Centro Universitário de Brasília - UniCEUB

Faculdade de Ciências Exatas e Tecnologia - FAET

Curso de Engenharia da Computação

Projeto Final

INSERÇÃO DE TEXTO COMPRIMIDO

EM IMAGEM DIGITAL

Aluno: Adriano Delfino de Medeiros – RA: 20015536

Orientador: Prof. MSc. Aderlon M. Queiroz

Brasília, DF - dezembro de 2004

Centro Universitário de Brasília - UniCEUB

Faculdade de Ciências Exatas e Tecnologia - FAET

Curso de Engenharia da Computação

Projeto Final

INSERÇÃO DE TEXTO COMPRIMIDO

EM IMAGEM DIGITAL

Monografia, sob a orientação do Prof. MSc. Aderlon Marcelino Queiroz avaliado por uma Banca Examinadora do Curso de Engenharia da Computação da Faculdade de Ciências Exatas e Tecnologia - FAET do Centro Universitário de Brasília - UniCEUB e constituiu requisito para obtenção do Título de Bacharel em Engenharia da Computação.

Brasília, DF - dezembro de 2004

"Calma e coragem. Olhe para o céu, como é bonito.

Deus é bom, e Pai. Abra os lábios ao sorriso.

A vida é bela ...."

(D. Frei Macelino)

Agradecimentos

A **Deus**,

Um voto: Todo o meu louvor e a minha mais profunda gratidão.

Aos **Meus pais**,

Todos os ideais que porventura eu conseguir na minha existência, qualquer sonho realizado, não me pertencerá exclusivamente. Beijarei sempre as mãos dos meus pais e testemunharei um reconhecimento infinito: ***"Devo-lhes a vida e glória de uma profissão excelsa. "***

A todos os **Meus Familiares** ,

Pelo incentivo, compreensão, apoio e amizade.

Ao meu primo **Márcio Anderson Medeiros,**

Que no altar de nossa amizade seja luz imperecível as lembranças destes ditosos tempos de faculdade em que sempre me ajudou, em especial na realização deste projeto.

Ao **MSc. Francisco Javier Obaldia,**

Que incentivou-me na idéia do projeto e sempre dispensou-me o seu apoio na elucidação de dúvidas.

Ao **MSc. Claudio Penedo de Albuquerque,**

Que muito contribuiu com este projeto, inclusive com o fornecimento de material de pesquisa sobre o tema e na elucidação de dúvidas.

Ao meu orientador **MSc. Aderlon M. Queiroz**,

Que não se limitou à orientação, mas, tornou-se um amigo dedicando-me apoio e ajuda em todos os sentidos.

Aos meus amigos: **Agner Vidal** e **Fernanda Sakamoto Alves**,

Que no altar de nossa amizade seja luz imperecível as lembranças destes ditosos tempos de faculdade em que sempre me ajudaram, incentivaram, para realização do curso.

Aos meus **colegas,**

Que seja inquebrantável o elo de nossas vidas e que as venturas lembranças do nosso fraterno convívio nunca sejam esquecidas.

A todos,

O meu gesto mais sincero de Gratidão.

Resumo

Este trabalho apresenta a projeção e a implementação de um protótipo que utiliza a esteganografia em conjunto com o algoritmo de Huffman para ocultar mensagens comprimidas (textos) em imagens digitais. O algoritmo de Huffman, baseia-se no conceito de árvore binária, associado ao método do *"bit"* menos significativo ("*Last Significant Bit"* ou LSB) da esteganografia, para ocultar o texto comprimido em uma imagem no formato *"Bitmap" "24-Bits"*.

Por fim, os resultados e simulações obtidas com as imagens esteganografadas que mostram os aspectos positivos e negativos do protótipo, utilizando os métodos de histograma de cores e gráficos de performance.

Palavras chaves: esteganografia, *"bit"* menos significativo, marca d´agua, compressão de algoritmo de Huffman, formato *"Bitmap"* e histograma de cores.

Abstract

This work presents the projection and the implementation of a prototype that it uses the steganographic together with the algorithm of Huffman to hide compressed messages (texts) in digital images. The algorithm of Huffman, bases on the concept of binary tree, associate to the method of the Last Significant Bit or LSB of the steganographic, to hide the compressed text in an image in the format Bitmap 24-bits.

Finally, the results and simulations obtained with the images esteganografadas showing like this, the positive and negative aspects of the prototype, using the methods: histogram of colors and performance graphs.

Key words: steganographic, watermarking, LSB, compression, algorithm of Huffman, format Bitmap and histogram of colors.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ASCII	*"American Standard Code for Information Interchange"*
AVI	*"Áudio Vídeo Interleaved"*
BIT	*"Binary Digit"*
BMP	*"Bitmap"* ou Mapa de *"Bits"*
EUA	Estados Unidos da América
GIF	*"Graphics Interchange Format"*
IEEE	*"Institute of Electrcal and Electronics Engineers"*
INTEL	*"Intel Corporation"*
JPEG	*"Joint Photographic Experts Group"*
LSB	*"Last Significant Bit"*
MB	*"Megabytes" correspondem à 1.024 "Kilobytes"*
MP3	*"MPEG Layer 3"*
Pixel	*"Picture Elements"*
RAM	*"Radom-Access Memory"*
RGB	*"Red, Green e Blue"*
RLE	*"Run Length Encoded"*
TIFF	Formato de arquivo imagem do tipo Bitmap
TXT	Formato de arquivo texto suporta somente caracteres
UnB	Universidade de Brasília
UniCEUB	Centro Universitário de Brasília

Capítulo 1: Introdução

1.1 Motivação

A expressiva existência de aplicações para a esteganografia, bem como, das suas diversas utilidades e considerando as recentes medidas de alguns governos com vistas à limitação do uso da criptografia, como é o caso dos EUA[1] [Rocha, 2003] são estímulos à busca de meios alternativos para a garantia de comunicações anônimas e principalmente da preservação dos direitos à liberdade de expressão.

A esteganografia pode, entre outros benefícios, favorecer o aumento da privacidade e da confidencialidade de informações particulares ou de interesse coletivo, tanto de cunho social, e político como econômico.

É importante ressaltar, que a estenografia não foi criada em substituição à criptografia, mas, para complementá-la no que tange a veiculação de métodos alternativos de troca de informações sigilosas de forma segura e eficiente. Ademais, os poderes da segurança digital aumentam consideravelmente quando, ao se transmitir uma mensagem, esta for criptografada e, em seguida, esteganografada, pois, é extremamente difícil se quebrar um código sem ao menos saber de sua existência.

A esteganografia é pouco difundida no Brasil, apesar de ser um assunto pouco explorado no Brasil, pela escassez de material bibliográfico em português, encontra-se em pleno crescimento, envolve segurança da informação e pertence ao foco do curso de Engenharia da Computação do Centro Universitário de Brasília.

1.2 Descrição dos Capítulos

A seguir, é apresentada uma breve descrição dos capítulos dispostos neste trabalho:

- Capítulo 2: Contem alguns conceitos de processamento de imagens que serão extremamente úteis para um bom entendimento do trabalho;
- Capitulo 3: Estão apresentados alguns aspectos históricos relacionados à arte da esteganografia. O estudo dos principais termos utilizados nesta área, as aplicações e também as principais técnicas de esteganografia;

[1] Estados Unidos da América

- Capitulo 4: Traz algumas elucidações de tópicos importantes relacionados à compressão de dados, e ainda detalhamento do algoritmo de Huffman;
- Capítulo 5: Detalha o Protótipo desenvolvido neste trabalho enfatizando sobretudo, a sua funcionalidade bem como, destacando as suas características;
- Capítulo 6: Reúne algumas simulações e descreve os resultados obtidos nas figuras esteganografadas;
- Conclusão: Expõe as considerações finais do trabalho e destaca idéias para novas versões do trabalho.

1.3 Metodologia

Para o desenvolvimento deste trabalho utilizou-se de métodos e de material necessários ao seu bom funcionamento, os quais estão descritos abaixo:

- Foi realizado um levantamento bibliográfico, com a utilização da *"Internet"* [2] e por meio de material bibliográfico encontrado em bibliotecas de Instituições de Ensino Superior (UniCEUB[3] e UnB[4]), bem como, em artigos científicos clássicos (IEEE[5]) e atuais relacionados ao tema: esteganografia, processamento de imagem e compressão de dados;
- Com base nas pesquisas preliminares apresentadas, iniciou-se um estudo de métodos com vistas à implementação computacional fazendo uso da arte da esteganografia, bem como da compressão de dados por meio do algoritmo de Huffman;
- A fase de testes teve início logo após o término da implementação, onde, o protótipo foi testado em um computador;
- Após as fases descritas e com o funcionamento do protótipo de forma satisfatória, iniciou-se a fase de finalização, que compreende a documentação do projeto (monografia).

