ABSTRACT

KRISH, KARTHIK Design and Development of a Cross Platform Interactive Image Processing System. (Under the direction of Dr. Wesley E. Snyder).

The objective of this thesis is to design and develop a cross-platform software system which will enable the user to perform image processing/analysis interactively. The software system will have the ability to render images of any data-type and visualize them in many different ways. The software will also integrate many commonly used image processing and analysis algorithms, that can be run on the images. The cross-platform nature of the tool will help in making sure that a uniform interface is presented to the user irrespective of the underlying architecture. The system is designed to be modular which makes it suitable for future expansion.
Design and Development of a Cross Platform Interactive Image Processing System

by

Karthik Krish

A thesis submitted to the Graduate Faculty of North Carolina State University in partial satisfaction of the requirements for the Degree of Master of Science

Electrical Engineering

Raleigh

2005

Approved By:

Dr. Edward Grant

Dr. Joel Trussell

Dr. Wesley E. Snyder
Chair of Advisory Committee
To Mom, Dad, Ashok and Raghav...
Biography

Karthik Krish was born in Bangalore, India on June 1st, 1982. He did his primary schooling at Vidya Mandir, Chennai and completed his high school at Mothers International School, New Delhi in 1999. He then went on to obtain a Bachelor of Engineering Degree in Electronics and Communication at the University Of Madras in 2003.

In August 2003, he enrolled in the Master of Science Program of the Electrical and Computer Engineering Department at North Carolina State University. After completing his graduation, he plans to continue on for a Phd. His research interests include computer vision, pattern recognition and image processing.
Acknowledgements

I would like to thank my advisor Dr. Wesley Snyder for his invaluable guidance and support throughout my research. He has always been a constant source of inspiration and a motivation to excel in all my endeavors. I shall always remember him for the rest of my life for his affable personality and technical acumen. I would also like to thanks Dr. Joel Trussell and Dr. Edward Grant for agreeing to be on my committee and for their time and patience. I would like to thank my friends in the Imaging group here at North Carolina State University for all their help and support.

I would like to thank my parents for their love and encouragement and having provided me with all the help they can to help me come up in life. I would also like to thank my brother Ashok for the constant encouragement and help during the course of my Masters. Finally, I would like to thank my friends Sudhakar, Arun, Chit, Nikhil, Karthik, Dinesh, Kumar and Kannan for all the crazy and fun times we had during the last two years in Raleigh.
Contents

List of Figures vii

List of Tables viii

1 Introduction 1
  1.1 Motivation .............................................. 1
  1.2 wxIFSView ............................................ 2
  1.3 Thesis Organization ................................. 3

2 Background Work 4
  2.1 Image File System(IFS) ................................. 4
     2.1.1 The Structure of an IFS image .................. 5
     2.1.2 Datatypes Supported .............................. 5
  2.2 WxWidgets ............................................ 6

3 Comparision with Existing Packages 7
  3.1 IMP .................................................. 7
     3.1.1 Features ......................................... 8
  3.2 Java IFSView ........................................... 8
     3.2.1 Features ......................................... 8
  3.3 ImageJ ................................................ 9
  3.4 Other Image Processing tools ......................... 11
     3.4.1 ImageMagick ..................................... 11
     3.4.2 Matlab ........................................... 11
     3.4.3 GNU Image Manipulation Program(GIMP) ........... 12

4 Architecture 13
  4.1 Imageframe Subsystem ................................. 13
     4.1.1 Initialization ................................... 13
  4.2 File Input/Output Subsystem ........................... 15
## 4.2.1 File Formats

### 4.3 Image Rendering Subsystem

- **4.3.1 Window and Level**
- **4.3.2 Intensity Scaling**
- **4.3.3 Colormaps**
- **4.3.4 Multiframe Images**
- **4.3.5 Image Resizing**

## 4.4 User Input Subsystem

## 4.5 Algorithms

## 4.6 Histograms

## 4.7 Image Statistics

## 5 Built-in Algorithms

- **5.1 Connected Component Labeling**
  - **5.1.1 Connectivity**
  - **5.1.2 Algorithm**
  - **5.1.3 Region of Interest**
- **5.2 Maximum Intensity Projection**
  - **5.2.1 Ray Tracing**
  - **5.2.2 Algorithm**
- **5.3 Histogram Equalization**
- **5.4 Profile plot**
- **5.5 Simple Thresholding**
- **5.6 FFT and Inverse FFT**

## 6 Summary and Future Work

- **6.1 Summary**
- **6.2 Future Work**

**Bibliography**

**A wxIFSView User Guide**
List of Figures

2.1 WxWidgets Rendering Architecture ........................................ 6

4.1 Overall Architecture ....................................................... 14
4.2 File Loading Process ........................................................ 18
4.3 Window/Level Example ....................................................... 19
4.4 The 7 Colormaps .............................................................. 22
4.5 Greyscale Colormap(0-255) ............................................... 23
4.6 Inverted Greyscale Colormap(0-255) .................................... 23
4.7 Log Colormap(0-255) ....................................................... 23
4.8 Heated Spectrum Colormap(0-255) ....................................... 24
4.9 Hot Metal Colormap(0-255) ............................................... 24
4.10 Bronson Colormap(0-255) ............................................... 24
4.11 Random Colormap(0-255) ............................................... 24
4.12 RGB Image Rendering ....................................................... 25
4.13 Histogram Example ......................................................... 28

5.1 Connectivity ................................................................. 31
5.2 Maximum Intensity Projection ............................................. 34
5.3 Left-handed Coordinate System ......................................... 35
5.4 Diagram indicating the MIP rendering technique. Rays are traced from every pixel on the projection plane into the volume image and the maximum intensity is taken to be the displayed value ....... 36
5.5 Example of Histogram Equalization .................................... 38
5.6 Profile plot of the image along the red line ............................ 39
List of Tables

3.1 Comparision table of the various programs .......................... 10
Chapter 1

Introduction

1.1 Motivation

The Image File System (IFS) is a set of subroutines and programs based on those subroutines that makes it easy to create and store images on the x86 (Microsoft Windows and GNU/Linux), Sparc (Sun Solaris) and the Power PC (Macintosh) without worrying about the underlying architecture of each platform. IFS images can be saved using a plethora of datatypes right from 8-bit character to 64-bit float including complex images. IFS also supports the creation of n-dimensional images (although the routines are optimized for 2 and 3 dimensions). IFS can be therefore used to store a variety of images right from simple 2-dimensional images to 3-dimensional images like MRI and CT slices.

This makes IFS well suited for research in image analysis and image processing. It can be used to write programs quickly and easily. There is, therefore, a need for a viewer to display, manipulate and visualize IFS images in number of ways interactively. Furthermore, the viewer should work on all of the platforms that IFS runs on. The viewer should also maintain the same functionality and feel on all the platforms so that user does not have to relearn the functionality on every platform.
This thesis deals with the design and development of such a viewer to display and manipulate IFS images natively. The viewer called \texttt{wxIFSView}, is designed to be cross platform and provides different ways to display and visualize IFS images. In preparation for the development of this tool, the author surveyed the literature for comparative products, developed the architecture and wrote the software, as well as wrote the user manual.

