NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A GRAPHIC USER INTERFACE FOR RAPID INTEGRATION OF STEGANOGRAPHY SOFTWARE

by

David Raiman Wootten

March 1996

Thesis Co-Advisors:	Cynthia E. Irvine
	Michael J. Zyda

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A GRAPHIC USER INTERFACE FOR RAPID INTEGRATION OF STEGANOGRAPHY SOFTWARE	5. FUNDING NUMBERS
6. AUTHOR(S) Wootten, David, R.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Steganography is a method an individual uses to secretly communicate, whereby the transmitting agent hides a message within some medium, so that only an intended recipient can detect the message's presence. Researchers who apply this methodology to digital imagery currently have no X Windows-based graphic user interface software package through which they may aggregate, test, and demonstrate their steganography programs, Such a package would contain features to encode data to and extract data from digital imagery, convert the files to other graphic file formats, display imagery, and offer some utility to analyze change between unencoded original images and their encoded equivalent. The steganography software development package presented in this thesis, named Steganography Toolbox, satisfies these requirements. It provides the above described features, plus the ability to delete unneeded files, all in an X Windows graphic user interface. It permits the user, who writes a separately executable steganography program, to attach it to the graphic interface with little additional programming effort. The thesis describes a method to create a menu-selected dialog box containing the necessary widgets, which invokes the desired program through a *system()* call. The thesis includes Steganography Toolbox's structured design documentation, from system requirements to process specifications. The thesis also describes how requirements-based software tests were performed on each module to verify proper function and error-handling.

14. SUBJECT TERMS Graphic user interface, steganography.	15. NUMBER OF PAGES 141
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

A GRAPHIC USER INTERFACE FOR RAPID INTEGRATION OF STEGANOGRAPHY SOFTWARE

David Raiman Wootten
Lieutenant, United States Naval Reserve
B.A., University of California, Santa Barbara, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1996

Author: [illegible]
David Raiman Wootten

Approved By: [illegible]
Cynthia E. Irvine, Thesis Co-Advisor

[illegible]
Michael J. Zyda, Thesis Co-Advisor

[illegible]
Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

Steganography is a method an individual uses to secretly communicate, whereby the transmitting agent hides a message within some medium, so that only an intended recipient can detect the message's presence. Researchers who apply this methodology to digital imagery currently have no X Windows-based graphic user interface software package through which they may aggregate, test, and demonstrate their steganography programs, Such a package would contain features to encode data to and extract data from digital imagery, convert the files to other graphic file formats, display imagery, and offer some utility to analyze change between unencoded original images and their encoded equivalent. The steganography software development package presented in this thesis, named Steganography Toolbox, satisfies these requirements. It provides the above described features, plus the ability to delete unneeded files, all in an X Windows graphic user interface. It permits the user, who writes a separately executable steganography program, to attach it to the graphic interface with little additional programming effort. The thesis describes a method to create a menu-selected dialog box containing the necessary widgets, which invokes the desired program through a *system()* call. The thesis includes Steganography Toolbox's structured design documentation, from system requirements to process specifications. The thesis also describes how requirements-based software tests were performed on each module to verify proper function and error-handling.

TABLE OF CONTENTS

I. INTRODUCTION

A. PURPOSE

The aim of this thesis is to produce a software implementation that performs steganographic encoding and decoding of digital images. A further objective is to make this software suitable for continued development of steganography software implementation and research at Naval Postgraduate School.

B. OVERVIEW OF STEGANOGRAPHY

1. Definition

Steganography is a method of concealing information within a medium; it is similar to encryption in this respect. It differs from encryption in that while the undesired viewer of a message may see an encrypted data string before him, it has been rendered meaningless by the encryption. Steganography, on the other hand, conceals the very presence of the message; if the unintended observer could see, hear or otherwise sense the message, he would readily understand it. It relies on the medium's capacity to hold the encoded message as well as enough of the original data of the medium not to invoke the viewer's suspicion the medium was changed. Stated another way, the medium must be able to hold more information than the viewer can sense. Kuhn (1995) state this in the following description: "In contrast to cryptography, where the enemy is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by a cryptosystem, the goal of steganography is to hide messages inside other harmless messages in a way that does not allow any enemy to even detect that there is a second secret message present." It is readily apparent that the strength of this method is limited in the sense that encoded data can be readily extracted if the enemy knows which media are encoded, and the method of encoding. It can be considered a useful augmentation to encryption if the party holding the message doesn't want anyone to think he or she has something to hide, by encrypting data. It is in this environment, with the recent surge in development of digital telecommunication and the resulting public concern over issues of security and privacy of data, that interest in steganography has bloomed. Digital media readily lend themselves to steganographic application because the information-carrying capacity of many digital files, especially digital image and audio files, is much greater than a human's ability to discern change in that file.

2. Historical Perspective

The term "steganography" comes from the words "steganos" (covered or secret) and "graphy" (writing or drawing). Brown (1995) states, "steganographic methods have long been a favorite of military and political leaders, who have been devising ingenious ways of hiding secret messages for thousands of years." He further describes, from the Histories of Herodotus ,a Medean nobleman who hid a message for an ally in the belly of an unskinned rabbit, to be carried by his servant, who assumed the guise of a hunter. Again, as described by Johnson (1996) regarding the same ancient document, "In ancient Greece text was written on wax-covered tablets. In one story, Demeratus wanted to notify Sparta that Xerxes intended to invade Greece. To avoid capture [of the message], he scraped the wax off the tablets and wrote a message on the underlying wood. He then covered the tablets with wax again. The tablets appeared to be blank and unused, so they passed an inspection by sentries without question.

Invisible ink was another common application of steganography in World War II. These substances appeared transparent and could be written on paper that wouldn't arouse suspicion. The ink could be rendered readable by some transformation process, usually heat or by some other chemical reaction. Johnson (1996) lists milk, vinegar, fruit juices and urine as commonly-used invisible inks. Brown (1995) describes how the Nazi spy George Dasch wrote messages on napkins using copper sulfate; the recipient could make the message readable by exposing it to ammonia fumes.

Other steganographic applications used in World War II were the "microdot" described by Brown (1995) as "a photograph the size of a printed period that, when developed, could reproduce a standard-sized typewritten page with perfect clarity." He additionally describes the United States Marines' use of Navajo Indian "codetalkers" to radio messages in their obscure and rare native tongue. Secrecy was in this case verified by identifying all the non-Navajo speakers of the language. All twenty-eight had no connection with the Germans of Japanese. Additionally, the codetalkers spoke a slang dialect that those who knew Navajo but were unfamiliar with the slang couldn't understand.

3. Recent Interest

While the recent development of digital telecommunications has drastically improved the ability to transmit information, the non-secure characteristic inherent to the medium through which the messages are sent (e.g., Twisted Pair) requires some method be applied to ensure secrecy and integrity of those messages. Public Key Encryption is a popular method to ensure these protections, as noted by the popularity of Pretty Good Privacy. Those concerned with privacy in an increasingly data-invasive world suggest that outside viewers don't even need to know that encrypted data is present. This suggests to the viewer that the holder of encrypted data "has something to hide," and there is no reason the outside viewer even needs to know this information. It it this mindset that has fostered renewed interest in steganography. Kurak and McHugh (1992) demonstrated the use of steganographic encoding of "nonclassified" imagery with supposedly classified data, showing the little apparent change in the encoded result, when directly compared to the unencoded original. Since then, resurgent interest started informally, as internet discussion groups and then as a steganography news mailing list. Such significant interest has emerged in the last year that Anderson (1996) has issued a call for papers on the first Workshop on Information Hiding. Johnson(1995) has evaluated some of the steganography shareware available on the Internet, and Aura (1995) attempts to lay a foundation of terminology for steganography.

II. BACKGROUND

A. OVERVIEW AND SCOPE

Applied steganography relates several distinct research areas; to provide a proper basis for the study of steganography, one would, for instance, necessarily also want to explore human factors: physiology and psychophysiology of human vision, and related genetic and pathological influences on visual perception. (Human factors research is a massive field in itself, so only a cursory description of pertinent topics in this area are discussed.)The primary application of this thesis relates to steganography performed on digital imagery, so some background regarding principles and techniques of image interpretation should also be included. This background is further restricted to steganography applied to an image's color. No attempt will be made to describe steganography applied to other image features, such as object size shape texture, etc. Currently implemented color metrics shall be identified, and some information regarding image files and file compression is also discussed. Finally, some currently available steganography shareware programs are presented.

B. HUMAN VISION

1. Physiology

A useful and unambiguous definition of "seeing" is provided by Cornsweet (1970): "An observer will be said to see a property of the external world if it can be demonstrated that he is assimilating the information that the property is present, through the interaction between his anatomy and the light carrying that information." Sabins (1978) states that vision is the human's most heavily relied-upon sensory system and accounts for most of the environment's input to the brain. This system can be considered to contain the eyes, optic nerves and brain, but for the purpose of this thesis structure of the eye alone is discussed. Figure 2-1 shows the structure of the human eye.

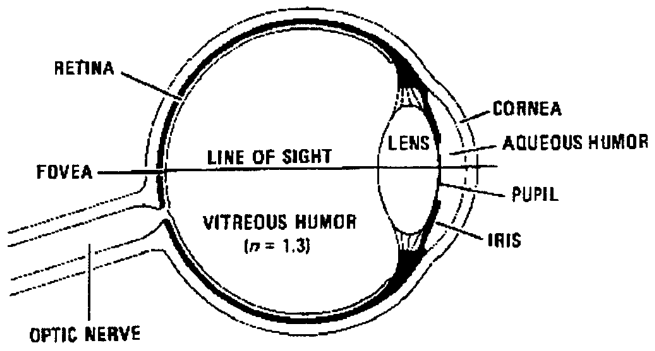


Figure 2-1 from Sabins (1978).
Structure of the Human Eye.

Light enters through the cornea, separated from the lens by a clear fluid called aqueous humor. The iris, the pigmented part seen when viewing the anterior eye, controls the size of the pupil. This is the hole in the iris that controls the amount of light passing through the lens, and it can vary its opening area by a factor of 16:1. Pupillary contraction not only limits light transmittance through the center and optically best portion of the lens, but when constricted, assists in increasing depth of field while viewing nearby objects, much like a camera aperture. A common misconception regarding the function of the lens is that it refracts light entering the eye to form an image on the retina. Light actually refracts most at the cornea, where the coefficient of refraction is greatest (1.0 for air, 1.3 for the cornea). The lens is instead important for accommodating, that is, focusing between near and far objects. Muscles on the periphery of the lens change it's thickness to perform this. The image is displayed inverted on the retina, a thin sheet if photosensitive nerve cells; the location on the retina where the visual axis contacts the retina is called the fovea. This region has the greatest concentration of neuroreceptors designed for photopic (daylight color) vision, as shown in Figure 2-2, from Cornsweet (1970). These sensory neurons are descriptively called cones; photoreceptors designed for scotopic (low-light) vision are similarly called rods. There is an additional distinction between foveal cone receptors, which are more slender (and physically resemble rod shape) and more densely packed together, and peripheral cones, which have the more characteristic conical shape. The combination of this ability of the light-transmitting structures of the anterior eye to produce a focused retinal image, along with the concentration of foveal receptors determine *spatial resoluton.* This is the ability to visually discriminate two closely-spaced objects.Myopia (near-sightedness) and hyperopia (far-sightedness) are common conditions in which aspherical eye shape induces reduced resolving power.

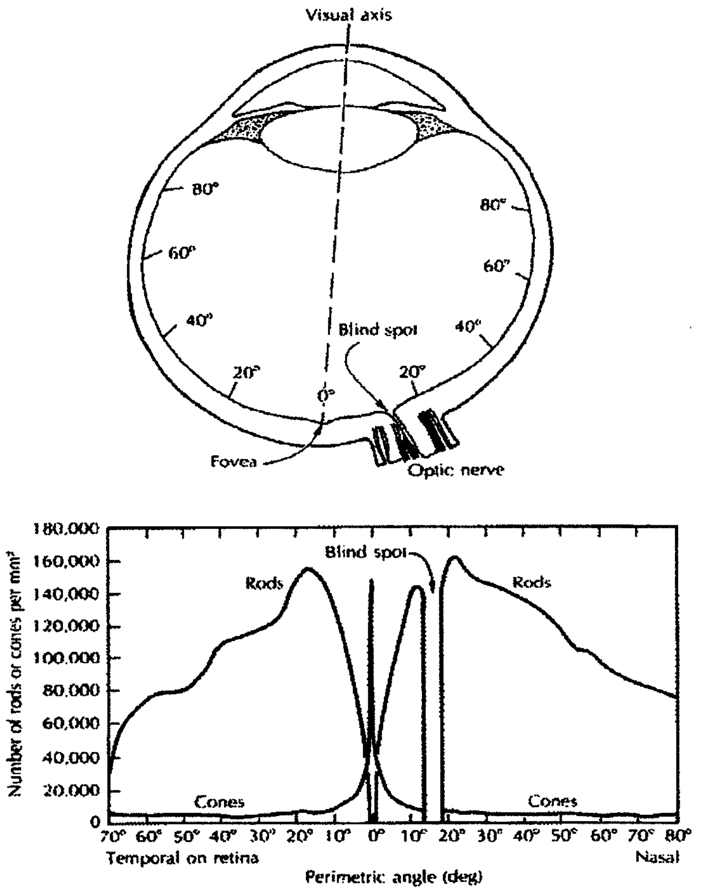


Figure 2-2 from Cornsweet (1970).
Ocular Neuroreceptor Density As
a Function of Angle From Fovea.

2. Features of Color Perception

a. Retinal Sensitivity

Sheppard (1968) states that "color vision has it's highest development in a foveal region approximately 2 degrees in extent." Consequently, most of the research in color perception is limited to data taken in this foveal field. Studies of the kinetics of cone pigments by Rushton (1963) and Wald (1964) provide estimates of the absorption spectra of cone pigments, shown in Figure 2-3 from Cornsweet (1970). Three different color systems were identified in the normal human eye, with color sensitivity based on the absorption spectra of the different pigments.

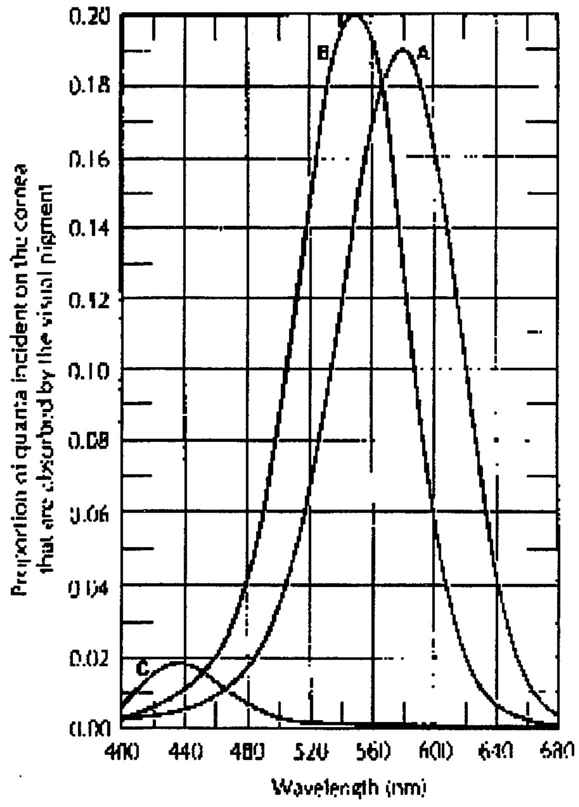


Figure 2-3 from Cornsweet (1970). Color Sensitivity of the Human Eye.

The peaks of the absorbtion curves occur at 440 nm, 550 nm, and 580 nm (these correspond to the approximate hues blue, green, and red, respectively), and the system that had peak absorbtion at 550 nm (green) also had the greatest relative absorbtion of all three color systems

b. Influences On Retinal Sensitivity

Dark and light adaptation is an important factor related to color perception. Wald (1945) showed that if a subject with normal vision is initially light-adapted and placed in a dark environment, photopic sensitivity diminishes and scotopic sensitivity increases after several minutes. This constitutes dark adaptation. Since scotopic receptors are a separate vision system not related to the three color vision systems, color perception is also impaired.

Riggs, et al (1953) determined that eye movement is a factor in color perception. They presented the subject with a low intensity filament as a viewing target, and created a device which stabilizes this image on the subject's fovea. Within seconds, the subject reported the initially clear and sharp image disappeared, and the field of view became uniformly gray. This was labeled the temporal modulation transfer property, and is exhibited whenever a viewer stares fixedly at a low-spatial-frequency target such as figure 2-4, from Cornsweet (1970).

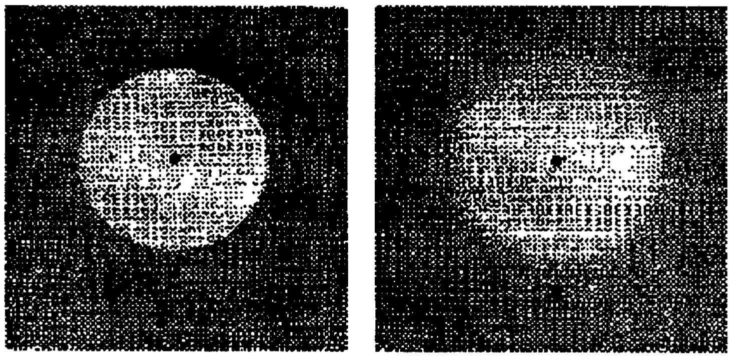


Figure 2-4 From Cornsweet (1970). Demonstration of Temporal Modulation Transfer Property. Staring at the Right-Handed (Low Spatial Frequency) Target Causes Optic Neural Saturation, Making the Disk Seem to Disappear.

Genetic deficiencies of the visual system may also limit color perception, commonly called "color blindness." Cornsweet (1970) discussed monochromacy and dichromacy as such conditions, in which respectively only one or two color systems participate in photopic vision, vice the normal three. The monochromat may perceive two different wavelengths of light as indistinguishable if both are displayed at the same intensity. The dichromat views two presented colors as indistinguishable if the same relative percentages of cones in his 2 active systems are stimulated. As a simplistic example, suppose two subjects, one with normal (trichromatic) color vision and the other with dichromatic vision view a pair of differently-colored objects, objects A and B. Objects A and B are both colored in such a way that stimulate 20 percent of the trichromat's "blue" color system receptors and 20 percent of his "green" color system receptors, but A stimulates 30 percent of his "red" system receptors, whereas B stimulates none. The trichromat will then recognize objects A and B as differently-colored. Conversely, the dichromat examines objects A and B, and his "blue" and "green" color system receptors are stimulated in the same fashion as those of the trichromat, but the dichromat has no "red" color system receptors, and therefore notes no color difference between objects A and B.

Pathologic conditions may also influence color perception. Daley, Watzke and Riddle (1992) describe the decrease in color sensitivity in the red and blue wavelengths of the diabetic.

3. Principles of Image Analysis

The aim of steganographically encoding data within a digital image is to do so without raising the suspicion of the unintended observer that the image was manipulated. If the observer is someone trained to analyze a digital image, then it is helpful in the study of steganography to understand what an analyst looks for in the image. Paine (1981) describes principles of photographic interpretation; while processes such as photographic geometric rectification may be performed differently than geometric rectification on a digital image, once the product is ready for viewing by the human analyst, the principles are essentially the same.

a. Absolute And Relative Size

Once the absolute size of a feature of interest is determined, the absolute size of other features of interest may be determined by comparison of relative size with the absolute.

b. Shape

Many objects may be identified by their two or three-dimensional shape. Vertical resolution is gained either through altimetry over the instantaneous field of view or by stereoscopic analysis of a pair of images of the same feature from different sensor locations.

c. Shadow

Shadows give the analyst a clue to profile shapes and relative sizes of structures. It is typically important to know the date time and location of the image, to determine the azimuth and altitude of the sun, or the direction of other emitters.

d. Tone Or Color

Digital imagery intended as photographic reproductions attempt to faithfully reproduce the various sensed colors. In other digital imagery, the knowledge of exact color may not be necessary or even desirable. Pixels with similar "digital signatures" (combinations of brightness values in different sensor channels) may be assigned a pseudotone or pseudocolor entirely independent of it's "real" color.

e. Texture

These are tonal or color changes on an object of interest that can be identified in a characteristic arrangement, typically ranging from fine to coarse.

f. Pattern

A spatial arrangement of objects. This differs from texture in that texture describes change within a single object.

g. Location, Association, And Convergence of Evidence

These principles are based on deductions the image analyst draws from other principles, plus any a priori knowledge has of the image. The analysis of a 6" diameter pipe may be identified as a gas line in one country and an irrigation pipe in another. Association is similar to location: a runway may be identified as a cropdusting airstrip if it adjoins farmland, whereas it might be labeled a community airport if located next to low buildings. Convergence of evidence is described as drawing together features or other information of an image to make an interpretation. The presence of dirt roads and stacked cut trees near a long building may lead the analyst to label the building as a sawmill.

C. COLOR MEASUREMENT AND ENCODING

Most images use color vice monochrome. Different color measurement schemes have been produced, and each has strengths and weaknesses, depending on the implementation. Russ(1995) provides a good introduction.

1. Look Up Tables (LUT's)

These are used commonly in scientific visualization, where the real color may not be of interest, but some other quality of the object is represented: elevation, velocity, and density are examples.

2. National Television Systems Committee (NTSC)

This scheme was developed during the transition of monochrome television broadcast to that of color television. It adds the color information within the same bandwidth the monochrome signal was sent. The result is that color pictures have less spatial resolution than monochrome. From typical viewing distances the viewer will tolerate the less sharply bounded color, and interprets the colors as bounded. Super VHS video was the subsequent improvement to NTSC, in that color information is separated from brightness information in recording and playback.

3. YIQ

This color scheme is commonly used in inexpensive solid-state color cameras where the photosensor chips have alternating vertical red, green, and blue sensor rows. This causes a reduction in lateral spatial resolution, since three sensors record the brightness values from one instantaneous-field-of-view (IFOV). More expensive cameras use beam-splitters to create three copies of the IFOV and route it to three separate monochrome cameras. YIQ encoding is related to RGB encoding in the following way: Y is the Luminance signal, a combination of the red, green, and blue, color intensities in the same proportion that the human eye is sensitive. I and Q color signal components are used for hardware compatibility. I represents red minus cyan, and Q represents green minus magenta. Table 2-1, taken from Russ (1995) shows the relationship between YIQ and RGB.

$Y = 0.299R + 0.587G + 0.114B$	$R = 1.000Y + 0.956I + 0.621Q$
$I = 0.596R + 0.274G + 0.322B$	$G = 1.000Y - 0.272I + 0.647Q$
$Q = 0.211R + 0.523G + 0.312B$	$B = 1.000Y - 1.106I + 1.703Q$

Table 2-1 From Russ (1995). Conversion Between RGB and YIQ Color Scales.

4. RGB

This scheme comes from the way camera sensors and display phosphors work. Figure 2-5, from Russ (1995), shows the RGB color space. Other encoding schemes better represent human perception, though this one is easiest to implement in hardware.

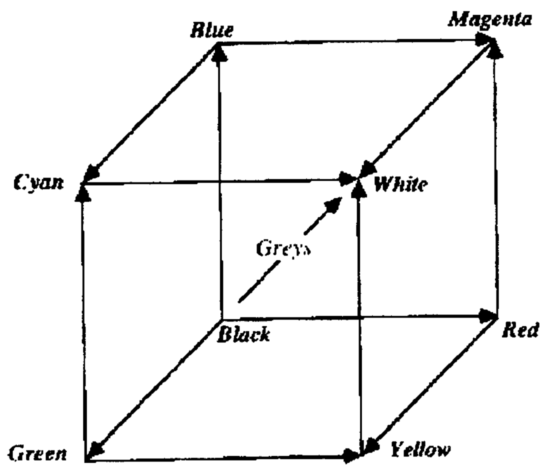


Figure 2-5. RGB Color Space.

5. CIE

The Commission Internationale L' Eclairage, or CIE chromaticity diagram is the oldest attempt to measure color the way it is perceived. Figure 2- 6, from Russ (1995) shows this two-dimensional diagram, which represents color components. A third axis, luminance, corresponds to panchromatic brightness, that would produce a monochrome image. The other two scales, x and y, represent a pair of values that describe all visible colors. A drawback in the CIE diagram is that it does not indicate the variation in color discerned by the eye, from Russ (1995) and does not directly correspond to hardware implementation.

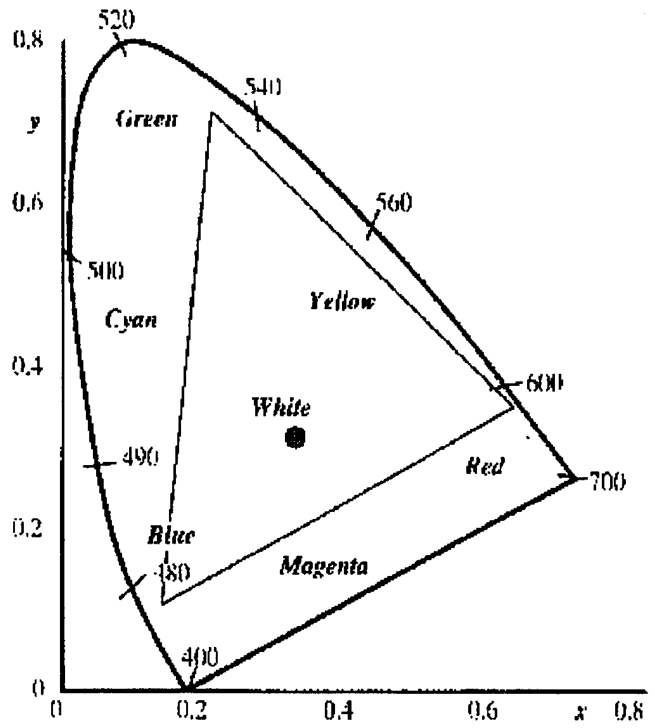


Figure 2-6 From Russ (1995).
CIE Chromacity Diagram.

6. HSI

The Hue, Saturation, Intensity color encoding approach better corresponds to hardware operation and human vision. Hue is the tint described by it's wavelength. Saturation is the amount, or purity, of color present, for example the distinction between pink and red. Intensity is the amount of light it emits, in other words, the difference between dark red and light red. Figure 2-7 shows the HSI colorspace and Table 2-2 indicates the relationship between RGB and HSI; both taken from Russ (1995).

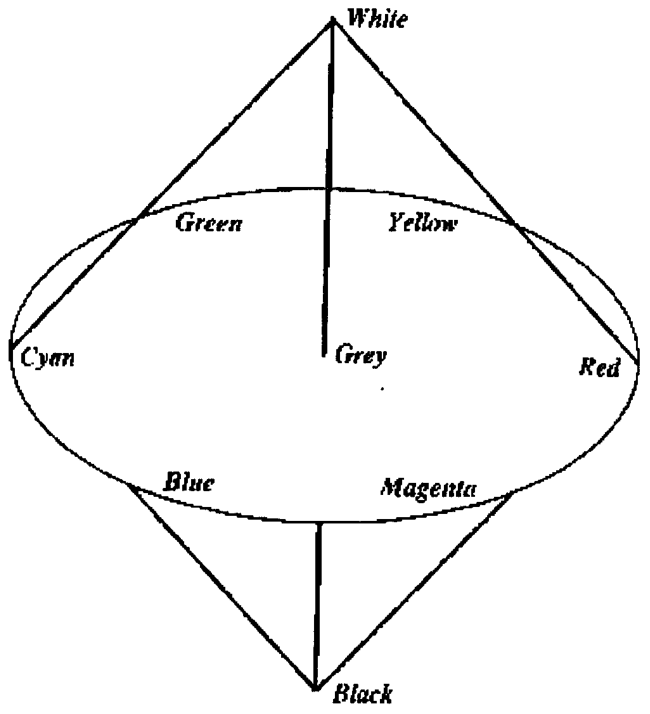


Figure 2-7 From Russ (1995).
Hue Saturation Intensity Color System.

$$
\begin{aligned}
H &= \left[\frac{\pi}{2} - \mathrm{atan}\,(\,\{\langle 2\times R - G - B\rangle / (\sqrt{3}\,) \times (G-B)\ \} + \pi)\,;(G<B)\right] / (2\pi) \\
I &= (R+G+B)/3 \\
S &= 1 - [\mathit{min}\,(\,(R,G,B)/I)\,]
\end{aligned}
$$

Table 2-2 From Russ (1995). Conversion from RGB Coordinates
to HSI Coordinates.

D. GRAPHICS FILE FORMATS AND FILE COMPRESSION

1. Graphics Files

Murray and VanRyper (1994) describe graphics files as "...files that store any type of persistent graphics data (as opposed to text, spreadsheet or numerical data, for example), and that are intended for eventual rendering and display. There are many different file formats in which graphics data are stored, but they can be generalized into two classes: vector and bitmap (formerly called raster).

a. Vector Data

Lines, polygons or curves are represented in vector data by numerically specifying keypoints and other attributes like color and line thickness. The rendering program uses a set of rules along with the vector data to draw objects.

b. Bitmap Data

This type of data is created from a set of values that specify the color of each picture element (pixel) of the image. Data in a header file usually specifies the dimensions of the rendered image, it's bit depth (the number of bits required to paint each pixel), interlacing method, and so on. These two classes of graphics data can be used in three different file formats: bitmap, vector, and metafile. As the former two names imply, bitmap format contains bitmap data, and vector file format holds vector data, while metafile format can hold both bitmap and vector data simultaneously.

2. Elements Of Graphics Files

A graphics file is composed of data structures that Murray and VanRyper (1994) call file elements. They describe the following data structures: fields, tags and streams. Fields are structures that are fixed both in size and location within the file. They are positioned either as an absolute offset from the beginning of the file, or as a relative offset from some other element. Tags may vary in size and position in graphics files, but can be optionally specified in both size and position as well. They may contain fields or tags. While tags and especially fields aid in random access, streams, conversely must be read sequentially. The trade-off is that sequential data can be found more quickly than random access data, especially if the stream is of a large body of data.