[2] Rede mundial e pública de computadores
[3] Centro Universitário de Brasília
[4] Universidade de Brasília.
[5] *"Institute of Electrcal and Electronics Engineers"*

1.4 Objetivo do Protótipo

O trabalho ora proposto, objetiva projetar e implementar um protótipo que utilize o método de compressão de dados por meio do algoritmo de Huffman em conjunto com a esteganografia, com a finalidade de esconder textos comprimidos em imagens digitais no formato *"Bitmap"*, considerando:

- Que a mensagem comprimida (texto) pelo algoritmo de Huffman, seja escondida em uma imagem, e que os caracteres utilizados sejam em maior número possível esteganografados;
- Com a busca da menor quantidade de caracteres da mensagem (texto), seja pesquisado qual a maior quantidade de informações (texto) possíveis para serem enviados sem que ocorra quebra de segurança (detecção da mensagem, isto é, percepção ao olho humano).

É importante ressaltar que, quanto menor a mensagem a ser enviada, menor as chances de sua detecção, isto é, altera menos *"bits"* [6] na imagem recipiente. Deste modo, pode-se utilizar a compactação dos dados no texto, além do mais, este tipo de operação, permite o envio de maior quantidade de informação num espaço relativamente pequeno de armazenamento [Fridrich et al., 2001].

[6] *"Binary Digit"* – é a menor unidade de informação usada no computador, representado pelos números binários 0 e 1.

Capítulo 2: Tópicos de Processamento de Imagens

2.1 Conceito de Processamento de Imagens

Está diretamente ligado ao tratamento e análise de imagens. O Processamento de Imagens visa a manipulação e a exibição de imagens prontas, envolvendo diversos processos de tratamento da imagem, bem como, os processos que permitem a "*interface"* [7] entre dispositivos de entrada e saída gráficas e o arquivo de imagem [Casacurta, 1998].

> *"Ao contrário da Computação Gráfica, o Processamento de Imagens não possui como fim a geração de uma imagem a partir de dados, mas sim, a manipulação de imagens previamente geradas e possivelmente a extração de informações a partir destas imagens"* [Casacurta, 1998].

2.1.1 Definição de Imagem

Uma imagem é composta por um conjunto de pontos, denominados *"pixels" ("Picture Elements"*) [Casacurta, 1998].

Estes *"pixels"* estão dispostos na tela do computador formando uma matriz de pontos denominada "*Bitmap"* ou Mapa de "*Bits"*. Este mapa de *"bits"* é um reticulado onde cada elemento da matriz possui uma informação referente a cor associada àquele ponto específico [Casacurta, 1998].

Uma determinada imagem possui também uma resolução associada a ela, que é o número de elementos que esta imagem possui na horizontal e na vertical. Cada elemento da imagem possuirá uma localização, que é definida pela suas coordenadas [Casacurta, 1998].

[7] Termo utilizado para dispositivos que conectam componentes de hardware ao computador

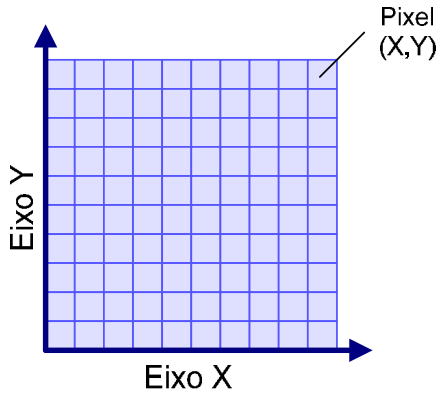


Figura 2.1 – Eixos x e y em um espaço de amostragem de uma imagem digitalizada

2.1.2 Definição de Cor

Um elemento de grande importância na criação de uma imagem é a sua cor.

Cor é um elemento relacionado especificamente a percepção visual do ser humano. Sabe-se que existem animais que não enxergam as cores, portanto a noção de cor está relacionada ao sistema visual do homem [Casacurta, 1998].

Para compreender como uma imagem é percebida visualmente pelo olho humano, é interessante fazer uma análise física da luz. A luz pode ser entendida como uma energia eletromagnética que incide sobre os corpos no espaço. A partir da reflexão dessa luz incidente, os corpos podem ser vistos apresentando a cor correspondente à freqüência da luz refletida.

O ser humano possui em seu sistema visual três tipos de sensores capazes de identificar três faixas diferentes de espectros de energia. Estas faixas correspondem às tonalidades de vermelho (*"Red"*), verde (*"Green"*) e azul (*"Blue"*) que é definida como RGB. O ser humano consegue, facilmente, enxergar a combinação resultante da mistura destas três cores básicas [Casacurta, 1998].

Através de testes realizados com o ser humano, conclui-se que a utilização de 256 variações de intensidade em cada uma das cores básicas é capaz de gerar um número de cores superior a capacidade visual do ser humano, ou seja, fica praticamente impossível de distinguir entre duas cores vizinhas [Casacurta, 1998].

No sistema RGB, o valor (0,0,0) equivale a cor preta com intensidade zero nas três componentes. O valor (255,255,255) equivale a cor branca onde as três componentes estão presentes com a sua intensidade máxima.

As diferentes combinações entre RGB serão capazes de gerar qualquer tipo de cor, considerando que se as três componentes tiverem sempre valores exatamente iguais definir-se

a uma escala de tons de cinza do preto ao branco, que é a chamada *"gray scale"* [Casacurta, 1998].

> "*Para obter todas as cores possíveis de serem distinguidas pelo ser humano se tem um conjunto de aproximadamente 16 milhões de cores (24 bits/pixel)"* [Basic et al., 1999].

2.2 Formato de Arquivo do tipo "*Bitmap"*

2.2.1 Conceito

Dos conceitos pesquisados o da fonte [Basic et al., 1999], é sem dúvida o mais apropriado, pois destaca que os dados matriciais ou "*bitmap"*, como são comumente encontrados na literatura, são formados por um conjunto de valores numéricos que especificam as cores de elementos individuais da tela .

Normalmente se fala que um "*bitmap"* é uma matriz de *"pixels"*, mais precisamente, um *"bitmap"* é uma matriz de valores numéricos que descrevem a cor ou intensidade dos respectivos "*pixels"* num dispositivo de saída quando o dado deve ser mostrado [Basic et al., 1999].

2.2.2 Plataformas de Utilização do Formato *"Bitmap"*

O formato BMP (*"Bitmap"*) foi projetado para sistemas operacionais que funcionem sob a plataforma INTEL[8] (Windows[9] e OS/2[10]).

Deste modo, se for necessária a utilização do formato BMP em outros tipos de arquiteturas, como por exemplo Macintosh[11], deve ser usado outro formato mais adequado (como por exemplo GIF[12]) [Oliveira, 2000].

[Oliveira, 2000] ressaltou que os arquivos BMP do Windows e OS/2 variam em termos de sua estrutura e afirma que a maioria das aplicações conseguem ler ambos os formatos.

---

[8] *"Intel Corporation"*
[9] Sistema Operacional desenvolvido pela *"Microsoft Corporation"*
[10] Sistema Operacional desenvolvido IBM
[11] Série de computadores pessoais lançados em 1984 lançados pela *"Apple Computer Corporation"*
[12] *"Graphics Interchange Format"*

2.2.3 Versões de *"Bitmap"* Quanto à Quantidade de Cor

A quantidade de *"bits"* usados para representar cada *"pixel"* é chamada de *"bit depth"* ou *"pixel depth"* [Basic et al., 1999].

De acordo com [Oliveira, 2000], os arquivos BMP podem ser classificados conforme a quantidade de *"bits"* para representar 1 *"pixel"* (*"bit/Pixel"*); existindo versões de:

- 1 *"bit/pixel"* (2^1=2 cores);
- 4 *"bits/pixel"* (2^4=16 cores);
- 8 *"bits/pixel"* (2^8=256 cores);
- 24 *"bits/pixel"* (*"true color"* com até 2^{24}=mais de 16 milhões de cores).

2.2.4 Uso de Técnicas de Compressão (RLE – *"Run Length Encoded"*)

O RLE – *"Run Length Encoded*," segundo [Basic et al., 1999] , é conveniente para compressão de qualquer tipo de dados, independente do seu conteúdo. No entanto, o conteúdo dos dados interfere na taxa de compressão

Normalmente o RLE não consegue obter a mesma taxa de compressão de algoritmos mais sofisticados, porém, é de fácil implementação e de rápida execução, sendo por isso uma boa alternativa [Basic et al., 1999].

O RLE trabalha reduzindo o tamanho físico de *"strings"* de caracteres repetidos [Basic et al., 1999].

Cada *"string"* de caracteres repetidos é dividido em dois *"bytes"*. O primeiro deles representa a quantidade de vezes que o caracter se repete e é chamado de contador, e o segundo representa o caracter que se repete [Basic et al., 1999].

Um exemplo da compressão RLE pode ser dado usando a *"string"* AAAAAAAAAAAAAAA, que comprimido seria representado por "15A".