\section{1.2 \texttt{wxIFSView}}

\texttt{wxIFSView} is a state-of-the-art image display and processing software that allows the user to display and manipulate any IFS image on all the platforms supported by IFS. This section lists all the features of \texttt{wxIFSView}. The software was written using the \texttt{wxwidgets}\cite{wxWidgets} which is a cross platform Graphical User Interface(GUI) library. Because of the cross platform nature of the software, certain platform specific functions were omitted. This will be discussed in detail later.

- Cross platform support utilizing the platforms native rendering display.
- Support for loading display of all IFS images except images with the struct datatype.
- Support for loading and display of JPEG, TIFF and Lossless JPEG(LJPEG) images.
- Saving images in IFS(including data-type conversions), JPEG, TIFF, PNG and BMP file formats.
- Support for colormaps(7 as of now).
- Window/Level Adjustment
- Histogram support with equalization
- Extracting image intensities to a file.
• Image Zooming

• RGB Rendering

• Cine-play of multi-frame images

• Algorithms included as a part of wxIFSView are Connected Components labeling, Maximum Intensity Projection, Profile plots, Histogram Equalization, Thresholding and FFT computation.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 gives a brief description of the libraries used and their architecture. Chapter 3 compares various existing software packages with wxIFSView. Chapter 4 describes the architecture and design of wxIFSView. Chapter 5 describes the various built-in image processing algorithms in wxIFSView. Chapter 6 talks about future work and improvements in wxIFSView.
Chapter 2

Background Work

2.1 Image File System (IFS)

Image File System (IFS) is a set of libraries that can be used to manipulate images using C or C++ programs. IFS routines are currently optimized for handling 2 or 3 dimensional data, although the standard supports any number of dimensions. IFS uses its own file format for saving and loading images. IFS also takes care of byte swapping across various platforms and CPU architectures.

IFS contains libraries like iptools, with many commonly used image processing routines and the flip library, which contains floating point image operations optimized for speed.

IFS also comes with various command-line utilities for processing IFS images which includes various algorithms and file format conversion tools.

IFS is primarily used for writing programs quickly without the hassle of data type conversions and dealing with file input-output.

wxIFSView uses the IFS libraries for all its image manipulation operations.
2.1.1 The Structure of an IFS image

An IFS image, whether it is in a disk file or in program memory, is stored as a set of three distinct pieces. More detailed information about the IFS header structure can be found in the IFS manual[11].

The first piece is a header for the image. This header contains information relevant to the processing of the image, along with information intended solely for the user’s benefit. Sample items in the header include the number of dimensions the image has, the height and width of the dimension.

The second entity in an image is the actual image data. In memory, the data is just stored in one long linear array, in exactly the same way that any C program stores arrays. However, on disk, IFS images are compressed (starting with IFS release 6.0) using a variant of the Lempel-Ziv compression algorithm with the zlib library.

The third part of the image is the tail. The tail is just a block of data at the end of the file which IFS places no particular interpretation upon. A sample usage for the tail might be to store the text of a spoken message for which the data block was the digitized message, or the color map for 8-bit images.

2.1.2 Datatypes Supported

IFS supports a wide range of data-types which are listed below:

1. 8-bit signed/unsigned byte.
2. 16-bit signed/unsigned integer.
3. 32-bit signed Integer.
4. 32-bit unsigned Integer (not available on all platforms).
5. 32/64-bit floating point.
6. 32/64-bit complex with two 32 or 64 bit floating point numbers respectively.
7. Structure data type that can be any arbitrary user defined data-type.
2.2 WxWidgets

WxWidgets [7] is a cross platform GUI library which can used to write programs that run on Microsoft Windows, Linux and Mac OSX. WxWidgets uses the native window manager under each platform and thus, the programs have the same look and feel as other native programs on a platform.

The rendering architecture of wxIFSView is shown in Figure 2.1.

WxWidgets converts all the rendering calls from the program into calls specific to the current platform. This allows programs to take advantage of any hardware acceleration present on the platform and thus, speed up the rendering process.

Since Unix based systems have multiple window managers, wxIFSView uses GTK under Linux, Motif under Sun Solaris, Carbon under Mac OSX and Win32 under Microsoft Windows for maximum compatibility.
Chapter 3

Comparision With Existing Software Packages

This chapter looks at some of the image processing and image analysis software and compares them with wxIFSView.

3.1 IMP

The IMP is a X-windows program for interactive image display and manipulation. The IMP was written by Craig Hamilton of the Wake Forest school of medicine for the SUN under SunOS using the XView toolkit (Version 3). However, once Sun switched to Solaris the XView toolkit was no longer supported.

The IMP supports loading of IFS and RAW images and saving to IFS, RAW and Postscript.
3.1.1 Features

The features of the IMP include:

- Window/level adjustment
- Arbitrary colormaps
- Profile plots
- FFT computation
- Thresholding
- Support for multi-frame images.
- Maximum Intensity Projection
- Convolution
- Median Filtering

The drawbacks of the IMP are support for display up-to only ten images simultaneously. Furthermore, it is not cross platform and runs only on X-windows.

3.2 Java IFSView

Java IFSView is a Java based viewer which can load and display IFS images. This was written to by Dr. Wesley Snyder to provide interim capability between the time when the IMP was no longer usable and wxIFSView was completed. The features of the program are given below.

3.2.1 Features

- Cross Platform (Java based)
• Window/Level Adjustment

• Arbitrary Colormaps

• Support for Image Scaling (Zoom)

The drawback of Java IFSView is that its feature set is very limited and can be used only to view IFS images. No image processing or analysis is possible.

3.3 ImageJ

ImageJ[2] is very powerful image processing and analysis tool written in Java with goals similar to wxIFSView. This makes it fully cross platform and it runs on Windows, Linux, Mac OS X and even on the Sharp Zaurus PDA. ImageJ can load JPEG, TIFF, BMP, PNG, DICOM and PGM images. ImageJ supports the following datatypes:

• 8-bit greyscale

• 16-bit unsigned integer

• 32-bit floating point

• 32-bit RGB color

ImageJ has a extensible plugin architecture which makes it very feature rich. There are over 300 plugins available for ImageJ. Since it is java based, all plugins are cross platform and no recompilation is required.

The drawbacks of ImageJ are that there is no support for complex images.

