1993

Digital darkroom| A digital image processing software package

SongTee Jeffrey Heng
The University of Montana

Let us know how access to this document benefits you.
Follow this and additional works at: https://scholarworks.umt.edu/etd

Recommended Citation
Heng, SongTee Jeffrey, "Digital darkroom| A digital image processing software package" (1993). Graduate Student Theses, Dissertations, & Professional Papers, 2071.
https://scholarworks.umt.edu/etd/2071

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.
DIGITAL DARKROOM -
A DIGITAL IMAGE PROCESSING SOFTWARE PACKAGE

SongTee Jeffrey Heng
B.S., University of Montana, 1987

Presented in partial fulfillment of the requirements
for the degree of
Master of Science
University of Montana
1993

Approved By

Chairman, Board of Examiners

Dean, Graduate School

Date
Digital Darkroom
- A Digital Image Processing Software Package

Director: Dr. Youlu Zheng

The Digital Darkroom presented is an electronic photographic darkroom whose results can be interactively modified and viewed in a matter of seconds, giving real-time feedback. The Digital Darkroom simulates the environment of a photographic darkroom and provides tools like color filters, dodge and burn capabilities, Sabattier effect and other darkroom techniques. Color selection and editing in the Digital Darkroom are intuitive. By using digital signal processing techniques, the Digital Darkroom can produce effects like edge detection/enhancement which are very difficult to fulfil in a traditional photographic darkroom.

Several design and implementation issues and problems that are present in developing such a graphics software package are discussed. The specifications and design of the Digital Darkroom were done using techniques suitable for a Graphical User Interface (GUI) application.
# Table of Contents

Abstract............................................................................................................ ii
Table of Contents.......................................................................................... iii
List of Illustrations......................................................................................... iv
1. Background................................................................................................. 1
2. Image Processing for Digital Darkroom.................................................... 16
3. Formal Specifications.................................................................................. 47
4. Design of the Digital Darkroom............................................................... 50
5. Implementation.......................................................................................... 73
6. Conclusion.................................................................................................. 79

Bibliography..................................................................................................... 81
Acknowledgments............................................................................................ 83
| Figure 1-1. | Linear colormap for an 8-bit frame buffer | 7 |
| Figure 1-2. | Hierarchy of visual classes | 10 |
| Figure 1-3. | RGB Coordinate System | 11 |
| Figure 1-4. | CMY Color Model | 13 |
| Figure 1-5. | Top View of the HSV hexagon | 14 |
| Figure 1-6. | HSV Color Model | 15 |
| Figure 2-1. | Histogram of Original Image | 18 |
| Figure 2-2. | Graph for sliding histogram by +96 brightness levels | 19 |
| Figure 2-3. | Histogram after sliding +96 brightness levels | 19 |
| Figure 2-4. | Graph for sliding histogram by -48 brightness levels | 20 |
| Figure 2-5. | Histogram after sliding -48 brightness levels | 20 |
| Figure 2-6. | Histogram of Original Image | 21 |
| Figure 2-7. | Graph for stretching histogram by a factor of 2 | 22 |
| Figure 2-8. | Histogram stretched by a factor of 2 | 22 |
| Figure 2-9. | Graph for compressing histogram by a factor of 2 | 23 |
| Figure 2-10. | Histogram compressed by a factor of 2 | 23 |
| Figure 2-11. | Graph for Binary Contrast | 24 |
| Figure 2-12. | Brightness level probability function of a light image | 30 |
| Figure 2-13. | Brightness level probability density function of a dark image | 30 |
| Figure 2-14. | Uniform probability density | 31 |
| Figure 2-15. | Original Histogram | 31 |
| Figure 2-16. | Equalized Histogram | 32 |
| Figure 2-17. | Graph for Complementing Image | 33 |
| Figure 2-18. | Graph for Brightness Selective Solarization | 33 |
| Figure 2-19. | Spatial Convolution | 36 |
| Figure 2-20. | Effects of High Pass Filtering | 39 |
| Figure 4-1. | Motif Application and Toolkits Relationship | 52 |
| Figure 4-2. | Basic Widget Class Hierarchy | 53 |
| Figure 4-3. | Primitive Class Widgets | 54 |
| Figure 4-4. | Shell Widgets | 54 |
| Figure 4-5. | Manager Widgets | 55 |
| Figure 4-6. | Dialog Widgets | 55 |
| Figure 4-7. | Overall Screen Design of the Digital Darkroom | 56 |
| Figure 4-8. | Complete Menu Tree of the Digital Darkroom | 57 |
| Figure 4-9. | Zoom Dialog | 61 |
| Figure 4-10. | Color Edit Dialog | 62 |
| Figure 4-11. | Widget Hierarchy of the Digital Darkroom | 63 |
| Figure 4-12. | Widget Hierarchy of Pulldown Menus | 64 |
| Figure 4-13. | Widget Hierarchy of the Color Edit Dialog | 64 |
| Figure 4-14. | Widget Hierarchy of the Zoom Dialog | 65 |
| Figure 4-15. | Level 0 Data Flow Diagram of the Digital Darkroom | 66 |
| Figure 4-16. | Level 1 DFD of Process Image | 67 |
| Figure 4-17. | Level 1 DFD of Add Special Effects | 68 |
| Figure 4-18. | Level 1 DFD of Modify Geometry | 69 |
| Figure 4-19. | Level 1 DFD of Color Enhance | 70 |
| Figure 4-20. | Level 1 DFD of Display Image Info | 71 |
Chapter One

Background

1.1. Project Overview

The *Digital Darkroom* is a digital image processing software package that simulates a photographic darkroom and also enables the user to create effects and enhancements that are beyond those possible in a traditional photographic darkroom. Photographers can use the *Digital Darkroom* like a color enlarger with its various color filters. In a photographic darkroom, a photographer would experiment with the various enlarger filters to achieve the desired preferences/effects and would only be able to view the results after printing them out - an expensive and laborious approach. With *Digital Darkroom* the photographer can experiment with the individual color components and the various special effects to achieve his/her desired effects in an interactive, quicker, cheaper manner and be able to do far more than a traditional photographic darkroom.

One of the typical uses of the *Digital Darkroom* is in correcting color temperature. Wrong color temperature, caused by using a daylight film in a tungsten (incandescent) lit surrounding or using a tungsten film in daylight, can be easily corrected by the *Digital Darkroom*. The user can interactively choose the color change that is needed to correct the look of the picture or can instruct the *Digital Darkroom* to correct the color temperature automatically. To make a color picture more brilliant (i.e. blues to be bluer,
and reds to be redder), the user can once again interactively choose the more saturated colors for the picture. Uses of the Digital Darkroom are not limited to photographers only. Desktop publishers can use the Digital Darkroom to enhance a picture before the final print. Artists can manipulate digital images to their artistic hearts’ content using Digital Darkroom.

The print quality of color printers priced under $5,000 does not approach that of a 35mm slide. However, there are expensive high resolution film printers that produce high quality photographic prints by writing digital images onto 35mm film. Such state-of-the-art machines are used at Disney Studios and may cost tens of thousands of dollars. As hardware technology advances and prices plummet, color printers that produce acceptable color richness and quality should become readily available and affordable to the masses. Long awaited color laser printers priced below $10,000 from Hewlett Packard, QMS, Apple and Tektronix are beginning to appear on the market.

The Digital Darkroom uses a Graphical User Interface (GUI) which gives the user a friendly and intuitive interface that is especially important in such an interactive application. It uses X11R4 graphics for its graphics capabilities and Motif as its GUI.

1.1.1 Hardware

The graphics hardware needed for the Digital Darkroom must be able to display at least 256 simultaneous colors from a palette of 16 million colors to satisfactorily display color images without too much ‘contouring’. Contouring occurs when there are
insufficient colors to display the actual image, producing an image that has very sudden
color changes and patches without smooth color graduations or shading. Desirable video
resolution would be at least 1024 x 768 pixels.

1.2.1 Motif

Motif is a consistent and user-friendly GUI developed by the Open Software
Foundation (OSF) that runs under X11. It also provides a complete toolkit that enables
the programmer to give a distinct, consistent and intuitive 'look and feel' to the software.
Graphics primitives and capabilities are not provided by Motif since these primitives are
available under X11. Low-level X11 routines make up Xlib, the X11 C library. Motif
applications typically access the Motif library, Xt Toolkit Intrinsics, and Xlib. These
three libraries complement each other in providing a complete graphics and windowing
system.