Um outro exemplo seria a compressão da "string*"* AAAAAAbbXXXXXt, que resultaria no *"string"* "6A2b5X1t".

Segundo [Oliveira, 2000], é muitíssimo raro, mas arquivos de formato BMP só podem, nas versões de 4 e 8 *"bits/pixel"*, utilizarem a compressão RLE, de forma a reduzir o tamanho do arquivo que armazena o Bitmap.

O formato BMP usado no sistema operacional OS/2 não usa nenhuma técnica de compressão [Oliveira, 2000].

2.2.5 Estrutura Geral do Formato

Todo arquivo BMP, segundo [Oliveira, 2000], está dividido em 3 ou 4 partes, que são:

- **Cabeçalho de arquivo:** Contém a assinatura BM e informações sobre o tamanho e *"layout"* do arquivo BMP (disposição dos dados dentro do arquivo);
- **Cabeçalho de mapa de *"bits"*:** Contém as informações da imagem contida no arquivo. Define as dimensões, tipo de compressão (se houver) e informações sobre as cores da imagem;
- **Paleta ou mapa de cores (opcional):** Somente estará presente em arquivos de imagens que usem até 256 cores. Nas demais, em seu lugar, vem diretamente a parte seguinte: área de dados da imagem;
- **Área de dados da imagem contida no arquivo:** Dados que permitem a exibição da imagem propriamente dita, os dados dos *"pixels"* a serem exibidos. Podem ser com ou sem compressão.

É importante lembrar que existem, dentro da estrutura do BMP, alguns campos ditos "Reservados", destinados a uso futuro, que sempre devem ser setados com o valor zero [Oliveira, 2000].

Maiores informações a respeito da estrutura do arquivo no formato *"bitmap"* encontra-se no anexo A, deste trabalho.

2.2.6 Armazenamento de um *"Bitmap"*

A forma mais simples de armazenamento de *"Bitmap"* é salvar as informações inerentes a cada *"pixels"* exatamente como elas se apresentam, *"bit"* a *"bit"*.

O espaço ocupado por um *"bitmap"* é calculado a partir de suas dimensões, ou seja, horizontal e vertical e de sua profundidade de cor. A fórmula a seguir fornece o tamanho de um arquivo em *"Kbytes"* dado por [Conci, 1995]:

$$Tamanho(Kb) = \frac{Horiz \times Vert \times BPP}{8 \times 1024} \quad (2.1)$$

Onde:

Horiz = corresponde à dimensão horizontal do *"bitmap"*;

Vert = corresponde à dimensão vertical do *"bitmap"*;

BPP = corresponde ao número de *"bits"* por *"pixel"* necessários para armazenar os atributos de cor de um *"pixel"* na profundidade de cor utilizada.

Exemplo: O tamanho do arquivo bitmap com a resolução de 128 X 128 *"pixels"* e profundidade de cor de *"24-bits"* é de: (128 x 128 x 24) / (8 x 1024) = 48 *"Kbytes"*.

2.3 Aspectos que Levaram a Utilização do Formato Bitmap no Projeto

Estes arquivos possuem as seguintes qualidades:

- Facilidade de implementação e manipulação na linguagem de programação;
- Facilidade implementação para o uso da esteganografia, formato de imagem indicada para o método utilizado no projeto de esteganografia (Método do *"Bit"* Menos Significativo) [Johnson, Jajodia, 1998];
- É indicado utilizar *"bitmap" "24-Bits"* para o processo LSB, pois permite alterar uma quantidade maior de *"bits"* sem que ocorram mudanças bruscas na imagem perceptíveis ao olho humano [Johnson, Jajodia, 1998] e;
- As imagens do tipo *"bitmap" "24-Bits"* são indicadas para o processo de esteganografia pois existe uma grande quantidade de cores.

Capítulo 3: Esteganografia

3.1 História

Os registros mais antigos de Esteganografia, datados da era Grega, são do historiador Grego Herodotus. Quando o tirano grego Histiaeus foi aprisionado pelo rei Darius em Susa durante o século V, ele enviou uma mensagem secreta para seu enteado Aristagoras em Miletus. Histiaeus raspou a cabeça de um escravo e tatuou uma mensagem em seu couro cabeludo. Quando o cabelo do escravo cresceu o suficiente ele foi enviado a Miletus [Rocha, 2003].

Outra história da Grécia antiga, também chega via Herodotus. O meio de comunicação utilizado para este fim, na época, era texto escrito em tabletes cobertos de cera. Demeratus, um grego, precisava avisar Esparta que Xerxes pretendia invadir a Grécia. Para evitar a captura, ele removeu a cera dos tabletes e escreveu a mensagem na madeira subjacente. Então ele cobriu os tabletes com cera de novo. Os tabletes pareciam estar em branco e sem uso, por isso passaram pela inspeção [Johnson, Jajodia, 1998].

O período que compreendeu a Segunda Guerra Mundial foi marcado por intensos experimentos esteganográficos. No início da guerra, a tecnologia esteganográfica consistia quase inteiramente em tintas invisíveis. Mais tarde, cifras nulas (mensagens não encriptadas) foram usadas para esconder mensagens secretas. As cifras nulas, que geralmente pareciam ser mensagens inocentes sobre acontecimentos ordinários, não gerariam suspeitas, não sendo então interceptadas. Por exemplo, a seguinte mensagem foi mandada por um espião Alemão durante a Segunda Guerra [Johnson, Jajodia, 1998]:

> *"Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.*"

Ao decodificar essa mensagem, utilizando apenas a segunda letra em cada palavra, revela-se a seguinte mensagem secreta:

"Pershing sails from NY June 1."

Houve certa atmosfera de paranóia sobre mensagens sendo transmitidas tão intensamente, que algumas restrições impostas parecem ridículas nos dias de hoje. Nos EUA, correspondência internacional de jogos de xadrez, instruções de tricô e desenhos de crianças foram proibidos [Rocha, 2003].

Com o advindo da era da computação, a Esteganografia recebeu um grande impulso. Velhos métodos, como ocultar mensagens em imagens, saíram de uma vida latente e voltaram a ser utilizados em conjunto com sistemas de computadores [Rocha, 2003].

3.2 Definição de Esteganografia

Conforme [Bento, Coelho, 2004], esteganografia é uma palavra de origem grega, onde *"Stegano"* significa escondido ou secreto e *"Grafia"* escrita ou desenho.

Esteganografia é a arte de esconder informações (textos, imagens e etc.) utilizando um portador aparentemente inofensivo. Este portador camufla a existência da mensagem secreta.

Não se deve confundir esteganografia com criptografia, pois a primeira, esconde a mensagem, bem como, a sua existência, já a segunda, esconde apenas o conteúdo de uma mensagem não se preocupando em esconder a sua existência. A utilização de ambas, concomitantemente, favorece a obtenção de maior segurança da informação [Bento, Coelho, 2004].

As marcas d´águas digitais são similares à esteganografia em relação à ocultação de dados, os quais aparentam ser parte do arquivo original, e são mais difíceis a sua detecção [Bento, Coelho, 2004].

A *"digital watermaking"* (marca d´água), de acordo com [Góis, 2003], é o conjunto de métodos que permitem identificar o dono ou distribuidor de determinado objeto digital, descobrir cópias não autorizadas do objeto, validar a identificação e verificar a integridade do mesmo e ainda, controlar seu uso e proteger o seu conteúdo. É importante não confundir os conceitos de esteganografia e *"digital watermarking*", defenderam [Peticolas, Anderson, 1998], argumentando que o propósito principal da esteganografia é possibilitar a comunicação encoberta de duas entidades sem que um possível intruso tome conhecimento da existência dessa troca de informação. Nesse caso, um ataque bem sucedido consiste da detecção da existência dessa comunicação. Por outro lado, *"digital watermaking"* possui o requisito adicional de exibir robustez contra possíveis ataques.

Conforme [Bento, Coelho, 2004] a esteganografia pode ser usada para manter a confiabilidade da informação valiosa, para proteger os dados de possíveis sabotagens, roubos, ou visualizações desautorizadas.

3.3 Aplicabilidade

A esteganografia apresenta-se como uma tecnologia indicada no aumento da privacidade "*on-line"* [13], em todos os âmbitos. A união da criptografia com a estenografia resulta, em uma forma robusta e altamente eficiente para manutenção de integridade e proteção de informações sigilosas. No entanto, o sigilo em alto grau preocupa as autoridades políticas e policiais. Uma vez que as comunicações estejam seguras, qualquer indivíduo de um país pode elaborar, em segredo, ou planos mais mirabolantes que desejar. Estes planos podem, inclusive tratar-se de um atentado terrorista e/ou ameaça à segurança nacional de um país [Rocha, 2003] [Amin et al., 2003].

É comum, atualmente, uma visão deturpada de que segurança e privacidade sejam termos antagônicos, isto é, caso as pessoas não possam ser vigiadas, elas representam perigo ou para outras pessoas ou para o país [Rocha, 2003].

