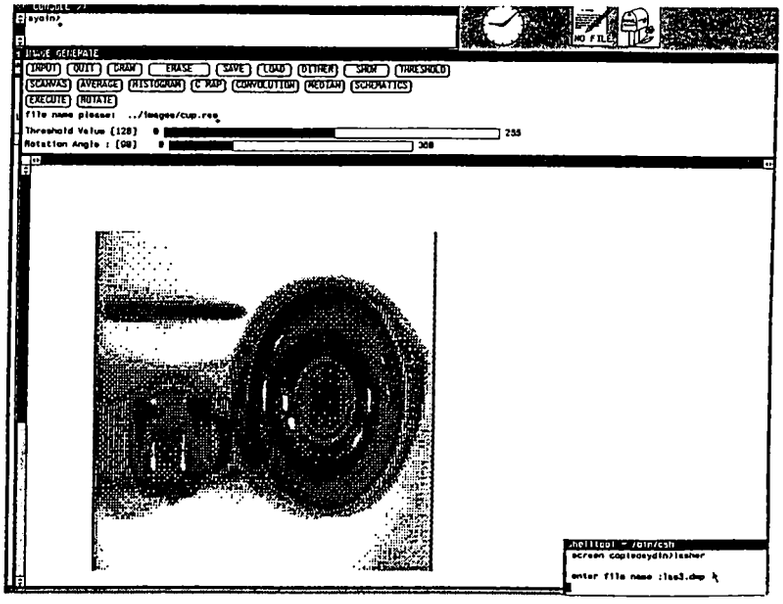

Figure 3.8: The dithered picture of the "CUP"

3.3.1 Convolution

Convolution is a classical image processing algorithm and it is used commonly for spatial filtering and finding the features of an image. The basic convolution operation is the replacement of a pixel's value with the sum of that pixel's value and its neighbors by weighting each value by a factor. The weighting factors are called convolution kernel, every pixel value is replaced by the value p(x,y) where

$$p(x,y) = \Sigma_{m=0}^{M}\Sigma_{n=0}^{N}k(m,n) * p(x+m, y+n) \tag{5}$$

M and N are the sizes of the convolution kernel matrix. By using different kernels it is possible to amplify vertical, horizontal or all edges in an image [5]. Figures 3.10, 3.11 , and 3.12 show how this can be achieved.

The convolution kernels below are used to amplify the vertical, horizontal and all edges of an image respectively.

$$K_V = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{6}$$

```
/* C partial code for the dither algorithm */
{
      int i, j, k, l, value, width, height, result;

      if (! pr_input) {
         printf("\n empty pixrect");
         return;
      }

      width=pr_input->pr_size.x;
      height=pr_input->pr_size.y;
      colormap[0][FOREGROUND_COLOR] = (u_char)0;
      colormap[1][FOREGROUND_COLOR] = (u_char)0;
      colormap[2][FOREGROUND_COLOR] = (u_char)0;
      colormap[0][BACKGROUND_COLOR] = (u_char)255;
      colormap[1][BACKGROUND_COLOR] = (u_char)255;
      colormap[2][BACKGROUND_COLOR] = (u_char)255;

      for (i=0; i < height; i++) {
          k=i % 16;
          for (j=0; j < width; j++) {
              l=j % 16;
              value=(short)pr_get(curr_dither->pr_input, j, i);
              if(value >= d[k][l])

                  pr_put(curr_dither->pr_input, j, i,
                         FOREGROUND_COLOR);
              else
                pr_put(curr_dither->pr_input, j, i,
                        BACKGROUND_COLOR);
           }
      }
}
```

Figure 3.9: C partial code for the dither algorithm

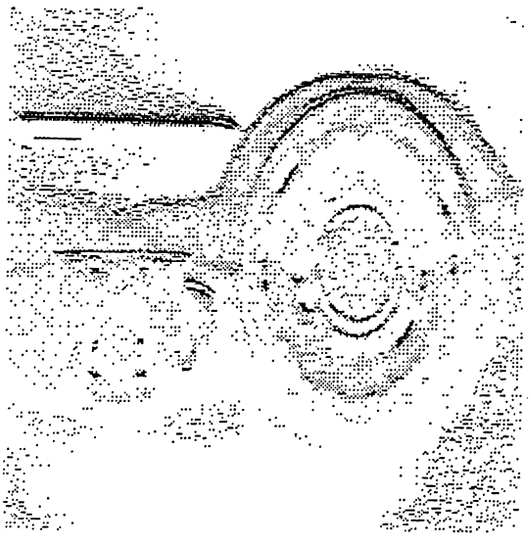

Figure 3.10: Horizontal lines of the "CUP" have been amplified

$$K_H = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{7}$$

$$K_A = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{8}$$

If we slightly modify the last kernel, by making the value in the center 9 instead of 8 the resultant image obtained is the same as if we added the original image to the image in figure 3.12 and the resulting image becomes sharper as shown in figure 3.13.

The partial code of this algorithm is given in figure 3.14.

3.3.2 Median

Median filtering replaces the pixel at the center of a neighborhood of pixels by the median of pixel values. The neighborhood pixel values are sorted in ascending order and the median(middle) value is used to replace the center pixel. This algorithm for which the code is in figure 3.17 and in figure 3.18,

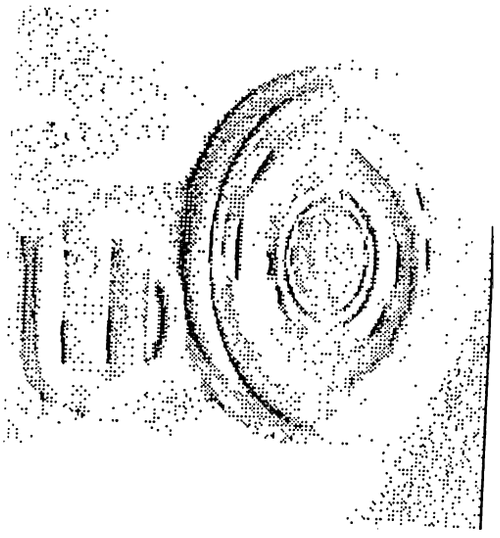

Figure 3.11: Vertical lines of the "CUP" have been amplified

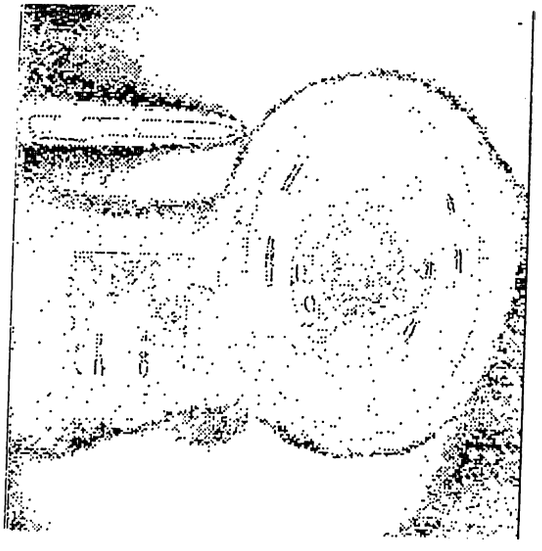

Figure 3.12: All lines of the "CUP" have been amplified

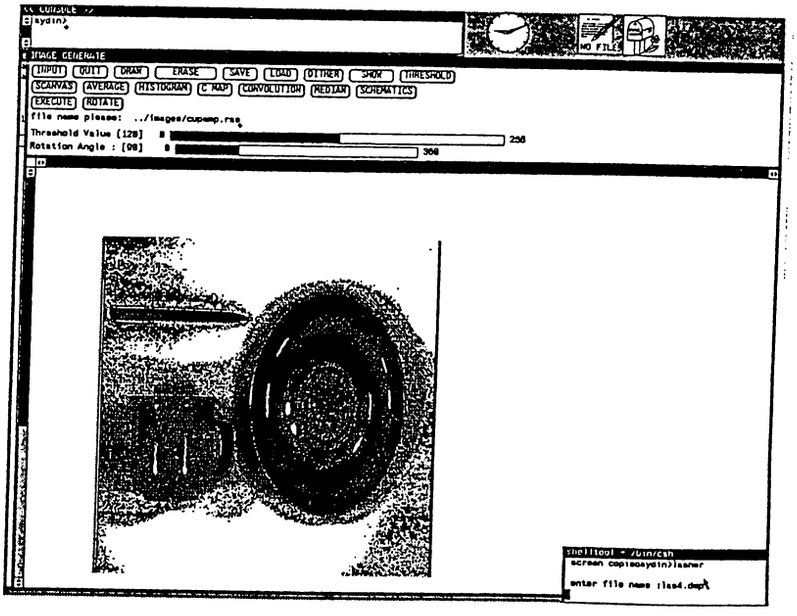

Figure 3.13: All lines of the "CUP" have been amplified and added to the original image

is used to remove the salt and pepper noise from an image (Figures 3.15 and 3.16) [5].

3.4 Geometric processes

The algorithms or processes obtaining a new image by changing the pixel positions of an original image are called *Geometric Processes.* Two such processes available in the system are rotation by an arbitrary angle and average.

3.4.1 Rotation

This algorithm is used to rotate an image with respect to its center by an angle between 1 and 360 degrees. The result of rotating the image in figure 3.1 by 90 degrees is given in figure 3.19.

```
/* C partial code of the convolution */
       int row_num, column_num, *convol_mat;
{
       int i, j, k, l, width, height, value,
           rel_x, rel_y, row, column, col1;
       Pixrect *pix;

       if (! pr) return;
       width=pr->pr_size.x;
       height=pr->pr_size.y;
       pix=mem_create(width, height, pr->pr_depth);

       rel_y=(int)(row_num/2.0);
       rel_x=(int)(column_num/2.0);

       for (i=rel_y; i < height-rel_y; i++) {
            for (j=rel_x; j < width-rel_x; j++) {
              value=0;
              row= -1;
              for (k=i-rel_y; k <= i+rel_y; k++) {
                row++; column= -1;
                for (l=j-rel_x; l <= j+rel_x; l++) {
                    column++;
                    value=(int)(value+convol_mat[row][column] *
                                    (int)pr_get(pr, l, k));
                 }
               }
               if (value > FOREGROUND_COLOR) {
                  value=FOREGROUND_COLOR;
               }
               pr_put(pix, j,i, value);
            }
       }
}
```

Figure 3.14: C partial code for the convolution algorithm

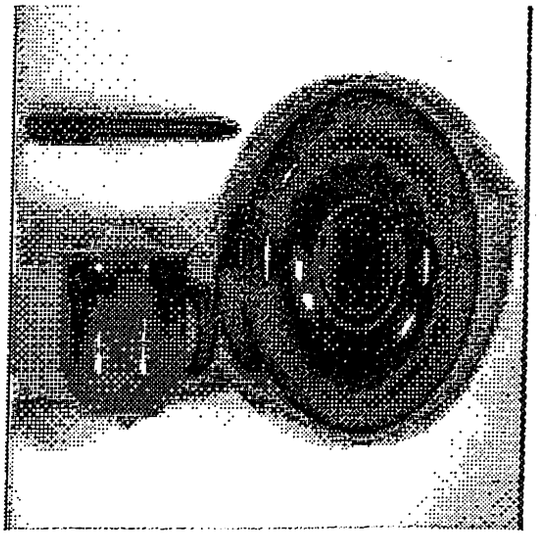

Figure 3.15: The dithered picture of the "CUP" will be filtered by median filter with a 3 by 3 window

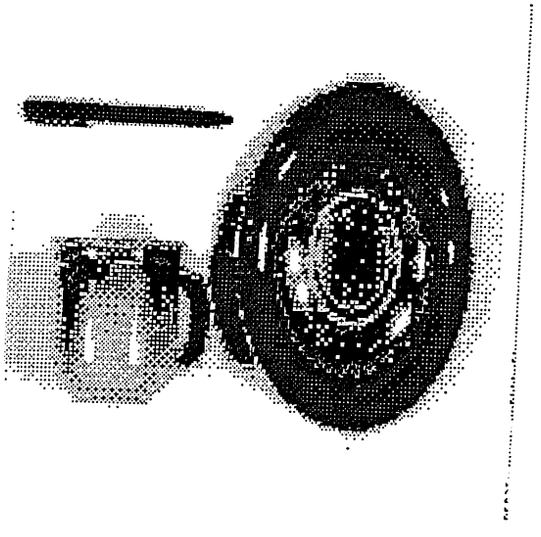

Figure 3.16: The result of the median filter

```
/* C partial code for the median filtering */
       int row_num, column_num;
{

     int i, j, k, l, width, height, value, rel_x, rel_y,
         row, column, el_num, value;
     int *median_array, *temp;
     Pixrect *pix;

     if (! pr) return;
     width=pr->pr_size.x;
     height=pr->pr_size.y;
     pix= mem_create(width, height, pr->pr_depth);

     median_array = (int *)
           malloc(sizeof(int)*row_num*column_num);
     if (! median_array) {
        printf("\n error in array median !");
        return;
     }
     rel_y=(int)(row_num/2.0);
     rel_x=(int)(column_num/2.0);

     for (i=0; i < height; i++) {
        for (j=0; j < width; j++) {
          el_num=0;
          temp=median_array;
          for (k=i-rel_y; k <= i+rel_y; k++) {
           if (k >= 0 && k <= height) {
             for (l=j-rel_x; l <= j+rel_x; l++) {
                if (l >= 0 && l <= width) {
                   el_num++;
                   *temp=(int)pr_get(pr, l, k);
                   temp++;
                }
             }
           }
          }
          quicksort(median_array, el_num);
          value = (int)(*(median_array+(int)(el_num/2.0))));
          pr_put(pix, j, i, value);
        }
     }
}
```

Figure 3.17: C partial code of the median filter

```
quicksort(a, n)
int *a, n;
{
    int k, pivot;

    if (find_pivot(a, n, &pivot) != 0 ) {
        k=partition(a, n, pivot);
        quicksort(a, k);
        quicksort(a+k, n-k);
     }
}

find_pivot(a, n, pivot_ptr)
int *a, n, *pivot_ptr;
{
    int i;

    for (i=1; i < n; i++)
        if (*a != *(a+i)) {
           *pivot_ptr = (*a > *(a+i)) ? *a : *(a+i);
           return(1);
         }
     return(0);
}

partition(a, n, pivot)
int *a, n, pivot;
{
    int i=0, j=n-1, temp;

    while (i <= j) {
          while (*(a+i) < pivot)
                ++i;
          while (*(a+j) >= pivot)
                --j;
          if (i < j) {
                temp= *(a+i);
                *(a+i)= *(a+j);
                *(a+j)=temp;
                ++i;
                --j;
          }
    }
    return(i);
}
```

Figure 3.18: C partial code for the quick sort algorithm

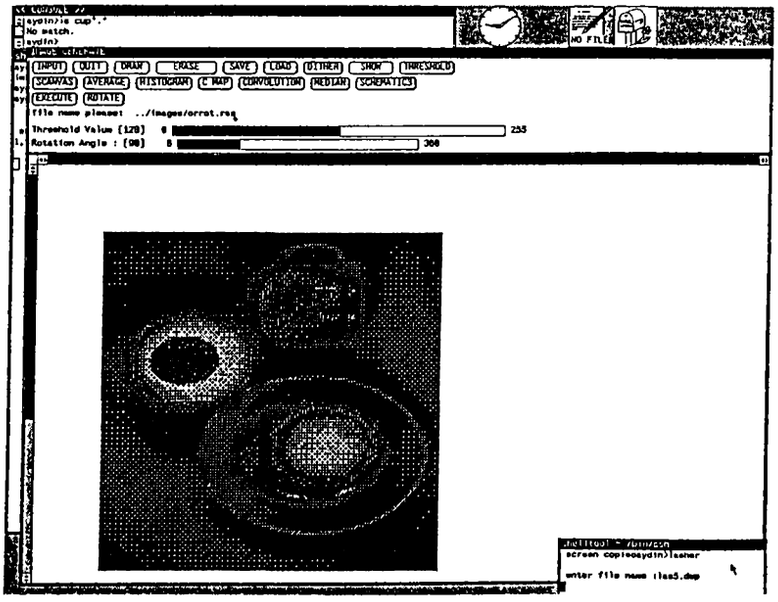

Figure 3.19: Rotation by 90 degrees

3.4.2 Average

Averaging is an algorithm that can be used for both enlarging and shrinking images. Shrinking average takes the pixels of an image in groups of four (2 horizontal and 2 vertical pixels) and produce an output pixel whose value is equal to the average of values of these four pixels (Figure 3.21).

Enlarging average takes a pixel and copies its value to its four neighbors. As a result the original image is enlarged by a factor of four (Figures 3.22, 3.23).

The algorithm in figure 3.20 can be used for both enlarging and shrinking a given image.

