Project Report on

Image Steganography

submitted by

Jeelesh Darji (CE031) (19CEUBF016)

&

Munaf Divan (CE035) (19CEUBG006)

for term work of

B.Tech CE Semester : VI
Subject : (CE-621) System Design Practice

Under the guidance of

Prof. Ashish K. Gor
(Assistant Professor Department of Computer Engineering)

Faculty of Technology

Department of Computer Engineering

Dharmsinh Desai University

Department of Computer Engineering

Dharmsinh Desai University

Certificate

This is to certify that the practical / term work carried out in the subject of System Design Practice and recorded in this journal is the bonafide work of

Jeelesh Darji (CE031) (19CEUBF016)
Munaf Divan (CE035) (19CEUBG006)

of B.Tech semester VI in the branch of Computer Engineering during the academic year 2021-22.

Prof. Ashish K. Gor
(Assistant Professor, Department of Computer Engineering)

Table of Contents

1. Abstract

Steganography is the art of hiding secret message in such a way that no one, apart from sender and intended recipient suspects existence of message, a form of security through obscurity. The goal of steganography is to hide messages inside other harmless data in a way that does not allow any enemy to even detect that there is data hidden.

In this project, an efficient and secure image and video steganography algorithm in the wavelet domain based on the skin detection algorithm and entropy based sub-band selection is implemented. The system also considers multiple skin areas for hiding different messages. The proposed algorithm includes five different phases. First, the secret message is encrypted and then converted into binary stream of bits. Second, skin detection algorithm is applied on the cover in order to identify all the skin areas. Third, out of all the skin areas, the largest skin area is taken as ROI and passed through the DWT process. Fourth, the most textured subband is selected based on entropy where embedding process is carried out and LSB method is used for hiding the modified segment. Fifth, the process of extracting the secret message from the frequency wavelet coefficients of a particular skin region is accomplished.

The use of **Skin areas as ROI**, **automatic entropy based sub-band selection**, **selecting random detail coefficients for hiding secret message bits** using a seed that generates random locations has made the system more **secure. The use of high frequency sub-band coefficients of DWT** which are less sensitive to human eyes for embedding secret bits and according to **performance measure results**, proposed approach provides **fine visual**

quality with **less cost.** The system can be made robust by hiding data in a compressed cover instead of directly hiding in cover and some other methods.

2. Introduction

In today's world, people communicate over the Internet and share private information. In order to block data from intruders and hackers, this secret information should be protected through a secure technique. The secret data can be hacked for the purpose of copyright violation, for tampering it or can be illegally accessed without the knowledge of owner. Due to these reasons there is a need of hiding secret data inside different types of digital data such that owner can prove copyright ownership, identify attempts to tamper with sensitive data and to embed annotations.

Steganography is a process that involves hiding important information (message) inside other carrier (cover) data to protect the message from unauthorized users. Our Human Visual System is not able to recognize the small changes that occur in cover data after hiding the secret data. Thus, the stego data will be seen by the Human Visual System (HVS) as one piece of data. The message and the cover data can be of any format such as text, audio, image, and video. However, there are many steganalytical detectors that detect a secret message from an unsecure steganography algorithm. Hence, researchers are working on developing secure steganography algorithms that are protected from both attackers and steganalysis detectors. A good steganography system should have high embedding payload and high embedding efficiency. First, the embedding payload is defined as the amount of secret information that is going to be embedded inside the cover data. The algorithm has a high embedding payload if it has a large capacity for the secret message. The embedding efficiency includes the stego visual quality, security, and robustness against attackers. Second, both a low modification rate and good quality of the cover data lead to a high embedding efficiency.

2.1 History

To understand steganography, we should first take a look at its predecessor: cryptography. Cryptography is the art of protecting information by transforming it into an unreadable format, called cipher text. To decipher this unreadable format, a secret key is required.[14]

- Cryptography was found as far back as 1900 BC. in an ancient Egyptian scribe.
- From 500 – 600 B.C. ATBASH, a reversed alphabet simple solution cipher, was used by Hebrew.
- From 50 - 60 B.C. a simple substitution with the normal alphabet in government communications was used by Julius Caesar.
- In today's scenario Quantum Cryptography is being used that combines physics and cryptography to produce a new cryptosystem that is very difficult to cryptanalyze without having the knowledge of the attempted and failed intrusion. Through the long history of cryptography, steganography was developed and flourished on its own.
- The use of steganography dates back to 440 BC.

  i. The Greek ruler Histaeus gave birth to steganography by: **shaving the head of a slave**, tattooing the message on the slaves scalp, waiting for the growth of hair in order to hide the secret message, and sending the slave on his way to deliver the message.

  ii. And Demaratus, who sent a warning about attack to Greece by writing it directly on the wooden backing of a wax tablet before applying its beeswax surface.
- In 1600s, Sir Francis Bacon used a variation in type face to carry each bit of the encoding.

- During the American Revolutionary War **Invisible Inks** were used by both the British and American forces.
- During World War II the Germans introduced microdots. The microdots were complete documents, pictures, and plans reduced in size to the size of a period and attached to common paperwork. Null ciphers were also used to pass secret messages. Null ciphers are unencrypted messages with real messages embedded in the current text..
- During the 1980's Margareth Thatcher, then Prime Minister in UK, became so irritated about press leaks of cabinet documents, that she had the word processors programmed to encode the identity of the writer in the word spacing, thus being able to trace the disloyal ministers.

2.2 Definition of Steganograpgy

Steganography comes from the Greek steganos (covered or secret) and graphy (writing or drawing).

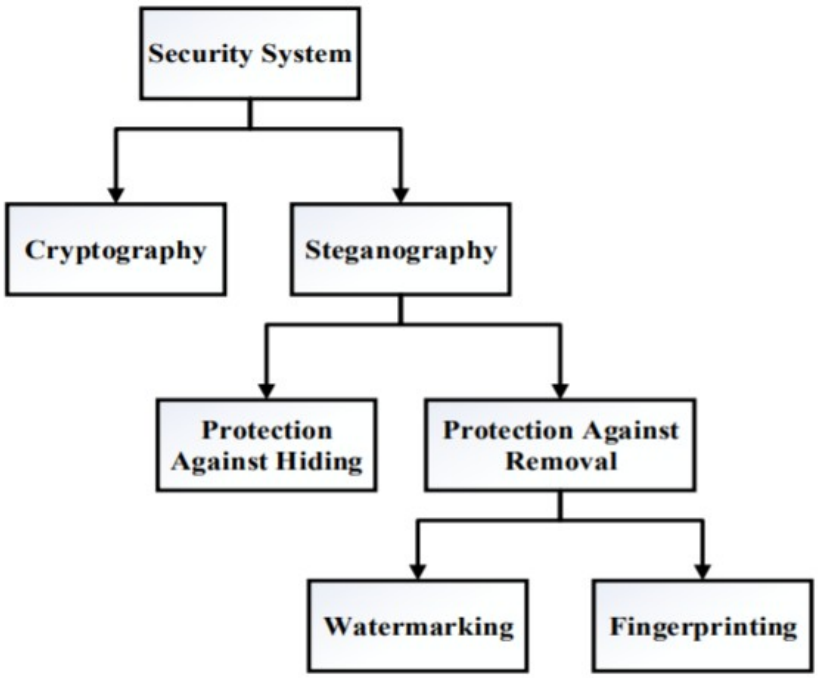
or

The art and science of concealing message in the form of text, image, video or file within another text, image video or file is called steganography .

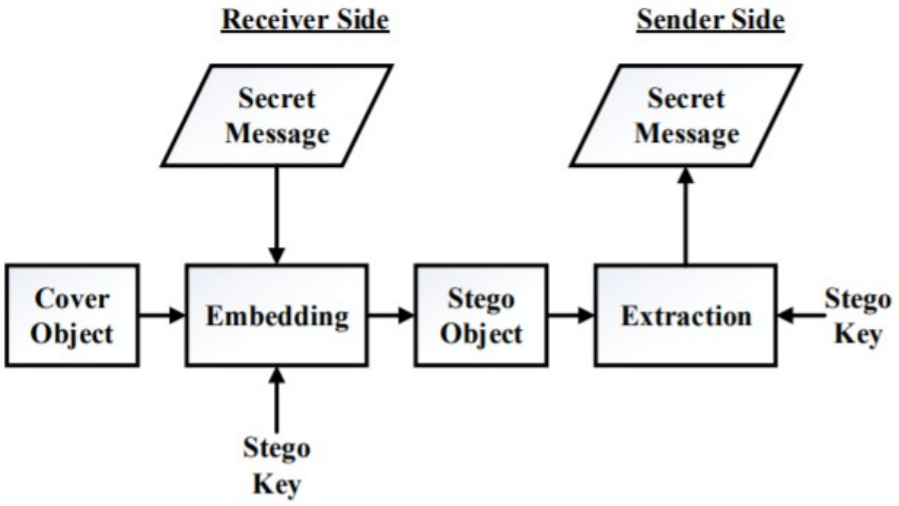
2.3 Overview of Steganograpgy

Often steganography is confused with cryptography because both have objective of protecting important information. The difference between the two is that while cryptography involves coding the message using an encryption key and sending it as cipher text, steganography involves hiding the intended message within a seemingly harmless cover. To exploit human perception is the work of steganography. Human senses are not trained to

look for files that have information inside it. The most common use of steganography is to hide a file inside another file. Steganography can be done by either providing protection against detection or by providing protection against removal. Protection against removal can be carried out by either watermarking or fingerprinting.



Basic Model of Steganography:



It consists of:

- **Cover Object:** It is the input image, video file, audio file or a text file in which concealment of secret data is to be performed.
- **Stego-Object:** After the concealment of secret data into the cover medium, the cover object becomes the stego-object.
- **Embedding:** Embedding is the process of making a stego-object from a cover object. Or we can define it as the process of concealment of secret message into some digital medium.
- **Extraction:** This is the reverse process of embedding. In this process, the conceded message is recovered from stego-object to read it.
- **Message:** It is the secret information that is to be embedded in the cover object for safe transmission.

Image Steganography is the art of writing hidden messages inside images, in such a way that no one apart from the sender and intended recipient realizes the existence of a hidden message. Steganography uses repeating portions of the Image files to embed the secret message. There are many techniques for hiding data within Image/Video.

Important Steganographic Measures

- Mean Squared Error(MSE)
- Peak Signal Noise Ratio(PSNR)
- Time Complexity
- Universal Image Quality Index(UIQI)
- Structural Similarity Index Metric (SSIM)
- Bit Error Rate(BER)
- Similarity Function(SF)

2.4 Taxonomy/Classification of Steganography

- **Based on carrier:** text, image, audio, video
- **Based on message format:** text, image, audio, video
- **Based on domain:** Spatial domain, Frequency domain
- **Based on methods used:** Spatial Domain Methods (LSB, Pseudo-random LSB Encoding), Frequency Domain Methods (DCT, DFT, DWT), Spread Spectrum Method, Statistical Method, Distortion Method, Visual Cryptography, Cover Generation Method

2.5 Applications of Steganography

- Confidential digital communication and secret data storing.
- Media Database systems.
- Military and intelligence agencies.
- Protection of tampering data by criminals.
- Law enforcement and counter intelligence agencies.
- Online Free Speech on the net, including anonymous remailers and Web proxies.
- Digital elections and digital cash.
- Marketers use email forgery techniques.
- Finger prints and forensic.
- Telecommunication.
- Medical images.

2.6 Problem Definition

The earlier techniques consists of linguistic or language forms of hidden writing. The later techniques, such as invisible try to hide messages physically. One disadvantage of linguistic steganography is that users must have a good knowledge of linguistry. In recent years, everything is rapidly trending towards digitization. And with the development of the internet technology, digital media can be transmitted smoothly over the network. Therefore, messages can be secretly carried through digital media by using the steganography techniques, and then be transmitted through the internet rapidly.

Many different carrier file formats can be used for steganography, but image and video are the most popular because of their high embedding capacity. For hiding secret information in an image or a video, there exists a large variety of steganography techniques. Some techniques are more complex than others and all of them have their respective strong and weak points.

Many of the existing steganography algorithms are designed without taking into account the visual quality or distortion as much as focusing on the embedding strategies. As a result, these algorithms are built with the lack of security, robustness, and imperceptibility[10]. Moreover, most techniques lack intelligent processing of the cover, that is, the whole cover is equally utilized in the hiding process without following any adaptive approach for **selecting the best regions for embedding data**. Additionally, the whole cover is involved in the hiding process which affects the visual quality of the resultant stego-object and also affects the quality of the extracted data.[11]

Image steganography in **wavelet domain** reduce frequency attacks and provides high imperceptibility as image is a collection of pixels with each pixel having capacity to store 1 to 64 bits of data. In order to design an efficient and secure steganography algorithm combination of skin detection algorithm, entropy based sub-band selection and choosing random positions based on a seed are used.

2.7 Purpose

Recent steganography algorithms are based on spatial domain which can simply be plagiarized by anyone. As well as Fourier transform is just based on only frequency domain, time domain information cannot be achieved by that. Moreover till now in most of the work embedding is just a simple substitution method; imperceptibility, security and capacity is still the issue.

2.8 Scope and Objective

This project is developed for hiding information in an image file. The main goal is to design an algorithm that is more efficient and secure along with high embedding payload at a less cost. The critical task is to select an appropriate frequency band and particular positions to embed secret data on the sender side as well as successfully retrieve back the exact secret data on the receiver side in order to fulfill the goal. The proposed work supports images of .jpg and .png format as cover object. Multiple skin areas are considered for hiding different secret message. Therefore images having more than one skin area are also preferred.

We also created basic GUI using qt framework. This GUI uses the code we developed for hiding and extracting messages to and from image. In this tool we developed user can also find out different performance measure for images.

2.9 Challenges/Requirements

- To design an easy, efficient and simple embedding and extracting algorithm.
- To design a highly imperceptible and highly secure algorithm.
- It should have high embedding payload (capacity) as well as high embedding efficiency (good quality of stego object, robust).
- A good coordination among all the methods used: entropy based sub-band selection method, skin detection and embedding (/extracting).
- To deal with different data types like single, double, signed and unsigned integers, logical etc.

