CONTRAST ENHANCEMENT BASED FORENSICS IN IMAGES

Main Project Report

Submitted by

SREETHU T NAIR

Reg no:FIT20MCA-2107

Submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

Of

A P J Abdul Kalam Technological University

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY-683577, ERNAKULAM(DIST)

JULY 2022

DECLARATION

I hereby declare that the report of this project work, submitted to the Department of Computer Applications, Federal Institute of Science and Technology (**FISAT**), Angamaly in partial fulfillment of the award of the degree of Master of Computer Applications is an authentic record of my original work.

The report has not been submitted for the award of any degree of this university or any other university.

Date :

Place: Angamaly **Sreethu T Nair**

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)®

ANGAMALY, ERNAKULAM-683577

CERTIFICATE

This is to certify that the project report titled **" CONTRAST ENHANCEMENT BASED FORENSICS IN IMAGES "** submitted by **SREETHU T NAIR**, **(Reg No: FIT20MCA-2107)** towards partial fulfillment of the requirements for the award of the degree of Master of Computer Applications is a record of bonafide work carried out by her during the year 2022.

Project Guide

Dr. Rakhi Venugopal

Head of the Department

Dr. Deepa Mary Mathews

Submitted for the viva-voce held on ..... at .....

Examiner :

ACKNOWLEDGEMENT

Gratitude is a feeling which is more eloquent than words, more silent than silence. To complete this project work I needed the direction, assistance and co-operation of various individuals, which is received in abundance with the grace of God.

I hereby express my deep sense of gratitude to **Dr. Manoj George** Principal, FISAT and **Dr. C Sheela** Vice principal, FISAT for allowing me to utilize all the facilities of the college.

My sincere thanks to **Dr. Deepa Mary Mathews** HOD, Department of Computer Applications FISAT ,who had been a source of inspiration. I express heartiest thanks to **Dr. Rakhi Venugopal** my internal guide for her encouragement and valuable suggestion. I express my heartiest gratitude to my scrum master **Dr. Shahna K U** and the faculty members in my Department for encouragement and constructive suggestions and comment during the project work.

Finally I wish to express my thanks to my parents, friends and well-wishers who extended their help in one way or other in preparation of my project. Besides all, I thank GOD for everything.

ABSTRACT

The project mainly focus on forensic and anti-forensic part. Steganography is already existing technique. Here detection of image embedding or text embedding is identified and extracted. Along with this the image may undergo any kind of compression variation. This is because of data signal embedding as noise. So the rate of compression done in an image is identified. For doing the forensic part, anti-forensic should be done. That is, hidding of data should be done and then the detection part is performed. Malicious users may also perform contrast enhancement locally for creating a realistic composite image. As such it is significant to detect contrast enhancement blindly for verifying the originality and authenticity of the images. Here, two novel algorithms are used to detect the contrast enhancement involved manipulations in images. First one is to focus on the detection of global contrast enhancement applied to the previously JPEG-compressed images, which are widespread in real applications. The histogram shows the difference between the original image and suspicious image. These difference is been analysed to verify the image authenticity. Second is to identify the composite image created by enforcing contrast adjustment on either one or both source regions. Detection of image that is hidden in another image and the text that is been hidden inside an image is identified easily. The consistency between regional artifacts is checked for discovering the image forgeries and locating the composition boundary. Extensive experiment-shave verified the effectiveness and efficiency of the proposed techniques.

Contents

1 INTRODUCTION **1**

2 PROOF OF CONCEPT **3**

2.1 EXISTING SYSTEM 4

2.2 PROPOSED SYSTEM 5

2.3 OBJECTIVES 6

3 IMPLEMENTATION **7**

3.1 TOOLS USED 9

3.2 SYSTEM ARCHITECTURE 10

3.3 MODULES 12

3.3.1 IMAGE PREPROCESSING 12

3.3.2 DCT HISTOGRAM GENERATION 12

3.3.3 QUANTIZATION MECHANISM 13

3.3.4 COMPARISON OF MATRICES 14

3.3.5 DEFECT DETECTION 14

3.4 DATASET 16

3.5 ALGORITHM 17

3.5.1 DISCRETE COSINE TRANSFORM ALGORITHM(DCT) . . . 17

3.5.2 DISCRETE WAVELET TRANSFORM ALGORITHM(DWT) . . 18

4 RESULT ANALYSIS **19**

4.1 RESULTS 20

4.1.1 IMAGE COMPRESSION 20
4.1.2 IMAGE EMBEDDING 22

5 CONCLUSION AND FUTURE SCOPE **23**
5.1 CONCLUSION 23
5.2 FUTURE SCOPE 24

6 APPENDIX **25**
6.1 SOURCE CODE 26
6.2 SCREENSHOTS 39

7 REFERENCES **49**

Chapter 1

INTRODUCTION

The project aims with detecting whether the image has undergone any compression or if it is embedded with a file. Forensic department mainly focuses on detecting any kind of malpractices done in the image, whereas the anti-forensic department tries to fool the forensic analyst by hiding the traces of compression. JPEG is the most commonly used image standard. JPEG has a property; it follows lossy compression which does not preserve all the bit values. So it leaves traces after the compression process. This enables the forensic analyst to determine whether the image is anti-forensically compressed by analyzing the histograms of original and suspected images. The histogram of original image exhibits a comb-shape which makes it different from the histogram of original image. Anti-forensic department further works to make the histograms same by adding a noise signal.

The project deals with the forensic department detecting compressed image even if it contains the presence of noise signal. In the applications such as law enforcement and news recording, it is also necessary to verify the originality and authenticity of digital images, and make clear the image manipulation history to get more information. First, extended to detect the global contrast enhancement in both uncompressed and previously JPEG-compressed images. The zero-height gap bins in gray level histograms were novelty exploited as identifying features. Also proposed a new method to discover the both source- enhanced composite image, which invalidated the previous detection methods.

The composition boundary was accurately located by detecting the inconsistency between detected and original images. The tests on both a specific composite image and quantitative synthetic samples verified the efficiency of the proposed composition detector. The proposed contrast enhancement based forensic methods could work particularly well when contrast enhancement is performed as the last step of manipulation.

Chapter 2

PROOF OF CONCEPT

A lot of studies have been made on Contrast Enhancement-Based Forensics in Images in the literature. In the paper ”Digital image forensics” by ”A.Swaminathan” proposed to estimate both in-camera and post-camera operation fingerprints for verifying the integrity of photographs and it is from the journal ”” IEEE Trans. Inf. Forensics Security, vol. 3, no. 1,pp. 101–117, Mar. 2008”.

A set of previous works deal with image manipulation detection by classifier-based approaches. Here three categories of statistical features including binary similarity, image quality and wavelet statistics were developed. ’Cao et al’ designed a new ensemble manipulation detector to simultaneously detect a wide range of manipulation types on local image patches. ’Fan et al’ proposed to correlate statistical noise features with exchangeable image file format header features for manipulation detection. Although such techniques could detect if manipulation occurred, they fail to determine which specific type of manipulation was enforced. There also exist another category of forensic techniques which focus on detecting specific image manipulations. Since each manipulation typically leaves behind unique fingerprints on images, it is feasible to design individual tests to identify each type of enforced manipulation.

2.1 EXISTING SYSTEM

The widespread availability of photo editing software has made it easy to create visually convincing digital image forgeries. To address this problem, there has been much recent work in the field of digital image forensics. There has been little work, however, in the field of anti-forensics, which seeks to develop a set of techniques designed to fool current forensic methodologies. The footprints left by JPEG compression play an important role in detecting possible forgeries, since JPEG is by far the most widely used image compression standard. To achieve lossy compression, a JPEG encoder quantizes each discrete cosine transform coefficient of an image to multiples of a quantization step size, specified by the JPEG quantization matrix. When an image is decoded, the distribution of reconstructed DCT coefficients differs from the original. This fact enables several forensic analysis tasks, including the identification of which camera took a picture, or the detection of double JPEG compression. But it has been shown that the statistical footprints of JPEG compression can be removed by adding a properly designed dithering noise signal to the quantized DCT coefficients of a JPEG-compressed image.

LIMITATIONS OF EXISTING SYSTEM

- Non professional user can produce photo realistic forgeries of original image content.
- Forgeries can be used for malicious purposes such as employed as evidence in court rooms.
- Digital pictures after making certain types of alterations or modifications (such as cropping, straightening verticals, adjusting colors and gamma etc.) are routinely handed to news editors as part of event coverage.

2.2 PROPOSED SYSTEM

To limit the hazards connected to misuse of digital images, in the past few years, a variety of digital image forensic techniques have been proposed by the forensic community. Forensic techniques analyze the image content in order to find traces left by specific acquisition, coding or editing operations. On the observation that the anti-forensic dither is a noisy signal which cannot replace the image content lost during quantization. As that, it introduces visible distortion in the attacked image, which appears as a characteristic grainy noise that allows to discriminate attacked images from original uncompressed images. These traces can be analysed in terms of distortion introduced in the tampered image.

Experiments demonstrate that it is possible to correctly detect attacked images with an accuracy equal to 93 percentage which rises above 99 percentage when excluding the case of nearly lossless JPEG compression. These results indicate that removing JPEG compression footprints is not as simple as previously thought, since the process of footprint removal inevitably introduces new traces in the doctored image.

ADVANTAGES OF PROPOSED SYSTEM

- Attacked images can be correctly detected with an accuracy in the range of 93 percentage to 99 percentage.
- Various hazards connected to the misuse of digital images can be minimized

2.3 OBJECTIVES

- The main objective of the project is that the loopholes in the existing forensics methods can be identified and hence forensics methods can be improved.
- By doing so, it can be made aware of which forensics techniques are capable of being deceived, thus preventing altered images from being represented as authentic and allowing forensics examiners to establish a degree of confidence in their findings.
- A large number of image forensics methods are available which are capable of identifying image tampering. But these techniques are not capable of addressing the anti-forensic method which is able to hide the trace of image tampering. To overcome this problem, anti-forensics method for digital image compression has been proposed.
- To obtain the authenticity of anti-forensics method, the capabilility of removing the traces of image compression is been proposed.
- Additionally, technique is also able to remove the traces of blocking artifact that are left by image compression algorithms that divide an image into segments during compression process.
- This method is targeted to remove the compression fingerprints of JPEG compression.

Chapter 3

IMPLEMENTATION

Contrast Enhancement Based Forensic in Images is an user friendly web application system. The project involved Image forgery.csv" dataset to train the model. The dataset was taken from the kaggle site. Here the project is done as two phase; forensic part and anti-forensic part. In anti-forensic part, an image is taken for compressing and for hidding data or text. And in forensic part, the compression rate undergone in an image is been identified. The hidden data is been identified and retrieved also.