3. Data Compression

A large drawback of graphics data is the sheer size of the uncompressed data. Data compression permits the data to be stored in a file of manageable size, though with some compression methods, specifically “lossy”, some of the graphics information is lost in the compression process. Murray and Van Ryper (1994) describe some common compression algorithms:

a. Pixel Packing

Though not a compression method per se, it is an efficient method used to store data in contiguous memory bytes. If it is known, for example, that that an image encodes a pixel value using only four bits, then the remaining bits in every byte of the file are wasted. A pixel packing algorithm fills the remaining bits of one pixel with the next pixel’s value , and hence minimizes such waste.

b. Run Length Encoding

Run Length Encoding (RLE) reduces the size of repeating characters (called a run) into sets of two-byte RLE packets. The first byte of the packet contains the length of the run. Permissable values are 1 to 128 and the value 256. The second byte contains the run value, the character that would be repeated for the run length in the uncompressed file.This method is commonly used in bitmap formats like .BMP, .TIFF, and .PCX, and reduces redundant data.

c. Lempel Ziv Welch (LZW)

This type of compression is used in .GIF and .TIFF file formats. It builds a translation table called a data dictionary when it writes a code value for a string not already contained in the dictionary. It writes the dictionary code in the output file in place of the string.

d. *CCITT Encoding*

This compression is used for facsimile transmission, standardized by the International Telegraph and Telephone Consultative Committee; it implements Huffman coding as part of the standard.

e. *Joint Photographic Expert's Group (JPEG)*

This is a set of compression standards that work well for continuous-tone imagery, that is, those that show a gradient of brightness tonal change, vice distinct boundaries of brightness value changes. It uses a coding scheme based on the Discrete Cosine Transform algorithm.

E. CURRENTLY AVAILABLE STEGANOGRAPHY SOFTWARE

There are several shareware and freeware programs available via file transfer protocol (FTP) from the Internet. This section provides a brief description of their functionality. Most of this information was compiled from Johnson (1996).

1. Hide And Seek, Version 4.1

This freeware, created by Maroney (1994), is a MS DOS program that embeds and extracts data from .GIF format image files. It will embed up to 19,000 bytes (the author emphasizes <u>not</u> 19Kb) into a .GIF file of maximum size 320 x 480 pixels. It traverses through the graphics data stream and randomly selects image bytes to encode with data from a user-specified message file. It uses Least Significant Bit (LSB) replacement to encode, removing the LSB of the image byte and replacing it with a bit of message file data. It also uses IDEA to encrypt the program-specific header information.

2. StegoDOS

This collection of programs, also known as Black Wolf's Picture Encoder (named after it's anonymous creator) is also freeware. It will encode data files up to 8Kb in size within a picture file of maximum size 320 x 480 pixels and of 256 colors. It doesn't specify any specific image format; additionally, the user must supply the graphics file display and screen capture software to use this utility . It encodes the screen capture file with data using the LSB replacement method, and since it uses screen capture, doesn't overwrite the original image. It is cumbersome to use, because it requires the user to perform multiple steps to encode and extract data.

3. White Noise Storm

This MS DOS shareware was created by Arachelian (1994) It also uses LSB encoding and extraction method on .PCX format files. This file format was originally used by "PC Paintbrush" before Microsoft bought rights to it, renamed it Microsoft Paintbrush, and bundled it with Windows 3. White Noise Storm additionally encrypts the message file prior to encoding into the image, and randomly encodes image bytes.

4. S-Tools Version 3

This shareware by Brown (1994) operates under MS Windows 3.0; it encodes and extracts using LSB replacement on not only .BMP and .GIF files, but on audio .WAV files. It also has a utility to steganographically encode unused disk space on floppy disks. It supports 24 bit .BMP color and supports encryption of the input message file using IDEA, MPJ2, DES, and NSEA. The graphic user interface makes it easy to use for those accustomed to using MS Windows.

5. Stego Version 1.0a2

This shareware program, created by Machado (1994) operates under Mac OS and performs encoding and extraction on the 8, 16 and 32 bit .PICT files used in the Apple Macintosh platform. It encodes using LSB replacement for each of the red, green, and blue color values, and in the case of indexed color, performs the replacement on the index values. It allows the user to display the image prior to and after encoding. A drawback of this program is that the input image is also the output image, and therefore the original image is corrupted.

III. FEATURES OF STEGANOGRAPHY TOOLBOX

A. GENERAL DESCRIPTION

The intent in creating Steganography Toolbox was to provide a platform for the study and development of steganography algorithms. It provides the functionality necessary to convert digital image files to other graphic file formats, display images, steganographically encode into and extract previously encoded data from those images using steganographic algorithms, and generate simple statistics. It fills a void of need for the student conducting research in steganography at Naval Postgraduate School since it is designed to operate in an X Windows environment. To date, no other software performs steganography using a graphic user interface and runs on the UNIX operating system; all other implementations are designed for use on personal computers, or as a command-line language implementation for UNIX. Figure 3-1 shows the menu structure of Steganography Toolbox.

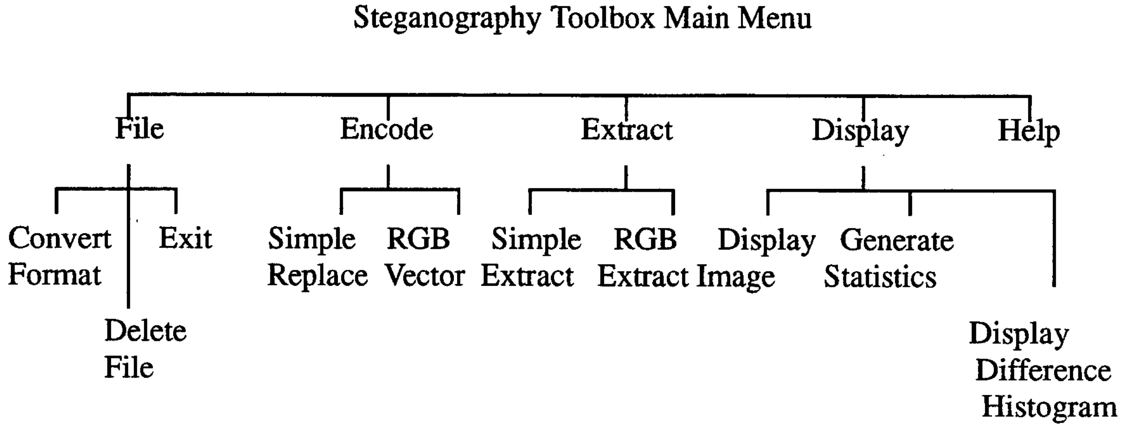


Figure 3-1. Menu structure of Steganography Toolbox

B. FILE MENU

1. Description

The File menu selection permits file manipulation and program exit. It contains submenus Convert Format and Exit.

2. Submenu Selections

a. Convert Format

Convert Format will convert an image of JPEG, Compuserve GIF and Many other popular image formats to Microsoft Windows Bitmap format. While none of the steganographic encoding or extraction modules in this version of the Steganography Toolbox accept graphics file formats other than Microsoft Windows Bitmap, Currie and Campbell(1996) studied survivability of data encoded in .BMP format and converted to a file format that uses lossy compression, noting any change in the result from the original .BMP file. As is discussed in the following chapter on Suggestions for Future Development, It is the author's belief that steganographic encoding that survives file compression is a logical next step in research, so the additional functionality was included in this version of Toolbox.

b. Delete File

The Delete File selection permits the user to browse the directory and delete unwanted files. The user must respond to a confirming dialog box to complete the deletion.

c. Exit

The Exit selection closes all windows not associated with independently running processes and terminates the program. Since some of the selections, like Display Image, operate as a UNIX system call to XV graphic image viewing utility and are made to run independently, windows associated with processes as these must be closed separately.

C. ENCODE MENU

1. Description

The Encode menu selection performs steganographic encoding of digital imagery. This version performs such encoding on images in Microsoft Windows Bitmap format only. It contains submenu selections Simple Replace and RGB Vector.

2. Submenu Selections

a. Simple Replace

The Simple Replace menu selection displays a panel through which the user supplies an input image (in .BMP format), a data file to be encoded into the image, the encoding "density", that is, the number of least significant bits to strip off the image bytes during the encoding process, and an output file name. It then performs simple replacement steganographic encoding with software adapted from Currie and Campbell (1996) an writes the result to a file of the indicated name. The user can view a help panel on this selection or cancel the selection altogether.

b. RGB Vector

The RGB Vector menu selection displays a panel through which the user supplies an input image (in .BMP format), a data file to be encoded into the image, and an output file name. It then performs RGB Vector steganographic encoding with software adapted from Currie and Campbell (1996) an writes the result to a file of the indicated name. The user can view a help panel on this selection or cancel the selection altogether.

D. EXTRACT MENU

1. Description

The Extract menu selection performs data extraction from previously steganographically encoded digital imagery. This version performs such extraction on images in Microsoft Windows Bitmap format only. It contains submenu selections Simple Replace and RGB Vector.

2. Submenu Selections

a. Simple Replace

The Simple Replace menu selection displays a panel through which the user supplies an input image (in .BMP format) that was previously encoded using the Simple Replacement method, and a file name for the output extracted data. It then performs data extraction from the previously encoded image, and writes the result to a file of the indicated name. The software which performs the extraction algorithm is adapted from Currie and Campbell (1996). The user can view a help panel on this selection or cancel the selection altogether.

b. RGB Vector

The RGB Vector menu selection displays a panel through which the user supplies an input image (in .BMP format) that was previously encoded using the The RGB Vector method, and a file name for the output extracted data. It then performs data extraction from the previously encoded image, and writes the result to a file of the indicated name. The software which performs the extraction algorithm is adapted from Currie and Campbell (1996). The user can view a help panel on this selection or cancel the selection altogether.

E. DISPLAY MENU

1. Description

The Display menu offers the user the opportunity to view digital imagery prior to and after steganographic encoding and extraction. It also provides limited statistical analysis of the change between an unencoded original image and it's encoded equivalent.

2. Submenu Selections

a. Display Image

The Display Image selection permits viewing of a digital image before and after steganographic encoding, using the XV Version 3.00 graphic file viewer. Since this is spawned as an independent process, the user can therefore display as many separate images as desired, depending only on the display capacity of the user's computer system. All of the graphic editing features of XV are also available to the user. A file browser panel is displayed and the user selects a file to view.

b. Generate Difference Image

The Generate Difference Image selection allows the user to perform a bytewise comparison of the graphic body of an unencoded digital image and it's encoded representation. The User selects the two files to compare using file browser boxes and activates the process by pressing the OK button. It then creates a Microsoft Windows Bitmap format image that shows the absolute change of each red green and blue byte, with darker pixels representing greater change in the pixel. The software which performs the comparison algorithm is adapted from Currie and Campbell (1996). The user can view a help panel on this selection or cancel the selection altogether.

c. *Display Histogram*

The Display Histogram selection shows a histogram of the per pixel absolute difference of two images that had the selection Generate Difference Image performed on them. The file names of each are listed , and the total number of pixels changed. Each histogram bin represents three bits of absolute change, i.e., one to three bits per pixel, four to six bits, and so on up to complete change (24 bits).The user can view a help panel on this selection while the panel is displayed.

F. HELP MENU

The Help menu provides a scrollable textbox of each menu selection in Steganography Toolbox. These are the same information panels that can be reached from menu selection panels when the "Help" button is pressed.

IV. IMPLEMENTING STEGANOGRAPHY TOOLBOX

A. THE DEVELOPMENT PACKAGE: SIMPLE USER INTERFACE TOOL (SUIT)

1. General Description

The Steganography Toolbox user interface was created using the Simple User Interface Tool (SUIT). Information regarding SUIT in this chapter is summarized from Conway et.al. (1992). SUIT was authored by Young (1990). It is a library of subroutines, written in the ANSI C, that create graphic user interfaces (GUI's). It acts as a window manager for screen objects, and permits interactive property modification of widgets. (A widget is a user interface component composed of data structures and callbacks. Examples are button widgets that activate callback functions, and scroller lists that select and pass text strings to other objects.) The widget property modifications are then saved in a corresponding SUIT properties file, called a ".sui file", unless those widget properties are "hard coded" into the main program. On subsequent execution of the main program, the ".sui" file provides the object property definitions that SUIT will apply to the object.(If it should become necessary to modify the ".sui" file for a given executable program, it will be the one with the same file name as the executable, plus the suffix "sui".) In it's SPARC platform implementation, SUIT is available via file transfer protocol (FTP) from the University of Virginia Computer Science Department as a 2.1 Mb gzipped tar file. Installation is brief and requires little user interaction. Packaged with the installation program are a tutorial, demonstration program source code samples that show widget function, and a makefile for the source code. The user should be familiar with C programming, and have available a C compiler and text editor.

2. Components of the Typical SUIT Graphic User Interface Implementation

Every program that uses SUIT to generate a GUI contains four elements: The first three are widgets, callbacks, and an external control loop; these are written in the main program source code. The fourth element is the corresponding ".sui" file, initially generated by SUIT when it compiles the main program, which maintains a list of the associated program's widget properties. Figures 4-1 and 4-2 (over the next several pages) show examples of each. Properties such as size location, and color, that the user desires to keep "hard coded", that is, permanently applied at program invocation, may be written directly into the body of the main program.

```
#include "suit.h"


void GetAFile() {
    char *fname;

    fname = SUIT_askForFileName("/usr/bin/","Print","File name:");

    if (fname != NULL)
printf ("user asked for file %s\n", fname);
    else
printf ("user pressed CANCEL\n");
}

void main (int argc, char *argv[])
{
    SUIT_init(argv[0]);

    SUIT_createButton("get a File Name", GetAFile);

    SUIT_createDoneButton(NULL);
    SUIT_beginStandardApplication();
}
```

Figure 4-1. Sample main program source code

```
/* SUIT version 2.3 */

#define THE_SCREEN_WIDTH 960
#define THE_SCREEN_HEIGHT 600
#define THE_SCREEN_DEPTH 7

#include "suit.h"


/*----------------------------------------------------------------------------*/
/*
  *//*
NOTE:                              */
/
*
  */
/*  This file contains all the permanent properties of this application's     */
/*  objects and is read in as a data file. Compiling this file as part of    */
/*  your application "hard codes" your interface. Please see "shipping"      */
/*  in the SUIT reference manual for further information.                   *//
*
  */
/
*
  */
/*                     !! DANGER
!!                          */
/
*
  */
/*    This file is machine-generated, and its contents are very
important.    */
/
*
  */
/*       If you must edit this file, do so with extreme
caution!          */
/
*
  */
/*                     !! DANGER
!!                          */
/
*
  */
/*-------------------------------------------------------------------------
----*/
```

Figure 4-2. Sample SUIT properties file

```
extern void SUIT_interiorInitFromCode (char *programName, SUIT_functionPointer
suiRoutine,
                              int width, int height, int depth);

extern SUIT_type *si_getType(char *typeName);

extern void si_addChildToObject(SUIT_object, SUIT_object, boolean);

static void MAKE (char *name, char *class, char *parent, boolean interactive)
{
   SUIT_object o = SUIT_name(name);
   SUIT_object p = SUIT_name(parent);
   if (p != NULL) {
      if (o == NULL)
         o = SUIT_createObjectByClass(name, class);
   if (interactive) {
      SUIT_setBoolean (o, INTERACTIVELY_CREATED, TRUE);
      SUIT_makePropertyTemporary (o, INTERACTIVELY_CREATED, OBJECT);
   }
      si_addChildToObject(SUIT_name(parent), o, FALSE);
   }
}



static void SET (char *objOrClass, char *propertyName, char *propertyType,
boolean atClass,
            boolean locked, char *stringValue)
{
   SUIT_type*type;
   booleanerrorStatus;
   Pointerretval;
   SUIT_level level = OBJECT;
   SUIT_object o;

   if (atClass)
      level = CLASS;
   if (level == CLASS)
      o = SUIT_dummyObjectInClass(objOrClass);
   else
      o = SUIT_name(objOrClass);
   type = si_getType(propertyType);
   retval = type->convertFromAscii(stringValue, &errorStatus);
   if (errorStatus == FALSE)
SUIT_setProperty(o, propertyName, propertyType, retval, level);

   if (locked)
      SUIT_lockProperty(o, propertyName, level);
}
```

Figure 4-2 (continued). Sample of a Properties File

```
static void INIT_suiRoutine(void)
{
/* This line is for parsing simplicity -- do NOT remove it! @ */
MAKE("Done","button","ROOT",0);
MAKE("get a File Name","button","ROOT",0);
SET("arrow button","border raised","boolean",1,0,"no");
SET("arrow button","changed class","boolean",1,0,"no");
SET("arrow button","darken background","boolean",1,0,"yes");
SET("arrow button","direction","SUIT_enum",1,0,"\"up\" of {\"up\" \"down\"
\"left\" \"right\"}");
SET("arrow button","draw filled","boolean",1,0,"no");
SET("arrow button","has background","boolean",1,0,"no");
SET("arrow button","intermediate feedback","boolean",1,0,"no");
SET("arrow button","shadow thickness","int",1,0,"3");
SET("borderless file box","changed class","boolean",1,0,"no");
SET("borderless file box","file filter","text",1,0,"");
SET("borderless file box","has border","boolean",1,0,"no");
SET("bounded value","arrowhead angle","int",1,0,"10");
SET("bounded value","arrowhead length","double",1,0,"0.200000");
SET("bounded value","button background color","GP_color",1,0,"black, black");
SET("bounded value","button foreground color","GP_color",1,0,"grey, white");
SET("bounded value","changed class","boolean",1,0,"no");
SET("bounded value","current value","double",1,0,"0.000000");
SET("bounded value","granularity","double",1,0,"0.000000");
SET("bounded value","has arrow","boolean",1,0,"yes");
SET("bounded value","has tick marks","boolean",1,0,"yes");
SET("bounded value","increase clockwise","boolean",1,0,"yes");
SET("bounded value","minimum value","double",1,0,"0.000000");
SET("bounded value","needle color","GP_color",1,0,"black, black");
SET("bounded value","start angle","double",1,0,"0.000000");
SET("button","changed class","boolean",1,0,"no");
SET("button","disabled color","GP_color",1,0,"white, white");
SET("button","interactively created","boolean",1,0,"no");
SET("button","justification","SUIT_enum",1,0,"\"center\" of {\"left\"
\"center\" \"right\"}");
SET("button","shrink to fit","boolean",1,0,"yes");
SET("dialog box","border type","SUIT_enum",1,0,"\"fancy motif\" of {\"simple\"
\"motif\" \"fancy motif\"}");
SET("dialog box","border width","int",1,0,"8");
SET("dialog box","cache using canvas","boolean",1,0,"yes");
SET("dialog box","changed class","boolean",1,0,"no");
SET("elevator","border raised","boolean",1,0,"no");
SET("elevator","changed class","boolean",1,0,"no");
SET("elevator","has background","boolean",1,0,"no");
SET("label","changed class","boolean",1,0,"no");
SET("label","has border","boolean",1,0,"no");
SET("label","justification","SUIT_enum",1,0,"\"center\" of {\"left\"
\"center\" \"right\"}");
SET("label","shrink to fit","boolean",1,0,"yes");
```

Figure 4-2 (continued). Sample of a Properties File.

```
SET("list","border raised","boolean",1,0,"no");
SET("list","changed class","boolean",1,0,"no");
SET("list","text spacing","double",1,0,"1.200000");
SET("place mat","border raised","boolean",1,0,"no");
SET("place mat","changed class","boolean",1,0,"no");
SET("scrollable list","border raised","boolean",1,0,"no");
SET("scrollable list","changed class","boolean",1,0,"no");
SET("scrollable list","has background","boolean",1,0,"no");
SET("scrollable list","has border","boolean",1,0,"no");
SET("type in box","any keystroke triggers","boolean",1,0,"no");
SET("type in box","backward char key","text",1,0,"C-b");
SET("type in box","beginning of line key","text",1,0,"C-a");
SET("type in box","beginning of text key","text",1,0,"M-<");
SET("type in box","border raised","boolean",1,0,"no");
SET("type in box","calculate lines","boolean",1,0,"no");
SET("type in box","changed class","boolean",1,0,"no");
SET("type in box","cursor color","GP_color",1,0,"black, black");
SET("type in box","cursor index","int",1,0,"0");
SET("type in box","cursor style","SUIT_enum",1,0,"\"vertical bar\" of {\"i-
beam\" \"vertical bar\"}");
SET("type in box","delete char key","text",1,0,"C-d");
SET("type in box","delete entire line key","text",1,0,"C-u");
SET("type in box","done editing key","text",1,0,"C-x");
SET("type in box","double click time","double",1,0,"400000.000000");
SET("type in box","end of line key","text",1,0,"C-e");
SET("type in box","end of text key","text",1,0,"M->");
SET("type in box","forward char key","text",1,0,"C-f");
SET("type in box","has a tab","boolean",1,0,"no");
SET("type in box","highlight block","boolean",1,0,"no");
SET("type in box","input sequence","text",1,0,"");
SET("type in box","kill line key","text",1,0,"C-k");
SET("type in box","last click time","double",1,0,"0.000000");
SET("type in box","last mark index","int",1,0,"0");
SET("type in box","mark end index","int",1,0,"0");
SET("type in box","mark index","int",1,0,"0");
SET("type in box","newline key","text",1,0,"C-m");
SET("type in box","next line key","text",1,0,"C-n");
SET("type in box","open line key","text",1,0,"C-o");
SET("type in box","previous line key","text",1,0,"C-p");
SET("type in box","repaint key","text",1,0,"C-l");
SET("type in box","scroll down key","text",1,0,"M-v");
SET("type in box","scroll up key","text",1,0,"C-v");
SET("type in box","set mark key","text",1,0,"C-`");
SET("type in box","shrink to fit","boolean",1,0,"yes");
SET("type in box","spacing gap","int",1,0,"3");
SET("type in box","start x","int",1,0,"0");
SET("type in box","start y","double",1,0,"0.000000");
SET("type in box","tab key","text",1,0,"C-i");
SET("type in box","tab length","int",1,0,"5");
SET("type in box","wipe block key","text",1,0,"C-w");
SET("type in box","yank key","text",1,0,"C-y");
```

Figure 4-2 (continued). Sample of a Properties File.

```
SET("ROOT","animated","boolean",0,0,"no");
SET("ROOT","background color","GP_color",0,0,"grey, white");
SET("ROOT","border color","GP_color",0,0,"grey, black");
SET("ROOT","border raised","boolean",0,0,"yes");
SET("ROOT","border type","SUIT_enum",0,0,"\"motif\" of {\"simple\" \"motif\"
\"fancy motif\"}");
SET("ROOT","border width","int",0,0,"2");
SET("ROOT","changed class","boolean",0,0,"no");
SET("ROOT","clip to viewport","boolean",0,0,"yes");
SET("ROOT","default object height","int",0,0,"80");
SET("ROOT","default object width","int",0,0,"80");
SET("ROOT","draw border on inside","boolean",0,0,"no");
SET("ROOT","font","GP_font",0,0,"times,,12.000000");
SET("ROOT","foreground color","GP_color",0,0,"black, black");
SET("ROOT","has background","boolean",0,0,"yes");
SET("ROOT","has border","boolean",0,0,"yes");
SET("ROOT","margin","int",0,0,"5");
SET("ROOT","show temporary properties","boolean",0,0,"no");
SET("ROOT","shrink to fit","boolean",0,0,"no");
SET("ROOT","springiness","SUIT_springiness",0,0,"63");
SET("ROOT","SUIT system font","GP_font",0,0,"helvetica,,14.000000");
SET("ROOT","viewport","viewport",0,1,"0 0 959 599");
SET("ROOT","visible","boolean",0,0,"yes");
SET("ROOT","window","window",0,1,"0.000000 0.000000 1.000000 1.000000");
SET("Done","active display","SUIT_enum",0,0,"\"standard\" of {\"button with
hotkey\" \"standard\"}");
SET("Done","border raised","boolean",0,0,"yes");
SET("Done","callback function","SUIT_functionPointer",0,0,"function ptr");
SET("Done","disabled","boolean",0,0,"no");
SET("Done","done callback function","SUIT_functionPointer",0,0,"function
ptr");
SET("Done","has background","boolean",0,0,"yes");
SET("Done","label","text",0,0,"Done");
SET("Done","viewport","viewport",0,0,"91 406 128 430");
SET("get a File Name","active display","SUIT_enum",0,0,"\"standard\" of
{\"button with hotkey\" \"standard\"}");
SET("get a File Name","border raised","boolean",0,0,"yes");
SET("get a File Name","callback function","SUIT_functionPointer",0,0,"function
ptr");
SET("get a File Name","disabled","boolean",0,0,"no");
SET("get a File Name","has background","boolean",0,0,"yes");
SET("get a File Name","label","text",0,0,"get a File Name");
SET("get a File Name","viewport","viewport",0,0,"220 372 311 396");


} /* end of INIT_suiRoutine */

void SUIT_initFromCode (char *programName)
{
  SUIT_interiorInitFromCode (programName, INIT_suiRoutine, THE_SCREEN_WIDTH,
                    THE_SCREEN_HEIGHT, THE_SCREEN_DEPTH);
}
```

Figure 4-2 (continued). Sample of a Properties File.

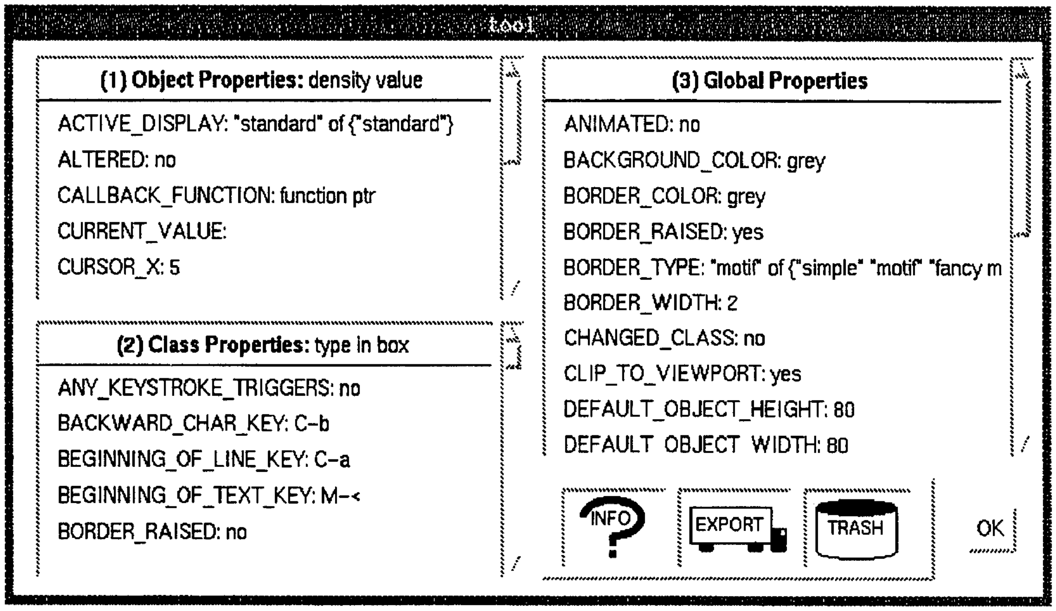
The SUIT library contains several predefined widgets and the more commonly used are presented:

- buttons - widgets that call functions when "pressed." Radio buttons toggle buttons, and pulldown menus are variants.
- bounded values - widgets that permit a double or integer type value to be passed to a function. Scrollbars, radial meters, pie slices and thermometers are the predefined representations of bounded values.
- bulletin boards - serve as a visual mounting surface on which to place other widgets.
- labels - widgets that permit text to be placed on other widgets.
- text boxes - widgets that permit a text string to be passed to a callback function. Multiple line text boxes with attached scroller widgets are also predefined.
- file browser boxes - permit a file, root path to a file, or both to be passes to a callback function. This is defined as a collection of widgets; namely, a textbox for the filename, a scrollable read-only list of files in the current directory, and "OK" and "Cancel" buttons.This widget also permits file filtering as a widget property, and directory traversal.

3. SUIT Properties and Inheritance

Each SUIT widget may have associated with it many properties. These are features which control the widget's appearance and function. As described previously, these properties may be written into the main program source code, or interactively manipulated by invoking SUIT's "property editor"while the program is running. When some newly compiled source code is executed, the user shall note that, unless the appearance was "locked in" at compilation time, the buttons, labels scrollboxes, and the like shall be randomly scattered about the window. This is because SUIT applies default values to the properties until the user modifies them. Upon exiting the program SUIT will store these modified properties in the ".sui" file, so that on subsequent invocation of the program, these properties may be applied to the appropriate widgets. The property editor is a dialog box which contains three scrollable lists, and an "OK" and "Cancel" button al child widgets. The scrollable lists display the object, class and global properties applied to the program. Figure 4-3 shows an example of a property editor

Each line of a text box represents a property. SUIT permits interactive manipulation of properties with mouse and keyboard strokes. For example of the boolean property HAS_BORDER can be interactively toggled between the values TRUE and FALSE. Enumerated properties like FOREGROUND_COLOR are presented as radio buttons. The user may change variable properties like MAXIMUM_VALUE by typing a value inside a text box. SUIT also provides error handling for invalid values.