Foi neste contexto que os EUA, assinaram em outubro de 2001, o Ato Anti-Terrorismo PATRIOT, um verdadeiro ato de desrespeito à liberdade individual de qualquer cidadão do mundo segundo a EFF ("*Eletronic Frontier Foundation"*). Nesse ato, o presidente americano George W. Bush, através de seu imenso poder à frente da nação mais poderosa da terra, impõe políticas, agora legítimas, que dão às autoridades americanas o direito de espionar qualquer cidadão sem aviso prévio ou posterior em nome da segurança nacional norte-americana [Rocha, 2003].

Atitudes como essas impulsionam os cidadãos a buscarem meios eficientes e sobretudo, eficazes de se protegerem das "insanidades" de seus governos. Mais uma vez, a esteganografia apresenta-se como uma poderosa ferramenta de auxílio à viabilização da segurança, da integridade e da confidencialidade [Rocha, 2003].

Infelizmente, como toda a tecnologia pode ser aplicada para o bem e/ou para o mal, com a esteganografia não é diferente. Nos itens a seguir, encontram-se os principais usos legais e ilegais deste poderoso campo de pesquisas [Rocha, 2003].

3.3.1 Usos legais

O mascaramento de informações tem uma grande variedade de usos legais, seguem algumas das aplicações mais interessantes [Amin et al., 2003] [Rocha, 2003]:

[13] Quando um usuário esta conectado a uma rede

- **Estruturas de dados aprimoradas**: informações podem ser escondidas de modo a funcionarem como estruturas de dados avançadas, como por exemplo, Seria possível armazenar informações médicas a respeito de um paciente em seu próprio raio X;
- **Comunicações privadas:** a esteganografia pode ser utilizada para estabelecer comunicações seguras entre os cidadãos através da rede mundial de computadores (*"Internet"*), como por exemplo, realizar troca de chaves de criptografia através da esteganografia.

3.3.2 Usos ilegais

A esteganografia é um poderoso campo de pesquisas que pode ser aplicado, também, com intenções ilegais. Segundo Rocha, poderiam fazer amplo uso da esteganografia como forma de ampliar seu poder para uso ilegal de:

- Comunicações criminais;
- Pornografia;
- Disseminação de vírus;
- Pedofilia.

Um intruso, com intenções de dano, com um mínimo de habilidade poderia esconder um vírus com alto poder de destruição dentro de uma imagem aparentemente inocente. Um indivíduo que obtivesse a imagem não desconfiaria de nada. Posteriormente, o intruso, valendo-se de falhas de segurança nos sistemas-alvo, poderia ativar o vírus. Com este tipo de vírus, operações sofisticadas poderiam ser feitas, tais como: capturar senhas de acesso, passar-se pelo indivíduo dono do sistema invadido, desviar dinheiro, entre outras coisas.

Além disso, Rocha comenta, que pedófilos estariam compartilhando imagens pela *"Internet"* através do seu mascaramento em imagens aparentemente inocentes. Uma operação de varredura pela rede apontou para uma irmandade pedófila inglesa que também atuava na maior parte dos países europeus.

3.4 Hierarquia de Ocultamento de Mensagem

Há um interesse cada vez maior, nos campos da esteganografia e marcas d'água. Com certeza, isso leva a certa confusão na terminologia. A seguir, encontra-se um estudo dos

principais termos utilizados nestas áreas. É importante salientar que estas definições ainda não estão totalmente aceitas, provavelmente existam pequenas variações na literatura [Rocha, 2003].

Pode-se delimitar a grande-área de pesquisa conhecida como ocultamento da informação (*"information hiding"* [14]) como apresentado hierarquicamente na figura 3.1.

No segundo nível da hierarquia têm-se: esteganografia e marcação de *"copyright"*.

Marcação de *"copyright"* é a tentativa de manter ou provar a propriedade intelectual sobre algum tipo de mídia, seja esta eletrônica ou impressa. Neste sentido, sistemas de marcação robustos (*"watermarking"* robusto) são aqueles que, mesmo após tentativas de remoção, permanecem intactos.

Por outro lado, sistemas de marcação frágeis (*"watermarking"* frágil) são aqueles em que, qualquer modificação na mídia acarreta perda na marcação. Estes sistemas são úteis para impedir a cópia ilegal. Ao se copiar um material original, o resultado é um material não marcado e, por conseguinte, pirata. Sistemas de marcação imperceptível são aqueles em que as logomarcas dos autores, por exemplo, encontram-se no material, mas não são diretamente visíveis. Em contrapartida, marcação visível é aquela em que o autor deseja mostrar sua autoria a todos que observarem a sua criação [Rocha, 2003].

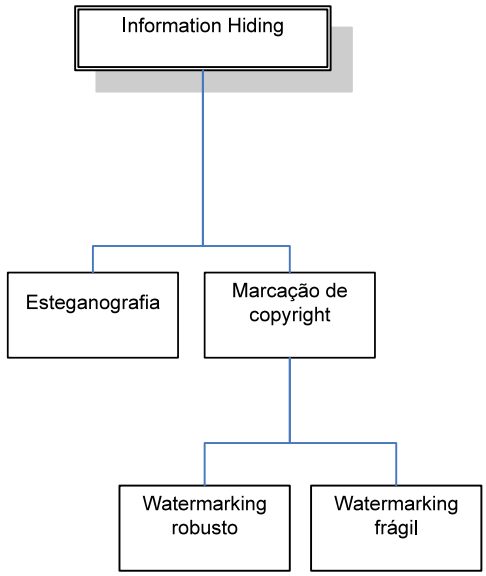


Figura 3.1 – Hierarquia Ocultamento da Informação

[14] Evento realizado em Cambridge, Inglaterra em Abril de 1996 sobre esteganografia

3.5 Termos usados na Esteganografia

A terminologia para descrever um subconjunto de informação oculta foi citada por [Bento, Coelho, 2004] no *"Information Hiding Workshop"*, realizado em Cambridge, Inglaterra em Abril de 1996.

Segundo eles, a descrição de esteganografia – escrita oculta – pode ser feita basicamente da seguinte forma:

O dado embutido (*"embedded data"*) é a informação que alguém deseja enviar em segredo. Este dado geralmente fica escondido em uma mensagem aparentemente inocente, chamada de recipiente ou de objeto de cobertura (container ou *"cover-object"*), produzindo um estego-objeto (*"stego-object"*).

Uma possível fórmula deste processo pode ser representada da seguinte forma:

RECIPIENTE + MENSAGEM EMBUTIDA = ESTEGO-OBJETO

Figura 3.2 – Fórmula do Processo de Esteganografia

[Bento, Coelho, 2004] ainda ressaltam que o termo recipiente é dado a qualquer tipo de informação digital que é transmitida por um sistema digital ou analógico, assim como arquivos de texto, áudio, vídeo, figuras, por exemplo; BMP[15], AVI[16], MP3[17] ou outros tipos.

3.6 Técnicas de codificação de imagem

Essencialmente, a esteganografia da imagem é explorar os poderes limitados do sistema visual humano. Desta forma, um texto pode ser codificado e encaixado em uma cadeia de *"bits"* podendo ser escondidos em uma imagem.

As aproximações mais comuns à informação que esconde nas imagens são [Amin et al., 2003]:

- Inserção do *"bit"* menos Significativo;
- Técnicas de filtragem e mascaramento;
- Algoritmos e Transformação.

[15] *"Bitmap"* ou Mapa de *"Bits"*
[16] *"Áudio Vídeo Interleaved"*
[17] *"MPEG Layer 3"*

Cada uma destes podem ser aplicados às várias imagens, com diversos graus de sucesso. Todos sofrem os graus variando de operações executadas em imagens, tais como, queda de definição ou diminuição na profundidade de cor [Rocha, 2003].

3.6.1 Inserção do *"bit"* menos significativo

O método de inserção do *"bit"* menos significativo é uma aproximação comum, simples de encaixar a informação em um arquivo de imagem. Infelizmente, é extremamente vulnerável a ataques como a manipulação da imagem.

Uma conversão simples de um formato do BMP a um formato da compressão do tipo *"lossy"* [18] como o JPEG[19] pode destruir a informação escondida na imagem [Johnson, Jajodia, 1998].

Ao aplicar técnicas de LSB (método do *"bit"* menos significativo) a cada *"byte"* de uma imagem *"24-bits*", três "bits" podem ser codificados em cada *"pixel"*, como cada *"pixel"* é representado por três *"bytes"*, todas as mudanças ocorridas no *"pixel"* serão imperceptíveis ao olho humano.

Para o exemplo, a letra A pode ser escondida em três "pixels". Suponha os três *"pixels"* originais são representados pelas três palavras *"24-bit"* abaixo no exemplo de [Johnson, Jajodia, 1998].