Table 3.1 shows the comparison between the various programs with respect to the features available.
<table>
<thead>
<tr>
<th>Features</th>
<th>wxIFSView</th>
<th>IMP</th>
<th>Java IFSView</th>
<th>ImageJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Formats (Loading)</td>
<td>IFS, TIFF, JPEG, LJPEG</td>
<td>IFS, RAW</td>
<td>IFS</td>
<td>TIFF(Uncompressed), JPEG, BMP, PNG, PGM and DICOM</td>
</tr>
<tr>
<td>File Formats (Saving)</td>
<td>TIFF, PNG, JPEG, BMP, IFS</td>
<td>IFS, RAW, Postscript</td>
<td>IFS</td>
<td>TIFF(Uncompressed), JPEG, BMP, RAW</td>
</tr>
<tr>
<td>Data-types</td>
<td>All IFS data-types</td>
<td>All IFS data-types</td>
<td>All IFS data-types expect complex</td>
<td>8-bit,16-bit unsigned integer, 32-bit floating point, 32-bit RGB color</td>
</tr>
<tr>
<td>Number of Frames (3-d Images)</td>
<td>Infinite</td>
<td>Infinite</td>
<td>Infinite</td>
<td>Infinite</td>
</tr>
<tr>
<td>Number of images which can be loaded at a time</td>
<td>Limited only by system memory</td>
<td>Maximum of 10</td>
<td>Only one at a time</td>
<td>Limited only by system memory</td>
</tr>
<tr>
<td>Platforms Supported</td>
<td>Windows, Linux, Solaris, Mac</td>
<td>Solaris</td>
<td>Solaris, Windows</td>
<td>Windows, Linux, Mac</td>
</tr>
<tr>
<td>Image Processing Routines</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RGB Image Rendering</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support for plugins</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3.1: Comparision table of the various programs
3.4 Other Image Processing tools

The tools discussed below are powerful image processing tools in their own right but have different goals from wxIFSView. These tools have been designed to do photo retouching, image composition, image authoring or even serve as a general purpose programming environment. They are not designed specifically for performing image processing and analysis. These tools are being discussed for the sake of completeness.

3.4.1 ImageMagick

ImageMagick is a free software suite to create, edit and compose bitmap images. Most of the functionality is accessed from the command line and using programming languages. ImageMagick has interfaces to C, C++, Perl, Python, Java, Ruby, PHP and Tcl/Tk. ImageMagick supports over 90 file formats.

Features

Some of the features include:

- Convert between various file formats.
- Resize, crop and apply various effects to an image.
- Add shapes to an image like lines, polygons, Bezier curves, ellipses.

One of the primary drawbacks of ImageMagick is that it has no display capabilities.

3.4.2 Matlab

Matlab is a computer program for numerical computation. It began as a “MA-Trix LABoratory” program, intended to provide interactive access to the famous LINPACK and EISPACK libraries of state-of-the-art numerical routines. These are carefully tested, high-quality general-use packages for solving linear equations and
eigenvalue problems. The goal of Matlab was to enable scientists to use matrix-based
techniques to solve problems, without having to write programs in traditional lan-
guages like C and Fortran. More capabilities have been added as time has passed, in
particular outstanding graphics support. This makes it an ideal tool for implementing
and testing image processing and analysis algorithms. Matlab, in fact, comes with
an image processing toolbox with a lot of built-in algorithms.

Matlab is more of a programming language, rather than an image viewer.

3.4.3 GNU Image Manipulation Program (GIMP)

The GNU Image Manipulation Program (GIMP)\cite{1} is a graphical tool for manipu-
lating images. The Gimp can be used for a lot of applications such as photo retouching
and image composition. The GIMP is very expandable and extensible with support
for plugins. Just about any functionality can be achieved through the use of plugins.
The only downside to the program is that the user interface has a bit of learning
curve to it.

Furthermore, the Gimp has been designed for the artist rather than the scientist
making it suitable for artistic manipulation.

Numerous other packages like Photoshop, iPhoto etc. allow users to manipulate
images but they are designed to provide artistic manipulation rather than algorithm
evaluation.
Chapter 4

Architecture

This chapter describes the architecture of wxIFSView. Fig. 4.1 shows the overall architecture.

4.1 Imageframe Subsystem

wxIFSView has a complete Object Oriented design and is implemented in C++. Every image loaded into wxIFSView is processed separately and is independent of each other. Every image is controlled by an instance of the Imageframe subsystem. The Imageframe subsystem handles the drawing of the window frame on every platform and acts as the central controller. The Imageframe subsystem and the relationship with the other subsystems is shown in Figure 4.1. The arrows indicate the input and output relationship with each subsystem.

4.1.1 Initialization

The following steps highlight the initialization procedure of the Imageframe Subsystem.
1. The Imageframe subsystem is initialized with the filename of the image to be processed. The Imageframe subsystem passes the filename to the File Input/Output subsystem to load the file into memory.

2. The image statistics and the histogram of the image are computed. These are described later in section 4.7 and 4.6 respectively.

3. The rendering window is initialized.

4. The image is then passed on the Image Rendering subsystem for display on screen.

5. The Imageframe subsystem then enters the user input mode where further action is taken depending on the user input.

Figure 4.1: Overall Architecture
4.2 File Input/Output Subsystem

wxIFSView has the ability to read TIFF, JPEG and LJPEG files in addition to reading and writing IFS files. It can also export to BMP, PNG, TIFF and JPEG image file formats.

4.2.1 File Formats

1. **JPEG**: The JPEG (Joint Photographic Exchange Group) file format stores data using a lossy compression algorithm. JPEG supports 8-bit greyscale and 24-bit color images. The algorithm uses the DCT (Discrete Cosine Transform) followed by quantization where the high frequency components are reduced. This is because the human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. The final step in the compression process is entropy coding where Huffman or Arithmetic coding is usually employed. More information about the JPEG standard can be found in [12].

2. **Lossless JPEG**: Lossless JPEG is used in medical imaging applications where lossy compression methods are not acceptable. Lossless JPEG supports 8 or 16-bit greyscale images. The algorithm used is Nearest Neighbour prediction followed by Huffman coding. There are totally eight different predictors that can used. More information about lossless JPEG can be found in [12].

3. **TIFF**: The Tagged Image File Format (TIFF) supports both greyscale (4 or 8 bit) and up to 64 bit color images. The TIFF format has support for lossless, lossy and no compression. Most common TIFF readers use JPEG for lossy compression and the Lempel-Ziv-Welch (LZW) algorithm for lossless compression. More information about the TIFF file format can be found in [6].

4. **PNG**: Portable Network Graphics (PNG) is an image file format that supports greyscale images up to 16-bit, 24 and 48-bit color RGB images and palette-indexed color. PNG uses the deflate algorithm for compression combined with
prediction resulting in lossless compression. More information about PNG can be found in [5].

5. **BMP**: BMP is a bitmapped graphics format that allows color depths of 1-bit, 2-bit, 4-bit, 8-bit, 16-bit and 24-bit RGB images. All BMP images have a color table which contain as many colors as specified by the color-depth($2^n$). 24-bit bitmaps do not have color tables as they have the red, green and blue values in the actual bitmap data area. The images are usually not compressed though simple Run Length Encoding(RLE) is supported.