(1) Object Properties: density value

ACTIVE_DISPLAY: "standard" of {"standard"}
ALTERED: no
CALLBACK_FUNCTION: function ptr
CURRENT_VALUE:
CURSOR_X: 5

(2) Class Properties: type in box

ANY_KEYSTROKE_TRIGGERS: no
BACKWARD_CHAR_KEY: C-b
BEGINNING_OF_LINE_KEY: C-a
BEGINNING_OF_TEXT_KEY: M-<
BORDER_RAISED: no

(3) Global Properties

ANIMATED: no
BACKGROUND_COLOR: grey
BORDER_COLOR: grey
BORDER_RAISED: yes
BORDER_TYPE: "motif" of {"simple" "motif" "fancy m
BORDER_WIDTH: 2
CHANGED_CLASS: no
CLIP_TO_VIEWPORT: yes
DEFAULT_OBJECT_HEIGHT: 80
DEFAULT OBJECT WIDTH: 80

Figure 4-3. Example of a property editor

SUIT manages properties in the following way: every object has a set of properties associated with it. SUIT first examines the object level property definitions. If it cannot bind all the associated properties with definitions, it looks at the class level properties for a binding. For example a widget named “radio_1” could be an object of class “radio_button”. If a binding is not found, SUIT applies a global property definition to the object. If the user changes the global property FOREGROUND_COLOR to black, and no object or class level bindings exist for any object, then all objects will be assigned the FOREGROUND_COLOR value black. Any property may also be “locked” with the command *SUIT_lockProperty()*, which can be changed only via program control, not through the property editor. Widgets may be defined as children of other widgets and thus inherit the parent widget properties. The SUIT property editor is invoked by placing the mouse cursor over the widget of interest and pressing SHIFT, CONTROL, and the letter "E" simultaneously. Properties are applied when the user presses "OK" in the property editor dialog box, or may press "Cancel" to close the editor window without modification.

4. Installing SUIT

SUIT is a platform-specific utility, with separate packages configured for Sun3, Sun 4 (Sparcstation), SGI IRIS, DEC, HP, IBM PC, and Macintosh. It is offered as a 2.1 Mb compressed (gzipped) tar file, so after downloading the compressed file the installer must place it in the desired directory, "gunzip" the file and then "tar xvf" the archive file. A README file is provided in the installation directory which guides the user through the rest of the installation. Usually the only modification required prior to executing the installation makefile is to list in the makefile the desired SUIT library path and the window environment library path. One of the subdirectories created contains a collection of Postscript files that make up a user's manual and tutorial. Source code for a demonstration of many different widgets, along with source code for implementing individual widgets and makefile are placed in another directory.

B. HOW STEGANOGRAPHY TOOLBOX WAS BUILT

1. Purpose

This section is included to assist the student interested in subsequent development of a steganography research tool using SUIT as the graphic user interface development utility. The coding structure of the interface and the manner through which processes are invoked by the interface are also presented.

2. Motivating Considerations

Shneiderman (1992) describes the interface form of Steganography Toolbox as "menu selection" and "form fill-in". Processes described by the functional specifications are invoked using dialog boxes. These dialog boxes are displayed via mouse selection through the main menu and linked drop-down menus. The overall menu structure is that of an acyclic graph; if the menu structure is considered independent of the help browser, however, the menu structure can be considered a tree of depth 3, where the leaf selections are help panels. The menu structure emphasized breadth over depth, as recommended by Kiger (1984) and Landauer and Nachbar(1985), and the organization of widgets within all the dialog boxes is similar as encouraged by Norman and Chin (1988), as discussed by Schneiderman (1992). It should be noted that SUIT appeared to impose some compiler -related limitations, which strongly influenced final development. SUIT was written in C, but the lengthy programs called by the interface were written in C++. A makefile to compile and link the GUI code and steganography programs was attempted using SUN (sparc) C++ version 2.1 and GNU C++ (also called g++), but the compiler error messages produced suggested the compilers rejected the way the SUIT button widgets were defined. Since a GUI is of limited utility without buttons, the makefile was changed to compile the GUI code in a C compiler (GCC), compile the steganography programs in a C++ compiler and then link the separately compiled object files. This didn't work either, because the GUI code had to tell the preprocessor to "#include" a header file for the steganography code (which itself had to "#include" C++ header files), again producing error messages. It was finally decided that, short of rewriting the steganography program source code or finding another GUI development utility for C++, that *system()* calls would be included in the GUI program, and that the front end of the steganography source code would be rewritten to have all parameters passed upon invocation, instead of interactively passed through standard input. While this may not be a preferred method to develop a software suite, it does provide one advantage. A student can design a steganography program to be called with all input parameters as single a command line entry,of the form *program_name argument1 argument2...* , and run it as an independent program at the command prompt, without worrying about how to write it into the rest of the GUI. The student can then just create a *system()* call "hook" to invoke the program from the user interface, adding functionality to the GUI in a relatively simple manner. It was the author's intent to have the image viewer (XV version 3.00) and file format converter (ImageMagick version 3.7) be invoked as system calls anyway, since both are full-featured utilities.

3. Coding the User Interface

a. Creating the External Control Loop

The heart of operation of the GUI is what Conway, et. al. (1992) call an "external control loop". It is essentially a continuous loop, which repeatedly checks for mouse or keyboard events, and responds accordingly. The segment of code which performs this in Steganography Toolbox is shown in Figure 4-4, lines 4 and 35. SUIT_init(argv[]) passes the executable name to the SUIT initialization program, which uses it to find the corresponding ".sui" file. *SUIT_beginStandardApplication(),* is listed after all objects are created and represents the external control loop.

```
1.void main (int argc, char *argv[])
2.{
3.  SUIT_object menuBar, fileMenu, encodeMenu,extractMenu, displayMenu,helpMenu;
4.  SUIT_init(argv []]);
5.
6.  fileMenu = SUIT_createPullDownMenu ("File");
7.  SUIT_addToMenu(fileMenu, "Convert Format", Convert);
8.  SUIT_addToMenu(fileMenu, "Delete File", Delete);
9.  SUIT_addToMenu(fileMenu, "Exit", Quit);
10.
11.  encodeMenu = SUIT_createPullDownMenu ("Encode");
12.  SUIT_addToMenu(encodeMenu, "Simple Replace (.BMP)", BMP_Simple);
13.  SUIT_addToMenu(encodeMenu, "RGB Vector (.BMP)", BMP_RGB);
14.
15.  extractMenu = SUIT_createPullDownMenu ("Extract");
16.  SUIT_addToMenu(extractMenu, "Simple Replace Extract   (.BMP)",BMP_Simple_X);
17.  SUIT_addToMenu(extractMenu, "RGB Vector Extract (.BMP)",  BMP_RGB_X);
18.
19.   displayMenu = SUIT_createPullDownMenu ("Display");
20.   SUIT_addToMenu(displayMenu, "Display Image", Displ);
21.   SUIT_addToMenu(displayMenu, "Generate Diff Image", Gen_Diff);
22.   SUIT_addToMenu(displayMenu, "View Histogram", genHist);
```

Figure 4-4. Main Program Code Segment

```
23.
24.  helpMenu = SUIT_createPullDownMenu ("Help Menu");
25.  SUIT_addToMenu(helpMenu, "Browse Help", GetHelpBox);
26.  SUIT_addToMenu(helpMenu, "About Help",  AboutHelpBox);
27.
28.  menuBar = SUIT_createMenuBar("Steganography toolbox");
29.  SUIT_addChildToObject(menuBar, fileMenu);
30.  SUIT_addChildToObject(menuBar, encodeMenu);
31.  SUIT_addChildToObject(menuBar, extractMenu);
32.  SUIT_addChildToObject(menuBar, displayMenu);
33.  SUIT_addChildToObject(menuBar, helpMenu);
34.  SUIT_createLabel("Steganography Toolkit");
35.  SUIT_beginStandardApplication();
36. }
```

Figure 4-4 (continued). Main Program Code Segment

b. Creating the Main Menu

Figure 4-4 also shows how the main menu was created. Each menu object was first created with *SUIT_createPulldownMenu()* as shown on lines 6, 11, 15, 19, 24, and 28.

c. Creating Submenus

Referring to Figure 4-4 once again, note that once each main menu object was created, the submenus were mapped to menu objects with the call *SUIT_addToMenu()*, The first parameter in *SUIT_addToMenu()* refers to the parent object; the second indicates the submenu object name; the third indicates the callback to invoke when the button is pressed.

d. Callback Functions

All but two of the callback functions, namely, *gen_Hist()* and *Quit()*, follow a nearly identical implementation form, the only difference being the number and type of widgets contained therein. Figure 4-5 is provided as an example which shows how a *system()* call is made to the simple replacement bitmap encoding program *bs_simple()*. The reader should keep in mind the desired output is a system call of the form: "bs_simple(char* inputfile, char* datafile, int density, char* output-file)”. First, all widget objects, including a dialog box object are declared. The dialog box is a predefined SUIT object which can contain a widget, an “OK” button, and a “Cancel” button.The switch statement examines the value of “button pressed”, which is either the value REPLY_CANCEL or REPLY_OK. Following execution of the switch structure, the dialog box is cleared from the screen with the statement *SUIT_destroyObject()*. Returning to Figure 4-5, once objects are declared, they are bound to widgets.

```
void BMP_Simple(SUIT_object menu){

  SUIT_object bs_container, bs_fbox1,bs_fbox2,bs_fbox3,density,
     bs_h_button,bs_dBox;
  char*  file1;
  char*  file2;
  char*  file3;
  char   bs[256];
  int    den_val;
  char*  den_char;

  bs_container = SUIT_createBulletinBoard (“BS Container”);
  bs_fbox1 = SUIT_createFileBrowser(
     “barney”,””,”BS File1”,”Select Input Image”,NULL);
  bs_fbox2 = SUIT_createFileBrowser(
     “wilma”,””,”BS File2”,”Select Input Data”,NULL);
  bs_fbox3 = SUIT_createFileBrowser(
     “dino”,””,”BS File3”,”Select Output Image”,NULL);
  density = SUIT_createTypeInBox(“density value”, NULL);
  bs_h_button = SUIT_createButton(“bs_help”,HelpBMP_Simple);


  SUIT_changeObjectSize(bs_container, 600, 340);

  SUIT_addChildToObject(bs_container, bs_fbox1);
  SUIT_addChildToObject(bs_container, bs_fbox2);
  SUIT_addChildToObject(bs_container, bs_fbox3);
  SUIT_addChildToObject(bs_container, density);
  SUIT_addChildToObject(bs_container, bs_h_button);

  SUIT_setViewport(bs_fbox1, VIEWPORT,
             SUIT_mapToParent(bs_fbox1, 0.05, 0.05, 0.3, 0.95));
  SUIT_setViewport(bs_fbox2, VIEWPORT,
             SUIT_mapToParent(bs_fbox2, 0.325, 0.05, 0.575, 0.95));
  SUIT_setViewport(bs_fbox3, VIEWPORT,
             SUIT_mapToParent(bs_fbox3, 0.6, 0.05, 0.85, 0.95));
  SUIT_setViewport(density, VIEWPORT,
             SUIT_mapToParent(density, 0.9, 0.775, 0.95, 0.825));
```

Figure 4-5. Example of How to Implement a Dialog Box

```
SUIT_setViewport(bs_h_button, VIEWPORT,
          SUIT_mapToParent(bs_h_button, 0.88,0.43,0.97,0.50));

bs_dBox = SUIT_createOKCancelDialogBox(“fred”, bs_container, NULL);
SUIT_activateDialogBox(bs_dBox);
switch (SUIT_getInteger(bs_dBox, “button pressed”)) {
 case REPLY_CANCEL:
  printf(“User selected CANCEL\n”);
  break;
 case REPLY_OK:
  file1 = SUIT_getText(bs_fbox1, CURRENT_VALUE);
  file2 = SUIT_getText(bs_fbox2, CURRENT_VALUE);
  file3 = SUIT_getText(bs_fbox3,CURRENT_VALUE);
  den_val =atoi(SUIT_getText(density, CURRENT_VALUE));
  printf(“found file1: %s\n”, file1);
  printf(“found file2: %s\n”,file2);
  printf(“found file3: %s\n”,file3);
  printf(“found density: %d %s\n “,den_val,” “);
    strcpy (bs, “bs_encode “);
    strcat (bs, file1);
    strcat (bs, “ “);
    strcat (bs, file2);
    strcat (bs, “ “);
    strcat (bs,(SUIT_getText(density, CURRENT_VALUE)));
    strcat (bs, “ “);
    strcat (bs, file3);
    strcat (bs, “&”);
    system (bs);
    printf (“done %s\n”, bs);
  break;
}

SUIT_destroyObject(bs_dBox);
}
```

Figure 4-5 (continued). Example of How to Implement a Dialog Box and Invoke a System Call

In this example, there was a bulletin board widget, three file browser widgets, a type-in box widget, and a button widget. The bulletin board widget is sized with the command *SUIT_changeObjectSize()*, and the other widgets are mapped to it as child objects. The bulletin board widget becomes the object to pass to the dialog box, and all the values generated by the child widgets are passed to the dialog box through the bulletin board object. Once the child widgets are oriented on the bulletin board (“hard-coded” as described previously) with the calls *SUIT_setViewPort()* and *SUIT_mapToParent()*, the dialog box widget and bulletin board are bound to the dialog box object name by the call SUIT_createOKCancelDialogBox(), and the box is displayed with

SUIT_activateDialogBox(). If the value of "button pressed" is equal to REPLY_OK, then the system command is invoked in the following way: a text string containing the name of the steganography executable program name is created and bound to a character array. the values of the necessary arguments are concatenated to the string. If it is to run as an independent process, an ampersand (&) is concatenated. Finally *system()* is called with the character array variable as the input parameter. The following submenu selections use the dialog box structure, where n*ame_1/name_2/name_3*...specifies the "path" of selections one must make to arrive at the ultimately desired selection.(the function of each submenu selection is described in Chapter III, FEATURES OF STEGANOGRAPHY TOOLBOX):

• File/Convert Format
• File/Delete File
• Encode/Simple Replace (.BMP)
• Encode/RGB Vector (.BMP)
• Extract/Simple Replace Extract (.BMP)
• Extract/RGB Vector Extract (.BMP)
• Display/Display Image
• Display/Generate Diff Image
•Display/View Histogram
•Help Menu/Browse Help

e. Displaying Histogram Data

As indicated in the previous subsection, all the steganography programs are executed using dialog boxes to gather the needed parameters and make the system call, with gen_Hist() indicated as one of the exceptions. The call gen_Hist() presents the viewer a histogram of values based on the last submenu selection of Generate Diff Image. During execution of the steganography executable program compare() (system-called by the selection Generate Diff Image), a file called "hist.dat" is opened and the character string values of the two input files are written. This is followed by the long integer values of the number of pixels that changed in value from one image to the next by one to three bits, then the number of pixels that changed by four to six bits, and so on, up to 22 - 24 bits change per pixel. Finally the long integer total number of changed pixels is written to "hist.dat" and it is closed. The callback function gen_Hist() looks at this file for data. Histogram scale heights are quantized from zero to 100, and the values passed to thermometer scroller objects' property CURRENT_VALUE using the call SUIT_setDouble() and SUIT_createBoundedValue. These widgets are mapped to the bulletin board with SUIT_addChildToObject() and positioned with *SUIT_setViewport()* and *SUIT_mapToParent()*. This is inserted into a dialog box that is activated with *SUIT_activateDialogBox()*.

f. Creating Help Widgets

The method used for creating a help widget is identical to that of any other dialog box widget with the difference being that the widget wrapped by the dialog box is a bulletin board with a text string mapped to it. It is displayed when either the "Help" button is pressed in an associated steganography function dialog box, or when an associated help topic selection is made via the Help Browser.If the help browser is used the text string selected is passed by the scrollable list in *GetHelpBox()* to *GetHelpTopic()*. *GetHelpTopic()* compares the text string to a set of text strings and if it finds a match, display that Help dialog box.

g. Communication from the Invoked Program back to the GUI

Main programs are implicitly defined to return an integer value to the operating system upon termination. By convention a program is written with a "return 0;" statement at the end, to indicate successful execution, while a returned "1" denotes an error occured. When using a *system()* call, however, *system()* appends a byte of its own on the value, to denote its own successful termination, regardless of what happened in the called program. Therefore, if a *system()*-called program returns "0", a zero is returned, but if the program returns a "1", *system()* will pass back "00000001 00000000", or "256". Keeping this in mind, if the program writer wants to pass back a value to indicate some program error by writing “exit(3)”, then Steganography Toolbox will see the value “768”. To get a graphic advisory panel to display, define an integer variable and assign it the value of the returned number, Then use this number as the argument to a switch statement, keeping in mind that it must be multiplied by 256 (that is,2^8). In the appropriate case block use, the SUIT_inform() callback to present an advisory callback,where the argument to SUIT_inform is the desired message, in quotes.

h. How Exit Works

The submenu selection File/Exit calls a function Quit(). In Quit(), there is the standard suit call SUIT_done(), which closes any active windows, saves any interactively created (and not “hard coded”) widget property values to the “.sui” file, and terminates the SUIT interface.

C. A SUGGESTED METHOD FOR IMPLEMENTING ADDITIONAL FEATURES IN STEGANOGRAPHY TOOLBOX

Assuming a requirements review has already been made and the user decides to add some feature to the current implementation, an ordered checklist is offered:

- Create separately executable code. The advantage of this is that it gives the student a specific task to focus upon, and his or her efforts may be devoted to creating and testing the code, not worrying about how it is linked to the GUI.

- Modify the main menu and submenus. The student can attach them to toy callback functions that print a specific text string to standard output to test.

- Write a callback function, in the form of a dialog box, which makes a *system()* call. This can also be tested with a toy callback function, that prints any values that ultimately will be passed in the *system()* call.

- Insert exit(integer) statements in a conditional statement within the system-called program to handle errors, and modify the integer-receiving switch statement in Steganography Toolbox to display a SUIT_inform panel.

- Add Help features.

V. REQUIREMENTS TESTING

A. TEST OVERVIEW

Requirements-based testing methodology was used on Steganography Toolbox. The following description of requirements-based testing was taken from Shimeall (1996). This chapter represents the Test design document. Test design documents are typically composed of the following elements:

- a description of the object tested,
- a list of expectations of proper behavior,
- a description of the planned test procedure,
- a description of the actual test procedure,
- results generated by the test procedure,
- an evaluation of the test procedure.

Since chapters III and IV provided a description of the test object, including expectations of proper behavior, these particular portions are omitted. Though a typical test design document additionally provides separate sections for a description of the actual tests performed (vice those planned), this chapter shall merge any discussion of actual testing that departed from the planned test battery with the evaluation of the test procedure. In order to develop a test plan, seven tasks are performed:

1. Determine goals,
2. Classify the goals,
3. Aggregate the goals,
4. Identify important cases,
5. Select data to exercise those cases,
6. Determine the expected results,
7. Sequence the cases,
8. Sequence the aggregates.

B. TEST GOALS

This section lists functional goals of Steganography Toolbox, as interpreted from Chapter III, FEATURES OF STEGANOGRAPHY TOOLBOX, chapter IV (IMPLEMENTING STEGANOGRAPHY TOOLBOX) as well as the Steganography Toolbox Functional Requirements of Appendix B. It must be stated that the scope of testing is, wherever possible, limited to the interface file source code the author wrote, Since the user interface uses the Simple User Interface Toolkit (SUIT) to create an executable GUI, the widget libraries will not fall within the scope of testing, nor will any of the independently executable programs called by Steganography Toolbox.Testing will instead include verification that the proper data is passed as an argument to *system()*, that erroneous data entered via widgets is properly handled, and that proper activations of callbacks within the graphic user interface occur. The test goals now follow:

1. The graphic user interface (GUI) shall present a main menu on program activation.
2. The GUI main menu shall consist of the submenu choices, File, Encode, Extract, Display, and Help.
3. The File submenu menu When Selected with a single left mouse button click shall present the selections: Delete File, Convert Format, Exit.
4. The Encode submenu menu When Selected with a single left mouse button click shall present the selections: Simple Replace, RGB Vector.
5. The Extract submenu menu When Selected with a single left mouse button click shall present the selections: Simple Replace, RGB Vector.
6. The Display submenu menu When Selected with a single left mouse button click shall present the selections: Display Image, Generate Diff Image, View Histogram.
7. The Help submenu menu When Selected with a single left mouse button click shall present the selections: Browse Help, About Help.
8. Delete File shall consist of a file browser widget mapped as a child to a dialog box.
9. When the Delete file OK button is pressed, the value of the filename to be

deleted shall be appended to a string preceded by the string, "rm" (blank space included).

10. When the Delete file OK button is pressed, a confirmation box will display asking, "Do you really want to do this?"
11. In Delete File, the character string shall be the argument in a *system()* call that causes the file specified by *fname* to be deleted from the directory in which it resides.
12. In Delete File, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
13. The Convert File panel shall consist of two file browser widgets mapped as children to a dialog box.
14. When the Convert File OK button is pressed, the value of the filename to be converted from shall be appended to a string containing the word "convert", and a blank space, which is followed by the filename which it will be converted.
15. In Convert File, the character string shall be the argument in a *system()* call that causes the first file specified by *fname*1 to be converted to the format specified by *fname*2.
16. In Convert File, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
17. The Exit Menu Selection shall cause the main menu panel to disappear and the program to terminate.
18. The Simple Replace panel shall consist of three file browser widgets, and a text box, mapped as children to a dialog box.
19. When the Simple Replace OK button is pressed, the value of the input image filename to be encoded into shall be appended to a string containing the word "bs_encode", and a blank space, which is followed by the input data filename to be encoded into the image, followed by a blank space, followed by the density, followed by a blank space, followed by the output filename.
20. In Simple Replace, the character string shall be the argument in a *system()* call that causes *bs_encode()* to be invoked.
21. In Simple Replace, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
22. The RGB Vector panel shall consist of three file browser widgets, mapped as children to a dialog box.
23. When the RGB Vector OK button is pressed, the value of the input image filename to be encoded into shall be appended to a string containing the word "br_encode", and a blank space, which is followed by the input data filename to be encoded into the image, followed by a blank space, followed by the output filename.
24. In RGB Vector, the character string shall be the argument in a *system()* call that causes *br_encode()* to be invoked.
25. In RGB Vector, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
26. The Simple Replace panel shall consist of two file browser widgets, mapped as children to a dialog box.

27.When the Simple Replace OK button is pressed, the value of the input image filename from which data is to be extracted shall be appended to a string containing the word "bs_extract", and a blank space, which is followed by the output data filename.
28.In Simple Replace, the character string shall be the argument in a *system()* call that causes *bs_extract()* to be invoked.
29.In Simple Replace, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
30.The RGB Vector Extract panel shall consist of two file browser widgets, and a text box widget, mapped as children to a dialog box.
31.When the RGB Vector Extract OK button is pressed, the value of the input image filename from which data is to be extracted shall be appended to a string containing the word "br_extract", and a blank space, which is followed by the output data filename, followed by a blank space, followed by a number that represents the filesize of the original input data file. This is nontestable as to whether it actually is the correct value, only whether it is a string of numbers ranging between zero and nine).
32.In RGB Vector Extract, the character string shall be the argument in a *system()* call that causes *br_extract()* to be invoked.
33.In RGB Vector Extract, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
34.Display Image shall consist of a file browser widget mapped as a child to a dialog box.
35.When the Display Image OK button is pressed, the value of the filename to be displayed shall be appended to a string preceded by the string, "xv" (followed by a blank space), followed by an ampersand(&).
36.In Display Image, the character string shall be the argument in a *system()* call that causes the file specified by *fname* to be displayed on the screen.
37.In Display Image, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
38.(Derived) In Display Image, pressing OK with no filename selected shall cause the "XV" panel to be displayed alone.
39.The Generate Diff Image panel shall consist of three file browser widgets, mapped as children to a dialog box.
40.When the Generate Diff Image OK button is pressed, the value of the first input image filename to be compared shall be appended to a string containing the word "compare", and a blank space, which is followed by the second input image filename to be compared, followed by a blank space, followed by the output image filename.
41.In Generate Diff Image, the character string shall be the argument in a *system()* call that causes *compare()* to be invoked.
42.In Generate Diff Image, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
43.The View Histogram submenu selection shall invoke callback *genHist()*.
44.The View Histogram panel displayed by *genHist()* shall contain eight scroller widgets mapped to display the eight long integer values contained in "hist.dat".

45.The View Histogram panel displayed by *genHist()* shall contain eight textbox widgets mapped to display the eight long integer values contained in “hist.dat”.
46.In View Histogram, pressing the Cancel or OK button shall cause the dialog box to be removed from the screen with no other action.
47.Browse Help shall consist of a scrollable list widget mapped as a child to a dialog box.
48.When the Browse Help OK button is pressed, the value of the help topic to be displayed shall be passed to *GetHelpTopic()*.
49.In *GetHelpTopic()*, the topic string passed to it shall cause the respective Help panel to be displayed.
50.In Browse Help, pressing the Cancel button shall cause the dialog box to be removed from the screen with no other action.
51.When About Help is Selected, an information dialog box shall display.
52. In Delete File, pressing OK with no filename selected shall be handled with an error message.
53. In Delete File, pressing OK with a filename selected that cannot be opened shall be handled with an error message.
54. In Convert File, pressing OK with insufficient parameters selected shall be handled with an error message.
55. In Simple Encode, pressing OK with not all arguments entered shall be handled with an error message.
56. In Simple Encode, pressing OK with an invalid density entered shall be handled with an error message.
57.In Simple Encode, attempting to open an input image file that cannot be opened shall cause an error message to be displayed.
58.In Simple Encode, attempting to open an input text file that cannot be opened shall cause an error message to be displayed.
59.In Simple Encode, attempting to open an output image file that cannot be opened shall cause an error message to be displayed.
60.In Simple Encode, an attempt to encode an image with a datafile of such a size and at such a density that does not embed all of the datafile shall cause an error message to be displayed.
61. In RGB Vector, pressing OK with not all arguments entered shall be handled with an error message.
62.In RGB Vector, attempting to open an input image file that cannot be opened shall cause an error message to be displayed.
63.In RGB Vector, attempting to open an input text file that cannot be opened shall cause an error message to be displayed.
64.In RGB Vector, attempting to open an output image file that cannot be opened shall cause an error message to be displayed.
65.In RGB Vector, an attempt to encode an image with a datafile of such a size and at such a density that does not embed all of the datafile shall cause an error message to be displayed.
66. In Simple Replace, pressing OK with not all arguments entered shall be handled with an error message.
67.In Simple Replace, attempting to open an input image file that cannot be

opened shall cause an error message to be displayed.

68. In Simple Replace, attempting to open an input text file that cannot be opened shall cause an error message to be displayed.
69. In Simple Replace, attempting to open an output image file that cannot be opened shall cause an error message to be displayed.
70. In RGB Vector Extract, pressing OK with not all arguments entered shall be handled with an error message.
71. In RGB Vector Extract, attempting to open an input image file that cannot be opened shall cause an error message to be displayed.
72. In RGB Vector Extract, attempting to open an input text file that cannot be opened shall cause an error message to be displayed.
73. In RGB Vector Extract, attempting to open an output image file that cannot be opened shall cause an error message to be displayed.
74. In Display Image, attempting to open an input image file that cannot be opened shall cause an error message to be displayed.
75. In Generate Difference Image, attempting to open an input image file that cannot be opened shall cause an error message to be displayed.
76. In Generate Difference Image, attempting to open an output image file that cannot be opened shall cause an error message to be displayed.
77. In Generate Diff Image, pressing OK with not all arguments entered shall be handled with an error message.
78. In View Histogram, attempting to open "hist.dat" that cannot be opened shall cause an error message to be displayed.
79. (Derived) In Browse Help, pressing OK with no topic selected shall be handled with an error message.

C. TEST GOAL CLASSIFICATION

Shimeall(1996) states test goal classification is typically divided into the following categories:

- •Non-testable goals: vocabulary definitions, examples, user action descriptions, and generalities.
- •Inspection goals: language usage, device usage, screen layout, and documentation.
- •Analysis goals: variable usage, expression usage, support software usage, and commenting.
- •Execution goals: calculation results, value transformations, value maintenance, efficiency, and timeliness.

In light of these descriptions the following goal classifications were made.

1.Non-Testable Goals

31.

2.Inspection Goals

1,2,8,13,18,22,26,30,34,39,47.

3.Analysis Goals

9,14,19,23,27,35,40,44,48.

4.Execution Goals

3-7,10-12,15-17,20,21,24,25,28,29,32,33,36-38,41-43,46,49-79

D. TEST GOAL AGGREGATES

Many of the goals have similar purpose. The most significant of these is the aggregation of goals that perform in the dialog box widget structure. A string is formed and *system()* call made. Therefore a test of proper string formation is appropriate. In the case of the Simple Encode selection, valid bit encode density values are an appropriate test aggregate, since it is independent of every the other goal. File selection and opening is also a necessary test aggregate, almost every menu selection needs to open image or text files.

1. String-formation

   9,14,19,23,27,35,40,48.

2. Valid Bit Encode Density

   56.

3. File Selection and opening.

   52-55,57-59,61-64, 66-78.

4. *System()* invocation

   11,15,20,24,28,32,36,41,43.

