

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

Graphical User Interface
for
Image Processing

By
Kumar Chebrolu

**Thesis submitted to the Faculty of the Graduate School of the
New Jersey Institute of Technology in partial fulfillment of the requirements
for the degree of Master of Science in Computer and Information Sciences
1991**

APPROVAL SHEET

Title of Thesis: GRAPHICAL USER INTERFACE FOR
IMAGE PROCESSING USING MOTIF TOOLKIT

Name of Candidate: Kumar Chebrolu
M.S. in Computer & Information Sciences, 1991

Thesis and Abstract
Approved:

Dr. Frank Shih
Assistant Professor
Department of Computer &
Information Sciences

Date

VITA

Name: Kumar Chebrolu
Permanent Address: 17-85/C, Srinagar Colony, Hyderabad, India - 500660
Degree and date to be conferred: M.S.C.I.S. December 1991

Collegiate institutions attended	Date	Degree	Date of Degree
New Jersey Institute of Technology	Sept. 1989- Dec. 1991	M.S.C.I.S.	Dec. 1991
Nagarjuna University, India	Sept. 1984 - Jun. 1988	B. Tech	Jun. 1988

ABSTRACT

Title of Thesis: **GRAPHICAL USER INTERFACE FOR
IMAGE PROCESSING USING MOTIF TOOLKIT**

Name of Candidate: Kumar Chebrolu
Master of Science in Computer and Information Sciences,
New Jersey Institute of Technology, Newark, NJ.
1991

Thesis Directed by: Dr. Frank Shih
Assistant Professor,
Computer and Information Sciences Department,
New Jersey Institute of Technology, Newark, NJ.

A user friendly, menu driven, highly interactive X Windows package for Image Processing Applications using Motif Widget Set under Motif Window Manager is developed. Modules related to Segmentation, Enhancement, Representation, Transformations are developed. The above routines are useful for image manipulation. The current gray scale/binary image is displayed on the window. On line histogram is provided so that the user can change the threshold value interactively. The OSF/Motif toolkit is used efficiently and also Xlib calls to display the image by allocating colormap. An on-line image manipulation help menu facility is incorporated in the tool to make it more versatile.

My Grandfather

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Frank Shih, for his remarkable guidance throughout the course of this work.

I am thankful to all my friends and family members for supporting and encourage me to complete my work successfully.

CONTENTS

Chapter 1 X WINDOW SYSTEM

1.1 The Server and Client.....	2
1.2 Window Manager.....	3
1.2.1 The Role of Window Manager in Motif Toolkit.....	4
1.3 Motif Toolkit.....	5
1.3.1 Intializing the Toolkit.....	8
1.3.2 Role of Xlib in Image Processing Toolkit.....	10
1.3.3 Role of Motif Toolkit in Image Processing Toolkit.....	13
1.3.4 Motif functions and macros used.....	15
1.3.5 Xt functions and macros used.....	16
1.3.6 Callback Procedures used.....	17

Chapter 2 THE REPRESENTATION SCHEMES

2.1 Rows Representation.....	19
2.1.1 Run Length Method.....	19
2.2 Block Representation.....	20
2.2.1 Quad Tree Method.....	20

Chapter 3 PIXEL CLASSIFICATION

3.1 Gray Level Thresholding.....21

 3.1.1 Automatic Thresholding.....21

 3.1.2 Interactive Thresholding.....22

 3.1.3 Histogram Equalization.....22

Chapter 4 EDGE DETECTION

4.1. Laplacian Operator.....24

4.2 Prewitt Operator.....25

4.3 Sobel Operator.....26

Chapter 5 IMAGE TRANSFORMS

5.1 Fast Fourier Transformation Definition.....28

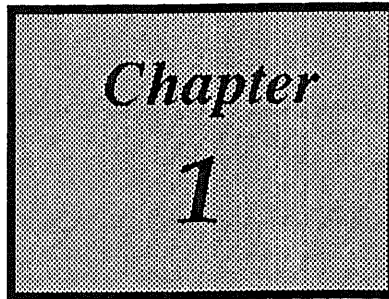
5.2 Rotation.....29

Chapter 6 GUI FOR IMAGE PROCESSING.....30

Coding

Figures

Bibliography

A rectangular box with a halftone background. Inside the box, the word "Chapter" is written in a serif font at the top, and the number "1" is written in a large, bold serif font below it.

Chapter
1

X WINDOW SYSTEM

The purpose of the X Window System is to provide a *network-transparent* and *vendor-independent* operating environment for *workstation software*. It offers a rich and complex environment to the programmer and use of application software. Network transparency means that X applications running on one cpu can show their output using a display connected to either the same cpu, or some other cpu. To the user, an X workstation looks like it is connected to many different host cpus at the same time. X applications are portable. The applications deal with X, so they do not know the details of any particular workstation's display hardware, as long as an X application is able to establish a connection to a workstation. Because the workstation hardware is *hidden* by the protocol, an X application running on a cpu from one vendor can use any workstation model, either from that vendor or from another. It is not necessary to recompile or relink any application to give it access to all kinds of X workstations.

The X Window System allows several applications at a time to be active on a workstation. Individual applications may start and stop at will during a user's session.

1.1. The Server and Client:

To allow applications to be run on one machine and display on another, X was designed as a network protocol - a predefined set of requests and replies - between two *processes*, one of which is an application program called a *client*, and the other of which the *server*, controls the display hardware, keyboard, and pointer. The user sits at the machine running the server. At first, this usage of the term server may seem a little odd, since file and print servers are normally remote machines. But the usage is consistent. The local display is accessible to other systems across the network, and for those systems the X server does act like other types of server.

The X server acts between user programs and the resources of the local system such as the keyboard and screen. It contains all device-specific code, and insulates applications from differences between display hardware. The server allows access to the display by multiple clients. It interprets network messages from clients and acts on them. Some requests command the server to do drawing, while others ask the server for information. Protocol requests are generated by client calls to *xlib*, *xt* and *motif toolkits*. Server passes user input to clients by sending network messages known as events, which represent key or button presses, pointer motion, and so forth. In X, the display is often used as a synonym for server. A screen is the actual hardware on which the graphics are drawn. A server may control more than one screen.

The application programs displaying on the screen(s) managed by a server are called its *clients*. There may be several clients connected to a single server. Clients may run on the same machine as the server if that machine supports multitasking, or clients may run on other machines in the network. In either case, X protocol is

used by the client to send requests to draw graphics or to query the server for information, and is used by the server to send user input and replies to information requests back to the client. The X protocol runs on top of any low-level network protocol that provides bidirectional communication, and delivers bytes unduplicated and in sequence. TCP/IP and DECnet are the currently-supported networks. The communication between a client and the server is called a connection. It is common for a user to have applications running on several different hosts in the network, all invoked from and displaying their windows on a single screen.

This use of the network is known as *distributed processing*. The most important application of this concept is to provide graphic output for powerful systems that don't have their own built-in graphics facilities. However, distributed processing can also help solve the problem of unbalanced system loads. When one host machine is overloaded, the users running clients on that machine can arrange for some of their programs to run on other hosts.

1.2.Window Manager :

A Window Manager allows the user to control the size and location of windows on the screen. In X, a window manager is an ordinary client application. It manages the positions and sizes of the main windows of applications on a server's display. The responsibility of the window manager is to mediate competing demands for the physical resources of a display, including screen space, color resources, and the keyboard. The Window Manager allows the user to move windows around on the screen, resize them, and usually start new applications. The Window manager also defines much of the visible behavior of the window system, such as whether windows are allowed to overlap or are tiled (side by side), and whether the

keyboard focus simply follows the pointer from window to window, or whether the user must click a pointer button in a window to change the keyboard focus.

1.2.1. The Role of Window Manager in Motif Toolkit :

In window system a top-level window resides at the top of the window (and widget) tree hierarchy for the application. Its parent is the root window, which is what the user perceives as the background behind all the windows on the desktop.

But in the Xt based toolkits, behind every visible top-level application window is a special kind of widget known as a shell widget. Every window that can be placed independently on the screen, including top-level windows and dialog boxes, has its parent an invisible shell widget.

The Motif function `XmIsMotifWMRunning` checks the `_MOTIF_WM_INFO` property to determine whether the Motif Window Manager is running on the screen containing the specified shell.

Typically, shell widgets contain only one managed child widget whose job is manage the layout of more primitive components such as Labels, Text widgets, Scrollbars, and Push Buttons. These are the items that the user actually sees and interacts with on the screen. These objects are really descendants of the shell widget because they are contained within its boundaries. One of the shell's main jobs is to communicate with the window manager on behalf of the application. The window manager frame is made up of window decorations that the window manager places on all toplevel windows (which are the windows associated with shell widgets). The controls allow the user to interactively move windows, resize them, cause them to redraw themselves, or even to close them down. The window

menu displays a list of window manager functions that allow the user to move, resize or even exit the application.

Motif provides window manager protocols that allow menu items like these to affect the application. The user can manipulate the window manager's window menu by using many of the same types of protocols. Also the user can specify which of the items in the window menu user want to appear, whether there are resize handles on the frame or whether the user want to allow the user to iconify the window.

1.3. Motif Toolkit :

X windows code runs on Unix machines. Unix can be considered the "bottom level" of code. On top of Unix run some system libraries (like `stdio.h`, `math.h`, etc.). On top of that runs the X windows system, which is made up of the X windows libraries, called *Xlib*. On top of that runs the "*Xt intrinsics*", which is another group of C library routines. Finally, on top of all of this is called the *Motif widget set*. Motif tries to be an *object oriented* set of routines that allows a programmer to build an X windows user interface in a finite amount of time. Then on top of the pile the user makes application program.

The Motif toolkit is based on the X Toolkit Intrinsics (*Xt*), which is the standard mechanism on which many of the toolkits written for the X Window System are based. *Xt* provides a library of user-interface objects call widgets and gadgets, which provide a convenient interface for creating and manipulating X windows, colormaps, events, and other attributes of the display. The widgets that *Xt* provides are generic in nature and impose no user-interface policy whatsoever. That is the job of a user-interface toolkit such as Motif.

Xt provides an *object-oriented programming* style, where the objects are widgets. Traditionally, object-oriented programming is defined in terms of *object*, *method*, *message*, *class*, and *instance*, and the *concept of encapsulation*.

An object contains two elements : the *data* that represents a state, and *code* that reads or writes that data. Inside the widget code, the state data is represented as structure members, and the methods are represented as pointers to functions. Some state data members are *public*; they are resources that can be set or retrieved from outside the object. Other state data members are *private*; they cannot be read or written from outside.

A method in object oriented programming is either a method or an action in Xt, methods are a set of functions that are fixed for a particular class, triggered in fixed ways in response to Xt function call made by the application. A widget's methods supply its most basic functions, such as the code needed to create a widget or to change its resources. Actions, on the other hand, are called in response to the events specified in a translation table, and are thus user configurable.

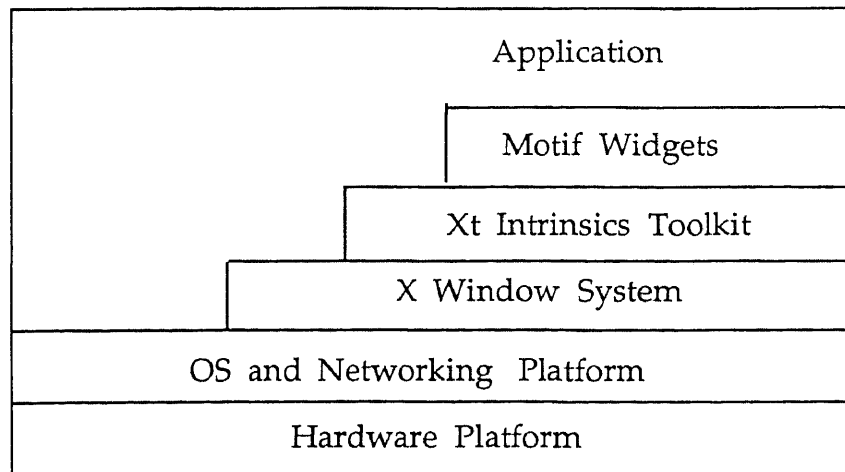
Inputs to objects and communication between them are called messages. The forms of communication are function calls, events, actions, and callbacks.

Widget library defines classes of widgets. Each class has unique characteristics shared with all other members of that class, but distinct from other classes. Each time you create a widget, its creates an instance of one of these hypothetical application. Widget characteristics can be inherited from other, more basic classes of widgets. Inheritance means that a new class of widget has to define

only its own unique features, and need not re-implement those common to all widgets, or already implemented by an existing superclass. The class hierarchy of a particular widget class is fixed, while the instance hierarchy is set differently in every application.

A program that uses an object must not depend on the internal implementation of the object, but instead only on the known inputs and outputs. The advantages of code encapsulation are that programmers can use the object without needing to understand the internal representation (hiding details), and that the internal implementation of the object can be changed at any time because no other code depends on it. It minimizes interdependencies.

Motif is a library of routines that makes the programming of user interfaces in an X Windows environment fairly easy and straight forward. The Motif Libraries handle a lot of the low level X windows, so that the user can create good looking and sophisticated interfaces without having to contend with the all of the complexity of X. But because Motif is built on top of Xwindows, and because Xwindows is a detailed and complicated environment, Motif is for many an intimidating, difficult to learn programming system.



1.3.1. Initializing the Toolkit :

Open the application's connection to the X display.

Parse the command line for any of a dozen or so standard x Toolkit command line options plus any custom command line options the user define in the application program.

Load the resource database from the app-defaults file.

Create the application's top-level window, a Shell Class widget that will handle all of the application's interaction with the Motif Window Manager, mwm, and act as the parent of all the other widgets in the application.

```
1.      XtAppInitialize (  
2.          &app,  
3.          "Tst",  
4.          NULL,      0,  
5.          &argc,     argv,  
6.          NULL, NULL  
          );
```

(1). The Widget returned by (1) is a *Shell Widget*. Shell widgets handle the application's interaction with the window manager, and act as the toplevel window of the application. All other widgets created by the application are created as children of the shell.

(2). *The Application Context.* The first argument (2) is the *address* of an application context, a structure in which Xt will manage some data internal to Xt that is associated with the application.

(3). *The Application Class.* The second argument (3) is the *class name* of the application. A class name is used in the *resource database* to specify values that will apply to all *instances* of an application, a widget, or a resource.

(4). *Command-line Arguments.* The third and fourth (4) arguments specify an *array of command-line arguments* defined for application program, if any and the number of arguments in the array. When these arguments are unused, they are specified as NULL and 0, respectively. The fifth and sixth (5) arguments contain the *value* and *count* of any actual command-line arguments.

(5). *Fallback Resources.* The seventh (5) argument is the start of a NULL-terminated list of *fallback resources* for the toplevel shell widget created by the initialization call.

(6). *Additional Initialization Parameters.* The eighth (6) parameter is the first of a NULL-terminated *list of resource-value pairs* that are applied to the toplevel widget return by (1) XtAppInitialize. If there are none NULL is passed as the eighth parameter.

1.3.2. Role of Xlib in Image Processing Toolkit:

Graphics Context:

The Graphics Context is a data structure that contains information about the attributes that determine the width of lines, foreground and background colors, fill patterns, fonts to be used when displaying text, and so on. X stores these attributes in an data structure known as graphics context, abbreviated as GC.

The Xlib function *XCreateGC* creates a graphics context and returns a resource identifier that applications can use to refer to the GC. The X server maintains the data associated with the graphics context, and all clients must reference the graphics context by its own ID. GC's are associated with a specific drawable, but can be used with any drawable of the same depth on a screen with the same visual type.

Screen :

The *XDefaultScreen* function return a workstation's default screen, given a Display pointer. This default screen comes from the default screen number in the display name, if any.

Visual :

Every screen has a default visual structure, which contains information about its color capabilities. Pointer is obtained to a screen's default Visual structure with the *DefaultVisual* macro. PseudoColor - In this visual class, each pixel value indexes a red-green-blue color map. The contents of the color map may be changed dynamically. Each pixel is treated as a single index into a single color map array. Each entry in the color map contains a red-green-blue triplet.