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

Figura 3.3 – Conjunto de três *"pixels"* de *"24-bits"* [Johnson, Jajodia, 1998]

O valor binário para a letra "A" é (10000011).

Introduzir o valor binário de "A" nos três *"pixels*", partindo do *"byte"* esquerdo superior, resultaria em:

(00100111 11101000 11001000)
(00100110 11001000 11101000)
(11001001 00100111 11101001)

Figura 3.4 – Inserção da letra "A" no conjunto de *"pixels"* [Johnson, Jajodia, 1998]

[18] Tipo de compressão com perda de dados o formato JPEG utiliza esse método de compressão
[19] *"Joint Photographic Experts Group"*

Os bits sublinhados são os únicos *"bits"* que mudaram realmente.

A vantagem principal da inserção de LSB é que os dados podem ser escondidos no menor e no segundo menor *"bit"* e ainda, o olho humano seria incapaz de observar a degradação da imagem, além de ser fácil implementação [Johnson, Jajodia, 1998].

3.6.2 Mascaramento e filtragem

Técnicas de mascaramento e filtragem escondem informações marcando uma imagem de maneira similar as marcas d'água. As técnicas de marca d'água são integradas na imagem, por isso, podem ser aplicadas, sem medo da destruição da imagem, na compressão do tipo *"lossy"* ou compressão com perda de dados. Cobrindo ou mascarando um sinal fraco mais perceptível com outro tornando primeiro não-perceptível, explorando o fato que o sistema visual humano não pode detectar mudanças ligeiras em determinados domínios (temporal) da imagem [Anderson, Petitcolas, 1998].

Tecnicamente, marcas d'água não são fórmulas esteganográficas. Estritamente, a Esteganografia esconde dados na imagem, a marca d'água estende a informação da imagem e transforma-se em um atributo da imagem de cobertura, fornecendo detalhes da licença, da posse ou do *"copyright"*, podendo esconder ou não dados na imagem [Góis, 2003].

Segundo [Johnson e Jajodia, 1998], técnicas de filtragem e mascaramento são restritas às imagens em tons de cinza (*"grayscale"*). Estas técnicas, escondem a informação através da criação de uma imagem semelhante às marcações de *"copyright"* em papel. Isto acontece porque as técnicas de *"watermarking"* garantem que, mesmo se a imagem for modificada por métodos de compressão, a marcação não será removida.

Filtragem e mascaramento são técnicas mais robustas que a inserção LSB no sentido de gerarem estego-imagens imunes a técnicas de compressão e recorte. Ao contrário das modificações LSB, filtragem e mascaramento trabalham com modificações nos "*bits"* mais significativos das imagens.

As imagens de cobertura devem ser em tons de cinza porque estas técnicas não são eficientes em imagens coloridas [Rocha, 2003]. Isto deve-se ao fato de que modificações em "*bits"* mais significativos de imagens em cores geram alta quantidade de "ruído" tornando as informações detectáveis.

3.6.3 Algoritmos e transformações

Por se tratar de imagens de elevada qualidade com boa compressão, é desejável usa-las no formato JPEG através da *"Internet"*.

As imagens JPEG usam o *"discrete cosine transform"* (DCT) para conseguir a compressão. DCT é uma compressão do tipo *"lossy"* ou compressão com perda de dados, porque os valores de co-seno não podem ser calculados precisamente, e o arredondamento de erros pode ser introduzido. As variações entre os dados originais e os dados recuperados dependem dos valores e dos métodos usados no cálculo do DCT. As imagens podem também ser processadas usando a transformação rápida de Fourier e a transformação de Wavelet [Anderson, Petitcolas, 1998].

Estes algoritmos são mais eficazes para processar imagens como o *"cropping"* [20], mas a custo do tamanho da mensagem. Mesmo se a imagem é *"cropped"*, há uma probabilidade que a marca d'água ainda seja legível [Anderson, Petitcolas, 1998].

3.7 Aspectos que Levaram a escolha da Técnica do *"Bit"* Menos Significativo no Projeto

Estes são os aspectos que levaram a escolha da Técnica do *"Bit"* Menos Significativo:

- Método de fácil implementação favorável em relação ao tempo necessário ao e desenvolvimento do projeto;
- Método relativamente de fácil compreensão.

[20] Manipulação de imagem por corte

Capítulo 4: Compressão

4.1 Compressão Estatística

A idéia da compressão estatística é realizar uma representação otimizada de caracteres ou grupos de caracteres.

Os caracteres de maior freqüência de utilização são representados por códigos binários pequenos, e os de menor freqüência são representados por códigos proporcionalmente maiores [Muller, 2004].

O objetivo central da codificação estatística é reduzir o comprimento médio dos códigos usados e logo aumentar a eficiência da compressão [Muller, 2004].

Para entender melhor como pode ser utilizado este tipo de compressão, a seguir, será descrita a Teoria da Harmonia, que elucida o processo de compressão estatística.

4.1.1 Teoria da Harmonia

Entropia é a propriedade de distribuição de energia entre os átomos, tendendo ao equilíbrio.

Sempre que um sistema físico possui mais ou menos quantidade de energia que outro sistema físico em contato direto, há troca de energia entre ambos até que atinjam a entropia, ou seja, o equilíbrio da quantidade de energia existente nos sistemas. Ao atingir o estado de equilíbrio, sabe-se que estes sistemas estão utilizando o mínimo de energia possível para sua manutenção, e assim, poderão manter-se até que outro sistema interaja com eles [Muller, 2004].

Aplicada na informação, a Teoria da Entropia permite a concepção de uma Teoria da Harmonia, ou seja, um ponto de equilíbrio onde a informação pode ser representada por uma quantidade mínima de símbolos. Para chegar a esta representação ideal, basta que se tenha a quantidade de símbolos utilizados e as probabilidades de ocorrência deles. Com base nisso, é possível calcular a quantidade média de *"bits"* por intervalo de símbolo conforme cita [Muller, 2004]:

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \qquad (4.1)$$

Onde, havendo n símbolos, cada qual com uma probabilidade P_i. A representação de quantidades em binário é dada pela base 2 do logaritmo. Foi utilizado um valor n logn por sua proporcionalidade entre quantidade de informação e tempo.

Dessa forma, a fórmula anteriormente apresentada permite verificar se é possível a otimização da quantidade de *"bits"* utilizados para representação de determinado conjunto de símbolos [Muller, 2004].

4.2 Codificação de Huffman

A técnica de compressão de Huffman permite a representação em binário dos caracteres a partir de sua probabilidade de ocorrência.

Esta representação é gerada por um sistema de decodificação em árvore binária, que impede a ambigüidade na análise do código [Muller, 2004].

A ambigüidade, neste caso, refere-se a uma decodificação que permite a confusão com outros caracteres.

Por exemplo, determinado caracter "A" tem o código binário 01 e outro caracter "B" tem o código 0100, isto implica que, ao se verificar a seqüência binária para "B" poderia ser interpretando como "A", ao serem lidos apenas os bits 01.

Portanto, a codificação Huffman utiliza o projeto em árvore binária para projeção dos *"bits"* que representam os caracteres, de forma que permitam uma decodificação única para cada caracter [Muller, 2004].

A codificação de Huffman necessita que cada caracter tenha um valor de probabilidade de ocorrência. A construção da árvore binária começa a partir dos caracteres de menor valor.

Por exemplo, veja a seguinte distribuição:

Tabela 4.1 – Freqüência de Caracteres [Muller, 2004]

C1	0,5
C2	0,2
C4	0,2
C3	0,1

Para os dois caracteres de menor probabilidade, serão atribuídos os valores 0 para C3 e 1 para C4. Eles formarão um ramo cuja probabilidade será 0,3, representado graficamente (na figura 4.1) desta forma:

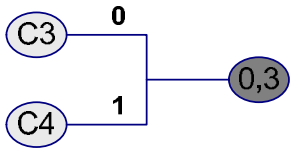


Figura 4.1 – Árvore de Huffman [Muller, 2004]

A probabilidade do ramo é a soma das probabilidades das folhas (0,3 = 0,2 + 0,1). Os valores binários serão membros do código formado.

Para a codificação dos próximos caracteres, basta continuarem a construção da árvore.

O próximo caracter será adicionado à árvore:

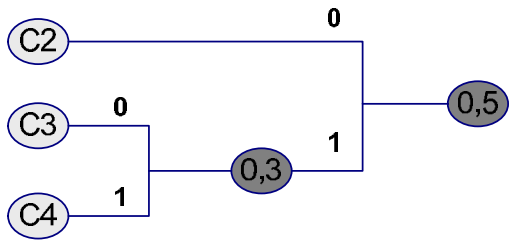


Figura 4.2 – Árvore de Huffman [Muller, 2004]

Mais uma vez, o ramo é a soma das probabilidades anteriores (0,5 = 0,2 + 0,3), e a codificação da divisão recebeu 0 para uma derivação e 1 para outra.

O objetivo desta numeração é a construção do código binário dos caracteres.