5. Proper image/Datafile Ratio

   60,65

6. Widget Organization

   2,8,13,18,22,26,30,34,39,44,45,47

7. Widget/Window Activation and Destruction

   1,3,4,5,6,7,10,12,16,17,21,25,29,33,37,38,42,46,49,50,51,

E. TEST SET-UP

1. String-Formation

This test determines whether the proper string is created prior to getting passed as an argument to the *system()* call. A *printf()* statement is inserted between the final string *strcat()* call and the *system()* call. A script is run on the terminal window that invokes Steganography Toolbox. The expected results are that proper strings will be written to standard output and the scriptfile for each selection. The following terminal window data appeared, as expected.

```
ac8> s-toolbox
rm /users/work4/woottend/thesis/suit/sparc/src/thesis/a.dat

../../../../ImageMagick/convert
/users/work4/woottend/thesis/suit/sparc/src/thesis/arc.gif
/users/work4/woottend/thesis/suit/sparc/src/thesis/arc2.bmp

bs_encode
/users/work4/woottend/thesis/suit/sparc/src/thesis/arc.bmp
/users/work4/woottend/thesis/suit/sparc/src/thesis/f16.bmp
/users/work4/woottend/thesis/suit/sparc/src/thesis/atest.bmp

bs_extract
 /users/work4/woottend/thesis/suit/sparc/src/thesis/atest.bmp
 /users/work4/woottend/thesis/suit

rgb_steg
/users/work4/woottend/thesis/suit/sparc/src/thesis/arc.bmp
 /users/work4/woottend/thesis/suit/sparc/src/thesis/lowder
 /users/work4/woottend/thesis/suit/sparc/src/thesis/atest.bmp

rgb_extract
/users/work4/woottend/thesis/suit/sparc/src/thesis/atest.bmp
 /users/work4/woottend/thesis/suit/sparc/src/thesis/atestout.bmp
5560

xv
/users/work4/woottend/thesis/suit/sparc/lsrc/thesis/arc.bmp&

compare
arc.bmp
atest.bmp
atestout.bmp
```

Figure 5-1. String Formation Test

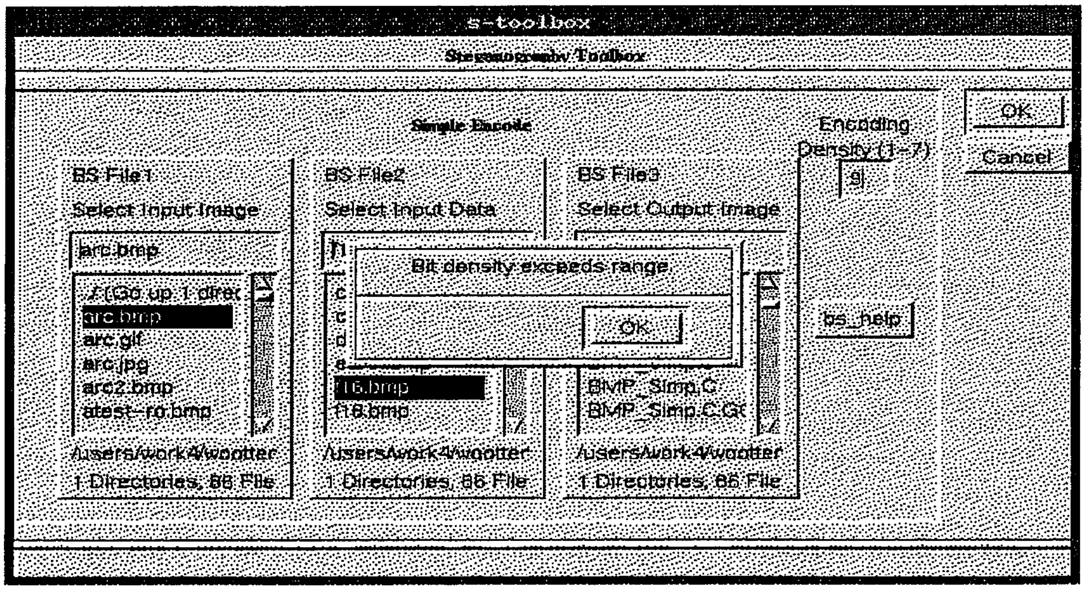
2. Valid Bit Code Density Test

The next test determines whether an invalid bit encode density value will elicit the appropriate error information box. This was tested by execution; the segment of code relevant to this test (Figure 5-2) is from generic_steg.C:

```
while(argv[3][j]){
     if (!(isdigit(argv[3][j])))
        exit(7);
     switch (argv[3][j]){
        case ‘1’:case’2’:case’3’:case’4’:case’5’:case’6’:case’7’:
        break;
        default:
           exit(9);
        break;
    }
     if (!(j==0))
       exit(9);
     j = j+1;
}
```

Figure 5-2. Code Segment From *bs_encode()*

The test values selected were, “6”, “9”, “2a”, and “77”. Since the values are part of a string it was necessary to use the code segment of Figure 5-2 rather than merely casting the value as an integer.The following screen displays resulted (Figures 5-3, 5-4, and 5-5):


s-toolbox

Steganography Toolbox

Simple Encode

Encoding Density (1-7)

9

OK

Cancel

BS File1

Select Input Image

arc.bmp

./ (Go up 1 direc

arc.bmp

arc.gif

arc.jpg

arc2.bmp

atest-ro.bmp

/users/work4/wootter

1 Directories, 88 File

BS File2

Select Input Data

f1

c

c

d

e

f16.bmp

f18.bmp

/users/work4/wootter

1 Directories, 88 File

Bit density exceeds range

OK

BS File3

Select Output Image

BMP_Simp.C

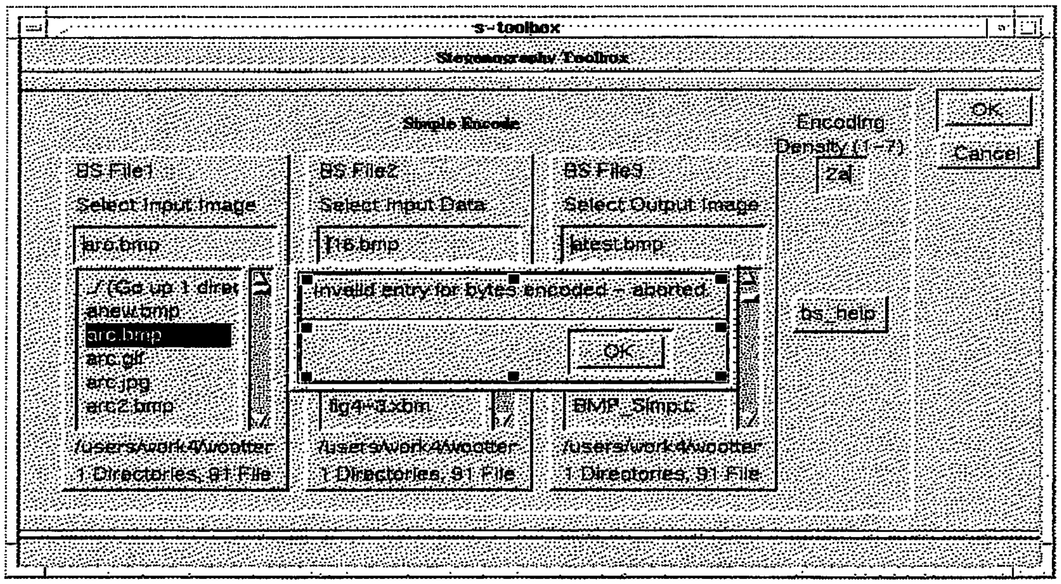
BMP_Simp.C.Gc

/users/work4/wootter

1 Directories, 88 File

bs_help

Figure 5-3. Error Raised Using “9”.


s-toolbox

Steganography Toolbox

Simple Encode

Encoding Density (1-7)

2a

OK

Cancel

BS File1

Select Input Image

arc.bmp

./ (Go up 1 direc

anew.bmp

arc.bmp

arc.gif

arc.jpg

arc2.bmp

/users/work4/wootter

1 Directories, 91 File

BS File2

Select Input Data

f16.bmp

fig4-3.xbm

/users/work4/wootter

1 Directories, 91 File

Invalid entry for bytes encoded - aborted.

OK

BS File3

Select Output Image

atest.bmp

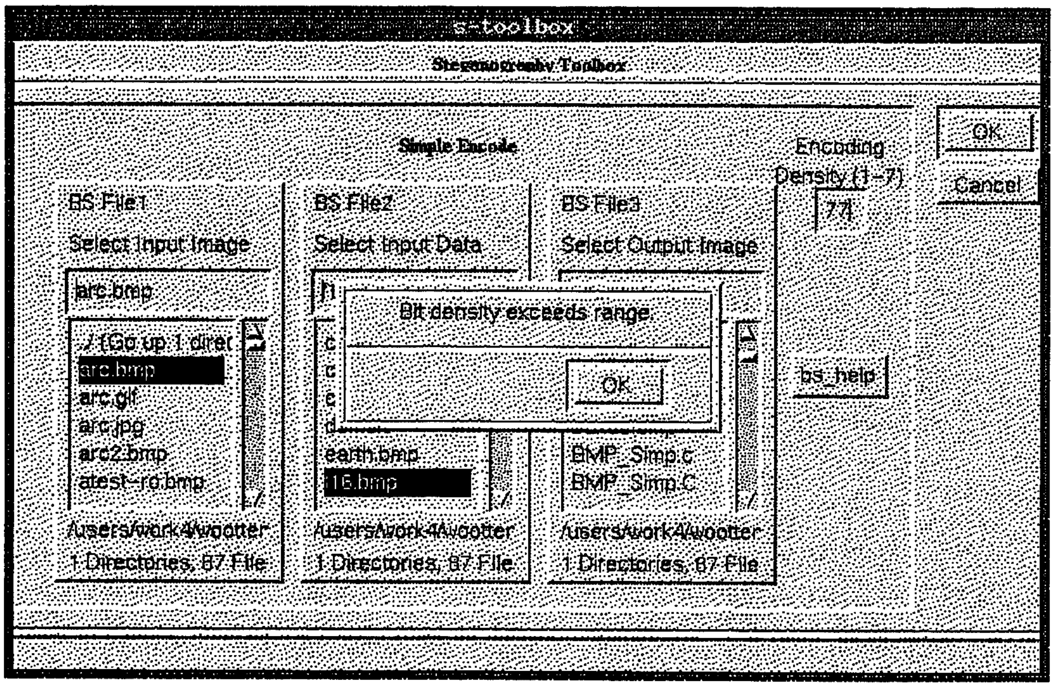
BMP_Simp.c

/users/work4/wootter

1 Directories, 91 File

bs_help

Figure 5-4. Error Raised Using “2a”



BS File1
Select Input Image
arc.bmp
../ (Go up 1 direc
arc.bmp
arc.gif
arc.jpg
arc2.bmp
atest-ro.bmp
/users/work4/wootter
1 Directories, 87 File

BS File2
Select Input Data
earth.bmp
16.bmp
/users/work4/wootter
1 Directories, 87 File

BS File3
Select Output Image
BMP_Simp.c
BMP_Simp.C
/users/work4/wootter
1 Directories, 87 File

Encoding Density (1-7)
77

Bit density exceeds range.
OK

Figure 5-5. Error Raised Using 77.

3. File Selection, Opening, and Comparison

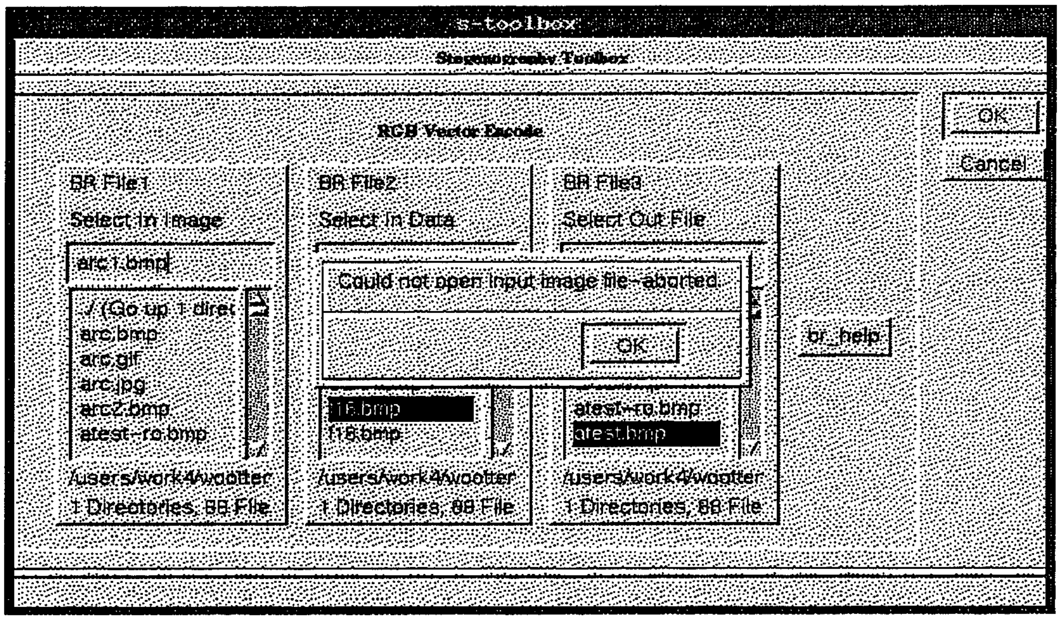
This test checks whether an error is raised when:

•An input file cannot be opened

•An output file cannot be opened.

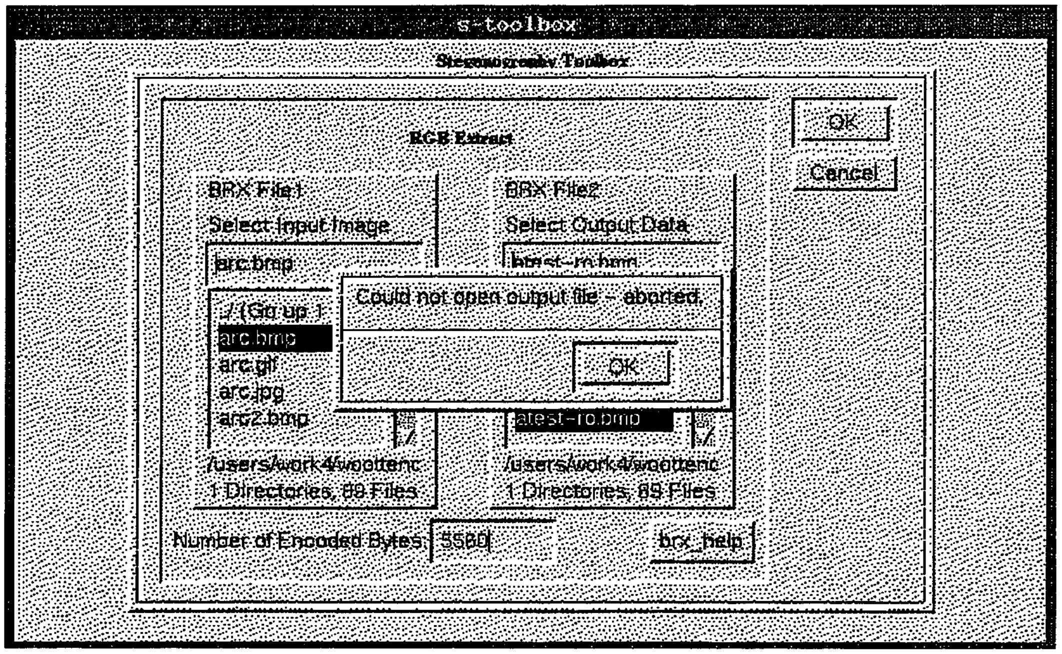
•Two files of are of different size, which causes the Generate Diff Image to abort.

The test plan for this is to attempt to read an input image file that does not exist in the directory, try to open an output file previously that had its read-write-execute mode set to read-only, and finally try to compare two images of different graphic data size using the selection Generate Diff Image. The following screen displays (Figures 5-6, 5-7, and 5-8) resulted, as expected.



Could not open input image file - aborted.

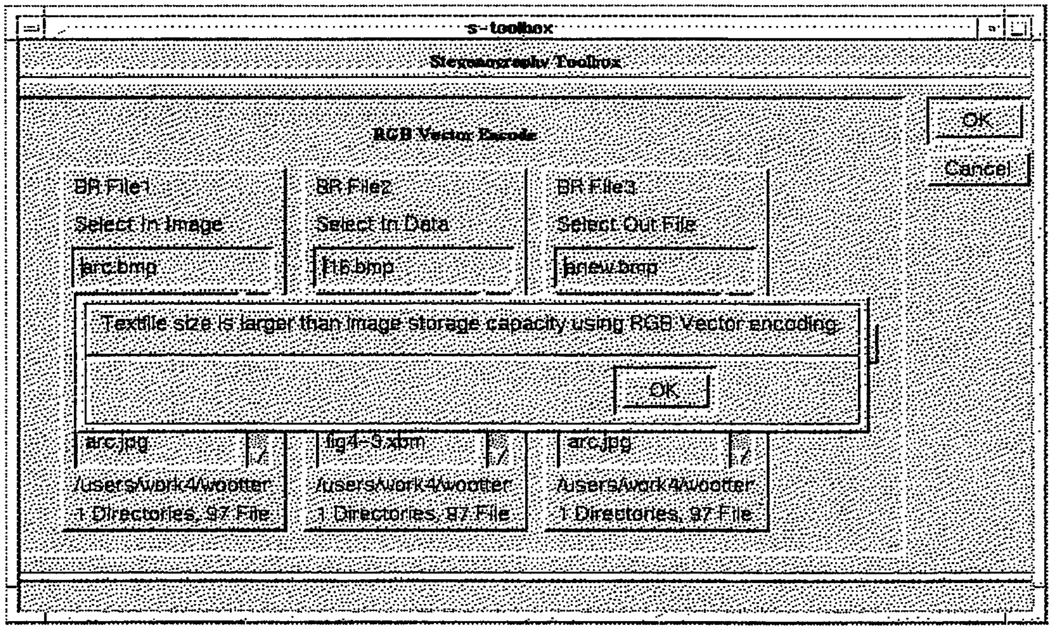
Figure 5-6. Input file cannot be opened.



Could not open output file - aborted.

Number of Encoded Bytes: 5580

Figure 5-7. Output File cannot be opened.



Textfile size is larger than image storage capacity using RGB Vector encoding.

Figure 5-8. Generate Diff Image Aborts when two files of different graphic data size are used as input images.

4. *System()* Invocation.

The plan for testing whether the system call resulted in a response was to open two terminal windows, one for Steganography Toolbox to operate from, and the other to test for a process fork using *ps()*, once prior to running the Simple Replace encode selection and one during. The expected result was that the process could be identified. Figure 5-9 shows the process bs_encode being forked.

```
ac8> ps
PID TT STAT TIME COMMAND
5637 co IW    0:00 /bin/csh /usr/bin/X11/xinit
5643 co IW    0:00 /usr/bin/X11/xinit.exec -- /usr/bin/X11/Xsun
5644 co S    12:22 /usr/bin/X11/Xsun:0
5645 co IW    0:00 sh /users/work4/woottend/.xinitrc
```

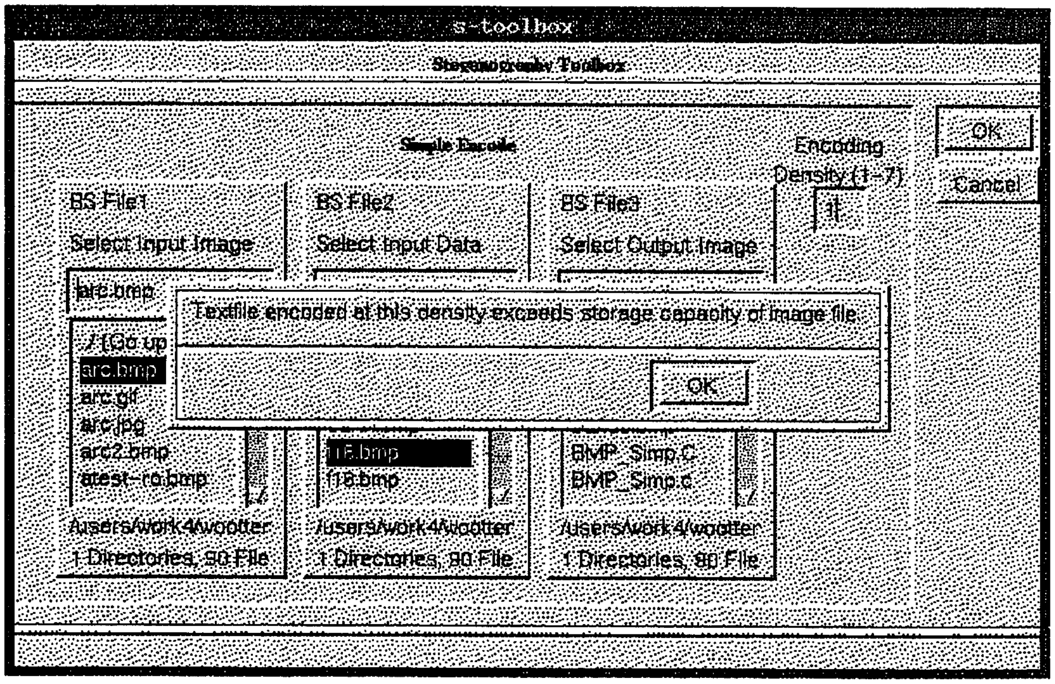
Figure 5-9. *bs_encode()* Getting Forked

```
 5657 co S    0:00 xclock -analog -bg MistyRose -fg DarkSlateBlue -hd black -g
 5660 co IW   0:04 xbiff
 5661 co S    1:11 mwm
 5663 p0 IW   0:00 -csh (tcsh)
 5662 p1 S    0:02 -csh (tcsh)
19123 p1 S    0:00 s-toolbox
19100 p2 S    0:00 -csh (tcsh)
19128 p2 R    0:00 ps
ac8> ps
  PID TT STAT TIME COMMAND
 5637 co IW   0:00 /bin/csh /usr/bin/X11/xinit
 5643 co IW   0:00 /usr/bin/X11/xinit.exec -- /usr/bin/X11/Xsun
 5644 co S   12:23 /usr/bin/X11/Xsun:0
 5645 co IW   0:00 sh /users/work4/woottend/.xinitrc
 5657 co IW   0:00 xclock -analog -bg MistyRose -fg DarkSlateBlue -hd black -g
 5660 co S    0:04 xbiff
 5661 co S    1:11 mwm
 5663 p0 IW   0:00 -csh (tcsh)
 5662 p1 IW   0:02 -csh (tcsh)
19123 p1 S    0:04 s-toolbox
19143 p1 S    0:00 sh -c bs_encode /users/work4/woottend/thesis/suit/sparc/sr
19144 p1 R    0:04 bs_encode /users/work4/woottend/thesis/suit/sparc/src/thesi
19100 p2 S    0:00 -csh (tcsh)
19145 p2 R    0:00 ps
ac8>
```

Figure 5-9(continued). *bs_encode()* Getting Forked.

5. Proper Image/Datafile Ratio

The only menu selections that represent this case are the encoding selections Simple Replace and RGB Vector encode. They are very important test cases how-the input datafile could get truncated without such an error raised. The test plan was to attempt to encode large data files into small image files to raise the error information box. Figure 5-10 applies.



Textfile encoded at this density exceeds storage capacity of image file.

Figure 5-10. Datafile Exceeds Image Capacity.

6. Widget Organization

Screen Layout is classified as an inspection category of goal.This ensures no additional widgets have been forgotten in the module, perhaps formatted so that it remains behind some other widget. The source code is balanced against the visual representation for this.

7. Widget/Window Activation and Destruction

This is an execution goal class. Do the windows appear and disappear at the right time? Table 5-1 summarizes the tests.

Table 5-1. Widget/Window Activation and Destruction.

Goal	Widget	Action	Stimulus	Expected
1	main menu	display	start-up	display
3	File menu	displays submenu	mouse button	display submenu
4	Encode menu	displays submenu	mouse button	display submenu
5	Extract menu	displays submenu	mouse button	display submenu
6	Display menu	displays submenu	mouse button	display submenu
7	Help menu	displays submenu	mouse button	display submenu
10	Delete dialog box	displays confir-mation. box	mouse on OK	displays confir-mation. box
12	Delete dialog box	clear dialog box	mouse on Cancel	clear dialog box
16	Convert dialog box	clear dialog box	mouse on Cancel	clear dialog box
17	Exit	clear window, ter-minate program	mouse on Exit	clear window, ter minate program
21	Simple replace Dialog Box	clear dialog box	mouse on Cancel	clear dialog box
25	RGB Vector dia-log box	clear dialog box	mouse on Cancel	clear dialog box
29	Simple Extract dialog box	clear dialog box	mouse on Cancel	clear dialog box
33	RGB Vector Extract dialog box	clear dialog box	mouse on Cancel	clear dialog box
37	Display Image dialog box	clear dialog box	mouse on Cancel	clear dialog box
38	Display Image dialog box	display XV panel alone	mouse on OK	display XV panel alone
42	Generate Diff Image dialog box	clear dialog box	mouse on Cancel	clear dialog box
46	View Histogram	clear dialog box	mouse on Cancel or OK	clear dialog box

Table 5-1. Widget/Window Activation and Destruction.

Goal	Widget	Action	Stimulus	Expected
49	specific Help Panel	display	topic string passed to Get-HelpTopic	display
50	Browse Help dialog box	clear dialog box	mouse on Cancel	clear dialog box
51	About Help information	clear dialog box	mouse on OK	clear dialog box

F. CONCLUSIONS

It becomes clear early on in the testing battery that the more features that are added to a program exponentially complicates the test plan.Two surprises did show up as part of the testing: though no expressed exception handler was made for a failed "rm" UNIX command or the ImageMagick "convert" command, both returned 1 from the stimulus of insufficient parameters, which is the way the author intended.

VI. SUGGESTIONS FOR FUTURE DEVELOPMENT

A. A SUMMARY ADMONITION

In the introductory chapter to this thesis, steganographic techniques during the previous millenia were described. A by-product of the recent explosive availability of automated computing devices has provided the necessary medium for a resurgent and rapidly-blooming interest in digital media steganography. The state of research in modern steganography applications is not mature; only recently have formal conferences on the topic been established, as with Anderson (1996). To that end, it is important for the student researcher who elects to "carry on the torch ", to at least contemplate the suggestions of predecessors regarding useful areas for supplemental investigation.

B. SUGGESTIONS FOR FUTURE RESEARCH

1. Develop Algorithms to Encode into Other Digital Media

Digital Imagery is but one medium suitable for application of steganographic methodology. The research and implementation climate for digital audio and video files is even more rarified than that of digital imagery.

2. Investigate Human Change Detection Thresholds for Digital Media

This is important human factors research for steganography. Since legitimate scientific research is founded not only upon reproducability of experiments and statistical evaluation, tests of the effectiveness of algorithms should be developed and performed on a population sample. An immediately tenable plan for implementation using Steganography Toolbox would be to hook up a display script that presents side-by-side comparisons of either identical images or an unencoded original image with it's encoded counterpart.Test subjects could compare many images in one test session, so that sufficient numerical data could be gathered..

3. Expand Statistical Representation of Data

In its current implementation, Steganography Toolbox offers but two ways to view statistical data: Generate Diff Image and View Histogram. Other Callback functions that present useful statistics should be added.

4. Develop Algorithms That Survive File Compression

Research by Currie and Campbell (1996) discuss this area of research, but as new methods of file compression become available, new approaches to steganographic encoding and extraction will certainly become necessary. An interesting research topic related to this is the effect that passing digital data through an analog medium (such as taking a photograph of a digital image, or recording sound generated via digital audio signal onto an analog tape) has on the ability to extract encoded data.

VII. STEGANOGRAPHY TOOLBOX VERSION 1.0 USER'S MANUAL

A. INTRODUCTION

This Users manual is presented in the following format:

MAIN MENU SELECTION

Submenu Selection

What the Selection Does

How To Invoke the Selection

An explanatory figure is placed at the bottom of each section to show what each respective panel looks like. Help dialog boxes are available both on the selection dialog box panel and via the Help Browser utility.To invoke Steganography Toolbox, change to the directory in which it resides and type: *s-toolbox*. A University of Virginia banner will display for a few seconds, and a window like that of Figure 7-1 will display. Use the left mouse button to select menu panel buttons (This version does not implement hot keys).

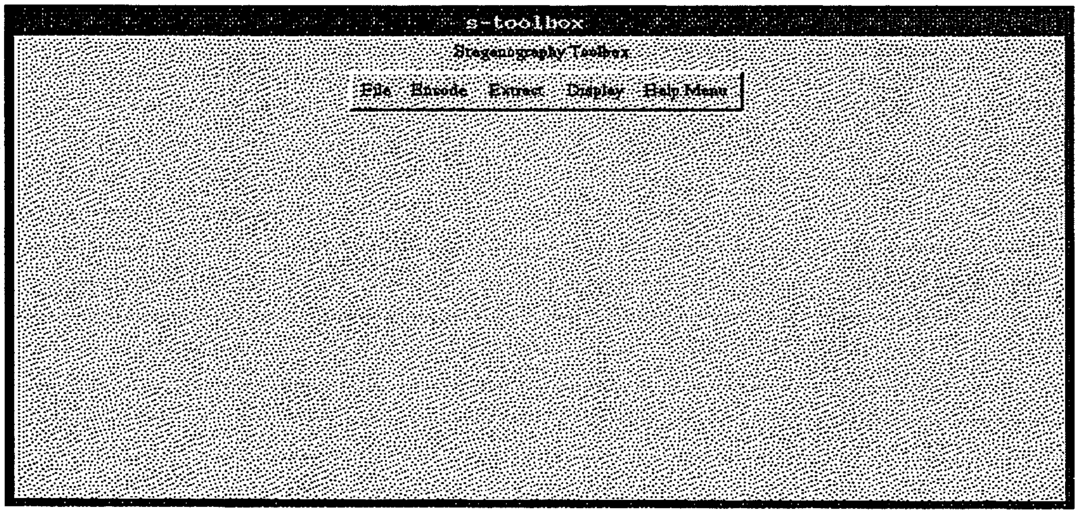


Figure 7-1. Main Menu.