Depth :

The depth of a window is the number of planes, or the number of bits in pixel values for that window. The depth in the default root window for a specified screen can be obtained from the *DefaultDepth* macro. The depth is greater than one for the *PseudoColor* workstations.

Colormap :

Color maps sometimes known as color lookup tables convert pixel values into colors on the screen. The exact structure of a color map, and of the pixel values which access it, depends on the visual class of the color map. A color map is an array of color cells. Each color cell in a color map contains one combination of red, green, and blue primary color values, which specifies one color in the gamut of the workstation's screen. XColor structure describes the contents of each color cell. Default Color map for a screen can be determined with the *DefaultColormap* macro.

XAllocColor :

The XAllocColor requests support the shared color cell strategy. The shared color cell strategy is often used in applications that display images, they may use shared color cells for such things as menus at the same time as they use the preallocated pixel values in a standard shared color map for displaying the image itself. *XAllocColor* return a single pixel value that user can use to display the color.

XQueryColors:

XQueryColors requests to obtain the primary color component values for specified color cells. Before calling XQueryColors the user should initialize the pixel fields in each element of the colors array.

Images :

Xlib provides support for *image manipulation*. Because the X display connection between the application and the workstation is a network link, images are hard to handle in the X environment. Images typically contain large amount of data. For a 512x512 pixel image with eight bits per pixel consumes about a quarter megabyte. *Xlib* takes a substantial amount of time to convert such an image to protocol wireformat and transmit it over a display connection. The application's CPU and the workstation's CPU can be different makes and models. Different computer models often encode the bits within pixel values in different orders. To reduce the effect of the two above mentioned problems on application programs, *Xlib* chose to represent images inside application programs using data structures known as *XImage* structures.

XCreateImage :

XCreateImage is a utility function used for allocating memory for an *XImage* structure and initializing the structure. *XCreateImage* allocates memory for the *XImage* structure, but not the image itself. This function returns a pointer to the *XImage* structure. When the user finishes the using an *XImage* structure created by *XCreateImage*, *XFree* is used to deallocate the *XImage* Structure.

XPutImage :

XPutImage is a graphic primitive request for sending images to the workstation. It uses a GC to set up the graphics pipeline. *XPutImage* draws a rectangular area of the image into the specified drawable using the attributes in the specified gc.

XClearWindow :

XClearWindow clears the contents of the window prior to drawing a new picture in the specified window.

1.3.3. Role of Motif Widget Set in Image Processing Toolkit :

Widget :

Widget is a basic object in a toolkit. A widget includes both *code* and *data*, and can therefore serve as an input or output object. Widgets consist of an X Window along with some procedures that operate on the window. Examples of widgets include manager widgets like rowcolumn, form and pushbuttons, scrollbars, menus, and dialog boxes.

A structure returned by Motif Toolkit routines to identify the widget on which the routine operates. The members of this structure should not be accessed directly from applications; they should regard it as an opaque pointer. *Widget* is really a pointer to a widget instance structure. Widget code accesses *instance* variables from this structure.

FormWidgetClass :

A *container* widget that constrains its children so as to define their layout and behavior when the Form is resized. Children may be attached to each other, to edges of the Form, or to absolute or relative positions in the Form. Form is a type of manager widget type subclassed from the Bulletin Board class.

RowColumnWidgetClass :

The RowColumn Widget is a general purpose manager widget capable of containing any widget type as a child. The RowColumn Widget lays out its children in row and/or column format. It has control over the spacing that occurs between each row or column and the spacing between the edges. The Motif Toolkit uses the Row Column widget to implement many convenience routines internally like popup shell, MenuBars, and pulldown menus.

DrawingAreaWidget :

The DrawingArea Widget provides a blank canvas for interactive drawing using basic Xlib drawing primitives. The widget does no drawing of its own, not does it define or support any Motif user interface design. Subclassed from the Manager Widget class, the DrawingArea widget may also contain other children, although there is frequently no assumed or regimented layout policy. DrawingArea inherits certain *translation* and *action* tables that pass event to gadget children.

1.3.4. Motif Functions and Macros used in the Program :

The following functions returns an instance of a respective widget, returning its widget ID.

XmCreateCascadeButton
XmCreateCascadeButtonGadget
XmCreateDialogShell
XmCreateDrawingArea
XmCreateFileSelectionBox
XmCreateForm
XmCreateLabel
XmCreateMenuBar
XmCreateMenuShell
XmCreateMessageBox
XmCreateMessageDialog
XmCreatePopupMenu
XmCreatePromptDialog
XmCreatePulldownMenu
XmCreatePushButton
XmCreatePushButtonGadget
XmCreateRowColumn
XmCreateScale
XmCreateSelectionDialog
XmCreateText
XmMessageBoxGetChild
XmCreateCreate
XmStringGetLtoR

XmStringFree

XmStringGetLtoR

1.3.5. Xt functions and macros used in the program :

XtAddCallback - A callback contains the information about the callback routine associated with a particular action.

XtRealizeWidget - It displays on the screen the widget that is passed to it and the children of that widget.

XtAppMainLoop - The application passes control to the Xt Intrinsics and the Motif Widgets on the *XtAppMainLoop* function is called.

XtGetValues - It will return the current value of specified arguments for a created widget.

XtSetValues - It will change the value of the specified arguments.

XtSetArg - The simplest way to set an element of an argument list is by using this *XtSetArg* macro.

XtWindow - It returns the window of the widget.

XtDisplay - It returns the pointer to the Display of the specified widget.

XtPopup - When the user wants to map a popup shell to the screen, *XtPopu* should

be used.

XtPopdown - When the dialog is to be dismissed, *XtPopdown* is used.

XtManageChild - It happens to pop up the shell and *XtUnmanageChild* causes it to popdown.

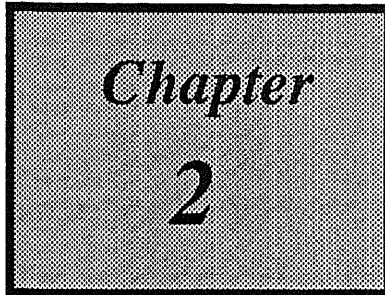
XtParent - It returns the parent of the specified widget.

1.3.6. Callback Procedures used in the Program :

XmAnyCallbackStruct

XmScaleCallbackStruct

XmSelectionBoxCallbackStruct

A rectangular box with a halftone background. Inside the box, the word "Chapter" is written in a serif font at the top, and the number "2" is written in a large, bold serif font below it.

Chapter
2

REPRESENTATION
SCHEMES

A subset S of a digital picture Σ can be represented by a binary picture χ_s of the same size as Σ , having 1's at the points of S and 0's elsewhere. If Σ is $n \times n$, the χ_s representation requires n^2 bits. χ_s can be regarded as the characteristic function of the subset S ; this is the function that maps points of S into 1 and points of \bar{S} into 0.

More generally, any partition of Σ into S_1, \dots, S_m can be represented by an m -valued picture having i 's at the points of S_i , $1 \leq i \leq m$. In particular, if Σ is any picture, then in this sense, Σ represents its own partition into sets of constant gray level, i.e., S_i is the set of points of Σ having gray level i . For an $n \times n$ picture, this representation requires $n^2 \log_2 m$ bits.

The storage requirements of this *trivial* representation are the same for all partitions of Σ into a given number of sets.

2.1. ROWS REPRESENTATION :

2.1.1. Run Length Method :

Each row of a picture consists of a sequence of maximal runs of points such that the points in each run all have the same value. Thus the row is completely determined by specifying the lengths and values of these runs. If there are only a few runs, this representation is very economical; for this reason, run length coding

is sometimes used for picture compression. Suppose that the row has length n , and there are r runs. Since it takes $\log_2 n$ bits to specify the length of a run, the number of bits needed to specify all the run lengths is $r \log_2 n$. Thus if there are m possible values, this representation of the row requires $r(\log_2 n + \log_2 m)$ bits, as compared with the $n \log_2 m$ bits that are required when the row is treated as a string of length n .

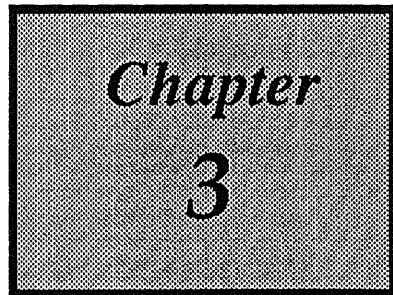
For an $n \times n$ picture, if the average number of runs in each row is r , the total number of bits required by the run length representation is $n(1 + r \log_2 n)$ as compared with n^2 in the binary case, or $nr(\log_2 m + \log_2 n)$ as compared with $n^2 \log_2 m$ in the general case.

2.2. BLOCK REPRESENTATION:

2.2.1. Quadtrees :

Maximal blocks can be of any size and in any position, they are analogous to runs in the one-dimensional case. Assume the size of the image is Σ is $2^k \times 2^k$. The root node of the tree represents the entire image. If the image has all one value, label the root node with that value and stop. Otherwise, add four descendants to the root node, representing the four quadrants of the picture. The process is then repeated for each of these new nodes; and so on. In general, the nodes at level h (if any) represent blocks of size $2^{k-h} \times 2^{k-h}$, in positions whose coordinates are multiples of 2^{k-h} . If a block has constant value, its node is a leaf node; otherwise, its node has four descendants at level $h + 1$, corresponding to the four quadrants of the block. The nodes at level k , if any are all leaf nodes corresponding to single pixels.

The tree constructed in this way is called a *quadtree representation*, since its nonleaf nodes all have degree 4. The chief advantage of the quadtree representation is that, unlike the nontree representations considered here, it is *shift-variant*. Two pictures that differ only by a translation may give rise to very different quadtrees.

A rectangular box with a gray, textured background and a black border. Inside the box, the word "Chapter" is written in a serif font at the top, and the number "3" is written in a larger serif font below it.

Chapter
3

PIXEL CLASSIFICATION

Segmentation is basically a process of *pixel classification*;

The picture is segmented into subsets by assigning the individual pixels to classes. In an attempt to distinguish dark objects from their light background, we segment the image by *thresholding* its gray level; it means classifying the pixels into *dark* and *light* classes.

In *edge detection*, we classify pixels into *edge* and *not edge* by thresholding the response of some difference operator that has high values when the rate of change of gray level is high.

3.1.Gray Level Thresholding :

3.1.1. Automatic Thresholding :

In this paper, *automatic thresholding* was implemented. The gray level histogram of the picture should display peaks corresponding to the two gray level ranges. The picture can thus be segmented by choosing a threshold that separates two peaks.

A histogram having two peaks is called *bimodal*. The average of the two peaks are taken as the *threshold value*. The pixels whose gray levels are *darker* than the threshold are displayed as black, and those are *lighter* than the threshold as the respective gray level.

3.1.2. *Interactive Thresholding* :

The gray level histogram is displayed on the screen. The user can *interactively* give the threshold value and do the segmentation. Thus the pixels whose gray values are *less* than the threshold are considered as background and the gray values *greater* than the threshold value are considered as the respective gray value.

3.1.3. *Histogram Equalization* :

Image Enhancement by Histogram Equalization :

A histogram of gray-level content provides a global description of the *appearance* of an image. The type and degree of enhancement obtained depends on the nature of the specified histogram.

A transformation function equal to the cumulative distribution of pixels, produces an image whose gray levels have a uniform density. In terms of enhancement, this implies an increase in the dynamic range of pixels, can have a considerable effect in the appearance of an image.

The concepts developed for *Histogram Equalization* are formulated in discrete form. Let the variable r represent the gray level of the pixels in the image to be enhanced. The gray levels in an image are random quantities. Assuming the moment that

they are continuous variables, the original and transformed gray levels can be characterized by their probability density functions $p_r(r)$ and $p_s(s)$, respectively. For gray levels that assume discrete values, we deal with probabilities given by the relation

$$p_r(r_k) = n_k/n \qquad 0 \leq rk \leq 1$$

$$k = 0, 1, \dots, L - 1,$$

where L is the number of levels, $p_k(r_k)$ is the probability of the k th gray level, n_k is the number of times this level appears in the image, and n is the total number of pixels in the image. A plot of $p_r(r_k)$ versus rk is called a *histogram*, and the technique for obtaining a uniform histogram is known as *histogram equalization* or *histogram linearization*.

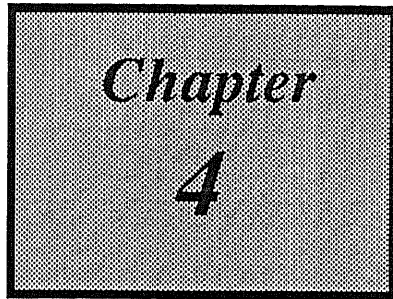
The discrete form is given by the relation

$$s_k = T(r_k) = k \sum_{j=0}^{k-1} n_j/n$$

$$= k \sum_{j=0}^{k-1} p_r(r_j) \quad 0 \leq rk \leq 1$$

$$k = 0, 1, \dots, L - 1.$$

Since a histogram is an approximation to a probability density function, perfectly flat results are seldom obtained when working with discrete level. For an image with the narrow range of values occupied by pixels, Histogram Equalization is as expected, not perfectly flat throughout the full range of gray levels, but considerable improvement over the original image can be achieved by the spreading effect of the Histogram Equalization technique.

A rectangular box with a halftone background. Inside the box, the word "Chapter" is written in a serif font at the top, and the number "4" is written in a large, bold serif font below it.

Chapter
4

EDGE DETECTION

4. EDGE DETECTION :

This section deals with local operations that can be used to detect various types of local features, such as edges in a image. The gray level is relatively consistent in each of two adjacent, extensive regions, and changes abruptly as the border between the regions is crossed.

4.1. Laplacian Operator :

The Laplacian Operator is an orientation-variant derivative operator. The analog of the Laplacian is given by

$$(\nabla^2 f(x, y) = [f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y)])$$

which is a digital convolution of f with

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

since the digital Laplacian is a second difference operator, it has zero response to linear ramps, but it responds to the shoulders at the top and bottom of a ramp, where there is a change in the rate of change of gray level.

It responds to each side of an edge once with positive sign and once with negative sign. If the user want only positive responses, use the absolute value, or positive value when it is greater than zero. These responses have values as high as four times the maximum gray level. To ensure that the gray level range is preserved it should be divided by 4. The digital Laplacian does not respond to edges, but it responds even more strongly to corners, lines, line ends, and isolated points. In a noisy image, the noise will produce higher Laplacian values than the edges, unless it has much lower contrast. When Laplacian Operator is applied to a image, low spatial frequencies are weakened, while higher ones remain relatively intact. The results of the images after applying Laplacian operator can be observed in the user interface.

4.2. Prewitt Operator :

The effects of noise on the responses of a difference operator can be reduced by smoothing the image before applying the operator. The image can be locally averaged before differencing, or equivalently an operator that computes the differences of local averages can be used.

A operator based on differences of averages responds blurrily to an edge in several positions. These responses can be sharpened by suppressing nonmaxima in the direction across the edge, i.e., setting a response to zero if there is a stronger response sufficiently close to it in that direction, on either side. Nonmaxima should not be suppressed in the direction along the edge, since the edge would

then compete with itself. The blurriness of responses to edges, can be reduced by averaging only in the direction along the edge. In this thesis, the Δ_{3x} and Δ_{3y} operators are used, since they gave values that are symmetric around (x, y) , where Δ_{3x} is defined as the convolution of f with

$$1/3 \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

and Δ_{3y} is defined as the convolution of f with

$$1/3 \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

The digital gradient values are obtained by using the maximum of the two perpendicular operators.