1.2.2 Xt Intrinsics

The Xt Intrinsics are routines written using Xlib functions and deal with the X11
windowing system in terms of widgets. Widgets are basically objects in the windowing
systems, like dialogs, scrollbars, menus and pushbuttons. This toolkit provides the
programmer with high-level means of constructing the application by viewing the system
as a collection of widgets, rather than pixels and lines.
1.2.3 X11 Windowing System

The X11R4 windowing system is the maturing child of the X windowing system that has come of age. X was first developed by Massachusetts Institute of Technology (MIT) and Digital Equipment Corporation (DEC) in 1984. Efforts to improve X resulted in the latest version, X11 Release 5. Like most GUI systems, X11 uses the event-driven paradigm.

In traditional software, the user is led through a step-by-step sequence of questions and the application appropriately reacts to the answers given by the user in response to those questions. In the GUI world, there is a numerous variety of user input events and internally generated actions occurring at any time. Such actions include mouse inputs, keyboard inputs, opening, closing, resizing, and moving of the various windows on the screen. The event-driven paradigm handles this myriad of actions efficiently. An event-driven program waits in an infinite loop for the occurrence of each type of event and then appropriately responds to that event. Upon completion of that response, the program returns to wait and respond to the next event. This is quite different from the typical sequential question-and-answer type of user interface used in a traditional character-oriented system. The following is a sample X11 code to illustrate the event-driven nature of an X11 application:

```c
while ( True )
{
    XNextEvent( display, &event ); /* Get the event */
    switch ( event.type )
    {
      
        
    }
```
case ButtonPress: /* A mouse button was pressed */
    /*
    * Action in response button press event
    */
    break;

case KeyPress: /* A key was hit */
    /*
    * Action in response to the keyboard action
    */
    break;

case MotionNotify: /* The mouse was moved */
    /*
    * Action in response to mouse movement
    */
    break;

case CreateNotify: /* A window was created */
    /*
    * Action in response to window creation
    */
    break;

case ConfigureNotify: /* A window’s size changed */
    /*
    * Action in response to window size change
    */
    break;
    :
    :

1.2.3.1. X Color Display Concepts and Implementation

Most workstations deal with color differently. Since X was designed with portability in mind, it uses a color model that shields the user from the underlying display hardware of the workstations. This color model is flexible enough to handle most color capabilities that are expected of graphics workstations. The main features of this color scheme are colormaps and virtuals.
1.2.3.1.1 Colormaps

Color displays are usually based on the RGB (Red, Green, Blue) color model. The physical display screen is made up of an array of dots of phosphor and can produce any color and shades by the mixture of the three primary colors Red, Green and Blue, and varying the intensity of each primary color. Each pixel on the screen is made of the three phosphor dots in a triangular fashion. The variation in the intensity of each of these three phosphor dots produces the different colors and shades.

For monochrome (bi-level) displays, one bit per pixel is sufficient to handle the black and white images. However, to handle multiple shades of colors, multiple bits must be used to handle one pixel. The bits of color information stored in the frame buffer control the intensity of each of the three electron beams.

There are basically two methods by which the bits in the frame buffer is used to manage color:

1. To directly control the intensity levels of the Red, Green, and Blue phosphor beams.

2. To be translated into the colors using a color lookup table, also known more commonly as a colormap.

The second method is most commonly used. In such a color mapping scheme, each pixel value is used as an index to the colormap. Each entry of the color stores the separate intensity levels of each of the three primary colors. The number of possible
colors that a color display adapter is capable of displaying is determined by the number of bits in the colormap. Thus, if eight bits were available for each primary color, then the number of possible colors available (the color palette) is $(2^8)^3$ which is approximately 16 million. However, the number of colors that can be displayed simultaneously on the screen is usually much lesser for moderately priced displays and is determined by the number of planes of the frame buffer. An n-plane system can index $2^n$ colormap entries and is thus capable of displaying $2^n$ simultaneous colors (see Figure 1-1). The size of the frame buffer is primarily determined by the amount of video memory available on the graphics display adapter. As technology advances and price of hardware plummets, frame buffers that are capable of displaying $2^{24}$ colors simultaneously are becoming increasingly common.

![Figure 1-1. Linear colormap for an 8-bit frame buffer](image)

The color mapping or color lookup scheme for high performance (in terms of color
capability and resolution) displays is usually different from that of a moderately priced displays that are capable of displaying only $2^8$ simultaneous colors. For displays that need to produce a bigger gamut of colors, like 16 million ($2^{24}$) colors, a similar scheme of using a colormap of linear, contiguous entries is not elegant nor manageable. A colormap of 16 million ($2^{24}$) entries will be needed to implement the above linear map scheme. For these high performance displays, the respective 8 bit color values can be used to control the RGB color display directly.

### 1.2.3.1.2 Visuals

A class of X11 data type known as *visuals* describes the characteristics and capabilities of the graphics display hardware. There may be more than one visual on a particular screen and each visual describes the characteristics of each of the colormaps available on the system. For example, a color system may have both color and monochrome visuals. The main characteristics of a colormap can be categorized by the visual type of the display which affects the accessibility and the method of accessing the colormap. The different types of visuals and their accessibilities are listed below:

<table>
<thead>
<tr>
<th>Colormap Type</th>
<th>Visual Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono/Gray</td>
<td>GrayScale</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>StaticGray</td>
<td>R</td>
</tr>
<tr>
<td>Single Index</td>
<td>PseudoColor</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>StaticColor</td>
<td>R</td>
</tr>
<tr>
<td>Decomposed Index</td>
<td>DirectColor</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>TrueColor</td>
<td>R</td>
</tr>
</tbody>
</table>
Visuals provide a means of utilizing the color resources of the screen, providing the use of different visuals for different applications. A read/write colormap may be used as read-only. On color systems that have DirectColor visuals, TrueColor visuals are also available since the main difference between these two visuals is their accessibilities. In fact, DirectColor visuals can provide all the other visuals since the DirectColor visual type is the superclass of all the other visual types (Figure 1-3). As listed above, PseudoColor, DirectColor, and Grayscale visuals are read/write while StaticColor, TrueColor and StaticGray visuals are read-only. Applications that utilize read-only visuals have the advantage of being able to share the colormap among other clients. The colors in the colormap is predictable since no client is able to change the colormap. The disadvantage is that a specific color needed by an application may not be available in the colormap. PseudoColor and DirectColor visuals are the most useful since they may also be used as other types of visuals.
1.3. Color Models

Electromagnetic waves, like light, have a property known as wavelength. Lights of different wavelength give rise to the perception of different colors. Each wavelength is perceived by humans as a color ranging from red at 700 nanometers to violet at 400 nanometers. However, describing color using wavelength is difficult since most objects or colored lights constitute a band of wavelengths, not just one dominant wavelength. Color models more accurately and conveniently characterize colors. Several color models exist to explain and describe specific colors, with each model being more suited to a particular class of application than the other. For example, the CMY color model is more suited for plotter or printer applications than the RGB color model.

1.3.1. RGB Color Model
As the name of the model implies, the RGB color model specifies color in terms of the three primary colors Red, Green and Blue. This model is commonly used in graphics applications mainly because a color monitor/display uses the red, green and blue electron guns to produce a specific color.

The RGB model is based on the addition of the three primary color components of light, with white light being the addition of the three primary colors at their maximum intensities. This model is a three-dimensional space model with Red being the X-axis, the Green being the Y-axis and Blue the Z-axis (Figure 1-4). A color is thus represented by a triple in this RGB three-dimensional space. Usually the axes are normalized to 1 as their maximum values. For example, black will be at location (0,0,0) and white at (1,1,1). However, for a brightness scale that ranges from 0 to 255, the value of 255 is used instead of 1 for 8-bit frame buffers.

![Figure 1-3. RGB Coordinate System](image)

11
1.3.2. CMY Color Model

The RGB color model is not as suited to specifying colors for color hard-copy devices since these devices work with colored pigments/paints (reflected light) rather than colored light (incident light). Colors produced by mixing colored pigments are more suitably described by a subtractive model than an additive one. One such subtractive color model is the CMY color model.

The primary colors in this model are Cyan, Magenta and Yellow. For a colored object to be perceived as red, no cyan component must be reflected by the object, that is, cyan is total absorbed by the object. In other words, cyan is subtracted from white to give red. To get red paint, we mix magenta paint and yellow paint. The yellow paint will absorb the blue light, and the magenta paint will absorb the green light, reflecting only the red light, giving the perception of a red object. Red is thus the complementary color of cyan, green the complementary color of magenta and blue the complementary of yellow.

