VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belgaum-590 014

2012 - 2013

A Report on

"Implementation of Image Convertor for Popular Image formats"

Submitted to RVCE (Autonomous institution affiliated to Visvesvaraya Technological University (VTU), Belgaum) in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING
by

Praveen Kumar D	**1RV09CS074**
Sham Prasad P S	**1RV09CS098**
Vikas Itnal	**1RV09CS117**
Mahanthesh Babu H S	**1RV10CS407**

Under the guidance of

Mr. Sesha Kalyur
Visiting Professor,
Department of CSE, RVCE,

R. V. College of Engineering,
(Autonomous institution affiliated to VTU)
Department of Computer Science and Engineering,
Bangalore – 560059

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

R.V. COLLEGE OF ENGINEERING,

(Autonomous institution affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mysore Road, R.V.Vidyaniketan Post, Bangalore - 560059

CERTIFICATE

Certified that the project work entitled " **Implementation of Image Convertor for Popular Image formats"** has been carried out by **Mr. Praveen Kumar D** (USN: **1RV09CS074**), **Mr. Sham Prasad P S** (USN: **1RV09CS098**), **Mr. Vikas Itnal** (USN: **1RV09CS117**) and **Mr. Mahanthesh Babu H S** (USN: **1RV10CS407**), bonafide students of **R.V. College of Engineering, Bangalore** in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the **Visvesvaraya Technological University, Belgaum** during the year 2012-2013. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirement in respect of project work prescribed for the said Degree.

Mr. Sesha Kalyur
Visiting Professor,
Department of CSE,
R V C E, Bangalore –59

Dr. N. K. Srinath
Head of Department,
Department of CSE,
R V C E, Bangalore –59

Dr. B. S. Satyanarayana
Principal,
R V C E,
Bangalore –59

Name of the Examiners	**Signature with Date**
1.____________________	____________________
2.____________________	____________________

VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM

R.V. COLLEGE OF ENGINEERING,

(Autonomous institution affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mysore Road, R.V.Vidyaniketan Post, Bangalore - 560059

DECLARATION

We, Praveen Kumar D, Sham Prasad P S, Vikas Itnal and Mahanthesh Babu H S, students of Eighth semester BE, in the Department of Computer Science and Engineering, R V College of Engineering, Bangalore declare that the project entitled **"Implementation of Image Convertor for Popular Image formats"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the academic year **2012 -2013** . The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Praveen Kumar D
1RV09CS074

Sham Prasad P S
1RV09CS098

Vikas Itnal
1RV09CS117

Mahanthesh Babu H S
1RV10CS407

ACKNOWLEDGEMENT

Our project was the result of the encouragement of many people who helped in shaping it and provided feedback, direction and valuable support. It is with hearty gratitude that we acknowledge their contributions to our project.

We would like to thank our principal **Dr. B S Satyanarayana** who has always been a great source of inspiration.

We would also like to thank our **HOD**, **Dr. N K Srinath** for his constant support extended towards us during the course of the project. His help and advice has been entirely responsible for completing the project.

We are highly indebted to our guide **Mr. Sesha Kalyur** for his continuous guidance and the help provided by him during the course of the project work.

Last, but not the least, we would like to thank our peers and friends who provided us with valuable suggestions to work with our project.

Praveen Kumar D
1RV09CS074

Sham Prasad P S
1RV09CS098

Vikas Itnal
1RV09CS117

Mahanthesh Babu H S
1RV10CS407

ABSTRACT

There are many formats to display an image. These formats will be having various capabilities, it may be in case of color representation, memory aspects, storage aspects and adherence to network protocols. Often conversion from one format to another is necessary to read, write, store or send images across networks efficiently. The aim was to build an image converter to convert an image from one format to other. The present application built, can convert images among popular formats. There are many company specific image formats as well for example Canon's .crw and .cr2 and Nikon's .nef. Instead of going in depth for company specific formats, the project is adhered to the more generalized popular formats.

There are many versions of a particular image in a particular format. One simple approach to convert an image from one format to another is, both images of different formats are to be read and finally a code to be written to convert them. Specifically, each image has header, RGB data. Header is different for different image formats. Few image formats have their data in compressed formats. That data is decompressed during reading that image and again compressed while writing it to the output file.

The project was intended to build an image format converter from GIF to RAW. TIFF/ep format is popularly known as RAW format. This application can convert an image from any popular format to other formats including GIF, TIFF/ep, JPG etc. It can handle gray scale and RGB color images as well.

Table of Contents

List of Figures

List of Snapshots

Glossary

LSB	Least significant Bit
PNG	Portable Network Graphics
BMP	Bitmap
JPG	Joint Photography Expert Group
SDLC	Software Development Life Cycle
DFD	Data Flow Diagram
UI	User Interface
GIF	Graphics Interchange Format
TIFF	Tagged Image File Format
TGA	Targa

Chapter 1

Introduction

A massive number of image file formats are available for storing graphical data, and, consequently, there are a number of issues associated with converting from one image format to another, most notably loss of image detail. Many image formats are native to one specific graphics application and are not offered as an export option in other software, due to proprietary considerations. An example of this is Adobe Photoshop's native PSD-format, which cannot be opened in less sophisticated programs for image viewing or editing, such as Microsoft Paint. Most image editing software is capable of importing and exporting in a variety of formats though, and a number of dedicated image converters exist.

Image file formats are standardized means of organizing and storing digital images. Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer[1]. An image file format may store data in uncompressed, compressed, or vector formats. Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.

1.1 Definitions

1.1.1 Different Image Formats

There are many formats of image to support different formats and to satisfy different functional requirements. Following are few popular formats of images.

- **Bit Mapped Pixel (BMP)**

The BMP file format handles graphics files within the Microsoft Windows OS[2]. Typically, BMP files are uncompressed, hence they are large; the advantage is their simplicity and wide acceptance in Windows programs.

- **Portable Pix Map (PPM)**

Netpbm format is a family including the Portable Pix Map (PPM) file format, the Portable Gray Map (PGM) file format and the Portable Bit Map (PBM) file format[1]. These are either pure ASCII files or raw binary files with an ASCII header that provide very basic functionality and serve as a lowest-common-denominator for converting pixmap, graymap, or bitmap files between different platforms. Several applications refer to them collectively as Portable Any Map format (PNM).

- **Tagged Image File Format (TIFF)**

TIFF describes image data that typically comes from scanners, frame grabbers, and paint- and photo-retouching programs. TIFF is not a printer language or page description language[3]. The purpose of TIFF is to describe and store raster image data.

A primary goal of TIFF is to provide a rich environment within which applications can exchange image data. This richness is required to take advantage of the varying capabilities of scanners and other imaging devices. Though TIFF is a rich format, it can easily be used for simple scanners and applications as well because the number of required fields is small[2]. TIFF will be enhanced on a continuing basis as new imaging needs arise. A high priority has been given to structuring TIFF so that future enhancements can be added without causing unnecessary hardship to developers.

Features:

- TIFF is capable of describing bi-level, grayscale, palette-color, and full-color image data in several color spaces.
- TIFF includes a number of compression schemes that allow developers to choose the best space or time tradeoff for their applications.
- TIFF is not tied to specific scanners, printers, or computer display hardware.
- TIFF is portable. It does not favor particular operating systems, file systems, compilers, or processors.
- TIFF is designed to be extensible—to evolve gracefully as new needs arise.
- TIFF allows the inclusion of an unlimited amount of private or special-purpose information.

- **Graphics Image Format (GIF)**

The Graphics Interchange Format defines a protocol intended for the on-line transmission and interchange of raster graphic data in a way that is independent of the hardware used in their creation or display.

The Graphics Interchange Format is defined in terms of blocks and sub-blocks which contain relevant parameters and data used in the reproduction of a graphic[4]. A GIF Data Stream is a sequence of protocol blocks and sub-blocks representing a collection of graphics. In general, the graphics in a Data Stream are assumed to be related to some degree, and to share some control information; it is recommended that encoders attempt to group together related graphics in order to minimize hardware changes during processing and to minimize control information overhead. For the same reason, unrelated graphics or graphics which require resetting hardware parameters should be encoded separately to the extent possible.

- **Portable Network Graphics (PNG)**

This document describes PNG, an extensible file format for the lossless, portable, well-compressed [5] storage of raster images. PNG provides a patent-free replacement for GIF and can also replace many common uses of TIFF. Indexed-color, grayscale, and true-color images are supported, plus an optional alpha channel. Sample depths range from 1 to 16 bits.

PNG is designed to work well in online viewing applications, such as the World Wide Web, so it is fully streamable with a progressive display option. PNG is robust, providing both full file integrity checking and simple detection of common transmission errors. Also, PNG can store gamma and chromaticity data for improved color matching on heterogeneous platforms.
The PNG format provides a portable, legally unencumbered, well- compressed, well-specified standard for lossless bitmapped image files. Although the initial motivation for developing PNG was to replace GIF, the design provides some useful new features not available in GIF, with minimal cost to developers.

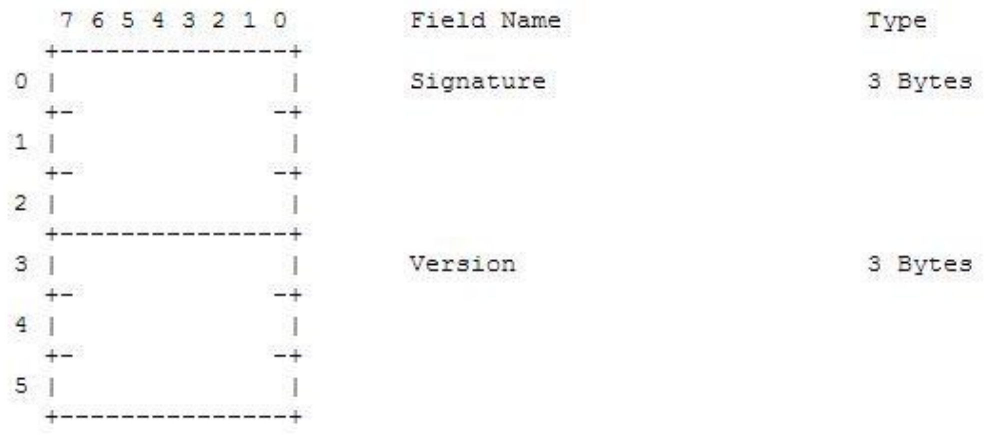
- **(Joint Photographic Experts Group) JPEG**

JPEG is a compression method; JPEG-compressed images are usually stored in the JPEG File Interchange Format (JFIF) file format. JPEG compression is lossy compression. The JPEG/JFIF filename extension is JPG or JPEG. Nearly every digital camera can save images in the JPEG/JFIF format, which supports 8-bit grayscale images and 24-bit color images (8 bits each for red, green, and blue). JPEG applies lossy compression to images, which can result in a significant reduction of the file size. The amount of compression can be specified, and the amount of compression affects the visual quality of the result. A regular image of any format contains a header, information and the RGB data. The RGB data format is in the following format.

1.2 Literature survey

1.2.1 The Graphics Interchange Format (GIF):

GIF HEADER: The Header identifies the GIF Data Stream in context. The Signature field marks the beginning of the Data Stream[1], and the Version field identifies the set of capabilities required of a decoder to fully process the Data Stream. This block is required; exactly one Header must be present per Data Stream.



```
     7 6 5 4 3 2 1 0        Field Name                  Type
    +---------------+
  0 |               |       Signature                   3 Bytes
    +-             -+
  1 |               |
    +-             -+
  2 |               |
    +---------------+
  3 |               |       Version                     3 Bytes
    +-             -+
  4 |               |
    +-             -+
  5 |               |
    +---------------+
```

Figure 1.1: GIF Image Header

The above block shows the header, this block must appear at the beginning of every Data Stream. Signature - Identifies the GIF Data Stream[9]. This field contains the fixed value 'GIF'. Version numbers can be ‘87a’ or ‘89a’.