Average algorithm is used when a user wants the fine details of an image to disappear or when an image is wished to be displayed in the form of blocks besides changing the size of an image. This can be achieved by applying shrinking algorithm and then enlarging the image continuously.

```
/* C partial code of the average algorithm */
{
     int i,j, x, y, value, width, height;
     Pixrect *pix;


     width=pr->pr_size.x;
     height=pr->pr_size.y;

     printf("\n width=%d height=%d",width, height);
     pix=(Pixrect *)0;

     y= -1;
     if (av_mode == -1) {
        pix=mem_create((int)width/2, (int)height/2,
                       pr->pr_depth);
        for (i=0; i < height; i+= 2) {
            y++; x= -1;
            for (j=0; j < width; j+= 2) {
                x++;
                value=(short)(pr_get(pr,j,i)+pr_get(pr,j+1,i)+
                      pr_get(pr,j,i+1)+pr_get(pr,j+1,i+1))/4;
                pr_put(pix, x,y, value);
            }
        }
    }
    else if (av_mode == 1) {
        pix=mem_create((int)width*2, (int)height*2,
                       pr->pr_depth);
        for (i=0; i < height; i++) {
            y = 2 * i;
            for (j=0; j < width; j++) {
                x = 2 * j;
                value = (short)pr_get(pr, j,i);
                pr_put(pix, x, y, value);
                pr_put(pix, x+1,y, value);
                pr_put(pix, x, y+1, value);
                pr_put(pix, x+1, y+1, value);
            }
        }
    }

}
```

Figure 3.20: C partial code for the average algorithm

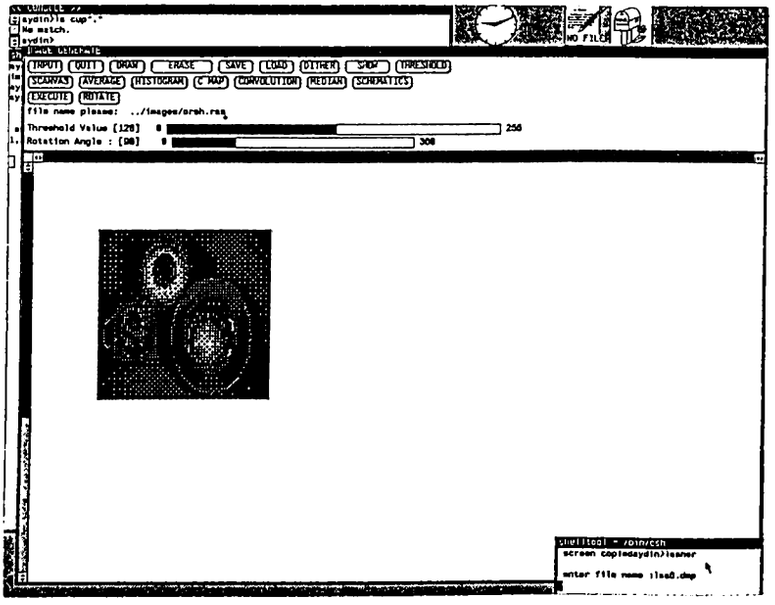

Figure 3.21: The "ORANGE" has been shrunk

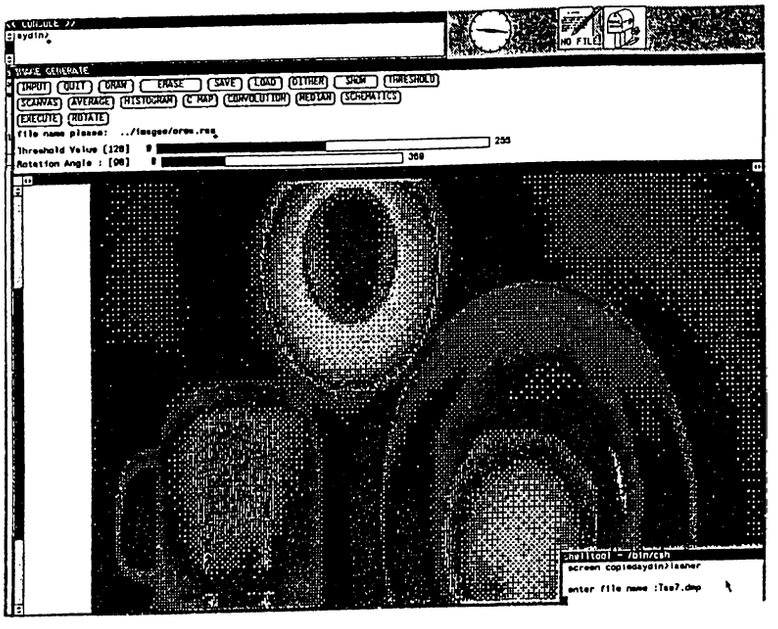

Figure 3.22: The "ORANGE" has been enlarged

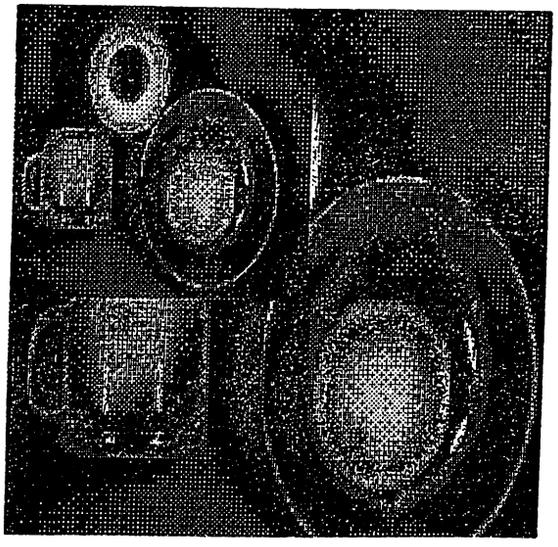

Figure 3.23: The "ORANGE" has been enlarged (another view)

3.5 Frame Processes

Frame processing algorithms are used to obtain a new image by using the pixel values of two or more images. Common algorithms in this class are difference that can be used to find out missing parts between two images. Addition of images is used to enhance an image or to obtain an image whose pixel values are the average of the pixel values of the processed images. AND operation is used to mask some parts of an image by using a mask image and OR operation combines two or more images into one [5] (Figures 3.24, 3.25, 3.26, 3.27, 3.28).

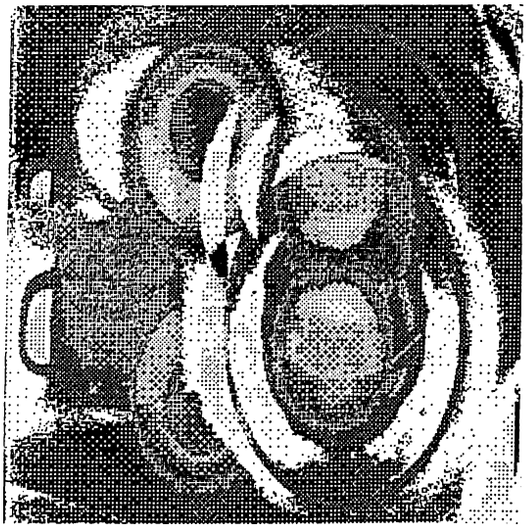

Figure 3.24: The "ORANGE" AND its mirror image

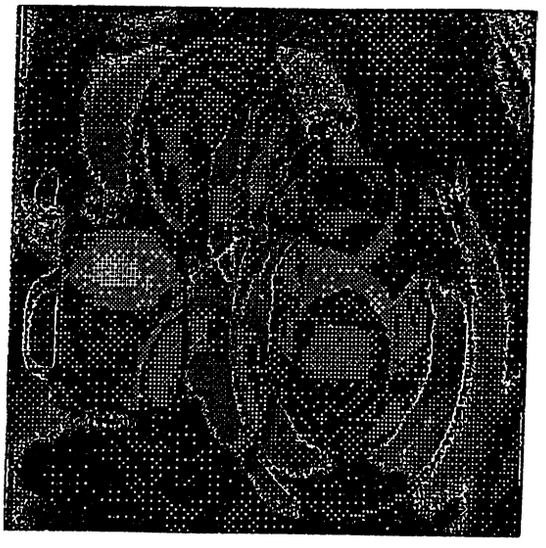

Figure 3.25: OR of the images

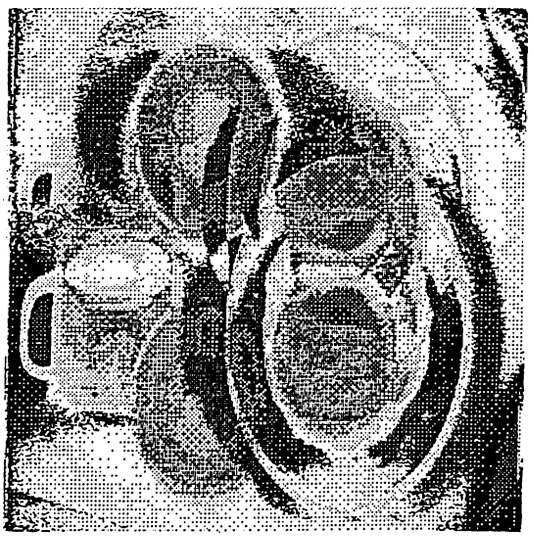

Figure 3.26: XOR of the images

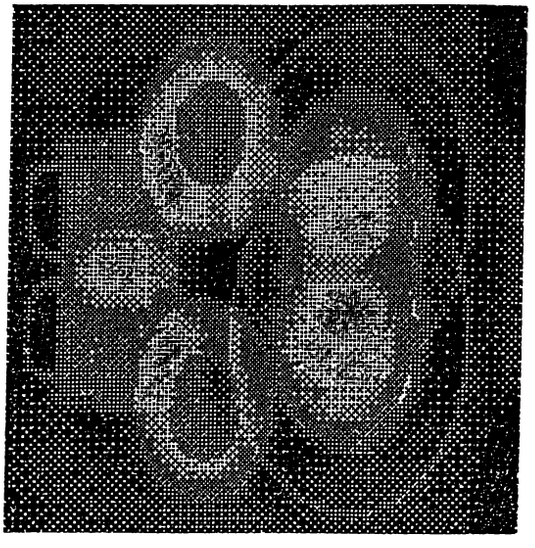

Figure 3.27: Addition of the images

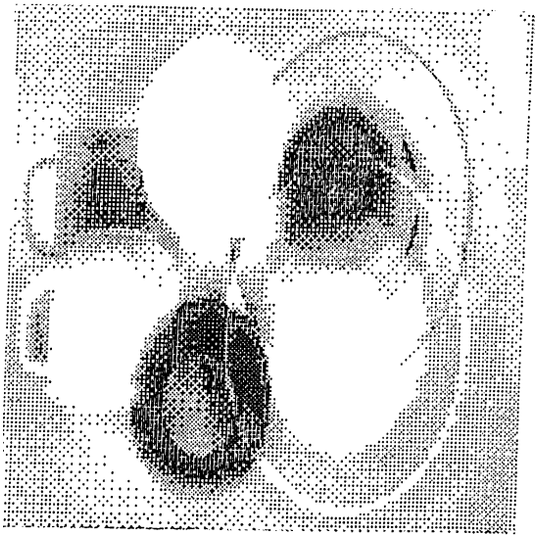

Figure 3.28: Difference of the images

4. IMPLEMENTATION OF THE SYSTEM

4.1 User Interface Design

When the performance of the user interface of an application (i.e the part of the system that determines how the user and the computer communicate) cannot be predicted it is very likely that users will react in unexpected ways when they use the system for the first time.

It is very important to pay careful attention to the design of interactive user interfaces. Bad user interfaces are not only difficult to learn but they also make a system inefficient to operate even in the hands of an experienced user. In extreme cases an entire system may be invalidated by poor user interface design, that is, it may prove impossible to train users to operate the system, or the user interface may be so inefficient and unreliable that the cost of using the system cannot be justified [4, 6].

Since our aim is to implement an image processing system through the use of functional blocks, and a functional block means the representation of a routine or function by a visual object in our intent, writing a user interface in old fashion (in the form of command languages) cannot be justified. In the traditional user interfaces command languages were heavily used to provide communication between the user and the computer. In this kind of user interface the user has to know the exact syntax of the available commands. For example, when someone wishes an image to be saved on the disk he or she has to give a command like

save(image,file-name)

where image is a pointer to a particular section of memory where the storage of the the image begins or the name of an array storing the image, and file-name is a string denoting the name of the file to be saved.

Even in this simple example there are many complexities for a novice user. First he has to remember the exact syntax of the command,second he should know about some other commands like accessing the value of a pointer, opening and closing a file.

An alternative to the above method is to represent a save function by an icon and when it is selected through a pointing device like a mouse, display the listing of images that the user can save, and then ask a name under which the image will be saved. The use of such methods leads us to some new concepts in software sciences, most important of which are the object oriented style or discipline of programming, iconic interface and use of multiple windows. Although our intention in this thesis is not to talk about these concepts in detail it is better to explain the user interface that we plan to employ by considering the ideas behind these new and important software techniques.

4.2 Object Oriented User Interface

From the previous section it can be assumed that a user interface in the form of a command language cannot be justified in a work like ours. Therefore we planned a user interface that is very user friendly and based on a toolkit, namely on the Sun View^{TM} presenting menus, windows, panels, and many kinds of items(sliders, buttons, choices, etc.) ready for special usage as components of the user interface. The components in a toolkit are usually used by a pointing device like a mouse as long as there is no need to enter text. 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 [10].

As previously stated the building blocks of the user interface are the visual representations of the various objects in the form of icons, data template windows, control panel items like buttons, sliders, choices etc. It is very important to note that the term object refers to both an entity in the real life,like a picture or a chair etc, and the term "object" in the "object oriented style of programming". The difference should be understood from the context.

The object classes of the functional blocks, the panel items and menu choices represent the available options to the user. The objects provided have their particular properties and the user tells the system what kind of

processing he wants by selecting a particular icon representing an object in a particular class and connecting those like a simple flowchart which we call a *schematics*. By pointing to the instances of various object classes in the schematics by the aid of the mouse, the properties of that object instance could be changed as explained later.

One of the most important kind of object classes (groups of object instantiations) used in our system are data template windows and they are used to set the arguments of various object instances (object variables) and to display an intermediate image in the process if the functional block happens to be a display device. Other visual object classes except the window are the class of menus, class of scrollbars and the functional blocks represented in the form of icons.

Icons are small pictures (usually they are composed of 64 by 64 pixels) and they represent available functions to be applied to the images, menus are used to select a particular choice for an object, and scrollbars are used to view different parts of an image both in vertical and horizontal directions [3].

Technically speaking an object is an entity presenting a functional interface. The implementation of how the objects perform their jobs are not exposed to the user, instead it is enough to choose a particular object and set the variables belonging to it. When an object is chosen it calls its associated functions in an indirect manner, that is there is not a main program that calls a function or routine in a proper order, instead all objects are active processes(in a multi- programming environment) and their associated functions or routines are called through user or system invoked events. The associated routines of an object may also be called by another object. When an object is to be invoked it is necessary to form a proper message and send it.

The objects and object classes form a hierarchy. In this sense the object is at the most upper level, below it we have object classes, namely menus, windows,scroll bars, and functional blocks, displayed to the user in the form of icons. Each class may contain many instantiations of objects for example a window may contain panels, text sub windows, drawing windows, and a panel may be further subdivided into many panel items.

4.3 Data Structures

The data structures of the image processing algorithms together with their describtion are given in figures 4.1 through 4.4. In these figures the data structures are presented by diagrams. The corresponding C structures are given in appendix C.

Currently there are 12 functional blocks representing one or more functions. Each of them is represented by a linked list that can be reached through the general linked list representing any object, namely the *object list.*

The linked list *object list* in figure 4.1 represents any kind of object. It contains information about the object it is representing(instance, and object id), position of particular object instance , and the links connected to a particular instance.

The diagrams in figure 4.2 represents the data structures for four of the 12 functional classes. These are average, histogram, convolution, and dither functional blocks. Each data structure for functional blocks contain common information for the instance number of the class, the colormap of an instance, a pointer to an image that is to be processed, and a pointer to the next instance in the same class.

Besides the common information kept in each functional block, there are some special attributes for almost all functional blocks in each class. The *opr* field of the average block keeps information so that either enlarging or shrinking average operation can be applied.

The histogram array of the histogram block is used to keep the count of pixels in each intensity between 0 and 255. The convolution block stores information related to the size of convolution kernel (row, and column number), and the entries of it. The dither block keeps information for storing the size and entries of the matrix to be used in the dither operation.

Figure 4.2 shows the diagrams of the data structures for threshold, median, AND/OR, and point operations blocks. The private information belonging to these blocks are explained below.

The threshold value in the data structure of the threshold block stores the the threshold that is to be applied in the threshold operation. Median block

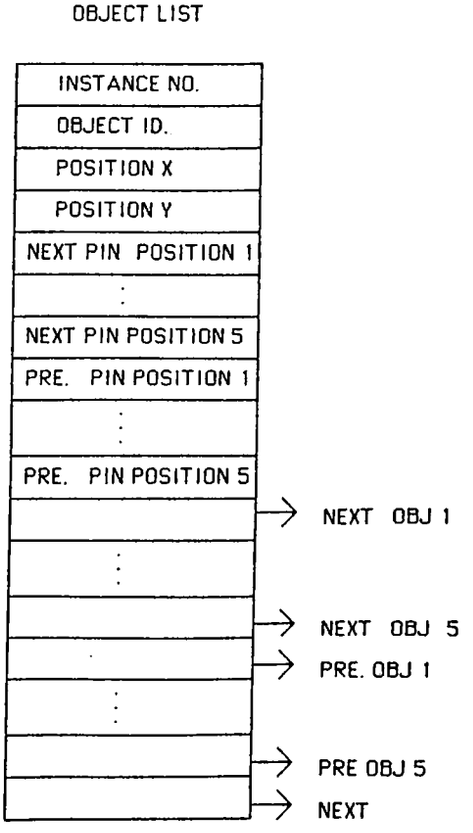

Figure 4.1: Data structure of the object list.