Like the RGB model, the CMY color model is described by a three-dimensional space with cyan, magenta and yellow as the three axes. Unlike the RGB color model, (0,0,0) is white and (1,1,1) is black since the CMY color model is a subtractive model (Figure 1-5). A color printer uses mainly three kinds of colored paints - cyan, magenta and yellow. By mixing these three colored paints together, most colors of the spectrum can be produced.
1.3.3. HSV Color Model

Since the RGB and CMY color models correspond directly to the primary and complementary colors of the spectrum, they are efficient and natural when dealing with the RGB electron guns of the color monitor or the CMY color paints of the color printer. However, these models are not intuitive to a user who needs to specify or experiment with colors. Describing a color in terms of red, green or blue is difficult for an artist who is more used to tints, shades and tones. A more intuitive user interface for a color graphics application needs to separate colors into tints, shades and tones. The need for this separation gives rise to the HSV model. The three parameters of this model are known as Hue(H), Saturation(S) and Value(V) (or intensity).

The HSV color model is derived from the RGB color model. By rotating the RGB
cube so that the corner of the cube representing white is at the front, a picture of a hexagon with all the different hues at its corners can be obtained (Figure 1-6).

![Hexagon with different hues](image)

**Figure 1-5. Top View of the HSV hexagon**

This color model has the shape of a hexcone, a hexagonal cone or pyramid, as illustrated in Figure 1-7. Hue (H) is represented by the angle measured about the central axis of the hexcone. The hues therefore ranges from 0 degrees to 360 degrees. Red is at 0 degrees, green at 120 degrees, blue at 240, and so on. Each hue is separated by a 60-degree angle and complementary colors are at 180 degrees apart. Saturation (S) can be thought of as color purity, with S=1 being the purest, and S=0 being no color at all, just gray shades. Pure colors are those that are not diluted by the addition of white. Values on the central axis of the hexcone represent gray shades since S=0 along the central axis. Hence white is at V=1 and S=0, the center point of the top hexagonal face of the hexcone. H is undefined when S=0 since hue is irrelevant when referring to gray shades. Black occurs when V=0 and S=0, represented by the apex of the hexcone.
Intensity or value (V) measures the total light of the color, with V=1 being the maximum and V=0 being black. At V=1 and S=1, hues have maximum purity. Pure red is therefore at V=1, S=1 and H=0. Adding white to a color decreases its saturation (S), making the color less pure.

Figure 1-6. HSV Color Model
Chapter Two

Image Processing For Digital Darkroom

2.1 General Concepts

Image Processing simply means manipulating an image/picture to suit one’s needs or preferences. Adjusting the contrast, tint, or brightness on a television set can be considered rudimentary image processing. Digital Image Processing converts an image into digital values at every point of the image. The video resolution of the converted image is limited by the graphics hardware and software used. Image processing can normally be categorized into the following processes based on the algorithmic nature of the processes:

1. Point Processing
2. Area Processing
3. Frame Processing

The frequency of the different brightness values in terms of pixels may be presented in the form of a histogram. The histogram is plotted with the number of the pixels on the y-axis and the discrete brightness values ranging from 0 to 255 (for an 8-bit frame buffer) on the x-axis. This form of statistical presentation yields information about the image which is useful for image processing, e.g. brightness investigation, contrast
2.2. Point Processing

Point processes deal with the image pixel by pixel, processing each pixel individually and independent of its surrounding pixels. Each pixel of the image is examined and modified individually, and the new value placed back into the same location. So, for an input image $I$, point processing transforms $I$ into the output image $O$ through the following transformation $T$:

$$O(x,y) = T(I(x,y))$$

indicating that a pixel at location $(x,y)$ would be modified by transformation $T$, and returned to location $(x,y)$.

Changing the contrast, brightness, and colors of an image are examples of point processing. By appropriately modifying the color components of each pixel of the image, a picture can be made to have a desired tint, like a picture during sunset which has an orange tint. The brightness value of each pixel of the image can also be modified to create a darker or brighter picture. In addition, by modifying the brightness value of each pixel logarithmically, we can brighten the dark areas of a picture normally caused by the non-linearity of scanning devices. The user can also achieve the photographic effects of 'burning' and 'dodging' by decreasing or increasing the brightness values of selective area of pixels respectively.
2.2.1. Brightness Adjustment

An image may be brightened or darkened by increasing or decreasing the brightness values of the pixels in the image. The brightness of a monochrome pixel is simply the pixel's gray scale level and the brightness of a color pixel is its V value as described by the HSV color model (See Color Image Processing Section 2.4). Given an image's histogram as illustrated in the following,

![Histogram of Original Image](image)

Figure 2-1. Histogram of Original Image

the image's brightness may be increased or decreased by adding or subtracting a constant to each of the pixels' brightness value, effectively sliding the histogram to the right or the left respectively. Modifying the brightness values may cause the values to go beyond the brightness range of the image (brightness saturation). In such cases, these values are clipped to the minimum or maximum brightness values appropriately. However, if most of the pixels are near the extremities of the brightness range, the image will lose details if the histogram slide clipped most of the pixels' values. By examining the histogram of
the original image before processing, one can avoid throwing the image into saturation by choosing a value for the histogram slide that will not cause the maximum brightness value of the image to exceed the maximum value of the horizontal axis (Maximum White) or the minimum brightness value of the image to go below the origin of the horizontal axis (Maximum Black).

Figure 2-2. Graph for sliding histogram by +96 brightness levels.

Figure 2-3. Histogram after sliding +96 brightness levels
2.2.2 Contrast Adjustment

The contrast of an image refers to the difference between the brightest and the lowest darkest parts of the image. A high contrast image is characterized by two major clusters of pixels at the two extreme ends of the brightness range. Low contrast occurs when the pixels are clustered around a single narrow region of the brightness range. These patterns
show up clearly on a histogram. The histogram may be used as a tool to further investigate and enhance the image.

Increasing or decreasing the contrast is achieved by multiplying or dividing the pixels' brightness values by a constant, thereby stretching or compressing the histogram along the horizontal axis (brightness values). Care should be taken when stretching/compressing the histogram to avoid excessive brightness saturation.

Figure 2-6. Histogram of Original Image
Figure 2-7. Graph for stretching histogram by a factor of 2

Figure 2-8. Histogram stretched by a factor of 2
2.2.3 Contrast Enhancement

An image of poor contrast may be enhanced by combining the techniques of histogram stretching and sliding. Poor contrast is characterized by a histogram that indicates the majority of pixels amassing on a particular region of the brightness range.
The contrast may be improved by simply stretching the histogram as outlined earlier. However, a better improvement is to first slide the histogram so that the darkest pixels are at the full black area and then stretch the histogram so that the histogram occupies the entire brightness range.

Sometimes image processing techniques are applied to images to figure out particular patterns or lettering that are obscured due to poor contrast. For example, if the contrast of a traffic 'STOP' sign is very poor, one can hardly make out the white 'STOP' letters on the red background. To enhance such a low contrast picture to bring out the lettering, a process known as binary contrast enhancement may be applied. A cut-off value is first selected. Any brightness level above this cut-off or threshold value is set to the maximum value of the brightness scale (full white), and brightness levels below the cut-off level are set to the minimum brightness value (full black). The correct threshold value will set the pixels that make up the lettering to go full white, and the other pixels to go full black for the 'STOP' sign where the lettering are of lighter color than the

![Figure 2-11. Graph For Binary Contrast](image-url)
background. For patterns or letters that are darker than the surrounding background, the reverse happens - black letters on white background. The trick is thus selecting the correct threshold value so that the lettering is visible. By allowing the user to interactively vary the threshold value and viewing the result immediately makes the selection of the correct threshold value a much less daunting task. Interactively 'playing' with the threshold value may even be fun, especially at the point when the lettering becomes visible! The graph for the binary contrast operation is shown in Figure 2-11.

2.2.4 Histogram Equalization

Brightness adjustment, contrast adjustment and contrast enhancement utilize the image histogram in very crude fashions but are often adequate for an application like the Digital Darkroom. However, there is a more powerful technique known as Histogram Equalization that uses a non-linear method to enhance the image.