LOGICAL SCREEN DESCRIPTOR:

The Logical Screen Descriptor contains the parameters necessary to define the area of the display device within which the images will be rendered. The coordinates in this block are given with respect to the top-left corner of the virtual screen; they do not necessarily refer to absolute coordinates on the display device. This block is required; exactly one Logical Screen Descriptor must be present per Data Stream.

```
      7 6 5 4 3 2 1 0        Field Name                    Type
     +---------------+
  0  |               |       Logical Screen Width          Unsigned
     +-             -+
  1  |               |
     +---------------+
  2  |               |       Logical Screen Height         Unsigned
     +-             -+
  3  |               |
     +---------------+
  4  | |     | |     |       <Packed Fields>               See below
     +---------------+
  5  |               |       Background Color Index        Byte
     +---------------+
  6  |               |       Pixel Aspect Ratio            Byte
     +---------------+


<Packed Fields>  =           Global Color Table Flag       1 Bit
                             Color Resolution              3 Bits
                             Sort Flag                     1 Bit
                             Size of Global Color Table    3 Bits
```

Figure 1.2: GIF Image Descriptor

This block is followed by the Global Color Table which contains the 256 required in the image. It is followed by the Image Descriptor block that contains details like height, width and aspect ratio of an image. It is then followed by the image data compressed by the LZW algorithm. At the end a trailer block that contains fixed data 0x3B must be present which indicates end of GIF data[2].

1.2.2 PNG (Portable Network Graphics):

This document describes PNG (Portable Network Graphics), an extensible file format for the lossless, portable, well-compressed storage of raster images. PNG provides a patent-free replacement for GIF and can also replace many common uses of TIFF. Indexed-color, gray scale, and true-color images are supported, plus an optional alpha channel. Sample depths range from 1 to 16 bits.

PNG is designed to work well in online viewing applications, such as the World Wide Web, so it is fully streamable with a progressive display option. PNG is robust, providing both full file integrity checking and simple detection of common transmission errors[3]. Also, PNG can store gamma and chromaticity data for improved color matching on heterogeneous platforms.

The PNG format provides a portable, legally unencumbered, well- compressed, well-specified standard for lossless bitmapped image files. Although the initial motivation for developing PNG was to replace GIF, the design provides some useful new features not available in GIF, with minimal cost to developers.
PNG is designed to be:

- Simple and portable: developers should be able to implement PNG easily.
- Legally unencumbered: to the best knowledge of the PNG authors, no algorithms under legal challenge are used. (Some considerable effort has been spent to verify this.)
- Well compressed: both indexed-color and true-color images are compressed as effectively as in any other widely used lossless format, and in most cases more effectively.

1.2.3 TIFF (Tagged Image File Format):

Image File Header:
A TIFF file begins with an 8-byte image file header, containing the following information:
Bytes 0-1:
The byte order used within the file. Legal values are:
“II” (4949.H) and “MM” (4D4D.H)
The “II” format, is little-endian byte order. The “MM” format, is big-endian byte order.

Bytes 2-3:

An arbitrary but carefully chosen number (42) that further identifies the file as a TIFF file. The byte order depends on the value of Bytes 0-1.

Bytes 4-7:

The offset (in bytes) of the first Image File Directory (IFD). The directory may be at any location in the file after the header but must begin on a word boundary. In particular, an Image File Directory may follow the image data it describes. The term byte offset is always used in this document to refer to a location with respect to the beginning of the TIFF file. The first byte of the file has an offset of 0.

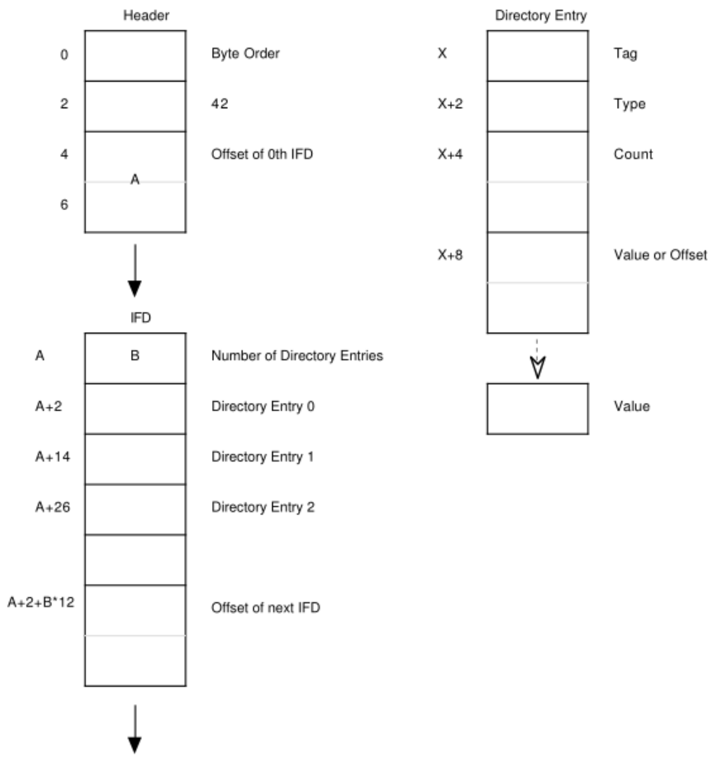


Figure 1.3: TIFF IFD format

Image File Directory:

An Image File Directory (IFD) consists of a 2-byte count of the number of directory entries (i.e., the number of fields), followed by a sequence of 12-byte field entries, followed by a 4-byte offset of the next IFD (or 0 if none).

The IFD entries are

Bytes 0-1: The Tag that identifies the field.

Bytes 2-3: The field Type (can be byte, asii, short, lon, rational).

Bytes 4-7: The number of values, Count of the indicated Type.

Bytes 8-11: The Value Offset, the file offset (in bytes) of the Value for the field. The Value is expected to begin on a word boundary; the corresponding Value Offset will thus be an even number. This file offset may point anywhere in the file, even after the image data.

1.3 Motivation

Images are used in various platforms. Different platforms support different formats. Each format has its own merits and specifications[7]. Each image format servers a different requirement under different situations. For example .JPEG is used in case of memory constrains and quality balance, BMP is used in case of memory constrains and no matter of quality. GIF, TIFF images are considered to be quality images and are used in case of image analysis, image processing[6].

An image convertor for these standard formats deals a greater concern in these cases because various programs might want to use a single image in various platforms for various purposes.

1.4 Scope

The scope of this project are :

- Successfully implementing the image conversion.
- The Conversion includes decoding an image from its current format to a user readable format.
- Read the rasterized RGB data (in case of RGB images)
- Encode it to the destined format to which the image is being converted.

1.5 Methodology

The methodology is a way to systematically solve the research problem. It may be understood as a science of studying how researches happen scientifically.

The present project proposes a cutting edge method of image conversion which is used by a vastly used genuine image convertors.

As explained earlier, a regular image has a very useful part called RGB data which accounts for the whole and sole appearance of the image. This RGB data representation is different in case different formats of the image. RGB data is nothing but the binary bit values for each pixel from the beginning that is (0, 0) to the full size of the image that is (length, breadth).

Therefore, conversion of an image from one format to another the significant change is made to this section of the image. Image conversion reduces to the mere mapping of the two RGB data representations.

Since, different formats have their own information except RGB data, we stick to one format. That is, while converting an image from format A to format B, the image is first converted to ppm format first. So conversion breaks down to two conversions. One from format A to ppm and then again from ppm to format B. ppm is yet another format which helps in reasonable conversions.

1.6 Organization of the report

The main body of the report is preceded by detailed contents and introductionincluding scope, purpose and motivation. This is followed by executive summary givingbriefly the

- Chapter 1 gives the introduction, motivation, purpose and objectives and scope of the project.
- Chapter 2 illustrates the overall description, Assumptions and dependencies, design of the Project including softwarerequirements, Functionality requirements, Performance requirements and the hardware requirements.
- Chapter 3 explains the Design considerations, Architectural design, System architecture and Data flow diagrams.
- In Chapter 4 states the detailed design, structured chart, and functional description of the modules.
- Chapter 5 gives the programming language selection, platform description, class declarations and problems encountered and strategies used to tackle them.
- Chapter 6concentrates on the testing which includes unit testing of the modules, Integration testing, Interface testing, System testing and functional testing of the GUI.
- Chapter 7 is the Conclusion having summary of the project, limitations and futureEnhancements.

The main report is followed the References which have been used to bring out the project, followed by the appendix containing the screen shots.

Chapter 2

Software Requirements Specification

Software requirement specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions of the users will with the software. Requirements analysis is critical to the success of a project. Requirements must be measurable, testable, related to identified business needs and defined to a level of detail sufficient for system design.

2.1 Overall Description

This section describes the general factors that affect the product and its requirements. It provides a background for those requirements, which are defined in detail in section 2.2 and makes them easier to understand.

2.1.1 Product Perspective

Image formats hold a key role in image viewing and analysing. So do the image convertors. This software is all about converting an image from the existing format to the format which is required. As per the formats are concerned the software covers almost all of the genuine image formats.

2.1.2 Product Functions

The following are the features that the product exhibit:

- The present image can be in any formats specified.
- Can be converted to any formats specified.
- The image can be selected from anywhere from the computer.
- The image can be saved to anywhere on the computer.

2.1.3 User Characteristics

This project is meant to be an academic project to be used for demonstrating a new and easier way to convert images among standard formats.

The GUI has been designed in such a way that any person can use this software for converting images very comfortably. All he has to do is, just to use the browse button to select an image from anywhere on the computer and select a particular image of some format. Mention the format of the image to which he wants to convert the image and say convert.

2.1.4 Constraints

During the development of this product, constraints were encountered under which the product was developed. Some specific constraints, under which Image conversion takes place, are:

- Today’s world contains way too many formats of images say hundreds of them.
- The image convertor allows you to convert from only popular formats.
- The UI used is programmed in open-source application called qt using C++.
- The product runs on a standalone system since objective was to build an application of image conversion.

2.1.5 Assumptions and Dependencies

- The Operating System is any one of Linux family.
- Image viewer is present in the system like picasa in Windows or IF Genome in Ubuntu.
- The formats of the images are known to the user.
- The system should contain the considerable space to hold the images.

2.2 Specific Requirements

This section shows the functional requirements that are to be satisfied by the system. All the requirements exposed here are essential to run this software successfully.

2.2.1 Functionality Requirement:

The functionality requirements for a system describe the functionality or services that the system is expected to provide. This depends on type of software system, which is being developed.

The requirements that are needed to be satisfied in the project are:

- There must be an option to choose a file using a browse button from the UI.
- Select the location where the user wants to save the converted image.
- Almost all the standard formats should be covered.
- The images after conversion should be able to open using normal image viewers.

2.2.2 Performance Requirement:

Performance of this software depends on many things from software behavior of the product to the hardware specifications but it largely depends on,

- Size of the image being converted
- The present format of the image
- The format of which it is being converted

2.2.3 Supportability

- All types of image file formats are supported.
- The converting image is of the specified format.
- The conversion can also be lossy.

2.2.4 Software Requirement:

- Operating System: Ubuntu 10+.
- Qt creator
- Image Viewer tools like Picasa in windows and IF-Genome in Ubuntu

2.2.5 Hardware Requirement:

- Intel Pentium Processor with a minimum 2.0 GHz speed.
- Minimum 2 GB RAM
- Minimum 500 GB hard disk space.

2.2.6 Design Constraints

- Although Qt and C++ can be run on any platforms, the software makes use of few UNIX system commands and API's so it does not run properly on other platforms.
- The image size is a major constraint. The smaller the size of the image the quicker the conversion.
- The conversion is lossy hence its always evident that the quality of the image degrades while converting from format of higher quality to the format of lower quality.

2.2.7 Interfaces

Interfaces are required for the easy access of underlying system. This project consists of user interface for all operations.

User Interfaces

- Login Screen: This is the first screen after login page which shows about the username and password. If correct password is entered user can proceed.
- Server connection Screen: It is the screen in which you connect to the server by entering server's IPv4 address.
- Main Screen: This screen has all the options to select an image, to choose the format to be converted.

- Choose image option: Here user can select the image from a location anywhere on the system.
- Select destination: Here user can select the destined location where he wants save the converted image.
- Error Messages: Proper error messages will appear when any error is encountered.