### 4.2.2 File Loading

The entire file loading is done in the initialization of the Imageframe subsystem. Currently, wxIFSView has the ability to load IFS, JPEG, TIFF and Lossless JPEG(LJPEG) files. However, addition of other file formats is also possible.

The IFS file input routines are implemented using the file I/O routines found in the IFS library.

The support for the JPEG file format is provided using the *libjpeg* library included with the IFS libraries. The support for TIFF is provided using the image I/O routines in wxWidgets[7]. Lossless JPEG(LJPEG) support is provided by code adapted from the PVR JPEG codec [9].

Internally, two copies of the image are maintained and all the processing is done using the IFS libraries. One copy is in the original data format as stored in the IFS file. This is the Master copy. The other copy is a floating point version of the original. The floating point version is used for all processing and display. The original is never modified.

JPEG files are converted to a 3 frame(RGB) unsigned 8-bit IFS format while loading. TIFF files are converted to a 3 frame(RGB) floating point IFS format by the TIFF loader in wxWidgets[7]. LJPEG files are converted to single frame unsigned short(16 bit) IFS format to retain the same data format in which they are stored on disk. In the current release, we use the device dependent RGB color space of the
display device. Images using other color spaces are converted to the display device dependent RGB color space before processing by the image I/O routines described earlier.

For the purpose of display, a vector valued image like a complex image must be converted to a scalar image. wxIFSView provides four different such conversions. They are:

1. **Real**: The real portion of the image pixels is used for display.
2. **Imaginary**: The imaginary portion of the image pixels is used for display.
3. **Magnitude**: The magnitude of the image pixels is used for display.
4. **Phase**: The phase of the image pixels is used for display.

The original complex image is still retained in the Master copy.

The entire file loading process is shown in Figure 4.2.

### 4.2.3 IFS File Output

wxIFSView has the ability to read, write and convert between any type of IFS datatype and thus, a separate subsystem is used for IFS file output. This component is used to write a IFS file to disk depending on the data type the user decides. The file I/O is performed using the IFS libraries. IFS writes files in the format native to the particular platform on which it is running (prior to compression). All cross-platform conversions are performed on read.

### 4.2.4 File Output to Other Formats

The third major component handles file output to other file formats apart from IFS. The file formats currently supported are BMP, PNG, TIFF and JPEG. The file I/O here is performed using the image loading and saving routines in wxWidgets[7].
Figure 4.2: File Loading Process
4.3 Image Rendering Subsystem

This section describes the Image Rendering Subsystem of wxIFSView. The Image Rendering Subsystem takes care of the display of the image on the screen by using the desired scaling and colormap parameters. The subsystem can display both single frame and multi-frame images. The image to be displayed is always in floating point IFS data format.

The following subsections describes the various steps performed in the display of the image.

4.3.1 Window and Level

A Window is defined as the range of pixel values around a fixed Level that are scaled to 0-255 for display on screen. The combination of Window and Level controls the visible pixel range of the image. Figure 4.3 illustrates the concept of window and level.

Any pixel value that does not lie within the Window is truncated to the end points
of the window. Mathematically, this is given by:

\[
c_{WL} = \begin{cases} 
    l - w/2 & c > l - w/2 \\
    c & l - w/2 \leq c \leq l + w/2 \\
    l + w/2 & c > l + w/2
\end{cases}
\] (4.1)

where,

- \( c_{WL} \) is the image pixel value after window/leveling.
- \( c \) is the image pixel value.
- \( w \) is the Window size.
- \( l \) is the Level.

Window and Level functions are performed before scaling the image for display.

### 4.3.2 Intensity Scaling

Since IFS is capable of handling images in a variety of data formats, the image must be scaled before it can be displayed on screen where the intensity level varies from 0-255. We can scale the image pixels to lie between 0-255 by using a Linear scaling procedure.

\[
c_{scaled} = 255 \times \frac{(c - c_{min})}{(c_{max} - c_{min})}
\] (4.2)

where \( c \) is the pixel value to be scaled. \( c_{max} \) and \( c_{min} \) are the maximum and minimum pixel values in the image.

Usually, the values of \( c_{min} \) and \( c_{max} \) are obtained from the frame which is being displayed. This is called Auto Scaling.

For a multi-frame image, it is possible to obtain the values of \( c_{min} \) and \( c_{max} \) using all the frames in the image. This is called Global Scaling. Global Scaling is equivalent to Auto Scaling for single frame images.

The default scaling procedure for all images is Auto Scaling.

In the current release of wxIFSView, non-linear scaling operations such as Gamma Correction are not supported, although equivalent functionality can be obtained
4.3.3 Colormaps

The final step before displaying the image on screen is applying a colormap. Colormaps are lookup tables which are used to visualize the image in a number of ways. Colormaps remap the pixels values in the image to a 3 color vector(RGB). wxIFSView supports 7 different colormaps as of now and can be extended to support more. The 7 colormaps when applied to an image are shown in Figure 4.4.

1. **Greyscale**: This is the default colormap in wxIFSView. The greyscale colormap is defined as follows.

   \[
   \text{colormap}(i) = \begin{bmatrix} 
   i \\ 
   i \\ 
   i 
   \end{bmatrix} 
   \] (4.3)

   where \( i \) is the scaled image pixel which varies from 0 – 255. That is, a pixel whose brightness is, say, 35, will be displayed as a pixel with red, green, and blue components of 35, 35, and 35. The greyscale colormap is shown in Figure 4.5.

2. **Inverted Greyscale**: The inverted greyscale colormap is defined as follows.

   \[
   \text{colormap}(i) = \begin{bmatrix} 
   255 - i \\ 
   255 - i \\ 
   255 - i 
   \end{bmatrix} 
   \] (4.4)

   where \( i \) is the scaled image pixel which varies from 0-255. The inverted greyscale colormap is shown in Figure 4.6.
3. **Log Colormap:** The Log Colormap converts the pixels values to a logarithmic scale. This is useful for visualizing data like Fourier transforms. Mathematically,
it is defined as:

$$colormap(i) = \begin{bmatrix} 255 \times \log_{256}(i) \\ 255 \times \log_{256}(i) \\ 255 \times \log_{256}(i) \end{bmatrix}$$ (4.5)

where $i$ is the scaled image pixel which varies from 0-255. The Log colormap is shown in Figure 4.7.

4. **Heated Spectrum:** This colormap assigns color values as shown in Figure 4.8.

5. **Hot Metal:** This colormap assigns color values as shown in Figure 4.9.

6. **Bronson Colormap:** This colormap does color quantization on the image and reduces the number of the colors present in the image. Currently, the
number of colors to which the image is quantized to is 10. The colormap is shown in Figure 4.10.