The discrete brightness values of an image (e.g. 0 - 255 ) may be normalized to convert to continuous values from 0.0 to 1.0 by dividing the discrete values by the highest brightness level (e.g 255). The histogram of an image may be converted to an equivalent probability density function of the same shape. The probability of a brightness level, \( p(r) \), is computed by dividing the number of pixels that has this brightness level by the total number of pixels in the image (see Eq. 2.2-6). By plotting a graph of \( p(r) \) against the brightness level \( r \), where \( r \) is continuous and \( 0 \leq r \leq 1 \), and approximating this graph by a continuous curve, the probability density function like that of Figure 2-12 may be obtained.
An image with a probability density function shown in Figure 2-12 is a predominantly light image since most of its brightness levels are at the bright region of the brightness scale. Figure 2-13 is a probability density function that belongs to a dark image as the majority of its brightness levels cumulate in the dark regions of the brightness level. These pictures of low contrast can be enhanced by **Histogram Equalization**. This technique attempts to spread the histogram evenly along the brightness level of the image, modifying its probability density function to 1 (in the brightness range of 0 to 1.0 of the image). See Figure 2-14.

Assume that for every brightness pixel value \( r \) in the original image, transformation \( T(r) \) transforms the pixel value to the final image pixel value of \( s \).

\[
s = T(r) \quad \text{where } r \text{ is continuous and } 0 \leq r \leq 1.
\]

Suppose \( s \) has the following transformation function,

\[
s = T(r) = \int_{0}^{r} p_r(w) dw \quad 0 \leq r \leq 1
\]  \hspace{1cm} (2.2-1)

where \( w \) is a dummy variable of integration. The integral is the cumulative distribution function (CDF) of \( r \). We can show that this definition for \( T(r) \) gives a uniform density function for the transformed image.

Differentiating Eq. (2.2-1) with respect to \( r \) yields

\[
\frac{ds}{dr} = p_r(r)
\]  \hspace{1cm} (2.2-2)

Using probability theory that if \( p_r(r) \) and \( T(r) \) are known, the resulting probability density
function of the transformed brightness levels is

\[ p_s(s) = p_r(r) \frac{dr}{ds} \bigg|_{r=T^{-1}(s)} \]  \hspace{1cm} (2.2-3)

where \( T^{-1}(s) \) is the inverse transformation from \( s \) back to \( r \).

Substituting Eq. (2.2-2) into Eq. (2.2-3) gives

\[ p_s(s) = p_r(r) \frac{1}{p_r(r)} \bigg|_{r=T^{-1}(s)} \]

\[ = [1]_{r=T^{-1}(s)} \]

\[ = 1 \quad 0 \leq s \leq 1 \]

which is a uniform probability density in the range \( 0 \leq s \leq 1 \), similar to that of Figure 2.14, the desired probability density function.

An actual image has discrete brightness values, and so to be able to utilize the above techniques, a discrete form of Eq. (2.2-1) must be formulated.

From Eq. (2.2-1)

\[ s = T(r_k) = \sum_{j=0}^{k} p_r(r_j) \quad k=0,1,...,M-1 \]  \hspace{1cm} (2.2-5)

where \( M \) is the number of brightness levels.

The probability of a pixel brightness value is computed by
\[ p_r(r_k) = \frac{n_k}{n} \quad 0 \leq r_k \leq 1 \quad k=0,1,...,M-1 \tag{2.2-6} \]

where \( p_r(r_k) \) is the probability of the \( k \)th level, \( n_k \) is the number of pixels that has a brightness value of \( k \), and \( n \) is the total number of pixels in the image.

Substituting Eq. (2.2-6) into Eq. (2.2-5)

\[ s_k = \sum_{j=0}^{k} p_r(r_j) \]

\[ = \sum_{j=0}^{k} \frac{n_j}{n} \quad 0 \leq r_k \leq 1 \quad k=0,1,...,M-1 \tag{2.2-7} \]

Take for example, a 64x64 monochrome image with 8 gray levels which are normalized to be between 0.0 and 1.0. The following table shows the gray level probabilities of the image. See Figure 2-15.

<table>
<thead>
<tr>
<th>( r_k )</th>
<th>( n_k )</th>
<th>( p_r(r_k)=n_k/n ) (( n = 64 \times 64 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>500</td>
<td>0.12</td>
</tr>
<tr>
<td>1/7</td>
<td>1100</td>
<td>0.27</td>
</tr>
<tr>
<td>2/7</td>
<td>880</td>
<td>0.22</td>
</tr>
<tr>
<td>3/7</td>
<td>650</td>
<td>0.16</td>
</tr>
<tr>
<td>4/7</td>
<td>400</td>
<td>0.10</td>
</tr>
<tr>
<td>5/7</td>
<td>310</td>
<td>0.07</td>
</tr>
<tr>
<td>6/7</td>
<td>166</td>
<td>0.04</td>
</tr>
<tr>
<td>1</td>
<td>90</td>
<td>0.02</td>
</tr>
</tbody>
</table>

By using Eq. (2.2-7), the following \( s \) values are obtained:
\[ s_0 = 0.12 \quad s_1 = 0.39 \]
\[ s_2 = 0.61 \quad s_3 = 0.77 \]
\[ s_4 = 0.87 \quad s_5 = 0.94 \]
\[ s_6 = 0.98 \quad s_7 = 1.00 \]

Since the gray levels are eight discrete and distinct values of \( 1/7, 2/7, \ldots, 1 \), the computed \( s \) values are approximated to the nearest discrete gray level, giving

\[ s_0 \approx 1/7 \quad s_1 \approx 3/7 \]
\[ s_2 \approx 4/7 \quad s_3 \approx 5/7 \]
\[ s_4 \approx 6/7 \quad s_5 \approx 1 \]
\[ s_6 \approx 1 \quad s_7 \approx 1 \]

The above assignments yield only six distinct histogram equalized values:

\( s_0, s_1, s_2, s_3, s_4, \) and \( s_5 \). See Figure 2-16.

29
In practice, the equalized histogram is very seldom perfectly flat. However, the approximation is good enough to yield at least satisfactory results. Such an effect increases the contrast of an image that previously had a poor contrast.

The Histogram Equalization function is not available in the current version of the
2.2.5 Creating Monochrome Negatives

Negatives can be produced from monochrome prints by producing the complement of the picture. An image complement is created by inverting the brightness levels of the
pixels, i.e. white pixels become black and vice versa. For any intermediate gray scale brightness level that in between black and white, the complement is produced by subtracting its brightness value from the maximum brightness value (full white). See Figure 2-15.

2.2.6 Solarization

Solarization or Sabattier Effect is also a point process. Solarization is a darkroom technique that causes an image to possess both the negative as well as the positive components giving the image a unique 'negative yet positive' appearance. By complementing areas of an image that exceeds a certain brightness value, the image will get a Solarized look (Figure 2-16). Instead of complementing the pixels that exceed a particular brightness level, areas of the picture may be selected to be inverted regardless of the brightness of the selected area. The area may be selected by using a mouse or other input device. This latter method is area-selective rather than the former method.
which is *brightness-selective*.

![Figure 2-17. Graph for Complementing Image](image1)

![Figure 2-18. Graph for Brightness-Selective Solarization](image2)

2.3. Area Processing

An image enhancement process can also treat the pixels of the image as small groups of pixels rather than individual pixels. The information of pixels adjacent to a particular
pixel is used to modify the value of that pixel. The general behavior or property of an area of pixels may be deduced by examining and using the values from adjacent pixels in the area. Such processing is known as *area processing* or neighborhood processing and is usually used for sharpening, blurring (to create soft focus effect or fog effect) and edge detection/enhancement of images.

An image may be examined in terms of its *spatial frequency* with sudden brightness transitions being the high frequency components and slow transitions the low. In other words, highest frequency occurs when a pixel value changes from 0 to 255 (darkest to brightest) within the distance of one pixel. *Spatial filtering* refers filtering an image either of its high or low frequency components, yielding accentuation on the edges of an image in the latter case. Low pass filtering and high pass filtering are the two main categories of spatial filtering. *Spatial convolution* is the technique used in spatial filtering to calculate the new value of a pixel based on the brightness values of its neighboring pixels.