Hardware Interfaces

- Apart from the recommended hardware configuration as described in section 2.2.5 no other specific hardware Interfaces are required to run the software.

Software Interfaces

- C++ compiler must be present on the computer for compiling the source code.

Chapter 3

High Level Design

The software development usually follows what is known as the Software Development Life Cycle (SDLC). The second stage of SDLC is the design stage. The objective of this stage is to produce overall design of the software. This design stage involves two sub stages namely.

- High level design.
- Detailed level design.

In the high level design, the technical architect of the project will study the proposed applications which can handle those needs. A high level design discusses an overall view of how something should work and the top-level components that will comprise the proposed solution.

The detailed design includes the explanations for all the modules. It throws light on purpose, functionality, input and output.

3.1 Design Considerations

There are several design consideration issues that are needed to be addressed or resolved before getting down designing a complete solution for the system. All the assumptions and dependencies of the software, any global limitations or constraints that have significant impact on software are described in the software requirement.

3.1.1 General Constraints

These are certain constraints that have impact on the design of the System’s software:

- Interoperability: Although C built software is a portable software which can be used in other platform, there are UNIX specific codes, API’s which make only basic functions accessible in other platform.

- Although the project covers almost all the standard formats of the images but it fails to access the company specific encoded images and works for only genuine formats.

3.1.2 Development Methods

The Design method employed has been highlighted in this section:

The Data Flow Diagram (DFD) is used in current scenario for developing the software on Image Convertor in C.A data flow model is classified as the behavioral model. It describes how data flows through a procedure[10]. It also high lights what data a procedure takes as input and outputs to the next module. Moreover, the data flow diagrams are simple and easy to understand. In addition to that DFD describes the end to end processing of system helpful in testing of the software easily. The section 3.4 describes the data flow diagrams. This is used for the bottom–up approach.

For the development and overall idea of the project, structure charts are described in chapter 4 to show the overall description of the program from a top-down approach. The structure chart explains the structure of the program, describing where the each module plays it role to establish the complete system.

3.2 Architectural Strategies

These are design decisions and or strategies that affect the overall organization of the system and its higher level structures. These stratergies will provide an insight into mechanisms used in system[3]. The bottom–up approach is used in case of developing software, where individual units are built and integrated to build the final system.

3.2.1 Use of Programming Language

The designing of the system required various processes hence the procedure oriented approach was the ideal approach as per the project requirements. Accessing the images from the core level i.e. bit by bit is quite easy if it had been used in C. Hence C language is used for implementing convertors. Qt is used for UI which supports C/C++ approch since C language is used for back-end.

3.2.2 Future Plans

Although this project has been implemented by considering all the needs and requirements for a robust image convertor system, a few improvements can be carried out in the future viz,

- The software can be put up online and it can be made a web app from the deskop app.
- More number of image formats can be kept as options for the conversion.
- The project can be extended to handle audio/video conversions as well.
- Videos of the demo can be added to show how to use the application.

3.2.3 User Interface Paradigm

There is a requirement for the user interface to be easy to use and simple to understand. Most of the softwares go unused if the interface is not proper and complex. Therefore, the option was to go with Graphical User Interface(GUI) design. The core part is done in C++ using and UI is done in Qt which supports C++.

3.2.4 Error Detection and Recovery

Error detection and recovery is an important part of a reliable software. In this project error detection and recovery is implemented in a simple manner.

- Login screen provides access to only authorised users whose username and password match else an error message saying invalid password is shown.
- If the user tries to select an image of unknown format i.e. the format which was not specified then the system throws an error saying unknown format.

3.2.5 Communication Mechanism

This project works as a client-server system. Here the client sends the image, the format to be converted to the server. The server on the other hand receives the image, converts it to the required format and sends it back to the client.

3.3 System Architecture

This section provides an overview of how the functionality and the working of image format identifying and conversion algorithm. The overall functionalities of the whole system are divided among different modules in an efficient way.

The system architecture is shown in Fig 3.1. Here first user selects an image format to be converted and the option whether it is to be a gray scale or not. The application works in this way, the application accepts the input image, output file name and gray scale value (0 or 1) as the input values. It identifies the format of the image to be converted depending on the extensions of the images[11]. It calls the appropriate conversion functions depending on the format pair.

The conversion function reads the header which contain signature of the file, type of file and its extensions, version, row, columns, depth and mode. It also reads the tags (if any), decompresses RGB data. It creates a new file for the new converted image depending on the location chosen in the GUI. The GUI of the application contains an option to select the destination location to save the converted image.

It writes the header, writes the tag (if any) and compresses RGB data and writes to the output file.

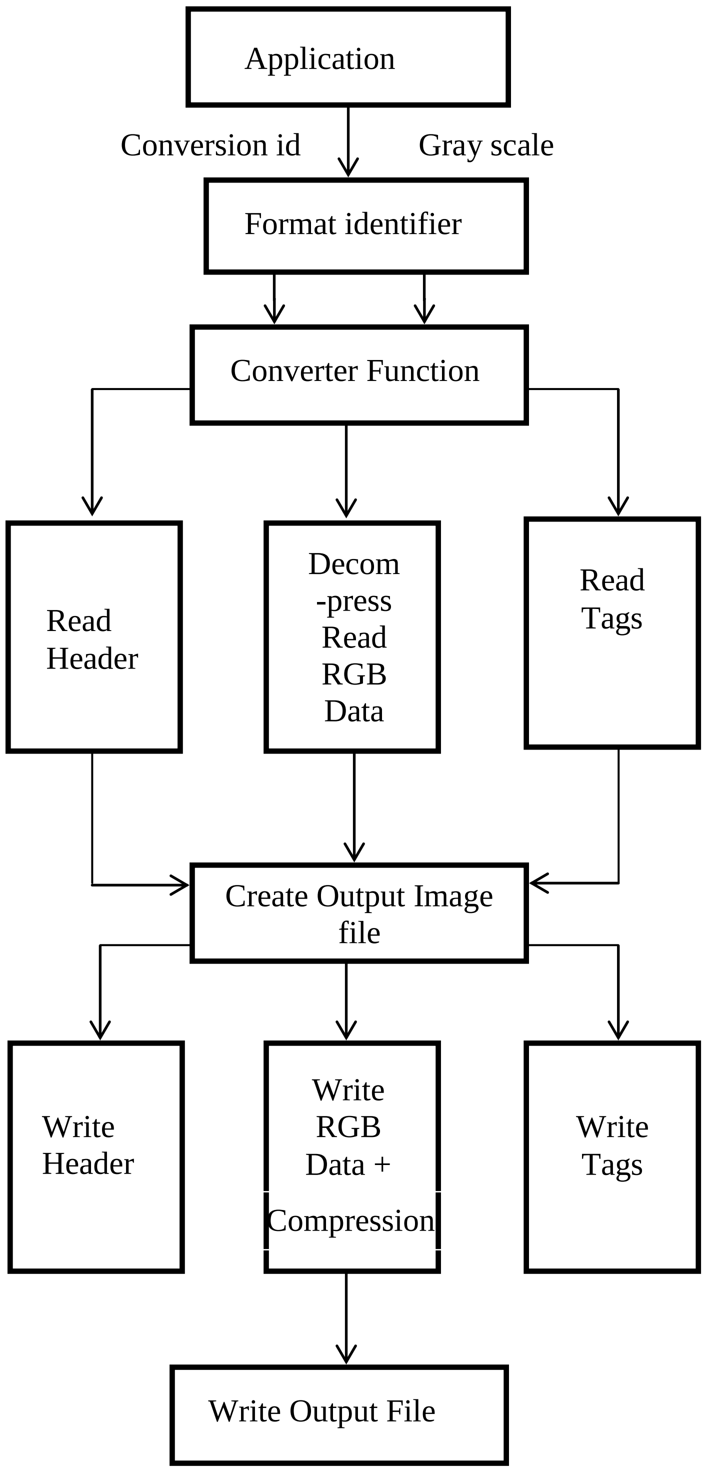


Figure 3.1 System Architecture of image converter

3.4 Data flow diagram(DFD)

A DFD is nothing but a figure which shows the flow of data between the different processes and how the data is modified in each of the process. It is a very important tool in software engineering that is used for studying the high level design.

There are many symbols to represent the various operations in a DFD. There are large number of symbols to represent the processes, data stores, data flow directions and external entities.

There are large numbers of levels to represent the software. The level 0 gives the general description and level 1 gives a detailed description. Going higher in the level numbers greater description of the processes will be given.

3.4.1 DFD Level 0

The level 0 DFD is in Figure 3.2, it gives the general operation of the image conversion system. There are two major components. One external entity called client and one major system called the convertor server system.

- Client (GUI) : This is the UI part which is accessible to the user. The user is supposed to choose an image and the format to which he wants to convert it and send it to the server.
- Convertor server: This part of the application receives the input image, destination location and the gray scale option as the inputs and it converts the image to the required format and saves the image in the proper location.

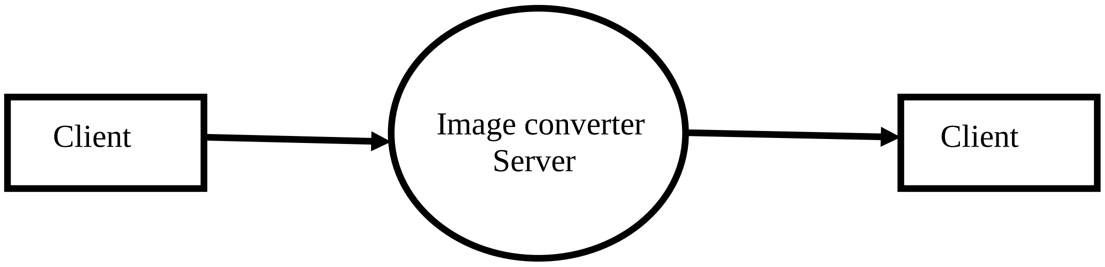


Figure 3.2 Data Flow Diagram Level 0

3.4.2 DFD Level 1

Level 1 DFD is shown in Figure 3.3. The processes involved in here are:

- Image Reader: As per the data flow perspective, Image reader basically accepts inputs, image itself which is to be converted, output image location and the gray scale option.
- Image Writer: An image writer writes the header, RGB data (or Image data) and tags (if any) to an output file.
- Intermediate format: Portable pix map (ppm) format is chosen as the intermediate format since it has a simple and general format. Hence it is easy to convert an image from any format to ppm or from ppm to any other format.

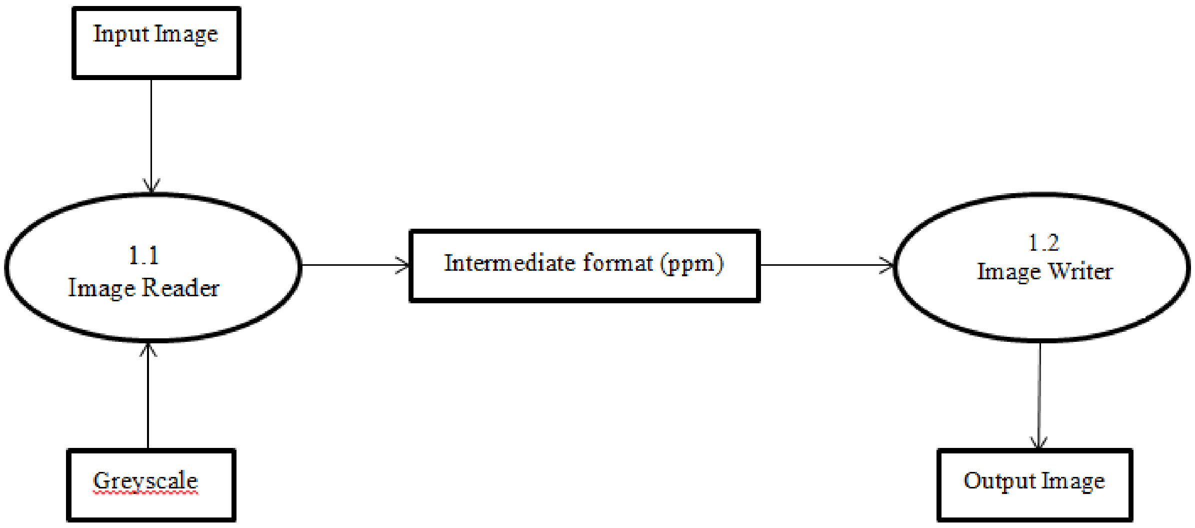


Figure 3.3 Data Flow Diagram Level 1

3.4.3 DFD Level 2

There are three major processes in level 2 DFD (figure 3.4(a)) of Image convertor server. Here are they,

- Image compressor: Since some image conversions are lossless and others are lossy, a part of the application is needed to compress and decompress an image.
- Many images have tags as the part of the image to describe its format. So there must be some mechanism to manage tags. There is a tags encoder to handle it.
- Processor: It is the final processing unit in which the image format is completely converted with all the compression, decompression and tags done.