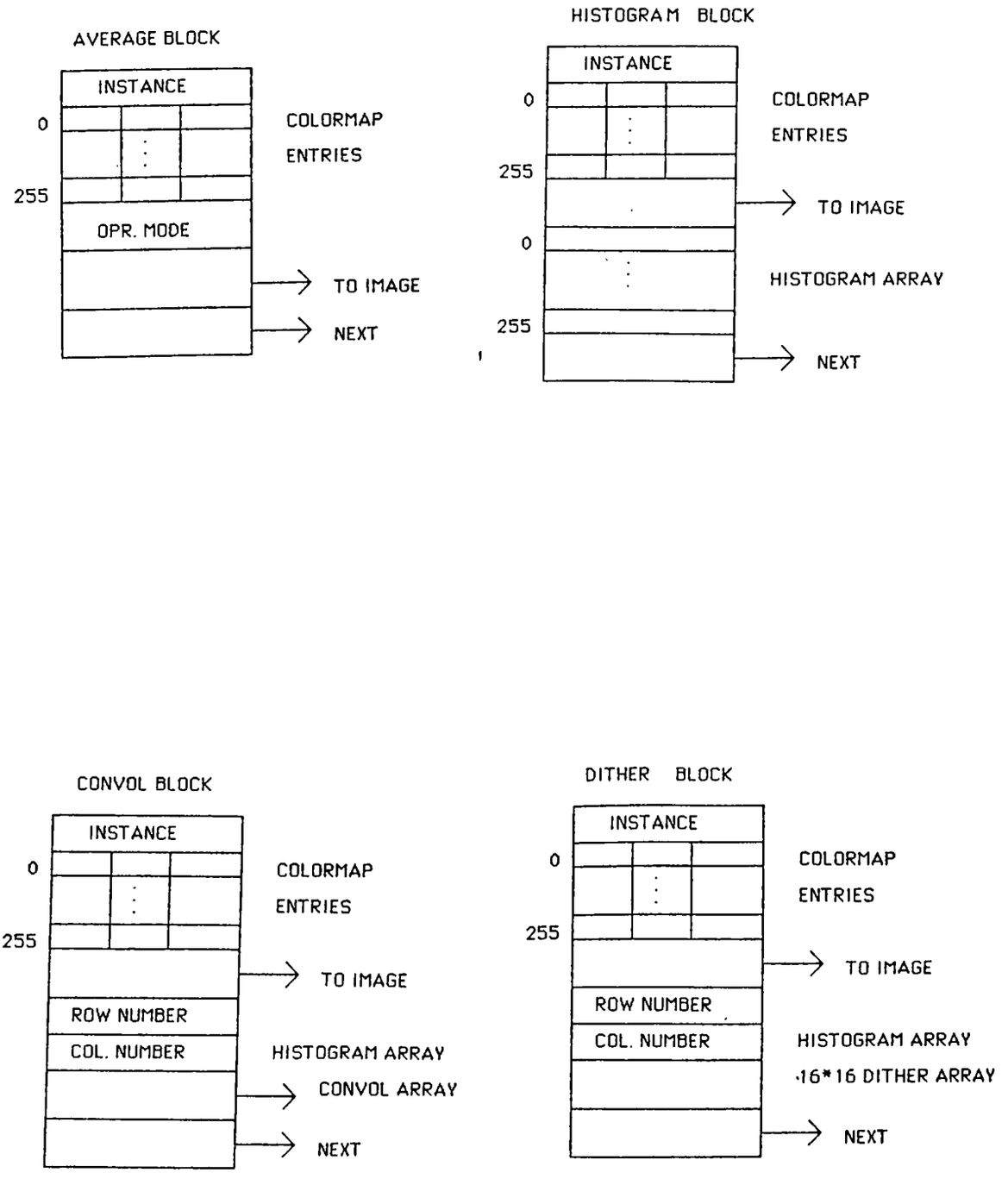

Figure 4.2: Data structures for average, histogram, convolution, and dither blocks

keeps the size of median filtering window in terms of row and column number. AND/OR block is used in the frame processing involving two images. Therefore it stores another pointer to an image and the operation code. This operation is one of AND, OR, XOR, PLUS, and MINUS operations. The private data for the point operations block contains the intensity, contrast, and a function pointer. The first two attributes are used in image enhancement. The function pointer points to a point operation function to be used for this particular instance.

The data structures for display, PLUS/MINUS, load, and save blocks are in figure 4.4.

As it is shown in figure 4.4 the display block does not contain any special information. The PLUS/MINUS block is designed for frame processing operations. Therefore it contains one more pointer to an image and the code for type of frame processing. The load and save functional blocks both contain the name of a disk file that an image is to be loaded from or an image is to be stored into.

In the system there are two modes or styles of interaction. These are named as *Direct mode* and *Indirect mode.* In both of the modes the user can use every function that he or she likes.

4.4 Direct Mode

This mode of operation is especially useful when the user does not want to construct a schematics for some trivial operations, that is any kind of operation that can be performed without doing anything else before. Loading an image or saving an image from or to disk are clearly trivial or simple operations in this sense.

In this mode directly applicable or trivial operations are invoked by just pressing a suitable panel button by the mouse. The user also has to supply a file name for some operations, such as save or load an image. The resulting image is displayed immediately after the operation is completed and it becomes the current image to be used in later operations until a new image

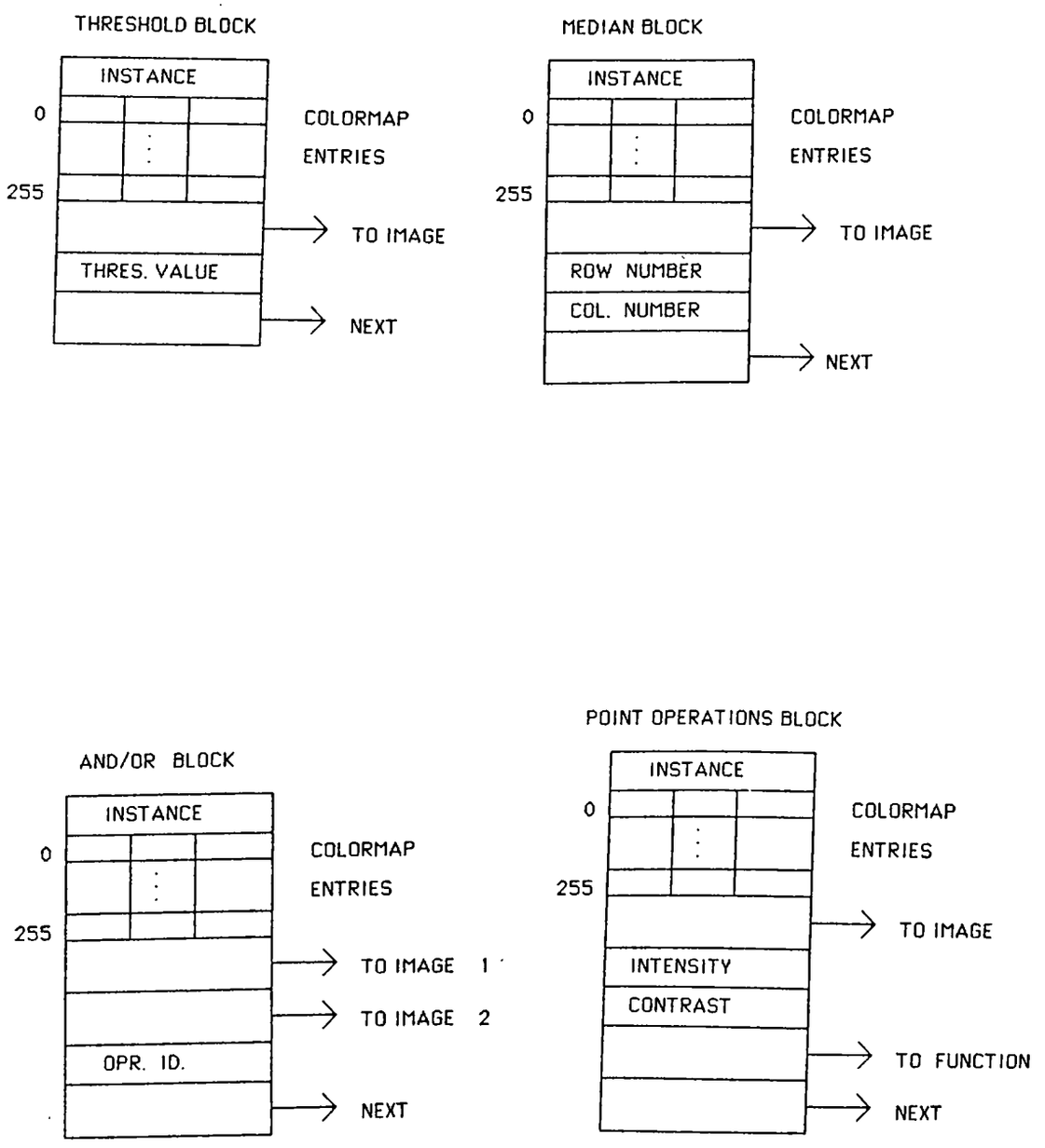

Figure 4.3: Data structures for threshold, median, AND/OR, and point operations blocks.

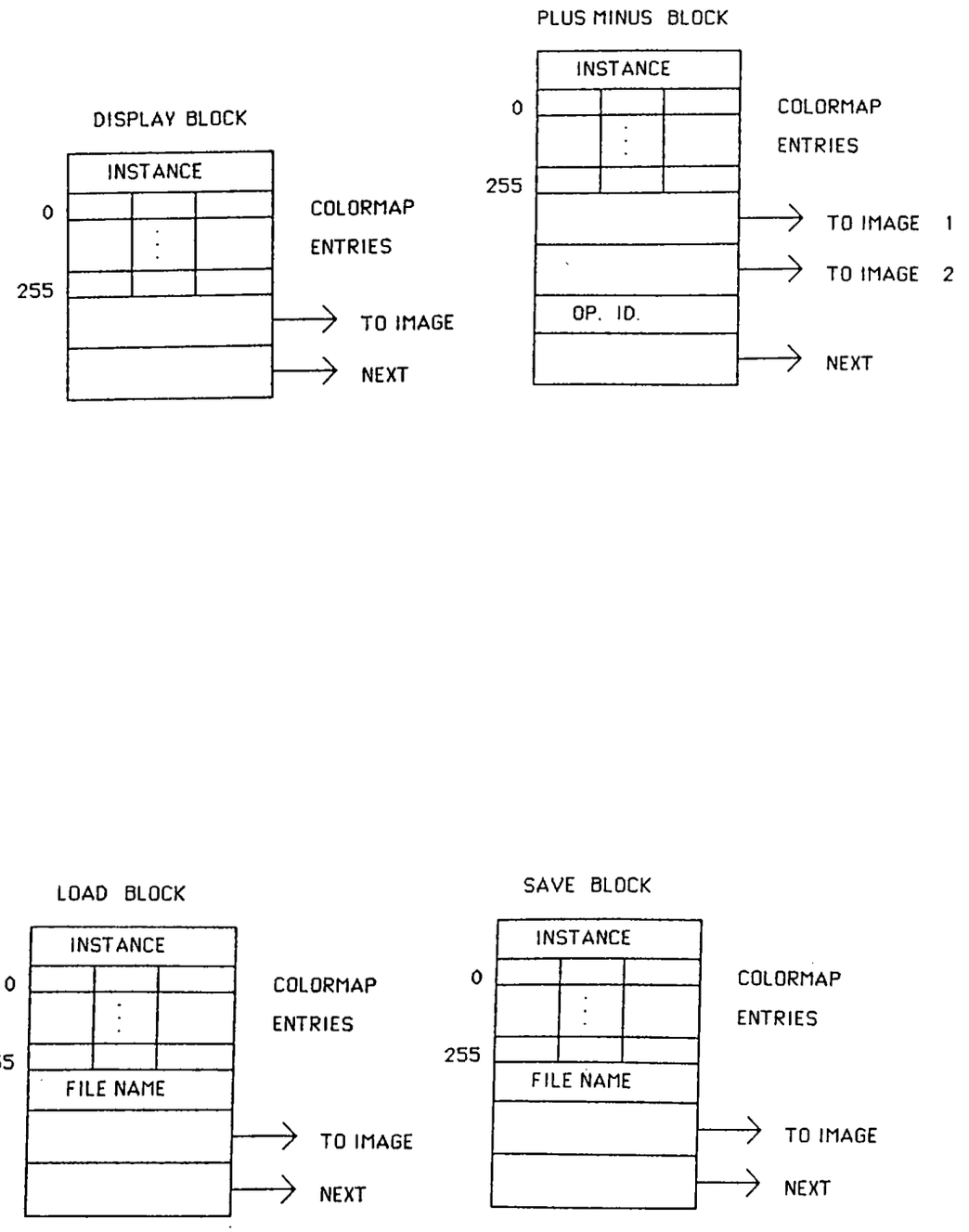

Figure 4.4: Data structures of the display, PLUS/MINUS, load, and save functional blocks.

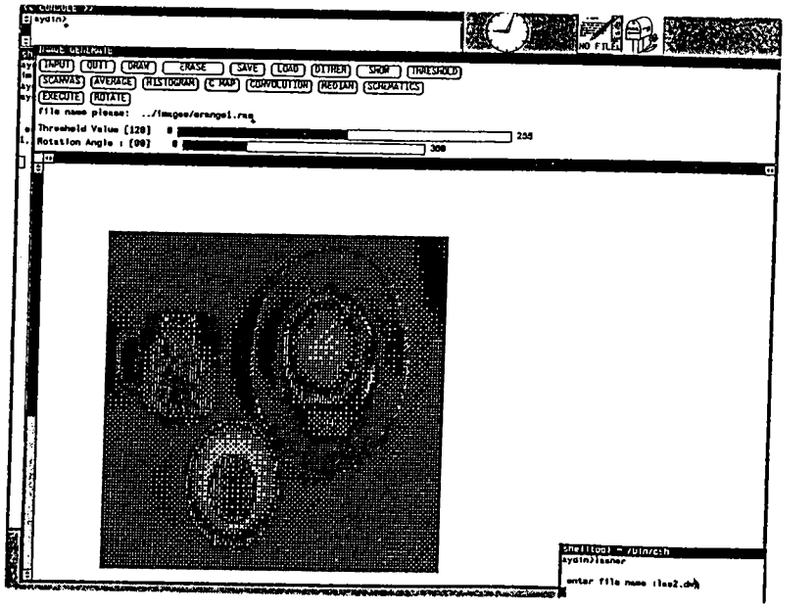

Figure 4.5: Layout of panel items

is loaded. As a result of this an image already processed in some way can be processed further.

4.4.1 Available Functions

When the system is started a window is opened through which some of the functions can be invoked directly. These functions are invoked by pressing a mouse button on one of the buttons shown in the top part of the screen (see figure 4.5).

As it is seen in figure 4.5 the available functions are as follows[1] :

- ERASE : It clears the canvas[2].
- SAVE : This is used to save an image which is on the screen or in the memory (current image) to disk in the raster file format. In raster file format every image has a 32 byte header to store the information about the length, width, depth (number of bits per pixel) of an image. After the header the related colormap is stored and it occupies 768 bytes. Finally the pixel values are stored in the file.

[1]The detailed descriptions of the functions are presented in Appendix A.
[2]Canvas is a drawing subwindow.

- LOAD : This function is used to load an image from the disk which has been stored in raster file format previously. Here one point to be stated is that loading or saving an image with sizes 512 by 512 is about two seconds.

- DITHER: The image which was loaded before or the image in the memory will be dithered and it will be displayed on the screen. The obtained image becomes the current image and other processing functions can be applied to it.

- SHOW : This function is useful for displaying an image not in raster file format and to store it in the raster file format for compatibility.

- THRESHOLD : This function applies the threshold operation to the image and the resulting image becomes the current image.

- SCANVAS : This is a function to save the whole drawing window or the canvas in a format which is suitable for printing by using an ordinary printer.

- AVERAGE : If someone wishes to enlarge or shrink the current image this function can be selected.

- HISTOGRAM : When the processing of the function is completed the histogram of the image will be displayed on the screen.

- C MAP : This function is used to change the colormap of the displayed image and to paint some portions of it in a simple manner.

- CONVOLUTION : This function is used to covolve an image with a convolution kernel that is a two dimensional table.

- MEDIAN : This button is used to pass the current image through the median filter to make it blurred or to remove salt and pepper noise.

- SCHEMATICS : When this key is pressed the user can draw a schematics to process the images in the indirect mode.

  The construction of the schematics is described in detail in appendix A.

- EXECUTE : This button is used to execute a previously constructed schematics in the schematics canvas.

- ROTATE : In the direct mode ROTATE button is used to turn an image with respect to its center by a definite angle.

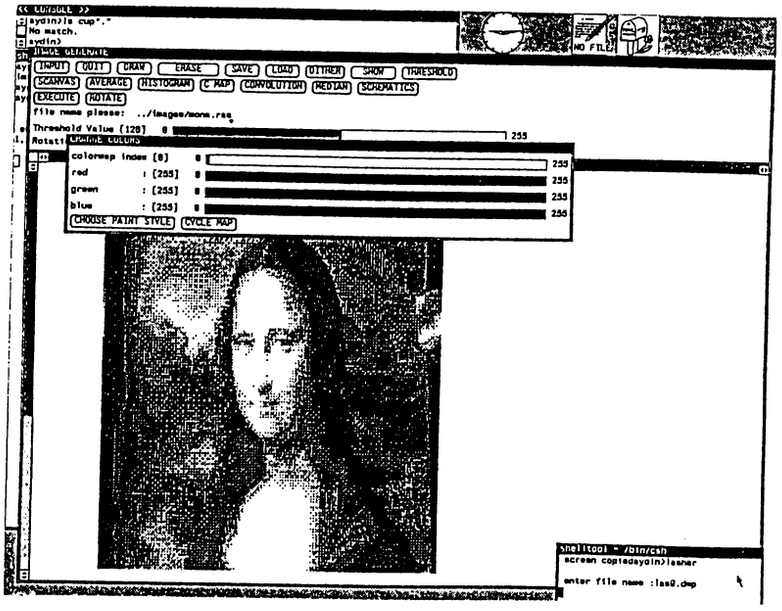

Figure 4.6: The colormap manipulation subwindow.

4.4.2 Colormap Manipulation

The window shown in figure 4.6 is opened as a result of pressing the C MAP button. It has four slider items and two buttons.

The "colormap index" slider is used to set the index of the colormap entries between 0 to 255. The index can also be set by pressing the middle mouse button in the canvas to see the color intensities of a particular pixel and its particular location in the colormap table.

The other sliders are used to change the intensities of three main colors, namely red, green, and blue from black (0) to white (255). By arranging these sliders a pixel value may have one of the over 16,000,000 colors. The affect of changing the values of color sliders is observed in real time. That is when a change of the color sliders is made the corresponding color value in the entry set by the colormap index will be changed immediately. As a result of this operations all the pixels of the image showing this entry will have another color at the same time.