7. **Random Colormap:** This colormap assigns a random RGB value to each pixel value. This is particularly useful for viewing the noise in an image. The colormap is shown in Figure 4.11.
4.3.4 Multiframe Images

wxIFSVView can display multiframe images as individual frames or as RGB images where the user can select the red, green and blue channel from any of the frames in the image. By default, the first three frames are taken as the red, green and blue channels for RGB display. The device dependent RGB color space of the display device is used where no color space conversions are performed before display. No colormap is applied when the image is displayed in RGB mode. Window/leveling and scaling are still done. A minimum of three frames is required for display in RGB mode. Figure 4.12 shows a multiframe image and its RGB rendering.

![Figure 4.12: RGB Image Rendering](image)
4.3.5 Image Resizing

The image which is finally displayed is resized depending on a user defined zoom factor which indicates whether the image is upscaled or downscaled. The zoom factor is an integer which can be used to indicate both upscaling and downscaling. The reason for an integer zoom factor is because the input widget used in wxWidgets[7] accepts only integer values and does not accept floating point values. Zoom-in is done using Nearest Neighbor interpolation and Zoom-out uses simple sub-sampling.

The actual zoom factor is calculated as follows:

\[ S = \frac{Z}{10} \]  

(4.6)

where \( Z \) is the integer zoom factor and \( S \) is the actual zoom factor. Such a scheme allows the user to zoom-in and zoom-out using one control parameter. The new height and width of the image after scaling are rounded off to the nearest integer. \( S \) less than 1 indicates downscaling and \( S \) greater than 1 indicates upscaling. For example, a zoom factor \( Z \) of 20 gives a \( S \) value of 2 which means the image will be upscaled by a factor 2. Similarly, a zoom factor \( Z \) of 5 will result in a \( S \) value of 0.5 which means the image will be downscaled by a factor of 2.

4.4 User Input Subsystem

The User Input Subsystem handles the keyboard and mouse inputs from the user and passes them to the Imageframe subsystem for further processing. The entire keyboard and mouse input is handled through wxWidgets to maintain cross platform compatibility. The user guide included in the appendix lists some of the keyboard shortcuts and the mouse actions used in wxIFSView.
4.5 Algorithms

Various image processing and image analysis algorithms can be run on the image being displayed. The algorithms currently implemented are described in detail in the next chapter.

4.6 Histograms

wxIFSView maintains a histogram of the image displayed internally which can be displayed and used in algorithms. The histogram uses 256 bins. This allows a complete histogram for commonly found 8-bit unsigned images. In images with more grey levels, the pixel values in the image are linearly scaled to values between 0-255 as described in section 4.3.2 before the histogram is computed. Future releases of wxIFSView will support user defined bin sizes. A separate histogram is computed for each frame in multi-frame images. Figure 4.13 shows an example of the histogram of an image with more than 256 grey levels.

4.7 Image Statistics

WxIFSView also computes some basic parameters of the image displayed which can be used by any algorithm. The following parameters are computed:

1. Mean of the image:

\[
\mu = \frac{1}{N} \sum_{i,j} f(i,j) \quad i, j \in Z
\]  

(4.7)

where \( N \) is the total number of pixels in the image \( f(i,j) \).

2. Standard Deviation of the image:

\[
\sigma = \sqrt{\frac{1}{N} \sum_{i,j}(f(i,j) - \mu)^2} \quad i, j \in Z
\]  

(4.8)
where $N$ is the total number of pixels in the image $f(i,j)$. This is a biased estimate of the actual standard deviation.

3. Maximum pixel value in the image

4. Minimum pixel value of the image
The default values for window and level are computed from the maximum and minimum values of the image as follows:

\[
\text{window} = \text{maximum pixel value} - \text{minimum pixel value} \quad (4.9)
\]

\[
\text{level} = \text{minimum pixel value} + \frac{\text{window}}{2}; \quad (4.10)
\]
Chapter 5

Built-in Algorithms

wxIFSView allows the user to run various image analysis and image processing algorithms on images displayed. The architecture allows any number and type of algorithms to be implemented. This chapter describes some of the algorithms that have been implemented.

5.1 Connected Component Labeling

Connected Components are used to identify different regions in a segmented image. They are used to group pixels into regions based on pixel connectivity. The output will be a label image where each region has a different label. The basic principle here is to group adjacent pixels with similar pixel intensities using a common label.

5.1.1 Connectivity

Two pixels are defined as being in the same region if they are adjacent (as defined below) and have similar brightness. The algorithm implemented has 3 kinds of adjacencies defined.
1. **Face Adjacency:** In two dimensions, there are 4 face adjacent neighbors. In three dimensions, a voxel (three dimensional pixel) has 6 face adjacent neighbors.

2. **Edge Adjacency:** A two-dimensional pixel has 4 edge adjacent neighbors. In three dimensions, a voxel has 12 edge connected neighbors.

3. **Vertex Adjacency:** In two dimensions, there are no vertex adjacent neighbors. In three dimensions, there are 8 vertex adjacent neighbors.

These are shown in Figure 5.1. One thing to note is that the algorithm uses only those pixels that have been labeled when checking for connectivity.

![Figure 5.1: Connectivity](image)

5.1.2 **Algorithm**

The algorithm used in wxIFSView uses an iterative region growing approach as described in [10]. The algorithm is implemented using a modified version of the ifs subroutine $CCL$.

Let $f(x, y), (x, y) \in Z^2$ be the image on which the labeling is to be performed.

1. The label image $L(x, y)$ is first initialized to zero. Each entry in the equivalence array($K$) is initialized with its own index i.e. $K[i] = i$. The equivalence array keeps track of the equivalent labels.
2. The image is scanned from left to right and top to bottom, frame by frame (for multiframe images).

3. A pixel is considered to be connected to another neighboring pixel if they are face, edge or vertex neighbors and satisfy the following criteria.

\[ |f(x, y) - f_i(x, y)| < T \]  \hspace{1cm} (5.1)

where \( T \) is a threshold that is user modifiable and \( f_i \) is a neighboring pixel to \( f \).

4. If a pixel not connected to one or more of its neighbors then it is assigned a new label and the next pixel is processed.

5. If a pixel is connected to one or more of its neighbors and all those neighbors have the same labels, then it is assigned the label of its connected neighbors.

6. If a pixel is connected to more than one neighbor with different labels, then the equivalence array is updated to indicate that all the labels are equivalent. Two equal labels are resolved by updating all the instances of the greater label in the equivalence array with the entry with the smaller label. The pseudo code for label resolution is shown in algorithm [1].