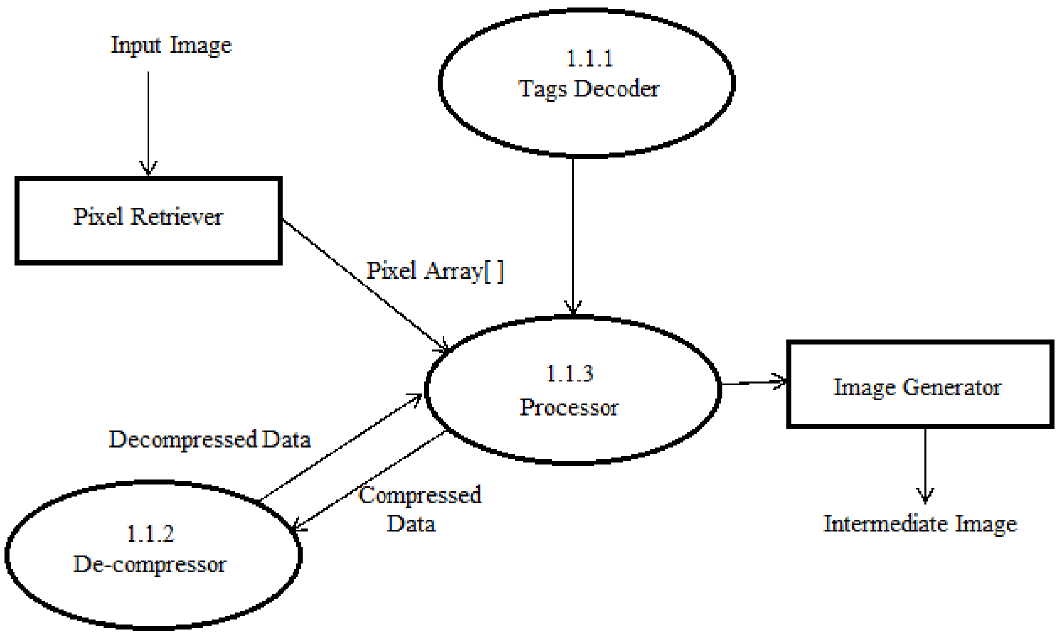


Figure 3.4(a) Data Flow Diagram Level 2: Reading Image

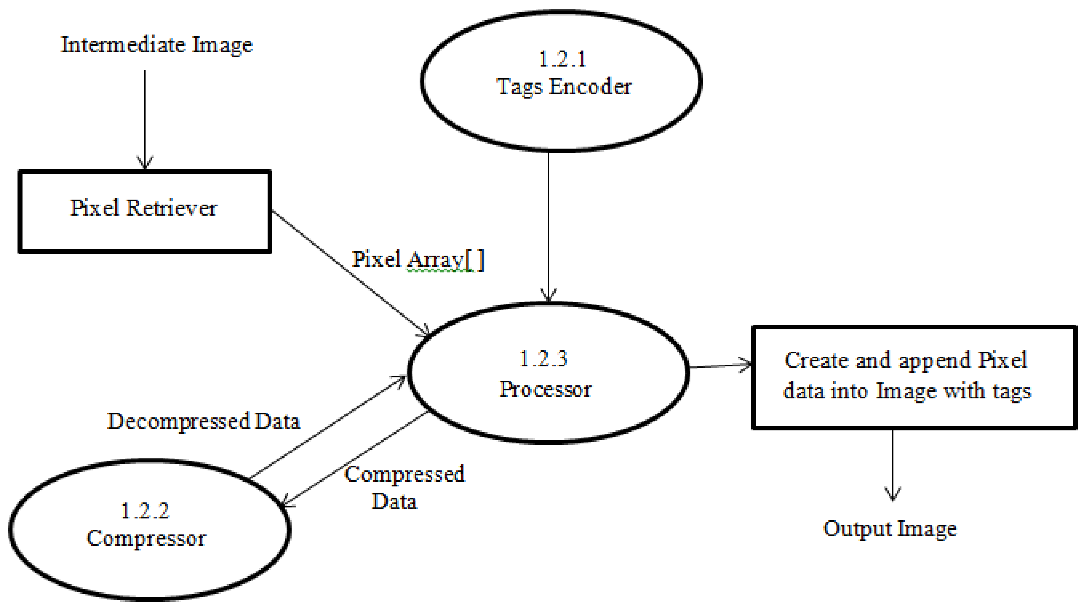


Figure 3.4(b) Data Flow Diagram Level 2: Writing Image

Since this particular image convertor works in two phases, figure 3.4(b) shows the second phase of the image converter server.

- The present format is converted to an intermediate PPM format then again PPM to the destined format.
- both the phases include the same mechanism hence only one explanation is provided and only difference between the two phases is , first one involves de-compression and second one involves compression.

Chapter 4
Detailed Design

4.1 Structured Chart

Structure charts are used to specify the high-level design, or architecture, of a computer program. As a design tool, they aid the programmer in dividing and conquering a large software problem, i.e., recursively breaking a problem down into parts that are small enough to be understood by a human brain. The process is called top-down design or functional decomposition [2].

The Structure chart of System is as shown in Fig 4.1.The structure starts with user entering his choice of input as to either embed or extractor exit. Based on this the GUI display module responds with success or failure.

The Image and File chosen by user is taken as input. The input is consist of two forms input file name and extension format, the image is given to Conversion id generator function (strcmp function), which returns the conversion id value. For ex. Jpg to tiff con_id : 51.

The file chosen is given to PROCESS A in flow chart. Next, After successful completion of the PROCESS A, Check the output file format if the format is PPM then its complete the conversion, if the output file extension not in PPM format, convert the PPM file to output file format specified during the input, Conversion function present in PROCESS B in flow chart.

The working procedure of the PROCESS A as follows, Open the input file in read mode and output or intermediate file in write mode, Read the header of the input file, header of the file consist of the signature of the file, type of file and its extensions, version, row, columns, depth and mode. Based on the header information selects the appropriate de compression algorithm and after de-compressing the input file and get the un-compressed RGB value.

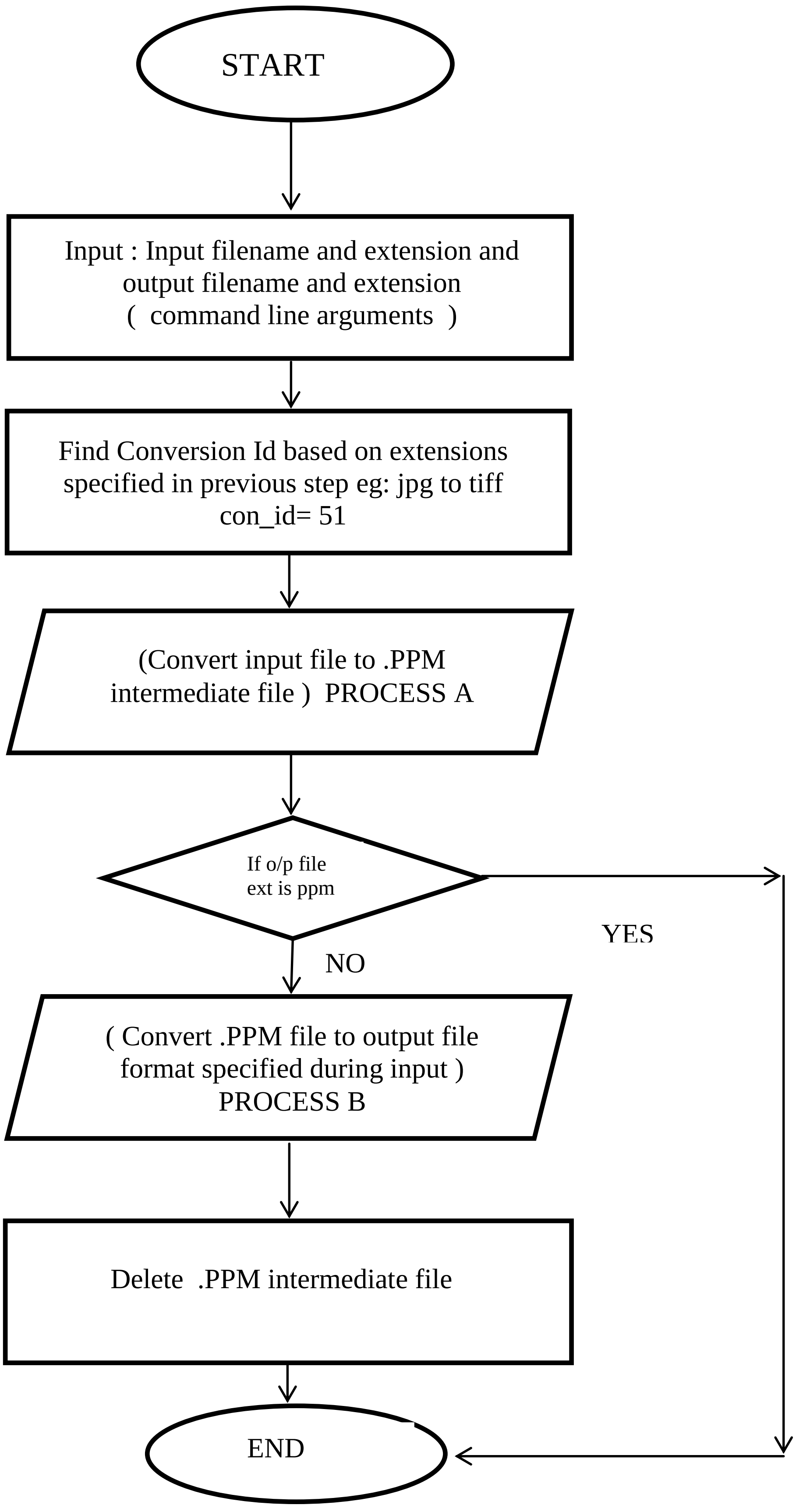

START
Input : Input filename and extension and output filename and extension (command line arguments)
Find Conversion Id based on extensions specified in previous step eg: jpg to tiff con_id= 51
(Convert input file to .PPM intermediate file) PROCESS A
If o/p file ext is ppm
YES
NO
(Convert .PPM file to output file format specified during input) PROCESS B
Delete .PPM intermediate file
END