Por fim, o último caracter é adicionado à árvore:

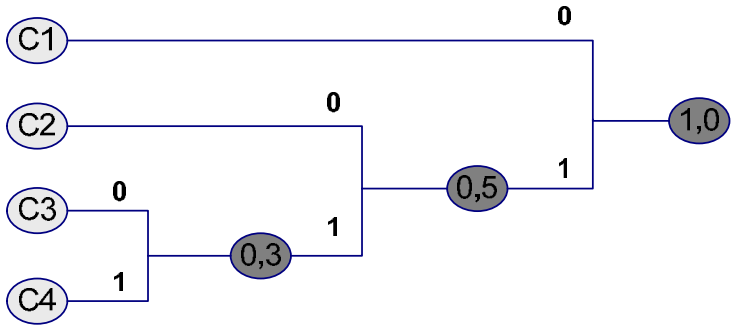


Figura 4.3 – Árvore de Huffman [Muller, 2004]

A probabilidade final da árvore é sempre 1,0, uma vez que, necessariamente, deve-se atingir 100% das ocorrências de caracteres, permitindo uma codificação total [Muller, 2004].

Uma vez terminada a árvore, basta a formalização da codificação, que é feita com a leitura dos valores binários, da raiz para as folhas [Muller, 2004].

Os valores binários lidos, serão o código do percurso da raiz até a folha correspondente ao caracter que se deseja o código [Muller, 2004].

A tabela de códigos fica a seguinte:

Tabela 4.2 – Código de Huffman [Muller, 2004]

C1	0
C2	10
C3	110
C4	111

Desta forma, estão codificados os caracteres através da técnica de Huffman.

Vale ressaltar que o caracter de maior freqüência possui o menor valor. Isso é exatamente o objetivo da compressão estatística, uma vez que permite a substituição do caracter de maior ocorrência por apenas um *"bit"* [Muller, 2004].

Assim acontece com os demais caracteres, em ordem crescente do número de *"bits"*, conforme a prioridade [Muller, 2004].

Por fim, a codificação Huffman mantém a propriedade permissiva a decodificação direta, não deixando que os *"bits"* da codificação de um caracter sejam confundidos com a de outro.

Exemplo: Aplicar o Algoritmo de Huffman no texto abaixo.

"A RAPOSA VERMELHA CORRE DEPRESSA NO DESERTO" (43 caracteres) [Moura, 2004]

O primeiro passo do algoritmo de codificação de Huffman consiste na construção de uma tabela de freqüências relativa aos caracteres que constituem a mensagem original, a qual se quer codificar.

Para tal basta saber o tamanho total da mensagem e em seguida contar separadamente o número de ocorrências de cada caracter existente. Com esta informação, cria-se uma lista ordenada dos caracteres e a sua freqüência associada, como na ilustração seguinte, exemplo de [Moura, 2004]:

Tabela 4.3 – Freqüência de Caracteres [Moura, 2004]

Caracter	**E**	**R, ' '** [21]	**A**	**O, S**	**D, P**	**L, V, M, H, C, N,T**
Nº. Ocorrências	7	6	5	4	2	1
Freqüência (número de ocorrências / 43)	0.162	0.139	0.116	0.093	0.046	0.023

O segundo passo no processo de codificação consiste na criação de uma árvore binária, cujos nós-folhas corresponderão aos caracteres e o caminho percorrido até eles, a sua codificação correspondente [Moura, 2004].

Antes de prosseguir, deve ser feita à introdução do conceito peso, valor diretamente correspondente à probabilidade de ocorrência de um caracter.

No exemplo dado, o peso da letra 'A' é de 0.116. Podendo ser observado que cada nó não-folha da árvore binária de codificação tem um peso distinto correspondente.

Para criar esta árvore binária, é necessário transformar a tabela de probabilidades em N nós-raíz distintos, cada um associado a um caracter e o seu peso correspondente.

Em seguida, pega-se no par de nós com menor peso e criando um novo nó-raiz, cujo peso é a soma dos dois nós tomados e estes como seus dois filhos. Repete-se este processo, juntando cada novo nó-raíz à árvore anteriormente criada, até que torne-se uma só árvore binária completa, cujos nós-folha são os caracteres e o caminho para chegar até eles o seu código binário correspondente, como ilustra o seguinte gráfico [Moura, 2004]:

[21] Símbolo para representar o caracter espaço

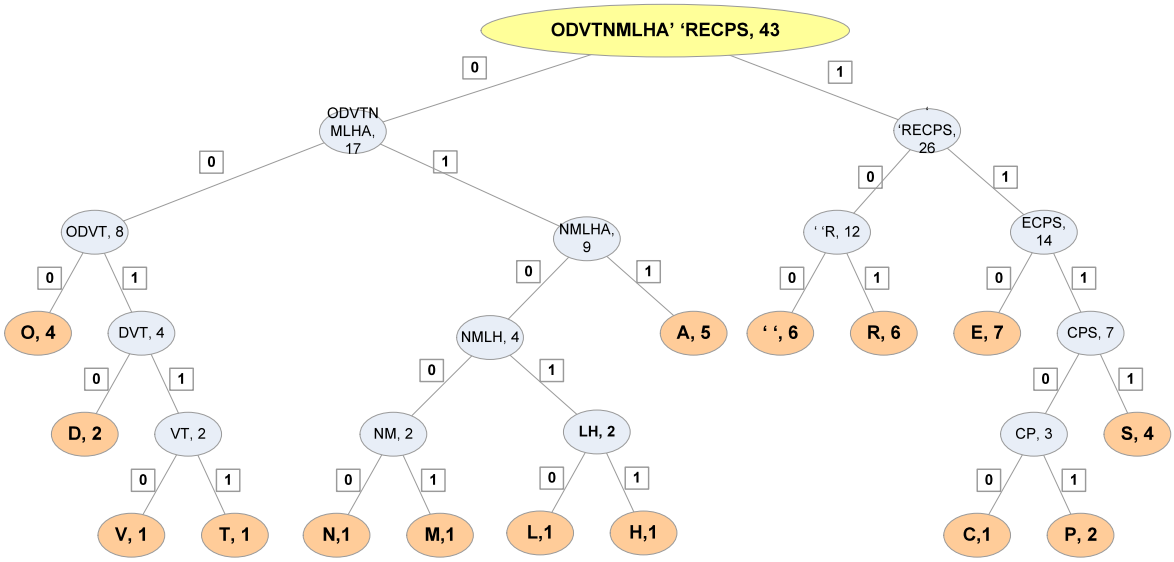


Figura 4.4 – Árvore Binária

e obtêm-se então os códigos de Huffman para os caracteres do texto:

Tabela 4.4 – Código de Huffman

Código Huffman	Caracter
110	E
101	R
100	' '
011	A
000	O
1111	S
0010	D
11101	P
01010	L
00110	V
01001	M
01011	H
11100	C
01001	N
00111	T

Como o objetivo era que os caracteres mais freqüentes fossem representados com um menor número de *"bits"*, e os menos freqüentes com maior número de *"bits*", foi então atingido [Moura 2004].

O terceiro e último passo consiste na simples substituição dos caracteres do texto original pelos seus novos códigos de Huffman [Moura 2004].

Para decodificar o texto comprimido, é necessário ler *"bit"* a *"bit"* os dados de entrada e seguir à direita ou à esquerda na árvore de codificação, consoante o *"bit"* seja 0 ou 1. Quando chega a um nó-folha, se tem o caracter de saída [Moura, 2004].

Aplicando-se a teoria da Entropia neste exemplo, utilizando a equação 4.2:

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \tag{4.2}$$

$$H = -[0{,}162\log_2(0{,}162) + 2\times 0{,}139\log_2(0{,}139) + 0{,}116\log_2(0{,}116) + 2\times 0{,}093\log_2(0{,}093) + 2\times 0{,}046\log_2(0{,}046) + 7\times 0{,}023\log_2(0{,}023)] \tag{4.3}$$

$$H \cong 3{,}5 \Rightarrow H = 4 \tag{4.4}$$

Aplicando o valor médio de bits utilizados com a codificação de Huffman (Q), aonde é multiplicada a freqüência de cada caracter vezes a sua quantidade de bits utilizando na codificação:

$$Q = 0{,}162\times 3 + 0{,}139\times 3 + 0{,}139\times 3 + 0{,}116\times 3 + 0{,}093\times 4 + 0{,}093\times 4 + 0{,}046\times 5 + 0{,}046\times 5 + 0{,}023\times 5 + 0{,}023\times 5 + 0{,}023\times 5 + 0{,}023\times 5 + 0{,}023\times 5 + 0{,}023\times 5 + 0{,}023\times 5 \tag{4.5}$$

$$Q \cong 3{,}588 \Rightarrow Q = 4 \tag{4.6}$$

Pode-se verificar que os valores ficaram praticamente iguais, pois o H=3,5 (valor ideal alcançado pelo método de compressão) e Q=3,588 (valor obtido com o método de compressão de Huffman).