B. FILE MENU SELECTIONS

1. Convert File

a. What It Does

Convert File creates a new file with the image graphic file format specified by the “convert to” filename extension. The user selects filenames to convert from with a file browser, and either types a filename (with the desired extension added) or uses the “convert to” file browser to select a file. The user may traverse the directory to select filenames.

b. How To Invoke It

From the main menu select File, and then select Convert File format the submenu choices (see Figure 7-2). A panel will display like the one shown in Figure 7-3. either type in the filename of the desired “convert from” (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Enter a “convert to” file name by typing it in the “convert to” type-in box or selecting the file from the scrollable file browser. Be sure the suffix of the filename represents the desired file format, since this is how the utility knows what conversion to perform. Select OK to run the file conversion utility, or Cancel to abort. If help is desired, press the help button.

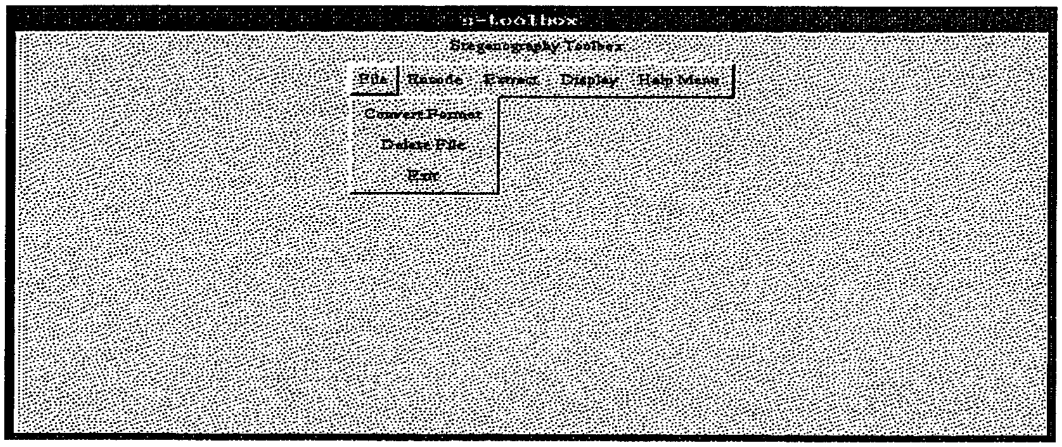


Figure 7-2. File Submenu.

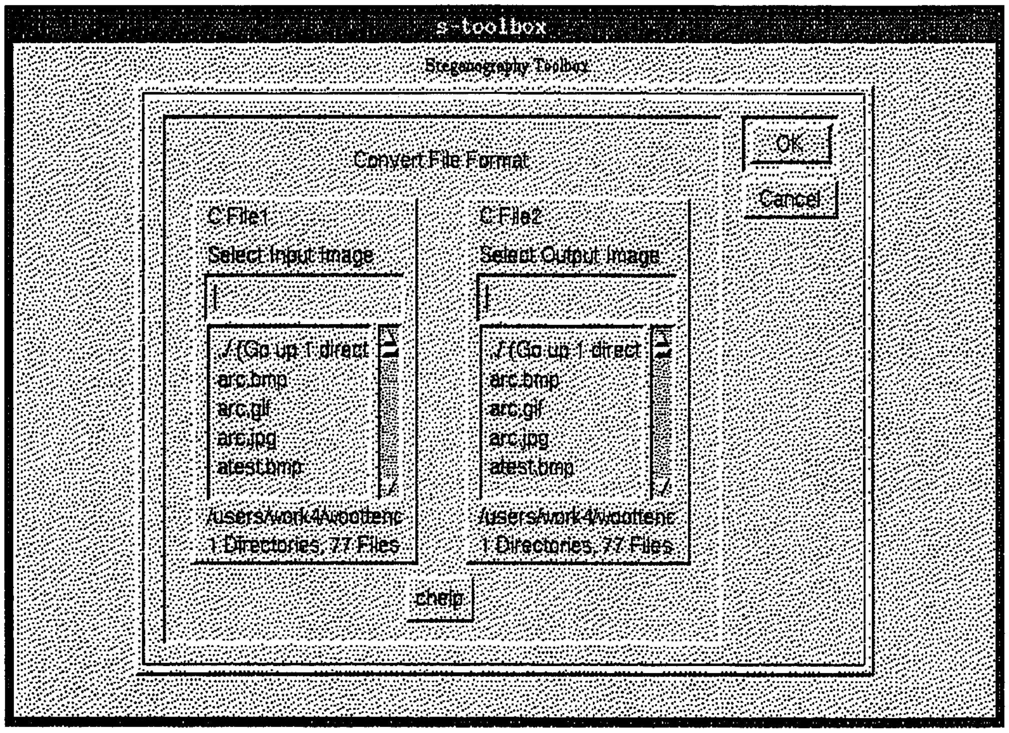


Figure 7-3. Convert File panel.

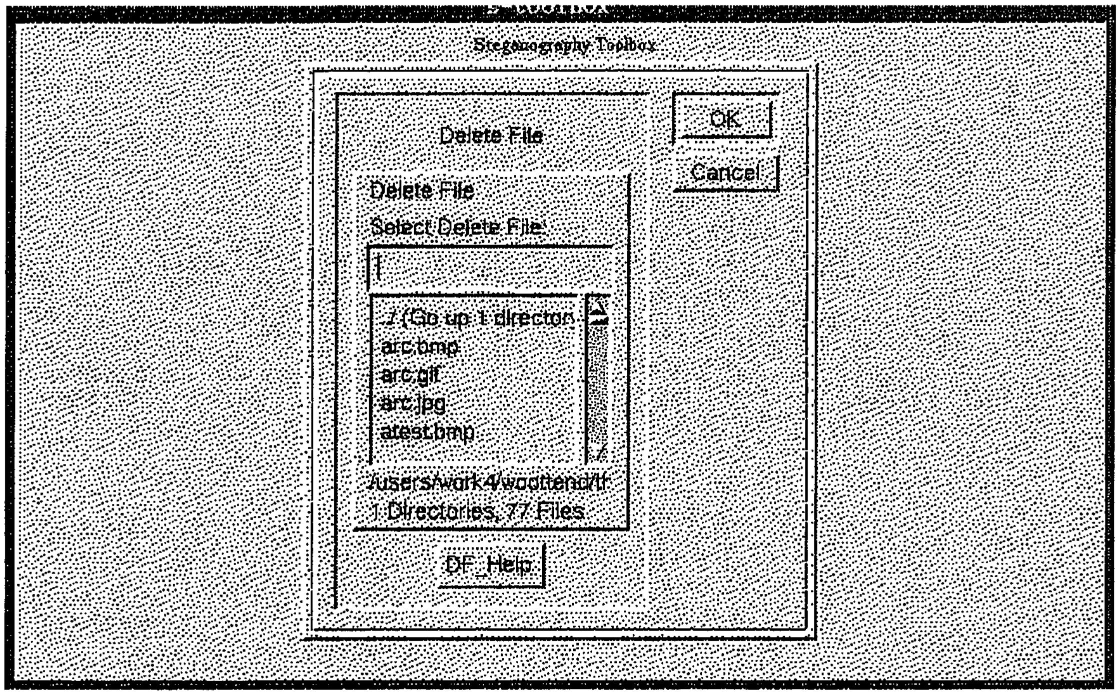
2. Delete File

a. What It Does

Delete file removes a file of the user's choosing It prompts the user with a "Are you sure" query prior to removing the file.

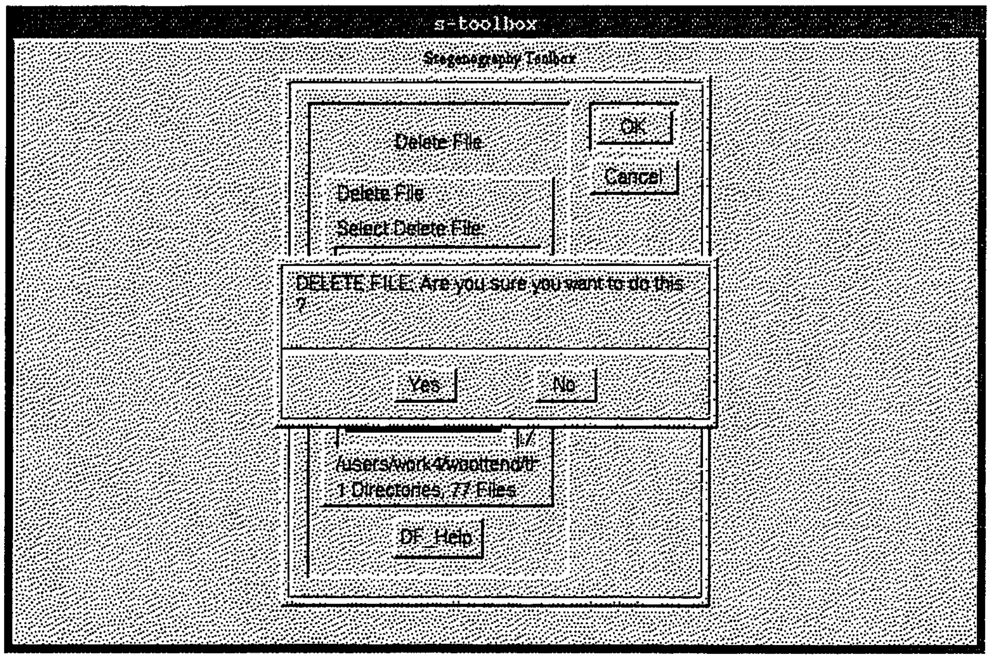
b. How To Invoke It

From the main menu select File, and then select Delete File from the submenu choices (see Figure 7-2). A panel will display like the one shown in Figure 7-4. Either type in the filename of the desired "Delete File" (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Select OK to delete the file, or Cancel to abort. If OK is selected, than a query dialog box appears asking "Are you sure?" See Figure 7-5 to see what the "Are you sure" query box looks like.Press OK to delete the file or Cancel to abort. If help is desired, press the help button



```
../ (Go up 1 director
arc.bmp
arc.gif
arc.jpg
atest.bmp
/users/work4/woottend/tr
1 Directories, 77 Files
```

Figure 7-4. Delete File Panel.



DELETE FILE: Are you sure you want to do this
?

```
/users/work4/woottend/tr
1 Directories, 77 Files
```

Figure 7-5. "Are You Sure?" Prompt.

3. Exit

a. What It Does

Exit closes any active windows and terminates the graphic user interface. Since Toolbox operations are run as background processes, any process in mid execution will continue to run until either it terminates by itself, or the user specifically terminates it. As an example, the graphics viewer XV will continue to run even after exiting Steganography Toolbox and may be closed manually from within XV.

b. How To Invoke It

From the main menu select File, and then select Exit from the submenu choices (refer back to Figure 7-2). If help is desired, select the Help topic in the Help Browser.

C. ENCODE MENU SELECTIONS

1. Simple Replace

a. What It Does

Simple Replace is a method for encoding data into Microsoft Bitmap (.BMP) format images; it is found in the Encode submenu (Figure 7-6). Selecting Simple replace presents a dialog box which, when filled with valid entries, opens the indicated input image and data files, opens an output encoded image file and performs simple replacement steganographic encoding with the two input files. A "steg header" is appended onto the regular bitmap header, namely the size of the data file (in bytes) and the encoding density. In Simple steganographic encoding the least significant bits of each byte of image data is stripped and replaced with the commensurate number of data bits. The user specifies the number of least significant bits to encode by entering an "Encode Density". Valid values are between one and seven. Once the entire data file has been encoded into the image, all files are closed and the program terminates.

b. How To Invoke It

From the main menu select Encode, and then select Simple Replace (.BMP) from the submenu choices (see Figure 7-6). A panel will display like the one shown in Figure 7-7. Either type in the filename of the desired file (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Do this for "Select Input Image", "Select Input Data", and "Select Output Image". Enter an encode density by typing an integer value between one and seven into the type-in box labeled "Encoding Density (1-7)". Select OK to run the simple replacement encoding utility, or Cancel to abort. If help is desired, press the help button.

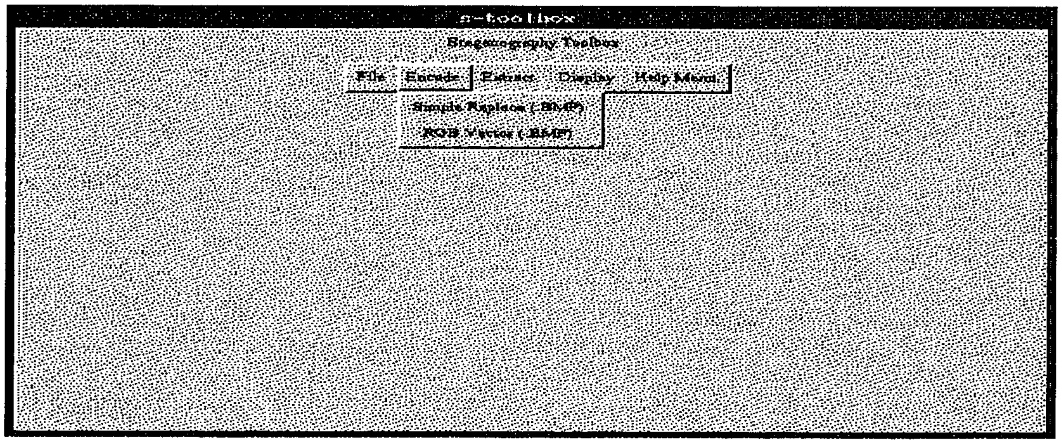


Figure 7-6. Encode Submenu.

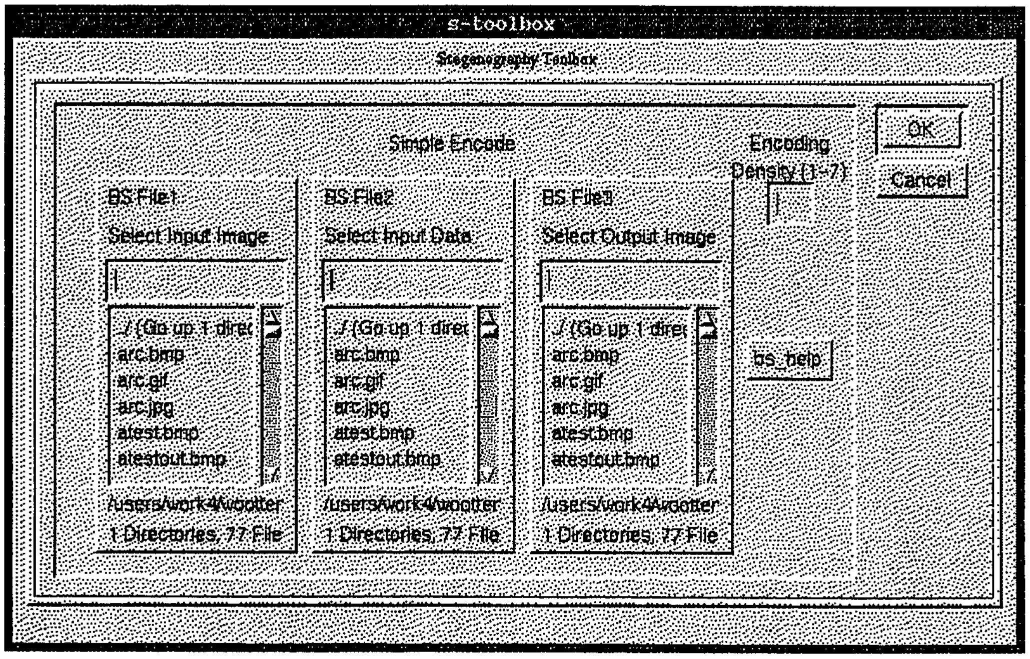


Figure 7-7. Simple Encode Panel.

2. RGB Vector

a. *What It Does*

RGB Vector invokes the RGB Vector encoding program developed by Campbell and Currie. The algorithm treats each three-byte triple as a a vector in RGB colorspace. The length of the vector is changed such that it's modulus mod 62 is changed depending on the value of the bit to be encoded. Currie and Campbell state, "Because the vector's direction is unmodified, the relative sizes of the color channel values are preserved. An input image and data file are required, as is an output data file.

b. *How To Invoke It*

From the main menu select Encode, and then select RGB Vector (.BMP) from the submenu choices (see Figure 7-6). A panel will display like the one shown in Figure 7-8. Either type in the filename of the desired file (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Do this for "Select In Image", "Select In Data", and "Select Out File". Select OK to run the RGB Vector encoding utility, or Cancel to abort. If help is desired, press the help button.

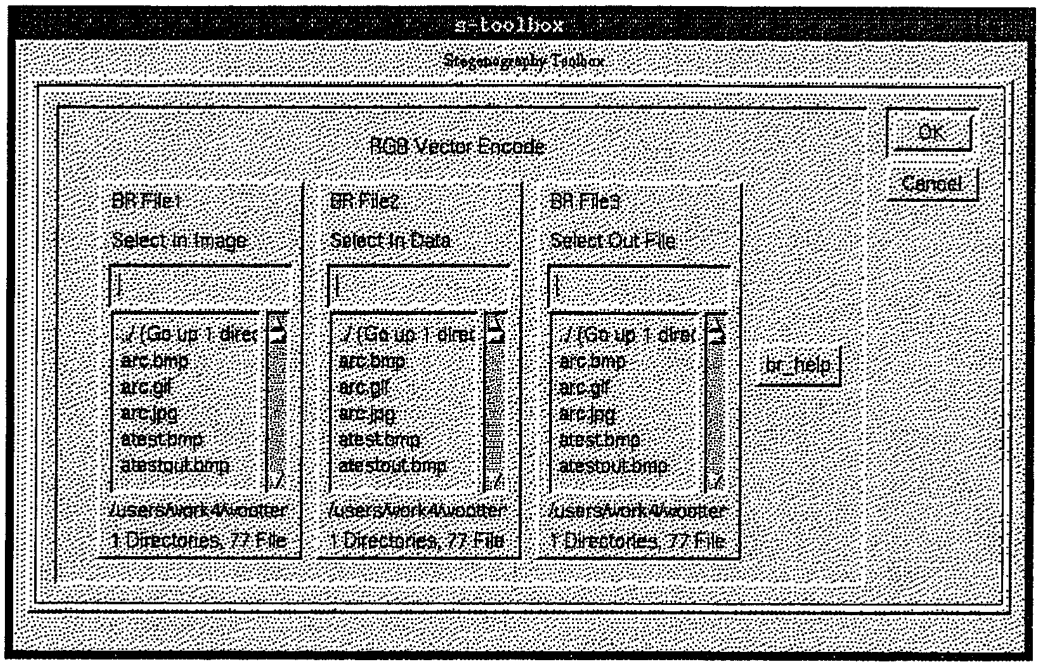


Figure 7-8. RGB Vector Panel.

D. EXTRACT MENU SELECTIONS

1. Simple Extract

a. What It Does

Simple Extract is a method for retrieving data from Microsoft Bitmap (.BMP) format images that were encoded using the simple replacement algorithm created by Currie and Campbell (1996); it is found in the Extract submenu (Figure 7-9). Selecting Simple Replace Extract presents a dialog box which, when filled with valid entries, opens the indicated steganographically encoded input image file, opens an output data file and performs bytewise extraction on the input image file, piping the output to the selected output data file. A "steg header", an additional header appended onto the regular bitmap header, which provides namely the size of the data file (in bytes) and the encoding density, is first read from the input image file. Once the entire data file has been extracted from the image, both files are closed and the program terminates.

b. How To Invoke It

From the main menu select Extract, and then select Simple Replace Extract (.BMP) from the submenu choices (see Figure 7-9). A panel will display as shown in Figure 7-10. Either type in the filename of the desired file (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Do this for "Get Input Image" and "Get Output Data". Select OK to run the simple replacement encoding utility, or Cancel to abort. If help is desired, press the help button.

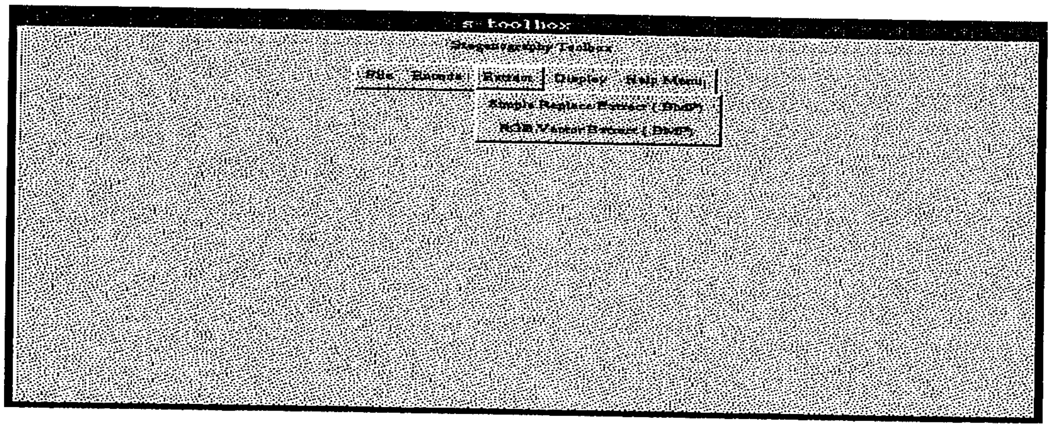


Figure 7-9. Extract Submenu.

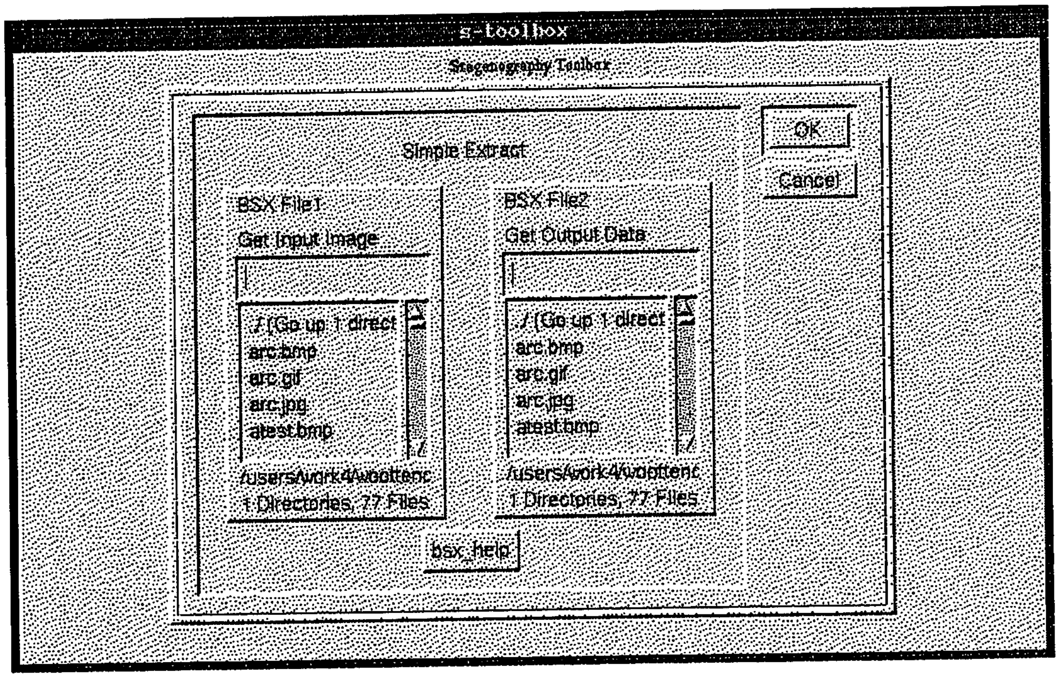


Figure 7-10. Simple Extract Panel.

2. RGB Vector Extract

a. What It Does

RGB Vector Extract is a method for retrieving data from Microsoft Bitmap (.BMP) format images that were encoded using the RGB Vector extraction algorithm created by Currie and Campbell (1996); it is found in the Extract submenu (refer back to Figure 7-9). Selecting RGB Vector Extract presents a dialog box which, when filled with valid entries, opens the indicated steganographically encoded input image file, opens an output data file and performs bytewise extraction on the input image file, piping the output to the selected output data file. No "steg header" is appended onto the regular bitmap header, and, in this version, the user must supply the original encoded data file size. This information is displayed on the terminal window from which Steganography Toolbox was invoked, when the RGB Vector encoding selection is invoked. Once the entire data file has been extracted from the image, both files are closed and the program terminates.

b. How To Invoke It

From the main menu select Extract, and then select RGB Vector Extract (.BMP) from the submenu choices (see Figure 7-9). A panel will display as shown in Figure 7-11. Either type in the filename of the desired file (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Do this for "Select Input Image" and "Select Output Data". Select OK to run the RGB Vector extraction utility, or Cancel to abort. If help is desired, press the help button.

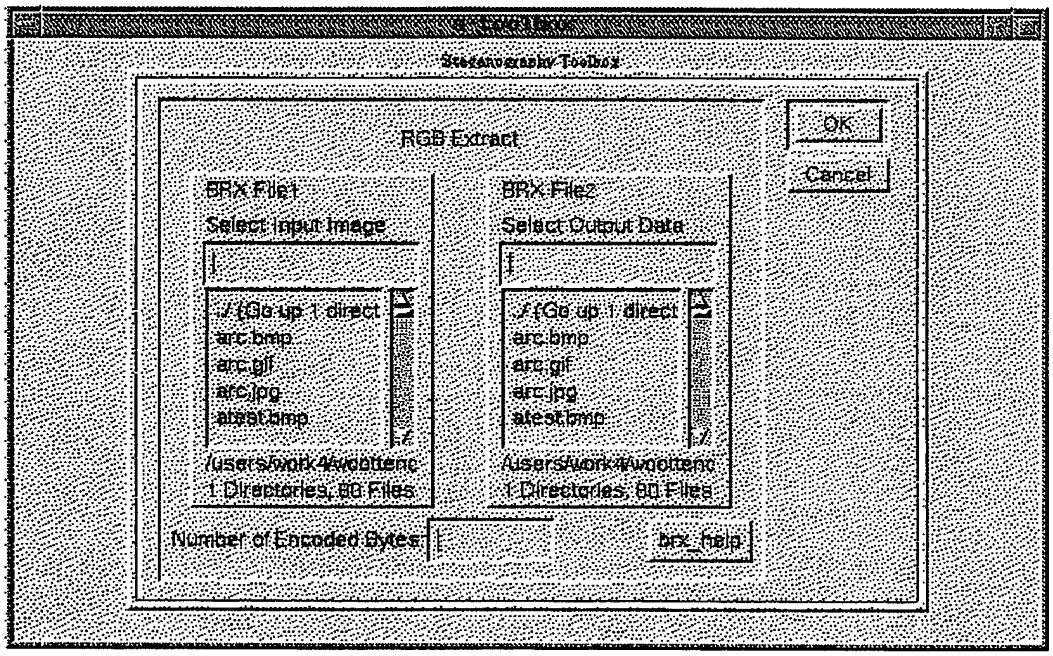


Figure 7-11. RGB Vector Extract.

E. DISPLAY MENU SELECTIONS

1. Display Image

a. What It Does

The Display Image submenu selection makes a *system()* call to XV version 3.00. All of XV's functionality is available to the user, and it is run as a background process, to the user can display as many images as s/he desires, while using other Steganography Toolbox utilities.

b. How To Invoke It

From the main menu select Display, and then select Display Image from the submenu choices (see Figure 7-12). A panel will display las in Figure 7-13. Either type in the filename of the desired "Select Display Image" (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Select OK to display the file, or Cancel to abort. If help is desired, press the help button

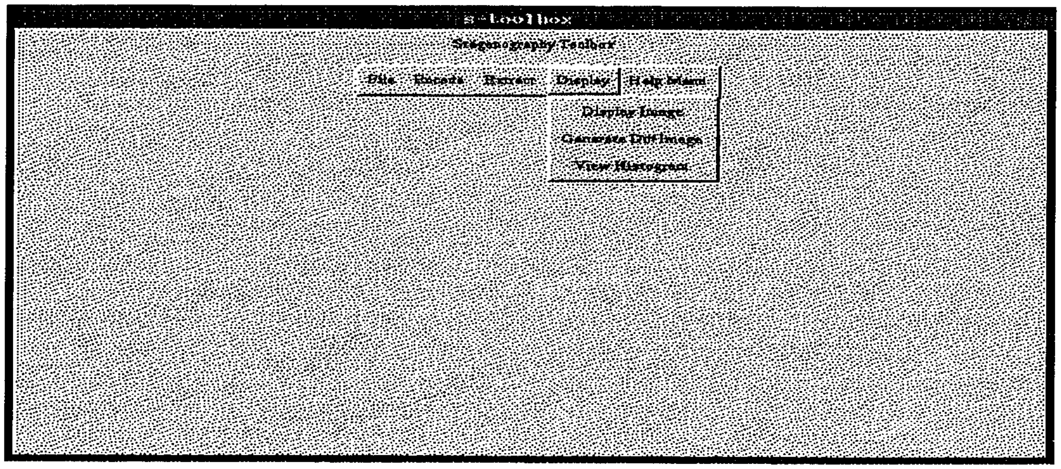


Figure 7-12. Display Submenu.