The two buttons are used to paint the image on the canvas and to cycle the colormap entries respectively. The first button has a menu associated with it and is shown in figure 4.7. The menu items are POINTS, INJECTION, and BLOCKS that are used to put points, to make injection onto the image

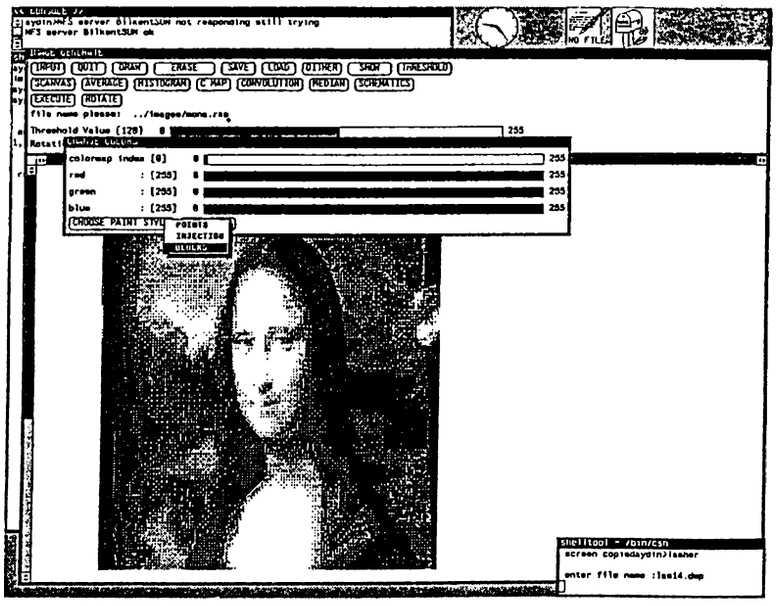

Figure 4.7: The menu associated with the CHOOSE PAINT STYLE key.

and to paint the image with blocks respectively. When the painting style is chosen from the menu the user can paint the image just by pressing the right mouse button and dragging it on the canvas. The color used in the painting process is the one in the colormap entry which is set by colormap index and of course one of the over 16,000,000 colors can be in that entry.

The CYCLE MAP button is used to cycle the colormap entries by a certain amount to obtain different views of the displayed image when the pixel values jump from one color to another. This technique is used in many art works, advertisement, and animation.

4.5 Indirect Mode

This mode of operation is used to process the images in continuous fashion and especially useful when the user wants to see the intermediate or final results of some predefined processing sequences.

This mode involves construction of a schematics on the schematics canvas that describes the processing steps and their execution order like in an algorithm or a flowchart.

4.5.1 Representation of the Functions

The 12 available functions in the indirect mode are represented as icons which are also the representations of the menu items. The menu containing these representative icons is associated with the CHOOSE BLOCKS button in the SCHEMATICS FRAME. Any menu associated with the panel buttons are always displayed by pressing the right mouse button on the related panel button.

The represented functions are as follows:

- AVERAGE : which is used to shrink or enlarge an image.
- HISTOGRAM
- CONVOL :Block representing the convolution operation.
- DITHER
- THRESHOLD
- MEDIAN
- AND/OR : This block represents the frame operations which are AND, OR, XOR, addition of two images, and difference of two images.
- POINT OP : This icon represent the intensity and contrast enhancement and it is possible to include more point operations in it.
- DISPLAY : This icon represent the display device. When another block is connected to it the stored image in that block will be transfered to the display block and be ready for displaying it.
- LOAD : It is used to load an image from the disk which should be in raster file format.
- SAVE : Any image obtained in one of the processing steps can be stored in the disk in the raster file format when this block is connected to the block which contains the desired image.

4.5.2 Operations on the Blocks

The functions that affect the placement, location, selection, or the settings of any block in the BLOCKS MENU are the items of the OPERATIONS MENU. These functions are shown in figure 4.8, and listed below[3].

- SELECT BLOCK TO REMOVE (F1) : After selecting this operation clicking the left mouse button or pressing the function key F1 when the cursor in on a block causes the block to be removed from the schematics.
- PUT BLOCK ON CANVAS (F2) : This operation is used to put a selected block on the drawing window or the canvas.
- SELECT FIRST BLOCK (F3) : This operation together with the next one is used to select two connected blocks to disconnect the link between them.
- SELECT SECOND BLOCK (F4) : The second connected block is selected similarly to the first block.
- CONNECT BLOCKS (F5) : Any pair of the blocks on the canvas can be connected after selecting this operation.
- DISCONNECT BLOCKS (F6) : This is used to disconnect two blocks already connected.
- UNDO (F7) : This operation is to cancel the last operation applied.
- CLEAR ALL (F8) : When this operation is selected the schematics constructed so far is destroyed and the canvas is cleared.
- RESET EXECUTION FLAGS (F9) : As it is explained in "Executing the Schematics" the execution of a schematics for a second time or later needs the execution flags of all the blocks to be reset which are set by the previous execution.
- CLEAR DISPLAY PIXRECTS (R1) : This operation enables the user to free the memory pixrects allocated to the loaded images.
- REDISPLAY ALL (R2) : The user may use this operation to refresh the whole canvas and to reconstruct the schematics automatically.

[3]Please refer to appendix A for more information.

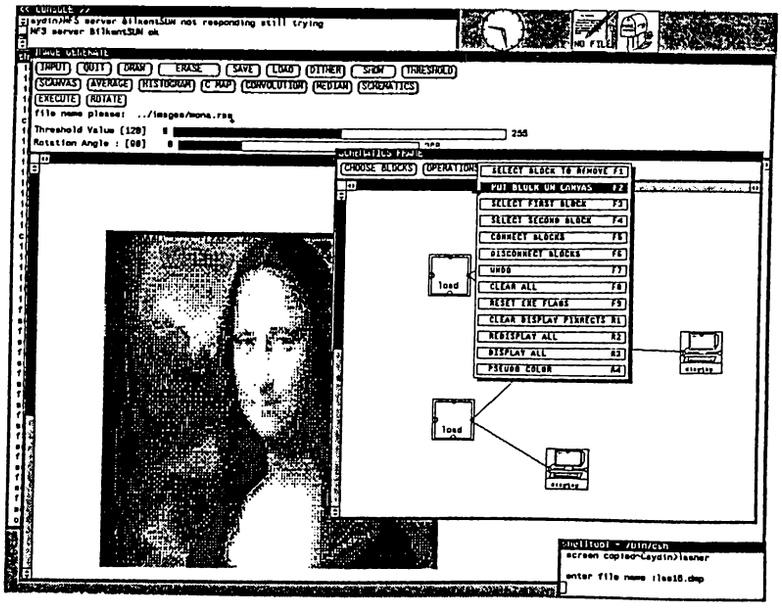

Figure 4.8: Operations on the blocks

- DISPLAY ALL (R3) : This operation causes the images stored in the display blocks to be displayed consecutively on the canvas.
- PSEUDO COLOR (R4) : This last operation is used to color a displayed image on the canvas in an unrealistic way.

4.5.3 Constructing a Schematics

The construction process of a schematics is straightforward. To start the process one of the icons representing an operation should be selected first. The selected block can be placed on the canvas after selecting the PUT BLOCK ON CANVAS operation from the operations menu. When the placement of at least two blocks has been done they can be connected to each other by selecting the CONNECT BLOCKS operation. Alternatively the blocks can be connected after placing all of the blocks on the canvas.

If the last selected functional block or the operation is to be repeated, it is unnecessary to select them again from the menus. That is the last operation or the block can be reused without selecting them again. Once the construction of a schematics has been completed the attributes of the instances can be set by using their control windows.

```
struct object_list {
      int                       inst_no;
      int                       obj_id;
      int                       execution_flag;
      struct block_pos          upper_left;
      struct block_pos          pin_pos_next[5];
      struct block_pos          pin_pos_pre[5];
      struct object_list        *next_obj[5];
      struct object_list        *pre_obj[5];
      struct object_list        *next;
} *obj_head, *curr_obj;
```

Figure 4.9: Data structure of the functional objects

4.5.4 Internal Representation of a Schematics

A schematics is represented by 13 linked lists internally. The most important of these is the one representing the objects and it is called the *object_list*. The C code for the data structure of the objects are shown in figure 4.9 again for convinence.

The other 12 linked lists represent the selected objects in each of the 12 object classes.

As it is shown in figure 4.9 any functional block is represented by some attributes to uniquely identify any instance of an object class (inst_no and obj_id [4]), the attribute to keep the position of the block on the canvas. The information related to the connectivity is also kept in the data structure. For practical considerations there can be 5 objects that may be executed prior to the current object and pass execution results to it. In the same manner an object after finishing its execution can pass the results to a maximum of 5 other objects. Therefore 5 previous objects and 5 next objects can be represented by this data structure. This can easily be modified to be dynamic. The link *next* in the structure represent the next object in the linked list in the order as the objects were placed on the canvas.

To simplify the discussion of how the schematics is represented internally, an example is going to be carried out.

In this example assume a schematics that will display the "and" of two

[4]inst_no is an integer and object_id changes from 1 to 12.

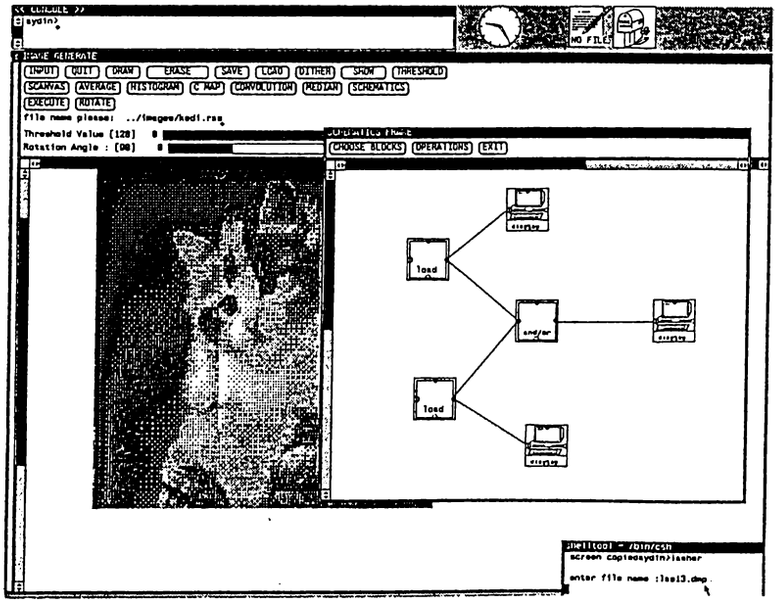

Figure 4.10: A schematics to find the AND of two images

images is to be constructed as shown in figure 4.10. To do that we have to place two instances of the load block, one instance of the AND/OR block and one instance of the display block on the canvas. The order of placing them on the canvas can be arbitrary, but assume they are placed as follows for the sake of simplicity:

- Place the first instance of the load block : an object instance as shown in figure 4.11 is generated and inserted into the object list[5].

  Besides inserting the above instance to the object list the load block instance itself is also inserted into its own linked list. This creates the instance in figure B.2 in appendix B.

- Place the second instance of the load block : In this case another object instance as in figure 4.11 will be generated. The values of the attributes are given in figure B.3 in appendix B.

- When the construction of the schematics is completed the object link will consists of 4 instances and the new situation is shown in figure 4.12. The attribute values are in figures B.4 through B.7.

One final point about constructing the schematics is : The connections of the instances are performed as soon as the connect block operation is chosen

[5]The values of attributes are given in appendix B.

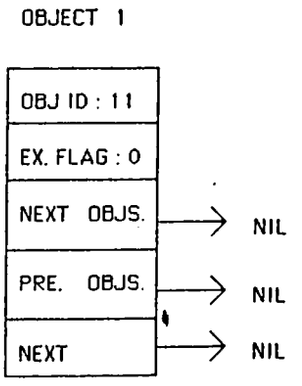

Figure 4.11: The first object instance.

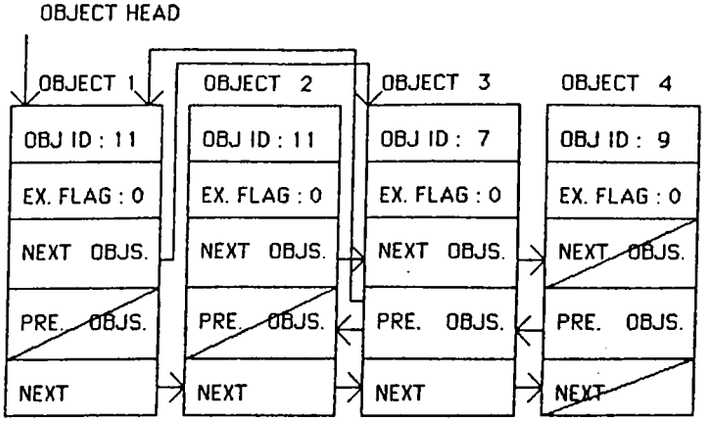

Figure 4.12: The final view of the object list.

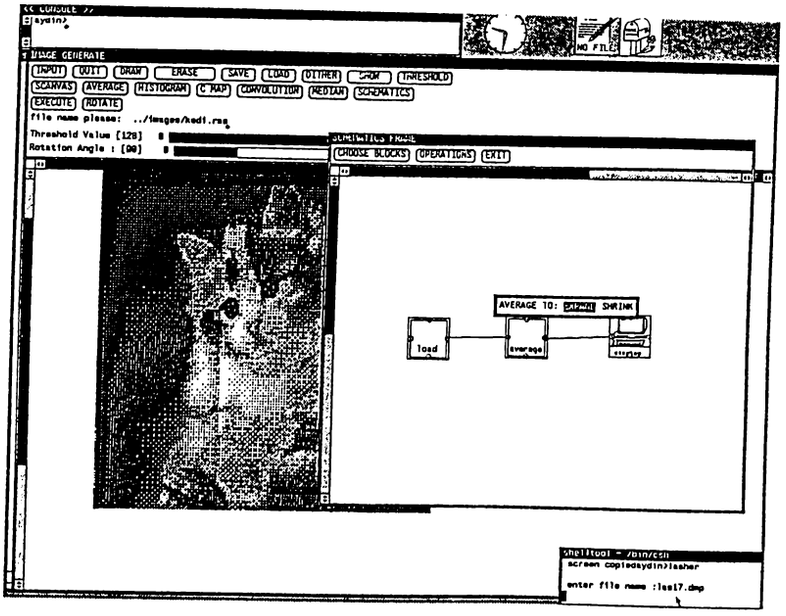

Figure 4.13: The control window of the average block

from the operations menu or by pressing the function key F5 and the link between the chosen blocks is drawn. In this case the captured event will invoke a procedure the function of which is to update the internal structure of the schematics linked lists.

4.5.5 Changing the Settings of Blocks in a Schematics

When the construction of a schematics is completed the attributes of the data structures related to the object classes like save, load, etc will have some default values. These attributes can be set again by the user. This is accomplished by using the data template or the control window of the related instance. Every object type in the system does not have a control window. The ones that have a data template window are listed below together with the functions of the data template windows :

- AVERAGE BLOCK : The control window given in figure 4.13 of this object type is used to select between expanding or shrinking average.
- CONVOL : This class has two control windows.The first one, given in figure 4.14, is used to enter the sizes of the convolution kernel and the other, given in figure 4.15, is used to enter the entries of it.

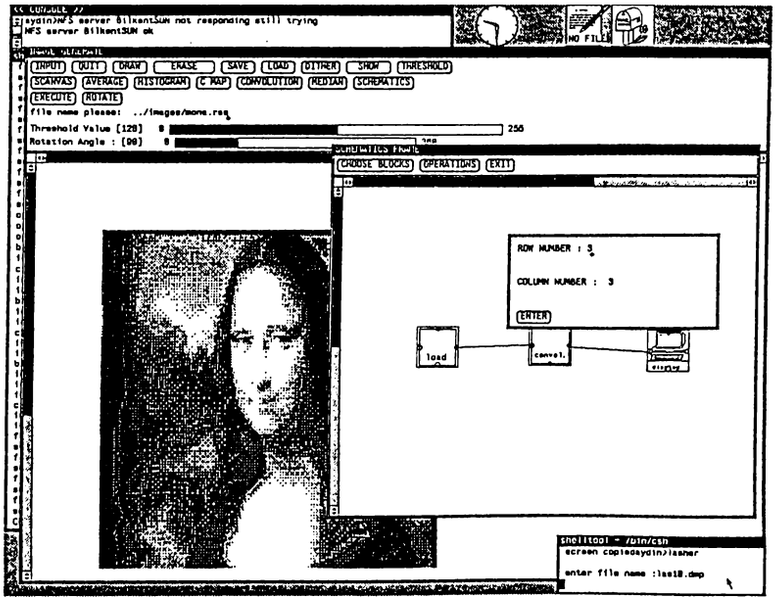

Figure 4.14: The first control window of the convolution block

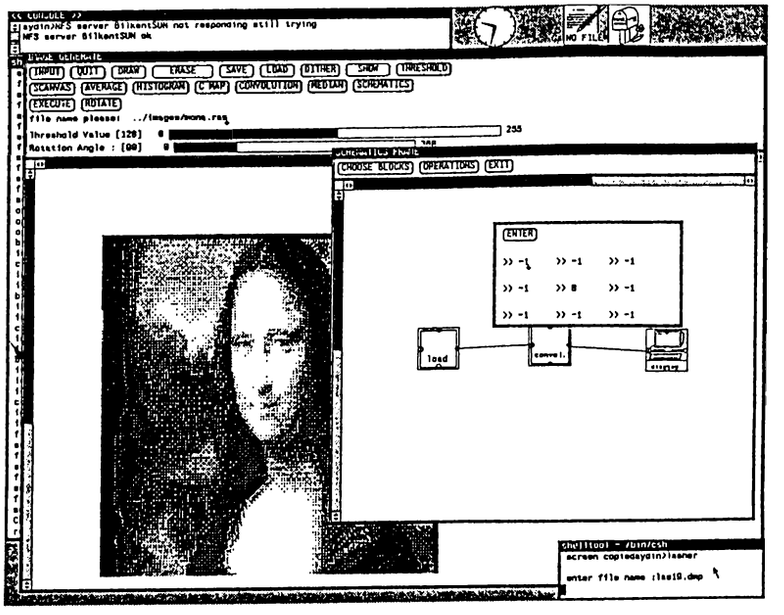

Figure 4.15: The second control window of the convolution block

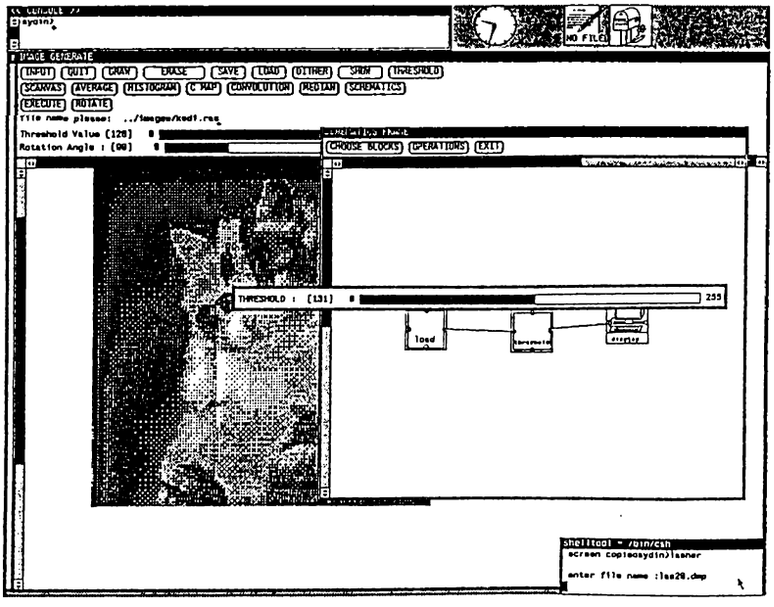

Figure 4.16: The control window of the threshold block

- THRESHOLD : The control window of this class contains a slider item. This item is used to select the threshold value that can range from 0 to 255. The control window is in figure 4.16.