4.3. Sobel Operator :

The Laplacian and the Prewitt operators are based on unweighted averages. Sobel operator uses the weighted averages whose x and y components are given by the convolutions of f with

$$1/4 \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

and

$$1/4 \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

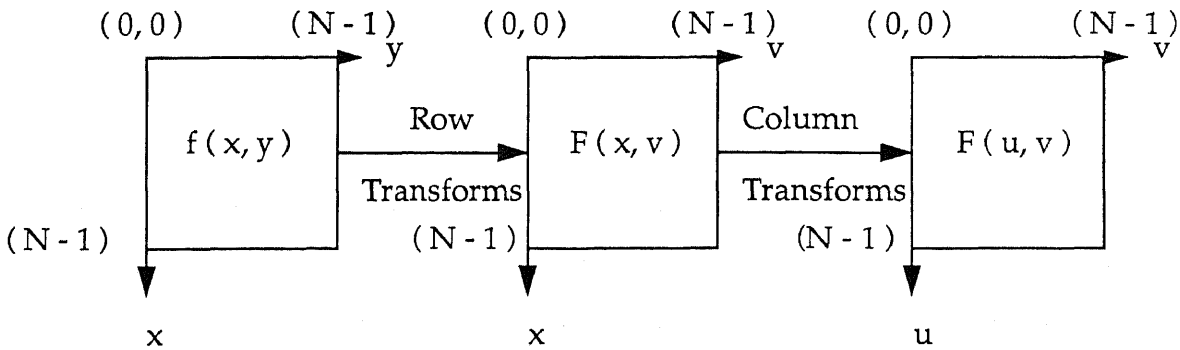
This operator gives greater weight to points lying closer to (x, y) . Therefore, its response to diagonal edges is not weakened as much as that of the Prewitt Operator. The values are obtained by using the maximum of the two above operators.

Chapter
5

**IMAGE
TRANSFORMS**

5.1. Fast Fourier Transformation:

Two-dimensional transforms are used in the following chapters for image enhancement, restoration, encoding and decoding. The development of a fast Fourier transform algorithm which can be used to reduce the number of calculations to a fraction of that required to implement the discrete Fourier transform was made. The number of complex multiplications and additions required are proportional to N^2 . The decomposition procedure is called fast Fourier transform algorithm, in which the proportionality is reduced from N^2 to $N \log_2 N$.



Computation of the two dimensional Fourier Transform
as a series of One-dimension transforms.

5.2. Rotation :

A geometrical transformation of the plane is defined by a pair of equations of the form

$$\begin{aligned}x' &= h_1(x, y), \\y' &= h_2(x, y); \end{aligned}$$

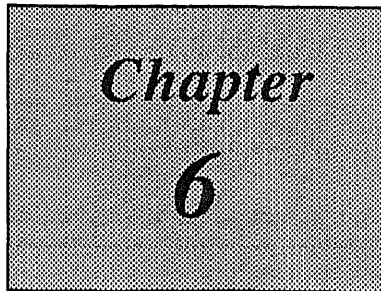
which specify the new coordinates of each point as functions of the old coordinates. It is nontrivial to apply such a transformation to a digital image, since the new coordinates are not necessarily integers. To make the results of the transformation into a digital image, they must be resampled or interpolated. In this thesis, three different types of Interpolation are used for rotation the image - *Biconstant Interpolation, Bilinear Interpolation, and Bicubic Interpolation.*

The simplest method is *bilinear interpolation*, which is defined as follows :

Let the integer parts $\lfloor x \rfloor, \lfloor y \rfloor$ of x'' and y'' be x and y , so that the point (x'', y'') , is surrounded by the four integer-coordinate points

$$\begin{array}{cc} (x, y+1) & (x+1, y+1) \\ & (x'', y'') \\ (x, y) & (x+1, y) \end{array}$$

In *bicubic interpolation*, in which the image is approximated by a linear combination of products of cubic polynomials, $\sum \sum c_{ij} g_i(x) g_j(y)$. The coefficients of these polynomials can be chosen so that the approximation has the same values at the sample points. Bicubic interpolation yields smoother results.

A rectangular box with a halftone background. Inside the box, the word "Chapter" is written in a serif font at the top, and the number "6" is written in a large, bold serif font below it.

USER INTERFACE TOOLKIT

The User Interface is developed in X Windows using Motif Toolkit. The operations one can do are listed below :

1. File :

The user can select any image file from any directory using *open* command. To change the directory, use the filter button. Then select the required file and press Ok button. The image file will be loaded.

Using *Save* command, it saves the current image on the original image file, i.e., the original image is overwritten.

Using *Saves* Command, the current image can be saved in to the new file. It will ask for the filename, filename is entered in the dialog and press Ok button.

The user can *print* the current image file to a postscript file "PRINT" and the file can be directed to any postscript printer.

The *undo* command goes to the previous command.

2. Transformations :

The image can be inverted using *invert* command i.e., the user can display the negative image.

The image can be enlarged twice to the original image by *zoomin* command.

The *Fourier* transformation can only be applied to the binary images.

The image can be *rotated* by any one of these methods - *Biconstant Interpolation*, *Bilinear Interpolation*, and *Bicubic Interpolation*. When the user selects the respective button, it will popup a dialog to enter the angle. The rotated image can be displayed on the window by pressing Ok.

3. Segmentation :

Pixel classification can be made by one of the commands - *Auto threshold*, *interactive threshold* or by *histogram equalization*.

Edges of the image can be detected by three operators - *Laplacian*, *Prewitt* and *Sobel*.

4. Representation :

Representation of the binary images can only be applied by *run length method* and *quad tree method*.

5. Help :

Online *Help* is provided for all functions listed in the menu.

Coding

```
/**-----**
```

Program : draw.c

Programmer : Kumar Chebrolu

This program sets the Graphics Context and Allocates colors in the Colormap
The displa function uses Xlib functions XCreateImage and XPutImage to
display the image on the drawing area.

```
***-----**/
```

```
#include "tst.h"
```

```
Visual *vis = NULL;  
unsigned long colors[256];  
XColor ctab[256];
```

```
/**-----**/
```

```
set_gc( d )  
Widget d;  
{  
    XGCValues values;  
  
    values.function = GXcopy;  
  
    gc = XCreateGC (  
        XtDisplay ( d ),  
        XtWindow ( d ),  
        GCFunction,  
        &values  
    );  
}
```

```
/**-----**/
```

```
set_v( dd )  
Widget dd;  
{  
    int i;  
  
    screen = DefaultScreen (  
        XtDisplay ( dd )  
    );  
  
    vis = DefaultVisual (  
        XtDisplay ( dd ),  
        screen  
    );  
  
    depth = DefaultDepth (  
        XtDisplay ( dd ),  
        screen
```

```
        );

    for ( i = 0; i < 256; i++ )
        ctab[i].pixel = i;

    colormap = DefaultColormap (
        XtDisplay ( dd ),
        screen
    );

    for ( i = 0; i < 256; i++ ) {
        allcolors[i].red = ( u_short ) ( i << 8 );
        allcolors[i].green = ( u_short ) ( i << 8 );
        allcolors[i].blue = ( u_short ) ( i << 8 );
        allcolors[i].flags = DoRed | DoGreen | DoBlue;
        if ( !XAllocColor (
            XtDisplay ( dd ),
            colormap, &allcolors[i] ) ) {
            allcolors[i].pixel = 0xffff;
        }
    }

    XQueryColors ( XtDisplay ( dd ),
        colormap,
        ctab, 256
    );

}

/***/-----***/

displa ( dr, IMAG )
Widget dr;
float *IMAG;
{
    int    i,
          j;

    set_gc ( dr );
    set_v ( dr );

    DIMAGE = ( unsigned char * ) calloc ( X * Y , sizeof ( unsigned char ) );
    ftod ( IMAG, DIMAGE );

    image = XCreateImage (
        XtDisplay ( dr ),
        vis, 8,
        ZPixmap, 0,
        (char * ) DIMAGE,
        X, Y,
        8,
        0
    );
}
```

```
image->byte_order = MSBFirst;  
image->bitmap_bit_order = MSBFirst;
```

```
XClearWindow (  
    XtDisplay ( dr ),  
    XtWindow ( dr )  
);
```

```
XClearWindow (  
    XtDisplay ( drawh ),  
    XtWindow ( drawh )  
);
```

```
XSetForeground (  
    XtDisplay ( dr ), gc,  
    WhitePixel ( XtDisplay ( dr ), screen )  
);
```

```
XPutImage (  
    XtDisplay ( dr ),  
    XtWindow ( dr ), gc,  
    image,  
    0, 0, 70, 50,  
    640, 640  
);
```

```
XBell ( XtDisplay ( dr ), 1000 );  
free ( DIMAGE );
```

```
}
```

```
/***/-----***/
```

```

/**-----***/

Program : edge.c

Programmer : Kumar Chebrolu

This Program is for detecting edges of the images.
The three operators used for detecting edges are Laplacian, Sobel and
Prewitt.

***/

#include "tst.h"

float lap[3][3] = {{0, 1, 0}, {1, -4, 1}, {0, 1, 0}};

float sob_h[3][3] = {{-1,0,1}, {-2,0,2}, {-1,0,1}};

float sob_v[3][3] = {{1,2,1}, {0,0,0}, {-1,-2,-1}};

float pre_h[3][3] = {{-1,0,1}, {-1,0,1}, {-1,0,1}};

float pre_v[3][3] = {{1,1,1}, {0,0,0}, {-1,-1,-1}};

/**-----***/

get_three_image(p,q,trans,orig)
int p, q;
float trans[3][3], *orig;
{
    if ((p==0)|| (q==0)) trans[0][0] = 0; else
        trans[0][0] = orig[(p-1)*X+(q-1)];
    if (p==0) trans[0][1] = 0; else
        trans[0][1] = orig[(p-1)*X+q];
    if ((p==0)|| (q+1==Y)) trans[0][2] = 0; else
        trans[0][2] = orig[(p-1)*X+(q+1)];
    if (q==0) trans[1][0] = 0; else
        trans[1][0] = orig[p*X+(q-1)];
        trans[1][1] = orig[p*X+q];
    if (q+1 == Y) trans[1][2] = 0; else
        trans[1][2] = orig[p*X+(q+1)];
    if ((q == 0)|| (p+1 == X)) trans[2][0] = 0; else
        trans[2][0] = orig[(p+1)*X+(q-1)];
    if (p+1 == X) trans[2][1] = 0; else
        trans[2][1] = orig[(p+1)*X+q];
    if ((p+1 == X) || (q+1 == Y)) trans[2][2] = 0; else
        trans[2][2] = orig[(p+1)*X+(q+1)];
}

/**-----***/

float
con(con_mat,orig_mat)

```

```
        free ( sob_image );
        displa ( draw, IMAGE );
    }

    /**-----***/

void
laplacian()
{
    float *lap_image,
          temp[3][3];

    int   a,
          b,
          dd;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    XClearWindow ( XtDisplay ( drawh ),
                  XtWindow ( drawh )
                  );

    lap_image = (float *)calloc( X*Y,sizeof(float));

    for (a = 0; a < Y; a++)
        for (b = 0; b < X; b++) {
            get_three_image(a,b,temp,IMAGE);
            lap_image[a*X+b] = (abs((int)(con(lap, temp))));
            dd = lap_image[a*X+b];
        }

    copy ( IMAGE, PREV_IMAGE );
    copy ( lap_image, IMAGE );
    free ( lap_image );
    displa ( draw, IMAGE );
}

    /**-----***/
```

```
void
prewitt()
{
    float *pre_image,
          temp[3][3],
          v,
          h,
          v1,
          h1;

    int   a,
          b,
          dd;
```

```
XCclearWindow ( XtDisplay ( draw ),
                XtWindow ( draw )
                );

XCclearWindow ( XtDisplay ( drawh ),
                XtWindow ( drawh )
                );

pre_image = (float *)calloc( X*Y,sizeof(float));

for (a = 0; a < Y; a++)
    for (b = 0; b < X; b++) {
        get_three_image(a,b,temp,IMAGE);
        h = (abs((int)(con(pre_h, temp))));
        v = (abs((int)(con(pre_v, temp))));
        h1 = h/3;
        v1 = v/3;
        pre_image[a*X+b] = max(h1, v1);
        dd = pre_image[a*X+b];
    }

    copy ( IMAGE, PREV_IMAGE );
    copy ( pre_image, IMAGE );
    free ( pre_image );
    displa ( draw, IMAGE );
}

/***/-----***/
```



```

/**-----**

Program : enlarge.c

Programmer : Kumar Chebrolu

This program enlarges the current image and inverts the current image.

**-----**/

#include "tst.h"

/**-----**/

getdata ( buf,k, len )
char *buf;
int len, k;
{
    int i;
    for ( i = 0; i < len; i++ )
        *buf++ = IMAGE[k*len+i];
}

/**-----**/

zoomin ( pp )
Widget pp;
{
    char *a, *b, *p, *q;
    int xs, ys;
    int ii, i, k, j, kk;
    int newxsize, newysize;
    int z;
    float *enimg;

    ZOOM_CNT++;
    if ( ZOOM_CNT < 2 ) {

        xs = ys = 2;

        enimg = ( float * ) XtMalloc ( X*Y*xs*ys* sizeof ( float ));

        newxsize = X * xs;
        newysize = Y * ys;

        a = ( char * ) XtMalloc ( X* sizeof ( char ));

        b = ( char * ) XtMalloc ( newxsize* sizeof ( char ));

```

```
if ( b == NULL ) {
    puts ( " memory allocation failure in b " );
    exit ();
}
kk = 0;

for ( i = 0; i < Y; i++ ) {
    getdata( a, i, X );
    p = a;
    q = b;
    for ( k = 0; k < X; k++, p++ ) {
        for ( j = 0; j < xs; j++ )
            *q++ = *p;
    }
    for(j=0; j< ys; j++) {
        for ( ii = 0; ii < newxsize; ii++ ) {
            enimg[kk] = b[ii];
            kk++;
        }
    }
}

X = newxsize;
Y = newysize;

XBell ( XtDisplay(draw ), 100 );
copy ( enimg, IMAGE );
XtFree ( enimg );
XtFree ( a );
XtFree ( b );
displa ( draw, IMAGE );

}
else warn ( p , error[3] );
}

/****-----****/

invert()
{
    int i, j;
    unsigned char *k;

    k = ( unsigned char * ) XtMalloc ( X * Y* sizeof ( unsigned char ) );

    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ ) {
            k[i*X+j] = IMAGE[i*X+j];
            k[i*X+j] = 0377 ^ k[i*X+j] ;
            IMAGE[i*X+j] = k[i*X+j];
        }

    XBell ( XtDisplay ( draw ), 100 );
```

```
XtFree ( k );  
displa ( draw, IMAGE );
```

```
}
```

```
/**-----**/
```

```
/**-----**
```