3. LITERATURE REVIEW

3.1 Image Steganography

I. Spatial Domain Based

These techniques use the pixel gray levels and their color values directly for encoding the message bits. These techniques are some of the simplest schemes in terms of embedding and extraction complexity. The major drawback of these methods is amount of additive noise that creeps in the image which directly affects the Peak Signal to Noise Ratio and the statistical properties of the image. Moreover these embedding algorithms are applicable mainly to lossless image-compression schemes like TIFF images. For lossy compression schemes like JPEG, some of the message bits get lost during the compression step.

The most common algorithm belonging to this class of techniques is the Least Significant Bit (LSB) replacement technique in which the least significant bit of the binary representation of the pixel gray levels is used to represent the message bit. This kind of embedding leads to an addition of a noise of 0:5p on average in the pixels of the image where p is the embedding rate in bits/pixel. This kind of embedding also leads to an asymmetry and a grouping in the pixel gray values (0,1);(2,3);. . . (254,255). To overcome this undesirable asymmetry, the decision of changing the least significant bit is randomized i.e. if the message bit does not match the pixel bit, then pixel bit is either increased or decreased by 1. This technique is popularly known as LSB Matching. It can be observed that even this kind of embedding adds a noise of 0:5p on average. To further reduce the noise, have suggested the use

of a binary function of two cover pixels to embed the data bits. The embedding is performed using a pair of pixels as a unit, where the LSB of the first pixel carries one bit of information, and a function of the two pixel values carries another bit of information. It has been shown that embedding in this fashion reduces the embedding noise introduced in the cover signal.

II. Frequency Domain Based

These techniques try to encode message bits in the transform domain coefficients of the image. Data embedding performed in the transform domain is widely used for robust watermarking. Similar techniques can also realize large capacity embedding for steganography. Candidate transforms include discrete cosine Transform (DCT), discrete wavelet transform (DWT), and discrete Fourier transform (DFT). By being embedded in the transform domain, the hidden data resides in more robust areas, spread across the entire image, and provides better resistance against signal processing. For example, we can perform a block DCT and, depending on payload and robustness requirements, choose one or more components in each block to form a new data group that, in turn, is pseudo randomly scrambled and undergoes a second-layer transformation. Modification is then carried out on the double transform domain coefficients using various schemes. These techniques have high embedding and extraction complexity. Because of the robustness properties of transform domain embedding, these techniques are generally more applicable to the “Watermarking” aspect of data hiding. Many steganographic techniques in these domain have been inspired from their watermarking counterparts.

.

Discrete Cosine Transform (DCT):

DCT is a well-known method which is utilized in many applications such as image and video compression. The DCT separates the signal into low, middle, and high frequency regions. The DCT is closely related to the discrete Fourier transform (DFT). It is a separable linear transformation; that is, the 2D-DCT is equivalent to a 1D-DCT performed along a single dimension followed by a 1D-DCT in the other dimension. For an input video frame, A, of resolution M x N the DCT frequency coefficients for the transformed frame, B, and the inverse DCT coefficients of the reconstructed frame are calculated according to the following equations, respectively:

$$B_{pq} = \propto_p \propto_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N} \quad (1)$$

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \propto_p \propto_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N} \quad (2)$$

Where $\propto_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 1 \le p \le M-1 \end{cases}$

And $\propto_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 1 \le q \le N-1 \end{cases}$

A (m, n) is the pixel value in row m and column y of the frame A, and B (p, q) is the coefficient in row p and column q of the 2D-DCT matrix. Each of low, middle, and high frequency coefficients were used as cover data to embed the encoded secret message.

Discrete Wavelet Transform

The Discrete Wavelet Transform can identify portions of cover image where secret data could be effectively hidden. DWT splits information into its high and low frequency components. The high frequency part of the signal contain details about the edge components, whereas the low frequency part contains most of the signal information of the image which is again split into higher and lower frequency parts. For each level of decomposition in two dimensional applications, first DWT is performed in the vertical direction followed by horizontal direction.

3.2 Diversion based on transforms

Table 3. 1 Diversity in Transformations

Domain	Description	Merits	Demerits
DFT	Signal composing of both sin and cosine waves	RST Invariant (Rotation-Scale-Time)	Either frequency or time representation
DCT	Signal solely composes of different frequencies and amplitudes	Simple, robust against filtering and compression techniques	Not RST Invariant (Rotation-Scale-Time)
DWT	Frequency-Time representation of a signal	Simultaneous localization in time and frequency domain, Finer detail and fast in computation	Shift sensitivity, Poor directionality, Absence of phase information.

3.3 Deviations of different wavelets:

Table 3. 2 Various types of Wavelets

Wavelet	Merits	Demerits
Haar	• It is conceptually simple. • It is fast. • It is memory efficient, since it can be calculated in place without a temporary Array.	• Discontinuous signal . • Not preferred for audio applications.
Daubechies	• Deals with finer values. • Picks up details missed by haar transform. • Imperceptibility.	• Higher computational overhead. • More complex.
Daubechies Lifting Method	• Memory efficient. • A temporary array is not required as required in daubechies. • Easy conversion that maps integers to integers. • Retains perfect reconstruction.	• Optimal only in the case when filter lengths are large.

4. PROPOSED APPROACH

Image steganography can be defined as the art and science of embedding secret data in images. Data hiding in images has gained practical significance nowadays due to the huge technological advancement of multimedia systems.The greatest advantage of image is the large amount of data that can be hidden inside it. Therefore any small but otherwise noticeable distortion might go unobserved by humans because of the continuous flow of information.

4.1 Introduction to proposed Design

In the proposed system:

- Image is used as cover.
- Secret message can be in the form of either a text or an image.
- Cover Image format used: .jpg or .png

The basic steps of the proposed system are divided into five phases:

- First, the secret message is converted into a binary stream of bits.
- Second, skin detection algorithm is applied on the cover in order to identify the skin areas as the regions of interest.
- Third, the largest skin area is passes through DWT in order to extract its frequency coefficients.
- Fourth, an entropy based sub-band selection method is used in order to find the most textured sub-band and embedding positions are randomly selected using a seed.
- Fifth, the process of extracting the secret message from a particular ROI's particular sub-band and from particular positions.

4.2 Steganographic Technique

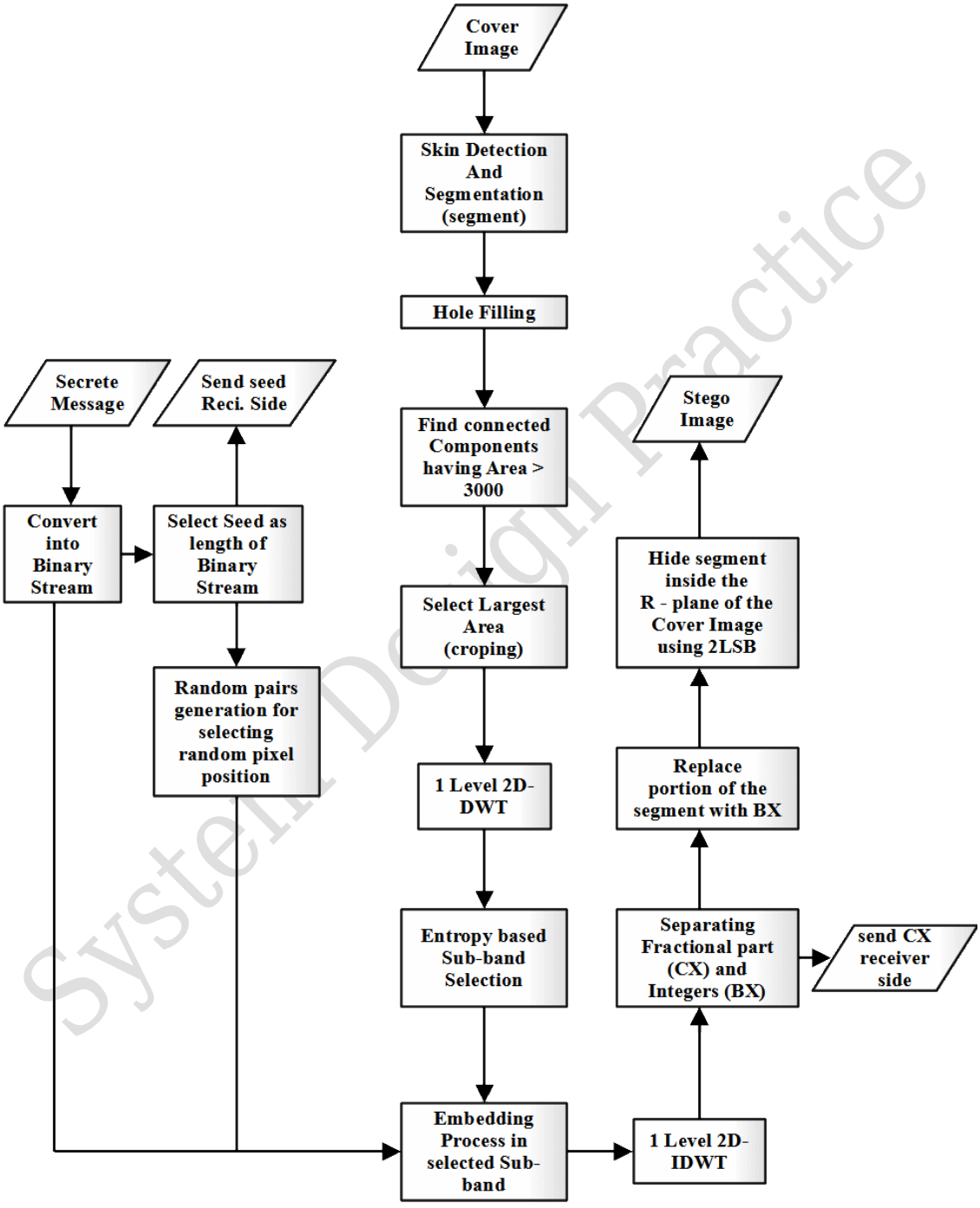
- Instead of using spatial domain frequency domain method of steganography is used in order to make it more robust against attacks.

- Further in frequency domain DWT technique is used instead of other techniques as it has fine visual quality and high embedding payload.

- Skin regions are taken as ROI as skin detection methodologies are mostly biased to the detection of skin pixels based on their color. The reason behind this is that skin color provides robust information against rotations, scaling and partial occlusions. Skin regions are less sensitive to HVS.

- Entropy based sub-band selection gives the most textured sub-band for hiding the data which adds to the imperceptibility of the system.

4.3 Embedding Process



4.3.1 Preprocessing Secret Data

In the proposed work the secret data is text and it is preprocessed before embedding phase.

In preprocessing secret data is converted into an array of binary bits.

4.3.2 Skin Detection and Segmentation

- In this section we are combining HSV and YCbCr color-space masks for skin detection. By doing so we can more accurately discriminate between skin pixels and non-skin pixels.
- In this approach, for finding skin regions we are using this range of values:
- HSV mask range = (0 ,15 ,0) to (17,170,255)
- YcbCr mask range = (0, 135, 85) to (255, 180 , 135)
- we combine this two mask and find skin regions.

4.3.3 Hole filling and Largest Skin part Extraction

The output of skin segmentation is passed through a morphological operation called hole filling in order to get a well-defined, proper area. Then connected components having area>3000 in image are searched. Among those connected components we find coordinates of largest area and using these coordinates we crop ROI from the binary image we get after skin segmentation.

4.3.4 Skin Decomposition

Level 1 2D-DWT is applied on the ROI producing four subbands, LL (approximation), LH (horizontal), HL (vertical), and HH (diagonal), using both a low pass filter and a high pass filter for the decomposition process.

LL is a low frequency sub-band, which is an approximation of the original image reduced to a quarter of its size. The LH, HL, and HH sub-bands are middle and high frequencies that contain detailed information about any image.

DWT is a recognized method that transfers the signal from the time domain to the transform domain at different frequency bands with different resolutions. DWT separates high, middle, and low frequencies and their boundaries from one another, while other methods, such as DCT, group the various frequencies into estimated regions.

In the proposed algorithm, the LH, HL, and HH coefficients of the first level are used as cover data for embedding secret message. Wavelet transform is used because it is resistant to different frequency attacks.

- **Resistant to different frequency attacks:**
  1. Histogram equalization
  2. Intensity adjustment
  3. Gamma correction.

4.3.5 Selecting Sub-band

After finding all three sub-bands (LH,HL and HH), we need to select most textured sub-band where we can hide our secret data. To find most textured sub-band, we find Shannon entropy of all sub-bands.

The Shannon entropy is defined as S = -sum(pk * log2(pk)), where pk are frequency/probability of pixels of value k.
after getting entropy of all sub bands one sub-band based on the maximum texture area is to be selected. So the sub-band having maximum entropy is selected as sub-band where secret data will be embedded.

4.3.6 Embedding secret data

- Embedding process uses a randomly generated pairs of integers where the first value of each pair lies within sub-band's height i.e. number of rows and the second value lies within sub-band's width i.e. number of columns. Random pairs are generated using a seed. Seed is taken to be the size of the binary stream of secret message. One pair indicates a particular pixel location to be considered for embedding.
- First using the seed and values within the specified range,
- Number of pairs generated = size of the binary stream of secret message
- All pairs are unique from each other.
- Each bit of the secret message are embedded in the selected sub-band using pairs generated as coordinates.
- The seed is sent through the channel to the receiver side.

4.3.7 Inverse DWT

Once we add secret message into a particular sub-band than we need to inverse our transform so that the original image with secret message can be retrieved back. Low frequency image is combined with high frequency images which were temporally localized and in this manner original image is reconstructed back.