Figure 4.1 Image convertor Flow chart

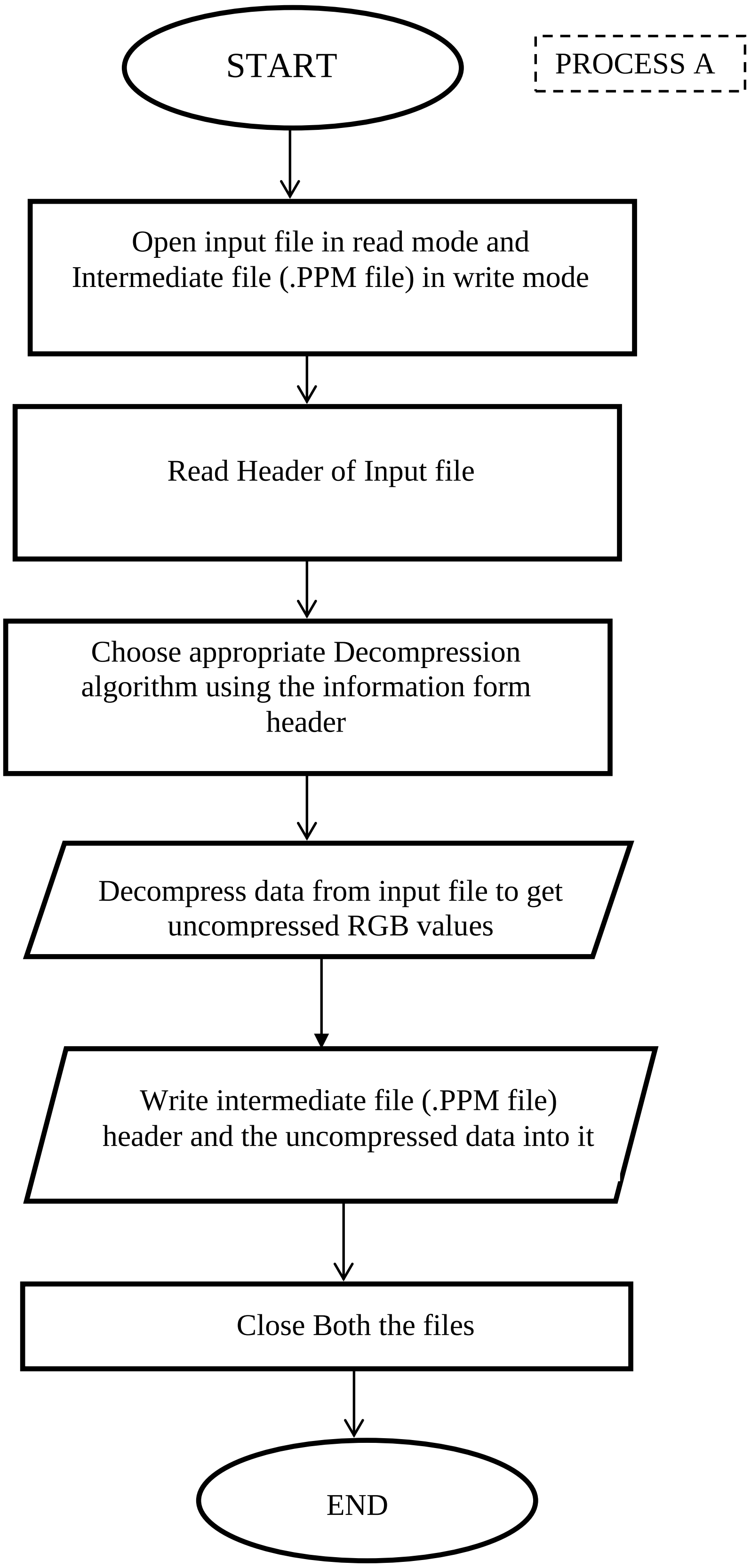

START
PROCESS A
Open input file in read mode and Intermediate file (.PPM file) in write mode
Read Header of Input file
Choose appropriate Decompression algorithm using the information form header
Decompress data from input file to get uncompressed RGB values
Write intermediate file (.PPM file) header and the uncompressed data into it
Close Both the files
END

Figure 4.2 Image convertor process A

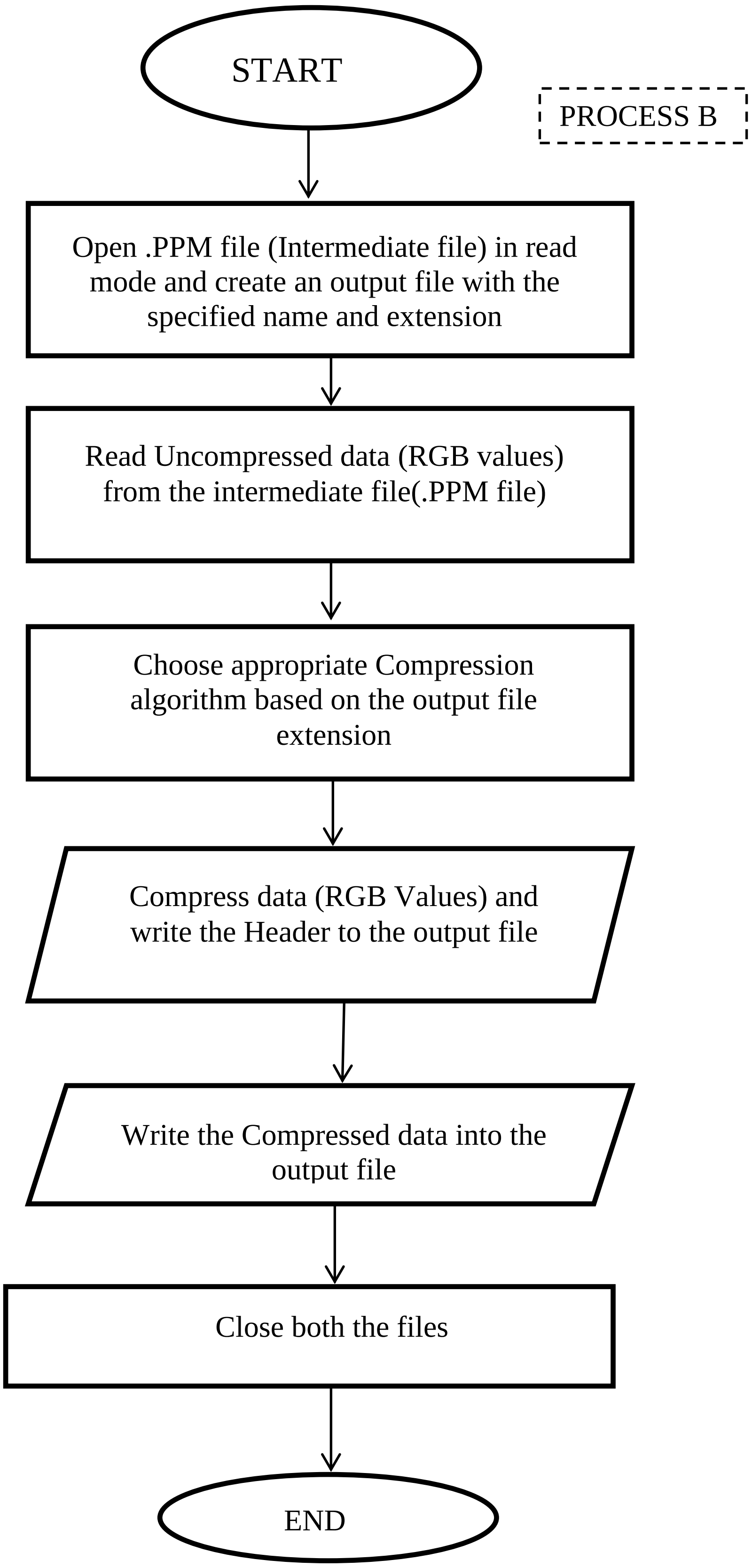

START
PROCESS B
Open .PPM file (Intermediate file) in read mode and create an output file with the specified name and extension
Read Uncompressed data (RGB values) from the intermediate file(.PPM file)
Choose appropriate Compression algorithm based on the output file extension
Compress data (RGB Values) and write the Header to the output file
Write the Compressed data into the output file
Close both the files
END

Figure 4.3 Image convertor Process B

After de-compressing, write the intermediate file header and the un-compressed data into the destination file and close the both the files.

The working procedure of the PROCESS B as follows, open the input file in read mode and create a new file with specified name and extension given by the user, before create the new file check for memory usage, if the buffer memory is less, free the un used memory space, it helps to avoid the failure of the conversion.

Read the input file, input file is un compressed format (RGB values), based on the destination file format and output file extension select the appropriate compression algorithm, after compression process completes successfully compress data, tag and header is write into the output file, check the written data is correct or not and close the both the files.

Working of the compression algorithm as follows, in this application they are two types of the compression algorithm is used

- Lossless compression
- Lossy compression

Some type of the image compression technique we use the lossless compression for example: The conversion of GIF to other format and other types of conversion uses lossy compression for example: other format to GIF and JPEG, in this application LZW[24], Huffman coding and arithmetic coding compression algorithm is used.

LZW ENCODING: If the message to be encoded consists of only one character, LZW outputs the code for this character[24]; otherwise it inserts two or multi-character, overlapping, distinct patterns of the message to be encoded in a Dictionary.

4.2 Algorithm

```
Prefix ← first input character;
CodeWord ← 256;
while(not end of character stream){
Char ← next input character;
if(Prefix + Char exists in the Dictionary)
Prefix ← Prefix + Char;
else{
Output: the code for Prefix;
insertInDictionary( (CodeWord , Prefix + Char) ) ;
CodeWord++;
Prefix ← Char;  }
}
Output: the code for Prefix;
```

LZW Decoding Algorithm: The LZW de-compressor creates the same string table during decompression. Initialize Dictionary with 256 ASCII codes and corresponding single character strings as their translations;

```
PreviousCodeWord ← first input code;
Output:string(PreviousCodeWord) ;
Char ← character(first input code);
CodeWord ← 256;
while(not end of code stream){
CurrentCodeWord ← next input code;
if(CurrentCodeWord exists in the Dictionary)
String ← string(CurrentCodeWord) ;
else
String ← string(PreviousCodeWord) + Char ;
Output: String;
Char ← first character of String ;
insertInDictionary((CodeWord,string(PreviousCodeWord)+Char));
PreviousCodeWord ← CurrentCodeWord ;
CodeWord++;}
```

Structures used to read PPM file:

```
typedef struct {
unsigned char red, green, blue;
} PPMPixel;
```

This structure is used to read an individual pixel of a .PPM file. It consists of a byte of red, green and blue components making it a 24- bit per pixel RGB image. These RGB spaces is replaced by a single gray field for gray scale images.

```
typedef struct
{
int x, y;
PPMPixel * data;
} PPMImage;
```

This structure stores image data as two dimensional arrays with width as x and height as y. It contains a pointer to the PPMpixel structure.

Structures used to read GIF file:

```
typedef struct gifImageStruct {
      unsigned char ** pixels;
      int sx;
      int sy;
      int colorsTotal;
      int red[gdMaxColors];
      int green[gdMaxColors];
      int blue[gdMaxColors];
      int open[gdMaxColors];
      int transparent;
      int *polyInts;
      int polyAllocated;
      int *style;
      int interlace;
} gifImage;
```

This structure above reads the global color table in the GIF image. It also reads the image data, height, width.

Chapter 5

Implementation

An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system. The low level design details are converted into a language specific program such that they satisfy the requirements of the given software. The technique used for implementing the software must support reusability, ease of maintenance and should be well documented. The choice for implementation also plays a very important role. The implementation phase should follow a well structured approach which is based on popular software engineering practices.

The implementation phase of any project development is its most important phase as it yields the final solution, which solves the problem at hand. The implementation phase involves the actual materialization of the ideas, which are expressed in the analysis document and developed in the design phase. Implementation should be a perfect mapping of the design document in a suitable programming language in order to achieve the necessary final product. Often the product is ruined due to incorrect language choice for implementation or an unsuitable method for programming. The factors concerning the programming language and platform chosen are described in the next couple of sections.

5.1 Platform

The programming language chosen should reflect the necessities of the project to be completed expressed in terms of the analysis and design documents. Some of the features that are required are:

- Image processing is an important necessity in the project as preprocessing module heavily requires reading and writing input images.
- Main requirement is to read the image data and manipulate the read data. For this a temporary large buffer space is needed.

- Considering the present emphasis on graphical user interface, QT packages that have tools to create pleasant looking and simple to use user interface. Linking the front end with the backend code should be also easy.
- Other than the above specific requirements few general requirements like that the language should be simple to learn and code, should provide debugging features etc must also make a contribution towards deciding the language of implementation.

All these requirements supports by combination of C++ in QT IDE for GUI development and C language for building a project.

QT key features:

- GUI and quick-start wizard for creating and simulating Support Vectors,
- Comprehensive set of learning functions.
- Modular network representation enabling easier implementation.
- Visualization functions for viewing performance.
- Pre and post processing features for improving performance.
- Easier extensibility and maintenance.

5.2 Challenges Encountered

Deciding on the coding language: Most of the object oriented languages like java already have inbuilt functions to read images of various formats. Hence not much of work is involved to convert image from one format to other. Therefore this application uses C language to read, convert and write images.