During compression, image is been uploaded and preprocessed. Split the original uncompressed image in pixels. Apply a DCT to pixels, thus removing redundant image data. The image data is divided up into 8*8 blocks of pixels. From this point on each color component is processed independently, so a pixel means a single value, even in a color image. A DCT is applied in each 8*8 blocks. DCT converts the spatial image representation into a frequency map, While successful higher order terms represent the strength of more and more rapid changes across the width or height of block. The highest AC term represents the strength of a cosine wave alternating from maximum to minimum at adjacent pixels. The DCT calculation is fairly complex in fact this is the most costly step in JPEG compression. Discard high frequency data easily without losing low frequency information. The DCT step itself is a lossless except for round off errors. Thus corresponding DCT coefficient block can be obtained. Quantize each block of DCT coefficients. Perform one to one division and round off.

Encode the resulting coefficients of image data. For the generation of the histogram RGB color values of concerned image is considered. It is transformed into luminance. The color space transformation is performed on pixel by pixel basis. DCT coefficient calculated value can be plotted with the DCT coefficient frequency. The histogram of a gray-scale image consists of a discrete array of bins, each representing a certain gray-level range and storing the number of pixels in the image whose gray-level falls into that range. In other words, it defines a discrete function that maps a gray-level range to the frequency of occurrence in the image. To discard appropriate amount of information, each DCT output value is divided by quantization coefficient. If quantization coefficient is large, more data is lost because actual DCT value is represented less accurately. Each of 64 positions of DCT output block has its own quantization coefficient. Quantization mechanism that controls the quality setting of most JPEG compressors.

In this step with the comparison of matrices, can find the incorrectly classified images. It can be done with comparing of image properties, histogram values, DCT comparison, noise presence etc. Image properties can have the comparison with the dimension, extension of the images. In histogram comparison RGB values are compared, antiforensically it would be difficult to find this comparison. Next DCT values are compared, the original DCT values and modified DCT values are compared and find the differences. Finally check the presence of noise, if the image is anti forensically treated there would be differences in the image matrices values, otherwise the image is original. Free space can be identified for compression. During data embedding, image is divided into matrix inorder to identify the RGB of the image. Blue region is identified to embed the data. If the image is embedded inside in the blue region it cannot be identified easily. Using forensic method, the embedded image or text is been identified and extracted.

3.1 TOOLS USED

Visual Studio .NET is a Microsoft-integrated development environment (IDE) that can be used for developing consoles, graphical user interfaces (GUIs), Windows Forms, Web services and Web applications. Visual Studio is used to write native code and managed code supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight. Visual Studio .NET's code editor supports IntelliSense and code refactoring, while the Visual Studio .NET integrated debugger supports both source and machine-level debugging. Visual Studio .NET includes other built-in tools, like a form designer, which is useful when building GUI applications; a Web designer that creates dynamic Web pages; a class designer that is used to create custom libraries, and a schema designer for database support. Here in this project, following tools are used :

- FRONT END :
  - Visual Studio Dot Net
- BACK END :
  - MS SQL Server

3.2 SYSTEM ARCHITECTURE

The diagram that describes the operation of the system. The project is been divided into two parts. First is anti-forensic part and second is forensic part. In anti-forensic part, there are two parts data embedding and image compression. During image compression, first image is taken for preprocessing. Then Quantization mechanism is applied for finding free space. After that inverse quantization is done. Add noise signal in that free space identified. Apply inverse DCT for image reconstruction. In data embedding part, the image taken for embedding is converted to byte stream and the find its RGB combinations. Find the blue region of the image so that the image can be hide inside that region. For that LSB algorithm is used.

In forensic part, two sections are detection of image compression and detection of data embedding. In detection of image compression, select the original image and suspected image for verifying. Check their image properties and histograms. Check the presence of noise signal by looking the DCT matrices. Then calculate the compression percentage. In data embedding detection, image is taken to identify whether the image contain any data embedded. Both original image and suspected image is taken and calculate its DCT. Quantization process is done for that image. Compare its quantization matrices and find its difference. Retrieve the embedded image.

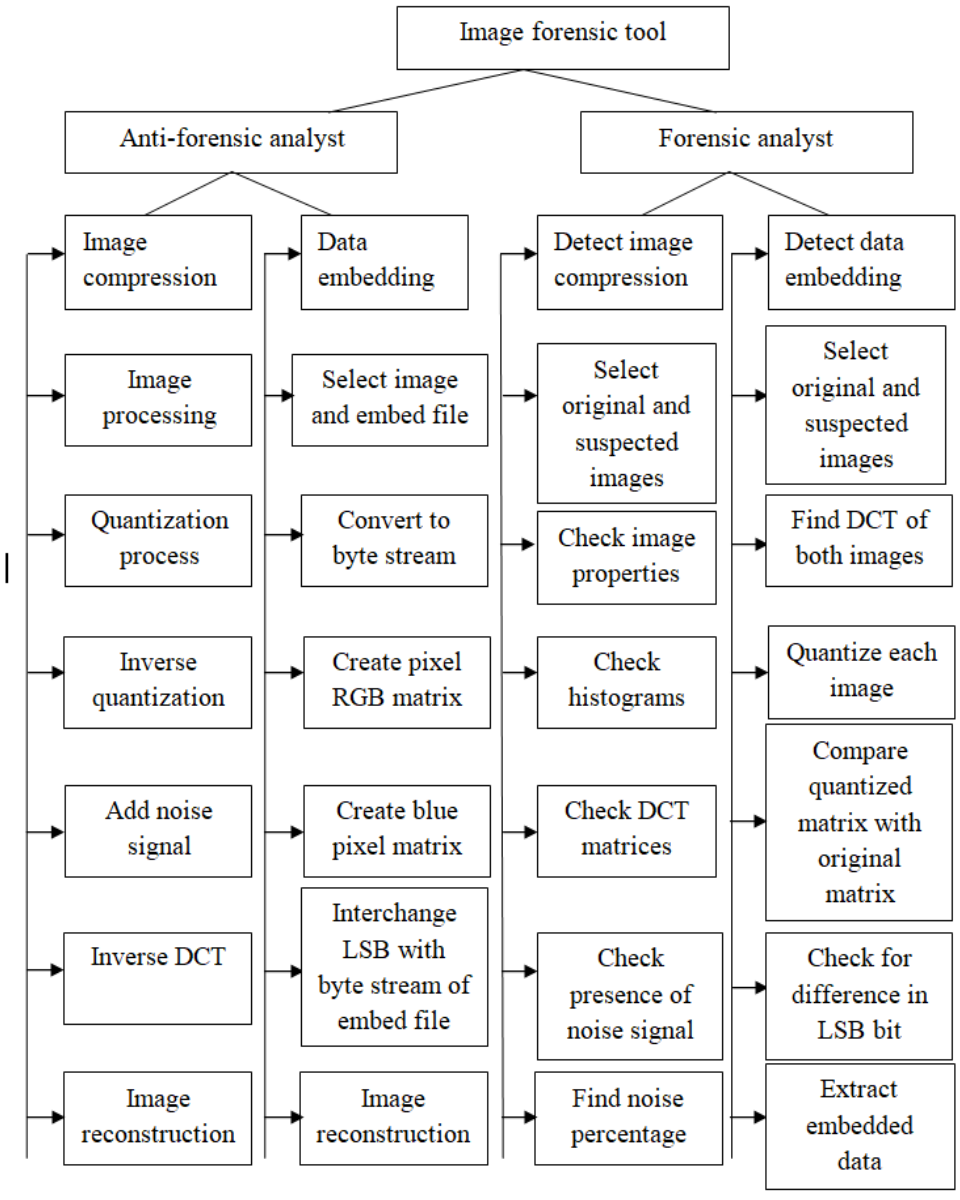

Figure 3.1: Structure or Architecture of the system

3.3 MODULES

3.3.1 IMAGE PREPROCESSING

In image preprocessing, the image is divided into segments for calculating its pixel and matrices. RGB For an image, consider the grayscale image of each picture element. Split the original uncompressed image in pixels. Apply a DCT to blocks of pixels, thus removing redundant image data. The image data is divided up into 8*8 blocks of pixels. From this point on each color component is processed independently, so a pixel means a single value, even in a color image. A DCT is applied in each 8*8 blocks. DCT converts the spatial image representation into a frequency map.

The DCT calculation is fairly complex in fact this is the most costly step in JPEG compression. Discard high frequency data easily without losing low frequency information. The DCT step itself is a lossless except for round off errors. Thus corresponding DCT coefficient block can be obtained. Quantize each block of DCT coefficients. Perform one to one division and round off. Encode the resulting coefficients of image data.

3.3.2 DCT HISTOGRAM GENERATION

Histgram is generated for comparing the image matrices. The difference between the original image and the suspected image can be identified using the histogram. For the generation of the histogram, RGB color values of concerned image is considered. The color space transformation is performed on pixel by pixel basis. DCT coefficient calculated value can be plotted with the DCT coefficient frequency. The histogram of a gray-scale image consists of a discrete array of bins, each representing a certain gray-level range and storing the number of pixels in the image whose gray-level falls into that range. In other words, it defines a discrete function that maps a gray-level range to the frequency of occurrence in the image.

Scatter-based histogram generation consists of two sub-tasks: Bin selection for each input pixel and accumulation of bin contents. It renders one point primitive for each input pixel. Then compute the bin index in the vertex shader and convert it to an output location that maps into our 1D bin texture. The fragment that is rasterized into our desired bin location in the histogram render target is accumulated by configuring the hardware blend units to add the incoming fragment to the contents of the render target. After scattering and accumulating all points in this manner, the output render target will contain the desired histogram. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.

3.3.3 QUANTIZATION MECHANISM

Quantization mechanism is performed for getting free space for compression. To discard appropriate amount of information, each DCT output value is divided by quantization coefficient. If quantization coefficient is large, more data is lost because actual DCT value is represented less accurately. Each of 64 positions of DCT output block has its own quantization coefficient. Quantization mechanism that controls the quality setting of most JPEG compressors. In the JPEG compression standard, the input image is first divided into non overlapping pixel blocks of size 8*8. For each block, the two-dimensional discrete cosine transform is computed.

There is a one-to-one mapping between the index and the position of a coefficient within a DCT block. Each DCT coefficient is quantized with a quantization step size. In many JPEG implementations, it is customary to define as a scaled version of a template matrix, by adjusting a quality factor. For instance, the quantization matrices adopted by the Independent JPEG Group which are obtained by properly scaling the image-independent quantization matrices of the JPEG standard. The quantization levels are obtained from the original coefficients. The quantization levels are entropy coded and written in the JPEG bit stream. When the bit stream is decoded, the DCT values are reconstructed from the quantization levels. Then, the inverse DCT is applied to each block, and the result is rounded and truncated in order to take integer values. Due to the quantization

process, the dequantized coefficients can only assume values that are integer multiples of the quantization step size.