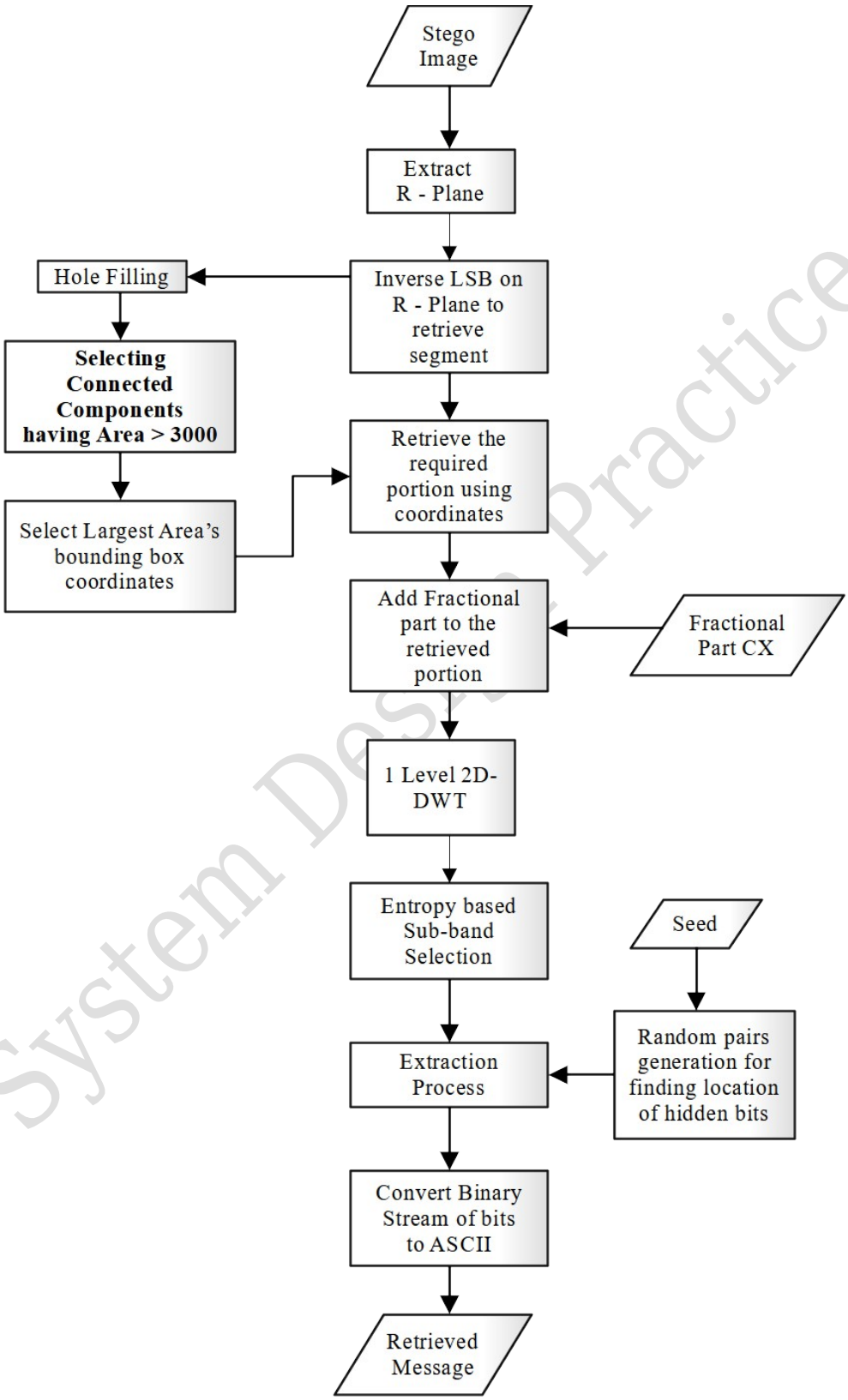
4.3.8 Separating Fractional and Integer Part

The modified cropped component is divided into two parts: Fractional and Integer part. Reason behind doing so: dwt2 and idwt2 function's outputs are in double form and the cover image is in uint8 form. So, the modified double sub-band cannot be hidden exactly in same form within the uint8 image. Thus, leading to failure in extracting the exact message. The specific area of the segment as indicated by the coordinates of the largest area's bounding box is replaced by the integer part of the modified cropped component. The fractional part is sent through the channel to the receiver side.

4.3.9 Hiding modified segment

The R-, G- and B–Planes of the original image are extracted. The modified segment is hidden within the R-Plane of the original image using 2-LSB. The Modified R-Plane, GPlane and the B-Plane are merged to produce the Stego Image.

4.4 Extraction Process



4.4.1 Extracting R-Plane and Segment

R-Plane of the Stego Image is extracted. Inverse 2-LSB is applied on the retrieved R-Plane in order to get the segment.

4.4.2 Largest Skin Component Detection

Hole filling and largest skin part extraction are carried out on the segment extracted from R-plane at the receiver side in order to get the coordinates of the component where the secret data is embedded.

4.4.3 Retrieve the ROI

Using the coordinates retrieved from above step 2 of extraction, the ROI is cropped from the segment we extracted from R-plane. Next step is to add the fractional part to the ROI.

4.4.4 Skin Decomposition

Apply level 1 2D-DWT to the modified ROI producing LL, LH, HL and HH sub-bands.

4.4.5 Sub-band Selection

The most textured sub-band is selected by calculating shannon entropy of each sub-band and selecting the band having the highest entropy.

4.4.6 Extracting binary array of secret message

Extracting the secret message bits from the pixels specified by the pairs generated using the seed. Convert the binary array of secret message to the ASCII code in order to get the secret message.

4.5 Performance metrics

Table 4. 1 Way of checking performance metrics

Performance Metrics	How to check
Visual Quality	**PSNR:** It is measured by calculating PSNR and MSE. PSNR is a nonperceptual objective metric measuring the difference between the original and distorted images. $PSNR = 10.\log_{10}(\frac{MAX_I{}^2}{MSE})$ Where, MAXI represents maximum value of pixel of the image. In the images with pixel having 8 bits per sample, its value is 255. The MSE stands for cumulative squared error between the stego image and the original image. $MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{h}[C(i,j,k) - S(i,j,k)]^2}{m \times n \times h}$ C and S refer to the cover image and stego image respectively. m and n defines as image resolutions and h indicated the R, G and B color channels. (k=1,2 & 3)
Time Complexity	It can be calculated by noting down the time taken for embedding process and extraction process.
Universal Image Quality Index	It measures distortion that has occurred in cover object due to embedding process. $UIQI(A,B)=L(A,B)* C(A,B)* S(A,B$ Where, Luminance distortion, $L(A,B) = 2\mu A\mu B/\mu A$^2+$\mu B$^2 Contrast distortion, $C(A,B)=2\sigma A\sigma B/\sigma A$^2 +$\sigma B$^2 Loss of correlation, $S(A,B)=2\sigma AB/\sigma A+\sigma B$ Where A is cover image, μA and σA are mean and standard

	deviation, respectively of A. B is stego image, μB and σB is mean and standard deviation, respectively of B. σAB is covariance between A and B.
Structural Similarity Index Metric (SSIM)	SSIM is an objective image quality metric and is superior to traditional measures such as MSE and PSNR. PSNR estimates the perceived errors, whereas SSIM considers image degradation as perceived change in structural information. Structural information is the idea that the pixels have strong interdependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. The SSIM Index quality assessment index is based on the computation of three terms, namely the luminance term, the contrast term and the structural term. The overall index is a multiplicative combination of the three terms. $SSIM(x,y)=[l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$ where, $$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$ $$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$ $$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$ where μx, μy, σx,σy, and σxy are the local means, standard deviations, and cross-covariance for images *x, y*. If α = β = γ = 1 and C3 = C2/2 the index simplifies to: $$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$
Hidden Ratio	The embedding payload of the proposed algorithm is tested by calculating the hidden ratio which is given by: (Total Size of message in Bits / m x n) *100 Where, m and n are width and height of selected sub-band respectively.

5. IMPLEMENTATION DETAILS

5.1 Library functions used

Table 5. 1 Details of Library Functions Used

Function	Description
dwt2	This function is used to apply 2D discrete wavelet transform on ROI and get sub-bands. It is single level only. pywt.**dwt2**(data,wavelet,mode='symmetric',axes=(-2,-1)) 2D Discrete Wavelet Transform. Parameters: **data**: *array_like (*2D array with input data) **wavelet** : *Wavelet object or name string, or 2-tuple of wavelets* Wavelet to use. This can also be a tuple containing a wavelet to apply along each axis in axes. **Mode** : *str or 2-tuple of strings, optional* **axes**: *2-tuple of ints, optional* Axes over which to compute the DWT. Repeated elements mean the DWT will be performed multiple times along these axes. Returns: **(cA, (cH, cV, cD))**: *tuple* Approximation, horizontal detail, vertical detail and diagonal detail coefficients respectively. Horizontal refers to array axis 0 (or axes[0] for user-specified axes).
idwt2	This function used to get original ROI back from sub-bands. It is single level only. pywt.**idwt2**(coeffs, wavelet, mode='symmetric', axes=(-2, -1)) 2-D Inverse Discrete Wavelet Transform. Reconstructs data from coefficient arrays. Parameters: **coeffs** : *tuple* (cA, (cH, cV, cD)) A tuple with approximation

	coefficients and three details coefficients 2D arrays like from dwt2. If any of these components are set to None, it will be treated as zeros. **Wavelet** : *Wavelet object or name string, or 2-tuple of wavelets* Wavelet to use. This can also be a tuple containing a wavelet to apply along each axis in axes. **Mode** : *str or 2-tuple of strings, optional* **Axes**: *2-tuple of ints, optional* Axes over which to compute the IDWT. Repeated elements mean the IDWT will be performed multiple times along these axes. Returns : 2D Array of type Float.
shannon_entropy	**skimage.measure.shannon_entropy (*image, base=2*)** Calculate the Shannon entropy of an image. The Shannon entropy is defined as S = -sum(pk * log(pk)), where pk are frequency/probability of pixels of value k. Parameters: **image:** (N, M) ndarray Grayscale input image. **Base :** float, optional The logarithmic base to use. Returns: **entropy:** float
connectedComponentsWithStats	connectedComponentsWithStats(image[, labels[, stats[, centroids[, connectivity[, ltype]]]]]) ->retval, labels, stats, centroids computes the connected components labeled image of boolean image and also produces a statistics output for each label image with 4 or 8 way connectivity - returns N, the total number of labels [0, N-1] where 0 represents the background label. ltype specifies the output label image type, an important consideration based on the total number of labels or alternatively the total number of pixels in the source image.

5.2 Functions Implemented

5.2.1 Skin Detection Function

```
def skin_detection(img):
    # converting from gbr to hsv color space
    img_HSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    # skin color range for hsv color space
    HSV_mask = cv2.inRange(img_HSV, (0, 15, 0), (17, 170, 255))
    HSV_mask = cv2.morphologyEx(
        HSV_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))

    # converting from gbr to YCbCr color space
    img_YCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
    # skin color range for YCbCr color space
    YCrCb_mask = cv2.inRange(img_YCrCb, (0, 135, 85), (255, 180, 135))
    YCrCb_mask = cv2.morphologyEx(
        YCrCb_mask, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))

    # merge skin detection (YCbCr and hsv)
    global_mask = cv2.bitwise_and(YCrCb_mask, HSV_mask)
    # removing noise from image
    global_mask = cv2.medianBlur(global_mask, 3)
    global_mask = cv2.morphologyEx(
        global_mask, cv2.MORPH_OPEN, np.ones((4, 4), np.uint8))
    # checking if there are skin pixels in image or not
    unique, counts = np.unique(global_mask, return_counts=True)
    if(unique[-1] != 255):
        raise SkinNotDetected(
            "Oops!!! Skin can not be Detected in given Image")
    return global_mask
```

5.2.2 Hole Filling Function

```
def FillHole(mask):

    contours, hierarchy = cv2.findContours(
        mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    len_contour = len(contours)
    contour_list = []
    for i in range(len_contour):
        drawing = np.zeros_like(mask, np.uint8)  # create a black image
        img_contour = cv2.drawContours(
            drawing, contours, i, (255, 255, 255), -1)
        contour_list.append(img_contour)
    out = sum(contour_list)
    if type(out) == type(1):
        if(out == 0):
            raise SkinNotDetected(
                "Oops!!! Hole Filling is not Possible in this Image")
    return out
```

5.2.3 Finding Largest Connected Component Coordinates

```
def largest_connected_component_coordinates(image):
    image = image.astype("uint8")
    nb_components, output, stats, centroids = cv2.connectedComponentsWithStats(
        image, connectivity=8
    )
    # states has sizes of all connected componets there is in the image and
    # we will take it as non-increasing array
    sizes = stats[:, -1]
    # find component of size >3000
    max_label = 1
    max_size = 3000  # find component of size >3000
    # 0th component is the background component so we cannot take that
    for i in range(1, nb_components):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]
    if(max_size == 3000):
        raise LargestComponentNotFound(
            "Oops!!! Skin Component larger than 3000 can't be Found")
    img2 = np.zeros(output.shape)
    img2[output == max_label] = 255
    y_nonzero, x_nonzero = np.nonzero(img2)
    return (y_nonzero, x_nonzero)
```

5.2.4 Coordinate Generation

```
def generate_coordinates(ROI, length):
    m, n = ROI.shape
    m = m - 1
    n = n - 1
    random.seed(length)
    coordinates = set()
    x, y = randint(1, m), randint(1, n)
    while len(coordinates) < length:
        while (x, y) in coordinates:
            x, y = randint(1, m), randint(1, n)
        coordinates.add((x, y))
    return coordinates
```

5.2.5 Hiding Message into Selected Sub-band

This function uses last bit of sub-band pixel to hide binary data.

```
def hide_in_maxband(coordinates, maxEntropy_Band, binary):
    modified_maxEntropy_Band = np.copy(maxEntropy_Band)
    i = 0
    for x, y in coordinates:

        fract_pixel, int_pixel = math.modf(maxEntropy_Band[x, y])
        bin_pixel = bin(int(int_pixel))
        new_bin_pixel = list(bin_pixel)
        new_bin_pixel[len(bin_pixel) - 1] = binary[i]
        bin_pixel = "".join(new_bin_pixel)
        int_pixel = int(bin_pixel, 2)
        modified_maxEntropy_Band[x, y] = int_pixel + fract_pixel
        i = i + 1
    return modified_maxEntropy_Band
```