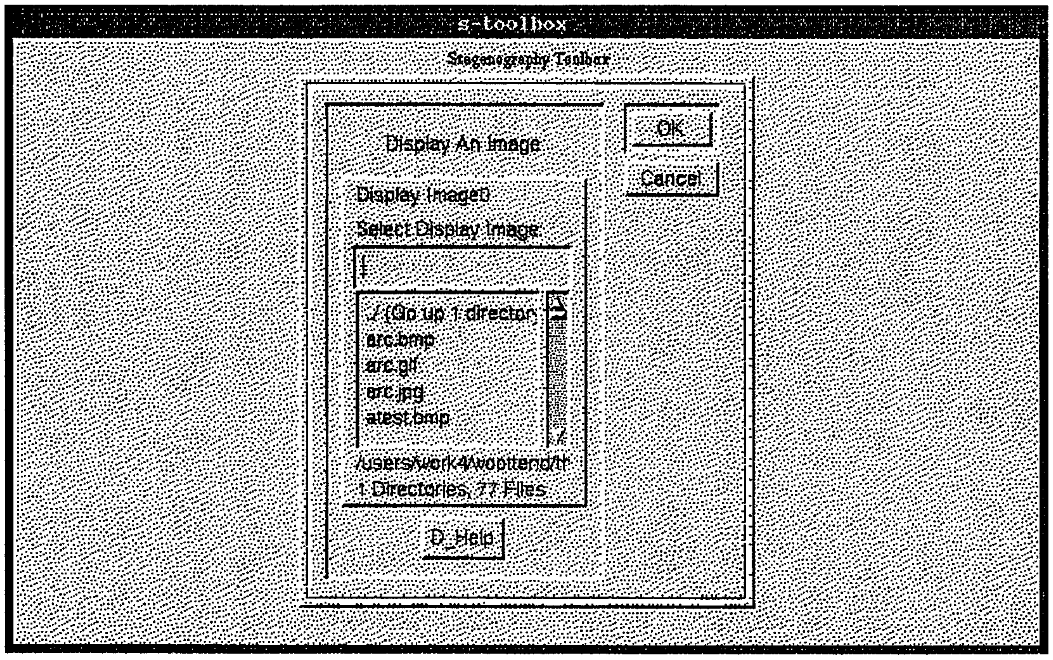


Figure 7-13. Display An Image Dialog Box.

2. Generate Diff Image

a. What It Does

The Selection Generate Diff Image accepts a Microsoft Windows Bitmap (.BMP) image format file, and its steganographically encoded equivalent, and creates an absolute difference image. It invokes a *system()* call to a program created by Currie and Campbell (1996). The program opens the two input image files, compares the header information to ensure they are the equivalent image, and makes a bytewise comparison of the image data. Brightness values are related to the absolute bit difference for the three-byte triple, representing the RGB brightness components of the pixel. Every three bits of change represents a different brightness level in the output image. Additionally, a file named "hist.dat" is opened, and the contributing filenames, numbers of pixels in each three-bit change histogram bin, and total number of pixels that were different are written to the file. Then all open files are closed and the program terminates.

b. *How To Invoke It*

From the main menu select Display, and then select Generate Diff Image from the submenu choices (refer back to Figure 7-12). A panel will display as shown in Figure 7-14. Either type in the filename of the desired file (including the full path if not in the current directory), or use the scrollable file browser box to select the file. Do this for "Select First BMP Image", "Select Second BMP Image", and "Name Diff BMP Image". Select OK to run the diff image generation utility, or Cancel to abort. A file "hist.dat" will also be created in the local directory, or overwritten if it already exists.If help is desired, press the help button.

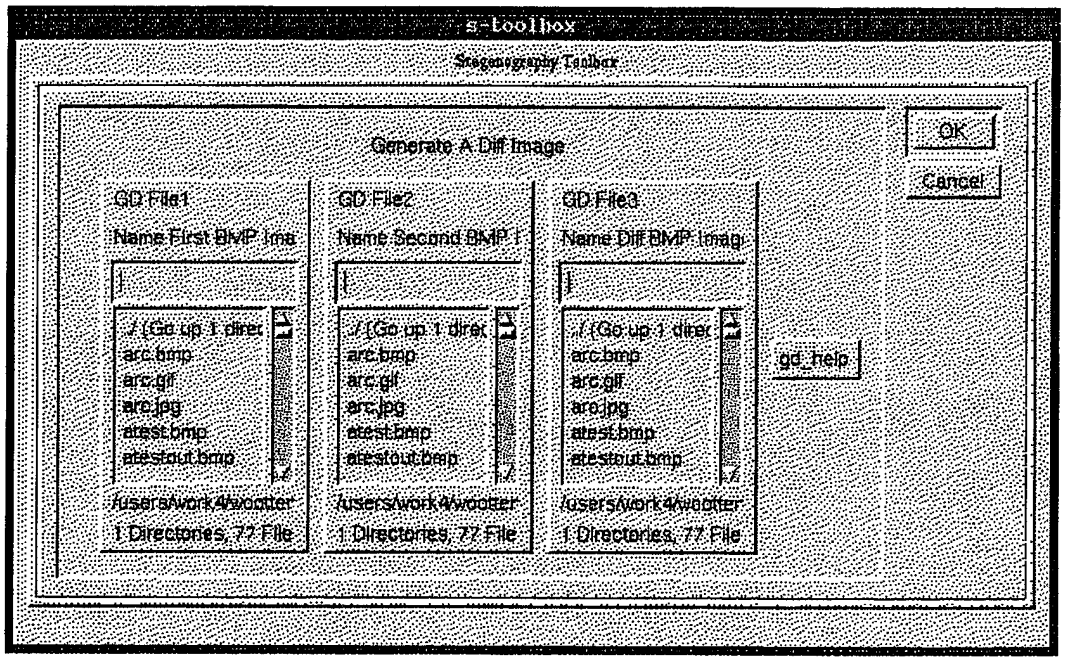


Figure 7-14. Generate Diff Image Dialog Box

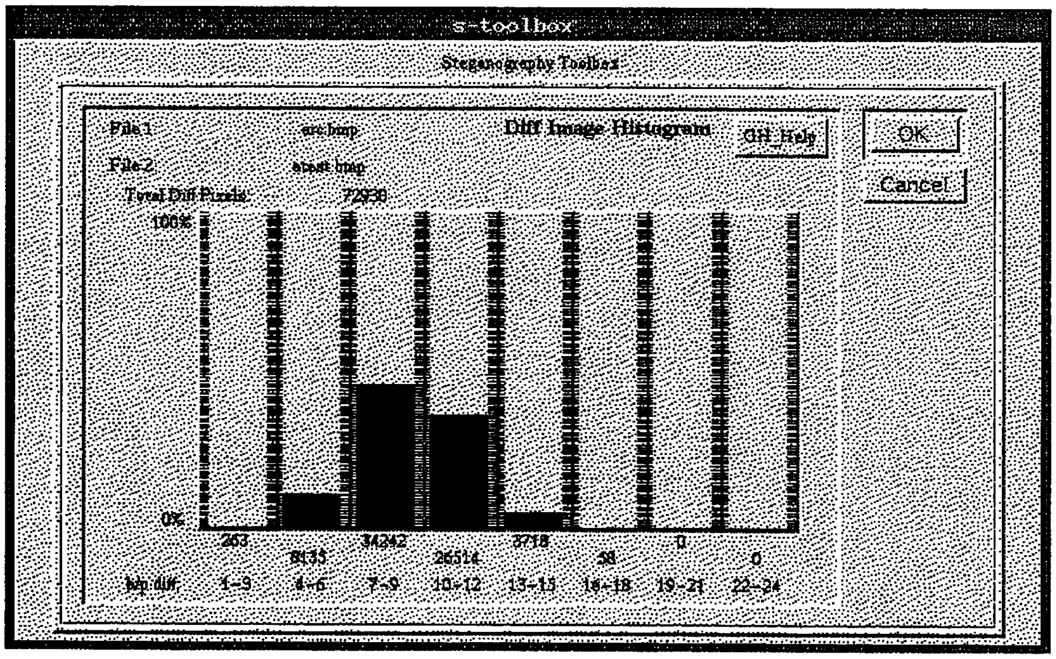
3. View Histogram

a. *What It Does*

View Histogram displays a dialog box containing a diff image histogram based on data contained in "hist.dat". This histogram data file is created when Generate Diff Image is invoked (see the above section on Generate Diff Image). The dialog box presents the names of the contributing image files, the histogram values of each bin, and the total number of pixels altered.

b. How To Invoke It

From the main menu select Display, and then select View Histogram from the submenu choices (refer back to Figure 7-12). A panel will display like the one shown in Figure 7-15. When done viewing the histogram select OK or Cancel (since the SUIT command *SUIT_createOKCancelDialogBox()* was used to contain all the histogram scroller widgets and the dialog box performs no other callback, REPLY_OK and REPLY_CANCEL contain a NULL callback response.(See Chapter IV for more information regarding SUIT program structures.).If help is desired, press the help button.



File 1 src.bmp
File 2 stest.bmp
Total Diff Pixels: 72930

```
bmp diff:  1-3   4-6   7-9    10-12  13-15  16-18  19-21  22-24
           263   8135  34242  26514  3718   58     0      0
```

Figure 7-15. View Histogram Dialog Box.

F. HELP MENU SELECTIONS

1. Browse Help

a. What It Does

The Browse Help selection presents a scrollable textbox, which provides help topics the user may select. The displayed Help dialog boxes typically list the function and proper use of a selection.

b. How To Invoke It

From the Main Menu, choose Help Menu. A submenu like that in Figure 7-16 will appear. Choose Browse Help from the available options. A dialog box will be presented to the user like that of Figure 7-17, which contains a scrollable list of help topics. Highlight the desired topic and select OK to view the help topic dialog box, or Cancel to abort.

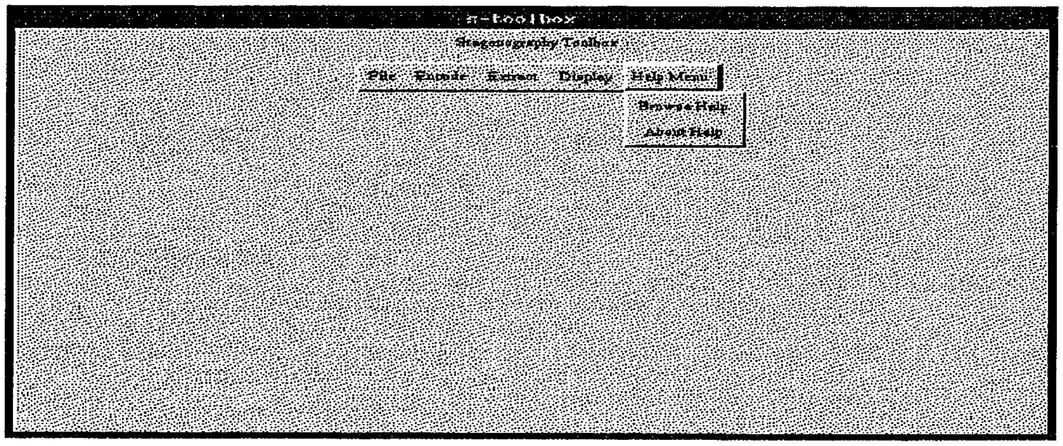


Figure 7-16. Help Submenu.

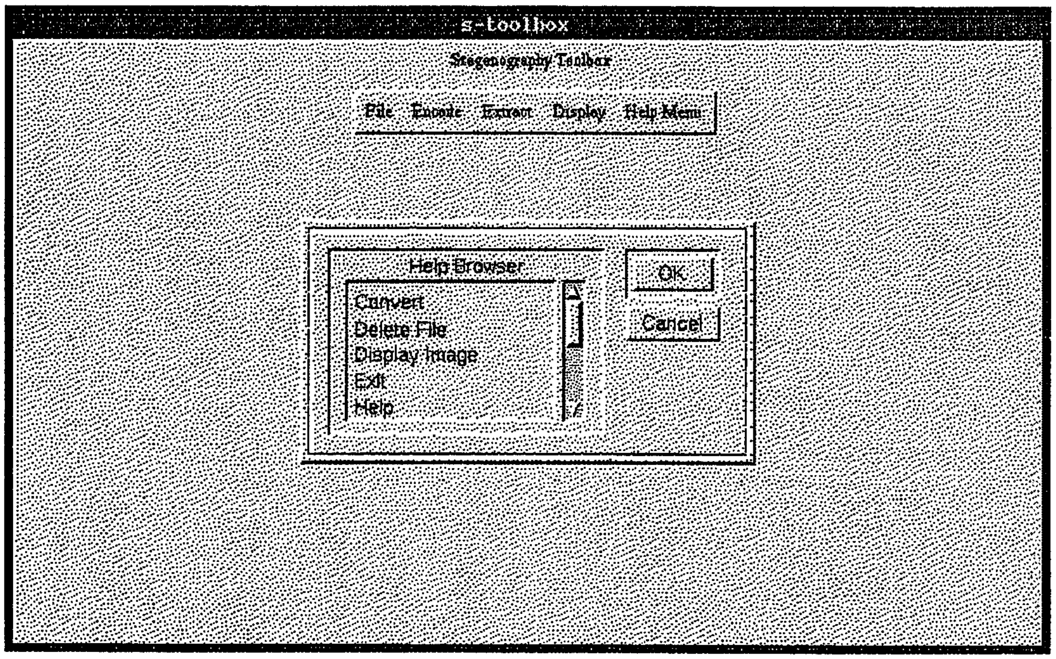


Figure 7-17. Help Browser Dialog Box.

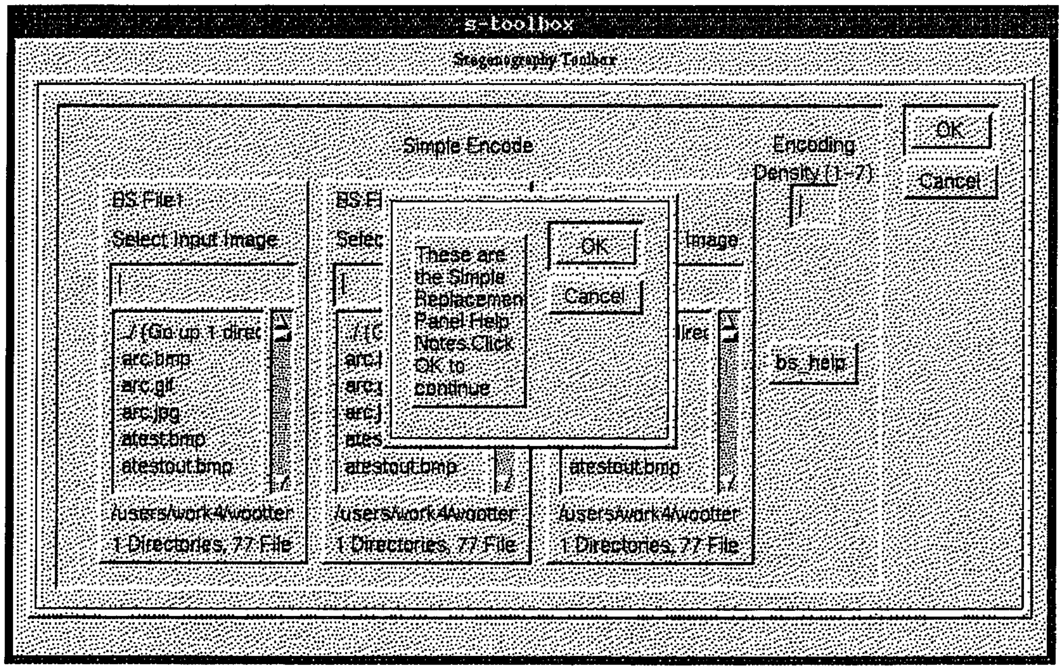
2. Help From Within Toolbox Panel

a. What It Does

Individual help topics related to specific Steganography Toolbox help panels are also accessible from the respective dialog box panel, as shown in Figure 7-18.

b. How To Invoke It

To access the help topic dialog box press the help button on the respective panel for which help is desired. Press OK on the presented dialog box to view successive help panels or select Cancel to abort.



These are the Simple Replacement Panel Help Notes.Click OK to continue

Figure 7-18. Help Dialog Box From Within a Steganography Toolbox Dialog Box.

APPENDIX A. SOFTWARE DESIGN

1. Scope

This appendix summarizes the functional requirements for Steganography toolbox.to include, system objectives and process specifications of its components. The Steganography Toolbox is a software package that shall assist the user in the study and development of steganography algorithms. It shall provide tools that display images and convert them from one image format to another, encode data into and extract data from images using steganographic algorithms, and generate and display useful statistics.

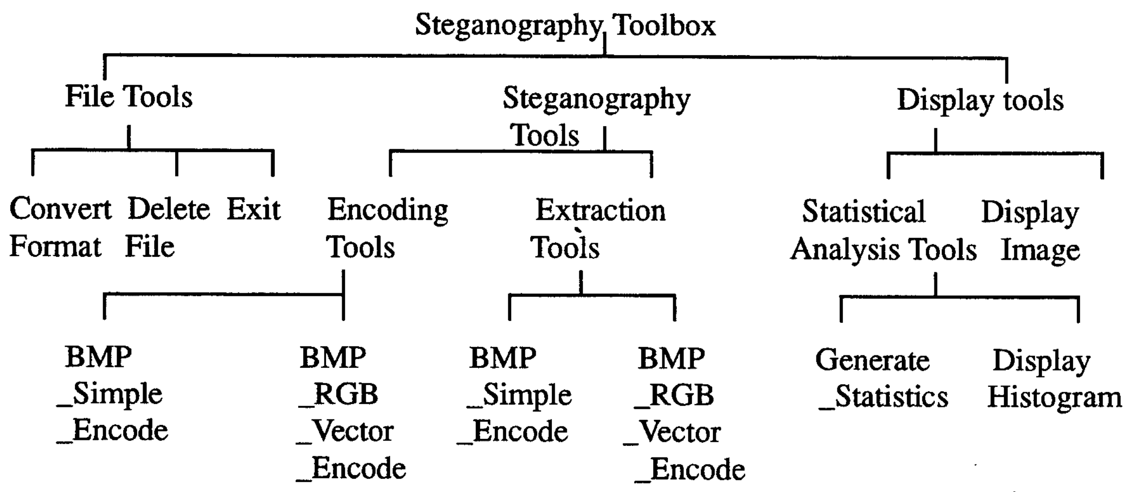


Figure A-1. Functional Diagram

Objectives are as follows:

1) Convert digital images from one digital image format to another

1.1) Convert .BMP image files to JPEG image format

2) Encode digital data into digital imagery

2.1) Encode digital data into a .BMP image using simple replacement

2.2) Encode digital data into a .BMP image using RGB Vector method

3) Extract previously encoded digital data from digital images

3.1) Extract digital data from .BMP images encoded using simple replacement method

3.2) Extract digital data from .BMP images encoded using RGB Vector method

4) Display digital images on the terminal screen

5) Generate statistics

6) Display statistics in a graphically coherent fashion

6.1) Display absolute difference image of two input images

7) Delete files

2. Hardware, Software, and User Interface

This system is capable of running in any X windows-compatible environment. Since the initial implementation of this system centers on tools to steganographically encode digital imagery (vice audio media) the workstation should have a 21" monitor to display and compare images. A 3 1/2" high density floppy drive should be available to copy image files.

Aside from the code that actually encodes and extracts data from image files, and that which creates an absolute difference image of 2 input images, all other tools in this toolbox are available as freeware on the Internet. Image display functions are performed by XV version 3.0, which can be downloaded as a 2.2Mb gzipped tar file. Image file format conversion is performed by ImageMagick version 3.7 which is a 700 Kb gzipped tar file. Finally, the component of image file format conversion which performs JPEG compression is performed by the International JPEG Group's compression utility version 6 which is available as a 531 Kb gzipped tar file. ImageMagick and the IJG jpeg conversion utility come as source code with Makefiles as part of their tar file, so the platform this system is placed on must also have available a C or C++ compiler.

The graphic user interface is itself a software tool, for the development of gui interfaces. It is called "Simple User Interface Toolkit", or SUIT, and it is available as a 2.1 Mb zipped tar file. This file is platform-specific, and is configured for Sun3, Sun 4 (Sparcstation), SGI IRIS, DEC, HP, IBM PC, and Macintosh.

3. Major Design Constraints and Limitations

- Platform of use is limited to those described as compatible with SUIT.
- Each of the freeware tools must be installed separately;
- Installation of IJG JPEG compression utility must precede installation of ImageMagick, since ImageMagick looks for it during it's installation.
- If the workstation the Steganography Toolbox is installed on has no Internet access, some Input/Output device with a capacity larger than the largest tar file.

4. Reference Documents

- Simple User Interface Toolkit Reference Manual
- ImageMagick version 3.7 README file
- IJG JPEG compression utility version 6 README file
- XV version3.00 Reference Manual

5. Process Specifications Outline

- Process Specification 1: Convert_Format
- Process Specification 2: BMP_Simple_Encode
- Process Specification 3: BMP_Simple_Extract
- Process Specification 4: BMP_Compare_Images
- Process Specification 5: BMP_RGB_Encode
- Process Specification 6: BMP_RGB_Extract
- Process Specification 7: Delete_File
- Process Specification 8: Display_Image
- Process Specification 9: Generate_Diff_image
- Process Specification 10: View_Histogram

6. Process Specifications

Process Specification 1

Name: Convert_Format

Requirement Satisfied: 1.1

Description: Convert_Format is invoked with a system call (*system()*) to the International JPEG Group program *convert()*, followed by the arguments *infile* and *outfile*. Convert restructures the image format recognized by the *infile* suffix to the format indicated by the *outfile* format suffix, which may include data compression.

Inputs:

infile: filename of the input image of the format recognized by convert

Outputs:

outfile: filename of output image in converted format described by *outfile*'s suffix

Errors:

1) input image filename not found

2)unrecognized input image filename suffix

3) no output filename provided

4) unrecognized output filename suffix

Process Specification 2

Name:

BMP_Simple_Encode

Requirement Satisfied:

2.1

Description:

BMP_Simple_Encode opens *infile textfile* and *outfile*. It reads the header information from *infile* and writes it to *outfile*. It then compares the size of the input file listed in *infile*.headersize to the amount of bytes it requires to encode all of the *textfile* divided into density-bit segments. Provided there is room, it writes the *textfile* size (in bytes) and density to *outfile*, called steghead1 and steghead2, respectively. These values are themselves encoded into the least significant 2 bits of the 24 bytes following the original *outfile* header. BMP_Simple_Encode then reads a textbyte, and if it is not the *textfile* end-of -file marker, it reads an image byte, register-shifts out the least significant density bits in it and logically AND's it to the most significant density bits of the textbyte. While it has textbits left to encode it will continue. If there are fewer than density bits remaining in textbyte, it will read another byte of the *textfile* and concatenate the byte to the remaining bits of textbyte. Once an image byte is encoded it is written to *outfile*. Once all of the *textfile* has been encoded into to *outfile*, *infile*, *textfile*, and *outfile* are closed and control is passed to the main program.

Inputs:

infile: input image in .BMP image format.

textfile: input data file

density: integer between 1 and 7, which describes the number of bits to encode per image byte

Outputs:

outfile: output image in .BMP image format

Errors:

1) *infile* is too small to encode all of *textfile* at *density* density

2) *infile* not found

3) *infile* not in .BMP format

Process Specification 3

Name:

BMP_Simple_Extract

Requirement Satisfied:

3.1

Description:

BMP_Simple_Extract opens *stegfile* and *data_out_file*. It reads the *stegfile* standard .BMP file header, and then from the next 24 bytes reads the least 2 significant bits of each byte. these are concatenated to form the textfile size that was originally encoded into *stegfile*, and the density (bits per byte) recorded. While the number of bytes written to *data_out_file* are less than *data_out_file* size it reads a *stegfile* byte; reads the least density significant bits of that byte, and writes it to a temporary register which concatenates and holds these segments. once the size of the bitstring in the register is greater than a byte, it writes that byte of the register to *data_out_file* and increments a counter that tracks the current number of bytes written to *data_out_file*. When the *data_out_file* size is reached, *stegfile* and *data_out_file* are closed and control is returned to the main program.

Inputs:

stegfile: input .BMP image filename with encoded data

Output:

data_out_file: output file which contains extracted data

Errors:

1) badly formed header

2) nonexistent input filename

3) no output filename indicated

4) unrecognized image file format

Process Specification 4

Name:

BMP_Compare Images

Requirement Satisfied:

6.1

Description:

BMP_Compare_Images opens 2 input image files of .BMP format and a *mapfile* .BMP image. it reads the input image file headers to verify if they are the same size. If they are it reads 3 bytes of each input file and compares each respective byte (these are the blue, green, and red component brightness values of each pixel.)It takes the average of the 3 component differences. If the absolute difference is 0, a white pixel is written to the *map-file*(3 bytes of 0xFFFF). If the averaged absolute difference is greater than 0, it increments a counter that corresponds to the number of pixels that differ within the specified range of

bits. The ranges are 1-3 bits difference, 4-6,7-9, and
so on up to 22-24 bits difference. If there is a difference in any of the blue, green or red brightness values, a dark pixel is written; the more bits difference, the darker the pixel. When all the input files' pixels have been compared, the counter values are displayed to standard output, all opened files are closed, and control is passed back to the main program.

Inputs:

file1,file2: input manges in .BMP format

Output:

mapfile: output absolute average difference image in .BMP format

Errors:

1) Bad value in Switch Statement (out of range error)

2) Bitmaps are of different size

3) One or both of the images is not in .BMP format

4) Input image filename not found

Process Specification 5

Name:

BMP_RGB_Encode

Requirement Satisfied:

2.2

Description:

BMP_RGB_Encode opens *infile textfile* and *outfile*. It reads the header information from *infile* and writes it to *outfile*. It then compares the size of the input file listed in *infile*.headersize to the amount of bytes it requires to encode all of the *textfile*. If there is room, BMP_Simple_Encode then reads a textbyte, and if it is not the *textfile* end-of-file marker, it reads 3 image bytes (a pixel), and converts it to a vector value using the three bytes as vector components. This distance modulo 62 is calculated. The remainder is the part of the value that gets embedded. If the text bit to be embedded is a zero, the remainder of the vector shall be assigned 17, which is the middle of the lower half of the remainder range.

If the text bit is a one, the remainder is assigned the value 47. Five pixels are embedded with the same bit value, but the distance between encoded pixels shall vary,based on the skip distance. This skip distance is calculated by dividing the number of image pixels by the product of the redundancy value (in this case, five) and the number of bits in the text-file., then While it has textbits left to encode it will continue to embed pixels. Once an image byte is encoded it is written to *outfile*. Once all of the *textfile* has been encoded into to *outfile*, *infile*, *textfile*, and *outfile* are closed and control is passed to the main program.

Inputs:

infile: input image in .BMP image format.

textfile: input data file

Outputs:

outfile: output image in .BMP image format

Errors:

1) *infile* is too small to encode all of *textfile*

2) *infile* not found

3) *Outfile* not found.

Process Specification 6

Name:

BMP_RGB_Extract

Requirement Satisfied:

3.2

Description:

BMP_RGB_Extract opens *stegfile* and *data_out_file*. It reads the *stegfile* standard .BMP file header, and the long integer value textbytes, which represents the user-supplied size of the file to be extracted. An extracted-byte counter is maintained. While the current value of the extracted byte counter is less than textbytes, it will read a pixel, convert it to a vector value of the three component byte values, and evaluate the remainder modulo 62. If the remainder is closer to 17 than 47, it will write a zero to the *data_out_file*. If the remainder is closer to 47 than 17, it shall write a one to the outfile. When *data_out_file* size is reached, *stegfile* and *data_out_file* are closed and control is returned to the main program.

Inputs:

stegfile: input .BMP image filename with encoded data

Output:

data_out_file: output file which contains extracted data

Errors:

1) *Stegfile* not found

2)*Data_out_file* not found.

Process Specification 7

Name: Delete File

Requirement Satisfied: 7

Description: Delete File is invoked with a system call (*system()*), where the argument is a character string, The character string starts with the UNIX command rm, followed by a blank space. Appended to this string is the filename to be deleted. Following the filename entry and prior to the *system()* call, the user is asked via a dialog box, if he is sure he wants to do this. If "Yes" is selected, the *system()* call is invoked, and the file is deleted. If "no" is answered, the query panel and the file selection dialog box are destroyed.

Inputs:

infile: filename of the input image of the format recognized by convert

Outputs:

None.

Errors:

1) image filename not found

2) no output filename provided

Process Specification 8

Name: Display_Image

Requirement Satisfied: 4

Description: Display_Image invokes a system call (*system()*) to the XV Version 3.00 graphics file viewer followed by the argument *infile.*

Inputs:

infile: filename of the input image of the format recognized by convert

Outputs:

none

Errors:

1) input image filename not found

2)unrecognized input image filename suffix (handled by XV)

Process Specification 9

Name: Generate_Diff_Image

Requirement Satisfied: 5,6,6.1

Description: Generate_Diff_Image invokes a system call (*system()*) to *convert()*, followed by the arguments *infile1 infile2 outfile.* It first opens the files and then examines the header value *infile1.filesize* and *infile2.filesize* to confirm they are equal. If they are it then writes a Microsoft Windows Bitmap format image that shows the absolute change between *infile1* and *infile2* of each red green and blue byte, with darker pixels representing greater change in the pixel.

Inputs:

infile1,infile2: filenames of the input images of the format recognized by convert

Outputs:

outfile: filename of the output image created by *convert()*

Errors:

1) input image filename not found

2)unable to open *outfile*

Process Specification 10

Name: View_Histogram

Requirement Satisfied: 5,6,6.1

Description: View_Histogram looks for the file “hist.dat”, an input file created originally by the most recent execution of Generate_Diff_Image It reads The character strings file1 and file2 eight long integer values representing the number if changed bytes compared, divided into eight histogram bins (There are three absolute bits of change per bin value). Finally a long integer, representing the total number of bytes changed, is read. These values are then displayed in histogram format, where the histogram is normalized to 100.The filenames and total bytes difference is also displayed.

Inputs:

“hist.dat”: generated by the most recent execution of Generate_Diff_Image.

Outputs:

none

Errors:

1) *“hist.dat”* not found

APPENDIX B. SOURCE CODE

1. INTRODUCTION

Source code for Steganography Toolbox is listed in this appendix.While the current application of Steganography Toolbox includes the files:

- toolbox.c
- toolbox_help.h
- generic_steg.C
- generic.extract.C
- rgb_steg.C
- rgb_extract.C
- compare.C

the latter four files, written by Campbell and Currie(1996) and adapted for use with the graphic user interface, will not be included. The modifications made to their source code merely enabled them to be executed in the "argc, argv[]" format, and pass error condition values depending on the error..