In spatial convolution, the new brightness value of each pixel in the image is determined by computing a *weighted average* of itself and its immediate neighbors. The pixel whose value is modified is the center pixel of a group of \((n \times n)\) pixels called the *kernel*. The kernel is a square matrix of pre-determined size, usually 3x3 for most purposes. The bigger the size of the kernel, the longer the computation time of the weighted average since more neighboring pixels are examined for each pixel processed. Each element of the kernel has an associated scalar factor called *convolution coefficient*. 
These coefficients determine which pixels have more or less weight on the overall value of the average computed. Each pixel of the kernel is multiplied by its corresponding convolution coefficient in the kernel. The total sum of these values is the weighted average. For example, take the case of a kernel $k$ of size 3x3. The weighted average $p$ for the center pixel at location $l, l$ will be:

$$w(l, l) = p(0, 0)k(0, 0) + p(1, 0)k(1, 0) + p(2, 0)k(2, 0) +$$

$$p(0, 1)k(0, 1) + p(1, 1)k(1, 1) + p(2, 1)k(2, 1) +$$

$$p(0, 2)k(0, 2) + p(1, 2)k(1, 2) + p(2, 2)k(2, 2)$$

So, for a kernel of size $m \times n$, and $p(x, y)$ contains the pixel values, the weighted average $w$ can be mathematically expressed as:

$$w(x, y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} p(x+j-n/2, y+i-m/2)k(j, i)$$

The kernel is sequentially applied to each pixel of the image, except those pixels near the edges (Figure 2-17). For a kernel size of $m \times m$, the kernel cannot be applied to the $(m \div 2)$ pixels on the edges of the image. In such cases, the pixel values are set to 0 or some value that is near it.

2.3.1 Low Pass Filtering

*Low pass filtering* removes the high frequency components of an image, allowing only the low frequency components through, yielding the term 'low pass'. Removing the high frequency components of the image which are the pixels with the fastest brightness.
transitions (within a distance of one pixel) yields a resultant image which is less sharp.

The convolution coefficients for a \((m \times m)\) kernel used for low pass filtering is often of form:

\[
\begin{array}{cccc}
\frac{1}{m \times m} & \frac{1}{m \times m} & \cdots \\
\frac{1}{m \times m} & \frac{1}{m \times m} & \cdots \\
\vdots & \vdots & \ddots
\end{array}
\]

The convolution coefficients are positive fractions that sum up to 1. So, for a 3x3 kernel, the low pass kernel has the following convolution coefficients:

\[
\begin{array}{ccc}
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9}
\end{array}
\]

As the low pass kernel is applied to the image, the pixels are treated as pixels with a 3x3 'influential neighborhood'. The collective brightness value of this 3x3 neighborhood is the important factor in the determination and calculation of the brightness value of the center pixel of the low pass kernel. The spatial frequency of a group of 3x3 pixels all having the same brightness value is 0. Since there is no change in the brightness levels amongst the pixels in the 3x3 neighborhood, the spatial frequency is the minimum. In other words, this neighborhood has the lowest spatial frequency. When the low pass
kernel is applied on these pixels having the same brightness values, the brightness value of the center pixel is not modified since its new value (the weighted average) is the same as its original value. The weighted average is the same as the original brightness value of the center pixel because the convolution coefficients of the kernel add up to 1 and the brightness values of the pixels in the kernel are equal. This behavior indicates that the low frequency components of the image are passed through practically untouched. Suppose the low pass kernel is applied to a high frequency area of the image where the brightness value of every other pixel in the neighborhood changes very abruptly (i.e. from white to black or vice versa), for example, a single-pixel black line drawn on a white background. The center pixel's brightness value will be modified from say, 255, to the weighted average of the kernel which will be less than 255 due to the low brightness levels of the neighboring pixels. By reducing the value of this center pixel, the frequency of the neighborhood is reduced since the brightness transition will now be slower due to the lower value of the center pixel. The image is removed or subdued of its high frequency components as the kernel is sequentially applied to the entire image. The resultant image is one that is blurrer (softer) than the original. Although blurring a picture provides soft focus effects for photography buffs, a more practical use is the blurring of an unsightly power cable for esthetic purposes.

2.3.2 High Pass Filtering

High pass filters accentuates the high spatial frequency components of an image, producing emphasis on the edges of objects in the image. A typical high pass kernel has
the following convolution coefficients:

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
-1 & 9 & -1 \\
-1 & -1 & -1 \\
\end{array}
\]

Like the low pass filter, the convolution coefficients of the high pass kernel has a total sum of 1. The convolution coefficients surrounding the center coefficient are significantly smaller than the center coefficient. This characteristic indicates that the center pixel carries much more weight than its surrounding pixels. If the center pixel has higher brightness value than its neighbors, the large weight it carries will render the negative effects of its neighbors negligible. This center pixel assumes the value of the weighted average of the kernel which is a much larger brightness value than its original value, causing the center pixel to be brighter. The effect of applying the high pass kernel on a region where the pixels’ brightness values in the neighborhood are equal is the same as that of a low pass filter since the convolution coefficients of the high pass filter also add up to 1. This indicates that all the low frequency components of the image are passed through the high pass filter untouched. The high pass filter only accentuates the high frequency regions of the image. It does not remove low frequency components from the image. By increasing the magnitude of the higher frequency components, the low frequency components are rendered insignificant.

To illustrate, consider an image that has uniform brightness of 50 on one side and brightness level of 100 on the other (see Figure 2-20). High pass filtering will cause the brightness values of the pixels on one side of the edge to be reduced while the brightness of the pixels on the other side of the edge is increased, thereby increasing the spatial
frequency of the area near the edge.

![Image of before and after high pass filtering]

Figure 2-20. Effects of High Pass Filtering

2.3.3 Frequency Domain Convolution

Another method of filtering uses the frequency domain of the image instead of the spatial domain of the image described earlier. Frequency domain based convolution is generally more precise and for large arrays, the convolution can be carried out faster. Techniques such as the Fast Fourier Transform are often used in frequency domain based image enhancement. Such techniques are not currently used in the Digital Darkroom.
2.4 Frame Processing

Frame processing encompasses all other image processing operations that do not fit into either the point or area processing categories. Such techniques include geometric operations, data compression, and multiple-image processing.

2.4.1 Geometric Operations

Geometric operations modify the spatial relationship of pixels in the image. Basic geometric operations include scaling (enlarging/shrinking), rotation, stretching, translation and shearing (geometric distortion). These operations can be characterized by the following expression:

\[ I(x,y) \rightarrow O(x',y') \]

where \((x',y')\) are the new coordinates of the pixel whose original coordinates were \((x,y)\)

2.4.2 Data Compression

Data compression is an issue in image processing since image files usually take a huge amount of disk space. Utilizing an effective compression algorithm reduces the storage requirements of the images. Images usually contain areas which are of the same color, for example, pictures containing portions of the sky. Although the sky may have many shades of blue, each shade of blue does not normally occupy only one pixel. Each
shade usually applies to a sequential collection of pixels. There are many data compression algorithms available to compress these repetitive patterns. One such simple, yet effective method is *Run Length Encoding*.

Run-length encoding is effective when the repetition is contiguous. Suppose a portion of the image contains a string of 50 F's. Instead of storing 50 F's, taking up 50 bytes of disk space, it is encoded with the 'F' and then followed by the frequency of the F's, using only 2 bytes of disk space. The Achilles Heel of run-length encoding is when there is not a sequence of continuous repetition, that is, when each byte of data is different from the previous byte. This infrequent occurrence requires twice the disk storage than that needed for the original image due to the addition of the extra byte needed to store the frequency, which in this case is always 1. However, for most purposes and images, this method is effective for its simplicity. Simplicity saves encoding and decoding time. The popular ZSoft PCX image file format uses run-length encoding for its data compression.

The PCX format also stores the colormap of the image in its header (See Section 5.5.1). The colormap (or color lookup table) stores the different colors of the image. For example, for an image that has 256 unique colors, the colormap will have 256 entries. Each entry contains the RGB values of that color. Using this color lookup table, the raw image data is stored using the colormap entry number of the color rather than storing the actual RGB values of that pixel at that location. Since the size of the colormap entry number is at least 3 bytes smaller the data's RGB values, there is substantial savings in disk space.
2.5 Color Image Processing

The image processing operations discussed above deal mainly with the brightness value of the pixels. For monochrome images, brightness is the only other attribute of the pixels besides the pixels' location. However, for color images, the pixels have color components, namely the red(R), green(G) and blue(B) values that constitute the images' color information. To apply the above image processing operations on color images there is a need to derive the brightness value from the color components. As discussed earlier, color may be characterized using the RGB and HSV color models. Since the HSV color model is derived from the RGB color model (see Section 1.3.3), the color information can be parametrized into H, S, and V values as well. By using the relationship between the RGB color model and the HSV color model, the brightness or intensity value (V) is derived from the RGB values of the pixels. Image processing operations are performed on the color image using the brightness values of the pixels gotten from the RGB-to-HSV conversion. After the image processing operation is done, the HSV values of the pixels are converted back to RGB values for display. Another technique that is usually necessary and employed in color processing is color quantization.