The energy of anti-forensic dithering is concentrated in the middle DCT frequencies, thus resulting in a grainy noise in the spatial domain. With this fact to select a proper set of DCT coefficients to analyze in order to detect JPEG antiforensics. Quantization table matrices are employed for luminance and chrominance data with chrominance data being quantized more heavily than the luminance data. It allows JPEG to exploit further the eye's differing sensitivity to luminance and chrominance. Quantization mechanism that controls the quality setting of most JPEG compressors.

3.3.4 COMPARISON OF MATRICES

In this step with the comparison of matrices we can find the incorrectly classified images. It can be done with comparing of image properties, histogram values, DCT comparison, noise presence etc. Image properties can have the comparison with the dimension, extension of the images. In histogram comparison RGB values are compared, antiforensically it would be difficult to find this comparison. Next DCT values are compared, the original DCT values and modified DCT values are compared and find the differences. Finally check the presence of noise, if the image is anti forensically treated there would be differences in the image matrices values, otherwise the image is original. It involves two novel algorithms. First, global contrast enhancement detection algorithm and second, identify source enhanced composite image algorithm.

3.3.5 DEFECT DETECTION

In defect detection check the presence of noise, if the image is anti forensically treated there would be differences in the image matrices values, otherwise the image is original. In order to find the originality of image the system performs four types of checking. Initially, it checks the properties such as dimension, size, extension etc of both original and suspected image. If the suspected image is once compressed and then decompressed, then the properties of both images will be same. So the forensic user can't

find any traces of JPEG compression. Then it will check the histogram of both images. It is the traditional way of finding traces left by JPEG compression. If the histograms are equalized by adding dithering signal then it is not possible to find the originality of image using this checking. In next step the system will perform DCT image checking.

If anti-forensic method is used, both DCT will be equalized using noise signal. So the forensic user can't find any changes in DCT. Finally, the forensic user will perform noise detection. Anti-forensic dither is a noisy signal which cannot replace content of the image lost during quantization. This introduces visible distortion in the attacked image, which appears as a characteristic grainy noise that allows to discriminate attacked images from original uncompressed images. If presence of noise is detected then the forensic user can identify the image as an anti-forensically treated image.

3.4 DATASET

Figure 3.2: Dataset image

The image forgery dataset is taken from Kaggle.com. In this project use the ”image-forgery.csv” file. It contain a set of images. Here 30 images are there for processing. Random images can also be taken for doing the project. The image is taken for embedding data, compressing the image and also for the detection phase. Image is preprocessed and its properties is been extracted for futher process.

3.5 ALGORITHM

3.5.1 DISCRETE COSINE TRANSFORM ALGORITHM(DCT)

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. Here in this project it is used for lossy image compression. Large amount of information is stored in very low frequency. It also divide the image into different pixels. Below given is the equation for calculating the DCT of an image.

sum += i n p u t [x , y] * System . Math . Cos (((2 . 0 * x + 1 . 0) / (2 . 0 * N)) * u * System . Math . P I) * System . Math . Cos (((2 . 0 * y + 1 . 0) / (2 . 0 * N)) * v * System . Math . P I)

How it works ?

Step 1. Start.

Step 2. Select the image.

Step 3. Process the image by finding the image properties and forming the histograms (RGB format).

Step 4. Form the RGB matrix (hex value matrix) based on the pixel values.

Step 5. Find the DCT matrix (dividing into 8*8 matrix).

Step 6. Quantize DCT matrix with the standard jpeg quantization matrix.

Step 7. Round off the resultant quantization matrix.

Step 8. Perform inverse quantization (multiplication with standard jpeg quantization matrix) to obtain the modified DCT.

Step 9. Compare modified DCT matrix with the original DCT matrix in order to find the difference and then add a noise signal to nullify the difference.

Step 10.Perform inverse DCT on the matrix to get modified RGB matrix.

Step 12.Stop.

3.5.2 DISCRETE WAVELET TRANSFORM ALGORITHM(DWT)

A discrete wavelet transform (DWT) is a transform that decomposes a given signal into a number of sets. In this project, DWT is used for image preprocessing. When we increases the space the size will get reduced and it will affect the quality of the image. So DWT is used to reduce the size of an image without compromising the quality and hence resolution increases. DWT is also taken for the retrieval of the embedded image. It finds the free space for compression.

How it works ?

Step 1. Start.

Step 2. Select the image..

Step 3. Image preprocessing is done.

step 4. Seperate the image into segments for seperating them into RGB combination.

step 5. Find the free space by reducing the resolution without affecting the quality using DWT transform algorithm.

Step 12.Stop.

Chapter 4

RESULT ANALYSIS

The project was completed successfully in time. This web application can successfully do forensic and antiforensic part using machine learning, image preprocessing, and cryptography. The image compression, image embedding can be done and it can be identified and retrieved easily using this web application.

4.1 RESULTS

4.1.1 IMAGE COMPRESSION

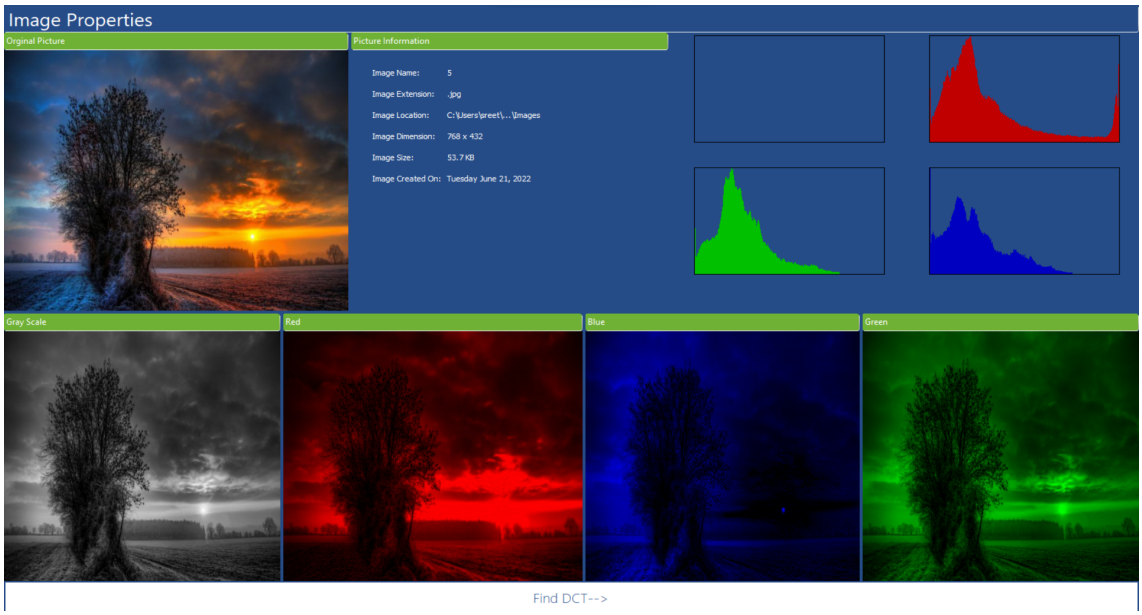

Figure 4.1: Image properties
Here image preprocessing is done inorder to find the RGB combination of the image. Histogram is also generated for showing the RGB combination differences.

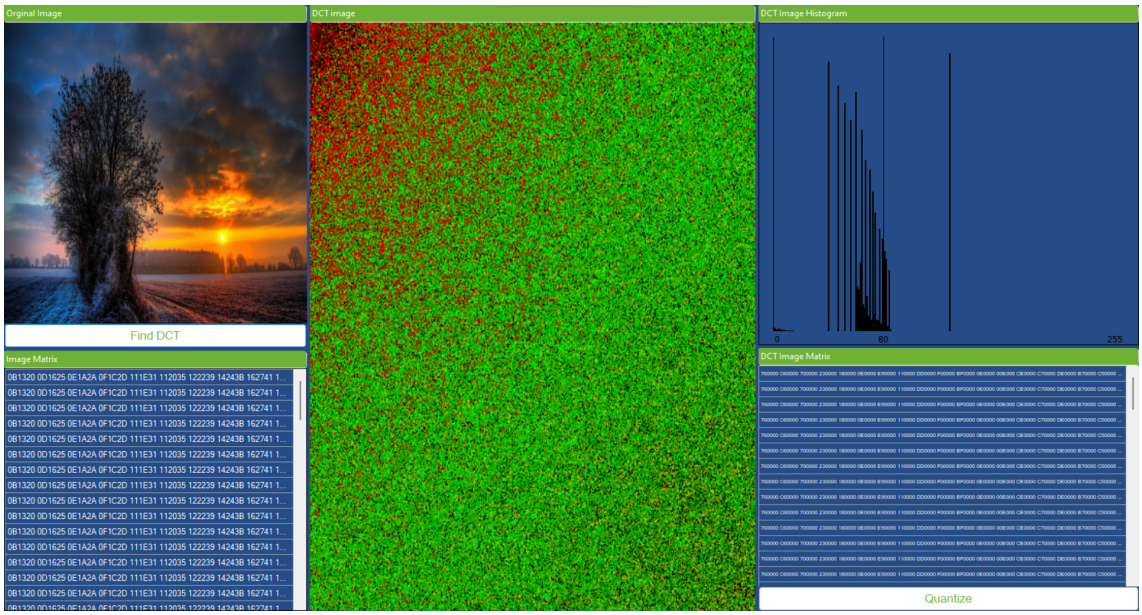

Figure 4.2: DCT Generation
DCT of an image is generated for calculating the image matrices. The histogram shows the property of the DCT image matrices.

Figure 4.3: Quantization mechanism
This is the result of quantization process. Quantization is done for finding the free space for compression.