5.2.6 Separate Fractional and Integer Part

```
def seperate_int_fract(idwt_ROI):
    rows, cols = idwt_ROI.shape
    int_ROI = np.empty((rows, cols), np.uint8)
    fract_ROI = np.empty((rows, cols), np.float64)

    for x in range(0, rows):
        for y in range(0, cols):
            fract_pixel, int_pixel = math.modf(idwt_ROI[x, y])
            if int_pixel >= 0:
                int_ROI[x, y] = int_pixel
                fract_ROI[x, y] = fract_pixel
            else:
                int_ROI[x, y] = abs(int_pixel)
                fract_ROI[x, y] = -fract_pixel
    return int_ROI, fract_ROI
```

5.2.7 Combine Fractional and Integer Part

```
def combine_roi(ROI, fract):
    width, height = ROI.shape
    new_roi = np.zeros_like(ROI, dtype="float64")

    for y in range(height):
        for x in range(width):
            fractd = fract[x, y]
            if fractd < 0:
                new_roi[x, y] = -(ROI[x, y] + abs(fractd))
            else:
                new_roi[x, y] = ROI[x, y] + fractd
    return new_roi
```

5.2.8 Hiding Modified Binary Image Into R Plane

This function basically uses last 2 bits of R plane pixel to hide binary image pixel.

```
def hide_in_R(image_to_hide, image_to_hide_in):
    width, height = image_to_hide_in.shape

    for y in range(height):
        for x in range(width):
            value = image_to_hide[x, y]
            pixel = image_to_hide_in[x, y]
            bin_pixel = bin(int(pixel))
            new_bin_pixel = list(bin_pixel)
            if value == 0:
                new_bin_pixel[len(bin_pixel) - 2] = "0"
                new_bin_pixel[len(bin_pixel) - 1] = "0"
            elif value == 1:
                new_bin_pixel[len(bin_pixel) - 2] = "0"
                new_bin_pixel[len(bin_pixel) - 1] = "1"
            elif value == 254:
                new_bin_pixel[len(bin_pixel) - 2] = "1"
                new_bin_pixel[len(bin_pixel) - 1] = "0"
            elif value == 255:
                new_bin_pixel[len(bin_pixel) - 2] = "1"
                new_bin_pixel[len(bin_pixel) - 1] = "1"

            bin_pixel = "".join(new_bin_pixel)
            int_pixel = int(bin_pixel, 2)
            image_to_hide_in[x, y] = int_pixel

    # return an Image object from the above data.
    return image_to_hide_in
```

5.2.9 Extracting Binary Image from R plane

```
def get_binary_from_R(combine_img):

    width, height = combine_img.shape
    hidden_img = np.zeros_like(combine_img, dtype="uint8")
    for y in range(height):
        for x in range(width):
            pixel = combine_img[x, y]
            bin_pixel = bin(int(pixel))
            new_bin_pixel = list(bin_pixel)
            if (
                new_bin_pixel[len(bin_pixel) - 2] == "0"
                and new_bin_pixel[len(bin_pixel) - 1] == "0"
            ):
                value = 0
            elif (
                new_bin_pixel[len(bin_pixel) - 2] == "0"
                and new_bin_pixel[len(bin_pixel) - 1] == "1"
            ):
                value = 1
            elif (
                new_bin_pixel[len(bin_pixel) - 2] == "1"
                and new_bin_pixel[len(bin_pixel) - 1] == "0"
            ):
                value = 254
            elif (
                new_bin_pixel[len(bin_pixel) - 2] == "1"
                and new_bin_pixel[len(bin_pixel) - 1] == "1"
            ):
                value = 255

            hidden_img[x, y] = value
    return hidden_img
```

6. DATASETS, TOOLS and TECHNOLOGY

6.1 Implementation Platform Details

Proposed approach is Implemented and Tested on platform given below.

6.1.1 Hardware Specification

Processor: Intel Core i5-8265U CPU 1.80GHz

RAM: 8 GB

Google Colab: 8GB Ram (Used for Testing of Some Images)

6.1.2 Software Specification

OS: Windows 10

System Type: 64 bit OS

Front End: Python

6.2 Tools and Technology

6.2.1 Technologies

1. Python
2. OpenCV
3. Scikit Image
4. PyQt
5. sewer
6. Numpy

6.2.2 Tools

1. Visual Studio Code
2. Google Colab
3. Qt Designer

6.3 Dataset Design

6.3.1 Secret Message Dataset

Secret messages: They are in **Text** form Text messages of various sizes and made up of various types of characters like symbols, capital letters and small letters are used.

Table 6. 1 Texts Used as Secret Message

Msg No.	Secret Message	Msg Bits
1	"I have a dream that one day every valley shall be exalted, every hill and mountain shall be made low." - Martin Luther King Jr.	128*8 = **1024**
2	This file is created to hide text. Tomorrow is Republic day for india. How ARE 12345 you. password1: ` 1 2 3 4 5 6 7 8 9 0 - = password2: ~!@ # $ % ^ & * () _ + password3: ABCDEFGHIJKLMNOPQRSTUVWXYZ. password4: abcdefghijklmnopqrstuvwxyz. password5: aVrt@56Hm Please change the size of the cover image.	310*8 =**2480**
3	Alan Turing, in full Alan Mathison Turing, (born June 23, 1912, London, England-died June 7, 1954, Wilmslow, Cheshire), British mathematician and logician who made major contributions to mathematics, cryptanalysis, logic, philosophy, and mathematical biology and also to the new areas later named computer science, cognitive science, artificial intelligence, and artificial life. The son of a civil servant, Turing was educated at a top private school. He entered the University of Cambridge to study mathematics in 1931. After graduating in 1934, he was elected to a fellowship at King's College (his college since 1931) in recognition of his research in probability theory. In 1936 Turing's seminal paper On Computable Numbers, with an Application to the Entscheidungsproblem [Decision Problem] was recommended for publication by the American	3736*8 = **29888**

mathematical logician Alonzo Church, who had himself just published a paper that reached the same conclusion as Turing's, although by a different method. Turing's method (but not so much Church's) had profound significance for the emerging science of computing. Later that year Turing moved to Princeton University to study for a Ph.D. in mathematical logic under Church's direction (completed in 1938).

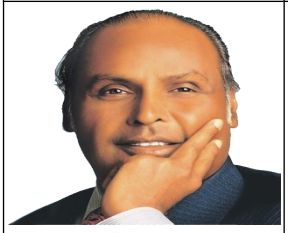
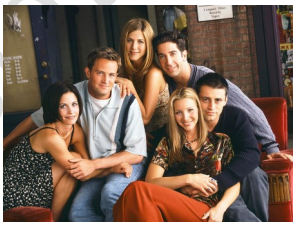
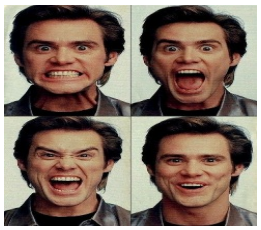
What mathematicians called an ‹effective' method for solving a problem was simply one that could be carried by a human mathematical clerk working by rote. In Turing's time, those rote-workers were in fact called Computers,‹ and human computers carried out some aspects of the work later done by electronic computers. The Entscheidungsproblem sought an effective method for solving the fundamental mathematical problem of determining exactly which mathematical statements are provable within a given formal mathematical system and which are not. A method for determining this is called a decision method. In 1936 Turing and Church independently showed that, in general, the EntscheidungsproblemResults/ problem has no resolution, proving that no consistent formal system of arithmetic has an effective decision method. In fact, Turing and Church showed that even some purely logical systems, considerably weaker than arithmetic, have no effective decision method. This result and others-notably mathematician-logician Kurt Godel's incompleteness results-❾ dashed the hopes, held by some mathematicians, of discovering a formal system that would reduce the whole of mathematics to methods that (human) computers could carry out. It was in the course of his work on the Entscheidungsproblem that Turing invented the universal Turing machine, an abstract computing machine that encapsulates the fundamental logical principles of the digital computer.

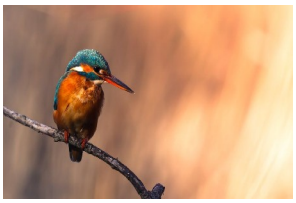
An important step in Turing's argument about the Entscheidungsproblem was the claim, now called the Church-Turing thesis, that everything humanly computable can also be computed by the universal Turing machine. The claim is important because it marks out the limits of human computation. Church in his work used instead the thesis that all human-computable functions are identical to what he called lambda-definable functions (functions on the positive integers whose values can be calculated by a process of repeated substitution). Turing showed in 1936 that Church's thesis was equivalent to his own, by proving that every lambda-definable function is computable by the universal Turing machine and vice versa. In a review of Turing's work, Church acknowledged the superiority of Turing's formulation of the thesis over his own (which made no reference to computing machinery), saying that the concept of computability by a Turing machine 'has the advantage of making the identification with effectiveness evident immediately.'

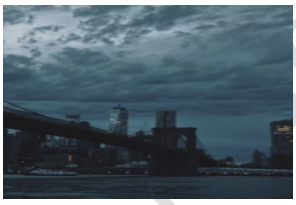
6.3.2 Cover Image Dataset

Cover Image: A set of colored images having skin colored areas are used as cover for Steganography. Cover images are in **.jpg** and .**png** format.

Table 6. 2 Images Used as Cover Object

Cover Image	Resolution	Size	Cover Image	Resolution	Size
 1.jpeg	960 x 720	56.9 KB	 2.png	2560 x 1440	529 KB
 3.jpg	1920 x 1080	212.7 KB	 4.jpg	2448 x 3264	2.8 MB
 5.png	493 x 532	306.6 KB	 6.jpg	1264 x 948	224.1 KB
 7.jpg	960 x 640	97.8 KB	 8.jpg	498 x 700	76.8 KB
 9.jpg	4252 x 3607	1.7 MB	 10.jpg	1040 x 780	213.3 KB

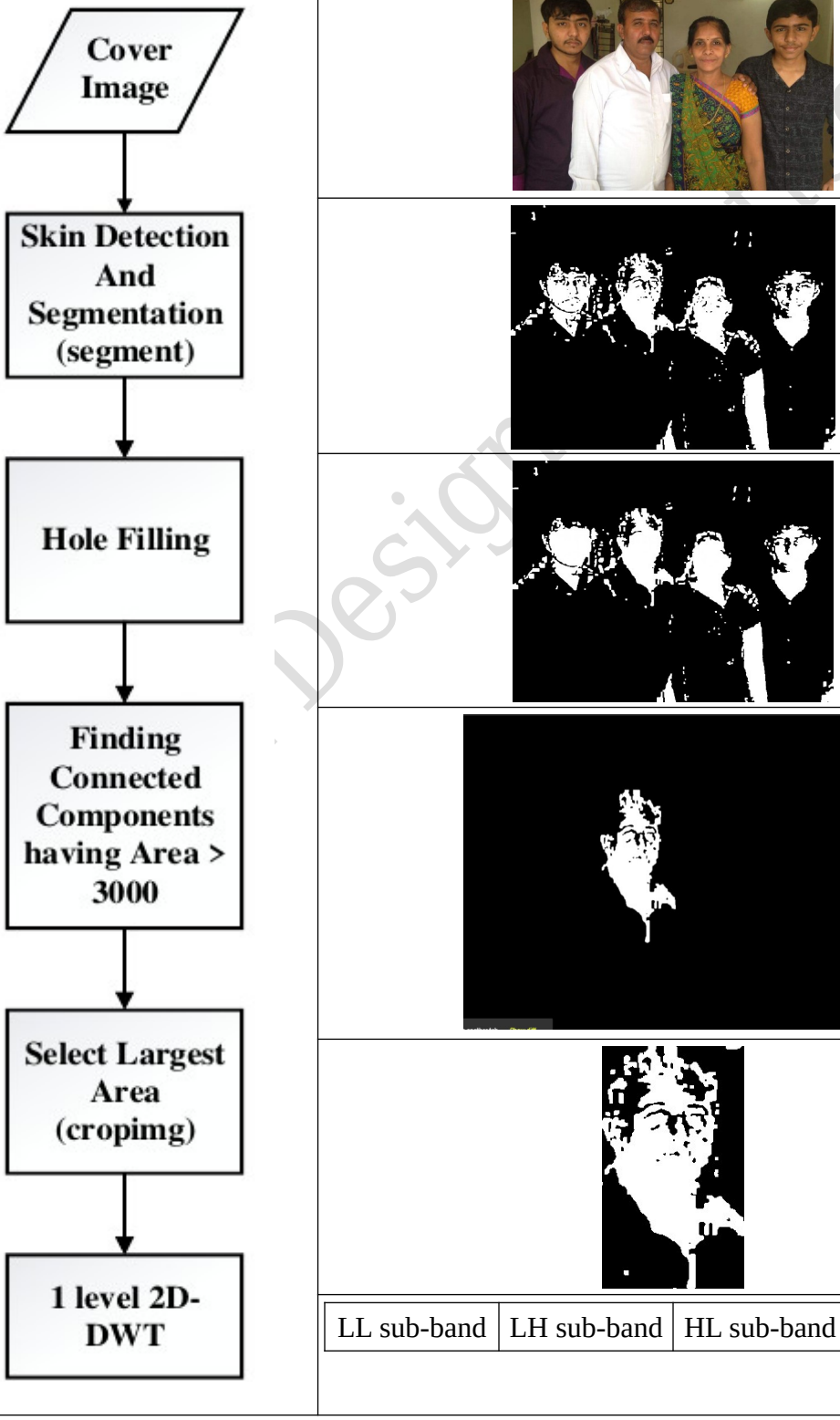
 11.jpg	480 x 320	19.9 KB	 12.jpg	960 x 640	80.8 KB
13.jpg 	509 x 541	53.7 KB	 14.jpg	800 x 825	53.2 KB
 15.jpg	1246 x 1367	278.1 KB	 16.jpg	510 x 340	86.1 KB
 17.jpeg	800 x 1066	148 KB	 18.jpg	1200 x 900	198.5 KB
 19.jpg	1600 x 1412	783.9 KB	 20.jpg	508 x 339	113.1 KB
 21.jpg	2119x 1414	662 KB	 22.jpg	976 x 549	37.8 KB