Program : file.c

Programmer : Kumar Chebrolu

This program loads the image from any directory, calculates the frequency, copies the images etc..

```
***-----**/
```

```
#include "tst.h"
```

```
/**-----**/
```

```
input()
{
    char *name,
        *tmp,
        *tmp1,
        *tmp2;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    XClearWindow ( XtDisplay ( drawh ),
                  XtWindow ( drawh )
                  );

    name = (char * )calloc ( 50, sizeof(char));

    strcpy ( name, Filename );

    tmp = strtok( name, ".");
    name = NULL;
    tmp1 = strtok ( name, "y" );
    name = NULL;
    tmp2 = strtok ( name, " " );
    name = NULL;

    if ( tmp == NULL)
        puts ("ERROR: input is wrong");
    else
        if ( tmp1 == NULL )
            puts ("ERROR: input is wrong");
        else {
            if ( tmp2 == NULL )
                X = Y = atoi ( tmp1 );
            else {
                X = atoi(tmp1);
                Y = atoi(tmp2);
            }
        }
}
```

```
        ZOOM_CNT = 0;
        free ( IMAGE );
        load_file ();
    }
}

/***/-----***/

input_longfile()
{
    char  *name,
          *tmp,
          *tmp1,
          *tmp2;

    int   i,
          count;

    name = (char * )calloc ( 50, sizeof(char));

    count = 0;
    for ( i = 0; i <= strlen ( Filename ); i++ )
        if (Filename[i] == '/' )
            count++;

    strcpy ( name, Filename );

    input ();
}

/***/-----***/

Frequency ( fimage )
float *fimage;
{
    int   f,
          i,
          j;

    for ( i = 0; i <= 255; i++ )
        FREQ[i] = 0;

    for ( i = 0; i < Y; i++ ) {
        for ( j = 0; j < X; j++ ) {
            f = fimage[i*X+j];
            if ( f > 255 )
                f = 255;
            FREQ[f]++;
        }
    }
}

/***/-----***/
```

```
load_file ()
{
    int    i,
          j,
          f,
          memo;

    unsigned char    ch;

    printf ( "Filename = %s0,Filename );

    if ( (fp = fopen ( Filename, "r" )) == NULL )
        puts ("ERROR: Input File not found");

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    XClearWindow ( XtDisplay ( drawh ),
                  XtWindow ( drawh )
                  );

    IMAGE = ( float * ) calloc ( 8* X * Y , sizeof ( float ));
    PREV_IMAGE = ( float * ) calloc ( 8 * X * Y , sizeof ( float ));
    TMP_IMAGE = ( float * ) calloc ( 8 * X * Y , sizeof ( float ));

    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ ) {
            fscanf ( fp, "%c", &ch );
            IMAGE[i*X+j] = ch&0377;
            f = IMAGE[i*X+j];
            if ( f > 1 )
                IM_TYPE = 1;
            else
                IM_TYPE = 0;
        }

    if ( IM_TYPE == 0 )
        printf ("BINARY IMAGE 0);
    else
        printf ("GRAY IMAGE 0);

    copy ( IMAGE, TMP_IMAGE );
    XBell ( XtDisplay ( draw ), 100 );

}

/****-----***/

ftod ( image, DIM )
float *image;
unsigned char *DIM;
{
```

```
int i,
    j;

XClearWindow ( XtDisplay ( draw ),
               XtWindow ( draw )
               );

XClearWindow ( XtDisplay ( drawh ),
               XtWindow ( drawh )
               );

for ( i = 0; i < Y; i++ )
    for ( j = 0; j < X; j++ ) {
        DIM[i*X+j] = ( unsigned char )image[i*X+j];
    }
}
```

```
/**-----**/
```

```
ctof ( image, DIM )
char *image;
float *DIM;
{
    int i,
        j;

    XClearWindow ( XtDisplay ( draw ),
                   XtWindow ( draw )
                   );

    XClearWindow ( XtDisplay ( drawh ),
                   XtWindow ( drawh )
                   );

    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ ) {
            DIM[i*X+j] = image[i*X+j];
        }
}
```

```
/**-----**/
```

```
after_th ( image )
float *image;
{
    int i,
        j;

    XClearWindow ( XtDisplay ( draw ),
                   XtWindow ( draw )
                   );
}
```

```
XClearWindow ( XtDisplay ( drawh ),
               XtWindow ( drawh )
               );

for ( i = 0; i < Y; i++ )
    for ( j = 0; j < X; j++ ) {
        if ( image[i*X+j] < ( float ) ( threshold ) )
            DIMAGE[i*X+j] = 0.0;
        else
            DIMAGE[i*X+j] = image[i*X+j];
    }

}

/***/-----***/

copy ( from, to )
float *from;
float *to;
{
    int  i,
        j;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    XClearWindow ( XtDisplay ( drawh ),
                  XtWindow ( drawh )
                  );

    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ )
            to[i*X+j] = from[i*X+j];
    XBell ( XtDisplay ( draw ), 100 );

}

/***/-----***/

loaded ()
{
    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    XClearWindow ( XtDisplay ( drawh ),
                  XtWindow ( drawh )
                  );

    displa ( draw, IMAGE );
```



```
    Frequency ( IMAGE );
    draw_h ();

}

/***/-----***/

undo ()
{
    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    XClearWindow ( XtDisplay ( drawh ),
                  XtWindow ( drawh )
                  );

    copy ( PREV_IMAGE, IMAGE );
    displa ( draw, IMAGE );
    Frequency ( IMAGE );
    draw_h ();

}

/***/-----***/

init ()
{
    int    i,
           j;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );
    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ )
            IMAGE[i*X+j] = 0.0;

    copy ( TMP_IMAGE, IMAGE );
    displa ( draw, IMAGE );
    Frequency ( IMAGE );
    draw_h ();

}

/***/-----***/
```

```
/**-----**
```

Program : help.c

Programmer : Kumar Chebrolu

This program is written to provide online help for all the functions used in this graphical user interface.

```
***-----**/
```

```
#include "tst.h"
```

```
char *hopen[] = {  
    "The user can select a required image file from any directory using filter from the popup file selectio  
    "Save the file",  
    "Save the file under the new name",  
    "The current image will be converted into postscript file named PRINT, the file can be directed to  
    "The user can go to the previous command",  
    "The image can be inverted with this command i.e., negative",  
    "Apply FFT to binary image",  
    "The image can be doubled",  
    "The image can be rotated by using three different algorithms - Bilinear Interpolation, - Biline  
    "The binary image can be represented by two methods - Quad Tree Method, - Run Length Meth  
    "The threshold is calculated automatically",  
    "The threshold is to given by the user",  
    "The gray levels of the image are equalized",  
    "The image is displayed on the window",  
};
```

```
char *one[] = {  
    "Open",  
    "Save",  
    "Save As",  
    "Print",  
    "Undo",  
};
```

```
char *two[] = {  
    "Invert",  
    "FFT",  
    "Zoomin",  
    "Rotate",  
};
```

```
char *three[] = {  
    "Edge Detection",  
};
```

```
char *four[] = {  
    "Representation",  
};
```

```
char *five[] = {
```

```
        "Auto_Threshold",
        "Interactive_Threshold",
        "Histogram Equalization",
    };

char *six[] = {
    "Loaded Image",
};

typedef struct {
    char **strs;
    int sizeh;
} ListItem;

ListItem help_items;
ListItem one_items = { one, XtNumber ( one ) };
ListItem two_items = { two, XtNumber ( two ) };
ListItem three_items = { three, XtNumber ( three ) };
ListItem four_items = { four, XtNumber ( four ) };
ListItem five_items = { five, XtNumber ( five ) };
ListItem six_items = { six, XtNumber ( six ) };

/****-----***/

help ( pp, str )
Widget pp;
char *str;
{
    Widget      helpdialog;
    XmString    h,
               *hstr;
    int         i;

    extern void helpcallback ();

    if ( ! ( strcmp ( str, "File" ) ) ) {
        help_items.strs = one_items.strs;
        help_items.sizeh = one_items.sizeh;
    }

    if ( ! ( strcmp ( str, "Transformations" ) ) ) {
        help_items.strs = two_items.strs;
        help_items.sizeh = two_items.sizeh;
    }

    if ( ! ( strcmp ( str, "Segmentation" ) ) ) {
        help_items.strs = three_items.strs;
        help_items.sizeh = three_items.sizeh;
    }

    if ( ! ( strcmp ( str, "Representation" ) ) ) {
        help_items.strs = four_items.strs;
        help_items.sizeh = four_items.sizeh;
    }
}
```

```
    }

    if ( ! ( strcmp ( str, "Histogram" ) ) ) {
        help_items.strs = five_items.strs;
        help_items.sizeh = five_items.sizeh;
    }

    if ( ! ( strcmp ( str, "Display" ) ) ) {
        help_items.strs = six_items.strs;
        help_items.sizeh = six_items.sizeh;
    }

    hstr = ( XmString * ) XtMalloc ( help_items.sizeh * sizeof ( XmString ) );

    h = XmStringCreateSimple ( "Menu" );

    for ( i = 0; i < help_items.sizeh ; i++ ) {
        hstr[i] = XmStringCreateSimple ( help_items.strs[i] );
    }

    helpdialog = XmCreateSelectionDialog (
        pp, "help",
        NULL, 0
    );

    XtVaSetValues ( helpdialog,
        XmNlistLabelString, h,
        XmNlistItems, hstr,
        XmNlistItemCount, help_items.sizeh,
        XmNmustMatch, True,
        NULL );

    XtUnmanageChild ( XmSelectionBoxGetChild
        ( helpdialog, XmDIALOG_HELP_BUTTON),
        False );

    XtUnmanageChild ( XmSelectionBoxGetChild
        ( helpdialog, XmDIALOG_APPLY_BUTTON),
        False );

    XtAddCallback ( helpdialog, XmNcancelCallback, XtDestroyWidget, NULL);

    XtAddCallback ( helpdialog, XmNokCallback, helpcallback, NULL );

    XmStringFree ( h );

    XtFree ( hstr );
```

```
XtManageChild ( helpdialog );

}

/****-----****/

helpcallback ( w, client_data, cbs )
Widget w;
XtPointer client_data;
XmSelectionBoxCallbackStruct *cbs;
{
    int      i,
            HH;

    char     *value,
            *match;

    Widget   dialog;
    Pixel    fg,
            bg;

    XmString buffer;
    Pixmap   map;
    Display  *dpy = XtDisplay ( w );
    int      screen = DefaultScreen ( dpy );

    Arg      arg_list[2];
    extern   DestroyShell(),
            GetTopShell ();

    XmStringGetLtoR ( cbs->value,
                    XmSTRING_DEFAULT_CHARSET, &value
                    );

    if ( ! ( strcmp ( value, "Open" ))) HH = 0; else
    if ( ! ( strcmp ( value, "Save" ))) HH = 1; else
    if ( ! ( strcmp ( value, "Save As" ))) HH = 2; else
    if ( ! ( strcmp ( value, "Print" ))) HH = 3; else
    if ( ! ( strcmp ( value, "Undo" ))) HH = 4; else
    if ( ! ( strcmp ( value, "Invert" ))) HH = 5; else
    if ( ! ( strcmp ( value, "FFT" ))) HH = 6; else
    if ( ! ( strcmp ( value, "Zoomin" ))) HH = 7; else
    if ( ! ( strcmp ( value, "Rotate" ))) HH = 8; else
    if ( ! ( strncmp ( value, "Edge",4 ))) HH = 9; else
    if ( ! ( strncmp ( value, "Repr",4 ))) HH = 10; else
    if ( ! ( strncmp ( value, "Auto",4 ))) HH = 11; else
    if ( ! ( strncmp ( value, "Inte",4 ))) HH = 12; else
    if ( ! ( strncmp ( value, "Hist",4 ))) HH = 13; else
    if ( ! ( strncmp ( value, "Load",4 ))) HH = 14;

    match = hopen[HH];

    buffer = XmStringCreateLtoR ( match ,
```

```
        XmSTRING_DEFAULT_CHARSET
    );

    XtSetArg ( arg_list[0],
              XmNdeleteResponse, XmDESTROY
    );

    XtSetArg ( arg_list[1],
              XmNhelpLabelString, XmStringCreateSimple ( "OK" )
    );

    XtSetArg ( arg_list[2],
              XmNtitle, value
    );

    dialog = XmCreateMessageDialog (
        w, "dialog",
        arg_list, 2
    );

    XtUnmanageChild ( XmMessageBoxGetChild
        ( dialog, XmDIALOG_OK_BUTTON ) , False
    );

    XtUnmanageChild ( XmMessageBoxGetChild
        ( dialog, XmDIALOG_CANCEL_BUTTON ), False
    );

    XtAddCallback ( dialog, XmNhelpCallback, XtDestroyWidget, NULL );

    XtVaGetValues ( dialog,
                  XmNforeground, &fg,
                  XmNbackground, &bg,
                  NULL
    );

    map = XCreatePixmapFromBitmapData (
        dpy, XtWindow ( w ),
        help_bits, help_width, help_height,
        fg, bg,
        DefaultDepth ( dpy, screen ) );

    XtVaSetValues ( dialog,
                  XmNmessageString, buffer,
                  XmNsymbolPixmap, map,
                  NULL
    );

    XtManageChild ( dialog );

    for ( i = 0; i < 5; i++ )
        XBell ( XtDisplay ( draw ), 1 );

    XtPopup ( XtParent ( dialog ), XtGrabNone );
```