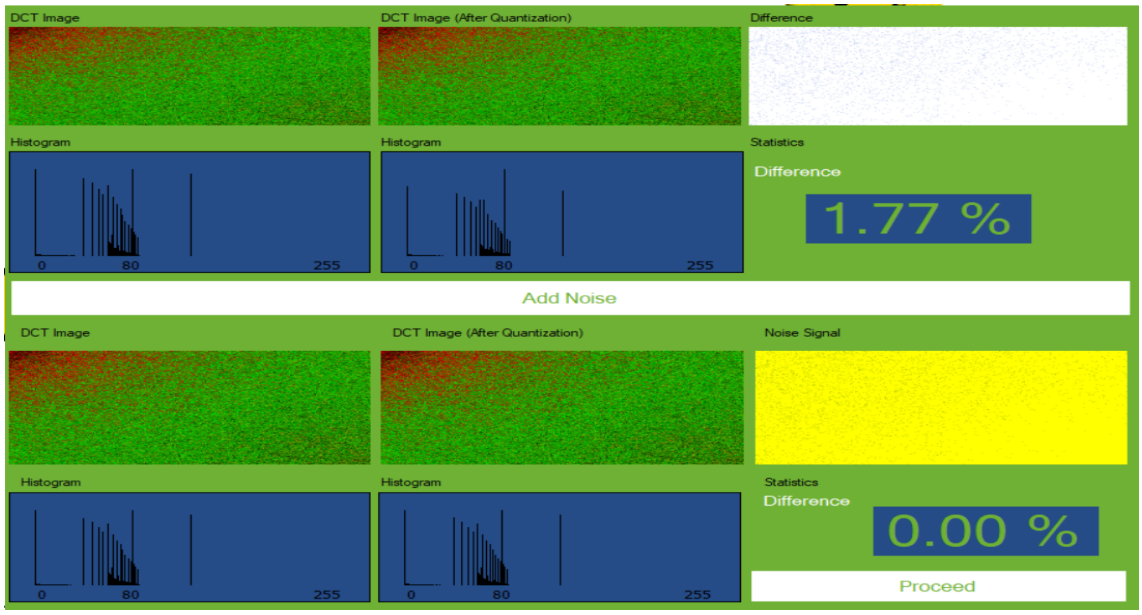

Figure 4.4: Compression
Free space is identified and noise signal is added to the free space. Compression rate is calculated using DCT mechanism.

4.1.2 IMAGE EMBEDDING

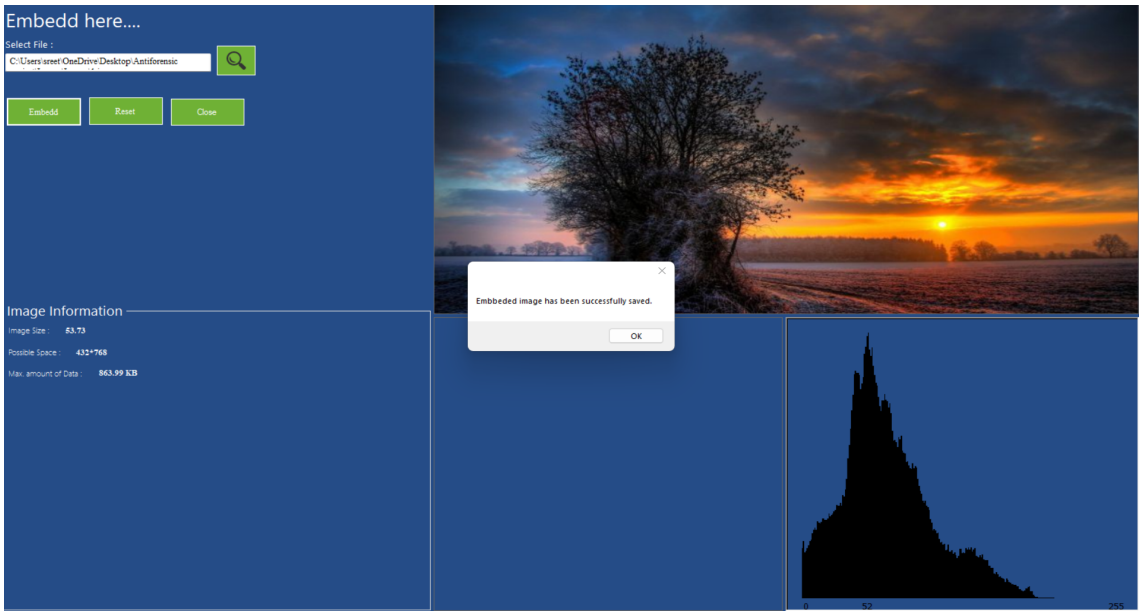

Figure 4.5: Image embedding
This is the result of image embedding. An image is embedding inside another image and saved in a desired location.

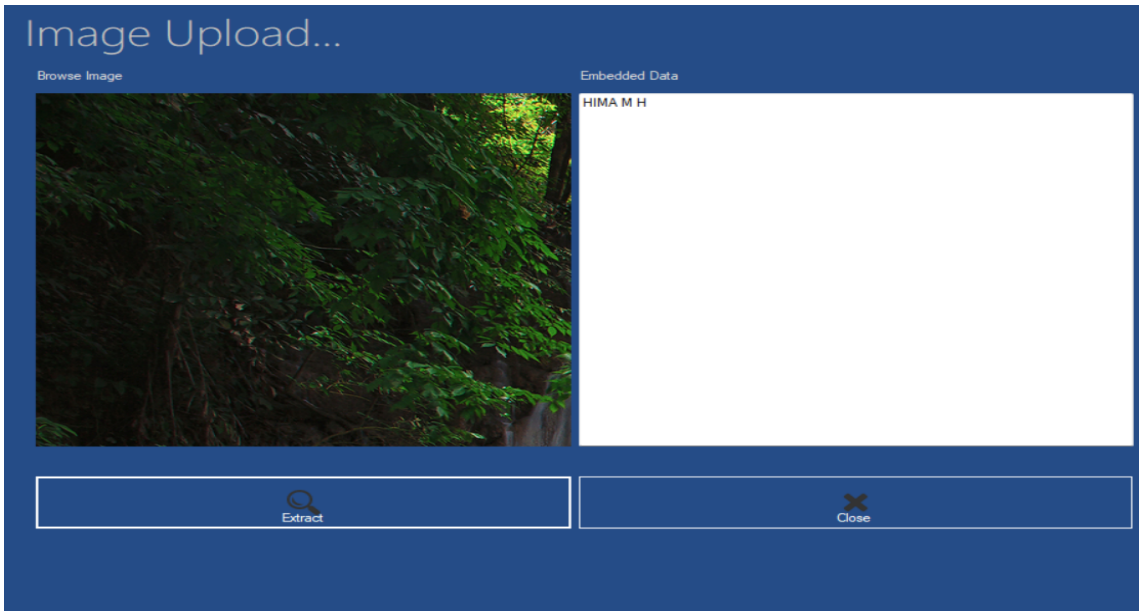

Figure 4.6: Text Embedding
This is the result of text embedding. A text is embedding inside an image and saved in a desired location.

Chapter 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

With the rapid development of digital media editing techniques, digital image manipulation becomes rather convenient and easy. While it benefits to legal image processing, malicious users might use such innocent manipulations to tamper digital photograph images. Currently, image forgeries are widespread on the Internet and other security-related applications such as surveillance and recognition that utilize images are therefore impacted. The proposed contrast enhancement based forensic methods could work particularly well when contrast enhancement is performed as the last step of manipulation. In the future work, we would try to improve the robustness of such methods against postprocessing, such as JPEG compression. It is also essential to enhance the security on countering the existing and potential anti-forensic techniques.

5.2 FUTURE SCOPE

The web application has got a lot of future scope.

- We are in the digital world. We need to know who and how we are being attacked digitally. Therefore, we need to know our security, safety, and privacy features while using safe browsing.
- The scope of forensics includes both cybercrime and physical crime. This is because of the richness of data held in personal devices.
- The future will be fully digitalized especially in the field of busines. The demand for digital forensics will be grown due to various factors. A report projects market size of digital forensics may reach around US dollor 5203 million by 2026 at a CAGR of 9.0 percent during 2021–2026.
- We can improve the robustness of such methods against postprocessing, such as JPEG compression.
- It is also essential to enhance the security on countering the existing and potential anti-forensic techniques.
- Digital forensics help to get evidence and can be produced before a court. This evidence will punish the culprit.

Chapter 6

APPENDIX

6.1 SOURCE CODE

a) Baseconnection.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace ImageForensic
{
    class BaseConnection
    {
        public SqlDataReader dr;
        public DataSet ds=new DataSet();
        public SqlConnection con()
        {
            SqlConnection con = new SqlConnection("server=localhost;
            database=Anti;uid=sa;pwd=yuva");
            con.Open();
            return con;
        }
        public void exec(string str)
        {
            SqlCommand cmd = new SqlCommand(str, con());
            cmd.ExecuteNonQuery();
        }
        public int exec1(string str)
        {
```

```
        SqlCommand cmd = new SqlCommand(str, con());
        return cmd.ExecuteNonQuery();
    }
    public SqlDataReader ret_dr(string str)
    {
        SqlCommand cmd = new SqlCommand(str, con());
        return cmd.ExecuteReader();
    }
    public DataSet ret_ds(string str)
    {
        DataSet ds = new DataSet();
        SqlDataAdapter sqlda = new SqlDataAdapter(str,con());
        sqlda.Fill(ds);
        return ds;
    }
    public string SERVER
    {
        get
        {
            return "LocalHost";
        }
    }
    public string USERNAME
    {
        get
        {
            return "sa";
        }
    }
```