 23.jpg	1000 x 667	395 KB	 24.jpeg	1280 x 720	253 KB
 25.jpg	1920 x 1440	357 KB	 26.jpg	1280 x 853	561 KB
 27.jpg	800 x 553	57 KB	 28.jpg	3024 x 4032	1.23 MB
 29.jpg	1200 x 667	96.4 KB	 30.jpg	3072 x 2040	926 KB

7. TESTING RESULTS

7.1 Embedding and Extracting Process in Details

Table 7. 1 Embedding Process for Message 2





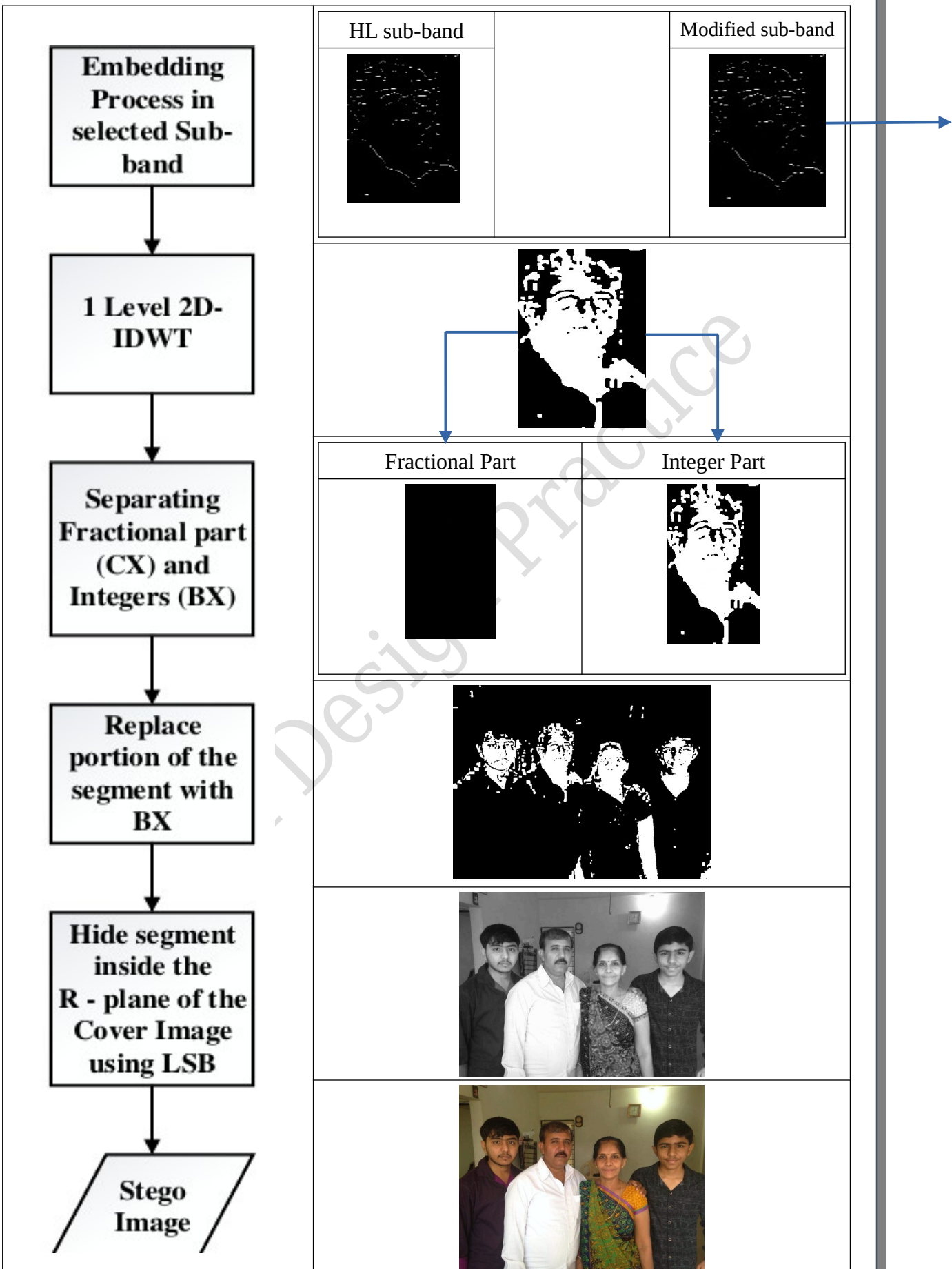
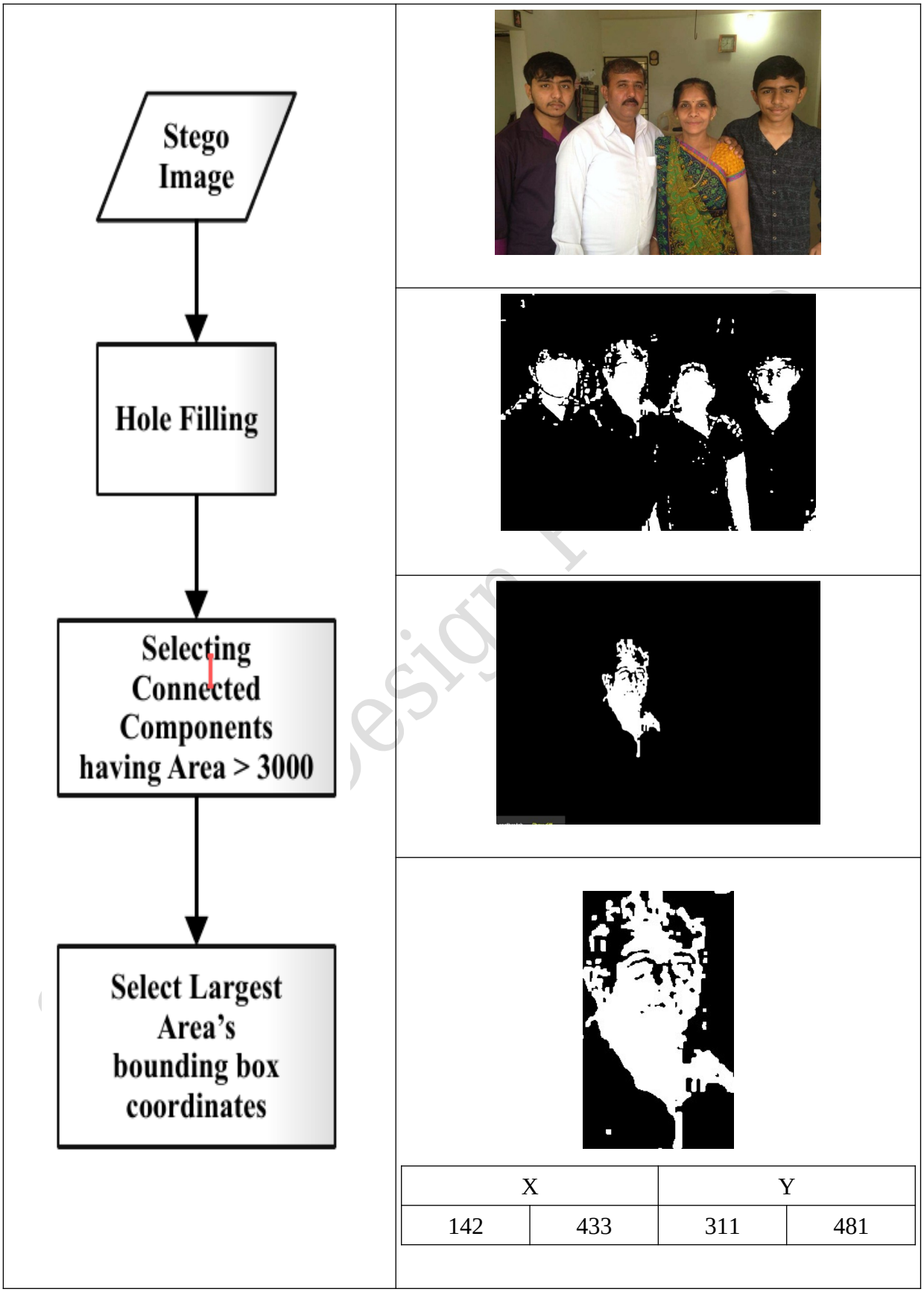
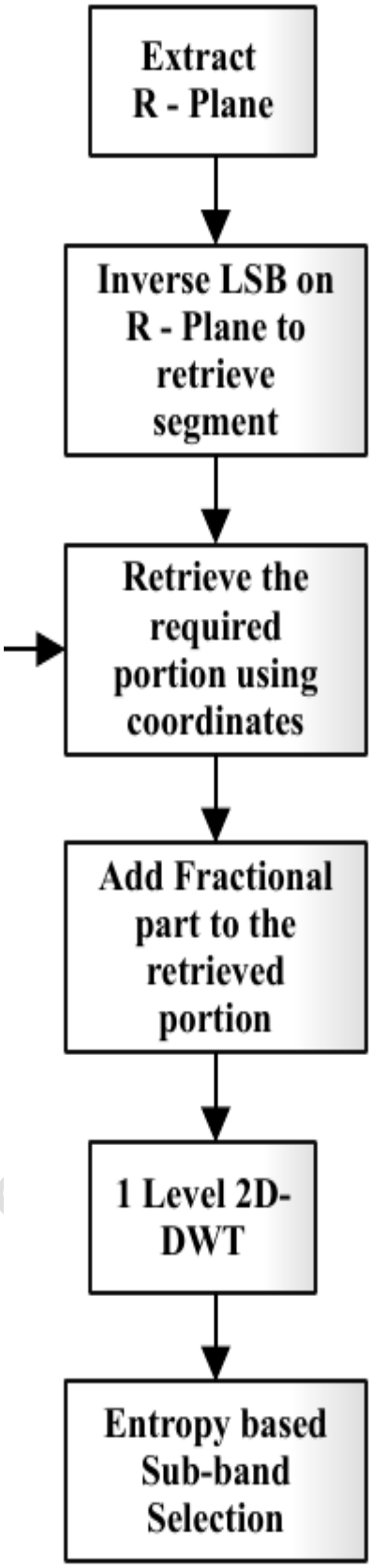



Table 7. 2 Extracting Process for Message 2



X		Y	
142	433	311	481



LL	LH	HL	HH
			

Extraction Process By using the pairs generated using the seed

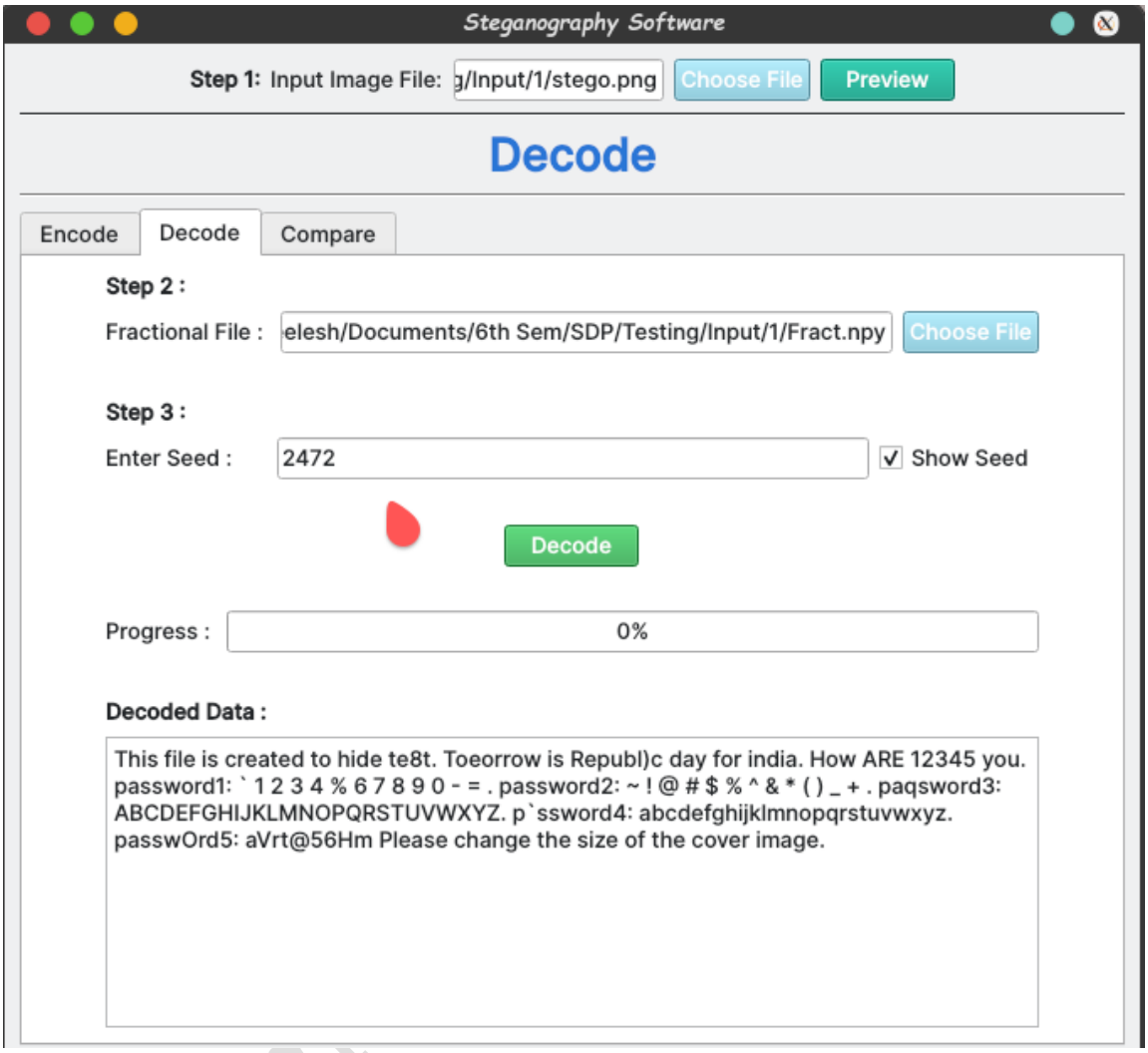
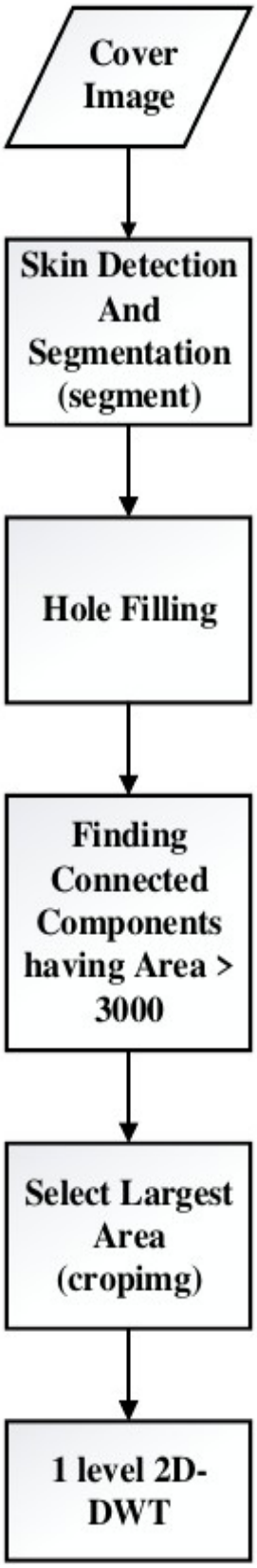
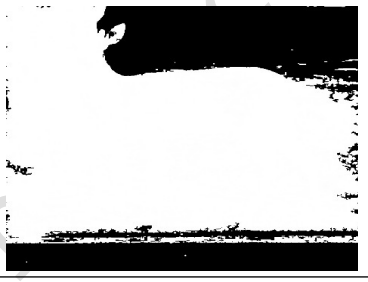
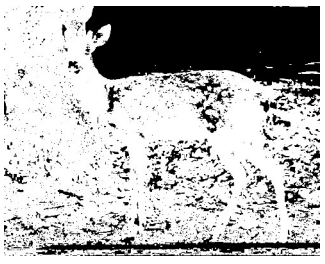