}

/***-----***/

```
/**-----**
```

Program : hist.c

Programmer : Kumar Chebrolu

This program calculates the threshold value, and histogram equalization is done for the gray image.

```
***-----**/
```

```
#include <Xm/Scale.h>
#include "tst.h"
```

```
char *ggrey[] = {
    "0",
    "30",
    "60",
    "90",
    "120",
    "150",
    "180",
    "210",
    "240",
};
```

```
/**-----**/
```

```
void
auto_th()
{
    int    i,
          j;

    float *aut;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    threshold = auto_threshold();

    aut = ( float * ) calloc ( X * Y, sizeof ( float ) );

    if ( aut == NULL ){
        puts ( " memory allocation failure in aut " );
        exit ();
    }

    for ( i = 0; i < Y; i++)
        for ( j = 0; j < X; j++)
            if ( IMAGE[i*X+j] < threshold )
                aut[i*X+j] = 0.0;
```



```
        else
            aut[i*X+j] = 1.0;

    copy ( IMAGE, PREV_IMAGE );
    copy ( aut, IMAGE );
    free ( aut );
    displa ( draw, IMAGE );
    Frequency ( IMAGE );
    draw_h ();
}

/**-----**/
```

```
int
h( j, k )
int j, k;
{
    int p;

    for ( p = j; p <= 255; p++ ) {
        if ( FREQ[p] == k )
            return ( p );
    }
}

/**-----**/
```

```
int
auto_threshold()
{
    int i,
        j,
        k = 0,
        k1,
        k2 = 0,
        k3 = 0,
        k4;

    for ( i = 0; i < 200; i++ )
        k = max ( FREQ[i], k );

    k1 = h ( 0, k );
    k2 = k1 + 15;

    for ( j = k2; j <= 255; j++ )
        k3 = max ( FREQ[j], k3 );

    k4 = h ( k1, k3 );
    thres = ( k1 + k4 ) / 2;

    return ( abs ( thres ) );
}
```

```
/**-----**/  
  
Widget  
GetTopShell ( w )  
Widget w;  
{  
    while ( w && !XtIsWMSHELL ( w ) )  
        w = XtParent ( w );  
  
    return w;  
}  
  
/**-----**/  
  
draw_h ( )  
{  
    int    i,  
          hf;  
  
    float gr;  
  
    set_gc( drawh );  
  
    XClearWindow ( XtDisplay ( drawh ),  
                  XtWindow ( drawh )  
                  );  
  
    XDrawLine ( XtDisplay ( drawh ), XtWindow ( drawh ), gc,  
                24, 10, 24, 175  
                );  
  
    XDrawLine ( XtDisplay ( drawh ), XtWindow ( drawh ), gc,  
                20, 170, 280, 170  
                );  
  
    for ( i = 1; i < 9; i++ )  
        XDrawLine ( XtDisplay ( drawh ), XtWindow ( drawh ), gc,  
                    25+i*30, 170, 25+i*30, 175  
                    );  
  
    for ( i = 0; i < XtNumber ( ggrey ); i++ )  
        XDrawString ( XtDisplay ( drawh ), XtWindow ( drawh ), gc,  
                      18+30*i, 188, ggrey[i], strlen ( ggrey[i] )  
                      );  
  
    gr = 0;  
    for ( i = 0; i <= 255; i++ )  
        gr = max ( gr, FREQ[i] );  
    gr = gr / 160;
```

```
for ( i = 0; i <= 255; i++ ) {
    if ( FREQ[i] != 0 )
        hf = FREQ[i] / gr;
    else
        hf = 0;
    XDrawLine (
        XtDisplay ( drawh ), XtWindow ( drawh ), gc,
        25 + i, 170 - hf, 25 + i, 170
    );
}
}
```

```
/***/-----***/
```

```
void
interactive (w)
Widget w;
{
    Widget      dialog,
              pane,
              form,
              widget,
              scale;
    Pixel      fg,
              bg;
    int        i;
    extern void DestroyShell(), value();

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    dialog = XtVaCreatePopupShell ( "dialog",
                                   xmDialogShellWidgetClass, GetTopShell ( w ),
                                   XmNdeleteResponse, XmDESTROY,
                                   NULL );

    pane = XtVaCreateWidget ( "pane",
                              xmRowColumnWidgetClass, dialog,
                              NULL );

    XtVaCreateManagedWidget ( "Threshold Value : ",
                               xmLabelGadgetClass, pane,
                               NULL );

    form = XtVaCreateWidget ( "form1",
                              xmFormWidgetClass, pane,
                              NULL );

    XtVaGetValues ( form,
                    XmNforeground, &fg,
```

```
        XmNbackground, &bg,  
        NULL );  
  
scale = XtVaCreateManagedWidget ( "scale",  
        xmScaleWidgetClass, form,  
        XmNleftAttachment, XmATTACH_FORM,  
        XmNtopAttachment, XmATTACH_FORM,  
        XmNorientation, XmHORIZONTAL,  
        XmNmaximum, 255,  
        XmNshowValue, True,  
        NULL );  
  
XtAddCallback ( scale, XmNvalueChangedCallback, value, NULL );  
  
for ( i = 0; i < 25; i++ )  
        XtVaCreateManagedWidget ( " ",  
        xmLabelGadgetClass, scale,  
        NULL );  
  
XtManageChild ( form );  
  
form = XtVaCreateWidget ( "form2",  
        xmFormWidgetClass, pane,  
        XmNfractionBase, 5,  
        NULL );  
  
widget = XtVaCreateManagedWidget ( "Ok",  
        xmPushButtonGadgetClass, form,  
        XmNtopAttachment, XmATTACH_FORM,  
        XmNbottomAttachment, XmATTACH_FORM,  
        XmNleftAttachment, XmATTACH_POSITION,  
        XmNleftPosition, 1,  
        XmNrightAttachment, XmATTACH_POSITION,  
        XmNrightPosition, 2,  
        XmNshowAsDefault, True,  
        XmNtopOffset, 20,  
        XmNdefaultButtonShadowThickness, 1,  
        NULL );  
  
XtAddCallback ( widget, XmNactivateCallback, DestroyShell, dialog );  
  
widget = XtVaCreateManagedWidget ( "Cancel",  
        xmPushButtonGadgetClass, form,  
        XmNtopAttachment, XmATTACH_FORM,  
        XmNbottomAttachment, XmATTACH_FORM,  
        XmNleftAttachment, XmATTACH_POSITION,  
        XmNleftPosition, 3,  
        XmNrightAttachment, XmATTACH_POSITION,  
        XmNrightPosition, 4,  
        XmNshowAsDefault, True,  
        XmNtopOffset, 20,  
        XmNdefaultButtonShadowThickness, 1,
```

```
        NULL );

XtAddCallback ( widget, XmNactivateCallback, DestroyShell, dialog );

XtManageChild ( form );
{
    Dimension h;
    XtVaGetValues ( widget,
                    XmNheight, &h,
                    NULL
                    );

    XtVaSetValues ( form,
                    XmNpaneMaximum, h,
                    XmNpaneMinimum, h,
                    NULL
                    );
}

XtManageChild ( pane );

XtPopup ( dialog, XtGrabNone );
}

/****-----****/

void
value ( w, client_data, cbs )
Widget w;
caddr_t client_data;
XmScaleCallbackStruct *cbs;
{
    float *inter;
    int    i,
          j,
          th;

    inter = ( float * ) calloc ( X * Y, sizeof ( float ) );

    if ( inter == NULL ) {
        puts ( " memory allocation failure in inter " );
        exit();
    }

    th = cbs->value;

    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ )
            if ( IMAGE[i*X+j] < th )
                inter[i*X+j] = 0.0;
            else
                inter[i*X+j] = IMAGE[i*X+j];
}
```

```
copy ( IMAGE, PREV_IMAGE );
copy ( inter, IMAGE );
free ( inter );
displa ( draw, IMAGE );
Frequency ( IMAGE );
draw_h ();

}

/***/-----***/

void
DestroyShell ( widget, shell )
Widget widget, shell;
{
    XtDestroyWidget ( shell );
}

/***/-----***/

hist_eq ( p )
Widget p;
{
    int    infreq[256] = { 0 },
          mapfreq[256] = { 0 },
          i,
          j,
          k,
          f,
          low,
          high;

    float *EQ_IMAGE,
          ff;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    if ( IM_TYPE == 0 )
        warn ( p, error[0] );
    else {
        EQ_IMAGE = ( float * ) calloc ( X * Y, sizeof ( float ) );

        if ( EQ_IMAGE == NULL ){
            puts ( " memory allocation failure in EQ_IMAGE " );
            exit 0;
        }

        for ( i = 0; i < Y; i++ )
            for ( j = 0; j < X; j++ ) {
                f = IMAGE[i*X+j];
                if ( f > 255 )
```

```
        f = 255;
        infreq[f]++;
    }

for ( i = 1; i < 256; i++ )
    infreq[i] += infreq[i-1];

for ( i = 0; i < 256; i++ ) {
    if ( !infreq[i] )
        continue;
    else {
        low = i;
        break;
    }
}

for ( i = 254; i > 0; i-- ) {
    if ( infreq[i] == infreq[i+1] )
        continue;
    else {
        high = i + 1;
        break;
    }
}

f = infreq[high] / ( high - low + 1 );

i = low;
mapfreq[low] = low;

for ( j = 1; j < 256; j++ ) {
    if ( i >= high )
        break;
    while ( infreq[i] < ( j+1 ) * f ) {
        mapfreq[i+1] = j + low;
        if ( ++i >= high )
            break;
    }
}

for ( i = 0; i < Y; i++ )
    for ( j = 0; j < X; j++ ) {
        k = IMAGE[i*X+j];
        EQ_IMAGE[i*X+j] = mapfreq[k];
    }

copy ( IMAGE, PREV_IMAGE );
copy ( EQ_IMAGE, IMAGE );
free ( EQ_IMAGE );
displa ( draw, IMAGE );
Frequency ( IMAGE );
draw_h ();
}
```

```
}  
  
/****-----***/  
  
warn ( p, err )  
Widget p;  
char err[];  
{  
  
    Widget      dialog;  
    Display     *dpy = XtDisplay ( p );  
    int         screen = DefaultScreen ( dpy );  
    Pixel      bg,  
             fg;  
    Arg        arg_list[2];  
    XtAppContext app = XtWidgetToApplicationContext ( p );  
    XmString   text;  
    char       buf[200];  
    TimeOutClientData *data = XtNew ( TimeOutClientData );  
    extern     blink(), destroy ();  
  
    text = XmStringCreateSimple ("Cancel");  
  
    XClearWindow ( XtDisplay ( draw ),  
                  XtWindow ( draw )  
                  );  
  
    XtSetArg ( arg_list[0],  
              XmNhelpLabelString, text  
              );  
  
    XtSetArg ( arg_list[1],  
              XmNdeleteResponse, XmDESTROY  
              );  
  
    dialog = XmCreateMessageDialog (  
        p, "warn",  
        arg_list, 2  
        );  
  
    XmStringFree ( text );  
  
    XtUnmanageChild ( XmMessageBoxGetChild  
                      ( dialog, XmDIALOG_OK_BUTTON),  
                      False  
                      );  
  
    XtUnmanageChild ( XmMessageBoxGetChild  
                      ( dialog, XmDIALOG_CANCEL_BUTTON),  
                      False  
                      );  
  
    XtAddCallback ( dialog, XmNhelpCallback, XtDestroyWidget, NULL);  
    XtAddCallback ( dialog, XmNdestroyCallback, destroy, data );  
}
```



```
XtVaGetValues ( dialog,
                XmNforeground, &fg,
                XmNbackground, &bg,
                NULL
                );

data->pix1 = XCreatePixmapFromBitmapData (
            dpy, XtWindow ( p ),
            star_bits, star_width, star_height,
            fg, bg,
            DefaultDepth ( dpy, screen )
            );

data->pix2 = XCreatePixmapFromBitmapData (
            dpy, XtWindow ( p ),
            star1_bits, star1_width, star1_height,
            fg, bg,
            DefaultDepth ( dpy, screen )
            );

data->dialog = dialog;
data->app = app;
data->id = XtAppAddTimeOut (
            app, 1000L,
            blink, data
            );
text = XmStringCreateLtoR ( err, XmSTRING_DEFAULT_CHARSET );

XtVaSetValues ( dialog,
                XmNmessageString, text,
                XmNsymbolPixmap, data->pix2,
                NULL
                );
XmStringFree ( text );

XtManageChild ( dialog );

XtPopup ( XtParent ( dialog ), XtGrabNone );

}

/**-----**/

void
pop (w)
Widget w;
{
    Widget      dialog,
              pane,
              form,
              widget,
              scale;
    Pixel      fg,
              bg;
```



```
XmNleftAttachment, XmATTACH_POSITION,  
XmNleftPosition, 1,  
XmNrightAttachment, XmATTACH_POSITION,  
XmNrightPosition, 2,  
XmNshowAsDefault, True,  
XmNtopOffset, 20,  
XmNdefaultButtonShadowThickness, 1,  
NULL );
```

```
XtAddCallback ( widget, XmNactivateCallback, DestroyShell, dialog );
```

```
widget = XtVaCreateManagedWidget ( "Cancel",  
    xmPushButtonGadgetClass, form,  
    XmNtopAttachment, XmATTACH_FORM,  
    XmNbottomAttachment, XmATTACH_FORM,  
    XmNleftAttachment, XmATTACH_POSITION,  
    XmNleftPosition, 3,  
    XmNrightAttachment, XmATTACH_POSITION,  
    XmNrightPosition, 4,  
    XmNshowAsDefault, True,  
    XmNtopOffset, 20,  
    XmNdefaultButtonShadowThickness, 1,  
    NULL );
```

```
XtAddCallback ( widget, XmNactivateCallback, DestroyShell, dialog );
```

```
XtManageChild ( form );
```

```
{  
    Dimension h;  
    XtVaGetValues ( widget,  
        XmNheight, &h,  
        NULL  
    );  
  
    XtVaSetValues ( form,  
        XmNpaneMaximum, h,  
        XmNpaneMinimum, h,  
        NULL  
    );  
}
```

```
displa ( drawp, IMAGE );
```

```
XtPopup ( dialog, XtGrabNone );
```

```
}
```

```
/**-----**/
```



```
        False
    );

XtUnmanageChild ( XmMessageBoxGetChild
    ( dialog3, XmDIALOG_OK_BUTTON ),
    False
    );

XtAddCallback ( dialog3, XmNokCallback, done1, NULL );

t1 = XmStringCreateLtoR ( "0ilename PRINT* is created0n your current directory...0he file PRINT* can be di

XtVaSetValues ( dialog3,
    XmNmessageString, t1 ,
    NULL
    );

XmStringFree ( t1 );

print_proc1 ();

XtManageChild ( dialog3 );
XtPopup ( XtParent ( dialog3 ), XtGrabNone );

}

/**-----**/

print_proc1 ()
{
    char  numbs[] = "0123456789ABCDEF";

    int   i,
          j,
          bits,
          count,
          value,
          px,
          py;

    float a,
           HWR,
           psize,
           gap,
           width,
           height;

    char  pfile[10];

    print_cnt++;
    sprintf ( pfile, "PRINT%d",print_cnt );
```

```
lp = fopen ( pfile, "w" );

if ( lp == 0 ) {
    puts ( "cannot open" );
    exit ();
}

px = X;
py = Y;
width = 6.0;
bits = 8;

HWR = ( float ) py / ( float ) px;

count = ( int ) ( py * px );

ptsize = 3.*width;

a = 4.*1820.*492./3./3./512./712.;

height = HWR * width;

fprintf ( lp,
         "%%!0);

fprintf ( lp,
         "%%%DocumentFonts: Times-Roman0);

fprintf ( lp,
         "%%%BoundingBox: 0 0 %d %d0, ( int )( width * 72. ), ( int )( height * 72. ) );

fprintf ( lp,
         "%%%EndComments0 );

fprintf ( lp,
         "/inch { 72 mul } def0);

fprintf ( lp,
         "/Times-Roman findfont %d scalefont setfont0, ( int ) psize );

fprintf ( lp,
         "/picstr %d string def0, px * bits / 8 + ( px * bits % 8 > 0 ? 1 : 0 ) );

fprintf ( lp,
         "/imagepinhead0 );

fprintf ( lp,
         " { %d %d %d", px, py, bits );

fprintf ( lp,
         " [%d 0 0 -%d 0 %d]0, px, py, py );
```

```
fprintf ( lp,
        " { currentfile picstr0 );

fprintf ( lp,
        " readhexstring pop } image0 );

fprintf ( lp,
        " } def0 );

fprintf ( lp,
        "gsave0 );

fprintf ( lp,
        " 1.25 inch 2.50 inch translate0 );

fprintf ( lp,
        " %.2f inch %.2f inch scale0, width, height );

fprintf ( lp,
        " {} settransfer0 );

fprintf ( lp,
        "imagepinhead0 );

if ( bits != 8 )
    px = ( px % 8 ) ? px / 8 + 1 : px / 8;

for ( i = 0; i < Y; i++ )
    for ( j = 0; j < X; j++ ) {
        value = IMAGE[i*X+j];
        fprintf ( lp, "%c", numbs[value>>4] );
        fprintf ( lp, "%c", numbs[value&0x0f] );
        if ( j % 38 == 37 )
            fprintf ( lp, "0 );
    }

fprintf ( lp,
        "0restore0 );

fprintf ( lp,
        "gsave 1.25 inch 2.50 inch translate0 );

fprintf ( lp,
        "grestore0 );

fprintf ( lp,
        "showpage0 );

fprintf ( lp,
        "%%%%Trailer0 );

}

/****-----****/
```



```
/**-----**
```