2. TOOLBOX.C

```
/*************************************
*  toolbox.c -- By LT D. R. Wootten
*  This program displays a menu of steganography
tool options. It invokes independently executable programs through the  use    of
system() calls. In order to "hook  the programs up", they must be of the form
program_name arg1  arg2  arg3 ... where arg is some parameter it needs to use. Write the
program using "argc" and "argv[]". Then set up this file to use
it, according to the following steps:

1. Add a submenu selection, adding a button to the main menu button bar, if
necessary.

2. Write the callback to display a dialog box widget, containing all the
child widgets on a single "bulletin board" widget.

3. Have it form a character string starting with the program you wish to make the system
call to. use strcpy() to start it and strcat() to append arguments   onto it.

4. invoke system() using the string as the argument. if you want error
message panels to display if the program aborts, define an integer variable
and assign the return value (exit status) to the variable. insert exit()
```

calls in the appropriate sections of the system()-invoked program. Remember
that the value returned will be bit-shifted 8 more significant bits by the
time it gets back to this program. Use SUIT_inform() in a switch statement to display
your message.

5. Make a help panel for it and hook it up to the BrowseHelp widget found in
"toolbox_help.h"

I commented Convert() pretty heavily, and then further down only as I thought
necessary. Quite honestly, you might keep one eye looking out for a good
GUI builder that compiles in C++ (that we can get a license for). This one was public
domain freeware and free is a good thing (but not always a great thing).

```
**************************************** */
#include <stdio.h>
#include "suit.h"
#include "toolbox_help.h"




/*************CONVERT***************/
void Convert(SUIT_object menu) {

/* Declare the SUIT_objects */
   SUIT_object c_container, c_fbox1,c_fbox2,c_h_button,
      c_dBox, c_label;

/*Declare anything else you need*/
   char*  file1;
   char*  file2;
   char* c_path = " ../../../../ImageMagick/convert ";
   char   cf[256];
   int errornum; /*This gets the return value of your system() call*/


/*Bind the object names to widgets*/
   c_container = SUIT_createBulletinBoard ("C Container");
   c_fbox1 = SUIT_createFileBrowser(
      "c_barney","","C File1","Select Input Image",NULL);
   c_fbox2 = SUIT_createFileBrowser(
      "c_wilma","","C File2","Select Output Image",NULL);
   c_h_button = SUIT_createButton("chelp", HelpConvert);
```

```
/*Size the panel that will hold the widgets*/
   SUIT_changeObjectSize(c_container, 400, 340);

/* Assign widgets to the panel wigget as children*/
   SUIT_addChildToObject(c_container, c_fbox1);
   SUIT_addChildToObject(c_container, c_fbox2);
   SUIT_addChildToObject(c_container, c_h_button);
   c_label = SUIT_createLabel("Convert File Format");
/*This one lets you set the font of the text label*/
   SUIT_setFont(c_label,FONT,GP_defFont("times","bold",12));
   SUIT_addChildToObject(c_container,c_label);

/* This allows you to set the location of the widgets on the parent
  widget with x,y coordinates from 0 to 1 for the lower left
  and upper right widget corners*/
   SUIT_setViewport(c_label, VIEWPORT,
               SUIT_mapToParent(c_label, 0.3, 0.9, 0.7, 0.95));
   SUIT_setViewport(c_fbox1, VIEWPORT,
               SUIT_mapToParent(c_fbox1, 0.05, 0.15, 0.45, 0.85));
   SUIT_setViewport(c_fbox2, VIEWPORT,
               SUIT_mapToParent(c_fbox2, 0.55, 0.15, 0.95, 0.85));
   SUIT_setViewport(c_h_button, VIEWPORT,
               SUIT_mapToParent(c_h_button, 0.45, 0.05, 0.55, 0.1));

/*This SUIT callback binds a dialog box widget, bontaining the panel you made
  with the child widgets on it, to an object name*/
   c_dBox = SUIT_createOKCancelDialogBox("c_fred", c_container, NULL);

/*This call displays the box, as it implies*/
   SUIT_activateDialogBox(c_dBox);


/*SUIT has other dialog box widgets that include yes,no,etc, but these two
  REPLY_OK and REPLY_CANCEL are the only two options for this widget.
  Be sure to look down at the Delete callback to see how a validation function
  is applied to the reply.*/

   switch (SUIT_getInteger(c_dBox, "button pressed")) {
    case REPLY_CANCEL:
      break;
    case REPLY_OK:
      file1 = SUIT_getText(c_fbox1, CURRENT_VALUE);
      file2 = SUIT_getText(c_fbox2, CURRENT_VALUE);
      strcpy (cf, c_path);
      strcat (cf, file1);
      strcat (cf, " ");
```

```
        strcat (cf, file2);

/*The return value of the system call is passed to errornum, which can then
  be used to display an advisory dialog box when special values come back.*/

        errornum = system(cf);
        advisory(errornum);
        break;
    }



/*After the switch statement clear the dialog box from the screen*/
    SUIT_destroyObject(c_dBox);

}


boolean DF_OK(SUIT_object df_panel){
        switch (SUIT_askYesNo("DELETE FILE: Are you sure you want to do this
?")){
          case REPLY_YES:
            return TRUE;
            break;
          case REPLY_NO:
            return FALSE;
            break;
        }
}

/**************DELETE*****************/
void Delete(SUIT_object menu){
  SUIT_object df_h_button,df_fbox1,df_panel,df_dBox, df_label;
  char *fdel;
  char df[256] ;
  int  errornum;

   df_panel = SUIT_createBulletinBoard("D F Panel");
   df_fbox1 = SUIT_createFileBrowser("df_barney","","Delete 1 File",
"Select Delete File:",NULL);
   df_h_button = SUIT_createButton("DF_Help",HelpDelete);
   SUIT_changeObjectSize(df_panel,200,340);
   SUIT_addChildToObject(df_panel,df_fbox1);
   SUIT_addChildToObject(df_panel,df_h_button);
   df_label = SUIT_createLabel("Delete File");
   SUIT_setFont(df_label,FONT,GP_defFont("times","bold",12));
   SUIT_addChildToObject(df_panel,df_label);
   SUIT_setViewport(df_label, VIEWPORT,
              SUIT_mapToParent(df_label, 0.3, 0.9, 0.7, 0.95));
   SUIT_setViewport(df_fbox1, VIEWPORT,
              SUIT_mapToParent(df_fbox1, 0.05,0.15,0.95,0.85));
   SUIT_setViewport(df_h_button, VIEWPORT,
              SUIT_mapToParent(df_h_button, 0.45,0.05,0.55,0.1));

/*The third parameter of the below SUIT callback is a validation function.
It permits you to dummy-proof the values passed when REPLY_OK gets sent.*/
   df_dBox = SUIT_createOKCancelDialogBox("df_fred", df_panel, DF_OK);
   SUIT_activateDialogBox(df_dBox);

   switch (SUIT_getInteger(df_dBox, "button pressed")) {
    case REPLY_CANCEL:
     break;
    case REPLY_OK:
     fdel = SUIT_getText(df_fbox1, CURRENT_VALUE);
     if (fdel != NULL){
          strcpy (df, "rm  ");
          strcat (df, fdel);
          errornum = system(df);
          advisory(errornum);
     }
     break;
    }

   SUIT_destroyObject(df_dBox);

}


/****************BMP_SIMPLE (ENCODE)*************/
void BMP_Simple(SUIT_object menu)
{

   SUIT_object bs_container, bs_fbox1,bs_fbox2,bs_fbox3,density,
      bs_h_button,bs_dBox, bs_label,bs_label1, bs_label2;

   char*  file1;
   char*  file2;
   char*  file3;
   char   bs[256];
   int    den_val;
   char*  den_char;
   int    errornum;
```

```
bs_container = SUIT_createBulletinBoard ("BS Container");
bs_fbox1 = SUIT_createFileBrowser(
   "barney","","BS File1","Select Input Image",NULL);
bs_fbox2 = SUIT_createFileBrowser(
   "wilma","","BS File2","Select Input Data",NULL);
bs_fbox3 = SUIT_createFileBrowser(
   "dino","","BS File3","Select Output Image",NULL);
density = SUIT_createTypeInBox("density value", NULL);
bs_h_button = SUIT_createButton("bs_help",HelpBMP_Simple);


SUIT_changeObjectSize(bs_container, 600, 340);

SUIT_addChildToObject(bs_container, bs_fbox1);
SUIT_addChildToObject(bs_container, bs_fbox2);
SUIT_addChildToObject(bs_container, bs_fbox3);
SUIT_addChildToObject(bs_container, density);
SUIT_addChildToObject(bs_container, bs_h_button);
bs_label = SUIT_createLabel("Simple Encode");
SUIT_setFont(bs_label,FONT,GP_defFont("times","bold",12));
SUIT_addChildToObject(bs_container,bs_label);
SUIT_setViewport(bs_label, VIEWPORT,
            SUIT_mapToParent(bs_label, 0.3, 0.9, 0.7, 0.95));
bs_label1 = SUIT_createLabel("Encoding");
SUIT_addChildToObject(bs_container,bs_label1);
SUIT_setViewport(bs_label1, VIEWPORT,
            SUIT_mapToParent(bs_label1, 0.875, 0.91, 0.975, 0.95));
bs_label2 = SUIT_createLabel("Density (1-7)");
SUIT_addChildToObject(bs_container,bs_label2);

SUIT_setViewport(bs_label2, VIEWPORT,
            SUIT_mapToParent(bs_label2, 0.875, 0.85, 0.975, 0.89));
SUIT_setViewport(bs_fbox1, VIEWPORT,
            SUIT_mapToParent(bs_fbox1, 0.05, 0.05, 0.3, 0.85));
SUIT_setViewport(bs_fbox2, VIEWPORT,
            SUIT_mapToParent(bs_fbox2, 0.325, 0.05, 0.575, 0.85));
SUIT_setViewport(bs_fbox3, VIEWPORT,
            SUIT_mapToParent(bs_fbox3, 0.6, 0.05, 0.85, 0.85));
SUIT_setViewport(density, VIEWPORT,
            SUIT_mapToParent(density, 0.9, 0.775, 0.95, 0.825));
SUIT_setViewport(bs_h_button, VIEWPORT,
            SUIT_mapToParent(bs_h_button, 0.88,0.43,0.97,0.50));

bs_dBox = SUIT_createOKCancelDialogBox("fred", bs_container, NULL);
SUIT_activateDialogBox(bs_dBox);
```

```
   switch (SUIT_getInteger(bs_dBox, "button pressed")) {
    case REPLY_CANCEL:
    break;
    case REPLY_OK:
      file1 = SUIT_getText(bs_fbox1, CURRENT_VALUE);
      file2 = SUIT_getText(bs_fbox2, CURRENT_VALUE);
      file3 = SUIT_getText(bs_fbox3,CURRENT_VALUE);
      den_char =SUIT_getText(density, CURRENT_VALUE);
      strcpy (bs, "bs_encode  ");
      strcat (bs, file1);
      strcat (bs, " ");
      strcat (bs, file2);
      strcat (bs, " ");
      strcat (bs,den_char);
      strcat (bs, " ");
      strcat (bs, file3);
      errornum = system(bs);
      advisory(errornum);
      break;
      }
   SUIT_destroyObject(bs_dBox);
}


/*********************QUIT***************/
void Quit     (SUIT_object menu) {


/*This SUIT call is the thing that saves properties you interactively created
vice hard-coded*/
            SUIT_done(SAVE_SUI_FILE, EXIT_APPLICATION);}



/******************DISPL(AY)****************/
void Displ (SUIT_object menu){
  SUIT_object d_h_button,d_fbox1,d_panel,d_dBox, d_label;

  char *fdispl;
  char xv[256] ;
  int  errornum;

   d_panel = SUIT_createBulletinBoard("D Panel");
   d_fbox1 = SUIT_createFileBrowser("d_barney","","Display Image0",
"Select Display Image:",NULL);
```

```
    d_h_button = SUIT_createButton("D_Help",HelpDisplay);

    SUIT_changeObjectSize(d_panel,200,340);
    SUIT_addChildToObject(d_panel,d_fbox1);
    SUIT_addChildToObject(d_panel,d_h_button);
    d_label = SUIT_createLabel("Display An Image");
    SUIT_setFont(d_label,FONT,GP_defFont("times","bold",12));
    SUIT_addChildToObject(d_panel,d_label);

    SUIT_setViewport(d_label, VIEWPORT,
              SUIT_mapToParent(d_label, 0.3, 0.9, 0.7, 0.95));
    SUIT_setViewport(d_fbox1, VIEWPORT,
              SUIT_mapToParent(d_fbox1, 0.05,0.15,0.95,0.85));
    SUIT_setViewport(d_h_button, VIEWPORT,
              SUIT_mapToParent(d_h_button, 0.45,0.05,0.55,0.1));


    d_dBox = SUIT_createOKCancelDialogBox("d_fred", d_panel, NULL);
    SUIT_activateDialogBox(d_dBox);

    switch (SUIT_getInteger(d_dBox, "button pressed")) {
     case REPLY_CANCEL:
      break;
     case REPLY_OK:
      fdispl = SUIT_getText(d_fbox1, CURRENT_VALUE);
      if (fdispl != NULL){
         strcpy (xv, "xv  ");
         strcat (xv, fdispl);
         strcat (xv, "&");
         errornum = system(xv);
         advisory(errornum);
      }
      break;
     }

    SUIT_destroyObject(d_dBox);

}


/***************BMP_RGB (ENCODE)****************/
void BMP_RGB (SUIT_object menu) {

   SUIT_object br_container, br_fbox1,br_fbox2,br_fbox3,
      br_h_button, br_dBox,br_label;
```

```
char* file1;
char* file2;
char* file3;
char  br[256];
int   errornum;

br_container = SUIT_createBulletinBoard ("B R Container");
br_fbox1 = SUIT_createFileBrowser(
   "br_barney","","BR File1","Select In Image",NULL);
br_fbox2 = SUIT_createFileBrowser(
   "br_wilma","","BR File2","Select In Data",NULL);
br_fbox3 = SUIT_createFileBrowser(
   "br_dino","","BR File3","Select Out File",NULL);
br_h_button = SUIT_createButton("br_help",HelpBMP_RGB);

SUIT_changeObjectSize(br_container, 600, 340);

SUIT_addChildToObject(br_container, br_fbox1);
SUIT_addChildToObject(br_container, br_fbox2);
SUIT_addChildToObject(br_container, br_fbox3);
SUIT_addChildToObject(br_container, br_h_button);
br_label = SUIT_createLabel("RGB Vector Encode");
SUIT_setFont(br_label,FONT,GP_defFont("times","bold",12));
SUIT_addChildToObject(br_container,br_label);

SUIT_setViewport(br_label, VIEWPORT,
         SUIT_mapToParent(br_label, 0.3, 0.9, 0.7, 0.95));
SUIT_setViewport(br_fbox1, VIEWPORT,
         SUIT_mapToParent(br_fbox1, 0.05, 0.05, 0.3, 0.85));
SUIT_setViewport(br_fbox2, VIEWPORT,
         SUIT_mapToParent(br_fbox2, 0.325, 0.05, 0.575, 0.85));
SUIT_setViewport(br_fbox3, VIEWPORT,
         SUIT_mapToParent(br_fbox3, 0.6, 0.05, 0.85, 0.85));
SUIT_setViewport(br_h_button, VIEWPORT,
         SUIT_mapToParent(br_h_button, 0.88,0.43,0.97,0.50));

br_dBox = SUIT_createOKCancelDialogBox("br_fred", br_container, NULL);
SUIT_activateDialogBox(br_dBox);

switch (SUIT_getInteger(br_dBox, "button pressed")) {
 case REPLY_CANCEL:
  break;
 case REPLY_OK:
  file1 = SUIT_getText(br_fbox1, CURRENT_VALUE);
  file2 = SUIT_getText(br_fbox2, CURRENT_VALUE);
  file3 = SUIT_getText(br_fbox3, CURRENT_VALUE);
```

```
        strcpy (br, "rgb_steg ");
        strcat (br, file1);
        strcat (br, " ");
        strcat (br, file2);
        strcat (br, " ");
        strcat (br, file3);
        errornum = system(br);
        advisory(errornum);
        break;
    }

    SUIT_destroyObject(br_dBox);
}


/***************BMP_SIMPLE_X (EXTRACT)****************/
void BMP_Simple_X (SUIT_object menu) {

    SUIT_object bsx_container, bsx_fbox1,bsx_fbox2,bsx_h_button,
        bsx_dBox,bsx_label;

    char* file1;
    char* file2;
    char  bsx[256];
    int errornum;

    bsx_container = SUIT_createBulletinBoard ("BSX  Container");
    bsx_fbox1 = SUIT_createFileBrowser(
        "bsx_barney","","BSX File1","Get Input Image",NULL);
    bsx_fbox2 = SUIT_createFileBrowser(
        "bsx_wilma","","BSX File2","Get Output Data",NULL);
    bsx_h_button = SUIT_createButton("bsx_help",HelpBMP_Simple_X);

    SUIT_changeObjectSize(bsx_container, 400, 340);

    SUIT_addChildToObject(bsx_container, bsx_fbox1);
    SUIT_addChildToObject(bsx_container, bsx_fbox2);
    SUIT_addChildToObject(bsx_container, bsx_h_button);
    bsx_label = SUIT_createLabel("Simple Extract");
    SUIT_setFont(bsx_label,FONT,GP_defFont("times","bold",12));
    SUIT_addChildToObject(bsx_container,bsx_label);
    SUIT_setViewport(bsx_label, VIEWPORT,
            SUIT_mapToParent(bsx_label, 0.3, 0.9, 0.7, 0.95));
    SUIT_setViewport(bsx_fbox1, VIEWPORT,
                SUIT_mapToParent(bsx_fbox1, 0.05, 0.15, 0.45, 0.85));
    SUIT_setViewport(bsx_fbox2, VIEWPORT,
                SUIT_mapToParent(bsx_fbox2, 0.55, 0.15, 0.95, 0.85));
    SUIT_setViewport(bsx_h_button, VIEWPORT,
                SUIT_mapToParent(bsx_h_button, 0.45, 0.05, 0.55, 0.1));

    bsx_dBox = SUIT_createOKCancelDialogBox("bsx_fred", bsx_container, NULL);
    SUIT_activateDialogBox(bsx_dBox);

    switch (SUIT_getInteger(bsx_dBox, "button pressed")) {
     case REPLY_CANCEL:
       break;
     case REPLY_OK:
       file1 = SUIT_getText(bsx_fbox1, CURRENT_VALUE);
       file2 = SUIT_getText(bsx_fbox2, CURRENT_VALUE);
       strcpy (bsx, "bs_extract  ");
       strcat (bsx, file1);
       strcat (bsx, " ");
       strcat (bsx, file2);
       errornum = system(bsx);
       advisory(errornum);
       break;
    }

    SUIT_destroyObject(bsx_dBox);
}



/****************BMP_RGB_X (EXTRACT)****************/
void BMP_RGB_X (SUIT_object menu) {

    SUIT_object brx_container, brx_fbox1,brx_fbox2, numbytes,
      brx_h_button, brx_dBox, brx_label, brx_label1;

    char*  file1;
    char*  file2;
    char   brx[256];
    long int stegbytes;
    char* stegval;
    int    errornum;

    brx_container = SUIT_createBulletinBoard ("BRX Container");
    brx_fbox1 = SUIT_createFileBrowser(
       "bs_barney","","BRX File1","Select Input Image",NULL);
```

```
brx_fbox2 = SUIT_createFileBrowser(
   "brx_wilma","","BRX File2","Select Output Data",NULL);
brx_h_button = SUIT_createButton("brx_help",HelpBMP_RGB_X);
numbytes = SUIT_createTypeInBox("steg bytes encoded", NULL);
brx_label = SUIT_createLabel("RGB Extract");
SUIT_setFont(brx_label,FONT,GP_defFont("times","bold",12));
brx_label1 = SUIT_createLabel("Number of Encoded Bytes:");

SUIT_changeObjectSize(brx_container, 400, 340);

SUIT_addChildToObject(brx_container, brx_fbox1);
SUIT_addChildToObject(brx_container, brx_fbox2);
SUIT_addChildToObject(brx_container, brx_h_button);
SUIT_addChildToObject(brx_container, numbytes);
SUIT_addChildToObject(brx_container,brx_label);
SUIT_addChildToObject(brx_container,brx_label1);
SUIT_setViewport(brx_label, VIEWPORT,
            SUIT_mapToParent(brx_label, 0.3, 0.9, 0.7, 0.95));

SUIT_setViewport(brx_fbox1, VIEWPORT,
            SUIT_mapToParent(brx_fbox1, 0.05, 0.15, 0.45, 0.85));
SUIT_setViewport(brx_fbox2, VIEWPORT,
            SUIT_mapToParent(brx_fbox2, 0.55, 0.15, 0.95, 0.85));
SUIT_setViewport(brx_h_button, VIEWPORT,
            SUIT_mapToParent(brx_h_button, 0.85, 0.05, 0.95, 0.1));
SUIT_setViewport(numbytes, VIEWPORT,
            SUIT_mapToParent(numbytes, 0.45, 0.05, 0.65, 0.1));
SUIT_setViewport(brx_label1, VIEWPORT,
            SUIT_mapToParent(brx_label1, 0.05, 0.05, 0.4, 0.1));
brx_dBox = SUIT_createOKCancelDialogBox("brx_fred", brx_container, NULL);

SUIT_activateDialogBox(brx_dBox);

switch (SUIT_getInteger(brx_dBox, "button pressed")) {
 case REPLY_CANCEL:
   break;
 case REPLY_OK:
   file1 = SUIT_getText(brx_fbox1, CURRENT_VALUE);
   file2 = SUIT_getText(brx_fbox2, CURRENT_VALUE);
   stegval = SUIT_getText( numbytes, CURRENT_VALUE);
   strcpy (brx, "rgb_extract ");
   strcat (brx, file1);
   strcat (brx, " ");
   strcat (brx, file2);
   strcat (brx, " ");
   strcat (brx, stegval);
```

```
    errornum = system(brx);
    advisory(errornum);
    break;
  }

  SUIT_destroyObject(brx_dBox);
}

/*****************GEN_DIFF**********************/
void Gen_Diff (SUIT_object menu) {
 SUIT_object gd_container, gd_fbox1,gd_fbox2,gd_fbox3,
    gd_h_button, gd_dBox, gd_label;
  char* file1;
  char* file2;
  char* file3;
  char  gd[256];
  int   errornum;


  gd_container = SUIT_createBulletinBoard ("G D Container");
  gd_fbox1 = SUIT_createFileBrowser(
    "gd_barney","","GD File1","Name First BMP Image",NULL);
  gd_fbox2 = SUIT_createFileBrowser(
    "gd_wilma","","GD File2","Name Second BMP Image",NULL);
  gd_fbox3 = SUIT_createFileBrowser(
    "gd_dino","","GD File3","Name Diff BMP Image",NULL);
  gd_h_button = SUIT_createButton("gd_help",HelpGen_Diff);

  SUIT_changeObjectSize(gd_container, 600, 340);

  SUIT_addChildToObject(gd_container, gd_fbox1);
  SUIT_addChildToObject(gd_container, gd_fbox2);
  SUIT_addChildToObject(gd_container, gd_fbox3);
  SUIT_addChildToObject(gd_container, gd_h_button);
  gd_label = SUIT_createLabel("Generate A Diff Image");
  SUIT_setFont(gd_label,FONT,GP_defFont("times","bold",12));
  SUIT_addChildToObject(gd_container,gd_label);
  SUIT_setViewport(gd_label, VIEWPORT,
            SUIT_mapToParent(gd_label, 0.3, 0.9, 0.7, 0.95));



  SUIT_setViewport(gd_fbox1, VIEWPORT,
            SUIT_mapToParent(gd_fbox1, 0.05, 0.05, 0.3, 0.85));
  SUIT_setViewport(gd_fbox2, VIEWPORT,
            SUIT_mapToParent(gd_fbox2, 0.325, 0.05, 0.575, 0.85));
    SUIT_setViewport(gd_fbox3, VIEWPORT,
                SUIT_mapToParent(gd_fbox3, 0.6, 0.05, 0.85, 0.85));
    SUIT_setViewport(gd_h_button, VIEWPORT,
                SUIT_mapToParent(gd_h_button, 0.88,0.43,0.97,0.50));

    gd_dBox = SUIT_createOKCancelDialogBox("gd_fred", gd_container, NULL);
    SUIT_activateDialogBox(gd_dBox);

    switch (SUIT_getInteger(gd_dBox, "button pressed")) {
     case REPLY_CANCEL:
       break;
     case REPLY_OK:
       file1 = SUIT_getText(gd_fbox1, CURRENT_FILE);
       file2 = SUIT_getText(gd_fbox2, CURRENT_FILE);
       file3 = SUIT_getText(gd_fbox3, CURRENT_FILE);
       strcpy (gd, "compare  ");
       strcat (gd, file1);
       strcat (gd, " ");
       strcat (gd, file2);
       strcat (gd, " ");
       strcat (gd, file3);
       errornum = system(gd);
       advisory(errornum);
       break;
    }
   SUIT_destroyObject(gd_dBox);
}


/************************GENHIST**********************/
void genHist()
{
   SUIT_object gh_h_button,gh_bound0,gh_bound1,gh_label;
   SUIT_object  gh_bound2, gh_bound3, gh_bound4, gh_bound5,gh_label1;
   SUIT_object  gh_bound6, gh_bound7, gh_panel,gh_dBox,gh_label2,gh_label3;
   SUIT_object gh_labelc0,gh_labelc1,gh_labelc2,gh_labelc3,gh_labelc4;
   SUIT_object gh_labelc5,gh_labelc6,gh_labelc7,gh_fbox1,gh_fbox2,gh_f1,gh_f2;
   SUIT_object gh_tpix, gh_col0,gh_col1,gh_col2,gh_col3,gh_col4,gh_col5,gh_col6;
   SUIT_object gh_col7,gh_label4,gh_label5,gh_label6;
/*

Using the same format described abovea bunch of scroller widgets and text boxes are
mapped to a panel. The trick isonce you get the numbers out of the file, noting the C
commands, to make them the CURRENT_VALUE property of the scroller and textbox
widgets.*/
```

```
  FILE *infile;
  char* fname1[256];
  char* fname2[256];
  char f1buf[256];
  char f2buf[256];
  char tpixbuf[32];
  char col0_buf[16];
  char col1_buf[16];
  char col2_buf[16];
  char col3_buf[16];
  char col4_buf[16];
  char col5_buf[16];
  char col6_buf[16];
  char col7_buf[16];
  int diff_bit[8];
  double bitscale[8];
  int tot_pixels;
  int i,j;
  int errornum = 8;

/*Open the datafile*/
  infile = fopen("hist.dat","r");
  if(!(infile==NULL)){
     fscanf(infile, "%s%s",fname1,fname2);
     for (i=0; i < 8; i++){
     fscanf(infile,"%d",&diff_bit[i]);
  }
   fscanf(infile,"%d",&tot_pixels);
   for(j=0; j<8; j++){
      bitscale[j] = (double)(100*diff_bit[j])/tot_pixels;
   }
   fclose(infile);
   gh_panel = SUIT_createBulletinBoard("GH Panel");
   SUIT_setFont(gh_panel,FONT,GP_defFont("times","",12));

/*These are the bounded values I used to make the histogram. Setting these properties
constrains the displayed values from 0 to 100*/
   gh_bound0 = SUIT_createBoundedValue("gh_barney0", NULL);
   SUIT_setEnumString(gh_bound0,ACTIVE_DISPLAY,"vertical thermometer");
   SUIT_setDouble(gh_bound0,MINIMUM_VALUE,0.0);
   SUIT_setDouble(gh_bound0,MAXIMUM_VALUE,100.0);
   SUIT_setDouble(gh_bound0,GRANULARITY,1.0);
   SUIT_setDouble(gh_bound0,CURRENT_VALUE,bitscale[0]);
```

```
gh_bound1 = SUIT_createBoundedValue("gh_barney1", NULL);
SUIT_setEnumString(gh_bound1,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound1,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound1,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound1,GRANULARITY,1.0);
SUIT_setDouble(gh_bound1,CURRENT_VALUE,bitscale[1]);

gh_bound2 = SUIT_createBoundedValue("gh_barney2", NULL);
SUIT_setEnumString(gh_bound2,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound2,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound2,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound2,GRANULARITY,1.0);
SUIT_setDouble(gh_bound2,CURRENT_VALUE,bitscale[2]);

gh_bound3 = SUIT_createBoundedValue("gh_barney3", NULL);
SUIT_setEnumString(gh_bound3,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound3,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound3,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound3,GRANULARITY,1.0);
SUIT_setDouble(gh_bound3,CURRENT_VALUE,bitscale[3]);

gh_bound4 = SUIT_createBoundedValue("gh_barney4", NULL);
SUIT_setEnumString(gh_bound4,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound4,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound4,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound4,GRANULARITY,1.0);
SUIT_setDouble(gh_bound4,CURRENT_VALUE,bitscale[4]);

gh_bound5 = SUIT_createBoundedValue("gh_barney5", NULL);
SUIT_setEnumString(gh_bound5,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound5,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound5,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound5,GRANULARITY,1.0);
SUIT_setDouble(gh_bound5,CURRENT_VALUE,bitscale[5]);

gh_bound6 = SUIT_createBoundedValue("gh_barney6", NULL);
SUIT_setEnumString(gh_bound6,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound6,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound6,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound6,GRANULARITY,1.0);
SUIT_setDouble(gh_bound6,CURRENT_VALUE,bitscale[6]);

gh_bound7 = SUIT_createBoundedValue("gh_barney7", NULL);
SUIT_setEnumString(gh_bound7,ACTIVE_DISPLAY,"vertical thermometer");
SUIT_setDouble(gh_bound7,MINIMUM_VALUE,0.0);
SUIT_setDouble(gh_bound7,MAXIMUM_VALUE,100.0);
SUIT_setDouble(gh_bound7,GRANULARITY,1.0);
SUIT_setDouble(gh_bound7,CURRENT_VALUE,bitscale[7]);

gh_h_button = SUIT_createButton("GH_Help",HelpDispl_Hist);

SUIT_changeObjectSize(gh_panel,500,350);
SUIT_addChildToObject(gh_panel,gh_bound0);
SUIT_addChildToObject(gh_panel,gh_bound1);
SUIT_addChildToObject(gh_panel,gh_bound2);
SUIT_addChildToObject(gh_panel,gh_bound3);
SUIT_addChildToObject(gh_panel,gh_bound4);
SUIT_addChildToObject(gh_panel,gh_bound5);
SUIT_addChildToObject(gh_panel,gh_bound6);
SUIT_addChildToObject(gh_panel,gh_bound7);
SUIT_addChildToObject(gh_panel,gh_h_button);
gh_label = SUIT_createLabel("Diff Image Histogram");
SUIT_setFont(gh_label,FONT,GP_defFont("times","bold",12));
SUIT_setFont(gh_label,FONT,GP_defFont("times","bold",14));

SUIT_addChildToObject(gh_panel,gh_label);
SUIT_setViewport(gh_label, VIEWPORT,
        SUIT_mapToParent(gh_label, 0.6, 0.95, 0.8, 0.99));
gh_label1 = SUIT_createLabel("File 1: ");
SUIT_addChildToObject(gh_panel,gh_label1);
SUIT_setViewport(gh_label1, VIEWPORT,
        SUIT_mapToParent(gh_label1, 0.03, 0.95, 0.1, 0.98));
gh_label2 = SUIT_createLabel("File 2: ");
SUIT_addChildToObject(gh_panel,gh_label2);
SUIT_setViewport(gh_label2, VIEWPORT,
        SUIT_mapToParent(gh_label2, 0.03, 0.86, 0.1, 0.92));
gh_label3 = SUIT_createLabel("Total Diff Pixels: ");
SUIT_addChildToObject(gh_panel,gh_label3);
SUIT_setViewport(gh_label3, VIEWPORT,
        SUIT_mapToParent(gh_label3, 0.03, 0.81, 0.25, 0.85));
gh_label4 = SUIT_createLabel("100% ");
SUIT_addChildToObject(gh_panel,gh_label4);
SUIT_setViewport(gh_label4, VIEWPORT,
        SUIT_mapToParent(gh_label4, 0.1, 0.76, 0.13, 0.79));
gh_label5 = SUIT_createLabel("0%");
SUIT_addChildToObject(gh_panel,gh_label5);
SUIT_setViewport(gh_label5, VIEWPORT,
        SUIT_mapToParent(gh_label5, 0.1, 0.15, 0.13, 0.17));
gh_label6 = SUIT_createLabel("b/p diff:");
SUIT_addChildToObject(gh_panel,gh_label6);
SUIT_setViewport(gh_label6, VIEWPORT,
            SUIT_mapToParent(gh_label6, 0.05, 0.01, 0.13, 0.05));
/*These are just labels*/
  gh_labelc0 = SUIT_createLabel("1-3");
  SUIT_addChildToObject(gh_panel,gh_labelc0);
  SUIT_setViewport(gh_labelc0, VIEWPORT,
            SUIT_mapToParent(gh_labelc0, 0.15, 0.01, 0.25, 0.05));
  gh_labelc1 = SUIT_createLabel("4-6");
  SUIT_addChildToObject(gh_panel,gh_labelc1);
  SUIT_setViewport(gh_labelc1, VIEWPORT,
            SUIT_mapToParent(gh_labelc1, 0.25, 0.01, 0.35, 0.05));
  gh_labelc2 = SUIT_createLabel("7-9");
  SUIT_addChildToObject(gh_panel,gh_labelc2);
  SUIT_setViewport(gh_labelc2, VIEWPORT,
            SUIT_mapToParent(gh_labelc2, 0.35, 0.01, 0.45, 0.05));
  gh_labelc3 = SUIT_createLabel("10-12");
  SUIT_addChildToObject(gh_panel,gh_labelc3);
  SUIT_setViewport(gh_labelc3, VIEWPORT,
            SUIT_mapToParent(gh_labelc3, 0.45, 0.01, 0.55, 0.05));
  gh_labelc4 = SUIT_createLabel("13-15");
  SUIT_addChildToObject(gh_panel,gh_labelc4);
  SUIT_setViewport(gh_labelc4, VIEWPORT,
            SUIT_mapToParent(gh_labelc4, 0.55, 0.01, 0.65, 0.05));
  gh_labelc5 = SUIT_createLabel("16-18");
  SUIT_addChildToObject(gh_panel,gh_labelc5);
  SUIT_setViewport(gh_labelc5, VIEWPORT,
            SUIT_mapToParent(gh_labelc5, 0.65, 0.01, 0.75, 0.05));
  gh_labelc6 = SUIT_createLabel("19-21");
  SUIT_addChildToObject(gh_panel,gh_labelc6);
  SUIT_setViewport(gh_labelc6, VIEWPORT,
            SUIT_mapToParent(gh_labelc6, 0.75, 0.01, 0.85, 0.05));
  gh_labelc7 = SUIT_createLabel("22-24");
  SUIT_addChildToObject(gh_panel,gh_labelc7);
  SUIT_setViewport(gh_labelc7, VIEWPORT,
            SUIT_mapToParent(gh_labelc7, 0.85, 0.01, 0.95, 0.05));

/*These are the boxes that display the file names*/
  gh_f1 = SUIT_createLabel("fbox 1");
  sprintf(f1buf,"%s",fname1);
  SUIT_setText(gh_f1,LABEL,f1buf);
  SUIT_addChildToObject(gh_panel,gh_f1);
  SUIT_setViewport(gh_f1, VIEWPORT,
            SUIT_mapToParent(gh_f1, 0.12,0.95,0.55,0.98));
  SUIT_setEnumString(gh_f1,JUSTIFICATION,"left");
  gh_f2 = SUIT_createLabel("fbox 2");
  sprintf(f2buf,"%s",fname2);
  SUIT_setText(gh_f2,LABEL,f2buf);
```

```
SUIT_addChildToObject(gh_panel,gh_f2);
SUIT_setViewport(gh_f2, VIEWPORT,
        SUIT_mapToParent(gh_f2, 0.12,0.86,0.55,0.92));
SUIT_setEnumString(gh_f2,JUSTIFICATION,"left");
gh_tpix = SUIT_createLabel("tot pixels");
sprintf(tpixbuf,"%d",tot_pixels);
SUIT_setText(gh_tpix,LABEL,tpixbuf);
SUIT_addChildToObject(gh_panel,gh_tpix);
SUIT_setViewport(gh_tpix, VIEWPORT,
        SUIT_mapToParent(gh_tpix, 0.27,0.81,0.5,0.85));
SUIT_setEnumString(gh_tpix,JUSTIFICATION,"left");
gh_col0 = SUIT_createLabel("col 0");
sprintf(col0_buf,"%d",diff_bit[0]);
SUIT_setText(gh_col0,LABEL,col0_buf);
SUIT_addChildToObject(gh_panel,gh_col0);
SUIT_setViewport(gh_col0, VIEWPORT,
        SUIT_mapToParent(gh_col0, 0.15,0.1,0.25,0.14));
gh_col1 = SUIT_createLabel("col 1");
sprintf(col1_buf,"%d",diff_bit[1]);
SUIT_setText(gh_col1,LABEL,col1_buf);
SUIT_addChildToObject(gh_panel,gh_col1);
SUIT_setViewport(gh_col1, VIEWPORT,
        SUIT_mapToParent(gh_col1, 0.25,0.07,0.35,0.1));
gh_col2 = SUIT_createLabel("col 2");
sprintf(col2_buf,"%d",diff_bit[2]);
SUIT_setText(gh_col2,LABEL,col2_buf);
SUIT_addChildToObject(gh_panel,gh_col2);
SUIT_setViewport(gh_col2, VIEWPORT,
        SUIT_mapToParent(gh_col2, 0.35,0.1,0.45,0.14));
gh_col3 = SUIT_createLabel("col 3");
sprintf(col3_buf,"%d",diff_bit[3]);
SUIT_setText(gh_col3,LABEL,col3_buf);
SUIT_addChildToObject(gh_panel,gh_col3);
SUIT_setViewport(gh_col3, VIEWPORT,
        SUIT_mapToParent(gh_col3, 0.45,0.07,0.55,0.1));
gh_col4 = SUIT_createLabel("col 4");
sprintf(col4_buf,"%d",diff_bit[4]);
SUIT_setText(gh_col4,LABEL,col4_buf);
SUIT_addChildToObject(gh_panel,gh_col4);
SUIT_setViewport(gh_col4, VIEWPORT,
        SUIT_mapToParent(gh_col4, 0.55,0.1,0.65,0.14));
gh_col5 = SUIT_createLabel("col 5");
sprintf(col5_buf,"%d",diff_bit[5]);
SUIT_setText(gh_col5,LABEL,col5_buf);
SUIT_addChildToObject(gh_panel,gh_col5);
SUIT_setViewport(gh_col5, VIEWPORT,
                SUIT_mapToParent(gh_col5, 0.65,0.07,0.75,0.1));
  gh_col6 = SUIT_createLabel("col 6");
  sprintf(col6_buf,"%d",diff_bit[6]);
  SUIT_setText(gh_col6,LABEL,col6_buf);
  SUIT_addChildToObject(gh_panel,gh_col6);
  SUIT_setViewport(gh_col6, VIEWPORT,
                SUIT_mapToParent(gh_col6, 0.75,0.1,0.85,0.14));
  gh_col7 = SUIT_createLabel("col 7");
  sprintf(col7_buf,"%d",diff_bit[7]);
  SUIT_setText(gh_col7,LABEL,col7_buf);
  SUIT_addChildToObject(gh_panel,gh_col7);
  SUIT_setViewport(gh_col7, VIEWPORT,
                SUIT_mapToParent(gh_col7, 0.85,0.07,0.95,0.1));
  SUIT_setViewport(gh_h_button, VIEWPORT,
                SUIT_mapToParent(gh_h_button, 0.88,0.93,0.98,0.98));
  SUIT_setViewport(gh_bound0, VIEWPORT,
                SUIT_mapToParent(gh_bound0, 0.15,0.15,0.25,0.8));
  SUIT_setViewport(gh_bound1, VIEWPORT,
                SUIT_mapToParent(gh_bound1, 0.25,0.15,0.35,0.8));
  SUIT_setViewport(gh_bound2, VIEWPORT,
                SUIT_mapToParent(gh_bound2, 0.35,0.15,0.45,0.8));
  SUIT_setViewport(gh_bound3, VIEWPORT,
                SUIT_mapToParent(gh_bound3, 0.45,0.15,0.55,0.8));
  SUIT_setViewport(gh_bound4, VIEWPORT,
                SUIT_mapToParent(gh_bound4, 0.55,0.15,0.65,0.8));
  SUIT_setViewport(gh_bound5, VIEWPORT,
                SUIT_mapToParent(gh_bound5, 0.65,0.15,0.75,0.8));
  SUIT_setViewport(gh_bound6, VIEWPORT,
                SUIT_mapToParent(gh_bound6, 0.75,0.15,0.85,0.8));
  SUIT_setViewport(gh_bound7, VIEWPORT,
                SUIT_mapToParent(gh_bound7, 0.85,0.15,0.95,0.8));
  gh_dBox = SUIT_createOKCancelDialogBox("gh_fred", gh_panel, NULL);
  SUIT_activateDialogBox(gh_dBox);

  switch (SUIT_getInteger(gh_dBox, "button pressed")) {
   case REPLY_CANCEL:
    break;
   case REPLY_OK:

    break;
   }

  SUIT_destroyObject(gh_dBox);
 }
 else{
  advisory(errornum);
 }


}


/************************** ADVISORY ******************/
/* This is the function that displays advisory dialog boxes. Note that if 0 is
returned for normal termination, nothing gets displayed, and since genHist()
isnt a system call, you dont have to multiply the exit value by 256 to create it
*/
int advisory(int error_val){
   switch(error_val){
      case 0:
      break;
      case 8:
        SUIT_inform("Could not open hist.dat.");
      break;
      case  256:
        SUIT_inform("Insufficient number of parameters-aborted.");
      break;
      case 512:
        SUIT_inform("Could not open input image file-aborted.");
      break;
      case 768:
        SUIT_inform("Could not open output file - aborted.");
      break;
      case 1024:
        SUIT_inform("Textfile could not be opened.");
      break;
      case 1280:
        SUIT_inform("First input image file could not be opened - aborted.");
      break;
      case 1792:
        SUIT_inform("Invalid entry for bytes encoded - aborted.");
      break;
      case 2304:
        SUIT_inform("Bit density exceeds range.");
      break;
      case 2560:
        SUIT_inform("Textfile encoded at this density exceeds storage capacity of image
file.");
      break;
      case 2816:
        SUIT_inform("Textfile size is larger than image storage capacity using RGB
Vector encoding.");
```

```
      break;
      case 3072:
        SUIT_inform("Remember the input file size - available from terminal window.");
      break;
      case 3328:
        SUIT_inform("These files are not the same size - aborting");
      break;
      default:
        SUIT_inform("Returned Default");
      break;
    }

}


/******************* MAIN *********************/
void main (int argc, char *argv[])
{

  SUIT_object menuBar, fileMenu, encodeMenu,extractMenu, displayMenu,helpMenu;
  SUIT_object main_label;
/*
This gets suit to hunt down the properties file and apply the property values to widgets*/
  SUIT_init(argv[0]);

/*This adds submenu selections to the menu buttons as pulldown menus*/
  fileMenu = SUIT_createPullDownMenu ("File");
  SUIT_addToMenu(fileMenu, "Convert Format", Convert);
  SUIT_addToMenu(fileMenu, "Delete A File", Delete);
  SUIT_addToMenu(fileMenu, "Exit", Quit);

  encodeMenu = SUIT_createPullDownMenu ("Encode");
  SUIT_addToMenu(encodeMenu, "Simple Replace (.BMP)", BMP_Simple);
  SUIT_addToMenu(encodeMenu, "RGB Vector (.BMP)", BMP_RGB);

  extractMenu = SUIT_createPullDownMenu ("Extract");
  SUIT_addToMenu(extractMenu, "Simple Replace Extract (.BMP)",BMP_Simple_X);
  SUIT_addToMenu(extractMenu, "RGB Vector Extract (.BMP)", BMP_RGB_X);

  displayMenu = SUIT_createPullDownMenu ("Display");
  SUIT_addToMenu(displayMenu, "Display Image", Displ);
  SUIT_addToMenu(displayMenu, "Generate Diff Image", Gen_Diff);
  SUIT_addToMenu(displayMenu, "View Histogram", genHist);

  helpMenu = SUIT_createPullDownMenu ("Help Menu");
  SUIT_addToMenu(helpMenu, "Browse Help", GetHelpBox);
```

```
   SUIT_addToMenu(helpMenu, "About Help",  AboutHelpBox);



   /* the menu bar packs the menu buttons left to right */
   menuBar = SUIT_createMenuBar("Steganography tool");
   SUIT_addChildToObject(menuBar, fileMenu);
   SUIT_addChildToObject(menuBar, encodeMenu);
   SUIT_addChildToObject(menuBar, extractMenu);
   SUIT_addChildToObject(menuBar, displayMenu);
   SUIT_addChildToObject(menuBar, helpMenu);

   main_label = SUIT_createLabel("Steganography Toolbox");
   SUIT_setFont(main_label,FONT,GP_defFont("times","bold",12));
/* This is the command to do the external control loop. Its a nonterminating
while loop that checks if any widgets have something to pass*/
   SUIT_beginStandardApplication();
 }
```