```
        public string PASSWORD
        {
            get
            {
                return "sree";
            }
        }
        public string DATABASE
        {
            get
            {
                return "securestorage";
            }
        }
    }
}
```

b) Discrete Cosine Transform.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Windows.Forms ;
namespace ImageForensic
{
    public static class DiscreteCosineTransform2D
    {
```

```
private static Double[,] InitCoefficientsMatrix(int dim)
{
    Double[,] coefficientsMatrix = new double[dim, dim];
    for (int i = 0; i < dim; i++)
    {
    coefficientsMatrix[i, 0] = System.Math.Sqrt(2.0) / dim;
    coefficientsMatrix[0, i] = System.Math.Sqrt(2.0) / dim;
    }
    coefficientsMatrix[0, 0] = 1.0 / dim;
    for (int i = 1; i < dim; i++)
    {
        for (int j = 1; j < dim; j++)
        {
            coefficientsMatrix[i, j] = 2.0 / dim;
        }
    }
    return coefficientsMatrix;
}
private static bool IsQuadricMatrix<T>(T[,] matrix)
{
    int columnsCount = matrix.GetLength(0);
    int rowsCount = matrix.GetLength(1);
    return (columnsCount == rowsCount);
}
public static Double[,] ForwardDCT(Double[,] input)
{
    double sqrtOfLength = System.Math.Sqrt(input.Length);
    if (DiscreteCosineTransform2D.IsQuadricMatrix<Double>
    (input) == false)
```

```
        {
        throw new ArgumentException("Matrix must be quadric");
        }
        int N = input.GetLength(0);
        double[,] coefficientsMatrix = InitCoefficientsMatrix(N);
        Double[,] output = new Double[N, N];
        for (int u = 0; u <= N-1; u++)
        {
            for (int v = 0; v <= N-1; v++)
            {
                double sum = 0.0;
                for (int x = 0; x <= N-1; x++)
                {
                    for (int y = 0; y <= N-1; y++)
                    {
                    sum += input[x, y] * System.Math.Cos
                    (((2.0 * x + 1.0) / (2.0 * N)) * u *
                    System.Math.PI) * System.Math.Cos
                    (((2.0 * y + 1.0) / (2.0 * N)) * v *
                    System.Math.PI);
                    }
                }
                sum *= coefficientsMatrix[u, v];
                output[u, v] = System.Math.Round(sum);
            }
        }
        return output;
    }
    public static Double[,] InverseDCT(Double[,] input)
```

```
{
    double sqrtOfLength = System.Math.Sqrt(input.Length);
    if (DiscreteCosineTransform2D.IsQuadricMatrix
    <Double>(input) == false)
    {
    throw new ArgumentException("Matrix must be quadric");
    }
    int N = input.GetLength(0);
    Double[,] coefficientsMatrix = InitCoefficientsMatrix(N);
    Double[,] output = new Double[N, N];
    for (int x = 0; x <= N-1; x++)
    {
        for (int y = 0; y <= N-1; y++)
        {
            double sum = 0.0;
            for (int u = 0; u <= N-1; u++)
            {
                for (int v = 0; v <= N-1; v++)
                {
                    sum += coefficientsMatrix[u, v] *
                    input[u, v] * System.Math.Cos
                    (((2.0 * x + 1.0) / (2.0 * N)) * u *
                    System.Math.PI) * System.Math.Cos
                    (((2.0 * y + 1.0) / (2.0 * N)) * v *
                    System.Math.PI);
                }
            }
            output[x, y] = System.Math.Round(sum);
        }
```

```
            }
            return output;
        }
    }
    public class FastDCT2D
    {
        public Bitmap Obj;
        public Bitmap DCTMap;
        public Bitmap IDCTImage;
        public int[,] GreyImage;
        from input Image
        public double[,] Input;
        public double[,] DCTCoefficients;
        public double[,] IDTCoefficients;
        private double[,] DCTkernel;
        int Width, Height;
        int Order;
        public FastDCT2D(Bitmap Input, int DCTOrder)
        {
            Obj = Input;
            Width = Input.Width;
            Height = Input.Height;
            Order = DCTOrder;
            ReadImage();
        }
        public FastDCT2D(int[,] InputImageData, int order)
        {
            int i, j;
            GreyImage = InputImageData;
```

```
        Width = InputImageData.GetLength(0);
        Height = InputImageData.GetLength(1);
        for (i = 0; i <= Width - 1; i++)
            for (j = 0; j <= Height - 1; j++)
            {
                Input[i, j] = (Double)(InputImageData[i, j]);
            }
        Order = order;//Order of Inverse 2D DCT
    }
    public FastDCT2D(double[,] DCTCoeffInput)
    {
        DCTCoefficients = DCTCoeffInput;
        Width = DCTCoeffInput.GetLength(0);
        Height = DCTCoeffInput.GetLength(1);
    }
    private void ReadImage()
    {
        int i, j;
        GreyImage = new int[Width, Height];
        Input = new double [Width, Height];
        Bitmap image = Obj;
        {
            byte* imagePointer1 = (byte*)bitmapData1.Scan0;
            for (i = 0; i < bitmapData1.Height; i++)
            {
                for (j = 0; j < bitmapData1.Width; j++)
                {
                    GreyImage[j, i] = (int)((imagePointer1[0]
                    +imagePointer1[1] + imagePointer1[2]) / 3.0);
```

```
                    Input [j,i]=(double)GreyImage[j,i];
                    imagePointer1 += 4;
                }
                imagePointer1 += bitmapData1.Stride -
                (bitmapData1.Width * 4);
            }
        }
        image.UnlockBits(bitmapData1);
        return;
    }
    public Bitmap Displayimage(double[,] image)
    {
        int i, j;
        Bitmap output = new Bitmap(image.GetLength(0);
        unsafe
        {
            byte* imagePointer1 = (byte*)bitmapData1.Scan0;
            for (i = 0; i < bitmapData1.Height; i++)
            {
                for (j = 0; j < bitmapData1.Width; j++)
                {
                    imagePointer1[0] = (byte)image[j, i];
                    imagePointer1[1] = (byte)image[j, i];
                    imagePointer1[2] = (byte)image[j, i];
                    imagePointer1[3] = 255;
                    imagePointer1 += 4;
                }
                imagePointer1 += (bitmapData1.Stride)
            }
```

```
        }
        output.UnlockBits(bitmapData1);
        return output;
    }
    public Bitmap Displaymap(int[,] output)
    {
        int i, j;
        Bitmap image = new Bitmap(output.GetLength(0),
        output.GetLength(1));
        BitmapData bitmapData1 = image.LockBits;
        unsafe
        {
            byte* imagePointer1 = (byte*)bitmapData1.Scan0;
            for (i = 0; i < bitmapData1.Height; i++)
            {
                for (j = 0; j < bitmapData1.Width; j++)
                {
                    if (output[j, i] < 0)
                    {
                        imagePointer1[0] = 0;
                        imagePointer1[1] = 255;
                        imagePointer1[2] = 0;
                    }
                    else if((output[j, i]>= 0)&&(output[j,i])
                    {
                        imagePointer1[0] = (byte)((output[j,i]);
                        imagePointer1[1] = 0;
                        imagePointer1[2] = 0;
                    {
```

```
                    imagePointer1[0] = 0;
                    imagePointer1[1] = (byte)(output[j,i]);
                    imagePointer1[2] = (byte)(output[j,i]);
                }
                else if ((output[j, i] >= 100)
                {
                    imagePointer1[0] = 0;
                    imagePointer1[1] = 0;
                    imagePointer1[2] = 0;
                }
                else if ((output[j, i] > 255))
                {
                    imagePointer1[0] = 0;
                    imagePointer1[1] = 0;
                }
                imagePointer1[3] = 255;
                imagePointer1 += 4;
            }
            imagePointer1 += (bitmapData1.Stride -
            (bitmapData1.Width * 4));
        }
    }
    image.UnlockBits(bitmapData1);
    return image;
}
public void FastDCT()
{
    double[,] temp = new double[Width, Height];
    DCTCoefficients = new double[Width, Height];
```

```
        DCTkernel = new double[Width, Height];
        DCTkernel = GenerateDCTmatrix(Order);
        temp = multiply(DCTkernel, Input);
        DCTCoefficients = multiply(temp, Transpose(DCTkernel));
        DCTPlotGenerate();
        return;
    }
    public double[,] GenerateDCTmatrix(int order)
    {
        int i, j;
        int N;
        N = order;
        double alpha;
        double denominator;
        double[,] DCTCoeff = new double[N, N];
        for (j = 0; j <= N - 1; j++)
        {
            DCTCoeff[0, j] = Math.Sqrt(1 / (double)N);
        }
        alpha = Math.Sqrt(2 / (double)N);
        denominator = (double)2 * N;
        for (j = 0; j <= N - 1; j++)
            for (i = 1; i <= N - 1; i++)
            {
                DCTCoeff[i, j] = alpha * Math.Cos(((2 * j + 1)
                * i * 3.14159) / denominator);
            }
        return (DCTCoeff);
    }
```

```
        private double[,] multiply(double[,] m1, double[,] m2)
        {
            int row, col, i, j, k;
            row = col = m1.GetLength(0);
            double[,] m3 = new double[row, col];
            double sum;
            for (i = 0; i <= row - 1; i++)
            {
                for (j = 0; j <= col - 1; j++)
                {
                    Application.DoEvents();
                    sum = 0;
                    for (k = 0; k <= row - 1; k++)
                    {
                        sum = sum + m1[i, k] * m2[k, j];
                    }
                    m3[i, j] = sum;
                }
            }
            return m3;
        }
    }
}
```