- MEDIAN : The control window is used to enter the size of the median window in terms of row and column numbers as shown in figure 4.17.

- AND/OR : A panel subwindow with five choice items, as given in figure 4.18, constitute the control window of the class. The choices are : AND, OR, XOR, PLUS, MINUS. The default choice is the AND operation.

- POINT OP : This block is used for intensity and contrast enhancement. Therefore its control window contains two panel items(figure 4.19) to accept the values of intensity and the contrast. The default values for the items are 0.0 and 1.0 respectively.

- LOAD : The control window(figure 4.20) of this class is used to enter the name of the file from which an image in the raster file format will be loaded. The default value of the file name is NULL.

- SAVE : The control window of this object type is exactly same as the control window of the LOAD block and is used in the same manner. The control window is given in figure 4.21.

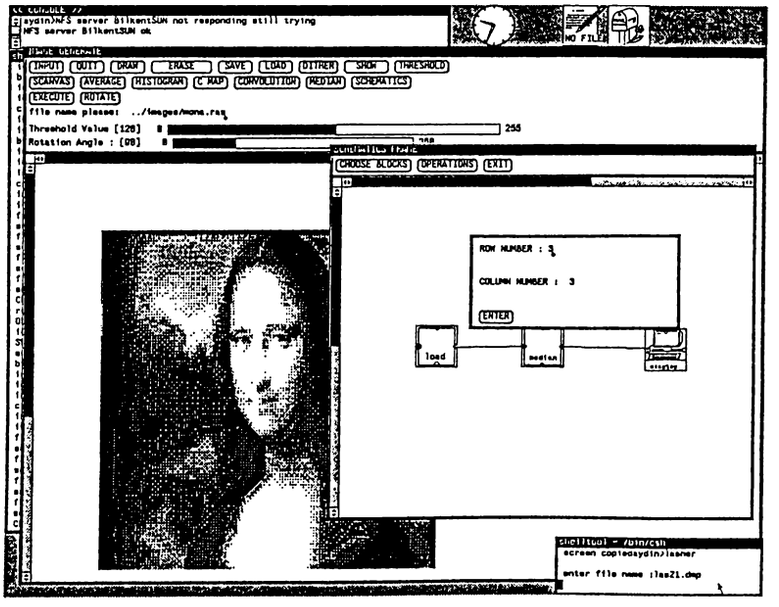

Figure 4.17: The control window of the median block

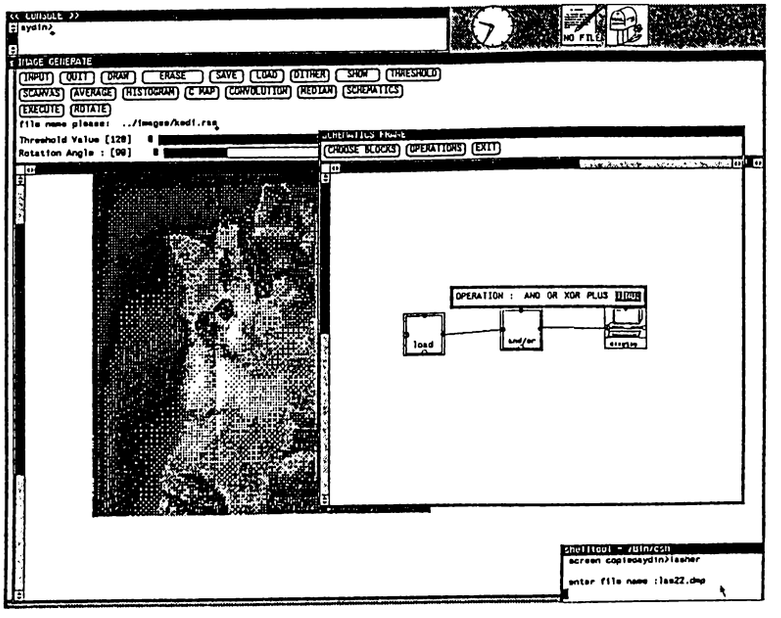

Figure 4.18: The control window of the AND/OR block

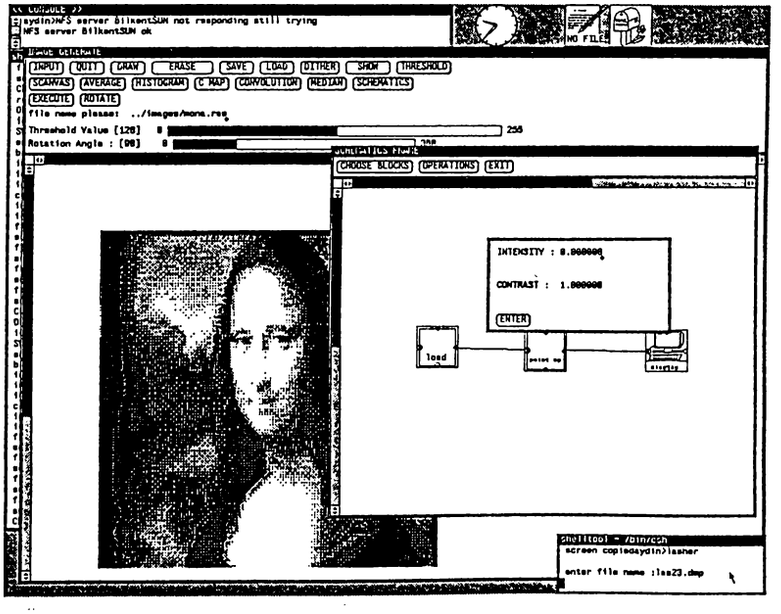

Figure 4.19: The control window of the point operations block

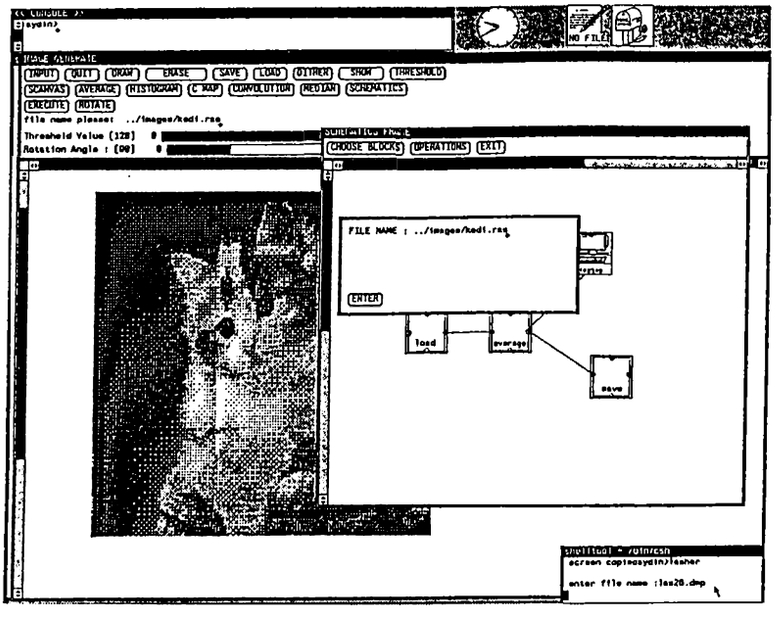

Figure 4.20: The control window of the load block

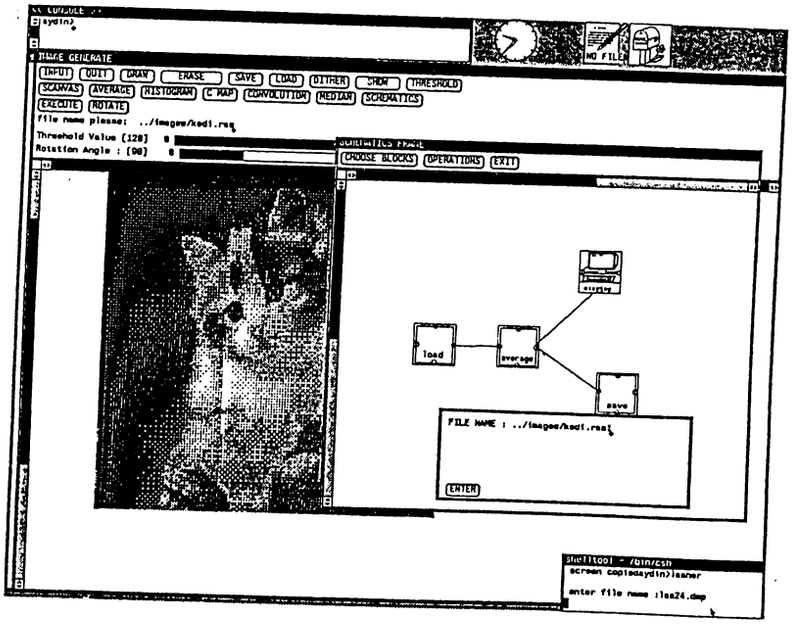

Figure 4.21: The control window of the save block

All of the control windows are opened when the mouse cursor is placed on any object instance and right mouse button is pressed. When the user tries to open the control window of an instance while there is already an open control window the old one will be closed automatically. That is, there can be only one open control window at any time.

4.5.6 Executing a Schematics

The execution of the schematics can be started after its construction. If the default values of the instances are wanted to be changed this should also be done before execution.

To start the execution of the schematics after completing the previous steps the user should press the EXECUTE button. When this is done the execution will begin if the schematics is executable. For a schematics to be executable it should contain at least one instance of an object class or block whose execution flag is equal to 0. When this flag is one it means that the instance has already been executed. In other words in an executable schematics there should be at least one block that can be executed. Another criteria for a schematics to be executable is that there should be no circular references. That is, if a block passes the results obtained after its execution to another block that is linked to itself then the schematics is circular and it

is unexecutable. When the rule checker finds that the schematics cannot be executed the user is informed and the execution stops.

There should thus be a mechanism to find out whether an object instance is executable, if there is an executable object instance or if the schematics is non cyclic. To perform this validity check the implementation policy followed in the system is straightforward, and works as follows :

- Decide if an instance is executable :

  1. check if the execution flag is zero.
  2. determine if all the instances of objects that will pass their results to the checked instance have completed their execution.

- If the answers for these two questions are positive the instance is executable.

- Otherwise if the execution flags of all the items are equal to one the execution of the whole schematics has been finished.

- Otherwise the schematics is not executable.

  When an excutable object instance is located by using the above policy, it is executed. To execute any object instance its procedures are invoked. For example if the instance belongs to the average class the average procedure would be invoked and the obtained image would be stored in the instance variables of the related object instance. When an instance is executed other instances of the objects which are linked to the already executed instance are located and the resultant images are passed to them. After passing the results to other object instances unneeded storage allocated to the completed instance is made free.

  The LOAD block is a special class in the sense that it does not need to wait other blocks to finish their exeucution. In other words the instances of this block are directly executable ones in case of their execution flags are equal to zero.

  The DISPLAY and SAVE blocks do not pass any processing results to other instances, that is they are final blocks in the schematics.

  During the execution of the schematics many potential errors are checked. Some of these are missing file names, insufficient memory, and trying to process a null image. When errors are captured the class of the object and its instance number is printed on the screen and the execution

is interrupted. As a feedback to the user the block that is executing currently is inverted that is it is displayed in reverse video.

After the execution is completed successfully the user can see the results of processing. To do that he locates the mouse cursor on any of the placed display blocks and presses the right mouse button. If an image is stored in the selected display block it will be displayed on the canvas, otherwise the user will be informed that the block cannot display.

4.5.7 Modifying the Constructed Schematics

During the construction of the schematics if mistakes are noticed it is possible to modify the schematics easily. The modification may also be necessary after executing a schematics to change the processing sequence.

In the system the following operations are used to modify the schematics:

- SELECT BLOCK TO REMOVE (F1) : After this operation is selected the desired instances will be removed by pressing the left mouse button on the icon of any instance. This operation will disconnect all the lines connected to the removed block and will erase it from the screen.
- SELECT FIRST BLOCK : This operation is used to select the first point of a link which is to be removed.
- SELECT SECOND BLOCK : This is used in the same way as the above operation , it selects the second connection point instead of the first.
- DISCONNECT BLOCKS : After selecting the start and finish points of a link by using this operation the link is broken and erased.
- CLEAR ALL : If it is necessary to start from the scratch, then this operation is selected to clear all of the schematics canvas and initialize all of the schematics linked lists to empty.
- RESET EXECUTION FLAGS : When a schematics is to be reexecuted this operation should be selected that will reset all execution flags to zero.

If necessary it is also possible to change the attributes of some of the blocks by using their control windows. This operation is clearly applicable to only the blocks having a control or data template window.

5. CONCLUSION

In this thesis an image processing system has been presented. The system allows the user to use image processing routines in direct and indirect mode as described.

Both modes of operation have been designed to be in a very user friendly manner. This has been achieved by constructing the system on Sun View which is an object oriented toolkit. In the direct mode of operation the procedures can be invoked by just pressing the related button in the control panel. On the other hand if the procedures are used in the indirect mode, a schematics has to be constructed. A schematics is nothing but a flowchart like drawing to select the desired procedures and their execution order. By using a schematics it is possible to process a group of images in a continuous manner. The biggest restriction on the schematics is that, there can be no circular references.

It is possible to execute a schematics more than once. The links between the blocks can be broken, any block can be removed from the constructed schematics. In short it is possible to modify the schematics in any way.

When it is desired to see the intermediate or final results of image processing after the execution of the schematics, it is necessary to link the display devices to appropriate places. Then by pressing the right mouse button on one these display blocks the image is displayed on the canvas. An option in the system allows all of the images stored in the display blocks to be displayed on top of each other. By using this option and storing many different instances of an object it is possible to obtain animation.

The modes of operation can be used interchangebly in the system. That is when a schematics is being constructed it is possible to execute a procedure in the direct mode or vice versa.

The main control of the system is dependent on the Sun View's notifier. This is a system to interpret many inputs from all sources and distribute them to appropriate procedures. In such a mechanism it is unnecessary to have a main program which is to interpret many inputs and events and call the suitable procedure, instead the notifier calls the registered procedures that were registered with particular events. This control mechanism allowed the application to be written procedure by procedure. Therefore the system is very modular and the probability of a bug in the procedures is very low, if not zero.

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 standarts both in data communication and in storage-retrieval techniques and formats, let alone in interaction principles. The suggested approach seems suitable since it represents the actual events taking place during image manipulation procedures. This is not much different, say, than the insertion of special commands and characters in a text for a text processor to format it.

Images displayed in some special device data template windows can be edited manually by operations such as cut and paste, colormap manipulation, painting, etc. This is indeed 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 interface is believed to be a feasible solution in computer imaging applications.

REFERENCES

[1] Donald Hearn and M. Pauline Baker, *Computer Graphics*, Prentice Hall, 1986.

[2] David F. Rogers, *Procedural Elements For Computer Graphics*, Mc Graw Hill, 1985.

[3] Sun Microsystems, Inc., *Sun View Programmer's Guide* , Mountain View, California, 1986.

[4] Bulent Ozguc, "Thoughts On User Interface Design For Multi Window Environments," Second International Symposium on Computer and Information Sciences, Istanbul, 1987, pp.477-488.

[5] Benjamin M. Dawson, "Introduction to Image Processing Algoritms" , BYTE, March 1987 pp.169-186.

[6] William M. Newman, Robert F. Sproull, *Principles of Interactive Computer Graphics*, Mc Graw Hill, 1979.

[7] Jonathan P. Jacky and Ira J. Kalet, "An Object-Oriented Programming Discipline For Standard Pascal", Communications of the ACM, vol. 30, no. 9, september 1987,pp.772-776.

[8] O. M. Nierstrasz, "What Is Object In Object-Oriented Programming", The proceedings of the CERN, Renesse, The Netherlands, 1986, pp.1-13.

[9] Kenneth R. Castleman, *Digital Image Processing*, Prentice Hall,Inc., Englewood Cliffs, N. J., 1979.

[10] Owen M. Densmore, David S. H. Rosenthal, "A User-Interface Toolkit in Object-Oriented POSTSCRIPT", North Holland Computer Graphics Forum 6 (1987), pp. 171-180.