Table 7. 3 Embedding Process for Message 3







LL sub-band	LH sub-band	HL sub-band	HH sub-band
			

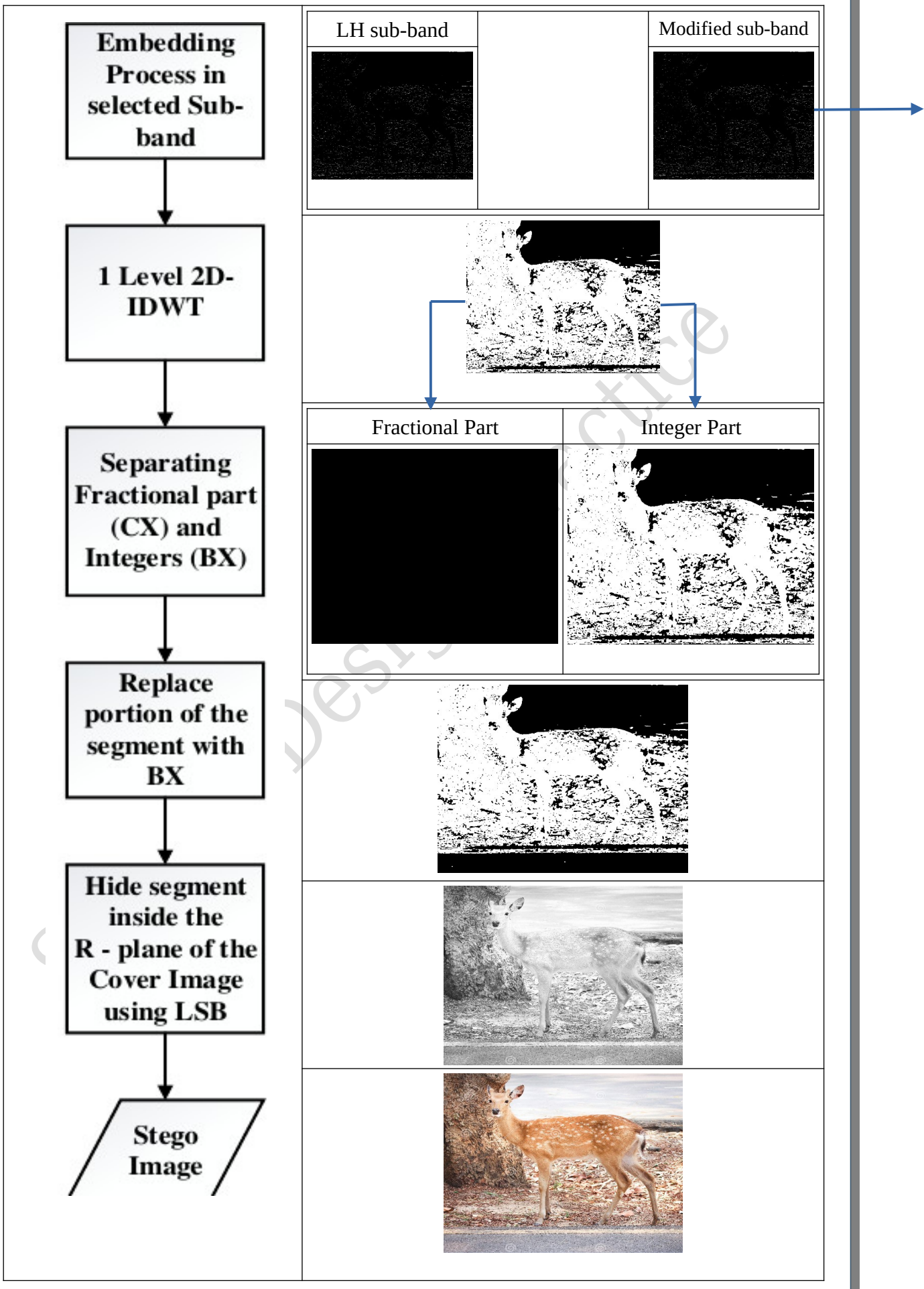
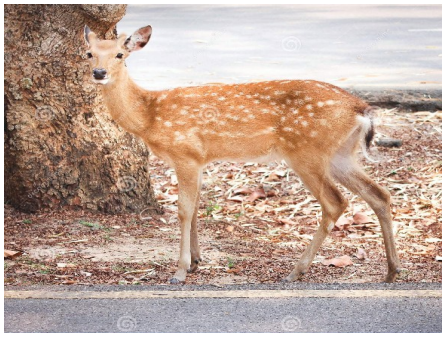
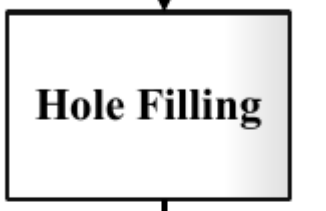
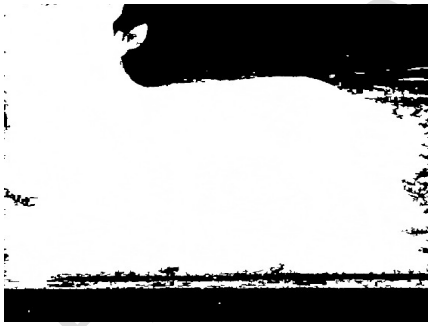
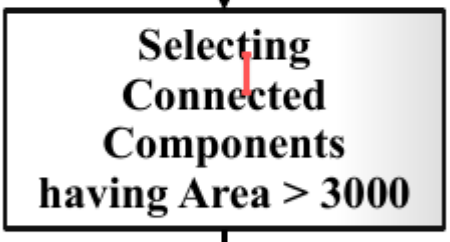

Embedding Process in selected Sub-band
1 Level 2D-IDWT
Separating Fractional part (CX) and Integers (BX)
Replace portion of the segment with BX
Hide segment inside the R - plane of the Cover Image using LSB
Stego Image
LH sub-band
Modified sub-band
Fractional Part
Integer Part

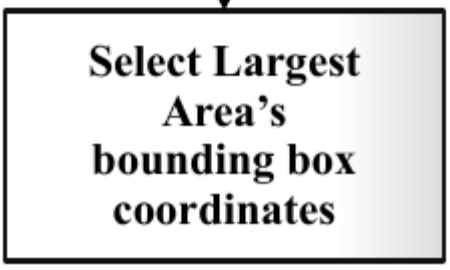
Table 7. 4 Embedding Process for Message 3