Avoiding piracy of the application: Many applications developed are affected by piracy. To avoid this project implements a client-server model where only the user with valid credentials can login to the server and avail the image conversion service. Appropriate messages are displayed if unauthorized person tries to login.

5.3 Coding Standards

This project uses C style of coding standards, It consist of following things

a Comments: The comments should describe what is happening, how it is being done, what parameters mean, which global are used and which are modified, and any restrictions or bugs.

b Declarations: Global declarations should begin in column 1. All external data declaration should be preceded by the extern keyword. If an external variable is an array that is defined with an explicit size.

c Function Declarations: Each function should be preceded by a block comment prologue that gives a short description of what the function does and (if not clear) how to use it. Source files avoid duplicating information clear from the code

d Formatting: Source files use the ANSI C type of formatting.

e Prototypes: Function prototypes should be used to make code more robust and to make it run faster.

File organization: The source files are placed under their respective modules.

1 Each module consists of headers (.h), source files (.c) and few object files (.o).

2 The main application executable (bildWandler) is placed in home folder.

3 The executable file (sever) is placed in home folder.

File Names:

The editor may not have enough temp space to edit the file, compilations will go more slowly, etc. Many rows of asterisks, for example, present little information compared to the time it takes to scroll past, and are discouraged. Lines longer than 79 columns are not handled well by all terminals and these things avoided in our project.

File names are made up of a base name and an optional period and suffix. The first character of the name should be a letter and all characters (except the period) should be lower-case letters and numbers. The base name should be eight or fewer characters and the suffix should be three or fewer characters, these rules apply to both program files and default files used in our project.

i C source file names must end in "**.c**"

ii Include header file names end in "**.h**"

Modules used For Various Image Formats:

- BMP

  BMP module contains a library to read and write bmp files. Functions bmp_read(),bmp_read_header() and bmp_read_data() opens a .bmp file, reads the header, and reads the data. Functions bmp_write(), bmp_write_data(), write data to a .bmp file.
- TIFF

  TIFF module contains libraries to read and write TIFF files. Tiff_io.c contains functions such as read_tags() to read the tiff tags and read_data() to extract data from tiff files. Functions like make_tag(), and write_data() write the appropriate tags and data to tiff files.
- PPM

  PPM module contains functions such as ReadPPM() and WritePPM() to read and write ppm files. Also function converttobw() is used to convert a color image to grayscale.
- JPEG

  This module contains executables cjpeg and djpeg to convert jpegs to ppm and vice-versa.
- TGA

  Targa module contains tga_read(), to read tga header and pixel data and function tga_write() to write tga header and pixel data.
- GIF

  GIF module contains functions such as GdImageCreate() to create a gif file. Functions like ColorAllocate() to create a color table and allocate a color. Functions such as GdImageGif() writes the gif header, image data and saves the file in GIF format.

These modules developed are integrated in one file bildWandler.c. This main file calls the above functions to read the input image, read the pixel data and convert the pixel data to required output format and finally write the output file.

The image conversion server calls the function in bildWandler to convert the file, upon receiving the request from the client and send the converted image. Server also handles the user login details, like appropriate error messages when invalid user tries to login.

Networking socket API's are used by the client and server to send and receive images. The client is a Graphical User Interface application that gets login details from user and messages the server. It also sends the selected image by the user and also receives the image upon completion of conversion by the server.

Chapter 6

Software Testing

Testing is a critical element which assures quality and effectiveness of the proposed system in (satisfying) meeting its objectives. Testing is done at various stages in the System designing and implementation process with an objective of developing a transparent, flexible and secured system.

6.1 Testing Objectives

- Testing is a process of executing a program with the intent of finding an error.
- A good case is one that has a high probability of finding an undiscovered error.
- A successful test is one that uncovers a yet undiscovered error. If testing is conducted successfully (according to the objectives) it will uncover errors in the software. Testing can't show the absences of defects are present. It can only show that software defects are present.

6.2. Testing Principles

Before applying methods to design effective test cases, a software engineer must understand the basic principles that guide software testing. All tests should be traceable to customer requirements.

6.3. Testing Design

Software developers can't test everything, but they can use combinatorial test design to identify the minimum number of tests needed to get the coverage they want. Combinatorial test design enables users to get greater test coverage with fewer tests.

6.3.1. White box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

6.3.2. Black box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

6.4. Testing Strategies

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be defined for software engineering process.

6.4.1 Unit Testing

Unit testing verification on the smallest unit of software designs the module. The unit test is always white box oriented. The system has been tested for each and every individual unit; it has satisfied the unit testing strategy.

Table 6.1: Test case for image format verification

Sl No. of Test Case	Unit test case 1
Name of Test Case	Image format verification
Feature being Tested	Image format
Description	Check whether the image is in bmp format
Sample Input	BMP image (Clouds.bmp)
Expected Output	BMP image
Actual Output	BMP image
Remarks	Successful verification of BMP image

The above test case is used to test the format of image loaded by the user which is used to convert to other formats. The format identifier works for bmp format hence it works for other formats as well.

Table 6.2: Test case for gray scale image verification

Sl No. of Test Case	Unit test case 2
Name of Test Case	Gray scale image verification
Feature being Tested	Gray scale implementation
Description	Check whether the output image is converted to gray or not
Sample Input	Image (demo.gif) with gray scale option selected
Expected Output	demo.gif to be in gray
Actual Output	demo.gif (in gray scale)
Remarks	Gray scale option is working correctly

The above test case is used test the gray scale option, an image with a gray scale option selected is given as the input to the application and application converted the image gray scale successfully.

Table 6.3: Test case for intermediate image generation

Sl No. of Test Case	Unit test case 3
Name of Test Case	Intermediate image generation
Feature being Tested	Any format to PPM conversion
Description	Each format is converted to PPM first, before it is converted to the destined format.
Sample Input	Waves.jpg
Expected Output	Waves1.ppm
Actual Output	Waves1.ppm
Remarks	Waves.jpg was given as the input to convert it to Waves.ppm which is intermediate image and it is converted successfully.

The above test case is used to check whether the intermediate image generator works properly or not.

6.4.2. Integration Testing

Integration testing is a systematic technique for constructing the program structure while conducting tests to uncover errors associated with the entire application. The integrations test cases for the current project are given below.

Table 6.4: Test case for conversion from GIF to JPG

Test Case no	1
Name of Test Case	GIF to JPG
Input Image	Image1.gif
Input Image Size	1.15 Mb
Output Image	Image2.jpg
Output Image Size	553 Kb
Remarks	Success

The above integration test case is used to check for the proper working of the entire image converter application. This module takes Image1.gif as the input and produces Image2.jpg as output. GIF to JPG format conversion is successful.

Table 6.5: Test Case for Conversion from JPG to TIFF

Test Case no	2
Name of Test Case	JPG to TIFF
Input Image	Image1.jpg
Input Image Size	152 Kb
Output Image	Image2.tiff
Output Image Size	953.5 Kb
Remarks	Success

The above integration test case is used to check for the proper working of the entire image converter application. This module takes Image1.jpg as the input and produces Image2.tiff as output. JPG to TIFF format conversion is successful.

Table 6.6: Test case for conversion from GIF to BMP

Test Case no	3
Name of Test Case	GIF to BMP
Input Image	Image1.gif
Input Image Size	412 Kb
Output Image	Image2.bmp
Output Image Size	2.3 Mb
Remarks	Success

The above integration test case is used to check for the proper working of the entire image converter application. This module takes Image1.gif as the input and produces Image2.bmp as output. GIF to BMP format conversion is successful.

Table 6.7: Test case for conversion from BMP to PPM

Test Case no	4
Name of Test Case	BMP to PPM
Input Image	Image1.bmp
Input Image Size	2.3 Mb
Output Image	Image2.ppm
Output Image Size	2.3 Mb
Remarks	Success

The above integration test case is used to check for the proper working of the entire image converter application. This module takes Image1.bmp as the input and produces Image2.ppm as output. BMP to PPM format conversion is successful.

Table 6.8: Test case for conversion from PPM to TGA

Test Case no	5
Name of Test Case	PPM to TGA
Input Image	Image1.ppm
Input Image Size	1.15 Mb
Output Image	Image2.tga
Output Image Size	1.15 Mb
Remarks	Success

The above integration test case is used to check for the proper working of the entire image converter application. This module takes Image1.ppm as the input and produces Image2.tga as output. PPM to TGA format conversion is successful.

Table 6.9: Test case for conversion from JPG to BMP

Test Case no	6
Name of Test Case	JPG to BMP
Input Image	Image1.jpg
Input Image Size	73 Kb
Output Image	Image2.bmp
Output Image Size	1.41 Mb
Remarks	Success

The above integration test case is used to check for the proper working of the entire image converter application. This module takes Image1.jpg as the input and produces Image2.bmp as output. JPG to BMP format conversion is successful.

6.4.3 Interface Testing

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been covered and corrected, interfacing is done using GUI. That GUI has to be tested. Following are the test cases of GUI.

Table 6.10: Test case for Login Form

Valid Input	Valid input credentials
Invalid Input	Invalid input credentials
Functions	Login verifications must be exercised.
Output	IP address verification page is opened on success error on failure

Login form verifies the authentication of the users. Each user is assigned username and password which are collectively known as input credentials. Whenever a user wants to access the application, he has to input those credentials in the login form. If those credentials are correct then the access is granted else access is denied.

Table 6.11: Test case for IP address entry

Valid Input	Valid IPv4 address of server
Invalid Input	Invalid IPv4 address of server
Functions	IP address verification and socket functions must be exercised.
Output	Home screen is opened on success error on failure

The application works on the client server basis. The client contains the GUI part. The server contains the conversion mechanism. The application to work properly, it is to be connected with the server. It is done by entering the server’s IP address. It is to be verified and further socket functions are to be called.

Table 6.11: Test case for Image selection

Valid Input	Images of formats used in the application such as jpeg, gif, bmp, tga, ppm, tiff, png.
Invalid Input	All other formats excepts the above formats
Functions	String matching for extensions of the image files to be exercised
Output	Proceeds further if succeeded, error on failure

The application works for only popular image formats. An error has to be thrown if the user selects the image of any other format except jpeg, gif, bmp, tga, ppm, tiff, and png.

Chapter 7

CONCLUSION

There are a lot of image formats. Some are company specific and some are not. Some formats are popular such as jpeg, gif etc. These poplar formats need to be converted from one to another according to the circumstances they are used in. For example highly compressed jpeg files are easier to transmit and uncompressed formats like bmp are easy for image processing.

The project is now able to convert between seven popular image formats. The project also handles gray scale images.

7.1 Limitations

- The application built does not support company specific image formats. It works fine for genuine formats such as jpeg,giff etc.
- Though the application is client server it is still a desktop application which runs on a standalone system. The user has to manually enter the IP address of server.
- Audio/Video formats are not supported.
- The connection between client and server is not secure.

7.2Future Enhancements

- Dynamic host discovery can be implemented to discover the image server. This does not require the user to manually enter the IP address. The application can also be made available as a web service where the user enters the URL address and uploads the image to be converted to the required format and then downloads the converted image.
- Enhancements can be done to the application so that it can handle audio/video conversions as well.
- Security features like secure socket layer(SSL) can be implemented to secure connection between client and server

References

[1] Mridu Kumar Mathur, Seema Loonker, Dr. Dheeraj Saxena, “Lossless huffman coding technique for image compression and reconstruction using binary trees”, ijcta | jan-feb 2012

[2] M. Swain and D. Ballard, “Color indexing,” International Journal of Computer Vision, vol..7, no.1, pp. 11-32, 2008.

[3] D.G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[4] D. Maheswari and V.Radha, “Secure Layer Based Compound Image Compression using XML Compression” 978-1-4244-5967-4/10 ©2010 IEEE.

[5] V. Singh, “Recent Patents on Image Compression – A Survey”, Recent Patents on Signal Processing, 2010,2, 47-62.

[6] . N. Dalal, and B. Triggs, “Histograms of oriented gradients for human detection,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 886–893, July 2008.