A. THE USER'S MANUAL

A.1 Introduction

This appendix gives a detailed information related to using the system. The available functions can be used in two different modes of operations. These are called direct and indirect modes. First using the operations in the direct mode and then using the operations in the indirect mode will be explained. After completing the description of using the system in each mode a detailed example will be carried out.

A.2 How to Start ?

The implementation of the system has been done by using the Sun View environment as described before. As a result of this the system can be used only in this environment. Hence the user should have some knowledge about using this environment, and the compenents of it such as windows. Before starting the system the user should make sure that he or she is in this environment, if not the "suntools" command or the one neccassary to go into this environment should be executed.

Once the user in the Sun View and he has opened a work window, the image processing system can be started by typing

im

in the Unix[1] prompt. About one second later after entering this command the main window of the system will be opened. This window consists of two parts and they are a panel subwindow and a drawing subwindow or a

[1] Unix is a trademark of Bell labaratories.

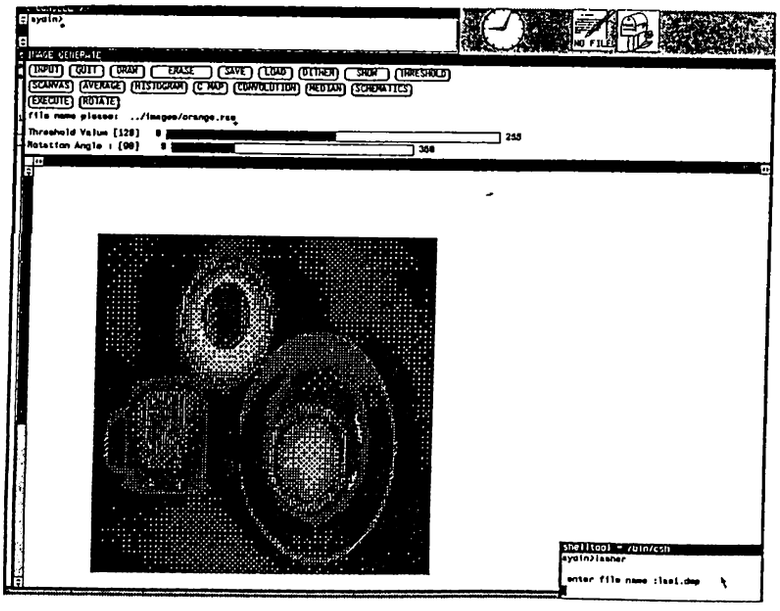

Figure A.1: Main window of the system.

canvas. The pane subwindow contains many buttons, two sliders and a text item to enter the name of a file to load an image from or to save an image. The buttons and the other items in the panel subwindow are used to use many image processing functions in the direct mode, to start the colormap manipulation functions, and to start the indirect mode of operations(figure A.1). The use of the system in the direct mode is explained in the next section and the indirect mode is described later.

A.3 Using the System in the Direct Mode

The image processing functions in this mode are applied to the current image. The current image is the one that has been loaded or the one obtained as a result of the last operation. Therefore when the system has just started an image should be loaded otherwise the functions are not applicable.

All of the items in the panel subwindow are listed and described below in the order of their placement.

- QUIT : This is a button and used to exit from the system comletely. After the user presses this button he is asked to confirm and if he does so the operations are terminated.

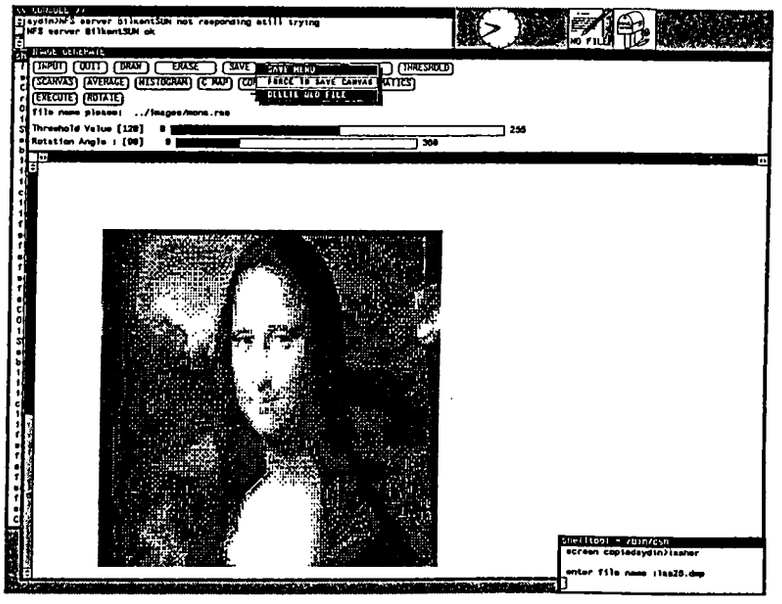

Figure A.2: The menu of the SAVE button.

- ERASE : This button is used to clear the canvas but the current image stored in the memory is not affected.

- SAVE : The user should select this button if he wishes the current image to be saved on the disk. This button has a menu and it can be used by pressing the right mouse button on it as shown in figure A.2. The menu items are used to confirm to overwrite an image file and to use the option of saving the image on the canvas respectively.

- LOAD : When an image should be loaded form the disk and displayed on the canvas this item should be selected. After an image is loaded it will be the current image, that is any selected operation will be applied to it and the resulting image will be the current one. This button has also a menu and the items in the menu are used to find OR, AND, and, XOR of the loaded and the image already on the canvas. There are two other menu items namely PIX SRC and PIX DST. PIX SRC of these are used to load an image and display it without any regard of the image on the canvas. That is the loaded image is displayed on the screen and the old image does not affect it. The PIX DST item affects the display of the loaded image so that only the pixels of the loaded image corresponding to the pixels of the image on the canvas having a value other than zero will be displayed.

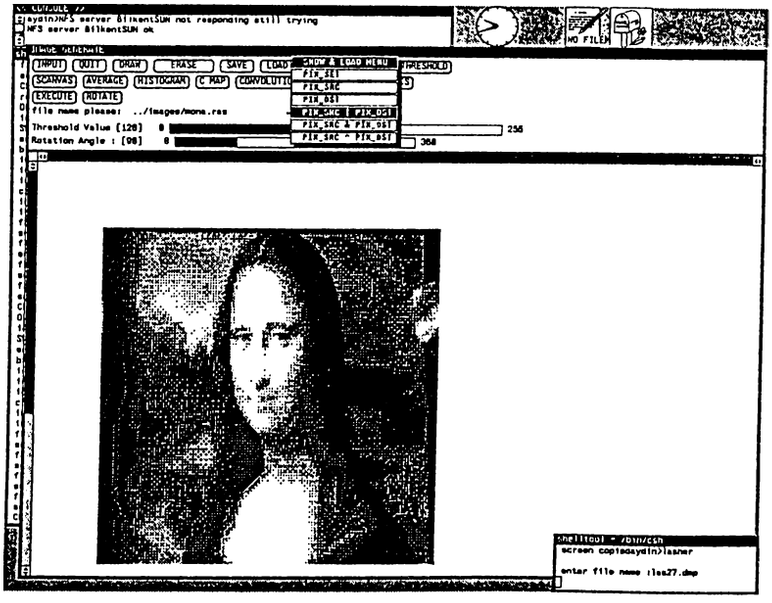

Figure A.3: The menu of the LOAD button.

- DITHER : After selection of this button the dithered image of the current one is obtained and then displayed on the canvas. This operation is especially usefull when a hard copy of a gray-scale image is to be obtained by using a bilevel matrix printer.

- SHOW : This fuction is usefull to transfer a file storing an image in a format other than the raster file format(chapter 4). In such a case the user should enter the length of the header of the file and the size of the image in terms of row and column number. The image is read byte by byte into the memory and becomes the current image. Consequently when it is saved it will be in the standard raster file format.

- THRESHOLD : Selection of this operation produces an image so that all of its pixels having a value less than a definite value, that is the threshold will have the background color and all other pixels will have the foreground color.

- SCANVAS : This button is used to save an image in such a format that it can be printed. The depth of the pixels of an image is usually eight and such images cannot be stored by using this operation since all pixels should have a value of zero (black) or one (white) to be printed. Hence an image should be transfered to the desired form before the application of SCANVAS operation. In the system two operations namely DITHER and THRESHOLD produce images so that they contain only black and

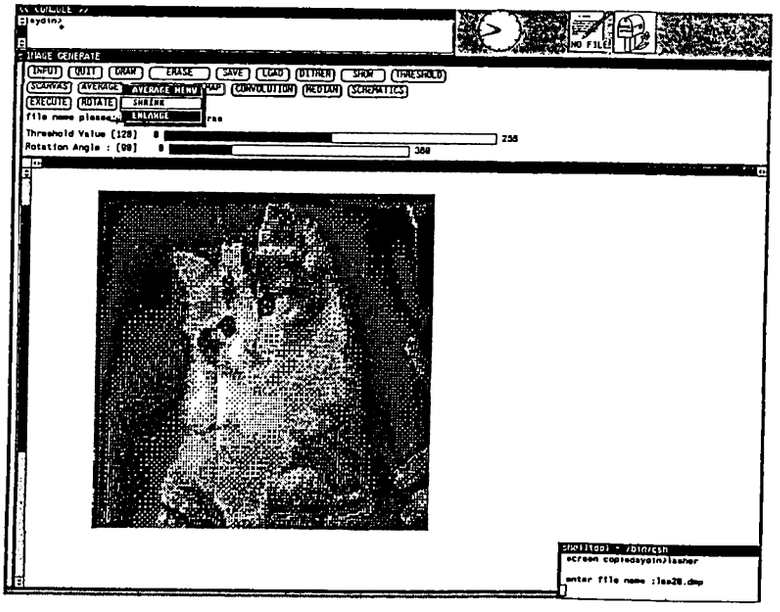

Figure A.4: The menu of the AVERAGE button.

white pixels.

- AVERAGE : This function produces an image that is either larger or smaller than the original image by a factor of four. The resulting image is also displayed on the canvas. The mode of average operation is selected by using the items of the menu associated with the button as shown in figure A.4.

- HISTOGRAM : The histogram of the current image will be found and displayed at the bottom of the canvas as a result of this operation.

- C MAP : After pressing this button a new window is opened. This new window contains four slider items that are used for selecting the index of the colormap of the current image and the intensities of the major colors (red, green, and blue) stored there. Two other items in the newly opened window are buttons and they are used to select the painting style of the displayed image and cycle the colormap entries respectively. The detailed explanation of changing the colormap of an image is presented later.

- CONVOLUTION : Before application of the convolution operation to the current image the size and the entries of the convolution kernel should be entered. To enter these values first the SHIFT key and the left mouse button should be pressed at the same time when the mouse

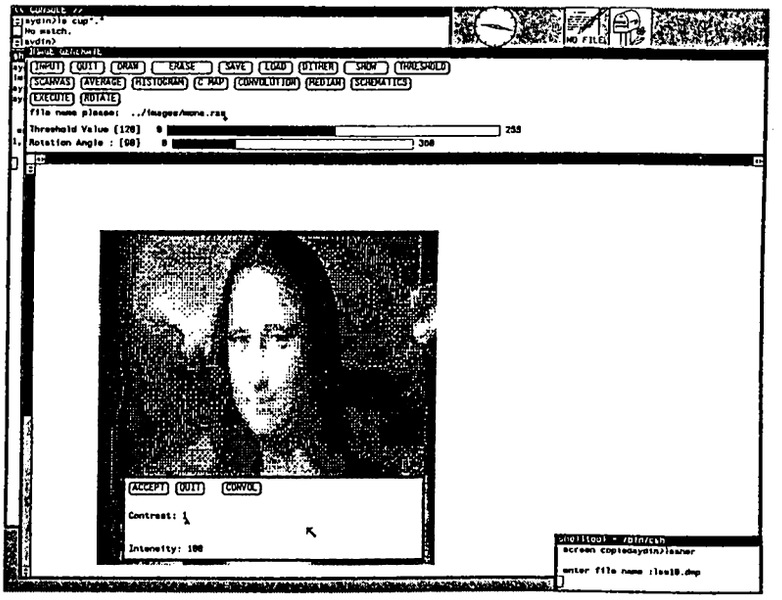

Figure A.5: The first window of the Convolution.

cursor is on the canvas. As a result of this the window shown in figure A.5 will be opened.

Now to enter the number of rows and columns the CONVOL button should be pressed. This will cause another window to be opened as shown in figure A.6. After entering the sizes of the konvolution kernel and pressing the ACCEPT button causes another window to be opened. In this window(figure A.7) the enries should be entered and then the ACCEPT button should be pressed.

When the entries are entered and the ACCEPT button is pressed, the CONVOL button can be pressed to start the convolution procedure. By using different convolution kernels it is possible to amplify vertical, horizontal, and all lines of the current image(chapter 4). It is also possible to superimpose the resultant image of this operation with the current image.

- MEDIAN : Median filtering of the current image is invoked after pressing this button. This filter is suitable to blur an image or to remove salt and pepper noise from it. Before the use of this operation the size of the median window has to be entered by using the window shown in figure A.5.

- SCHEMATICS : This button is used to invoke the indirect mode of operations. That is when the SCHEMATICS button is pressed a new

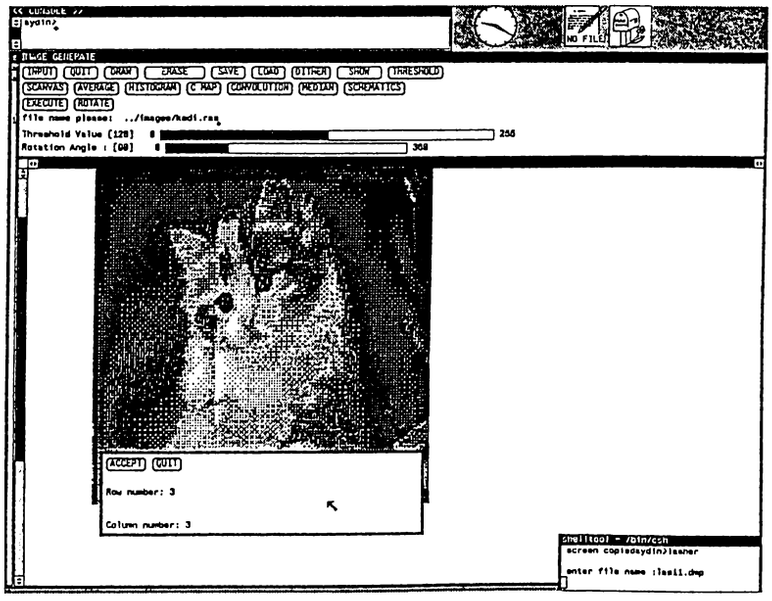

Figure A.6: The second window of the Convolution.

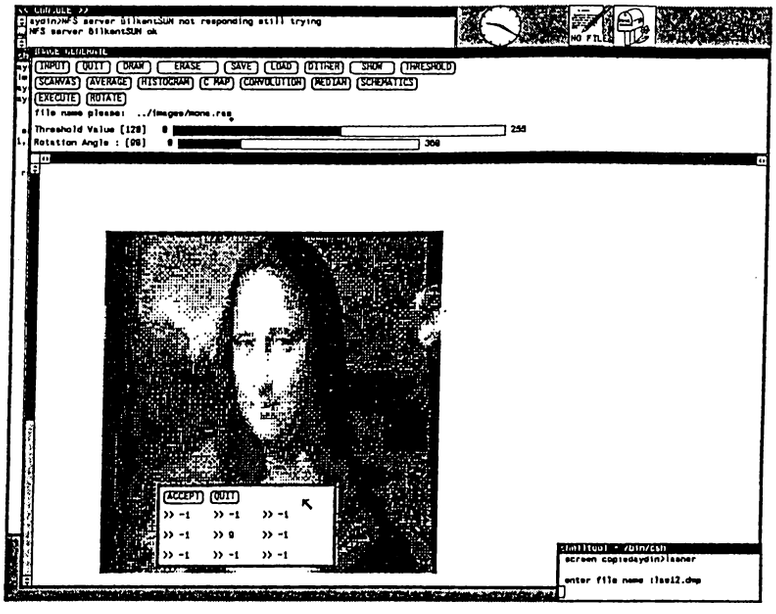

Figure A.7: The third window of the Convolution.

window as shown in figure A.8 is opened and from now on the operations can be applied to the images by use of a schematic. Using the indirect mode and construction of a schematic is described later in this chapter.

- EXECUTE : A constructed schematics in the indirect mode is executed by pressing this button. How a schematics is executed is expalained in detail later.

- ROTATE : This button rotates the current image with respect to its center by an arbitrary angle. The value of the rotation angle can change from 0 to 360 degrees and is selected by using the rotation angle slider.

- File name please : This is a text item and used to enter the name of the file from which an image will be loaded or to which an image will be stored.

- Threshold value : This is one of the slider items and used to enter the threshold value that is to be used in the threshold operation described before. By using this item it is possible to select a threshold value between 0 to 255, that is from black to white.

- Rotation angle : This item is a slider and is used for selecting the rotation angle. The rotation angle can be chosen from 0 to 360 degrees by using this slider.

A.4 An Example of Using the Direct Mode

To illustrate the uses of many panel items and to give an overview of using the system, a detailed example is carried out below. While going through this example please refer to any of the figures related with the layout of the screen.

Assume a previously stored image with name "orange.ras" will be loaded, and many operations will be performed on it. When an operation is completed its dither will be found and shown. Applying the dither operation is necessary to print the resultant image. After taking the dither of the loaded image it is displayed as in figure A.8. Note that to load and then apply the dither operation it is necessary to enter the name of the image in the file name item and then the DITHER button should be pressed.