3. TOOLBOX_HELP.H

```
/* Title: Toolbox_Help.H
  Author:LT Dave Wootten
  Purpose: This file contains all the help-related functions:
       - help browser callbacks
       - topical help panels*/
void HelpConvert(){
SUIT_object h_c_panel,h_c_dbox;
char* Notes = "These are the Convert Panel Help Notes.\n";
h_c_panel = SUIT_createTextBox("convert notes",Notes);
h_c_dbox = SUIT_createOKCancelDialogBox("H C Box", h_c_panel,NULL);
SUIT_activateDialogBox(h_c_dbox);
switch (SUIT_getInteger( h_c_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_c_dbox);

}

void HelpDelete(){
SUIT_object h_df_panel,h_df_dbox;
char* Notes = "These are the Delete File Panel Help Notes.\n";
```

```
h_df_panel = SUIT_createTextBox("del file notes",Notes);
h_df_dbox = SUIT_createOKCancelDialogBox("D F Box", h_df_panel,NULL);
SUIT_activateDialogBox(h_df_dbox);
switch (SUIT_getInteger( h_df_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_df_dbox);

}
void HelpBMP_Simple(){
SUIT_object h_bs_panel,h_bs_dbox;
char* Notes = "These are the Simple Replacement Panel Help Notes.Click OK to con
tinue";
h_bs_panel = SUIT_createTextBox("bs notes",Notes);
h_bs_dbox = SUIT_createOKCancelDialogBox("H BS Box", h_bs_panel,NULL);
SUIT_activateDialogBox(h_bs_dbox);
switch (SUIT_getInteger( h_bs_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_bs_dbox);

}

void HelpExit(){
SUIT_object h_e_panel,h_e_dbox;
char* Notes = "These are the Exit Panel Help Notes.\n";
h_e_panel = SUIT_createTextBox("exit notes",Notes);
h_e_dbox = SUIT_createOKCancelDialogBox("H E Box", h_e_panel,NULL);
SUIT_activateDialogBox(h_e_dbox);
switch (SUIT_getInteger( h_e_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_e_dbox);

}

void HelpDisplay(){
```

```
SUIT_object h_d_panel,h_d_dbox;
char* Notes = "These are the Display Panel Help Notes.\n";
h_d_panel = SUIT_createTextBox("display notes",Notes);
h_d_dbox = SUIT_createOKCancelDialogBox("H D Box", h_d_panel,NULL);
SUIT_activateDialogBox(h_d_dbox);
switch (SUIT_getInteger( h_d_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_d_dbox);

}

void HelpBMP_RGB(){
SUIT_object h_br_panel,h_br_dbox;
char* Notes = "These are the RGB Vector Panel Help Notes.\n";
h_br_panel = SUIT_createTextBox("bmp rgb notes",Notes);
h_br_dbox = SUIT_createOKCancelDialogBox("H BR Box", h_br_panel,NULL);
SUIT_activateDialogBox(h_br_dbox);
switch (SUIT_getInteger( h_br_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_br_dbox);

}

void HelpBMP_Simple_X(){
SUIT_object h_bsx_panel,h_bsx_dbox,bsx_label;
char* Notes = "These are the Simple Extract Panel Help Notes.\n";
h_bsx_panel = SUIT_createTextBox("bsx notes",Notes);
h_bsx_dbox = SUIT_createOKCancelDialogBox("H BSX Box", h_bsx_panel,NULL);
SUIT_activateDialogBox(h_bsx_dbox);
switch (SUIT_getInteger( h_bsx_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_bsx_dbox);

}
```

```
void HelpBMP_RGB_X(){
SUIT_object h_brx_panel,h_brx_dbox;
char* Notes = "These are the BMP RGB Extract Panel Help Notes.\n";
h_brx_panel = SUIT_createTextBox("brx notes",Notes);
h_brx_dbox = SUIT_createOKCancelDialogBox("H BRX Box", h_brx_panel,NULL);
SUIT_activateDialogBox(h_brx_dbox);
switch (SUIT_getInteger( h_brx_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_brx_dbox);

}

void HelpGen_Diff(){
SUIT_object h_gd_panel,h_gd_dbox;
char* Notes = "These are the Generate Difference Image Panel Help Notes.\n";
h_gd_panel = SUIT_createTextBox("gen diff notes",Notes);
h_gd_dbox = SUIT_createOKCancelDialogBox("G D Box", h_gd_panel,NULL);
SUIT_activateDialogBox(h_gd_dbox);
switch (SUIT_getInteger( h_gd_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_gd_dbox);

}

void HelpDispl_Hist(){
SUIT_object h_dh_panel,h_dh_dbox;
char* Notes = "These are the Display Histogram Panel Help Notes.\n";
h_dh_panel = SUIT_createTextBox("displ hist notes",Notes);
h_dh_dbox = SUIT_createOKCancelDialogBox("D H Box", h_dh_panel,NULL);
SUIT_activateDialogBox(h_dh_dbox);
switch (SUIT_getInteger( h_dh_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
   break;
}
   SUIT_destroyObject(h_dh_dbox);
```

```
}


void HelpHelp(){
SUIT_object h_h_panel,h_h_dbox;
char* Notes = "These are the Help Panel Help Notes.\n";
h_h_panel = SUIT_createTextBox("help notes",Notes);
h_h_dbox = SUIT_createOKCancelDialogBox("H H Box", h_h_panel,NULL);
SUIT_activateDialogBox(h_h_dbox);
switch (SUIT_getInteger( h_h_dbox,"button pressed")){
  case REPLY_CANCEL:
  break;
  case REPLY_OK:
  break;
}
  SUIT_destroyObject(h_h_dbox);

}




void GetHelpTopic(SUIT_object scrollbox){
  char* h_val;
  h_val = SUIT_getText(scrollbox,CURRENT_VALUE);
  if (strcmp(h_val,"Convert")==0){
    HelpConvert();
  }
  else if (strcmp(h_val,"Delete File")==0){
    HelpDelete();
  }

  else if (strcmp(h_val,"Display Image")==0){
    HelpDisplay();
  }
  else if (strcmp(h_val,"Exit")==0){
    HelpExit();
  }

  else if (strcmp(h_val,"Help")==0){
    HelpHelp();
  }
  else if (strcmp(h_val,"RGB Vector")==0){
    HelpBMP_RGB();
  }
```

```
   else if (strcmp(h_val,"RGB Vector Extract")==0){
      HelpBMP_RGB_X();
   }
   else if (strcmp(h_val,"Simple Replace")==0){
      HelpBMP_Simple();
   }
   else if (strcmp(h_val,"Simple Replace Extract")==0){
      HelpBMP_Simple_X();
   }
   else if (strcmp(h_val,"Generate Diff Image")==0){
      HelpGen_Diff();
   }
   else if (strcmp(h_val,"Display Histogram")==0){
      HelpDispl_Hist();
   }
}
void GetHelpBox(SUIT_object menu) {
   SUIT_object h_browser,h_dbox,h_panel, h_label;
   char* HelpTopics[] = {"Convert","Delete File","Display Image","Exit","Help",
"RGB Vector","RGB Vector Extract","Simple Replace","Simple Replace Extract", "Ge
nerate Diff Image","Display Histogram"};
   h_panel = SUIT_createBulletinBoard("Help Browse");
   h_browser = SUIT_createScrollableList("help list",NULL);
   SUIT_changeObjectSize(h_panel,200,130);
   SUIT_addChildToObject(h_panel,h_browser);
   h_label = SUIT_createLabel("Help Browser");
   SUIT_setFont(h_label,FONT,GP_defFont("times","bold",12));
   SUIT_addChildToObject(h_panel,h_label);
   SUIT_setViewport(h_label, VIEWPORT,
               SUIT_mapToParent(h_label, 0.3, 0.9, 0.7, 0.95));
   SUIT_setViewport(h_browser, VIEWPORT,
               SUIT_mapToParent(h_browser, 0.05, 0.05, 0.95, 0.85));

   SUIT_setTextList(h_browser,LIST,SUIT_defTextList (HelpTopics, 11));
   h_dbox = SUIT_createOKCancelDialogBox("H Box", h_panel,NULL);
SUIT_activateDialogBox(h_dbox);
switch (SUIT_getInteger( h_dbox,"button pressed")){
   case REPLY_CANCEL:
   break;
   case REPLY_OK:
      GetHelpTopic(h_browser);
   break;
}
   SUIT_destroyObject(h_dbox);
)
}
```

LIST OF REFERENCES

Anderson, R., “Workshop on Information Hiding”, http://www.cl.cam.ac.uk/users/rja14/ ihws.html, 1996.

Aura,T., “Invisible Communication”, ftp://saturn.hut.fi/pub/aaura/stell95.ps, 1995.

Arachelian, R. A., *White Noise Storm* (shareware), ftp://ftp.csua.berkeley.edu/pub/ cypherpunks/steganography/wns210.zip, 1994.

Black Wolf (alias), *StegoDOS -Black Wolf s Picture Encoder Version 0.90a* (public domain software), ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/ stegodos.zip, 1994.

Brown, A., “Steganography", ftp://ftp.crl.com/users/ro/smart/tfp/steganography.html, 1994.

Brown, A., *S-Tools for Windows* (shareware), ftp://ftp.dsi.vnimi.it/.1/security/crypt/codes/ s-tools.zip, 1994.

Conway, M. et. al., *A Tutorial for SUIT, the Simple User Interface Toolkit*, University of Virginia, (1992).

Cornsweet, T. N., *Visual Perception*, Academic Press, 1970.,pp. 135-198,384-418.

Currie, D. and Campbell, H.,*Implementing and Efficiency of Steganographic Techniques in Bitmapped Images and Embedded Data Survivability Against Lossy Compression Schemes* (Thesis), Naval Postgraguate School,1996.

Daley, M. L., R. C. Watzke, and M. C. Riddle, “A Model for the Apparent Decrease in Optical Transmittance of the Diabetic Eye”, *IEEE Trans. on Biomedical Engineering*, Volume 39, Number 1, January 1992.

BiomedicalEngineering,Volume 39, Number 1, January 1992.

Johnson, N., "Steganography", http://www.patriot.net/users/johnson/html/neil/stegdoc/stegdoc.html, 1996.

Kiger, J. I., "The Depth/Breadth Tradeoff in the Design of Menu-Driven User Interfaces", *International Journal of Man-Machine Studies*, Volume 20 (1984), pp. 201-213.

Kuhn, M., (untitled), Steganography Newsgroup Electronic Mail Posting, March 7, 1995.

Kurak, C and J. McHugh, "A Cautionary Note on Image Downgrading", *IEEE Eighth Annual Computer Security Applications Conference*, 1992.

Landover, T. K. and D. W. Nachbar, "Selection FromAlphabetic and Numeric Menu Trees Using a Touch Screen: Breadth, Depth, and Width", P*roceedings of Human Factors in Computing Systems*,ACM SIGCHI, New York , April 1985, pp. 73-78.

Machado, R., "Announcing Stego Version 1.0a2 (shareware), http://www.fqa.com/romana/romansoft/Stego1a2.Readme.txt, November 28, 1993.

Maroney, C., *Hide and Seek* (public domain software), ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/hdsk41b.zip, 1994.

Murray, J. D., and W. VanRyper, *Encyclopedia of Graphics File Formats*, O'Reilly and Associates, Inc., 1994, pp. 7-16,125-170.

Norman, K. 1., and J. P. Chin, "The Effect of Tree Structure on Search in a Hierarchical Menu-Driven Selection System", *Behavior and Information Technology,* Volume 7, 1988, pp.51-65.

Paine, D. P., *Aerial Photography and Image Interpretation for Resource Management*, John Wiley and Sons, 1981, pp. 251-278.

Riggs, L.A., F. Ratcliff, J. C. Cornsweet, and T. N. Cornsweet, "The Disappearance of

Steadily Fixated Visual Test Objects", *Journal of the Optical Society of America*, Volume 43, pp 495-301.

Rushton, W. A. H., "Cone Pigment Kinetics in the Protanope", *Journal of Physiology* (London), Volume 186, 1963.

Russ, J. C., *The Image Processing Handbook*, IEEE Press, 1995, pp 32-48.

Sabins, F. F., *Remote Sensing Principles and Interpretation*, W. H. Freeman and Co., 1978, pp. 18-21.

Sheppard. J. J., *Human Color Perception*, American Elsevier Publishing Co., 1968.

Schneiderman, B., *Designing the User Interface*, Second Edition, Addison-Wesley Publishing Co.,1992, pp. 97-137.

Shimeall, T. J., CS 4540 (Software Testing) Class Notes, Naval Postgraduate School, January 1996.

Wald, G.,, "The Receptors for Human Color Vision", *Science*, Volume 145, pp 1007-1017.

Young, N. *Simple User Interface Toolkit*, ftp://suit@uvacs.cs.Virginia.EDU, 1990.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir,VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code CS/LT 2
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5118

4. Raymond Isbell... ...2
   Central Imagery Office
   8401 Old Courthouse Road
   Vienna VA 22182-3280

5. Dr. Blaine Burnham..2
   National Security Agency
   Research and Development Building
   R23
   9800 Savage road
   Fort Meade MD 20755-6000

6. William Marshall... 2
   National Security Agency
   Research and Development Building
   R23
   9800 Savage road
   Fort Meade MD 20755-6000

7. Dr. Cynthia E. Irvine... 5
   Computer Science Department
   Code CS/Ic
   Naval Postgraduate School
   Monterey, CA 93943-5118

8. Dr Harold Fredericksen.. 2
Mathematics Department, Code MA/FS
Naval Postgraduate School
Monterey, CA 93943-5118

9. Dr. Michael J. Zyda...2
Computer Science Department
Code CS/Zk
Naval Postgraduate School
Monterey, CA
93943-5118

10. LT Daniel L. Currie III... 2
Fleet Information Warfare Center
2555 Amphibious Drive
NAB Little Creek
Norfolk, VA 23521-3225

11. LT Hannelore Campbell.. 2
USS BOXER LDH 4
FPO AP 96661

12. LT David R. Wootten..4
Naval Reserve readiness Command, Region ONE
344 Easton Street
Newport, RI 02841-1515