6.2 SCREENSHOTS

Here I add some sample screenshots of our system,

- Login Screen

- Home Screen

- Image Compression

- Image Embedding

- Text Embedding

- Image Compression Detection

- Detection Of Embedded Image

- Detection of Embedded Text

Figure 6.1: Login Screen

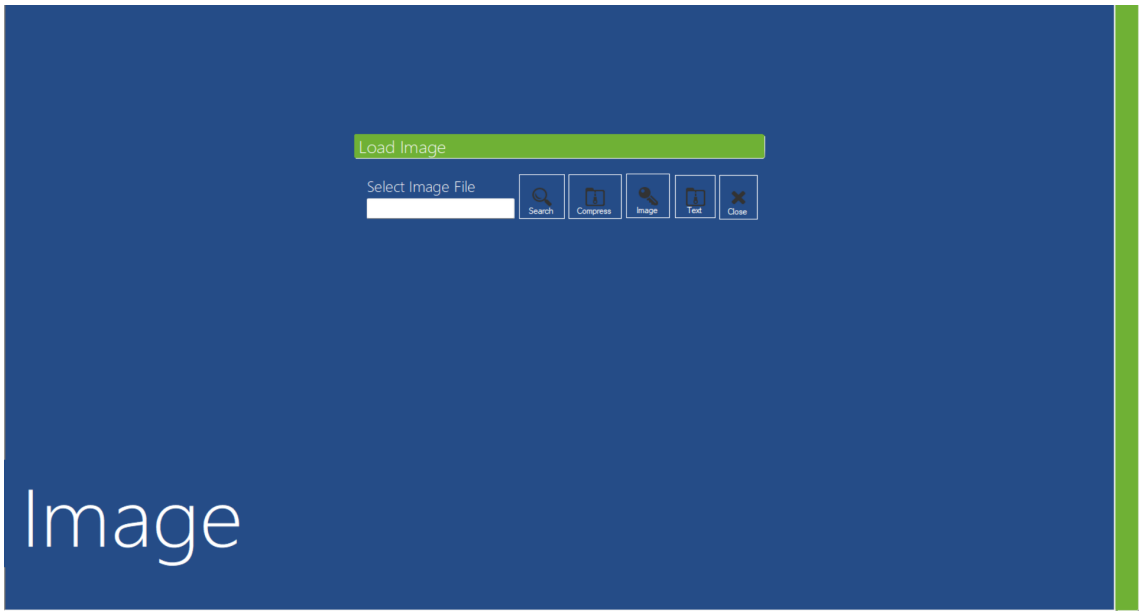

Figure 6.2: Home Screen

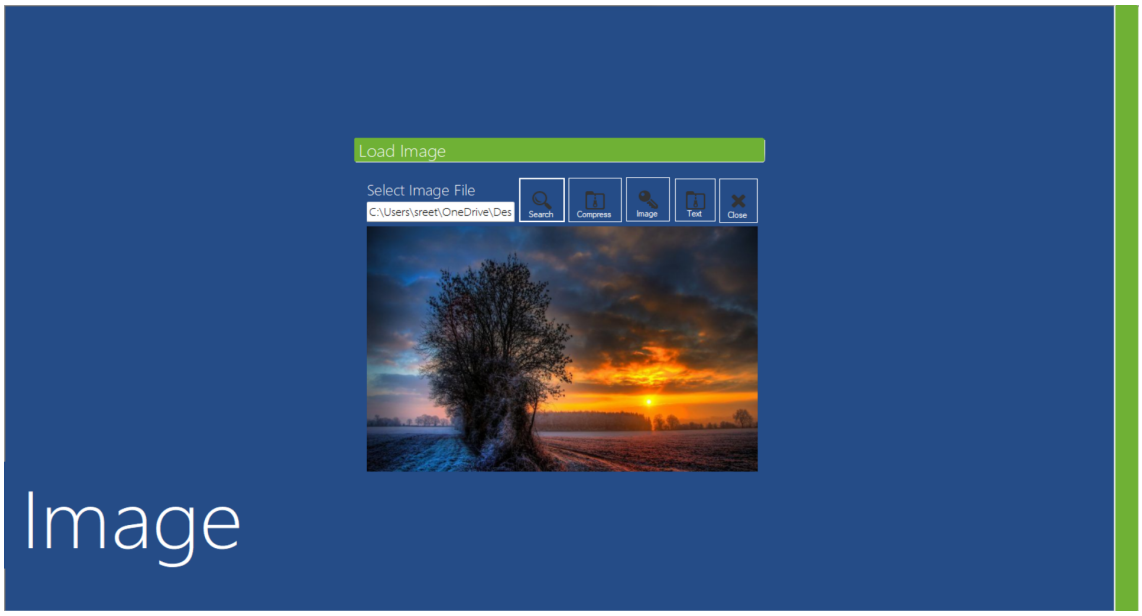

Figure 6.3: Image Uploading

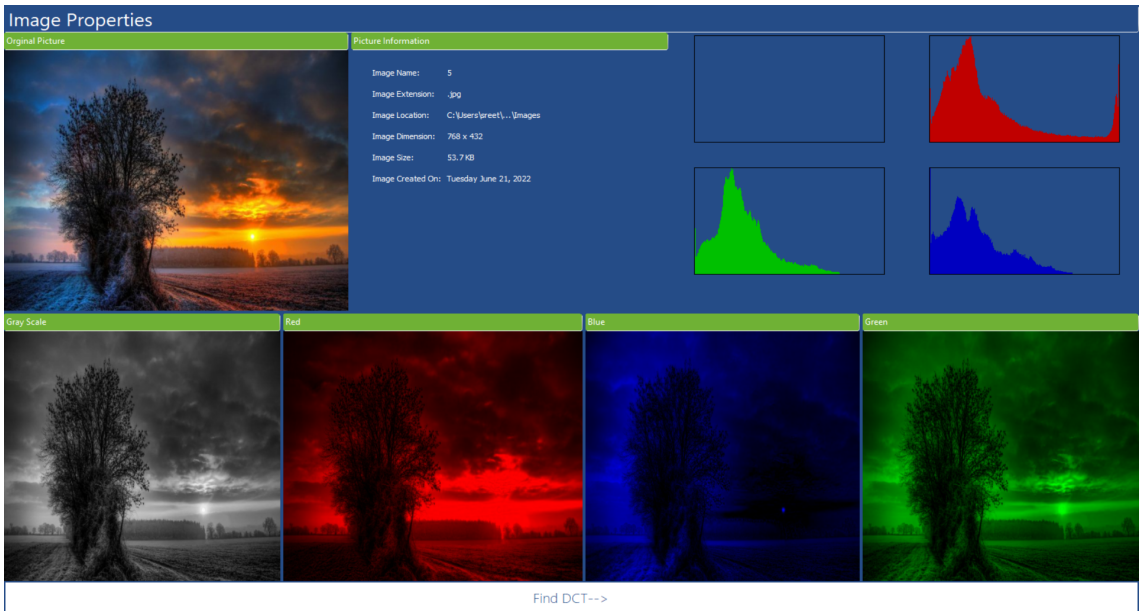

Figure 6.4: Image properties

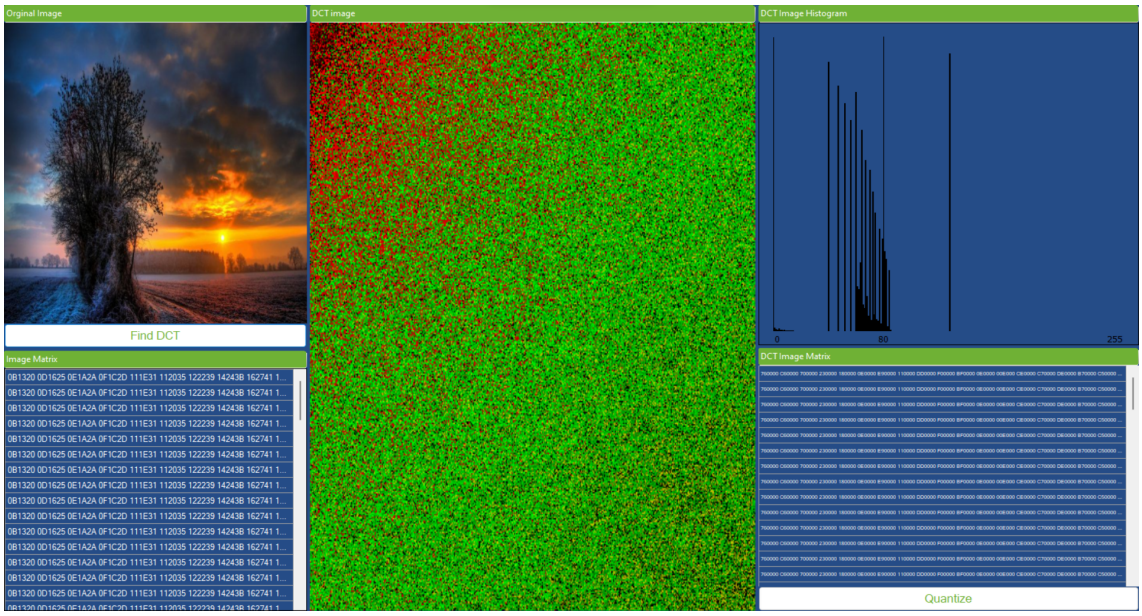

Figure 6.5: DCT Conversion

Figure 6.6: Quantization

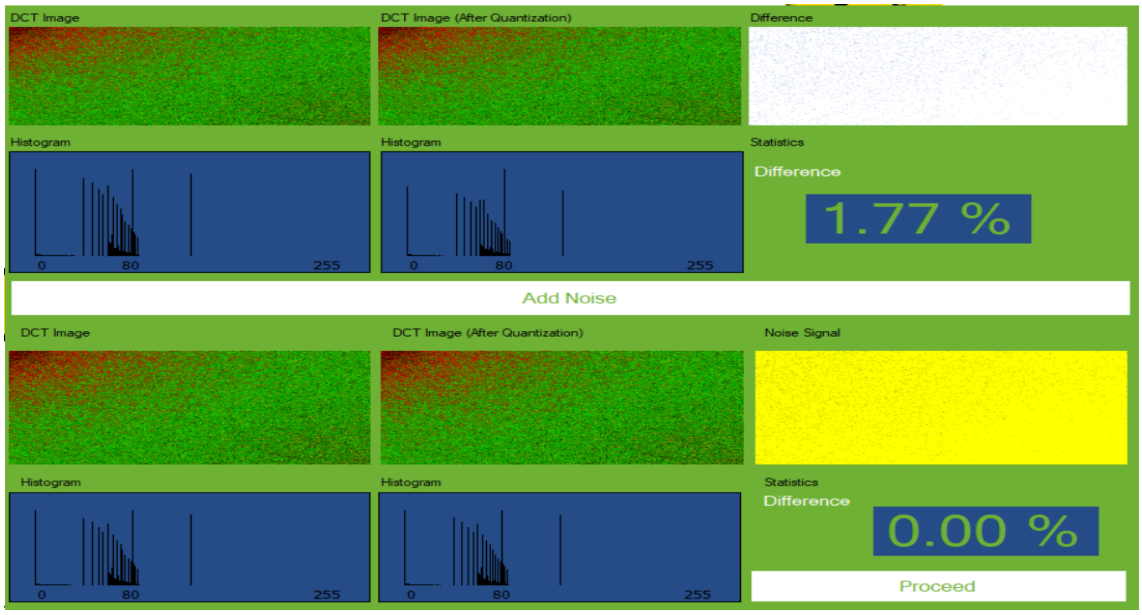

Figure 6.7: Image Compressing and Noise adding

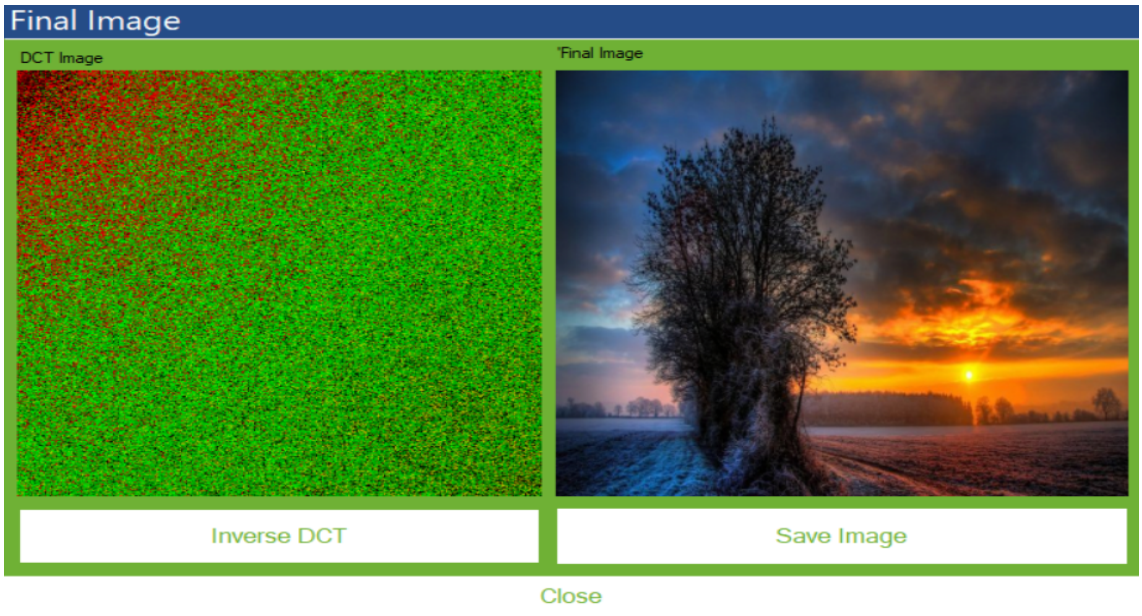

Figure 6.8: Inverse DCT

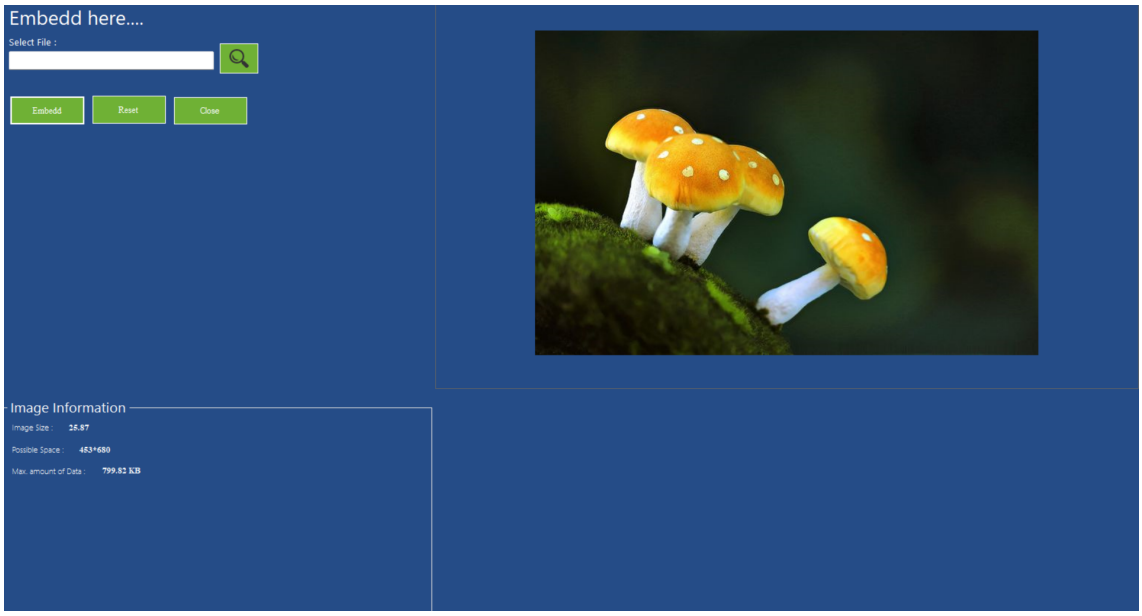