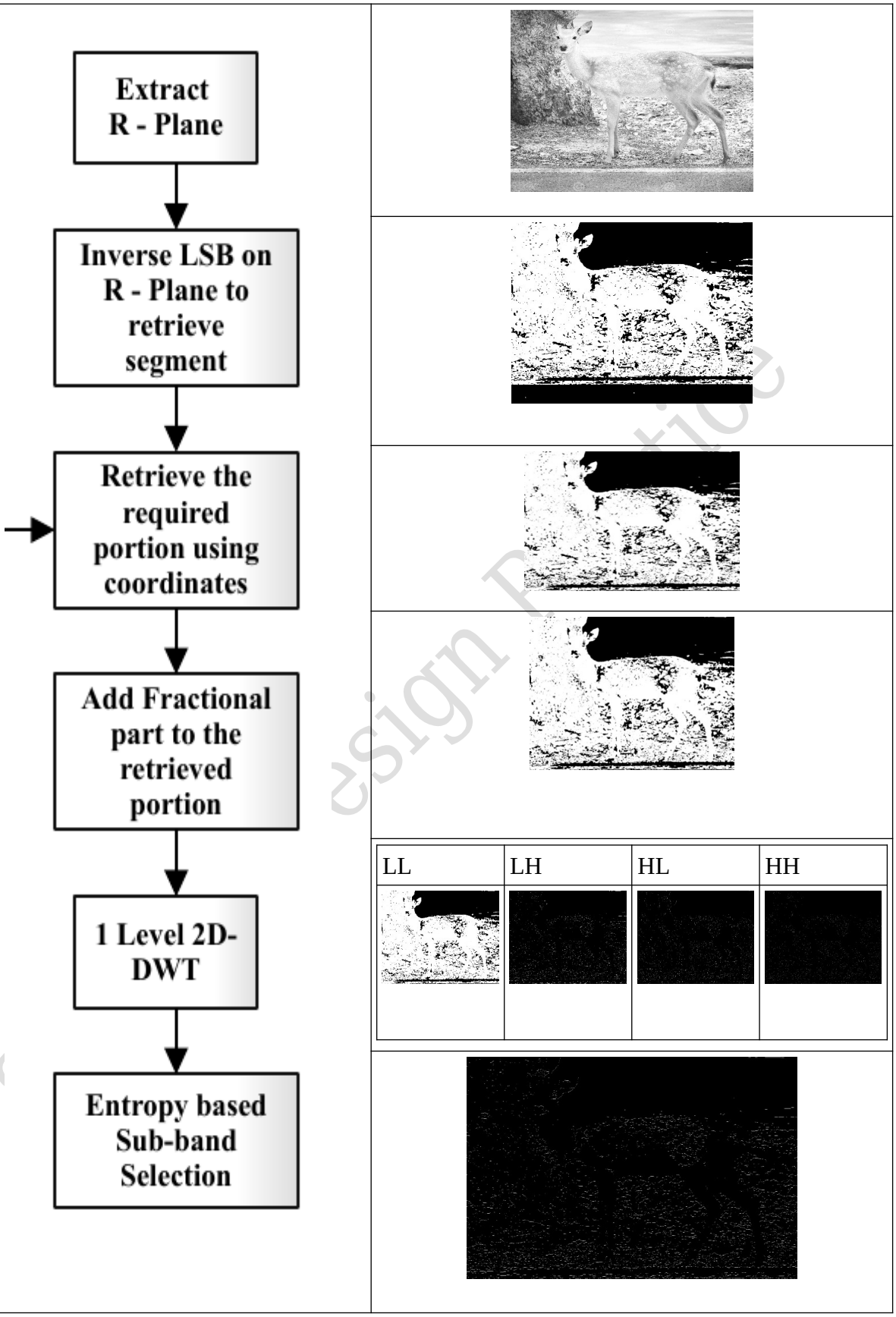




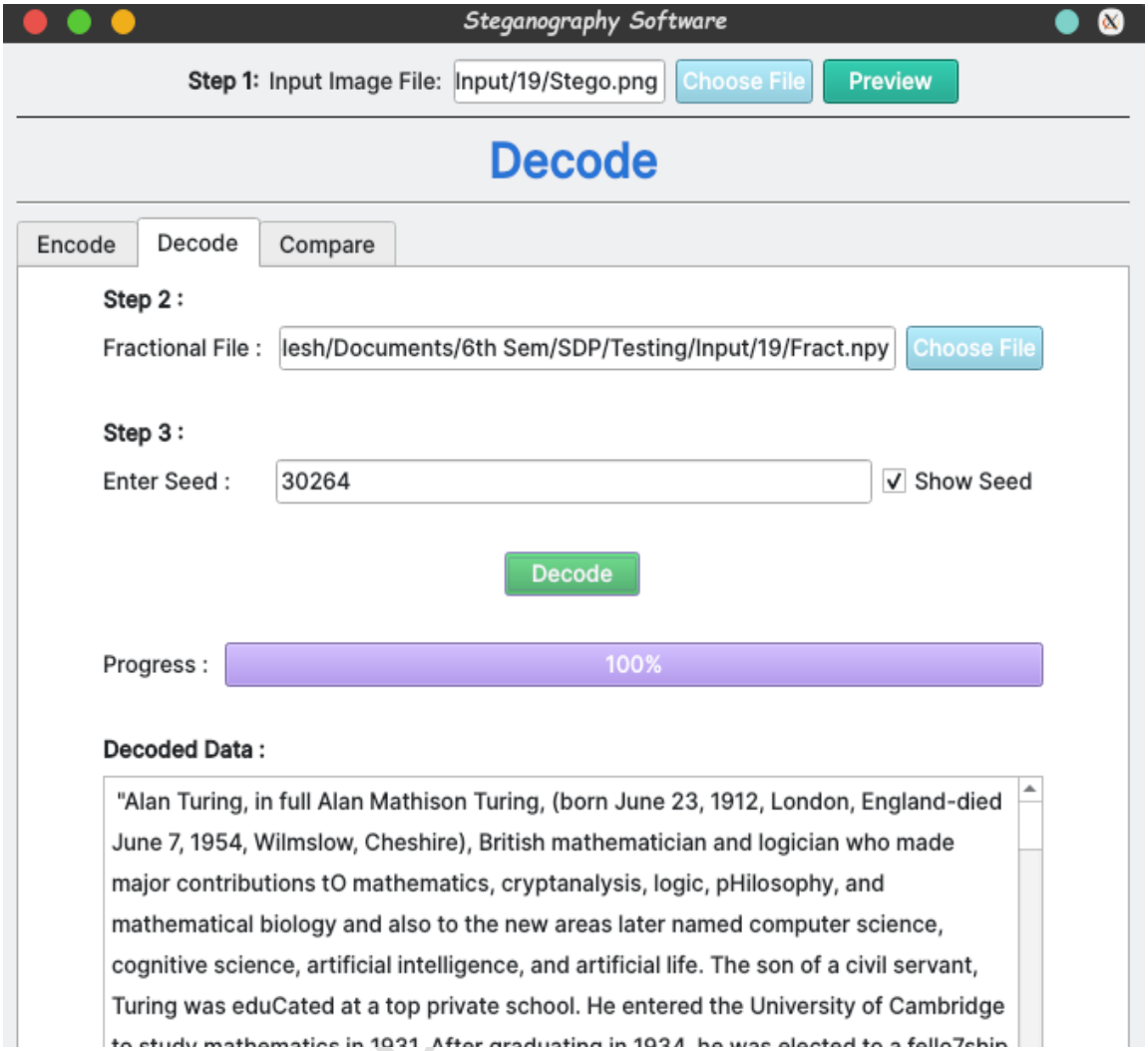




X		Y	
0	1265	0	1599

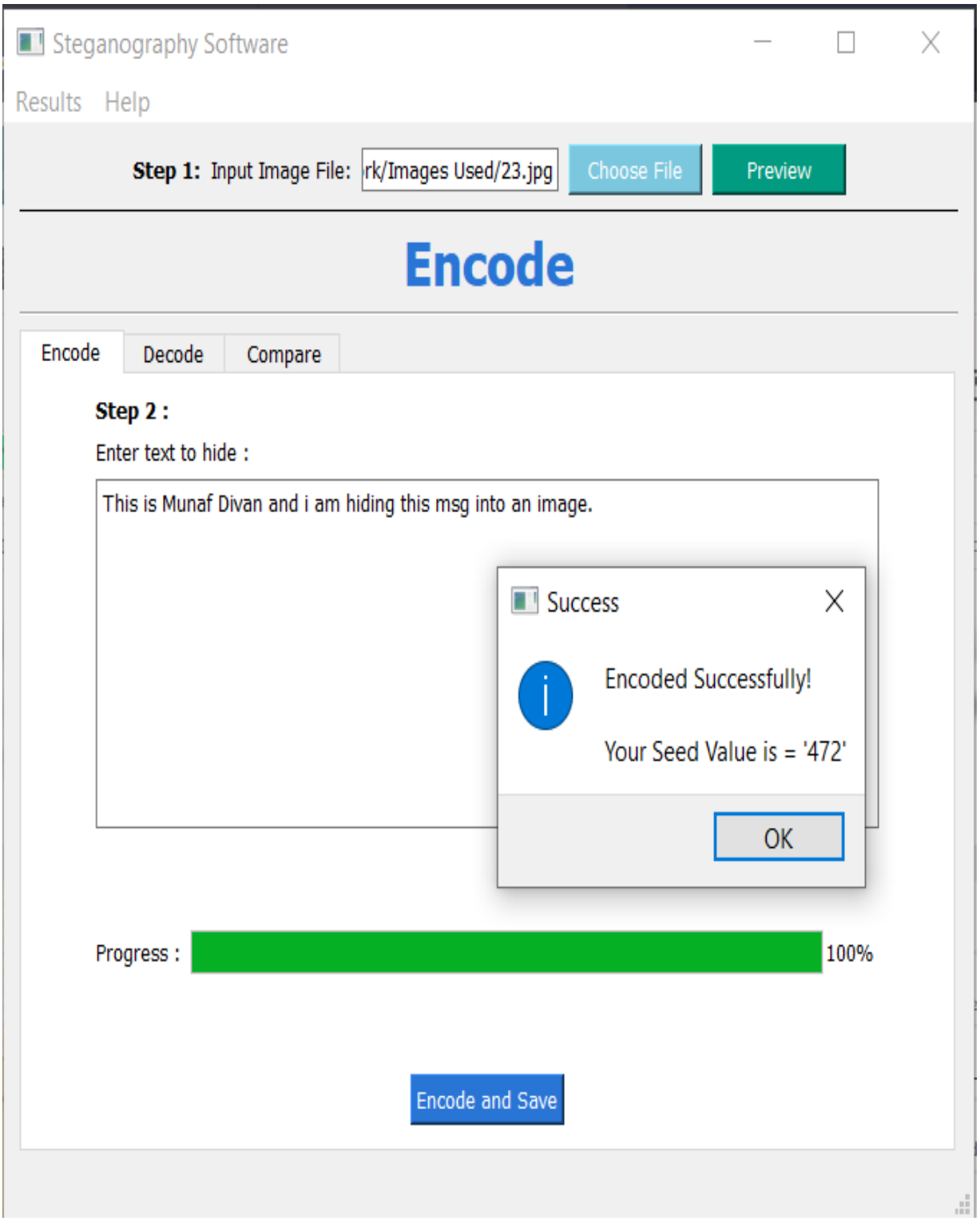

Extract R - Plane
Inverse LSB on R - Plane to retrieve segment
Retrieve the required portion using coordinates
Add Fractional part to the retrieved portion
1 Level 2D-DWT
Entropy based Sub-band Selection
LL
LH
HL
HH

Extraction Process By using the pairs generated using the seed

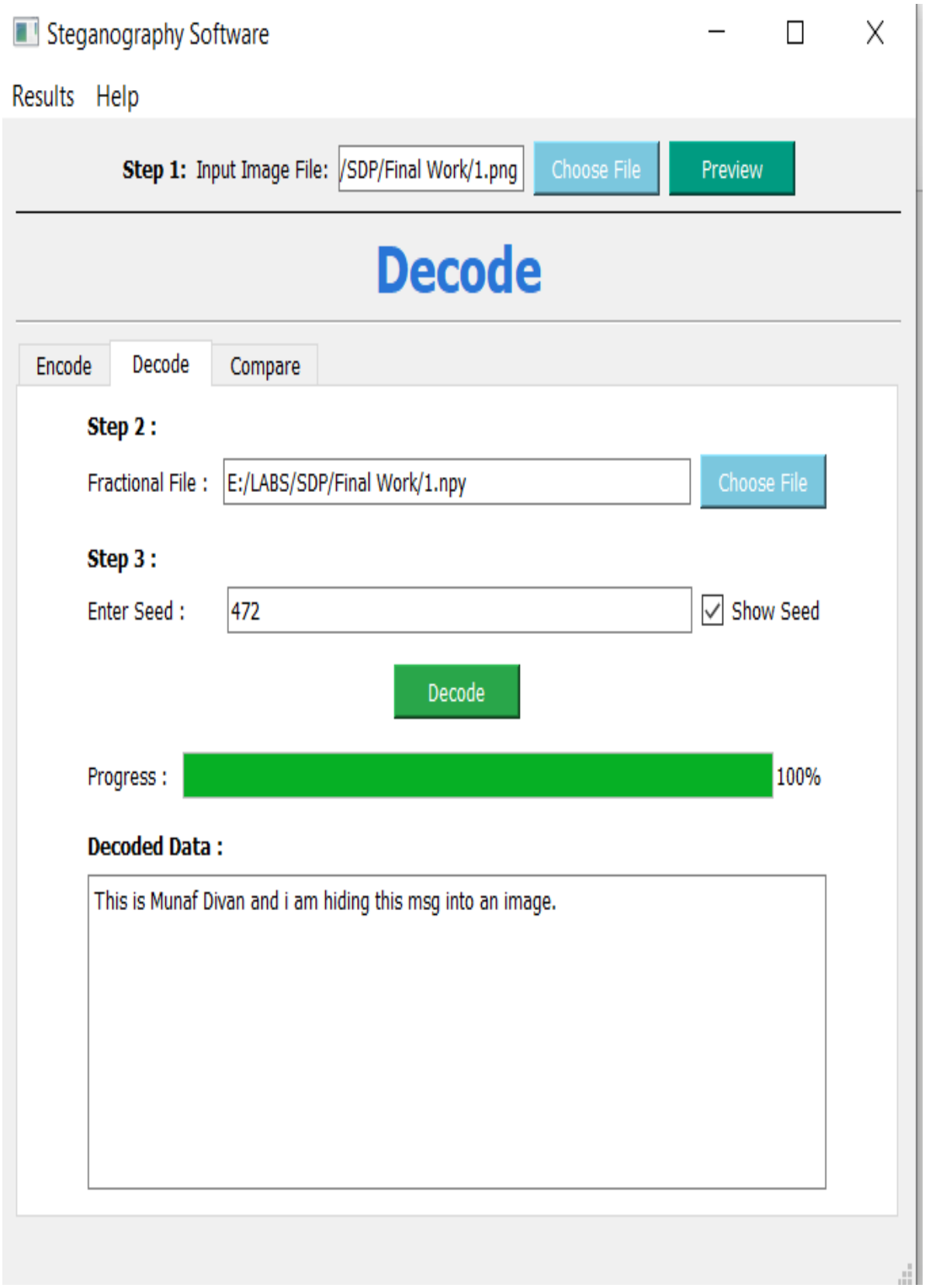


7.2 Embedding and Extracting Using GUI

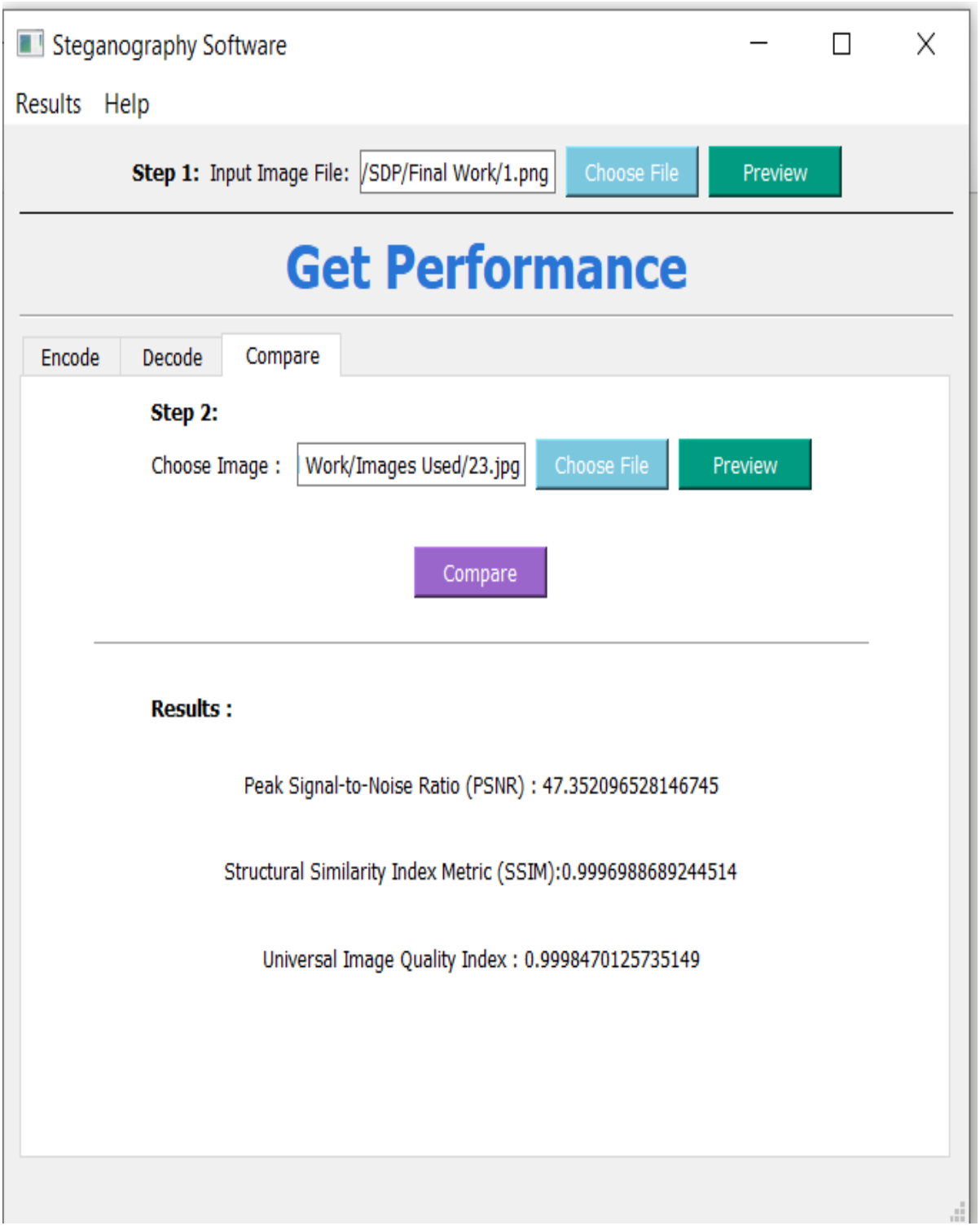
Embedding



Extracting


Steganography Software
Results Help
Step 1: Input Image File: /SDP/Final Work/1.png
Choose File
Preview
Decode
Encode
Decode
Compare
Step 2 :
Fractional File : E:/LABS/SDP/Final Work/1.npy
Choose File
Step 3 :
Enter Seed : 472
Show Seed
Decode
Progress :
100%
Decoded Data :
This is Munaf Divan and i am hiding this msg into an image.