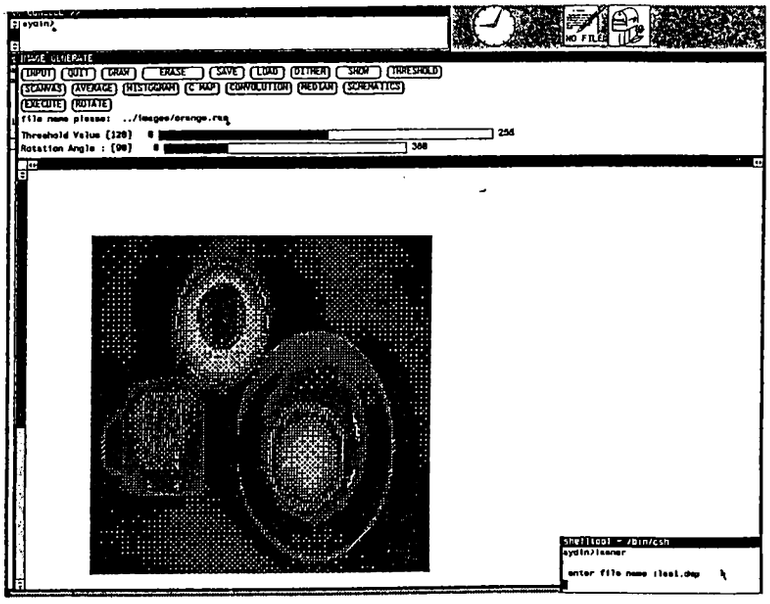

Figure A.8: The result of dither operation.

At this moment the loaded image "orange" is the current one and all of the operations described can be applied to it in sequence. First assume we want to apply the threshold operation to it, when the threshold value is 120. To do this it is only necessary to press the THRESHOLD button after adjusting the threshold value by using the related slider item. After completion of the operation the image in figure A.9 is obtained.

The average of the current image is taken for both enlarging and shrinking and the results are shown in figures A.10 and A.11 respectively. Before the AVERAGE button is pressed its mode should be selected by using the menu associated with the button. For example to enlarge the image by a factor of four the second menu item should be selected first and then the AVERAGE button should be pressed.

Note that when the image is enlarged by a factor of four the resultant image becomes the current image and application of the shrinking average does not make much sense since the original image is obtained so the shrinking of an image cannot be expalined well. Therefore first the original image will be loaded again and then the shrinking average will be applied.

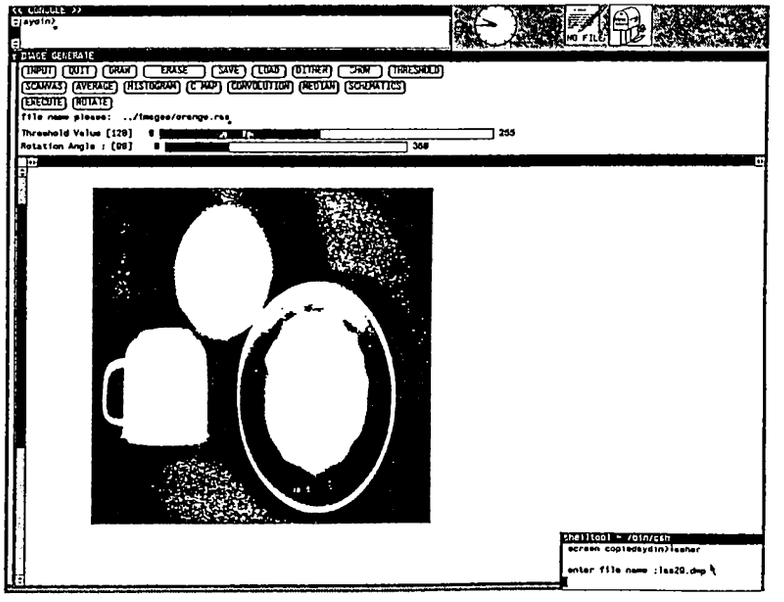

Figure A.9: The result of the threshold operation.

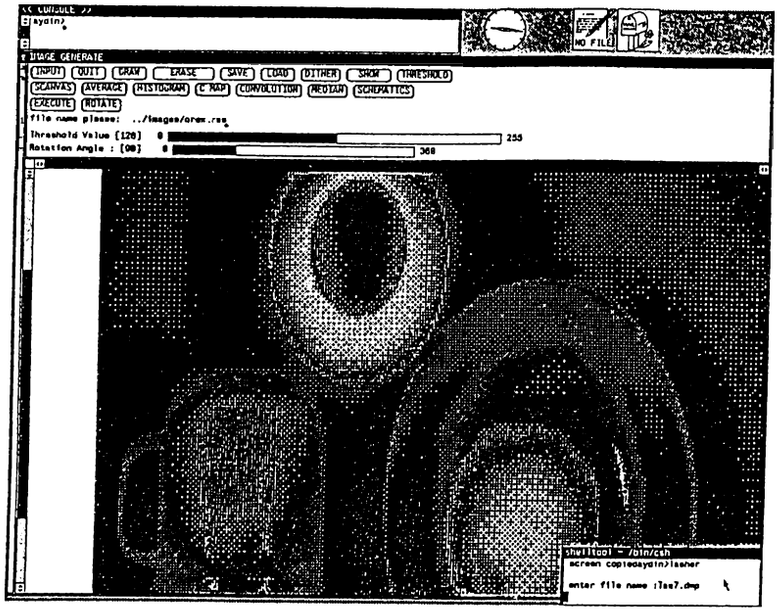

Figure A.10: The result of enlarging average.

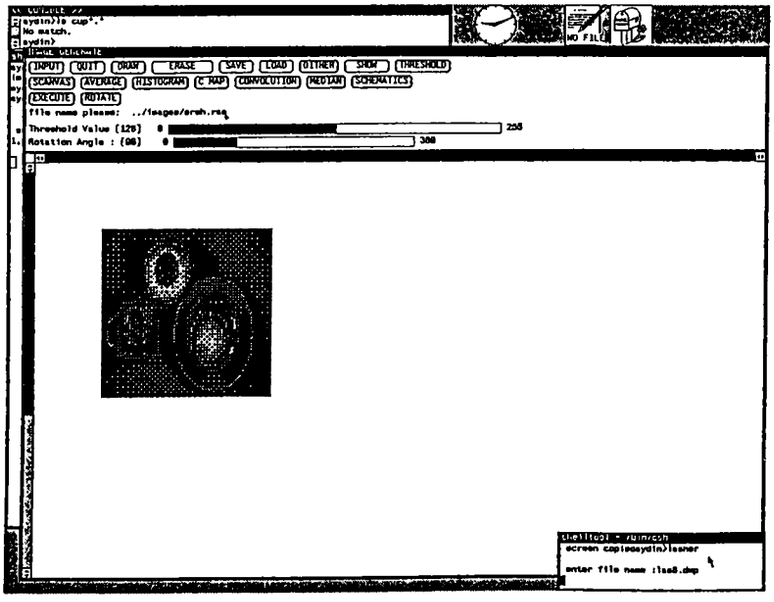

Figure A.11: The result of shrinking average.

A.5 Changing the Colormap

The content of the colormap entries or the values of some pixels of the displayed image can be changed by using the colormap change utilities. To start the colormap manipulation the C MAP button should be pressed. Note that this utlity is available only in the direct mode.

Following the pressing of the C MAP button a new window shown in figure A.12 is opened. This window consists of a panel subwindow containing four sliders and two buttons.

The silider labeled colormap index shows the active entry of the colormap of the displayed image. There are 256 entries and each of them can be made active.

The other three slider items are used to adjust the values of the three main colors(red, green, and blue) that will be stored in the active entry. Making any modification in the red, green, and blue sliders cause the color of all pixels having an index value equal to the active colormap inedex to be changed in real time. Therefore by choosing an arbitrary index and then changing the colors it is possible to modify an image globally.

The active entry can also be set to the index of a particular pixel. This

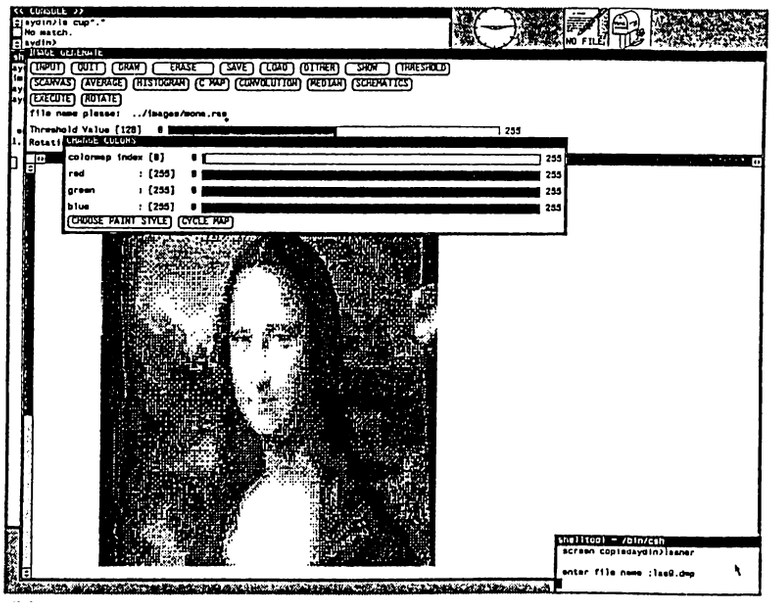

Figure A.12: The colormap manipulation window.

can be done by pointing to the particular pixel of the image and then pressing the middle mouse button. As a result of this operation the color of a group of pixels can be changed immediately.

There are two button items in the colormap manipulation window and they are used to change the paint style for the image on the canvas and to cycle the values of the colormap entries respectively.

The first button has a menu associated with it and its items are POINTS, INJECTION, and BLOCKS. These items are used to select the pixels that will be affected by the paint process. The POINTS cause only one pixel to be affected. The INJECTION and BLOCKS cause a group of pixels to be affected that were randomly chosen in a 10 by 10 window and all of the pixels in a 10 by 10 window respectively. The values of the pixels are changed to the active colormap index as the mouse cursor is dragged on the canvas while the right mouse button is pressed.

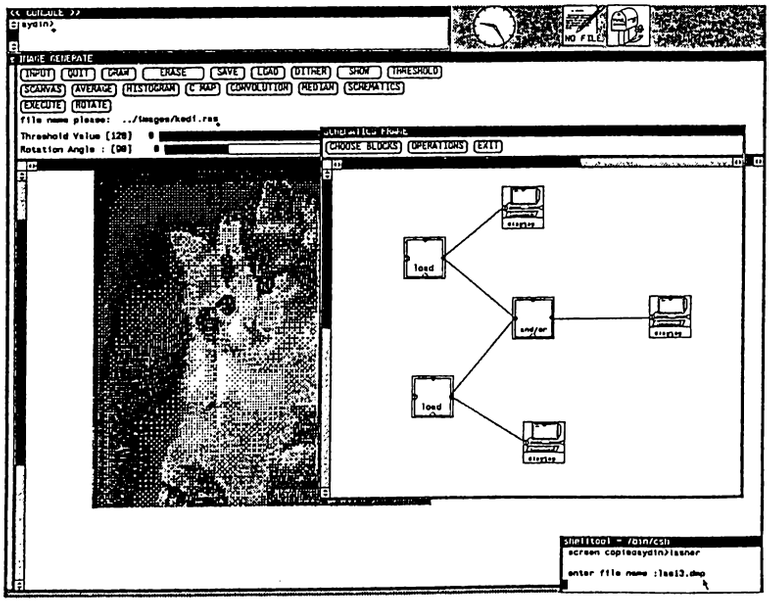

Figure A.13: The window of the indirect mode.

A.6 Using the System in the Indirect Mode

A.6.1 Introduction

The second mode of using the system is the *indirect mode.* To start working in this mode first the SCHEMATICS button should be pressed in the direct mode. When this button is pressed the window of the indirect mode will be opened(figure A.13).

As it is seen in figure A.13 the indirect mode window consists of two subwindows. The subwindows are panel sunwindow and a canvas. In the panel subwibdow there are three button items, namely CHOOSE BLOCKS, OPERATIONS, and EXIT. Of these the EXIT button is used to quit from the indirect mode and return back to the direct mode. The other two buttons are used through their menus. The menu associated with the CHOOSE BLOCKS button is used to select the representative icon of a procedure and the second one is used to select an operation. Some of the operations are applicable to particular instances of the blocks and some others are global to the indirect mode.

The icons of available procedures and the opeartions in the indirect mode are shown in figures A.14 and A.15 respectively.

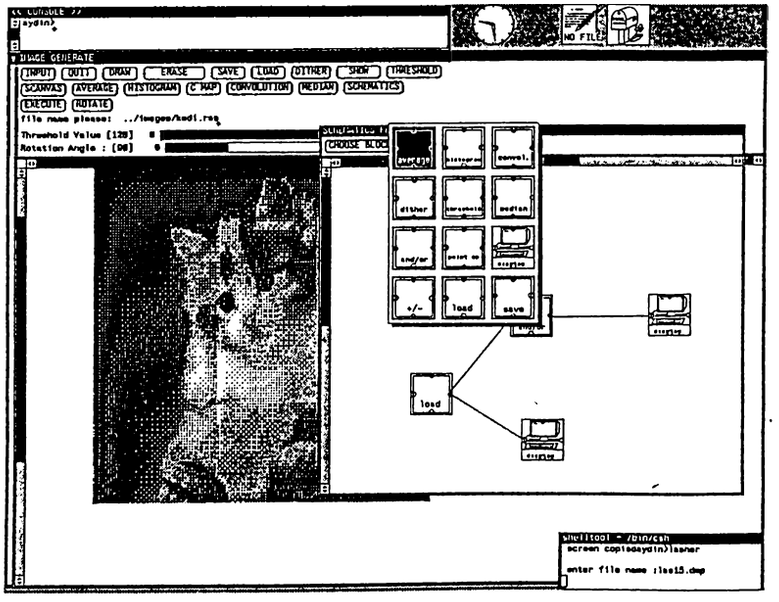

Figure A.14: The icons representing the procedures.

A.6.2 Icons Representing the Procedures

The icons and the procedures they represent are explained below.

1. AVERAGE : This icon represent the average operation. By using this icon it is possible to enlarge and shrink an image.
2. HISTOGRAM : When the user wishes to obtain the histogram of an image this icon should be selected.
3. CONVOL : By using this icon the procedure to convolve an image with a convolution kernel is performed.
4. DITHER : This icon represents the dither procedure.
5. THRESHOLD : The threshold to be applied to an image is chosen and then applied to the related image by using this icon.
6. MEDIAN : The median filtering algorithm is represented by this.
7. AND OR : The frame processes AND, OR, XOR, PLUS, and DIFFERENCE are used after selecting this icon.
8. POINT OP : Point operations to be applied to an image are represented by this icon.

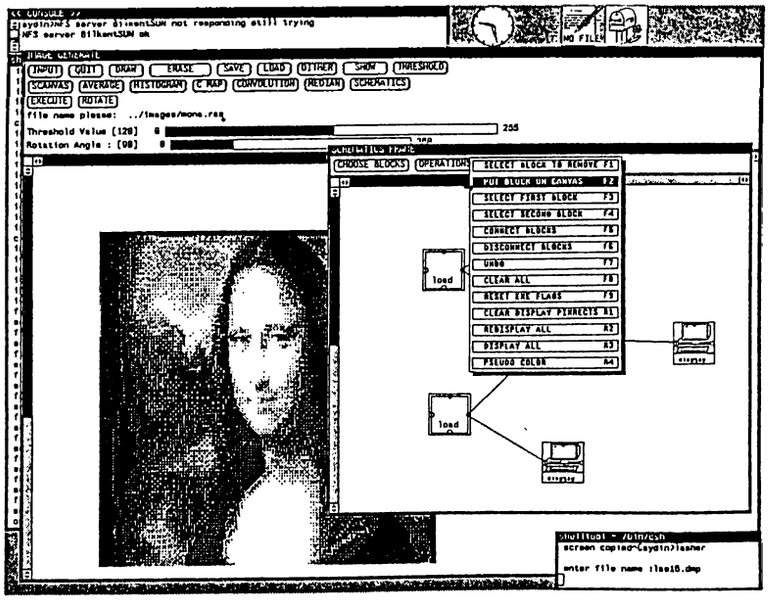

Figure A.15: The operations in the indirect mode

9. DISPLAY : This icon represents the display device.

10. +/- : Another group of frame operations can be represented by using this icon.

11. LOAD : This one represents the loading of an image from the disk.

12. SAVE : This icon represents the procedure to save an image to the disk.

A.6.3 The Operations On the Blocks

The operations used to place, connect, or disconnect the functional blocks are in the menu associated with the OPERATIONS button. This menu also includes some other opearations such as clearing canvas, and redisplaying the schematic. The available operations are shown in figure A.15.

As it is shown in figure A.15 it possible to choose any operation from the menu or by pressing the related function keys shown in the menu items. Some of the operations in the menu are executed as soon as they are selected on the other hand some others are executed after pointing to the necessary icon that is wanted to be affected by the operation. The difference between these operations can be understood if nothing happens although the user has selected an operation.

The items in the operations menu are listed and explained below.

1. SELECT BLOCK TO REMOVE (F1) : This operation is executed after pointing to the block to be removed. It is used to delete any functional block from the schematics. The functional block is chosen by pressing the left mouse button when the mouse cursor is on it.

2. PUT BLOCK ON CANVAS (F2) : After selecting this operation previously selected functional block can be placed on the canvas. To place the block on the canvas the user should dragg the mouse while the left mouse button is pressed. When the block is in the proper position left mouse button should be released.

3. SELECT FIRST BLOCK (F3) : This operation together with the next one is used to select the end points of a link that is to be removed.

4. SELECT SECOND BLOCK (F4) : The second end point of a link that is to be removed is selected by this operation. In this and the previous operation the end points are selected by clicking the left mouse button on the block to which the link is attached.

5. CONNECT BLOCKS (F5) : After selecting this operation any two blocks can be connected to each other. To do that the user selects the first block to be connected by pressing the left mouse button on it. Dragging the mouse while the mouse button is pressed causes the link to rubber band. When the second block is reached the user releases the mouse button. If any of the ends points is not selected properly there will be no link formed.