2.5.1 RGB to HSV Conversion

The algorithm for the conversion from RGB values to HSV values is outlined below:

**Input**: R, G, B values - The red, green, blue triple of the color space (or pixel).

\[ 0 \leq R \leq 1. \]
\[ 0 \leq G \leq 1. \]
\[ 0 \leq B \leq 1. \]
Output: H, S, V values - Hue, Saturation, Value.
0 ≤ H ≤ 360. H = -1 if H is undefined.
0 ≤ S ≤ 1.
0 ≤ V ≤ 1.

Functions Used:
Maximum( a, b, c ) - Function that returns the largest of the three values passed in.
Minimum( a, b, c ) - Function that returns the least of the three values passed in.

Algorithm:

RGB_TO_HSV (R, G, B, H, S, V)
begin
/* Get the dominant RGB component */
MaxValue ← Maximum( R, G, B )

/* Get the least dominant RGB component */
MinValue ← Minimum( R, G, B )
V ← MaxValue
if ( MaxValue ≠ 0 )
    S ← (MaxValue - MinValue ) / MaxValue ;
else
    S ← 0 ;
endif
if ( S = 0 ) /* No color, just monochrome (gray,black,white) shades */
    H ← -1 /* H is undefined. Set H to -1. */
else
    rc ← ( MaxValue - R ) / ( MaxValue - MinValue )
    gc ← ( MaxValue - G ) / ( MaxValue - MinValue )
    bc ← ( MaxValue - B ) / ( MaxValue - MinValue )
    if ( R = MaxValue )
        H ← ( bc - gc )
    else if ( G = MaxValue )
        H ← ( 2 + rc - bc )
    else if ( B = MaxValue )
        H ← ( 4 + gc - rc )
endif
    H ← H * 60
    if ( H < 0 )
        H ← H + 360
endif
end

43
2.5.2 HSV to RGB Conversion

The algorithm for HSV to RGB conversion is listed as follow:

**Input**: H, S, V values - Hue, Saturation and Hue.
- $0 \leq H \leq 360$. $H = -1$ if H is undefined.
- $0 \leq S \leq 1$.
- $0 \leq V \leq 1$.

**Output**: R, G, B values - Red, green, blue triple of the color space.
- $0 \leq R \leq 1$.
- $0 \leq G \leq 1$.
- $0 \leq B \leq 1$.

**Algorithm**:

```plaintext
HSV_TO_RGB (H, S, V, R, G, B)
begin
    if (V \geq 1) 
        V <- 1
    endif
    if (V < 0) 
        V <- 0
    endif
    if (S = 0)
        if (H = -1) /* if H is undefined, there is no color, */
            R <- V /* just gray shades, so set all the R,G,B */
            G <- V /* values to be the same as V, the intensity */
            B <- V
        endif
    else
        if (H = 360)
            H <- 0
        endif
        H <- H / 60 /* Convert H from degrees to its hexagon section. */
    endif
end
```

/* Get the dominant color */
ipart ← \lfloor H \rfloor  /* Put the integer part of H into ipart.  
* ipart ranges from 0 to 5.  
*/

/* Get the second most dominant color */
fpart ← (H - ipart) /* Put the fractional part of H into fpart. */

hp1 ← V * (1 - S)
hp2 ← V * (1 - (S * fpart))
hp3 ← V * (1 - (S * (1 - fpart)))
case ipart of
  0 : R ← V
     G ← hp3
     B ← hp1
  1 : R ← hp2
     G ← V
     B ← hp1
  2 : R ← hp1
     G ← V
     B ← hp3
  3 : R ← hp1
     G ← hp2
     B ← V
  4 : R ← hp3
     G ← hp1
     B ← V
  5 : R ← V
     G ← hp1
     B ← hp2
end
end
end /* of HSV_TO_RGB */

2.5.3 Color To Monochrome Conversion

To convert a color picture to monochrome (black and white), the RGB values of the
image are first converted to HSV values. The V(Value) component gotten from the RGB-
to-HSV conversion, which is the brightness value, is then used to set the Red, Green and
Blue components of the color image. By setting the Red, Green and Blue components
of the image to be the same value, the color image is converted to monochrome.

2.5.4 Color Quantization

Color quantization refers to the process of selecting the best colors of an image to
best display the image. Of course, the best way is to display all the colors of the image.
However, some display devices or software may not support the entire color gamut of the
input image. In other words, the image may contain 1024 unique colors, but the output
display device is only capable of displaying 256 unique colors simultaneously. By
applying color quantization on the image, the best 256 colors that best represent that
image would be determined and the image can thus be displayed on the 256-color output
display device. There are many methods of color quantization, like the popularity
algorithm, the median cut algorithm and the octree quantization algorithm. The popularity
algorithm pre-scans the image to obtain a histogram of the colors in the image. The
colors that have the highest frequencies are selected.
Chapter Three

Formal Specifications

3.1 Specifications of the Digital Darkroom

The Digital Darkroom operates in the X11 Windowing System using the Motif Graphical User Interface. The Digital Darkroom shall as closely simulate an actual photographic darkroom as possible and also have the capabilities to perform a variety of image processing that are not possible in a photographic darkroom. The user interface must be intuitive. Inputs to the Digital Darkroom are digitized images with video resolution not exceeding 1280x1024 pixels and the number of unique colors in the input image may be unlimited. Image file formats used must at least support the ZSoft PCX image file format. Capability must be provided to save a modified image onto disk. The number of colors of the saved image need not have to be equal to that of the original image. It may be limited by the color capability of the Digital Darkroom. The number of colors displayed is only limited by the capability of the video hardware on the computer.

The user interface for editing/selection of colors must utilize a user-friendly and interactive color model for easy perception, modification or selection of colors. No technical knowledge of color models or color specification is necessary for user to pick or modify a color. Selection or modification of colors must be as intuitive as paint
selection and mixing on an artist’s painting palette.

Zoom capability on the image displayed must be provided for ease of pixel or area color editing. The user must be able to edit a pixel or an area while the zoomed image is displayed.

The following image processing capabilities that are present in an actual photographic darkroom must be provided:

1. Adjust the brightness of a picture.

2. Adjust the contrast of a picture.

3. Focus or blur a picture to produce a soft focus effect.

4. Rotate a picture in 90-degree increments.

5. To add color filters to create an overall tint of a specific color (e.g. create sunset effect with an orange filter).

6. To convert a color picture to black and white while closely maintaining the contrast and brightness of the original picture.

7. To create the Sabattier Effect.

To extend the capabilities of the Digital Darkroom, it must also have the following features not found in most photographic darkrooms:
1. Edge enhancement/detection.

2. Auto color saturation. Automatically chooses more saturated colors for images. The user selects the amount of saturation.

3. Color temperature correction. Corrects for color temperature imbalance caused by incorrect lighting when the photograph was taken.

4. Create a negative image from a positive image and vice versa.

5. Display at least the following statistics of the picture:
   i. Frequency of colors used.
   ii. Video resolution of picture.

The user interface for the *Digital Darkroom* must conform to the OSF/Motif Style Guide. Although the mouse shall be the primary input device, menus and dialogs must also be accessible through the keyboard.
4.1 The Graphical User Interface Design

The user interface design of a system should be separate from the functional design of the system. However, this does not imply that the system be designed with its user interface as an afterthought. Most often, (though the trend is changing) the user interface is simply thought of as a means to get input from user and output results to the user. Not much attention is paid to the effectiveness, user-friendliness and natural flow of the user interface, e.g. the DOS command line.

With the advent and proliferation of Graphical User Interfaces (GUIs) on the desktops and workstations, the design of a system that appeals to the user does depend on the user interface the system possesses. Unlike traditional data processing that use character-oriented interfaces, GUIs' windows, menus, dialogs and events demand more attention during the design phase. Such attention is especially necessary in developing interactive systems like painting programs, drawing programs and presentation programs. The user interface portion of the system is quite often the most important criterium by which most systems are judged in these days of GUIs. The overall design of interactive applications needs to be driven by the user interface. Developing the user interface must not be an afterthought. Although the user interface used in the system should drive the design of
the system, care should be taken to separate the user interface portion of the system from the functional portion of the system. This approach helps design a system that is more user-friendly since the user interface portion of the system is taken into consideration when the design of the system is developed.