4.4 Aspectos Levaram a Escolha do Algoritmo de Huffman no projeto

Estes são os aspectos que levaram a escolha do algoritmo de Huffman:

- A compressão de Huffman é do tipo *"lossless"*, isto é, compressão sem perdas, baseada em técnicas, que garantem uma cópia exata do fluxo de dados de entrada depois de um ciclo de compressão/expansão. Utilizando o método com perda se tornaria inconveniente para o usuário, pois poderia perder tanto com o extravio de letras e caracteres com na alteração do sentido real do texto;
- Método de fácil e rápida implementação e bastante adequado ao desenvolvimento do projeto;

- Vasta documentação e grande quantidade de material teórico impresso e publicado na Internet e bastante material com ênfase na prática (algoritmos de programação);
- Sugestões de outros projetos concluídos a respeito de esteganografia que indicam a utilização da esteganografia por meio do algoritmo de Huffman, com vistas à uma degradação menor da imagem.

Capítulo 5: Protótipo – Inserção de Texto Comprimido em Imagem Digital

5.1 Características do Protótipo

As principais características do protótipo são:

- Interface agradável ao usuário;
- Manipulação de imagens no formato BMP *"24-bits"* (resolução máxima de até 345x245 *"pixels"*);
- Permissão ao usuário para digitar o texto ou utilizar um arquivo no formato TXT[22], que o mesmo seja esteganografado na imagem;
- Exibição de resultados inerentes ao processo de compressão no texto em gráfico;
- Permissão ao usuário para comprimir o texto, possibilitando assim, uma quantidade maior de caracteres inseridos na imagem;
- Possibilidade de aumento da quantidade de *"bits"* menos significativos que serão utilizados no processo de esteganografia, favorecendo maior quantidade de caracteres inseridos na imagem;
- Utilização de toda a tabela do padrão *"ASCII"* [23] de caracter.

5.2 Execução do Aplicativo

5.2.1 Menu de Configurações

A partir do Menu Configurações, o usuário poderá definir:

- **O tipo de procedimento a ser realizado:** se é esteganografar (codificação) ou decifragem (decodificação);
- **Huffman:** se aplica o algoritmo de Huffman no texto;

[22] Formato de arquivo texto suporta somente caracteres
[23] *"American Standard Code for Information Interchange"*

- **Componentes do RGB:** a quantidade de componentes de cores utilizadas no processo de esteganografia;
- **Bits menos significativos:** a quantidade de *"bits"* menos significativos utilizados no processo de esteganografia;
- **Deslocamento de pixel:** a configuração da quantidade de deslocamento entre um *"pixel"* e outro no processo de esteganografia.

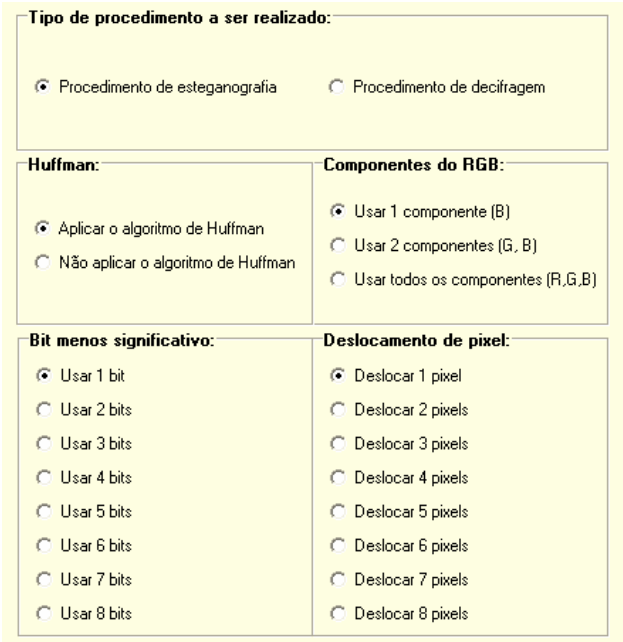


Figura 5.1 – Menu de Configurações do Aplicativo

5.2.2 Processo de Codificação (Cifragem)

O processo de codificação consiste na inserção de um texto em uma imagem, conforme a configuração definida pelo usuário, que segue as seguintes etapas:

- O usuário define a configuração desejada no Menu Configurações, conforme mostra a figura 5.2;

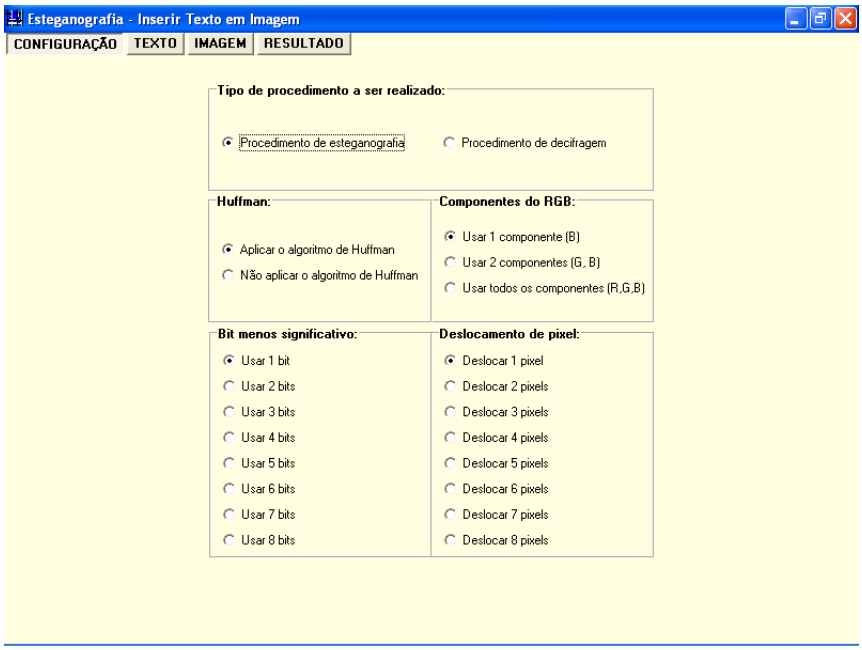


Figura 5.2 – Selecionando o Procedimento de Esteganografia no Menu Configurações

- O usuário pode inserir o texto a ser codificado na imagem, de duas formas:

1. Clicar em **Novo** e digitar o texto;
2. Clicar em **Abrir** para utilizar um arquivo no formato TXT;

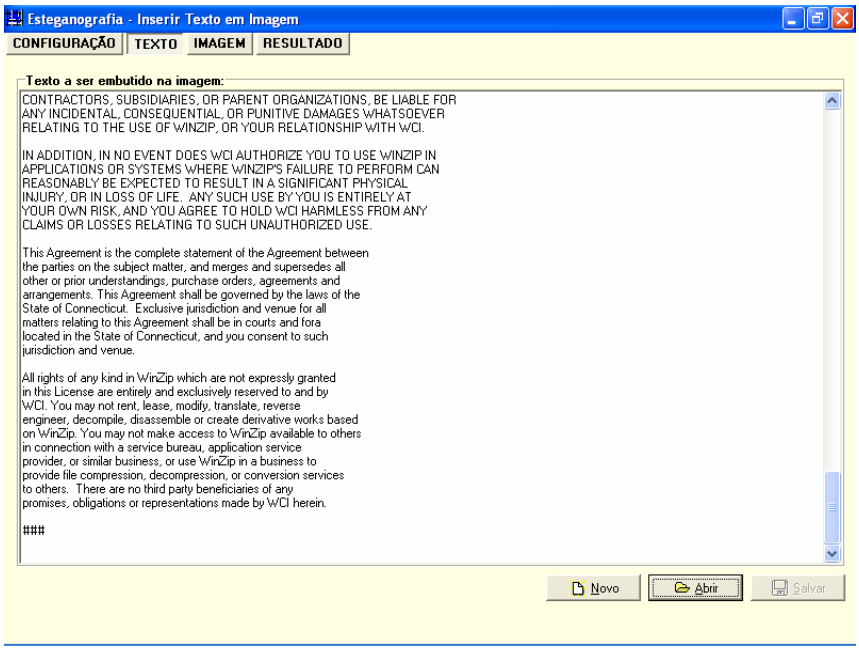


Figura 5.3 – Inserindo um texto no Menu Texto

- Para utilizar a figura desejada, o usuário deve clicar em **Abrir** e escolher a figura desejada (Formato BMP *"24-Bits"*). E o usuário pode definir a cor

desejada no painel **Configuração da cor do pixels modificados**, mostrando assim, a quantidade de *"pixels"* alterados no processo de esteganografia;