[7] Thrasyvoulos N. Pappasa, Jana Zujovica, David L. Neuhoffb, “Image analysis and compression: Renewed focus on texture”, Proc. SPIE Vol. 7543, (San Jose, CA), Jan. 19 - 21, 2010.

[8] D.Kesavaraja, R.Balasubramanian, D.Jeyabharathi, D.Sasireka, “Secure and Faster Clustering Environment for Advanced Image Compression”, Volume: 02, Issue: 03, Pages: 671-678 (2010).

[9] Navrisham Kaur “A Color Image Compression Using Pixel Correlation and Its Comparison with Existing Algorithms” International Journal of Computer Trends and Technology- volume4Issue2- 2013.

[10] Jagadish H. Pujar, Lohit M. Kadlaskar, “a new lossless method of image compression and decompression using huffman coding techniques”, © 2005 - 2010 JATIT. All rights reserved.

[11] Manjinder Kaur, Gaganpreet Kaur, “A Survey of Lossless and Lossy Image Compression Techniques”, International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 2, February 2013.

[12] S.Narasimhulu, Dr.T.Ramashri, “Gray-Scale Image Compression Using DWT-SPIHT Algorithm” ISSN: 2248-9622 Vol. 2, Issue 4, July-August 2012, pp.902-905.

[13] Varun Biala, Dalveer Kaur, “Lossless Compression Technique Using ILZW with DFRLC”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012.

[14] Jacob Ziv, Fellow and Abraham Lempel”,A Universal Algorithm for Sequential Data Compression”, IEEE transactions on Information theory ,May 1977

[15] Jer Min Joa and Pie Yin Chin “Fast and Efficient Lossless Data Compression Method” IEEE Transactions on Communications September 1999

[16] Paul G Hoverd and Jeffrey Scott Vitter ,“New Methods for lossless Coding using arithmetic coding”, Department of Computer Science , Brown University, August 1991

[17] James F Blinn California Institute of technology,” What’s the deal with DCT ”, IEEE Computer Graphics and Applications, July 1993

[18] David A Huffman “A Method for the Construction ofMinimum-Redundancy Codes” September 1952

[19] Gregory K Wallace, “The JPEG still picture compression standard”, DEC Maynard,Massachusetts, IEEE transactions on consumer electronics, February 1992.

[20] Ian H Witten, RM.Neal And John G Clergy ,“Arithmetic Coding for data Compression”, Communications of ACM , June 1987

[21] CompuServe Incorporated, “GRAPHICS INTERCHANGE FORMAT(sm)”,Version 89a Copyright, Columbus, Ohio,1990

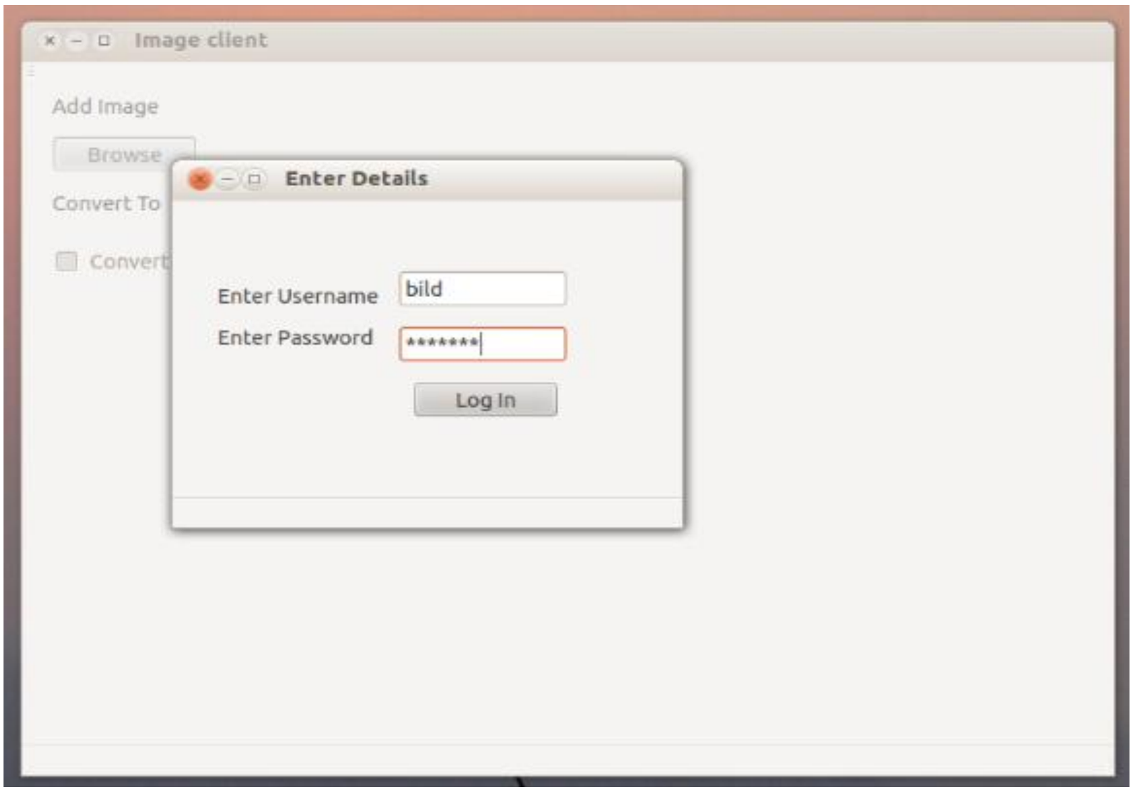
[22] Takuya Kida, Masayuki Takedaand SetsuoArikawa,”Shift-And Approach to Pattern Matchingin LZW Compressed Text“,Department of Informatics, Kyushu University 33Fukuoka 812-8581, Japan, Springer-Verlag Berlin Heidelberg 1999

[23] Adobe Developers Association,” TIFFRevision 6.0Final — June 3, 1992”

[24] Wei CuiNew LZW Data Compression Algorithm and Its FPGA Implementation (School of Information Science and Technology, Beijing Institute of Technology, Beijing) 2007.

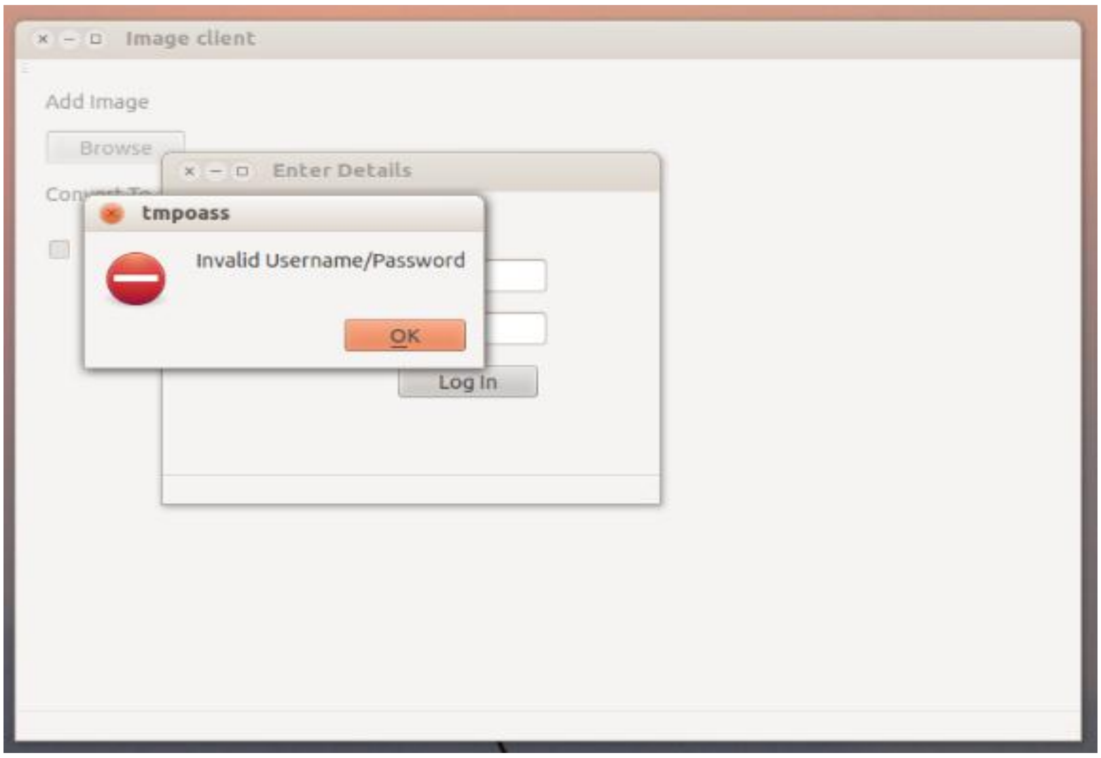
Appendix A

Snapshots



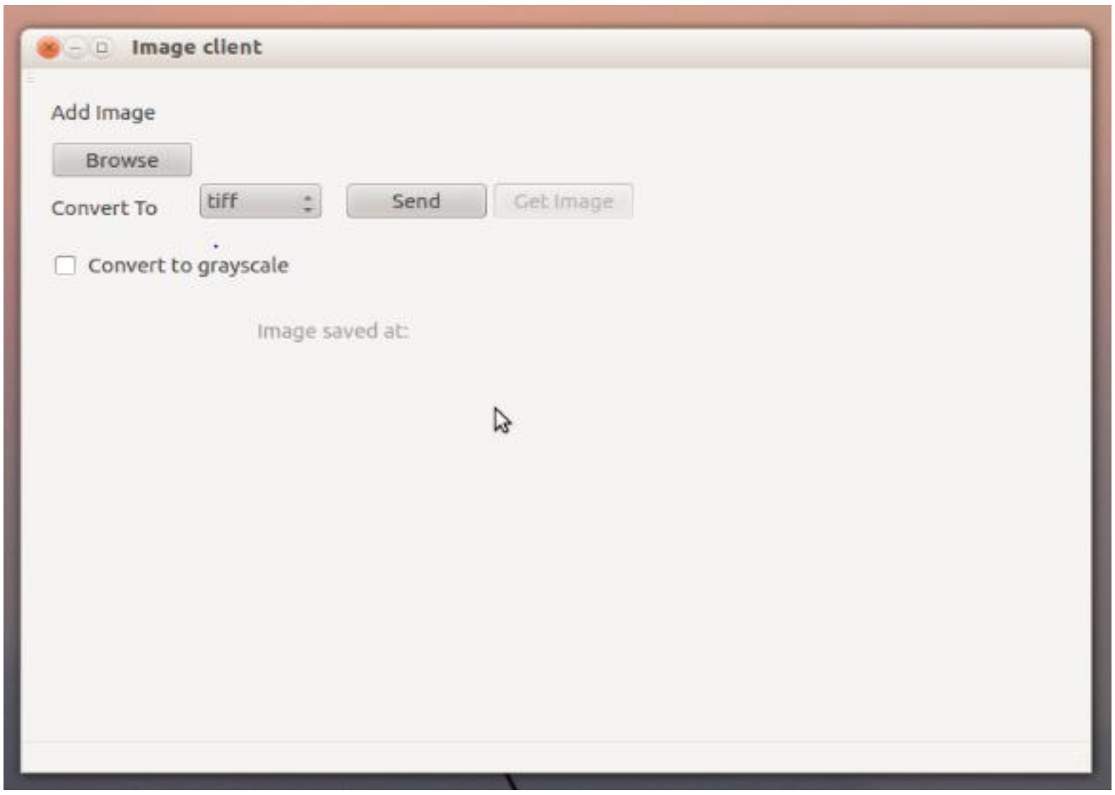
Enter Username bild
Enter Password *******