The Digital Darkroom is a system that has to be interactive and user-friendly to be useful. Design of its user interface is important in order to meet these goals. Most GUIs have toolkits that are conceptually object-oriented and categorized into various classes. Since the specifications of the Digital Darkroom requires that Motif be the GUI of choice, the various classes of objects available in the Motif Toolkit are examined. With the adequate knowledge of the Motif classes, screen layouts are drawn up and used as a template for which to develop the Digital Darkroom. Screen layouts are an effective tool for prototyping an interactive application. It provides an effective communication medium by which the analyst/designer can present his/her understanding of the system’s specifications to the customers. By using this same communication tool, the customers can give rapid and interactive feedback even before the details of the actual implementation are completed or even began. Awkward dialogs popping up at inappropriate times, and excessive keystrokes can be identified by the customers and corrected by the designer well before the guts of the system is implemented, eliminating unnecessary re-coding and design. Continually showing the customers screen layouts of the system assures them that the system is being designed according to their specifications.
4.2 Motif Toolkit and Class Library

The OSF Motif Toolkit is comprised of an extensive set of widgets that are used to construct a Motif application. It is built upon the Xt Intrinsics Toolkit which is in turn built upon the X11 Windowing System. A Motif application, besides accessing Motif functions and widgets, may also access Xt Intrinsics and low-level X11 (Xlib) functions. Most graphics primitives are only available through accessing low-level X11 functions, for example, line draws, colormaps, and polygon fills. See Figure 4-1.

![Motif Application and Toolkits Relationship](image)

**Figure 4-1. Motif Application and Toolkits Relationship**

The Motif Widget set consists of various object-oriented classes of widgets. Each Motif widget belongs to a Motif widget class and may inherit resources (attributes) of parent classes. The basic Motif class is the Core class which is inherited by the rest of the Motif widget classes. The following figures illustrate the hierarchy of the various Motif widget classes:
Figure 4-2. Basic Widget Class Hierarchy
Figure 4-3. Primitive Class Widgets

Figure 4-4. Shell Widgets
Figure 4-5. Manager Widgets

Figure 4-6. Dialog Widgets
4.3 Digital Darkroom User Interface

The user interface design is outlined using sample screen layouts and detailed descriptions and conforms to the OSF/Motif Style Guide. The main screen layout of the Digital Darkroom is shown in Figure 4-7. The complete screen design is illustrated in the form of a menu tree in Figure 4-8.

Figure 4-7. Overall Screen Design of the Digital Darkroom
Figure 4-8. Complete Menu Tree of the Digital Darkroom
MENU BAR OPTIONS:

File: All functions pertaining to file/disk operations.

Open - Opens a digitized image stored on disk for display and/or manipulation. The image format supported will be of PCX format.

Save - Saves modified image onto disk.

Quit - Exits Digital Darkroom.

Edit:

Cut - Removes an area of the image that may be retrieved by the Paste procedure.

Paste - Retrieves the area of the image removed by the latest Cut procedure.

View:

Zoom - Selects an area to zoom in. Cursor changes to crosshair type and user will click on left mouse button to select area to be zoomed in. A Zoom Dialog will display the zoomed area (See Figure. 4-9).

Rotate - Rotates image clockwise by increments of 90 degrees.

Quantize:

The method of reducing the number of colors needed by the raw image data should the colormap (frame buffer) not able to accommodate all the colors of the raw input image.

Error Distance -
By masking out an appropriate number of least significant bits of each unique color needed, the color requirements of the image can be reduced.

Popularity Algorithm -
Quantization using the n most popular colors to fit into an n-size frame buffer.

Special Effects:
Contrast Enhancement -
Applies contrast enhancement to the image using the algorithm outlined in Section 2.2.3.

Solarization -
Also known as Sabattier Effect. An effect whereby the negative as well as the positive components of the image are present.

Fog -
Creates a fog-like appearance over the entire image.

To Negative -
Converts a 'positive' image to its 'negative' component.

To Positive -
Converts a 'negative' image to 'positive'.

To Monochrome -
Converts a color image to monochrome.

Edge Detection -
Enhances the edges of subjects in the image.

Color Enhance :

Image enhancements. All processes under this group invoke the Color Edit Dialog.

Pixel Color -
Edits the color of a pixel.

Area Color -
Edits the color of a group of pixels. The group of pixels will assume the same color after editing, regardless of their individual original colors.

Color Correction -
Corrects color imbalance caused by wrong color temperature. User will choose tint to correct color imbalance.

Tint -
Allows the user to apply an overall tint to the picture. Tint is selected by using the Color Edit Dialog (Figure 4-10).
Auto Color Saturation -
Automatically selects more saturated colors for the image. User will be presented with the Color Edit Dialog (Figure. 4-10) to select the amount of saturation.

Info : Information on image displayed.

Image Info -
Displays the following information about the image :
1. Video resolution of image.
2. Number of bytes in the image.
3. Type of image file format (e.g. PCX, etc ).
4. Number of colors contained in the image.

Histogram -
Displays the histogram of the image with the number of pixels on the vertical axis and the brightness values on the horizontal axis.

Color Frequency -
Pixel frequency of each individual color in the colormap (after any quantization done).

SLIDER CONTROLS:

Brightness :
Modifies the brightness of the image. Values range from -255 to 255, with 0 being the original brightness of the image.

Contrast :
Controls the contrast of the image. Values range from 0.1 to 5.0.

Focus :
Controls the sharpness of image. No numeric values.
DIALOGS:

Zoom Dialog:

The Zoom Dialog is invoked when the Zoom option of the Edit pulldown menu is selected. The dimensions of the zoomed display area are 640 pixels by 480 pixels. To edit the color of a pixel, the user pushes the 'Edit Pixel' button and then proceeds to the zoomed-in display to click on the pixel to be color edited. A Color Edit Dialog shall be displayed for editing the color of the pixel (See Figure. 4-10). To edit an area, the user pushes the 'Edit Area' button and proceeds to 'lasso' the area to be edited. 'Lasso' is activated by the left button and completed when the right button is pressed. If the ending point of lasso does not meet the starting point of lasso when the right button is pressed, the system will automatically join the two points by a straight line. The Color Edit Dialog shall then be invoked and the procedure from this point on will be the same as that for editing a pixel.

Figure 4-9. Zoom Dialog

Color Edit Dialog:

The Color Edit Dialog enables the user to modify the color of a pixel or group.
of pixels. It also has a display area that shows the 256 colors of the system arranged in a 16x16 grid. The user may select a color from the system color display area or modify the selected color by varying the Hue(H), Saturation(S) and Value(V) slider controls or the Red(R), Green(G) and Blue(B) controls. If either the H, S or V controls were changed, the R, G, or B controls are automatically updated with the correct values and vice versa. The user may exit this dialog without making permanent color change. See Figure 4-10.

Figure 4-10. Color Edit Dialog
4.3.1 Widget Hierarchy of the Digital Darkroom

The widget hierarchy of the Digital Darkroom is listed in the following pages. The labels in bold reflect the Motif widget type and those in italics are the variable names used for those widgets.

![Diagram of widget hierarchy]

Figure 4-11. Widget Hierarchy of the Digital Darkroom
Figure 4-12. Widget Hierarchy of Pulldown Menus

Figure 4-13. Widget Hierarchy of the Color Edit Dialog
4.4 Functional Design of The Digital Darkroom

The functional design of the Digital Darkroom is presented in the form of Data Flow Diagrams in the following pages (Figure 4-15 to Figure 4-20).
Figure 4-15. Level 0 Data Flow Diagram of the Digital Darkroom
Figure 4-16. Level 1 DFD of Process Image
Figure 4-17. Level 1 DFD of Add Special Effects

T2* = A dummy transformation that renames each individual data to T2 as Image With Special Effects to facilitate the construction and comprehension of the Data Flow Diagram.
Figure 4-18. Level 1 DFD of Modify Geometry

T3* = A dummy transformation that renames each individual input data to T3 as Geometrically Modified Image to facilitate the construction and comprehension of the Data Flow Diagram.
Figure 4-19. Level 1 DFD of Color Enhance

T4* = A dummy transformation that renames each individual input data to T4 as Color Enhanced Image to facilitate the construction and comprehension of the Data Flow Diagram.
As described earlier, the functional design of the system should be separate from the user interface design. By using Data Flow Diagrams to show the flow of the data and their transformations, the functional design of the system is presented in a manner that is not intermingled with the User Interface Design aspects. The functional aspects (or requirements) of the system list the different types of image processing functions available in the Digital Darkroom. The functional design describes the design of the different image processing functions and the modifications of the image as they go through the
different image processing transformations. The functional design does not dictate or drive the design of the user interface of the *Digital Darkroom*.
Chapter 5

Implementation

5.1 Implementation of the Digital Darkroom

The Digital Darkroom was implemented using the design described in Chapter 4. The user interface was implemented using the OSF/Motif User Interface Language (UIL) for ease of prototyping and separation of the user interface from the functional requirements of the system. The use of Motif and X11 required extra consideration and examination of Motif's use of colors and X11 colormaps during implementation. The input image file format of choice was the ZSoft PCX format.