\textbf{Input:} \( L(x, y) \) - Label of pixel at \((x,y)\)

\textbf{Input:} Labels - Labels of the connected pixels

Calculate the number of connected labels

\( \text{LengthLabels} = \text{Length} (\text{Labels}) \)

Resolve labels if connected to more than 1 neighbor

\( \text{if} \ \text{LengthLabels} > 1 \ \text{then} \)

\( \qquad \text{NewLabel} = \text{Labels}[0] \)

\( \qquad \text{for} \ i \leftarrow 1 \ \text{to} \ \text{LengthLabels} \ \text{do} \)

\( \qquad \qquad \text{NewLabel} = \text{ResolveTwoLabels} (\text{NewLabel}, \text{Labels}[i]) \)

\( \qquad \quad L(x, y) = \text{NewLabel} \)

\textbf{Algorithm 1: Resolving Labels}
7. At the end of one pass, the labels in the label image \( L(x, y) \) are relabeled by mapping them to the correct labels using the equivalence array \( K \) as shown in algorithm [2].

\[
\text{for } \text{Every pixel } p \in L(x, y) \text{ do} \\
L(x, y) = K[L(x, y)]
\]

\textbf{Algorithm 2: Relabeling the Label Image}

5.1.3 Region of Interest

The usual definition of Region of Interest is a spatial area explicitly defined by the user. Here, the Region of Interest is the region a user defined pixel belongs to. The connected components algorithm can be used to extract a particular region to which a pixel belongs. This is done by determining the label of the pixel using the Label image obtained after connected component analysis and then extracting all the pixels with the same label.

5.2 Maximum Intensity Projection

Maximum Intensity Projection is a commonly used volume visualization technique. The algorithm uses ray tracing to project parallel rays from a projection plane through the volume object (multi-frame image). The maximum pixel value is calculated among pixels at unit distances along the ray trace which is then projected on the projection plane. It is possible that certain pixels are skipped because of unit sampling. Future versions of wxIFSVView will include a user modifiable sampling parameter to overcome this limitation. Figure 5.2 shows the basic principle of the algorithm.

This is a “winner-takes-all” algorithm and is more of a search algorithm compared to traditional ray tracing. Maximum intensity projection can be used to identify long, thin and bright objects in a volume image.
There is, however, loss in spatial information as only the highest intensity is displayed on projection plane. This is overcome by casting rays from different directions on the volume image. This will produce multiple images that are animated while viewing. The human brain integrates all the frames and makes the image appear three dimensional.

### 5.2.1 Ray Tracing

Consider the figure shown in 5.2. Let a point on the projection plane be \( S = (x_s, y_s, z_s) \). Let the unit normal to the projection plane be \( N = (x_n, y_n, z_n) \). Then, the equation of the ray originating at \( S \) and normal to the plane is given by:

\[
P(t) = S + tN \quad t \geq 0
\]

where \( P(t) = (x(t), y(t), z(t)) \) is any point on the ray.
5.2.2 Algorithm

The Maximum Intensity Projection rendering technique proceeds as follows. A left-handed coordinate system is used and is shown in Figure 5.3.

1. The initial location of the projection plane and the number of frames are obtained from the user. There are two possible starting locations for the projection plane:

   - **XY Plane**: The projection plane initially located in the XY-Plane and is then rotated about the Y-axis.
   - **YZ Plane**: The projection plane is initially located in the YZ-Plane and is then rotated about the Z-axis.

2. Rays are cast from every pixel on the projection plane through the volume image. The maximum pixel intensity along each ray trace is obtained and is stored at the corresponding projection plane location. This is done by calculating the maximum pixel intensity among pixels that are at unit distances along the ray trace. This is shown in Figure 5.4.

3. The projection plane is then rotated and the above steps are repeated to obtain the next image.

4. The images obtained are then animated to visualize the final rendering.
Figure 5.4: Diagram indicating the MIP rendering technique. Rays are traced from every pixel on the projection plane into the volume image and the maximum intensity is taken to be the displayed value

5.3 Histogram Equalization

Histogram Equalization is a commonly used image enhancement technique[8]. The basic principle here is to find a greyscale transformation which maps histogram of an image to a histogram that is constant for all brightness values.

Consider an image $F$ with grey level values $r$ with $0 \leq r \leq 1$. Our objective here is to determine an transformation $s = T(r)$. It is assumed that $T$ is a monotonically increasing function with $0 \leq T(r) \leq 1$.

Consider a probability density function $p(r)$. We know from probability theory that a probability density function $p(s)$ where $s = T(r)$ is given by:

$$p_s(s) = \left[ p_r(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)}$$  \hspace{1cm} (5.3)

Consider the transformation given below:

$$s(r) = \int_{0}^{r} p_r(w)dw$$  \hspace{1cm} (5.4)

Finding the derivative of $s$ in equation 5.4 yields,

$$\frac{ds}{dr} = p_r(r)$$  \hspace{1cm} (5.5)
Substituting for \( dr/ds \) in equation 5.3 gives,

\[
p_s(s) = \left[ p_r(r) \cdot \frac{1}{p_r(r)} \right]_{r=T^{-1}(s)}
\]

\[
\Rightarrow p_s(s) = 1 \quad 0 \leq s \leq 1
\]

which is a uniform density in the interval of the transformed variable \( s \).

The discrete formulation is as follows.

\[
P_r(i) = \frac{n_i}{N} \quad i = 0, 1, \ldots, L-1
\]

where \( L \) is the number of grey levels, \( n_i \) is the total number of pixels with a grey level of \( i \) and \( N \) is the total number of pixels in the image. \( P_r(i) \) is the probability of the \( i \)th grey level in the image. This is the normalized histogram of the image.

Therefore, the discrete equivalent of the transformation function is given below:

\[
s_k = (L-1) \sum_{j=0}^{k} P_r(j) \quad k = 0, 1, \ldots, L-1
\]

where \( (L-1) \) is a scaling factor which makes sure that \( s_k \) varies from 0 to \( L-1 \). \( s_k \) is also rounded off to the nearest integer. Since, the histogram is only an approximation to the probability density function, a perfectly uniform histogram is rarely achieved.

Figure 5.5 shows an example of histogram equalization.

### 5.4 Profile plot

Profile plots are used to look at the variation of pixel intensity along a line. The pixel values at unit distances along a user defined line are plotted from the start point to the end point. Figure 5.6 shows an example of a profile plot.
5.5 Simple Thresholding

Thresholding is a very simple and easy technique to separate an image into foreground and background pixels. wxIFSView allows the user to pick a threshold from the histogram. The image is then divided into two regions as follows:

\[
I(x, y) = \begin{cases} 
0 & I(x, y) < T \\
255 & I(x, y) \geq T 
\end{cases} \quad (5.9)
\]

5.6 FFT and Inverse FFT

wxIFSView also has functions that compute the Discrete Fourier Transform of the image using the Fast Fourier Transform (FFT) and the Inverse Fast Fourier Transform (IFFT). The FFT is an efficient algorithm to compute the DFT of an image.