A.1 Login Form

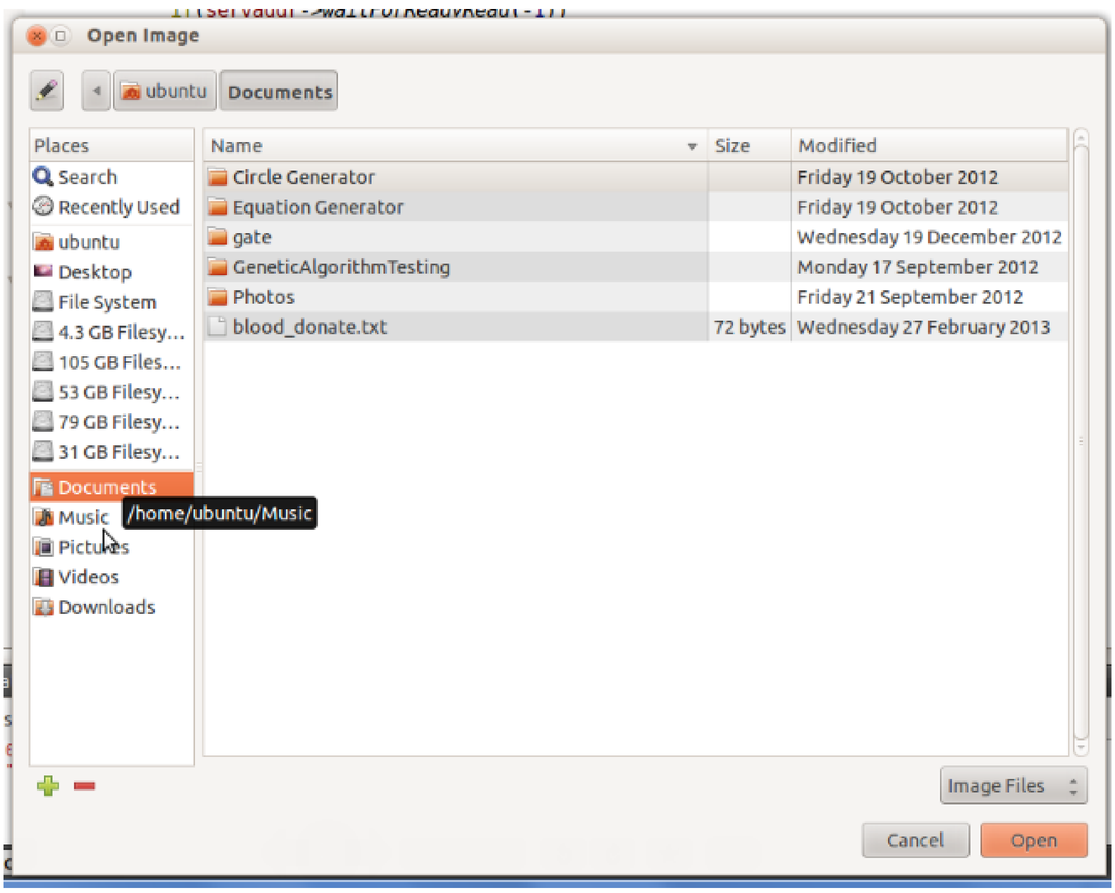


Invalid Username/Password

A.2 Invalid Login

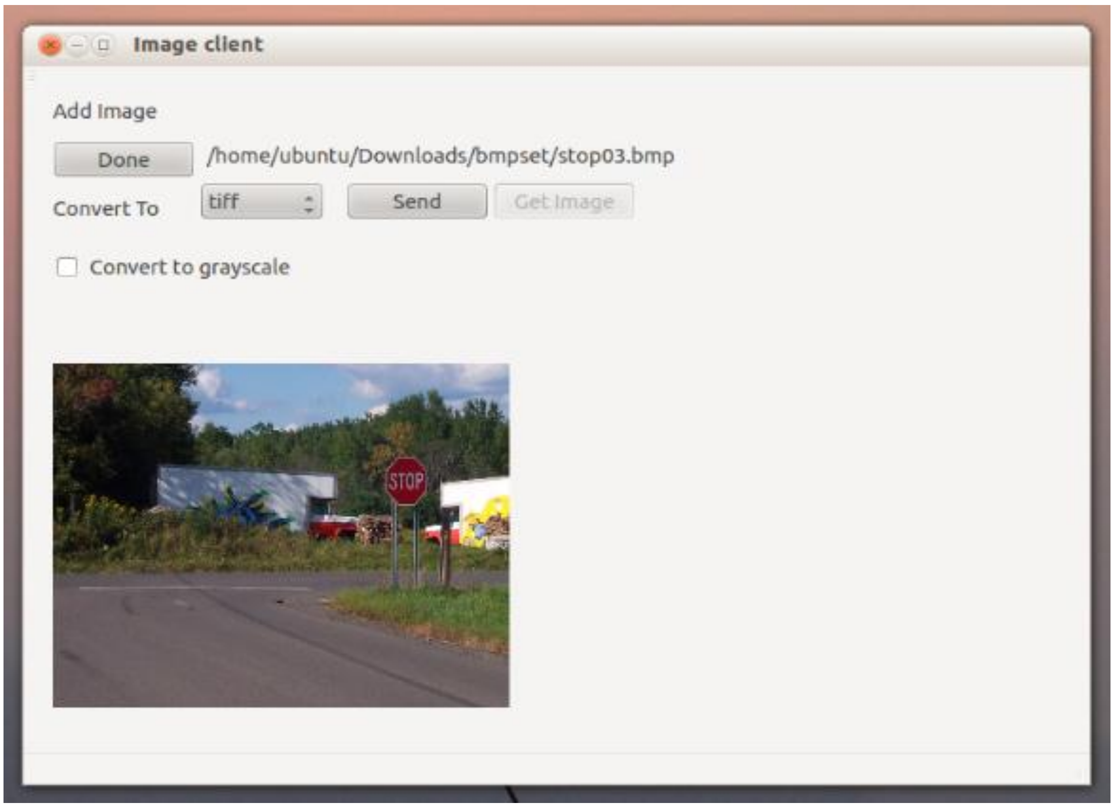


A.3 Home Screen

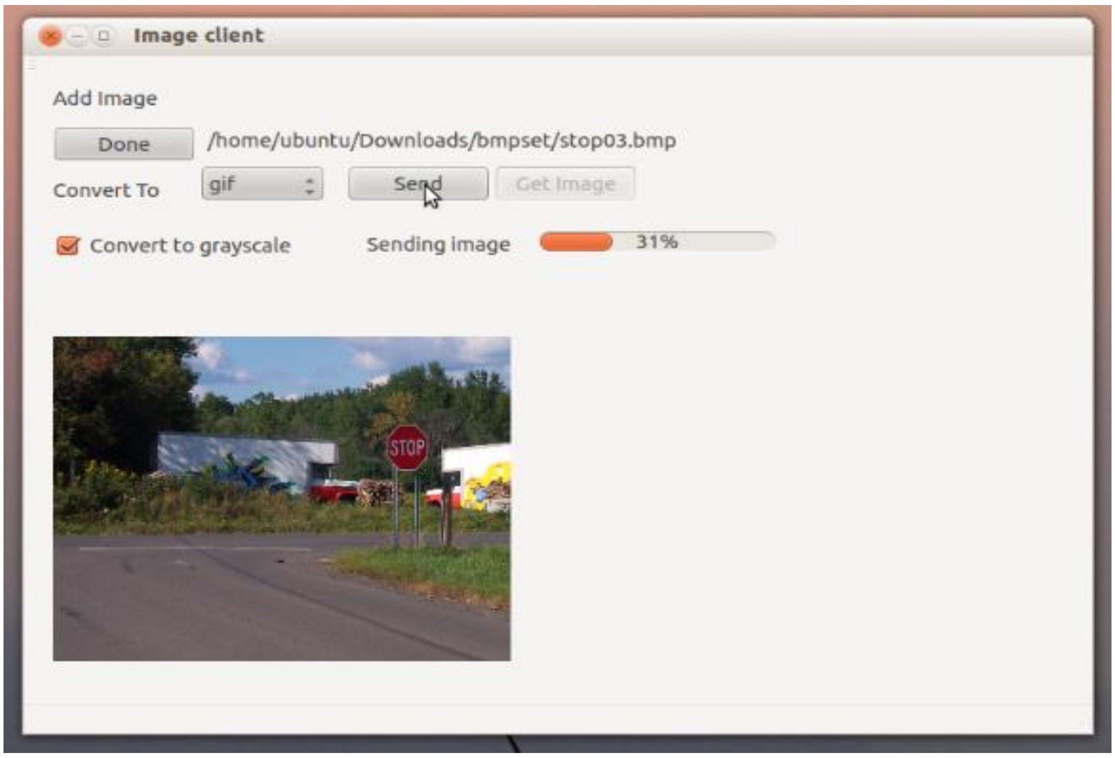


Name	Size	Modified
Circle Generator		Friday 19 October 2012
Equation Generator		Friday 19 October 2012
gate		Wednesday 19 December 2012
GeneticAlgorithmTesting		Monday 17 September 2012
Photos		Friday 21 September 2012
blood_donate.txt	72 bytes	Wednesday 27 February 2013

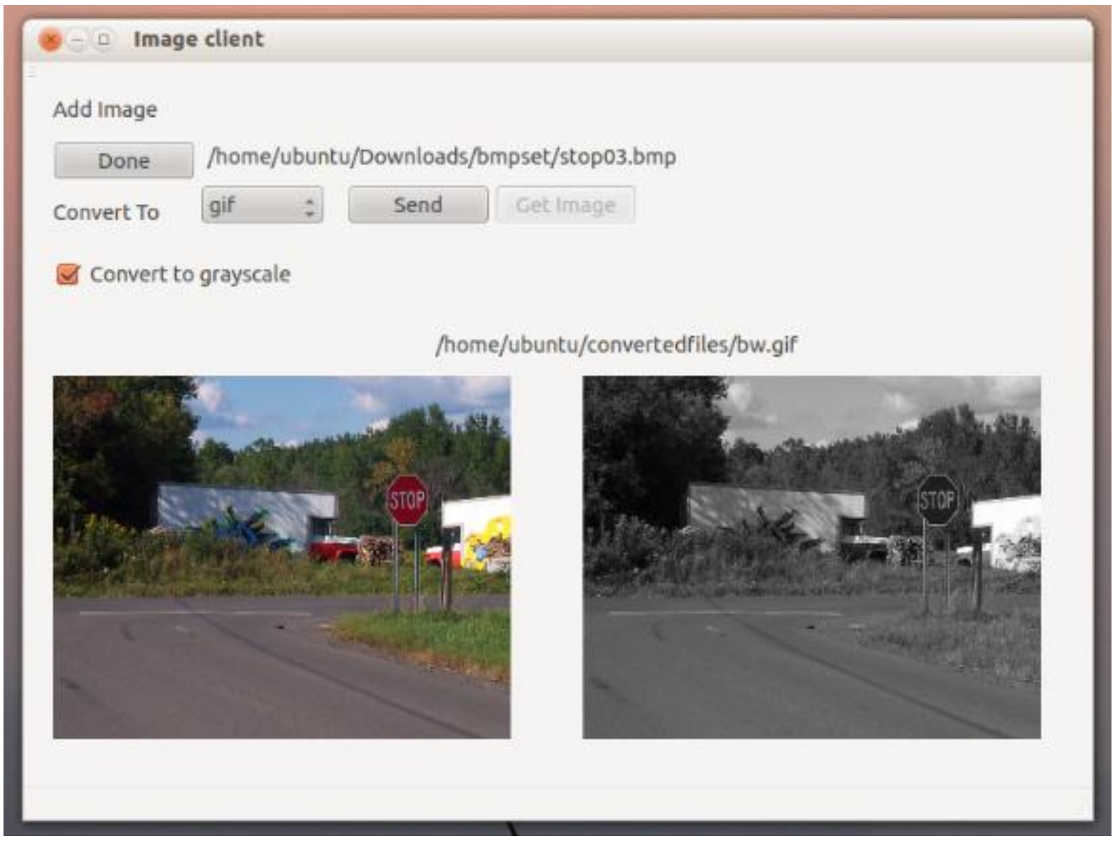
A.4 Image Selection



A.5 Chosen Image and Format



A.6 Sending Image to the Server



Add Image

Done /home/ubuntu/Downloads/bmpset/stop03.bmp

Convert To gif Send Get Image

☑ Convert to grayscale

/home/ubuntu/convertedfiles/bw.gif

A.7 Display Converted Image