Program : menu.c

Programmer : Kumar Chebrolu

This program provides a warn dialog, and the routines for File submenu in the GUI.

```
***-----**/
```

```
#include "tst.h"
```

```
/**-----**/
```

```
void
```

```
get_filename ( nw,  
              client_data,  
              cbs )
```

```
Widget nw;  
char *client_data;  
XmSelectionBoxCallbackStruct *cbs;  
{
```

```
    XmStringGetLtoR ( cbs->value,  
                    XmSTRING_DEFAULT_CHARSET, &Filename );
```

```
    XtDestroyWidget ( XtParent(nw) );  
    input ();
```

```
}
```

```
/**-----**/
```

```
void
```

```
get_longfilename ( nw,  
                 client_data,  
                 cbs )
```

```
Widget nw;  
char *client_data;  
XmSelectionBoxCallbackStruct *cbs;  
{
```

```
    XmStringGetLtoR ( cbs->value,  
                    XmSTRING_DEFAULT_CHARSET, &Filename );
```

```
    XtDestroyWidget ( XtParent(nw) );  
    input_longfile ();
```

```
}
```

```
/**-----**/
```

```
new ( p )
```

```
Widget p;
```

```
{
```

```
static Widget    dialog;
Arg             arg_list[3];

XClearWindow ( XtDisplay ( draw ),
               XtWindow ( draw )
               );

XtSetArg ( arg_list[0],
          XmNselectionLabelString,
          XmStringCreateSimple ( " Enter the New filename : " )
          );

XtSetArg ( arg_list[1],
          XmNautoUnmanage, False
          );

dialog = XmCreatePromptDialog (
        p, "newd",
        arg_list, 2
        );

XtUnmanageChild ( XmSelectionBoxGetChild
                  ( dialog, XmDIALOG_HELP_BUTTON ),
                  False
                  );

XtAddCallback ( dialog, XmNokCallback, get_filename, p );
XtAddCallback ( dialog, XmNcancelCallback, XtDestroyWidget, NULL);

XtManageChild ( dialog );

XtPopup ( XtParent ( dialog ), XtGrabNone );
}

/****-----****/

void
opend ( p1 )
Widget p1;
{
    Widget    dialog1;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    dialog1 = XmCreateFileSelectionDialog (
            p1, "opend",
            NULL, 0
            );

    XtAddCallback ( dialog1, XmNokCallback, get_longfilename, p1 );
    XtAddCallback ( dialog1, XmNcancelCallback, XtDestroyWidget, NULL );
}
```

```
        XtManageChild ( dialog1 );
    }

    /**-----**/

    void
    blink ( data )
    TimeoutClientData *data;
    {
        XBell ( XtDisplay ( draw ), 100 );

        data->id = XtAppAddTimeout ( data->app, 250L, blink, data );

        XtVaSetValues ( data->dialog,
                        XmNsymbolPixmap,
                        (data->which = !data->which) ? data->pix1 : data->pix2,
                        );
    }

    /**-----**/

    void
    destroy ( dialog,
            data )
    Widget dialog;
    TimeoutClientData *data;
    {
        Pixmap    symbol;

        XtRemoveTimeout ( data->id);

        XFreePixmap ( XtDisplay ( data->dialog ), data->pix1 );
        XFreePixmap ( XtDisplay ( data->dialog ), data->pix2 );

        XtFree ( data );
    }

    /**-----**/

    void
    done (dialog )
    Widget dialog;
    {
        XtDestroyWidget ( dialog );
        XClearWindow ( XtDisplay ( draw ),
                     XtWindow ( draw )
                     );
    }

    /**-----**/
```

```
saved ( p2 )
Widget p2;
{
    Widget          dialog2;
    Display         *dpy = XtDisplay ( p2 );
    int             screen = DefaultScreen ( dpy );
    Pixel          bg,
                 fg;
    Arg            arg_list[2];
    XmString       text;

    XtAppContext   app = XtWidgetToApplicationContext ( p2 );

    TimeOutClientData *data = XtNew ( TimeOutClientData );

    text = XmStringCreateSimple ("Save");

    XtSetArg ( arg_list[0],
              XmNokLabelString, text );

    XtSetArg ( arg_list[1],
              XmNdeleteResponse, XmDESTROY );

    dialog2 = XmCreateMessageDialog (
                p2, "saved",
                arg_list, 2
            );

    XmStringFree ( text );

    XtUnmanageChild ( XmMessageBoxGetChild
                    ( dialog2, XmDIALOG_HELP_BUTTON),
                    False
                    );

    XtAddCallback ( dialog2, XmNokCallback, done, NULL);
    XtAddCallback ( dialog2, XmNcancelCallback, XtDestroyWidget, NULL );
    XtAddCallback ( dialog2, XmNdestroyCallback, destroy, data );

    XtVaGetValues ( dialog2,
                  XmNforeground, &fg,
                  XmNbackground, &bg,
                  NULL
                  );

    data->pix1 = XCreatePixmapFromBitmapData (
                dpy, XtWindow ( p2 ),
                star_bits, star_width, star_height,
                fg, bg,
                DefaultDepth ( dpy, screen )
                );

    data->pix2 = XCreatePixmapFromBitmapData (
```

```
        dpy, XtWindow ( p2 ),
        star1_bits, star1_width, star1_height,
        fg, bg,
        DefaultDepth ( dpy, screen )
    );

    data->dialog = dialog2;
    data->app = app;
    data->id = XtAppAddTimeOut (
        app, 1000L,
        blink, data
    );

    text = XmStringCreateSimple ( "Overwriting on the file " );

    XtVaSetValues ( dialog2,
        XmNmessageString, text,
        XmNsymbolPixmap, data->pix2,
        NULL
    );

    XmStringFree ( text );

    XtManageChild ( dialog2 );

    XClearWindow ( XtDisplay ( draw ),
        XtWindow ( draw )
    );

    XtPopup ( XtParent ( dialog2 ), XtGrabNone );

}

/**-----**/

getnewfile ( sw, client_data, cbs )
Widget sw;
char *client_data;
XmSelectionBoxCallbackStruct *cbs;
{

    char      *savefile;
    unsigned char  chs;
    int      i,
            j;

    XmStringGetLtoR ( cbs->value,
        XmSTRING_DEFAULT_CHARSET, &savefile
    );

    sa = fopen ( savefile, "w");

    if ( sa == NULL ) {
        puts ( " unable to open new file " );
    }
}
```

```
        XtDestroyWidget ( XtParent(sw) );
        return ;
    }

    for ( i = 0; i < Y; i++ )
        for ( j = 0; j < X; j++ ) {
            chs = IMAGE[i*X+j];
            fprintf ( sa, "%c", chs );
        }

    XtDestroyWidget ( XtParent(sw) );
}

/***/-----***/

saveas_proc ( p )
Widget p;
{
    static Widget    dialogs;
    Arg            arg_list[3];
    int            i,
                  j;

    XtSetArg ( arg_list[0],
               XmNselectionLabelString,
               XmStringCreateSimple ( "Enter the new file name : " )
               );

    XtSetArg ( arg_list[1],
               XmNautoUnmanage, False
               );

    dialogs = XmCreatePromptDialog (
                p, "savea",
                arg_list, 2 );

    XtAddCallback ( dialogs, XmNokCallback, getnewfile, p );

    XtAddCallback ( dialogs, XmNcancelCallback, XtDestroyWidget, NULL );

    XtManageChild ( dialogs );

    XtPopup ( XtParent ( dialogs ), XtGrabNone );
}

/***/-----***/
```

```
/**-----**
```

Program : rep.c

Programmer : Kumar Chebrolu

This program will represent the binary image using runlength method and quadtree methods. The result will be displayed on the regular window. The future work can be done to display the tree on the drawing area widget.

```
***-----**/
```

```
#include "tst.h"
```

```
/**-----**/
```

```
void  
run_proc( p )  
Widget p;  
{  
    int    a,  
          b,  
          count,  
          ans;  
  
    XClearWindow ( XtDisplay ( draw ),  
                  XtWindow ( draw )  
                  );  
  
    if ( IM_TYPE == 1 )  
        warn ( p, error[1] );  
    else {  
  
        for (a = 0; a < Y;a++) {  
            for (b = 0; b < X; b++)  
                if (b == 0) {  
                    count = 1;  
                    ans = IMAGE[a*X+b];  
                    printf(" %d",ans);  
                }  
                else if (ans == IMAGE[a*X+b])  
                    count++;  
                else {  
                    printf(" %d",count);  
                    ans = IMAGE[a*X+b];  
                    count = 1;  
                }  
                printf(" %d0,count);  
            }  
        }  
    }  
}
```

```
/**-----**/
```

```
char
*get_level( l )
int l;
{
    if ((l>=0) && (l<=50))
        return "A";
    else
        if ((l>=51) && (l<=100))
            return "B";
        if ((l>=101) && (l<=150))
            return "C";
        if ((l>=151) && (l<=200))
            return "D";
        if (l>255)
            return "E";
}

/****-----****/

void
run_proc1()
{
    int    a,
          b,
          count,
          ans;

    char *level;

    XClearWindow ( XtDisplay ( draw ),
                  XtWindow ( draw )
                  );

    for (a = 0; a < Y; a++) {
        for (b = 0; b < X; b++)
            if (b == 0) {
                count = 1;
                ans = IMAGE[a*X+b];
                level = get_level(ans);
                printf("%s ",level);
            }
        else if (level == get_level((int)IMAGE[a*X+b]))
            count++;
        else {
            level = get_level(ans);
            printf("%s ",level);
            printf("%d0,count);
            ans = IMAGE[a*X+b];
            count = 1;
        }
        printf("%d0,count);
    }
}
```



```
    }  
  
    /**-----**/  
  
    int  
    get_power( n )  
    int n;  
    {  
        int    p,  
              pp;  
  
        for ( p = 0; n > shift(p); p++)  
            pp = p + 1;  
  
        return pp;  
    }  
  
    /**-----**/  
  
    void  
    quad_proc( pp )  
    Widget pp;  
    {  
        int    p,  
              q,  
              r,  
              X_new,  
              Y_new,  
              gp,  
              nsize;  
  
        float *con_image;  
  
        XClearWindow ( XtDisplay ( draw ),  
                      XtWindow ( draw )  
                      );  
  
        if ( IM_TYPE == 1 )  
            warn ( pp, error[2] );  
        else {  
            if ( X != Y )  
                warn ( pp, error[5] );  
            else {  
  
                gp = get_power ( max ( X, Y ) );  
  
                X_new = shift ( gp );  
                Y_new = shift ( gp );  
                nsize = X_new * Y_new;  
  
                con_image = ( float * ) calloc ( nsize , sizeof ( float ) );  
  
                if ( con_image == NULL ){
```

```
        puts ( " memory allocation failure in con_image " );
        exit ();
    }

    for ( p = 0; p < Y; p++)
        for ( q = 0; q < X; q++ )
            con_image[p*X_new+q] = IMAGE[p*X+q];

    if ( check ( 0, 0, X_new, Y_new ) == 1 )
        printf ( "(%d)", IMAGE[0] );
    else
        divide( 0, 0, X_new, Y_new );
    }
}

/***/-----***/

int
check(ix,iy,cnx,cny)
int ix, iy, cnx, cny;
{
    int    temp,
           p,
           q;

    temp = IMAGE[iy*X+ix];

    for ( p = iy; p < cny; p++ )
        for ( q = ix; q < cnx; q++ )
            if ( temp != IMAGE[p*X+q] )
                return ( 0 );
    return ( 1 );
}

/***/-----***/

divide(ix, iy, dnx, dny)
int ix, iy, dnx, dny;
{
    int    halfx,
           halfy;

    halfx = (ix + dnx) /2;
    halfy = (iy + dny) /2;

    printf( "(" );

    if ( check ( ix, iy, halfx, halfy ) )
        printf ( "%d", ( int ) ( IMAGE[iy*X+ix] ) );
    else
        divide ( ix, iy, halfx, halfy );

    if ( check ( halfx, iy, dnx, halfy ) )
```

```
        printf( "%d", ( int ) ( IMAGE[iy*X+halfx] ) );
else
    divide ( halfx, iy, dnx, halfy );

if ( check ( ix, halfy, halfx, dny ) )
    printf ( "%d", ( int ) ( IMAGE[iy*X+halfx] ) );
else
    divide ( ix, halfy, halfy, dny );

if ( check ( halfx, halfy, dnx, dny ) )
    printf( "%d", ( int ) ( IMAGE[halfy*X+halfx] ) );
else
    divide ( halfx, halfy, dnx, dny );

printf( "" );
}

/**-----**/
```

```
/**-----**
```

Program : rot.c

Programmer : Kumar Chebrolu

This program rotates the current image using three different interpolation methods - Biconstant, Bilinear, and Bicubic.

```
***-----***/
```

```
#include "tst.h"
```

```
/**-----***/
```

```
proc_arr ()
```

```
{
```

```
    float  xoff,  
           yoff,  
           tranx,  
           trany,  
           finx,  
           finy,  
           *rotimage,  
           c,  
           s,  
           tx = 0.0,  
           ty = 0.0;
```

```
    int    i,  
           j,  
           con;
```

```
    rotimage = ( float * ) calloc ( X * Y , sizeof ( float ) );  
    XClearWindow ( XtDisplay ( draw ),  
                  XtWindow ( draw )  
                  );
```

```
    if ( rotimage == NULL ) {  
        puts ( " memory allocation failure " );  
        exit ();  
    }
```

```
    xoff = X / 2.0 - 0.5 ;  
    yoff = Y / 2.0 - 0.5 ;
```

```
    tranx = tx + xoff;  
    trany = ty + yoff;
```

```
    c = cos ( rad( angle ) );  
    s = sin ( rad( angle ) );
```

```
    for ( i = 0; i < Y; i++ ) {  
        for ( j = 0; j < X; j++ ) {
```

```
        finx = ( i - tranx ) * c
              + ( j - trany ) * s
              + xoff;

        finy = ( j - trany ) * c
              - ( i - tranx ) * s
        switch ( ROT_TYPE ) {
        case 0 : con = biconstant ( finx, finy, IMAGE );
                break;
        case 1 : con = bilinear ( finx, finy, IMAGE );
                break;
        case 2 : con = bicubic ( finx, finy, IMAGE );
                break;
        }
        if ( con < 0 ) con = 0;
        else if ( con > 255 ) con = 255;

        rotimage[i*X+j] = con;
    }
}

copy ( IMAGE, PREV_IMAGE );
copy ( rotimage, IMAGE );
free ( rotimage );
displa ( draw, IMAGE );
}

/**-----**/

biconstant ( fx, fy, im )
float fx, fy;
float *im;
{
    if ( ( fx >= 0 ) && ( fx < ( X - 1 ) ) &&
        ( fy >= 0 ) && ( fy < ( Y - 1 ) ) )
        return ( *( im + ( ( int ) ( fx + 0.5 ) * X ) + ( int ) ( fy + 0.5 ) ) );
    else
        return ( 0 );
}

/**-----**/

bilinear ( fx, fy, im )
float fx, fy;
float *im;
{
    float *imm;
    float coord[2][2] = {
        { 0, 0 },
        { 0, 0 }
    };

    float d0,
          d1,
```

```
    dx,
    dy;

int   val;

imm = im + ( int )fy * X + ( int ) fx;

if ( fx >= 0 && fx < X-1 && fy >= 0 && fy < Y-1 ) {
    coord[0][0] = *imm;
    coord[1][0] = *(imm+1);
    coord[0][1] = *(imm+X);
    coord[1][1] = *(imm+X+1);
}
else
if ( fx < 0 && fx >= -1 )
    if ( fy < 0 && fy >= -1 ) {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = 0;
        coord[1][1] = *(imm+X+1);
    }
    else if ( fy >= Y-1 && fy < Y ) {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = 0;
        coord[1][1] = *(imm+1);
    }
    else {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = *(imm+1);
        coord[1][1] = *(imm+X+1);
    }
}

else if ( fx >= X-1 && fx < X )
    if ( fy < 0 && fy >= -1 ) {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = 0;
        coord[1][1] = *(imm+X);
    }
    else if ( fy >= Y-1 && fy < Y ) {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = 0;
        coord[1][1] = *(imm);
    }
    else {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = *(imm);
        coord[1][1] = *(imm+X);
    }
}

else if ( fy < 0 && fy >= -1 ) {
```

```
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = *(imm+X);
        coord[1][1] = *(imm+X+1);
    }
    else if ( fy >= Y-1 && fy < Y ) {
        coord[0][0] = 0;
        coord[1][0] = 0;
        coord[0][1] = *(imm);
        coord[1][1] = *(imm+1);
    }
    else return ( 0 );

    dx = fx - (int)fx;
    dy = fy - (int)fy;
    d0 = coord[0][0] + dx * ( coord[1][0] - coord[0][0] );
    d1 = coord[0][1] + dx * ( coord[1][1] - coord[0][1] );
    val = (( int )( d0 + dy * ( d1 - d0 ))+ 0.5 );
    return ( val );

}

/***/-----*/

lcub ( cub )
float cub;
{
    return ( ( cub * cub * cub / 6 ) + ( cub * cub + 2 ) * ( cub + 4.0/3 ) );
}

/***/-----*/

hcub ( cub )
float cub;
{
    return ( ( cub * cub * cub / (-2) ) - ( cub * cub ) * ( 2.0/3 ) );
}

/***/-----*/

bicubic ( fx, fy, im )
float fx, fy;
float *im;
{
    float *imm;
    float ox,
        oy,
        val = 0;

    float xy[4][2] = {
        { 0, 0 },
        { 0, 0 },
    }
```

```
    { 0, 0 },  
    { 0, 0 }  
};
```

```
if ( fx >= 1 && fx < X-2 && fy >= 1 && fy < Y-2 ) {  
    ox = fx - ( int )fx;  
    oy = fy - ( int )fy;  
    xy[0][0] = - ( 1 + ox );  
    xy[0][1] = - ( 1 + oy );  
    xy[1][0] = - ( ox );  
    xy[1][1] = - ( oy );  
    xy[2][0] = - ( 1 - ox );  
    xy[2][1] = - ( 1 - oy );  
    xy[3][0] = - ( 2 - ox );  
    xy[3][1] = - ( 2 - oy );  
}  
  
imm = im + ( ( int ) fy - 1 ) * X + ( ( int ) fx - 1);  
val += ( *imm++) * lcub ( xy[0][0] ) * lcub ( xy[0][1] );  
val += ( *imm++) * hcub ( xy[1][0] ) * lcub ( xy[0][1] );  
val += ( *imm++) * hcub ( xy[2][0] ) * lcub ( xy[0][1] );  
val += ( *imm++) * lcub ( xy[3][0] ) * lcub ( xy[0][1] );  
  
imm = im + ( ( int ) fy ) * X + ( ( int ) fx - 1);  
val += ( *imm++) * lcub ( xy[0][0] ) * hcub ( xy[1][1] );  
val += ( *imm++) * hcub ( xy[1][0] ) * hcub ( xy[1][1] );  
val += ( *imm++) * hcub ( xy[2][0] ) * hcub ( xy[1][1] );  
val += ( *imm++) * lcub ( xy[3][0] ) * hcub ( xy[1][1] );  
  
imm = im + ( ( int ) fy + 1 ) * X + ( ( int ) fx - 1);  
val += ( *imm++) * lcub ( xy[0][0] ) * hcub ( xy[2][1] );  
val += ( *imm++) * hcub ( xy[1][0] ) * hcub ( xy[2][1] );  
val += ( *imm++) * hcub ( xy[2][0] ) * hcub ( xy[2][1] );  
val += ( *imm++) * lcub ( xy[3][0] ) * hcub ( xy[2][1] );  
  
imm = im + ( ( int ) fy + 2 ) * X + ( ( int ) fx - 1);  
val += ( *imm++) * lcub ( xy[0][0] ) * lcub ( xy[3][1] );  
val += ( *imm++) * hcub ( xy[1][0] ) * lcub ( xy[3][1] );  
val += ( *imm++) * hcub ( xy[2][0] ) * lcub ( xy[3][1] );  
val += ( *imm++) * lcub ( xy[3][0] ) * lcub ( xy[3][1] );  
  
return ( ( int ) val + 0.5);  
}
```

```
/***/-----***/
```

```
get_angle ( aw,  
            client_data,  
            cbs )  
Widget aw;  
char *client_data;  
XmSelectionBoxCallbackStruct *cbs;  
{
```



```
char *ang;
int a;

XmStringGetLtoR ( cbs->value,
                  XmSTRING_DEFAULT_CHARSET, &ang
                  );

a = atoi ( ang );

angle = (double)(a);
XtDestroyWidget ( XtParent ( aw ) );
proc_arr ();
}

/**-----**/

rotm ( p )
Widget p;
{
    Arg      arg_list[3];
    static Widget  dialog;

    XtSetArg ( arg_list[0],
              XmNselectionLabelString,
              XmStringCreateSimple (" Enter the angle for rotation : ")
              );

    XtSetArg ( arg_list[1],
              XmNautoUnmanage, False
              );

    dialog = XmCreatePromptDialog (
              p, "rotm",
              arg_list, 2
              );

    XtAddCallback ( dialog, XmNokCallback, get_angle, p );

    XtAddCallback ( dialog, XmNcancelCallback, XtDestroyWidget, NULL);

    XtManageChild ( dialog );

    XtPopup ( XtParent ( dialog ), XtGrabNone );
}

/**-----**/
```

```
/**-----**
```