This is done using the \texttt{cfft2d} function in iptools that is part of the IFS library.
The prototype for the \texttt{cfft2d} function is given below:

\begin{verbatim}
int cfft2d(IFSIMG image, int direction)
\end{verbatim}

where image is the input image (has to be complex) for which the FFT or IFFT is to be computed. A direction of -1 indicates forward FFT and +1 indicates inverse FFT.

The FFT and the inverse FFT algorithms are optimized for images whose dimensions are a power of two. Therefore, the image is automatically resized by \texttt{wxIFSView} to the closest power of two before being passed to the function for both the forward
and the inverse FFT algorithms. This is done by padding the image with zeros. The input image is converted to the complex datatype by wxIFSView before being passed to the function. The output image is also complex.
Chapter 6

Summary and Future work

6.1 Summary

This thesis looked at design of a state-of-the-art interactive image processing tool called wxIFSView, that was able to display and process images of a wide variety of data-types particularly Image File System (IFS) images in many different ways. Various features which are commonly used in image processing were implemented along with many common algorithms. The tool designed was able to run on many different platforms using a common code base.

6.2 Future Work

Although, the tool has many different features, there is a lot of scope for future expansion. Here are some possible futurework.

- The modular architecture of wxIFSView can be utilized to increase the feature set and add more algorithms to the program.
• The image statistics could be obtained for a user defined region rather than the entire image.

• Loading of arbitrary colormaps from a file.

• Add support for non-linear scaling.
Bibliography


Appendix A

wxIFSView User Guide

The complete user guide for wxIFSView is attached here.
wxIFSView User Guide

Version 2.00
Contents

1 Introduction 49

2 Running wxIFSView 50
   2.1 System Requirements ........................................... 50
   2.2 Running ............................................................. 51

3 Using wxIFSView 52
   3.1 Base Window ...................................................... 52
   3.2 Image Window ..................................................... 54
   3.3 The Menu ........................................................... 55
      3.3.1 File ........................................................... 56
      3.3.2 Edit ........................................................... 57
   3.4 Control Panel ..................................................... 59
      3.4.1 Colormap ...................................................... 59
      3.4.2 Zooming ...................................................... 61
      3.4.3 Window/Level ................................................. 62
      3.4.4 Animation Speed ............................................. 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.5 Color Bands</td>
<td>63</td>
</tr>
<tr>
<td>3.5 Built-in Algorithms</td>
<td>63</td>
</tr>
<tr>
<td>3.5.1 Connected Component Labeling</td>
<td>63</td>
</tr>
<tr>
<td>3.5.2 Maximum Intensity Projection</td>
<td>65</td>
</tr>
<tr>
<td>3.5.3 Profile Plot</td>
<td>66</td>
</tr>
<tr>
<td>3.5.4 Segmentation</td>
<td>67</td>
</tr>
<tr>
<td>3.5.5 Fast Fourier Transform and Inverse Fast Fourier Transform</td>
<td>68</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This document is the official user-guide to wxIFSView. wxIFSView is a cross-platform IFS(Image File System) image viewer. The program has been ported to work on Win32, Sun, Linux and the Macintosh platforms. wxIFSView can be used to analyze, visualize and process IFS images.

wxIFSView uses the wxWidgets GUI toolkit to achieve cross-platform compatibility. wxWidgets makes it easy to write code on a single platform and compile the same code on the other platforms with minimal or no changes in the code. wxWidgets uses the native Graphical User Interface(GUI) of the platform for its display.

This document is organized as follows. Chapter 2 describes how to run the program on the different platforms. Chapter 3 describes the usage of wxIFSView in depth.
Chapter 2

Running wxIFSView

This chapter contains information on how to start and run wxIFSView.

2.1 System Requirements

wxIFSView has been tested and verified to work on the following platforms.

- Microsoft Windows XP, Microsoft Windows 2000, Windows 95, 98 and ME.
- Sun Solaris 8
- Redhat Enterprise Linux, Fedora and Slackware Linux 10. It should work on any recent linux distribution.
- Mac OSX 10.3+ (Tiger Compatible)
2.2 Running

wxIFSView is included with the the IFS libraries and is found in the /ifsbin folder of the IFS package. It is called wxIFSViewPLAT where PLAT indicates the platform which can be one of the following: Motif (Sun Solaris), GTK (Linux), Win32 (Microsoft Windows) and MacX (Macintosh).

It can be run either by executing it from the console (under Linux and Solaris) or by clicking on the wxIFSView icon (under Windows or Mac). One can also specify the IFS image to be viewed on the command line but this is optional. The command line usage is shown below:

`wxIFSViewPLAT Image-File`

where *Image-File* can be a image stored in one of the following file formats: IFS, JPEG, LJPEG or TIFF.

One can also associate Image files to be opened by wxIFSView on the Win32 and the Mac platforms. Please refer to your operating system documentation for more information.
Chapter 3

Using wxIFSView

wxIFSView uses a Single Document Interface(SDI) where each image has its own window and tools. There is one base window which is used for file-open operations. Whenever a new file is opened, the image is loaded in a new separate window. The base window and image window are shown in the Figure 3.1 and 3.2.

The program has the same layout and functionality across all platforms.

3.1 Base Window

The base window is used to open files for display and processing. The following file formats are supported as of now:

- IFS
CHAPTER 3. USING WXIFSVIEW  

Figure 3.1: The Base Window

Figure 3.2: The Image Window
• JPEG

• TIFF

• Lossless JPEG (LJPEG)

The image is opened in a new image window described in the next section. Large images are automatically scaled to fit in the image window for display.

### 3.2 Image Window

The Image window has 3 main components.

- The toolbar on the top.

- The image display area in the middle.

- The status bar at the bottom.

The first field of the status bar in the image window shows the height, width, the number of frames in the image followed by the frame currently displayed. The current image scale factor is also displayed in this field.

The second field shows the pixel value of the pixel selected and the color it is mapped to for on-screen display. The third field shows coordinates of the pixel selected.

A pixel can be selected by clicking on the image with the mouse. For multi-frame images, one can move between frames as follows:
• To move to the next frame, the keyboard shortcut is Shift+Right Arrow.

• To move to the previous frame, the keyboard shortcut is Shift+Left Arrow.

Vector valued images like complex IFS images have to be converted to a scalar for display purposes. wxIFSView provides four such conversions. They are:

1. **Real**: The real portion of the image pixels is used for display.

2. **Imaginary**: The imaginary portion of the image pixels is used for display.

3. **Magnitude**: The magnitude of the image pixels is used for display. This is the default.

4. **Phase**: The phase of the image pixels is used for display.

### 3.3 The Menu

This section describes the menu items in wxIFSView.
3.3.1 File

Save as IFS Image

This allows the user to save the image as a new IFS file. One can specify the data type of the saved IFS image.

Internally, two copies of the image are maintained. One is the master copy which is in the native data format as stored on disk and the other is the display copy which is in floating point IFS format. The display copy is used for all processing and display and the master copy is never modified.