6. DISCONNECT BLOCKS (F6) : Selecting this operation causes the link between two already marked end points to be removed. If the end points were not selected properly nothing will happen. In such a case the end points should be marked again and the operation should be repeated.

7. UNDO (F7) : This operation causes the last operation to be cancelled.

8. CLEAR ALL (F8) : When this operation is selected the schematic constructed so far is destroyed and the canvas is cleared. This operation should be used when a completely new schematic is to be constructed.

9. RESET EXECUTION FLAGS (F9) : Before reexecuting a schematic this operation should be selected. In this case the execution flags of

the blocks are reset to zero. Without applying this operation prior to reexecution keeps the old results.

10. CLEAR DISPLAY PIXRECTS (R1) : The number of images that can be stored in the memory at the same time is limited by the physical constraints. As a result of this from time to time it may be necessary to release the memory by using this operation.

11. REDISPLAY ALL (R2) : During the construction of a schematic the contact of the lines and the blocks causes some portions to be erased. In such a case the schematic can be redisplayed by using this operation.

12. DISPLAY ALL (R3) : When there are more than one display instances on the canvas each of them store an image. Those images can be displayed one after another quickly. If the stored images are different instances of an object this operation may be used to obtain animation.

13. PSEUDO COLOR (R4) : The displayed image on the canvas is made colored in an unrealistic way. To see the resulting image the right mouse button on the functional block having the displayed image should be clicked.

A.6.4 Constructing a Schematics

To construct a schematics the user first has to press the SCHEMATICS button in the main frame. This will cause a new frame to be opened containing a panel with three buttons and a canvas as shown in figure A.16.

The two of the three buttons in the panel of the schematics frame have menus connected to them and they are used after pressing the right mouse button. The menus contain available blocks and the operations which were described in the previous two sections.

Once the user is in the schematics frame he can select any of the functional blocks to be placed on the canvas or he can make any kind of changes in the partially or fully constructed schematics by using the operations. The operations can also be used through function keys or meta keys by an advanced user.

When the user has selected a functional block from the blocks menu, he can place it on the canvas after selecting the "PUT BLOCK ON CANVAS"

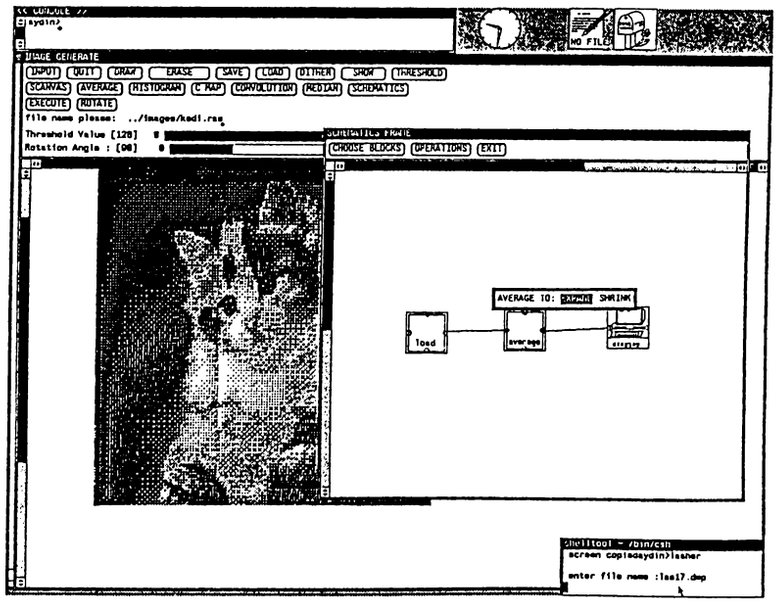

Figure A.16: Main frame and the schematics frame

item from the operations menu or by pressing the meta key F2. To put the selected block on the canvas after completing the previous two steps the user should press the left mouse button. In this case the selected icon representing the related function will appear. As the user moves or drags the mouse the icon will also move. This continues until the left mouse button is released. In this case releasing the mouse button event will be interpreted by the system as a signal to put the block into the internal representation of the schematics. The user continues in this fashion until he places all of the functional blocks that he or she wishes on the canvas.

Once the placement of the blocks has been completed, blocks can be connected to each other. To do this the user selects the first block by pressing the left mouse button on the icon representing the first connection point of the link between two blocks. After this, dragging the mouse while the left button is pressed will cause the line connected to the mouse cursor to extend in a straight manner. When the line is attached to the second block, releasing the mouse button will cause the proper event to be generated and this will invoke a suitable procedure to establish the connection between the two blocks. If either of the end points of the line is not on any block there will be no connection and the line drawn will be erased automatically. Furthermore the user will be warned and asked to retry.

If the last selected functional block or the operation is to be repeated,

it is unnecessary to select them again from the menus. That is the last operation or the block can be reused without selecting them again. Once the construction of a schematics has been completed the attributes of the instances can be set by using their control windows as described later.

Only some of the blocks have control windows associated with them. The blocks having a control window are listed below and the use of the windows are explained.

1. AVERAGE BLOCK : The control window of the average block is used to select between the expanding or the shrinking average. To open a control window for any block the method is exactly the same which requires the clicking of the right mouse button when the mouse cursor is on the block.

2. CONVOL : This block has two control windows. The first one is used to enter the row and column number of the convolution kernel. The second window is for entering the entries of the convolution kernel. The second window is opened when the user enters the size of the kernel and presses the ACCEPT button in the first window.

3. THRESHOLD : The threshold value is selected by using this control window for the related instance. The control window consists of a slider and by using it, a threshold value between 0 and 255 can be selected.

4. MEDIAN : The control window of the median block is used to select the size of the window.

5. AND/OR : BY using the control window of this block it is possible to choose one of the 5 frame processes. The control window contains five selection items and they are AND, OR, XOR, PLUS, and, MINUS.

6. POINT OP : The control window of this block is used to enter the value of contrast and intensity to be used in the enhancement process of the related image.

7. LOAD : The control window of the load block is used to enter the name of the file that stores an image.

8. SAVE : By using this control window it is possible to enter the name of a disk file for saving an obtained image.

There can be only one control window open on the canvas. That is, trying to open the control window of a block while there is already an open one causes the existing one to be closed.

To clarify the process of constructing a schematics an example is carried out below.

In this example assume that a schematics to find and then display the AND of two images is to be constructed. The schematics can be constructed particularly in the following manner.

1. Place the first load block : To find the AND of two images it is necessary to load two images into the memory. Therefore assume a load block is to be placed first. To do so first the LOAD block from the BLOCKS menu should be selected and then PUT BLOCK ON CANVAS operation from the opereration menu should be chosen. When these are done by pressing the left mouse button and keeping it pressed while dragging the mouse the block is positioned on the canvas. When the place of the block is decided upon, the left mouse button should be released.

2. Place the second load block : Now the second load block can be placed on the canvas without making any selection from the operations or the blocks menu.

3. Place the AND/OR block : To do that the AND/OR block from the blocks menu has to be selected and the process of placing a block should be repeated.

4. Place the display block : The display block can be placed after selecting it from the blocks menu.

5. Connect the load blocks to AND/OR block : To achieve this first CONNECT BLOCKS operation should be selected. After this press the left mouse button when the mouse cursor is on one of the load instances and drag the mouse to the AND/OR block while the left button is presed. When the mouse cursor is on the AND/OR block release the left button. To connect the other load block to the AND/OR block repeat the above procedure.

6. Connect the AND/OR block to the display block : The connection procedure is similar to above.

A.6.5 Executing a Schematics

After the construction of a schematics and adjusting the values of the attributes of the instances the schematics can be executed. To do this the only necessary thing to do that is to press the EXECUTE button in the control panel of the main window.

As soon as EXECUTE button is pressed the system starts to interpret the constructed schematics. As the blocks are executed they are displayed in reverse video as a feed back for the user. When the schematics is executed an error number is displayed. If its value is zero then the schematics has been executed successfully otherwise there is an error and it should be coorected before reexecuting the schematics.

It should be noted that for a schematics to be executed successfully it should contain no circular references(chapter 4).

After executing the schematics the resultant images can be displayed on the canvas. For doing this the right mouse button should be clicked on one of the display instances.

When the schematics is to be reexecuted possibly after some modification it is important to reset the execution flags of the instances by selecting the RESET EXECUTION FLAGS operation from the operations menu or by pressing the function key (F9). Otherwise the execution stops immediately without any processing.

A.6.6 An Example of Using the Indirect Mode

In this part lets construct a schematics to perform the jobs that were done in the direct mode. To achieve this goal the only necessary thing is to construct the schematics shown in figure A.17.

As shown in figure A.17 first the image *orange* from disk is loaded and displayed. Then the threshold operation with a threshold value of 120 is applied and the resultant image is also displayed. Finally by using two average blocks the orange is enlarged and shrunk by a factor of four. These resultant images are also displayed.

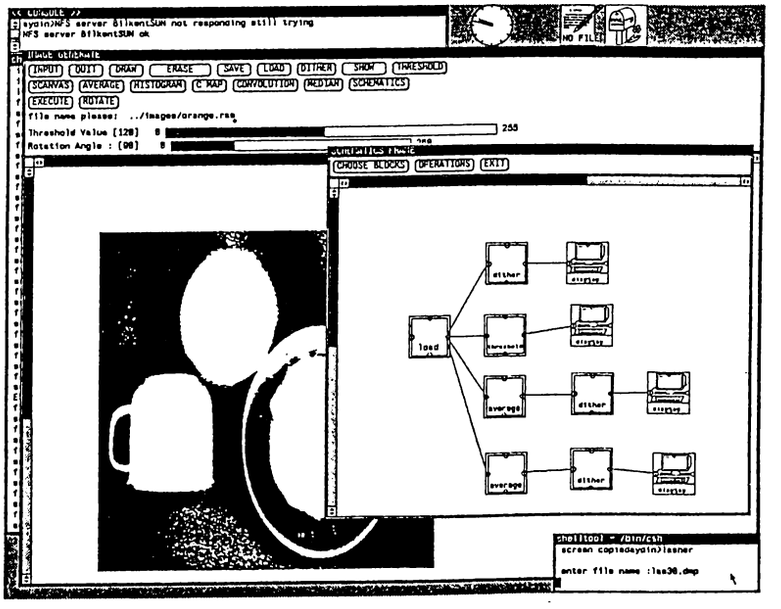

Figure A.17: An example of using the system in the indirect mode.

B. ATTRIBUTE VALUES OF THE OBJECT INSTANCES

```
inst_no         = 1
obj_id          = 11
execution_flag = 0
upper_left      = {
        point_x = 76
        point_y = 132
}
pin_pos_next[*]  = no value
pin_pos_pre [*]  = no value
next_obj        = ((nil), (nil), (nil), (nil), (nil))
pre_obj         = ((nil), (nil), (nil), (nil), (nil))
next            = (nil)
```

Figure B.1: The first object instance.

```
instance        = 1
colormap[*][*]  = gray ramp
file_name       = NULL
pr_input        = (Pixrect *)0
next_load       = NULL
```

Figure B.2: The first load instance.

```
object in adress: 0x12f498

      inst_no         = 1
      obj_id          = 11
      execution_flag = 0
      upper_left      = {
              point_x = 76
              point_y = 132
      }
      pin_pos_next[*] = no value
      pin_pos_pre [*] = no value
      next_obj        = ((nil), (nil), (nil), (nil), (nil))
      pre_obj         = ((nil), (nil), (nil), (nil), (nil))
      next            = 0x12f664

object in adress : 0x12f664

      inst_no         = 2
      obj_id          = 11
      execution_flag = 0
      upper_left      = {
              point_x = 85
              point_y = 246
      }
      pin_pos_next[*] = no value
      pin_pos_pre [*] = no value
      next_obj        = ((nil), (nil), (nil), (nil), (nil))
      pre_obj         = ((nil), (nil), (nil), (nil), (nil))
      next            = (nil)
```

Figure B.3: The instance values after placing the second load

```
object in address : 0x12f498

      inst_no        = 1
      obj_id         = 11
      execution_flag = 0
      upper_left     = {
              point_x = 76
              point_y = 132
      }
      pin_pos_next[*] = ({point_x = 132
                          point_y = 160}
                          others undefined )
      pin_pos_pre [*] = undefined
      next_obj       = (0x12f6f8, (nil), (nil), (nil), (nil))
      pre_obj        = ((nil), (nil), (nil), (nil), (nil))
      next           = 0x12f664
```

Figure B.4: The final attribute values of the object instance 1

```
object in address 0x12f664

      inst_no        = 2
      obj_id         = 11
      execution_flag = 0
      upper_left     = {
              point_x = 85
              point_y = 246
      }
      pin_pos_next[*]  = ( {point_x = 145
                            point_y = 276}
                            others undefined )
      pin_pos_pre [*]  = undefined
      next_obj       = (0x12f6f8, (nil), (nil), (nil), (nil))
      pre_obj        = ((nil), (nil), (nil), (nil), (nil))
      next           = 0x12f6f8
```

Figure B.5: The final attribute values of the object instance 2

```
object in address 0x12f6f8

      inst_no         = 1
      obj_id          = 7
      execution_flag = 0
      upper_left      = {
              point_x = 211
              point_y = 200
      }
      pin_pos_next[*] = ( { point_x = 269
                            point_y = 228}
                          others undefined )
      pin_pos_pre [*] = ( {point_x = 216
                           point_y = 234}
                          {point_x = 213
                           point_y = 232}
                          others undefined )
      next_obj        = (0x12f78c, (nil), (nil), (nil), (nil))
      pre_obj         = (0x12f498, 0x12f664, (nil), (nil),
                        (nil))
      next            = 0x12f78c
```

Figure B.6: The final attribute values of the object instance 3

```
object in address 0x12f78c

     inst_no         = 1
     obj_id          = 9
     execution_flag = 0
     upper_left      = {
             point_x = 371
             point_y = 191
     }
     pin_pos_next[*] = unfdefined
     pin_pos_pre [*] = ({point_x = 376
                         point_y = 224}
                         others undefined )
     next_obj        = ((nil), (nil), (nil), (nil), (nil))
     pre_obj         = (0x12f6f8, (nil), (nil), (nil),
                       (nil))
     next            = (nil)
```

Figure B.7: The final attribute values of the object instance 4

C. THE DATA STRUCTURES

```
struct functional_object {
       int                    inst_no;
       int                    obj_id;
       int                    execution_flag;
       struct block_pos       upper_left;
       struct block_pos       pin_pos_next[5];
       struct block_pos       pin_pos_pre[5];
       struct object_list     *next_obj[5];
       struct object_list     *pre_obj[5];
       struct object_list     *next;
} *obj_head, *curr_obj;
```

Figure C.1: The data structure of the functional object

```
struct average_block {
        int                     instance;
        u_char                  colormap[3][256];
        int                     opr;
        Pixrect                 *pr_input;
        struct average_block    *next_av;
} *av_head, *curr_av;
```

Figure C.2: The data structure of the average block.

```
struct histogram_block {
        int                     instance;
        u_char                  colormap[3][256];
        Pixrect                 *pr_input;
        int                     histogram_array[256];
        struct histogram_block  *next_histo;
} *histo_head, *curr_histo;
```

Figure C.3: The data structure of the histogram block.

```
struct convol_block {
        int                     instance;
        u_char                  colormap[3][256];
        Pixrect                 *pr_input;
        int                     row_num;
        int                     col_num;
        int                     *convol_mat;
        struct convol_block     *next_convol;
} *convol_head, *curr_convol;
```

Figure C.4: The data structure of the convolution block.

```
struct dither_block {
        int                         instance;
        u_char                      colormap[3][256];
        Pixrect                     *pr_input;
        int                         row_num;
        int                         col_num;
        float                       dither_mat[16][16];
        struct dither_block         *next_dither;
} *dither_head, *curr_dither;
```

Figure C.5: The data structure of the dither block.

```
struct threshold_block {
        int                         instance;
        u_char                      colormap[3][256];
        Pixrect                     *pr_input;
        int                         threshold_value;
        struct threshold_block      *next_thres;
} *thres_head, *curr_thres;
```

Figure C.6: The data structure of the threshold block

```
struct median_block {
        int                         instance;
        u_char                      colormap[3][256];
        Pixrect                     *pr_input;
        int                         row_num;
        int                         col_num;
        struct median_block         *next_med;
} *med_head, *curr_med;
```

Figure C.7: The data structure of the median block

```
struct and_or_block {
        int                         instance;
        u_char                      colormap[3][256];
       Pixrect                      *pr_input1;
       Pixrect                      *pr_input2;
       int                          op_id;
       struct and_or_block          *next_and_or;
} *and_or_head, *curr_and_or;
```

Figure C.8: The data structure of the AND/OR block

```
struct point_op_block {
        int                         instance;
        u_char                      colormap[3][256];
       Pixrect                      *pr_input;
       float                        intensity;
       float                        contrast;
       void                         (*point_ops)();
       struct point_op_block        *next_point;
} *point_head, *curr_point;
```

Figure C.9: The data structure of the point operations block

```
struct display_block {
      int                          instance;
      u_char                       colormap[3][256];
      Pixrect                      *pr_input;
      struct display_block         *next_disp;
} *disp_head, *curr_disp;
```

Figure C.10: The data structure of the display block

```
struct plus_minus_block {
        int                         instance;
        u_char                      colormap[3][256];
        Pixrect                     *pr_input1;
        Pixrect                     *pr_input2;
        int                         op_id;
        struct plus_minus_block     *next_plus;
} *plus_head, *curr_plus;
```

Figure C.11: The data structure of the plus minus block

```
struct load_block {
        int                         instance;
        u_char                      colormap[3][256];
        char                        file_name[80];
        Pixrect                     *pr_input;
        struct load_block           *next_load;
} *load_head, *curr_load;
```

Figure C.12: The data structure of the load block

```
struct save_block {
        int                         instance;
        u_char                      colormap[3][256];
        char                        file_name[80];
        Pixrect                     *pr_input;
        struct save_block           *next_save;
} *save_head, *curr_save;
```

Figure C.13: The data structure of the save block