Program : tst.c

Programmer : Kumar Chebrolu

This is the main program for the GUI. It will create the main window, menu bar, and callbacks for the different subroutines are called from here.

```
***-----**/
```

```
#include "tst.h"
```

```
char *titles[] = { "File",  
                  "Transformations",  
                  "Segmentation",  
                  "Representation",  
                  "Display" ,  
                  "Help",  
                  };
```

```
char *file_items[] = { "Open",  
                      "Save",  
                      "Save As",  
                      "Print .....",  
                      "Undo",  
                      "Quit",  
                      };
```

```
char *trans_items[] = { "Invert",  
                       "FFT",  
                       "Zoomin",  
                       "Rotation",  
                       };
```

```
char *rot_items[] = { "Biconstant Interpolation",  
                    "Bilinear Interpolation",  
                    "Bicubic Interpolation",  
                    };
```

```
char *seg_items[] = { "Pixel Classification",  
                    "Edge Detection",  
                    };
```

```
char *edge_items[] = { "Sobel",  
                      "Laplacian",  
                      "Prewitt" ,  
                      };
```

```
char *rep_items[] = { "Quad Tree Method",  
                    "Run Length Method" ,  
                    };
```

```
char *his_items[] = { "Auto_Threshold",
                    "Interactive",
                    "Equalization" ,
                    };

char *dis_items[] = { "Display Image",
                    "Initial Image",
                    "Pop",
                    };

char *quit_items[] = { "No",
                    "Yes",
                    };

/**-----**/

void printwidgettree ( w )
Widget w;
{
    while (XtParent (w)) {
        w = XtParent ( w);
        printf ("%s.", XtName (w));
    }
}

/**-----**/

void
call_proc ( parent,
           client_data,
           call_data)
Widget parent;
char *client_data;
XmAnyCallbackStruct *call_data;
{
    Widget par;
    int i;

    par = XtParent ( parent );

    for ( i = 0; i < XtNumber ( titles ) -1 ; i++ ) {
        if (!( strcmp ( titles[i], client_data ))) {
            help ( par, titles[i] );
            break;
        }
    }

    if (!( strcmp ( "Open", client_data )))
        opend( par );
    else if (!( strcmp ( "Save", client_data )))
        saved ( par );
    else if (!( strcmp ( "Save As", client_data )))
```

```
        saveas_proc ( par );
    else if (!( strcmp ( "Prin", client_data, 4 )))
        print_proc ( par );
    else if (!( strcmp ( "Undo", client_data, 4 )))
        undo ( );
    else if (!( strcmp ( "Yes", client_data, 3 )))
        exit (0);

    else if (!( strcmp ( "Invert", client_data, 6 )))
        invert ();
    else if (!( strcmp ( "FFT", client_data, 3 )))
        fast ( par );
    else if (!( strcmp ( "Zoomin", client_data, 6 )))
        zoomin ();
    else if (!( strcmp ( "Bicon", client_data, 5 ))) {
        rotn ( par );
        ROT_TYPE = 0;
    }
    else if (!( strcmp ( "Bilin", client_data, 5 ))) {
        rotn ( par );
        ROT_TYPE = 1;
    }
    else if (!( strcmp ( "Bicub", client_data, 5 ))) {
        rotn ( par );
        ROT_TYPE = 2;
    }

    else if (!( strcmp ( "Sob", client_data, 3 )))
        sobel();
    else if (!( strcmp ( "Lap", client_data, 3 )))
        laplacian ();
    else if (!( strcmp ( "Pre", client_data, 3 )))
        prewitt ();

    else if (!( strcmp ( "Run", client_data, 3 )))
        run_proc ( par );
    else if (!( strcmp ( "Quad", client_data, 4 )))
        quad_proc ( par );

    else if (!( strcmp ( "Auto", client_data, 4 )))
        auto_th ();
    else if (!( strcmp ( "Inter", client_data, 5 )))
        interactive ( parent );
    else if (!( strcmp ( "Equalizat", client_data, 9 )))
        hist_eq ( par );

    else if (!( strcmp ( "Disp", client_data, 4 )))
        loaded ();
    else if (!( strcmp ( "Init", client_data, 4 )))
        init ();

}

/****-----****/
```

```
Widget
menu_option ( name1,
              but )
char *name1;
Widget but;
{
    Widget      sub_button;
    Arg         arg_list[5];
    int         ar;
    XmString    name;

    name = XmStringCreateSimple ( name1 );
    ar = 0;

    XtSetArg ( arg_list[ar],
              XmNlabelString, name
              ); ar++;

    sub_button = XtCreateManagedWidget (
                name1,
                xmCascadeButtonGadgetClass, but,
                arg_list, ar
                );

    XtAddCallback ( sub_button, XmNactivateCallback, call_proc, name1 );

    return ( sub_button );
}

/****-----****/
```

```
Widget
create_pulldown_button ( name,
                        mbi1 )
XmString name;
Widget mbi1;
{
    Widget      button,
              cascade_but;
    Arg         arg_list[5];
    int         ar;

    ar = 0;
    button = XmCreatePulldownMenu (
                mbi1, name,
                arg_list, ar
                );

    ar = 0;
    XtSetArg ( arg_list[ar], XmNsubMenuId, button ); ar++;
    XtSetArg ( arg_list[ar], XmNlabelString, name ); ar++;

    cascade_but = XtCreateManagedWidget (
```

```
        "name",
        xmCascadeButtonGadgetClass, mbi1,
        arg_list, ar
    );

    return ( button );

}

/**-----**/

void
create_menu_items ( mbi )
Widget mbi;
{
    Widget    but,
             subbut,
             subbut1;

    XmString one,
              two;

    int    i,
           j, k;

    Arg arg_list[5];
    int ar;

    for ( i = 0; i < XtNumber ( titles ) ; i++ ) {
        but = create_pull_down_button (
            XmStringCreateSimple ( titles[i] ),
            mbi
        );

        if ( i == 0 ) {
            for ( j = 0; j < XtNumber ( file_items ); j++ )
                if ( j == 5 ) {
                    subbut1 = create_pull_down_button (
                        XmStringCreateSimple ( file_items[5] ),
                                                                but
                    );
                    for ( k = 0; k < XtNumber ( quit_items ); k++ )
                        subbut = menu_option ( quit_items[k], subbut1 );
                    else
                        subbut = menu_option ( file_items[j], but );
                }

            if ( i == 1 ) {
                for ( j = 0; j < XtNumber ( trans_items ); j++ )
                    if ( j == 3 ) {
                        subbut1 = create_pull_down_button (
                            XmStringCreateSimple ( trans_items[3] ),
                            but
                        );
                        for ( k = 0; k < XtNumber ( rot_items ); k++ )
```

```
        subbut = menu_option ( rot_items[k], subbut1 );
    }
    else
        subbut = menu_option ( trans_items[j], but );
    }
    if ( i == 2 ) {
        for ( j = 0; j < 2; j++ ) {
            subbut1 = create_pulldown_button (
                XmStringCreateSimple ( seg_items[j] ),
                but
            );
            if ( j == 0 ) {
                for ( k = 0; k < XtNumber ( his_items ); k++ )
                    subbut = menu_option ( his_items[k], subbut1 );
            }
            else {
                for ( k = 0; k < XtNumber ( edge_items ); k++ )
                    subbut = menu_option ( edge_items[k], subbut1 );
            }
        }
    }
    if ( i == 3 )
        for ( j = 0; j < XtNumber ( rep_items ); j++ )
            subbut = menu_option ( rep_items[j], but );
    if ( i == 4 )
        for ( j = 0; j < XtNumber ( dis_items ); j++ )
            subbut = menu_option ( dis_items[j], but );
    if ( i == 5 )
        for ( j = 0; j < XtNumber ( titles ); j++ )
            subbut = menu_option ( titles[j], but );
    }
}
```

/***/-----***/

```
main(argc, argv)
int argc;
char *argv[];
{
    Widget          form,
                  menubar,
                  frame;

    Arg            arg_list[5];
    int            ar;
    XtAppContext    app;

    top = XtAppInitialize (
        &app,
        "Tst",
        NULL, 0,
        &argc, argv,
```

```
        NULL, NULL
    );

form = XtCreateManagedWidget (
    "form",
    xmFormWidgetClass, top,
    NULL, 0
);

menubar = XmCreateMenuBar (
    form, "menubar",
    NULL, 0
);

XtManageChild ( menubar );

ar = 0;
XtSetArg ( arg_list[ar], XmNtopAttachment, XmATTACH_FORM ); ar++;
XtSetArg ( arg_list[ar], XmNleftAttachment, XmATTACH_FORM ); ar++;

XtSetValues ( menubar, arg_list, ar );

create_menu_items ( menubar );

draw = XtVaCreateManagedWidget (
    "draw",
    xmDrawingAreaWidgetClass, form,
    XmNtopAttachment, XmATTACH_WIDGET,
    XmNtopWidget, menubar,
    XmNleftAttachment, XmATTACH_FORM,
    XmNunitType, Xm1000TH_INCHES,
    XmNwidth, 6400,
    XmNheight, 6400,
    NULL
);

drawh = XtVaCreateManagedWidget (
    "drawh",
    xmDrawingAreaWidgetClass, form,
    XmNleftAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNunitType, Xm1000TH_INCHES,
    XmNx, 200,
    XmNwidth, 4800,
    XmNheight, 2400,
    NULL
);

XtRealizeWidget ( top );
XtAppMainLoop ( app );
}

/****-----****/
```



```
/**-----**
```

Program : tst.h

```
***-----**/
```

```
#include <Xm/Xm.h>
#include <Xm/Form.h>
#include <Xm/RowColumn.h>
#include <Xm/PanedW.h>
#include <Xm/CascadeB.h>
#include <Xm/CascadeBG.h>
#include <Xm/PushB.h>
#include <Xm/PushBG.h>
#include <Xm/DrawingA.h>
#include <Xm/Frame.h>
#include <Xm/SelectioB.h>
#include <Xm/MessageB.h>
#include <Xm/TextF.h>
#include <Xm/FileSB.h>
#include <Xm/DialogS.h>
#include <Xm/LabelG.h>
#include <Xm/MainW.h>
#include <Xm/Text.h>

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/Xatom.h>
#include <X11/Intrinsic.h>

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <ctype.h>

#include "bitmap/star"
#include "bitmap/star1"
#include "bitmap/help"

#define max( A, B ) ((A) > (B) ? (A) : (B))
#define pi 3.1415926
#define rad(theta) (theta)/ 180.0 * pi
#define conv(a) ((a) & 255 )
#define shift(x) (1<<x)

typedef struct {
    XtIntervalId id;
    int which;
    Pixmap pix1, pix2;
    Widget dialog;
    XtAppContext app;
```

```
    } TimeOutClientData;
char *Filename;
unsigned char *DIMAGE, *re_im;

float *IMAGE, *TMP_IMAGE, *PREV_IMAGE;

int X, Y;
int EN, oX, oY;
int FREQ[255];
int thres, threshold;
int IM_TYPE;
int ROT_TYPE;
int ZOOM_CNT;
int print_cnt;
int screen;

double angle;
FILE *lp, *fp, *fopen(), *sa;

GC gc;
GC *theCopyGC, *theXorGC;
Colormap colormap, mycolormap;
XColor alcolors, allcolors[256];
XImage *image;

Widget drawp, draw, top, drawh;
Pixmap pixmap;
int depth;

Status res;
Visual *vis_to_use;
XVisualInfo vinfo_ret;
int class;

static char *error[] = { "The Histogram Equalization can be done only to Gray Level Image,
    "The Run Length Method can be done only with Binary Image,
    "The Quad Tree Method can be done only with Binary Image,
    "The Zooming can be done two times only ... 0,
    "The FFT can be done only for the Binary Image,
    "The Quad Tree Representation can be done to the square image only,
    };
```

!Please don't change the executable file name, it should be "tst".