The master copy is used here when saving as a new IFS file.

Save Select Frames as IFS Image

If the image has multiple frames, then this allows the user to save selected frames as a new IFS Image. No datatype conversion is performed and the image is saved in its original data format. The master copy is used here.

Export Displayed Image as

This allows the user to save the displayed image as JPEG, BMP, TIFF, PNG or IFS. The display image copy is used here. Other utilities exist in IFS for conversion to other formats like AVI.

Close

This closes the image window.
3.3.2 Edit

Undo

Restores the displayed image to the original image. The display image is regenerated using the master copy.

View As RGB Rendering

If the image has more than 2 frames, then its is rendered as an RGB image using the frames set in the options dialog. The default choice is to use the first 3 frames.

Animate

Animates a multi-frame image i.e. the frames are displayed in a time sequence, as a movie.

Controls

Displays the control panel where various options can be set. This is described later in section 3.4.

Show Histogram

Displays the Histogram of the current frame displayed. One can click anywhere on the histogram for the exact pixel values. The histogram uses 256 bins.
This menu item is a toggle and can be used to turn on and turn off the histogram window.

**Equalize Histogram**

Equalizes the histogram of the currently displayed frame.

**Autoscale Image**

wxIFSView maps the grey levels in the image to values between 0 and 255 for displaying it on screen. Autoscaling uses the current frame to compute the maximum and minimum pixel values to map the colors.

**Globalscale Image**

Similar to Autoscale expect that it uses the maximum and minimum pixel values from the entire image across all frames. It is equivalent to Autoscale for single frame images.

**Record Clicks**

This allows the user to save selected image intensities to a file. Once record clicks is enabled, the user can click anywhere in the image with the mouse to save the image intensity at a point to a file.

The file is a text file with each line contain information about the intensity at the point of click. The format of each line is shown below:

Frame x y pixel-intensity
The file name of the saved intensities is the same name as the IFS file but with an appended extension of .clicks.log.

3.4 Control Panel

Figure 3.3 shows the control panel dialog.

3.4.1 Colormap

This allows the user to pick the colormap to apply on the displayed image. The following colormaps are currently available:

1. Greyscale
2. Inverted greyscale
3. Log
4. Heated Spectrum
5. Hot Metal
6. Bronson
7. Random

Figure 3.4 shows the colormaps as applied to an image. The default colormap is greyscale.
Figure 3.3: The Control Panel
3.4.2 Zooming

The zooming control allows the user to zoom-in and zoom-out of the image. A zoom factor of 10 indicates a 1:1 scale. Any value less than 10 will downscale...
the image and any value greater than 10 will upscale the image. For example, a zoom factor of 20 will upscale the image by a factor of 2 and zoom factor of 5 will downscale the image by a factor of 2.

### 3.4.3 Window/Level

These parameters control the visible pixel range of the image displayed on the screen. *Window* is defined as the range of pixel values around a fixed *Level* which are scaled to 0-255 for display on screen. Figure 3.5 illustrates the concept of window and level.

![Figure 3.5: Window/Level Example](image)

The default window/level are calculated as follows:

\[
\text{window} = \text{maximum pixel value} - \text{minimum pixel value} \quad (3.1)
\]

\[
\text{level} = \text{minimum pixel value} + \frac{\text{window}}{2}; \quad (3.2)
\]
3.4.4 Animation Speed

This option appears for multi-frame images and sets the speed at which they are animated. The default value is 30 frames/sec.

3.4.5 Color Bands

This allows the user to select the frames to be used for RGB rendering in multi-frame images. The default value is the first 3 frames for red, green and blue respectively.

3.5 Built-in Algorithms

This section describes the algorithms and their parameters which can be run on the displayed image. All algorithms are under the Tools menu.

3.5.1 Connected Component Labeling

Connected component labeling is used to group pixels into components based on pixel connectivity. The output is a label image where each component has a different label.

Parameters

Figure 3.6 shows the parameters which can be set for connected component labeling.
CHAPTER 3. USING WXIFSVIEW

![Connected Components Options Dialog](image)

**Figure 3.6: Connected Components Options Dialog**

1. **Lower Background Threshold**: All pixel values below this value are considered to be part of the background and are ignored while labeling.

2. **Upper Background Threshold**: All pixel values above this value are considered to be part of the background and are ignored while labeling.

3. **Lower Background Label**: The label to be assigned for pixels below the lower background threshold.

4. **Upper Background Label**: The label to be assigned for pixels above the upper background threshold.
5. **Threshold:** The difference between the values of two adjacent (described below) pixels. If the difference is below the threshold, then they are considered to be connected.

6. **Connectivity:** The type of pixel adjacency to be used while labeling. Figure 3.7 shows the different types of adjacencies supported.

7. **Enable ROI:** This enables Region of Interest (ROI) support where the region corresponding to the current pixel selected (the last pixel clicked on) is extracted.

![Figure 3.7: Adjacency](image)

(a) Face Adjacency  
(b) Edge Adjacency  
(c) Vertex Adjacency

### 3.5.2 Maximum Intensity Projection

Maximum Intensity Projection is a volume visualization technique which traces parallel rays from a projection plane through the volume object and projects the maximum intensity along the ray on the projection plane. The loss in spatial information is made up by casting rays from different angles through the volume object to produce multiple images which are animated.
while viewing. The human brain integrates all the frames and makes the image appear three dimensional.

Parameters

1. **No. of Frames:** The number of images to produce.

2. **Viewing Plane location:** There are two possible starting locations for the projection screen:
   - **XY Plane:** The projection plane initially located in the XY-Plane and is then rotated about the Y-axis.
   - **YZ Plane:** The projection plane is initially located in the YZ-Plane and is then rotated about the Z-axis.

   The output is a multi-frame floating point IFS image which contains the maximum intensity projection of the volume image from different angles.

### 3.5.3 Profile Plot

Profile plots are used to look at the variation of pixel intensity along a line. The line can be traced with the mouse on the image. The pixel values along the user defined line is plotted from the start point to the end point. Figure 3.8 shows an example of a profile plot.


3.5.4 Segmentation

Simple segmentation using thresholding can be done in wxIFSView. The user can pick the threshold by clicking on the histogram of the image and then use the Segment option to do simple thresholding. Thresholding can also be done on multi-frame images by using the same threshold across all

Figure 3.8: Profile plot of the image along the red line
the frames. This is done using the Segment 3d option under the Tools menu.

3.5.5 Fast Fourier Transform and Inverse Fast Fourier Transform

The Fast Fourier Transform (FFT) and the Inverse FFT of the image can be obtained in wxIFSView. The image is automatically resized to the closest power of two by zero padding.

The output is a complex IFS image and thus, the magnitude and phase images can be viewed separately. The log colormap is best suited for visualizing the magnitude of the FFT.