Comparison


Steganography Software
Results Help
Step 1: Input Image File: /SDP/Final Work/1.png
Choose File
Preview
Get Performance
Encode
Decode
Compare
Step 2:
Choose Image : Work/Images Used/23.jpg
Choose File
Preview
Compare
Results :
Peak Signal-to-Noise Ratio (PSNR) : 47.352096528146745
Structural Similarity Index Metric (SSIM):0.9996988689244514
Universal Image Quality Index : 0.9998470125735149

7.3 Details of Cover Images

Table 7. 5 Details of Cover Images

Sr No	Image Name	Entropy Of Subbands	Selected band	Subband Size	Msg. No	Hidden Ratio
1	1.jpeg	LH :0.34597142 HL :0.34252107 HH :0.22058842	LH	146 x 85 = 12410 (1551)	1	8.251410153102336
					2	19.91941982272361
					3	Can Not Be Hidden
2	2.png	**Skin Can Not be Detected**				
3	3.jpg	LH :0.17221861 HL :0.19192905 HH :0.11824127	HL	343 x 531 = 182133 (22766)	1	0.562226504806926
					2	1.357249921760471
					3	16.51540357870347
4	4.jpg	LH :0.0742927851 HL :0.0931736704 HH :0.0514357852	HL	1263 x 1018 = 1285734 (160716)	1	0.079643223248354
					2	0.192263718622981
					3	2.322719940516467
5	5.png	LH :0.22824801 HL :0.25214131 HH :0.16090277	HL	254 x 143 = 36322 (4540)	1	2.819228016078
					2	18.57926325642
					3	37.92632564286
6	6.jpg	LH :0.33631720 HL :0.39780161 HH :0.22530243	HL	271 x 310 = 84010 (10501)	1	1.218902511605761
					2	2.942506844423283
					3	35.80526127841924
7	7.jpg	LH :0.15501503 HL :0.10958742 HH :0.08319961	LH	92 x 170 = 15640 (1955)	1	6.547314578005115
					2	15.80562659846547
					3	Can Not Be Hidden
8	8.jpg	LH :0.38878449 HL :0.39998170 HH :0.24570093	HL	147 x 88 = 12936 (1617)	1	7.915893630179345
					2	19.10946196660482
					3	Can Not Be Hidden
9	9.jpg	LH: 0.05043241984 HL: 0.05882194934 HH: 0.03852028375	HL	1231 x 1059 = 1303629 (162953)	1	0.078549955547168
					2	0.189624502063086
					3	2.29083581294985
10	10.jpg	LH :0.39295133859 HL :0.39884444322 HH :0.25707398099	HL	75 x 72 = 5400 (675)	1	18.96046196660482
					2	45.77777777777778
					3	Can Not Be Hidden
11	11.jpg	LH :0.14081465817 HL :0.07002356296	LH	98 x 240 = 23520	1	4.35374149659864
					2	10.51020408163265

		HH :0.04610011420		(2940)	3	Can Not Be Hidden
12	12.jpg	LH :0.32953007642 HL :0.27722413685 HH :0.16607319000	LH	203 x 480 = 97440 (12180)	1	1.050903119868637
					2	2.536945812807881
					3	30.87027914614121
13	13.jpg	LH :0.11990073020 HL :0.12035701449 HH :0.07273728031	HL	228 x 191 = 43548 (5443)	1	2.351428308992376
					2	5.676494902176908
					3	69.07320657665105
14	14.jpg	LH :0.1554836271 HL :0.1850335742 HH :0.1018694656	HL	412 x 400 = 164800 (20600)	1	0.621359223300970
					2	1.5
					3	18.252427184466
15	15.jpg	LH :0.54856428124 HL :0.44895810016 HH :0.30309103293	LH	53 x 75 = 3975 (496)	1	25.76100628930817
					2	62.18867924528302
					3	Can No Be Hidden
16	16.jpg	**Skin Component larger than 3000 can't be Found**				
17	17.jpeg	**Skin Area Cannot Be Detected**				
18	18.jpg	LH :0.26816158962 HL :0.27166938745 HH :0.15036007715	HL	381 x 600 = 228600 (28575)	1	0.447944006999125
					2	1.081364829396325
					3	13.06386701662292
19	19.jpg	LH :0.33813434541 HL :0.28602762373 HH :0.19616546646	LH	633 x 800 = 506400 (63300)	1	0.202211690363349
					2	0.488151658767772
					3	5.897314375987362
20	20.jpg	LH :0.36183724545 HL :0.27054430154 HH :0.19105425809	LH	49 x 254 = 12446 (1555)	1	8.227542985698216
					2	19.8618029889121
					3	Can Not Be Hidden
21	21.jpg	LH :0.35729138671 HL :0.40626837356 HH :0.25381846286	HL	153 x 116 = 17748 (2218)	1	5.769664187514086
					2	13.92832995267072
					3	Can Not Be Hidden
22	22.jpg	LH :0.11058705903 HL :0.20485004608 HH :0.05833870943	HL	274 x 457 = 125218 (15652)	1	0.817773802488460
					2	1.974157070069798
					3	23.84960628663610
23	23.jpg	LH :0.7254707354 HL :0.7062247375 HH :0.44328729090	LH	89 x 46 = 4094 (511)	1	25.01221299462628
					2	60.38104543234001
					3	Can Not Be Hidden
24	24.jpeg	LH :0.37508445704 HL :0.46660387325	HL	109 x 68 = 7412	1	13.81543443065299
					2	33.35132218024824

		HH :0.24485639600		(926)	3	Can Not Be Hidden
25	25.jpg	LH :0.26037710290 HL :0.26670187821 HH :0.16980433766	HL	162 x 289 = 46818 (5852)	1	2.18719295997266
					2	5.280020504934
					3	63.78743218420266
26	26.jpg	LH :0.53731005221 HL :0.52321990740 HH :0.34621582907	LH	70 x 87 = 6090 (761)	1	16.81444991789819
					2	40.59113300492611
					3	Can Not Be Hidden
27	27.jpg	**Skin Component larger than 3000 can't be Found**				
28	28.jpg	**Skin Component larger than 3000 can't be Found**				
29	29.jpg	LH :0.35995771874 HL :0.28220121955 HH :0.16963787021	LH	279 x 600 = 20925	1	0.6117084826762246
					2	1.476702508960573
					3	17.83990442054958
30	30.jpg	LH :0.14484994663 HL :0.14838842989 HH :0.09064491390	HL	621 x 765 = 475065 (59383)	1	0.2155494511277404
					2	0.520349846863060
					3	6.2862976645301165

7.4 Performance Metrics

Table 7. 6 Performance Metrics

Image Name	Performance Metrics				
	PSNR	SSIM	UIQI	Embedding Time(sec)	Extracting Time(sec)
1.jpeg	47.2050790578	0.99673412110	0.99980867558	3.475354	1.502115
2.png	-	-	-	-	-
3.jpg	47.4344218836	0.99730012359	0.99965655913	10.830151	5.555525
4.jpg	46.1332117515	0.99687825870	0.99993037285	13.75661	7.1231
5.png	45.4472750264	0.99785080713	0.99983950476	1.559817	0.818293
6.jpg	47.4828522648	0.99726877664	0.99960482139	7.193649	3.121452
7.jpg	47.2080198543	0.99702088986	0.99994501726	2.328070	0.921692
8.jpg	47.7101507396	0.99703844320	0.98732663844	1.813942	0.631138
9.jpg	47.4624213219	0.99500983410	0.99994799862	85.251232	52.061343
10.jpg	47.3989198394	0.99858175187	0.99858151372	4.357786	1.412093
11.jpg	47.7398396050	0.99597414478	0.99982362708	1.153282	0.642122
12.jpg	47.2685825444	0.99595632565	0.99988969805	4.239719	2.258020
13.jpg	45.7008504717	0.99842727859	0.99987550712	1.720362	0.966695

14.jpg	47.4306956267	0.99601306875	0.99995630016	4.697467	3.388616
15.jpg	47.4370485294	0.99786332918	0.99976012805	6.465124	2.162131
16.jpg	-	-	-	-	-
17.jpeg	-	-	-	-	-
18.jpg	47.2428627146	0.99890570528	0.99968987034	7.814843	4.683663
19.jpg	47.4556893599	0.99856348687	0.99996813199	21.847965	17.292678
20.jpg	47.3362543920	0.99777615040	0.99994418466	2.839934	1.430821
21.jpg	47.3420882825	0.99784808407	0.99971940287	4.412093	2.35724
22.jpg	47.3070046508	0.99554122652	0.99994603771	4.534011	2.567133
23.jpg	47.3543048761	0.99969907762	0.99984702531	4.820455	2.489365
24.jpeg	47.3055015804	0.99880302157	0.99836312058	4.164551	1.632640
25.jpg	47.4397952229	0.95409352166	0.88150126457	11.3453	7.321
26.jpg	47.5512152899	0.99867240152	0.99461865506	6.084922	2.617235
27.jpg	-	-	-	-	-
28.jpg	-	-	-	-	-
29.jpg	47.7064270858	0.99836426388	0.99953617396	5.708399	3.684108
30.jpg	47.4649164532	0.99566034705	0.99859449557	38.639227	24.469758

7.5 Discussion of Testing Result

Overall, the system is tested on **30 cover images** with **3 texts** as secret data. There is variation observed in the hidden ratio due to secret data size. Also, as **size** of the **cover image** and **secret data increases time of embedding and extracting increases**.

The **Average value** of Performance Metrics is listed below

1. PSNR : **47.222617136984**
2. SSIM : **0.9956737775832**
3. UIQI : **0.9942269889932**

The Average Embedding time is **10.4421706**s & Average Extracting time is **6.12439904**s**.** While Embedding & Extraction time range from **0.631138**s to **85.251232**s

The **Average Hidden Ratios** for

Message 1: **6.18775015549056**

Message 2: **15.408795937154**

Message 3: **24.5557772220883**

7.6 Limitation

We tested this system thoroughly and while doing so we found **some flows** in it.

- If size of the cover image is greater than **1.5MB,** then our system uses too much ram and as a result we are unable to proceed with embedding process.
- In this system, skin detection algorithm uses **fixed range**s of **HSV** and **YcbCr** color-space for finding skin. As a result, we are unable to find skin in some cover images even if it has skin areas.
- If **ROI** size is smaller than needed for hiding secret data then we can not proceed with embedding.

8. CONCLUSION AND FUTURE WORK

8.1 Conclusion

A new and efficient and secure steganographic method for embedding secret messages into images without producing any major changes has been proposed.

- **Security**

  - The proposed approach uses biometric feature i.e. skin tone regions of pictures in DWT domain for embedding secret data. By embedding secret data in mere bound region and not in whole image security is enhanced.
  - The automatic entropy based sub-band selection, selecting random detail coefficients for hiding secret message bits using a seed that generates random locations has added to the security of the system.

- **Visual Quality**

  - Features obtained from DWT coefficients are utilized for embedding secret data. This increases the quality of stego image because secret messages are embedded in high frequency sub-bands which are less sensitive to human eyes.
  - According to performance measure results, proposed approach provides fine image quality.

- **Cost**

  - The Embedding Time for Image Steganography ranges from 0.83s to 21.84s and Extraction Time ranges from 0.43s to 17.29s.

- **Comparison with Existing system**

  - The existing system considers only single skin area images for hiding single secret data. While the proposed approach considers multiple skin regions for hiding different secret data.
  - In the existing systems the coordinates of the cropped region are sent as a key along with the stego image to the receiver side in order to know the info of cropped image. While in the proposed work there is no need for sharing such a key.
  - Coordinates of the ROI are calculated without need for having any key at receiver side by finding largest skin area

8.2 Future Work

- The proposed system has not proved to be much robust if Attacker knows about seed value. So in future we are planning to take care of this Problem by Encryption of Text. For that we will take passcode from user and taking care of the integrity of the that by using Hashing algorithms like MD5 and SHA. After that we will apply algorithms like DES or AES to encrypt the Data.

- We are also planning to compress the data using Compression library like ZLIB so we can store as much data as possible into cover image
- The proposed system can be tried using other wavelets like daubechies, symlets.
- The proposed System can be useful for Video Steganography. In that we can use Multiple video frames to hide data.
- This proposed System only used 1-LSB to hide data, we can utilize more bits of selected sub-band to increase capacity.

9. BIBLIOGRAPHY

[1] Anjali A. Shejul and Umesh L. Kulkarni, "A Secure Skin Tone based Steganography Using Wavelet Transform", International Journal of Computer Theory and Engineering, Vol. 3, No. 1, February 2011.

[2] Manjare et al., "Steganography Using Concept of Skin Tone Detection ", IJSRET, Vol. 2, Issue 4.

[3] B. Forouzan, "Cryptography & Network Security", Tata McGraw-Hill Publication, Special Indian Edition 2007.

[4] Ankit Rai , "Show different planes of an RGB image in Python" , link

[5] https://stackoverflow.com/questions/57885980/how-to-crop-a-region-of-small-pixels-in-an-image-using-opencv-in-python

[6] https://docs.opencv.org/3.4/df/d0d/tutorial_find_contours.html

[7] Leodanis Pozo Ramos , "Qt Designer and Python: Build Your GUI Applications Faster", Link

[8] https://code.adonline.id.au/structural-similarity-index-ssim-in-python/

[9] https://sewar.readthedocs.io/en/latest/

[10] https://www.geeksforgeeks.org/image-segmentation-using-pythons-scikit-image-module/

[11] https://learnopencv.com/filling-holes-in-an-image-using-opencv-python-c/

[12] https://stackoverflow.com/questions/50313114/what-is-the-entropy-of- an-image-and-how-is-it-calculated

[13] https://stackoverflow.com/questions/47055771/how-to-extract-the largest-connected-component-using-opencv-and-python

[14] Atanu Maity, Project Report on "Steganography", Mumbai University, October 2010.

[15] https://pywavelets.readthedocs.io/en/latest/

[16] Ketan Shah, Swati Kaul and Manoj S. Dhande, “Image Steganography using DWT and Data Encryption Standard (DES)”, IJSR, Volume 3, Issue No. 5, May 2014.