tst.height: 840

tst.width: 740

tst.background: lightblue

tst.x: 50

tst.y: 2

*dialog.x: 400

*dialog.y: 400

tst.form.menubar*FontList: 8x13bold

tst.form.menubar*name*FontList: 9x15bold

tst.form.menubar*background: grey

!tst.form.menubar*newd*background: LightSeaGreen

tst.form.menubar*newd*background: grey

tst.form.draw.background: grey

tst.form.drawh.background: black

tst.form*drawh*foreground: red

*pop*background: grey

```
#define star1_width 16
#define star1_height 16
static char star1_bits[] = {
    0x00, 0x00, 0x44, 0x11, 0x4a, 0x29, 0x54, 0x15, 0x68, 0x0b, 0x50, 0x05,
    0xbe, 0x3e, 0x40, 0x01, 0xbe, 0x3e, 0x50, 0x05, 0x68, 0x0b, 0x54, 0x15,
    0x4a, 0x29, 0x44, 0x11, 0x00, 0x00, 0x00, 0x00};
```

```
#define star_width 16
#define star_height 16
#define star_x_hot 7
#define star_y_hot 7
static char star_bits[] = {
    0x00, 0x00, 0x80, 0x00, 0x80, 0x00, 0x88, 0x08, 0x90, 0x04, 0xa0, 0x02,
    0x40, 0x01, 0x3e, 0x3e, 0x40, 0x01, 0xa0, 0x02, 0x90, 0x04, 0x88, 0x08,
    0x80, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```

#define help_width 64
#define help_height 64
static char help_bits[] = {
    0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0xfe, 0xff, 0x01, 0x00,
    0x00, 0x00, 0x00, 0xf8, 0xfc, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0x7c,
    0xf8, 0xff, 0x07, 0x00, 0x00, 0x00, 0x00, 0x3e, 0xf8, 0xff, 0x07, 0x00,
    0x00, 0x00, 0x00, 0x1f, 0xf0, 0xff, 0x0f, 0x00, 0x00, 0x00, 0x80, 0x0f,
    0xe0, 0xff, 0x1f, 0x00, 0x00, 0x00, 0x80, 0x0f, 0xc0, 0xff, 0x3f, 0x00,
    0x00, 0x00, 0xc0, 0x07, 0xc0, 0xff, 0x3f, 0x00, 0x00, 0x00, 0xe0, 0x03,
    0x80, 0xff, 0x7f, 0x00, 0x00, 0x00, 0xf0, 0x01, 0x00, 0xff, 0xff, 0x00,
    0x00, 0x00, 0xf8, 0x00, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x00, 0xf8, 0x00,
    0x00, 0xfe, 0xff, 0x01, 0x00, 0x00, 0x7c, 0x00, 0x00, 0xfc, 0xff, 0x03,
    0x00, 0x00, 0x3e, 0x00, 0x00, 0xf8, 0xff, 0x07, 0x00, 0x00, 0x1f, 0x00,
    0x00, 0xf0, 0xff, 0x0f, 0x00, 0x80, 0x0f, 0x00, 0x00, 0xf0, 0xff, 0x0f,
    0x00, 0xc0, 0x07, 0x00, 0x00, 0xe0, 0xff, 0x1f, 0x00, 0xc0, 0x07, 0x00,
    0x00, 0xc0, 0xff, 0x3f, 0x00, 0xe0, 0x03, 0x00, 0x00, 0x80, 0xff, 0x7f,
    0x00, 0xf0, 0x01, 0x00, 0x00, 0x80, 0xff, 0x7f, 0x00, 0xf8, 0x00, 0x00,
    0x00, 0xf8, 0xff, 0xff, 0xff, 0xff, 0x0f, 0x00, 0x00, 0x08, 0x00, 0x00,
    0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00,
    0x00, 0x08, 0x21, 0x00, 0x02, 0x00, 0x08, 0x00, 0x00, 0x08, 0x21, 0x00,
    0x02, 0x00, 0x08, 0x00, 0x00, 0x08, 0x21, 0x3e, 0x02, 0x3e, 0x08, 0x00,
    0x00, 0x08, 0x3f, 0x02, 0x02, 0x42, 0x08, 0x00, 0x00, 0x08, 0x21, 0x02,
    0x02, 0x42, 0x08, 0x00, 0x00, 0x08, 0x21, 0x02, 0x02, 0x42, 0x08, 0x00,
    0x00, 0x08, 0x21, 0x3e, 0x02, 0x3e, 0x08, 0x00, 0x00, 0x08, 0x21, 0x02,
    0x3e, 0x02, 0x08, 0x00, 0x00, 0x08, 0x00, 0x02, 0x00, 0x02, 0x08, 0x00,
    0x00, 0x08, 0x00, 0x02, 0x00, 0x02, 0x08, 0x00, 0x00, 0x08, 0x00, 0x02,
    0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00,
    0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00,
    0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x08, 0x00, 0x00,
    0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0xf8, 0xff, 0xff,
    0xff, 0xff, 0x0f, 0x00, 0x00, 0x00, 0xf8, 0x80, 0xff, 0x7f, 0x00, 0x00,
    0x00, 0x00, 0x7c, 0x00, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x00,
    0xfe, 0xff, 0x01, 0x00, 0x00, 0x00, 0x3e, 0x00, 0xfe, 0xff, 0x01, 0x00,
    0x00, 0x00, 0x1f, 0x00, 0xfc, 0xff, 0x03, 0x00, 0x00, 0x80, 0x0f, 0x00,
    0xf8, 0xff, 0x07, 0x00, 0x00, 0xc0, 0x07, 0x00, 0xf0, 0xff, 0x0f, 0x00,
    0x00, 0xe0, 0x03, 0x00, 0xf0, 0xff, 0x0f, 0x00, 0x00, 0xe0, 0x03, 0x00,
    0xe0, 0xff, 0x1f, 0x00, 0x00, 0xf0, 0x01, 0x00, 0xc0, 0xff, 0x3f, 0x00,
    0x00, 0xf8, 0x00, 0x00, 0x80, 0xff, 0x7f, 0x00, 0x00, 0x7c, 0x00, 0x00,
    0x80, 0xff, 0x7f, 0x00, 0x00, 0x3e, 0x00, 0x00, 0x00, 0xff, 0xff, 0x00,
    0x00, 0x3e, 0x00, 0x00, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x1f, 0x00, 0x00,
    0x00, 0xfc, 0xff, 0x03, 0x80, 0x0f, 0x00, 0x00, 0x00, 0xfc, 0xff, 0x03,
    0xc0, 0x07, 0x00, 0x00, 0x00, 0xf8, 0xff, 0x07, 0xe0, 0x03, 0x00, 0x00,
    0x00, 0xf0, 0xff, 0x0f, 0xe0, 0x03, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x1f,
    0xf0, 0x01, 0x00, 0x00, 0x00, 0xe0, 0xff, 0x1f, 0xf8, 0x00, 0x00, 0x00,
    0x00, 0xc0, 0xff, 0x3f, 0x7c, 0x00, 0x00, 0x00, 0x00, 0x80, 0xff, 0x7f,
    0x3e, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff];

```

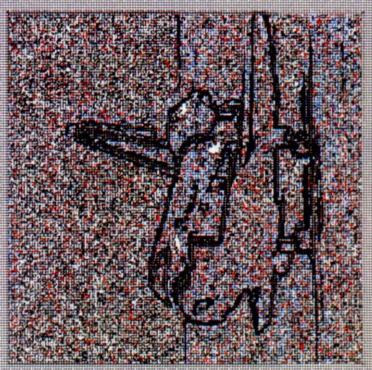
```
EXEC= tst
CC= gcc
CFLAGS= -O -s
LIBS= -lXm -lXt -lX11 -lm
OBJECTS= tst.o      menu.o      file.o      edge.o      hist.o      rot.o draw.o las.o rep.o en

$(EXEC): $(OBJECTS)
$(CC) -o $(EXEC) $(OBJECTS) $(LIBS)
```

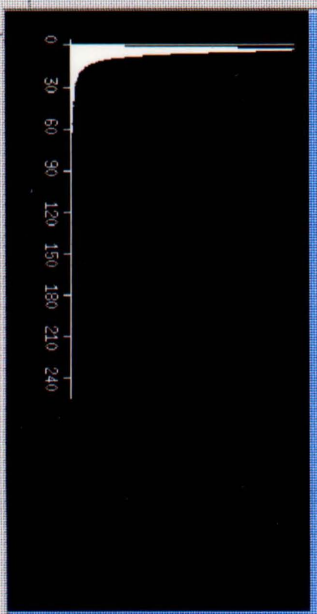

Figures

File Transformations Segmentation Representation Display Help

1st



0 30 60 90 120 150 180 210 240



xterm

```
Filename = /home/earth/grad/srinivas/the/3/plc/shut  
tle.256  
GRAY IMAGE
```

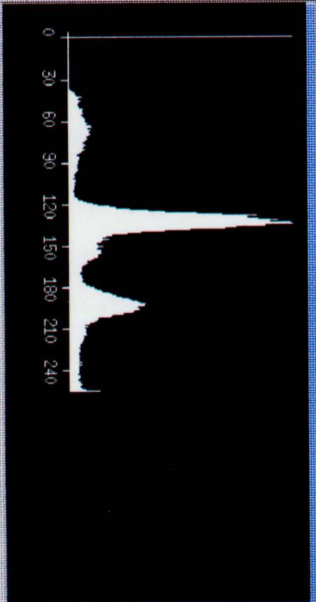
xterm

```
worthern% tekdump
```

I

File Transformations Segmentation Representation Display Help

1st

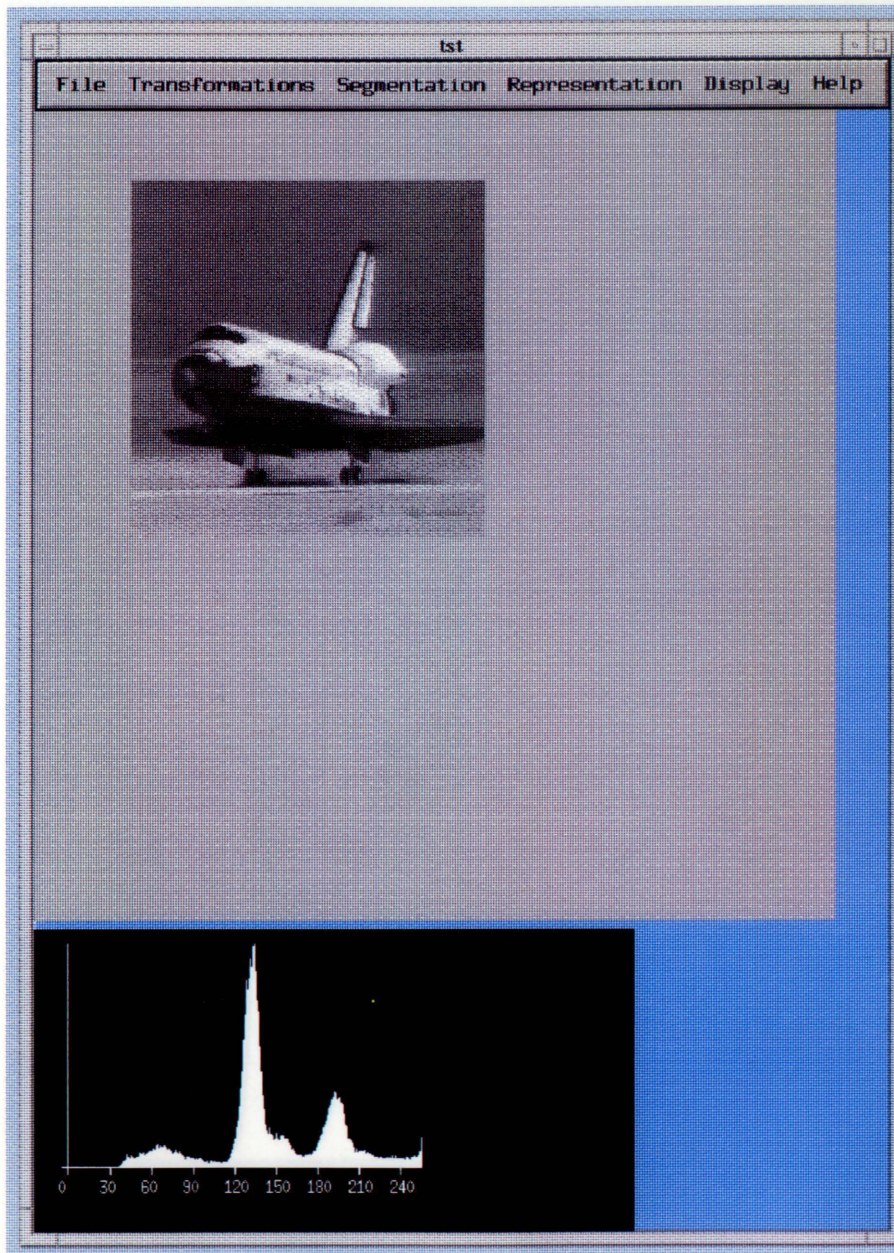


x1am

Filename = /home/earth/grad/sr1nivas/the/3/pic/shut
t1e.256
GRAY IMAGE

worthernx tekdump

I



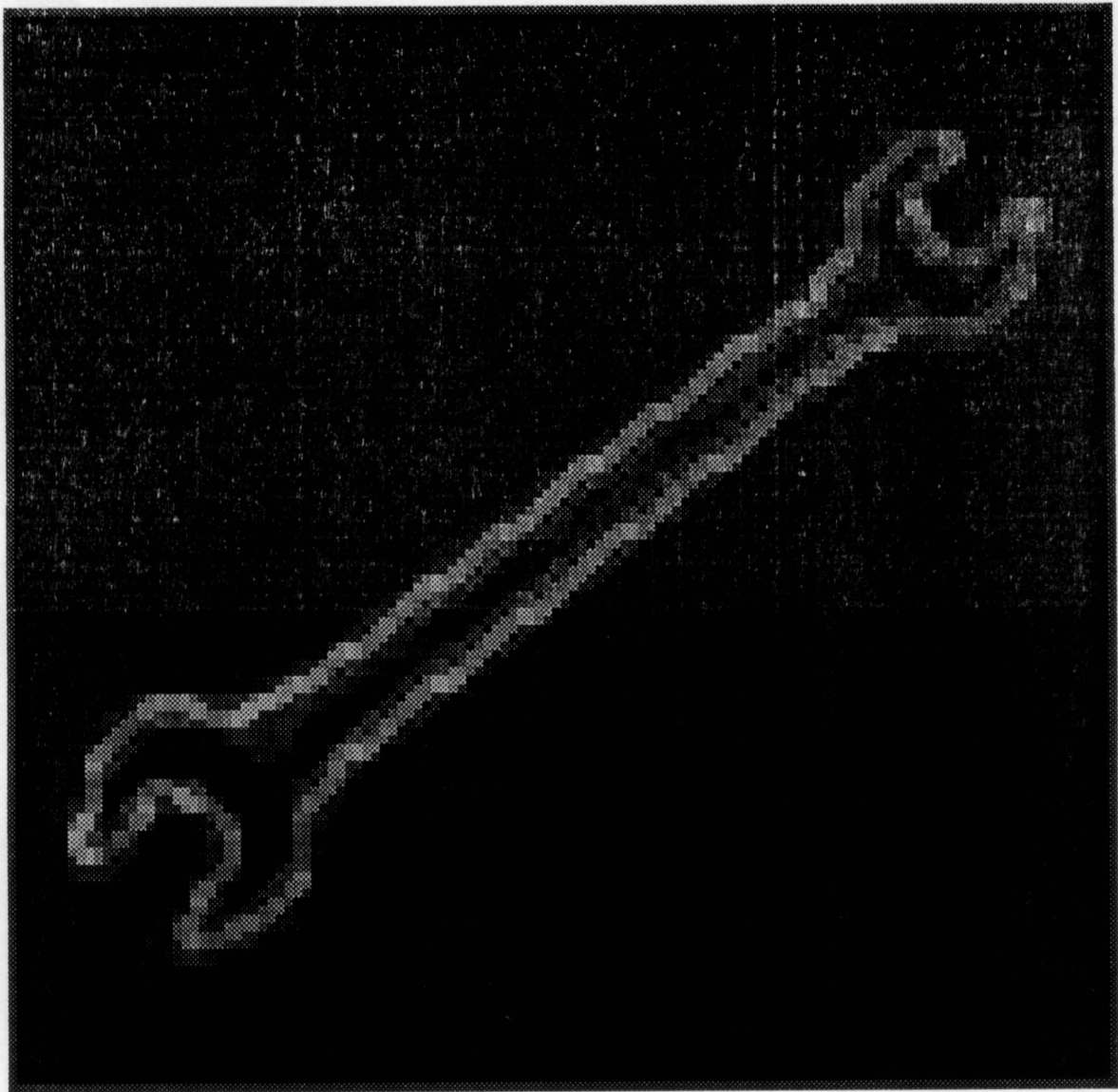
xterm

```
lisa% tst
Filename = /home/earth/grad/srinivas/the/3/pic/shuttle.256
GRAY IMAGE
□
```

Filename = /home/earth/grad/srinivas/the/3/pic/shuttle.256
GRAY IMAGE

worthiness% tekdump

I





Bibliography

- [1] Rosenfeld. A., and Kak.C.A, " *Digital Picture Processing, Volume 1 & 2*", Academic press, 1982.
- [2] Pratt. W. K.,, " *Digital Image Processing*", John wiley, 1978.
- [3] Gonzalez. R. C., and Wintz. P.,, " *Digital Image Processing*", Addison-Wesley, 1987.
- [4] Frank Y. Shih and O. Robert Mitchell, " *Decomposition of Gray Scale Morphological Structuring Elements*", Presented at IEEE workshop on Computer Vision, Florida, Nov. 30- Dec. 2, 1987
- [5] Dan Heller, " *Motif Programming Manual for OSF/Motif 1.1, Volume 6*", O'Reilly & Associates, Inc, 1991.
- [6] Jones O., " *Introduction to the X Window System*", Prentice Hall, 1989.
- [7] " *OSF/MOTIF Programmer's Guide*", Open Software Foundation, 1991.
- [8] Nye, A., " *Xlib Programming Manual, Volume 1*", O'Reilly & Associates, Inc, 1991.
- [9] Young, D., " *The X Window System: Applications and Programming with Xt (Motif Version)*", Prentice Hall, 1989.