Figure 6.9: Image embedding

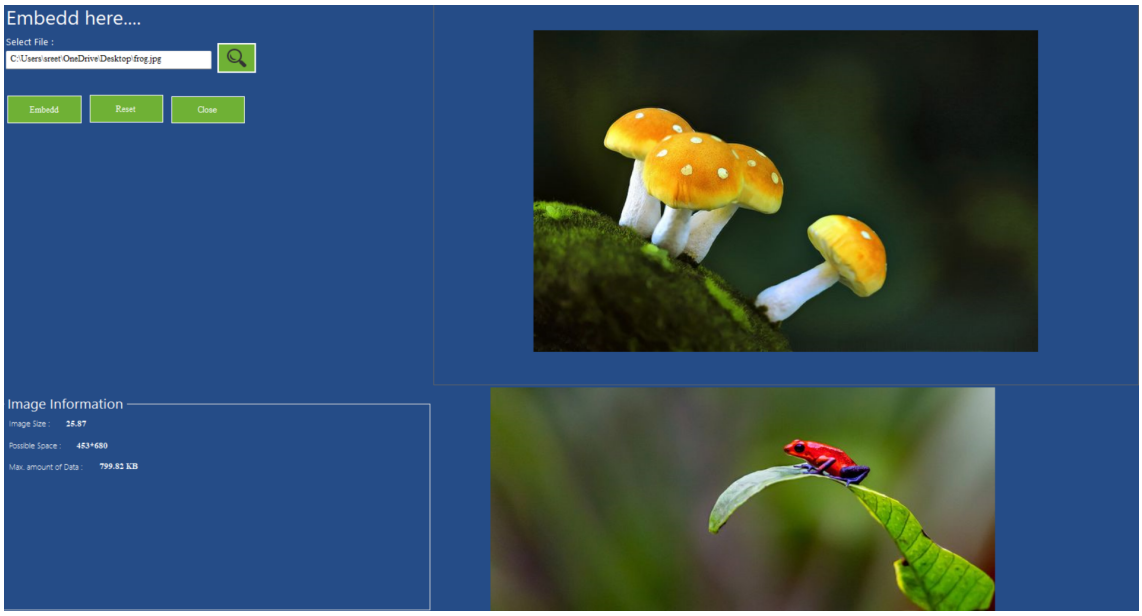

Figure 6.10: Selecting of image that is to be embedded

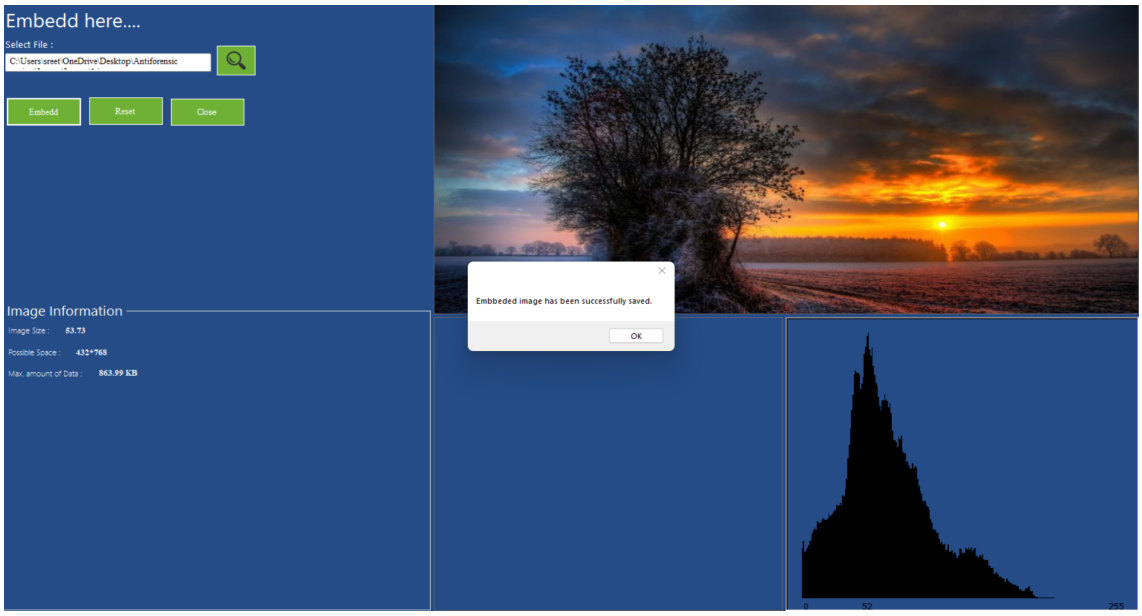

Figure 6.11: Saving the embedded image to desired location

Figure 6.12: Compression Detection

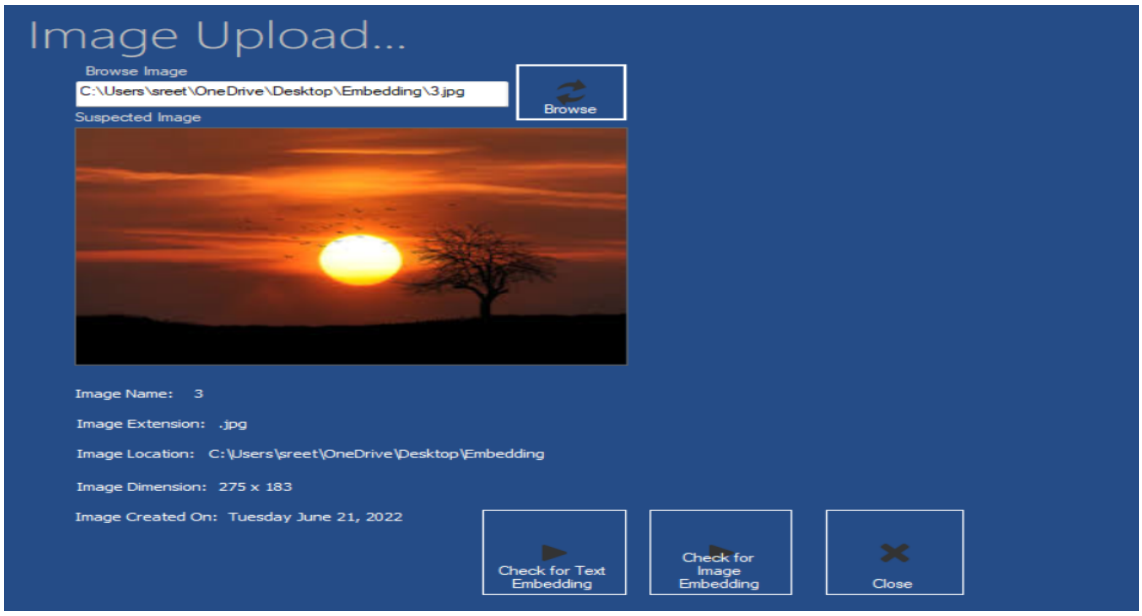

Figure 6.13: Detection of image embedding or text embedding

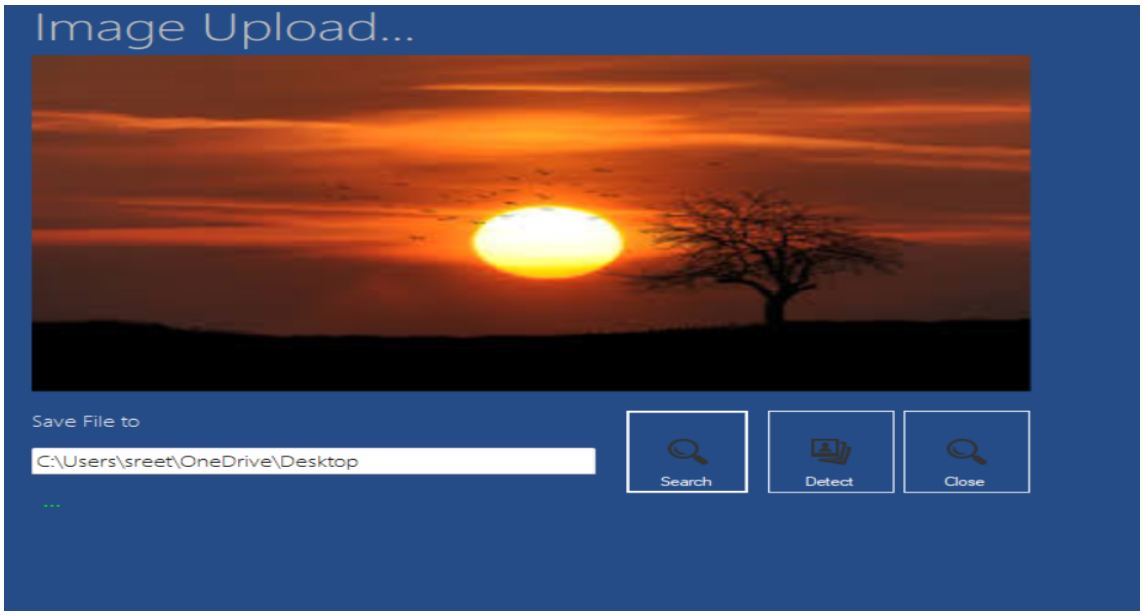

Figure 6.14: Detecting Hidden Image

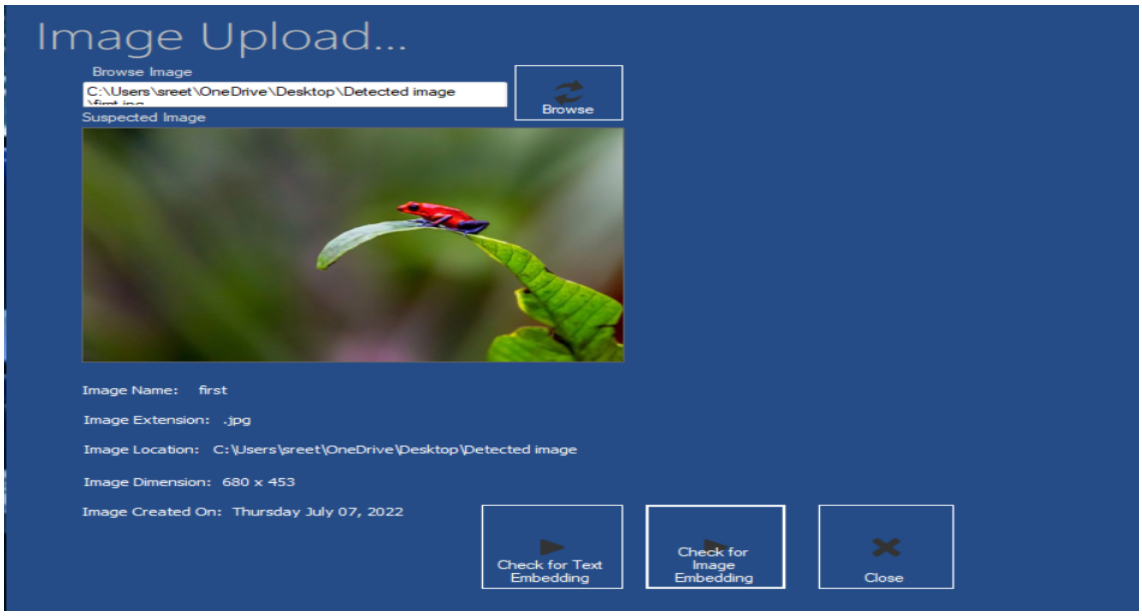

Figure 6.15: Embedding identified and saved

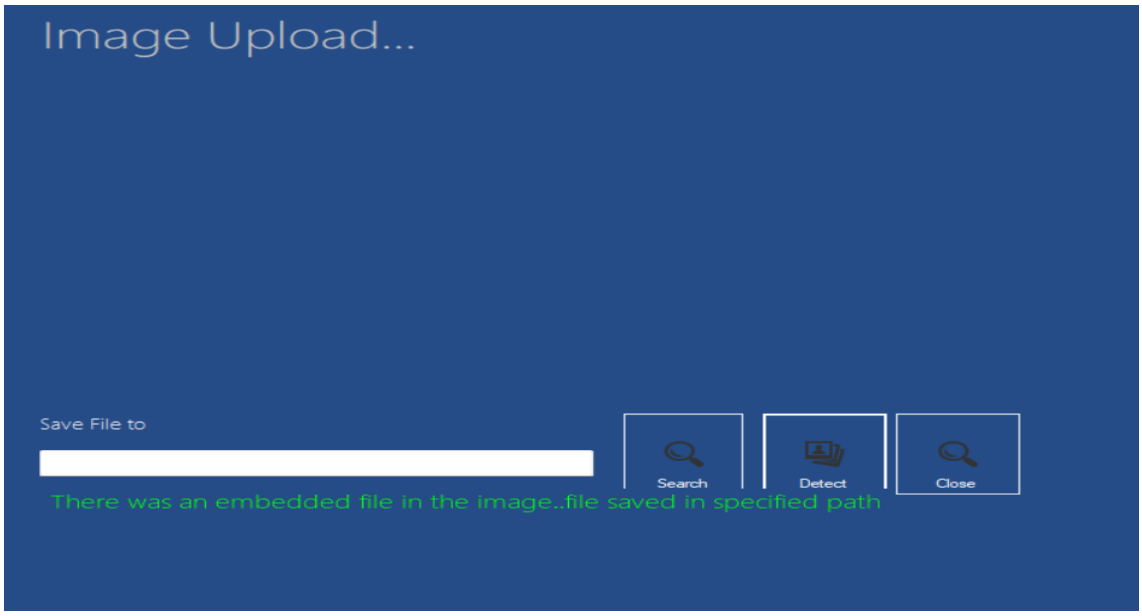

Figure 6.16: Embedding identified and saved

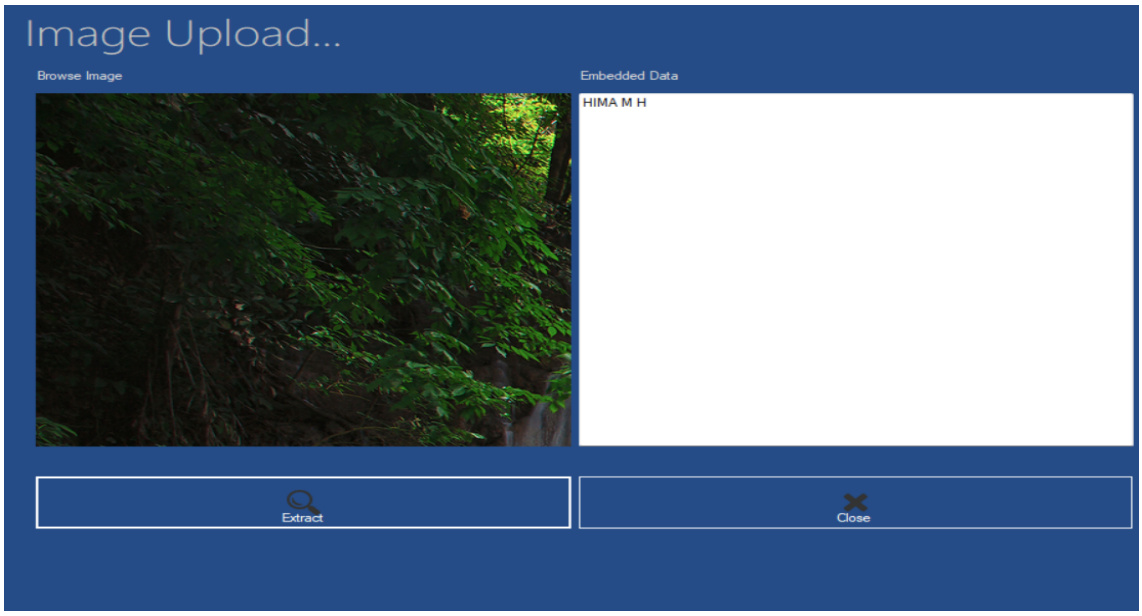

Figure 6.17: Text Embedding

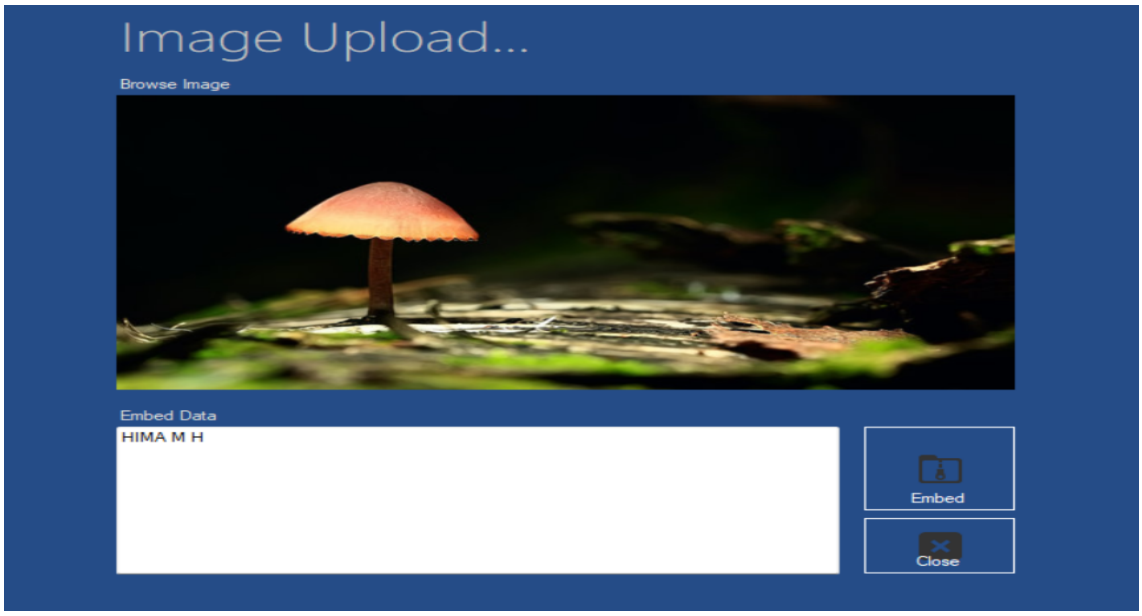

Figure 6.18: Text Extraction

Chapter 7

REFERENCES

[1] Y.-F. Hsu and S.-F. Chang, "Image splicing detection using camera response function consistency and automatic segmentation," in International Conf. on Multimedia and Expo, Beijing, 2007

[2] T. Bianchi and A. Piva, "Detection of non-aligned double JPEG com-pression based on integer periodicity maps," IEEE Trans. Inf. Forensics Security, vol. 7, no. 2, pp. 842–848, Apr. 2012.

[3] M. C. Stamm and K. J. R. Liu, "Forensic detection of image manipulation using statistical intrinsic fingerprints," IEEE Trans. Inf. Forensics Security, vol. 5, no. 3, pp. 492–506, Sep. 2010.

[4] C. Chen, J. Ni, and J. Huang, "Blind detection of median filtering in digital images: A difference domain based approach," IEEE Trans.Image Process., vol. 22, no. 12, pp. 4699–4710, Dec. 2013