5.2 User Interface Language (UIL)

The OSF/Motif User Interface Language (UIL) is a proprietary language for specifying and developing Motif user interfaces. UIL helps separate the user interface code from the actual C code that implements the other functions and algorithms of the system. This separation allows the programmer to modify and customize the user interface without fear of corrupting or introducing bugs into the actual C code of the application.

UIL does not replace the Motif Toolkit. It is just another method of specifying widgets other than using C code. Both methods access the Motif Toolkit to actually create
and manage the widgets. The UIL compiler compiles the UIL code into UID files which are read by the C application at run-time. This behavior allows the use of different UID files for the same application code. Different and separate UID files for the same application are created when the need for the application to be presented in say, another language, and so on. UIL also eases rapid prototyping and allows better maintenance. By keeping the UIL code separate from the C code, the person responsible for modifying the user interface need not possess programming skills. Only knowledge of the UIL is needed to modify the user interface, freeing the programmer to do actual programming tasks. Rapid prototyping aids the specification and design of the system and better maintenance ensures a more reliable system since the chances of introducing new bugs into the C code is lessened. Generally, a bug in user interface is more likely to be detected sooner since the user interface is visual whereas a logic bug in the C code may not be detected for some time. By this reasoning, a bug in user interface is generally less devastating than one that is introduced into the application code. The use of UIL is therefore very much in keeping with the concept of separating functional aspects of the system from user interface details.

5.3 Use of X11 Colormaps

Any number of colormaps may be stored on an X11 system. However, only one may be active (or loaded into the frame buffer). The X11 system is able to switch among the different colormaps available in the systems by loading and unloading colormaps at appropriate times. The colormap of an application that has the 'focus' is normally the
one that is loaded into the frame buffer. An application obtains the 'focus' when the user clicks on the window of the application or through keystrokes that perform the equivalent. When a specific colormap of an application is active, the colors of the other applications that are running will be different since the active colormap may not be the same as the colormaps of these other applications. Suppose there are 2 Motif applications A and B running on the system, each with its own specific colormap. A has the following colormap defined:

<table>
<thead>
<tr>
<th>Index</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>4</td>
<td>255</td>
<td>255</td>
<td>0</td>
</tr>
</tbody>
</table>

and B has the following colors defined in its colormap:

<table>
<thead>
<tr>
<th>Index</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>3</td>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>

A is to draw a filled red circle and B is to draw a blue square. Assume both are running on the system. With A's colormap active, A's circle was drawn with a pixel value of 1 which is the A's colormap index for red. With B's colormap active, its blue square is drawn with a pixel value of 2 which is B's colormap index for blue. When focus is set
to A, its circle would be displayed as red, and B’s square would be green instead of blue since it is A’s colormap that is loaded into the frame buffer. B’s square is displayed as green since the color defined for pixel value of 2 is green using A’s colormap. If focus were to be set to B, then B’s square would be correctly displayed as blue and A’s red circle would be green.

To use its own colormap instead of the default system colormap, an application has to explicitly tell X to use that application’s colormap each time it is given the focus.

5.4 Motif Colors

To achieve the three-dimensional appearance, Motif uses several color shades to draw its widgets. Motif saves these colors in the first 20 entries of the colormap. If an application were to replace every entry of the colormap with its own color, the three-dimensional look of the Motif windows would be destroyed. To preserve the three-dimensional look, the first 20 entries of the colormap are left intact and not replaced by the image’s colors, leaving only 236 entries for the display of the image. Therefore, the image is quantized to 236 colors instead of 256.

5.5 PCX Image File

The image file format supported by the Digital Darkroom is the ZSoft PCX format. It was first developed by the ZSoft Corporation in 1982, and has since become a popular image file format. The image data is compressed using Run Length Encoding (See
Section 2.4.2). It is capable of storing monochrome as well as color images. The PCX format consists of three main parts: the Header (128 bytes), the Encoded Image Data (Variable Length) and the optional Colormap (769 bytes).

5.5.1 PCX File Header

The header contains information regarding the PCX image type and version, and essential video resolution and color information as described below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (Bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCX Flag</td>
<td>1</td>
<td>PCX Image Type (0x0A)</td>
</tr>
<tr>
<td>Version</td>
<td>1</td>
<td>PCX version number</td>
</tr>
<tr>
<td>Encoding</td>
<td>1</td>
<td>Run Length Encoding Used (0x01 for RLE)</td>
</tr>
<tr>
<td>Bits Per Pixel</td>
<td>1</td>
<td>Bits Per Pixel Per Plane</td>
</tr>
<tr>
<td>XMin</td>
<td>2</td>
<td>Top Left X Coordinate</td>
</tr>
<tr>
<td>YMin</td>
<td>2</td>
<td>Top Left Y Coordinate</td>
</tr>
<tr>
<td>XMax</td>
<td>2</td>
<td>Bottom Right X Coordinate</td>
</tr>
<tr>
<td>YMax</td>
<td>2</td>
<td>Bottom Right Y Coordinate</td>
</tr>
<tr>
<td>HDPI</td>
<td>2</td>
<td>Horizontal Resolution</td>
</tr>
<tr>
<td>VDPI</td>
<td>2</td>
<td>Vertical Resolution</td>
</tr>
<tr>
<td>ColorMap16</td>
<td>48</td>
<td>Colormap info for 16 or lesser colors</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>Set to 0x0</td>
</tr>
<tr>
<td>NPlanes</td>
<td>1</td>
<td>Number of Color Planes</td>
</tr>
<tr>
<td>Bytes Per Line</td>
<td>2</td>
<td>Bytes Per Line</td>
</tr>
<tr>
<td>Color Info</td>
<td>2</td>
<td>Color Or Monochrome</td>
</tr>
<tr>
<td>HScreen Size</td>
<td>2</td>
<td>Horizontal Screen Size</td>
</tr>
<tr>
<td>VScreen Size</td>
<td>2</td>
<td>Vertical Screen Size</td>
</tr>
<tr>
<td>Filler</td>
<td>54</td>
<td>Filler.</td>
</tr>
</tbody>
</table>

5.5.2 PCX 256-Color Colormap

If the image contains more than 16 colors, the colormap is appended to the end of
the encoded image data. A constant byte of value 12 (decimal) immediately after the image data indicates the presence of a 256-color colormap. This is followed by the colormap information sequentially arranged in 3-byte RGB triples. First 3 bytes represent the Red, Green and Blue values of the first colormap entry, the second 3 bytes represent the Red, Green and Blue values of the second colormap entry, etc, occupying a total of 768 bytes (256 x 3 bytes). This colormap is used as a lookup table for the image data which are indices to this colormap. So instead of using 3 bytes of RGB information to store each pixel's RGB information, using the indices to store the pixel information reduces the image data space needed by 2 bytes per pixel.
Chapter 6

Conclusion

6.1 Conclusion

The *Digital Darkroom* extends the creativity of a photographer or artist. By being able to view the changes made to the image instantaneously, the photographer’s creative thoughts are not interrupted by having to wait for the feedback. The *Digital Darkroom* is designed with the best effort to be user-friendly through its graphical user interface. Having a software that is very powerful but difficult to use is unlikely to attract many users. The ease of use of the *Digital Darkroom* should determine its efficiency and usability.

Special artistic effects added to an advertisement will most likely draw the readers’ attention more so than one that is plain. Its ability to selectively blur out subjects can shift the focus of a picture from one subject to another. Being able to produce the positive color image from a negative film in a few seconds is a boon to photographers. Minute and obscure details in a picture can be brought out by increasing the brightness of the picture.

The potential of the *Digital Darkroom* is only limited by the creativity of the user. The ability of the software to provide the enormous choice of colors, the modifications
of the colors, and the special effects available to the user provide combinations of changes and enhancements that are almost endless.
Bibliography


Barkakati, Nabajyoti. X Window System Programming. SAMS, 1991


Acknowledgments

I would like to acknowledge Dr. Youlou Zheng, Dr. Alden Wright and Dr. Randy Jeppesen for their time and effort in providing me with invaluable help and suggestions.

I am forever indebted to my family for their endless patience, support and understanding. Thanks also to my friends for their encouragement and support.