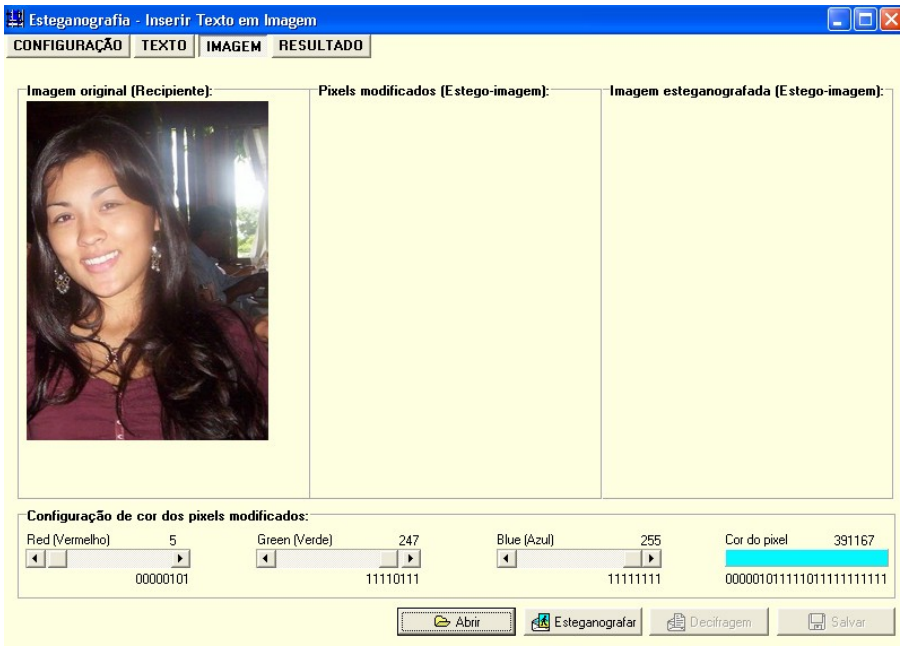


Figura 5.4 – Inserindo uma imagem no Menu Imagem

- Para realização do processo de cifragem (codificação), basta clicar no botão **Esteganografar** e será apresentado o resultado (conforme mostra a figura 5.5).;

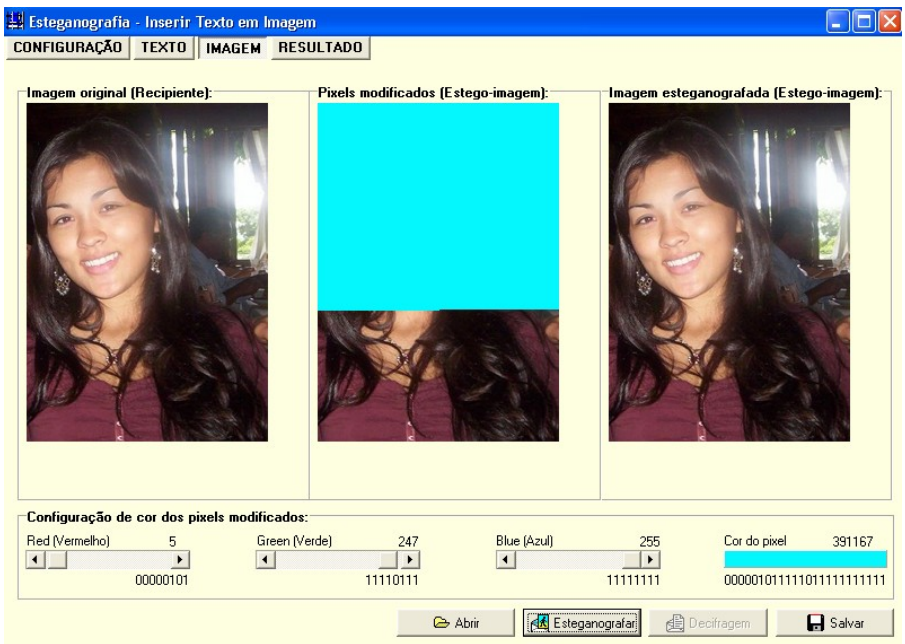


Figura 5.5 – Resultado da codificação

- O usuário pode salvar a figura que mostra os *"pixels"* modificados, lembrando que no processo salvar é necessária a inserção ao final do nome do arquivo **.bmp**;

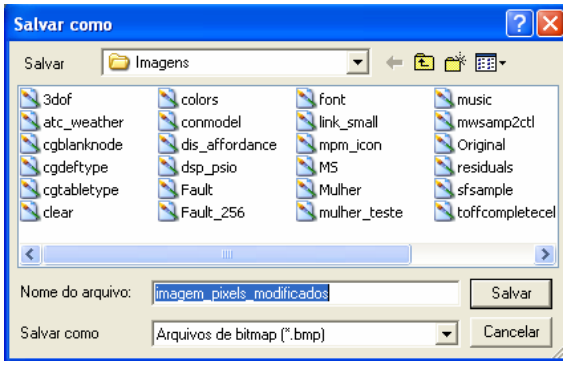


Figura 5.6 – Salvando a imagem com detalhes no formato bmp

- O usuário pode salvar a figura esteganografada ou estego-imagem, lembrando que no salvar precisa inserir ao final do nome do arquivo **.bmp**;

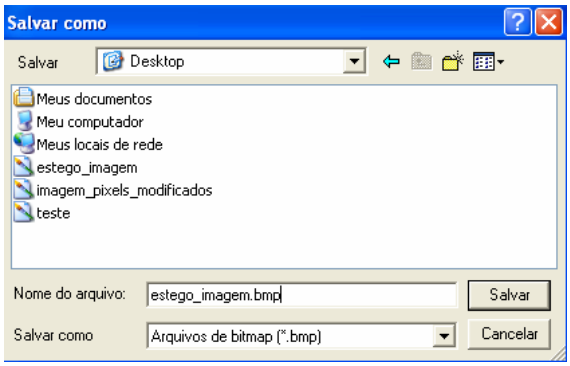


Figura 5.7 – Salvando a estego imagem no formato bmp

- O Menu Resultados mostrará nas Configurações Utilizadas:

1. A configuração definida pelo usuário no processo de esteganografia;
2. Tempo de execução do processo de esteganografia;
3. Quantidade de caracteres do texto, inclusive em *"bits"*;
4. Quantidade de *"bits"* utilizados no processo com algoritmo de Huffman;
5. Mostra um gráfico de desempenho, referente a compressão do texto utilizando o algoritmo de Huffman, que usa essa formula:

$$Economia(\%) = \frac{B_{TOTAL} - B_{HUFFMAN}}{B_{TOTAL}} \times 100 \qquad (5.1)$$

Onde:

B_{TOTAL} = é o número total de *"bits"* para representar o texto em código *"ASCII"*;

$B_{HUFFMAN}$ = é o número total de *"bits"* utilizando o algoritmo de Huffman.

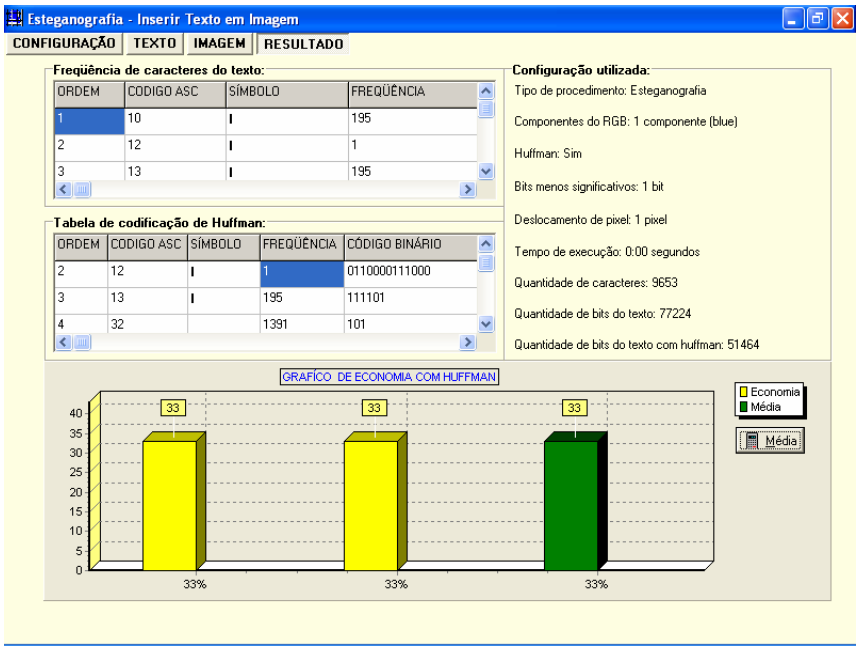


Figura 5.8 – Resultados do processo de esteganografia no Menu Resultados

5.2.3 Processo de Decifragem (Decodificação)

O processo de decodificação, consiste na utilização de uma estego-imagem para recuperar o texto escondido, segues as etapas:

- Para definir o processo de decifragem (decodificação), basta clicar em **Procedimento de decifragem** no Menu Configurações;

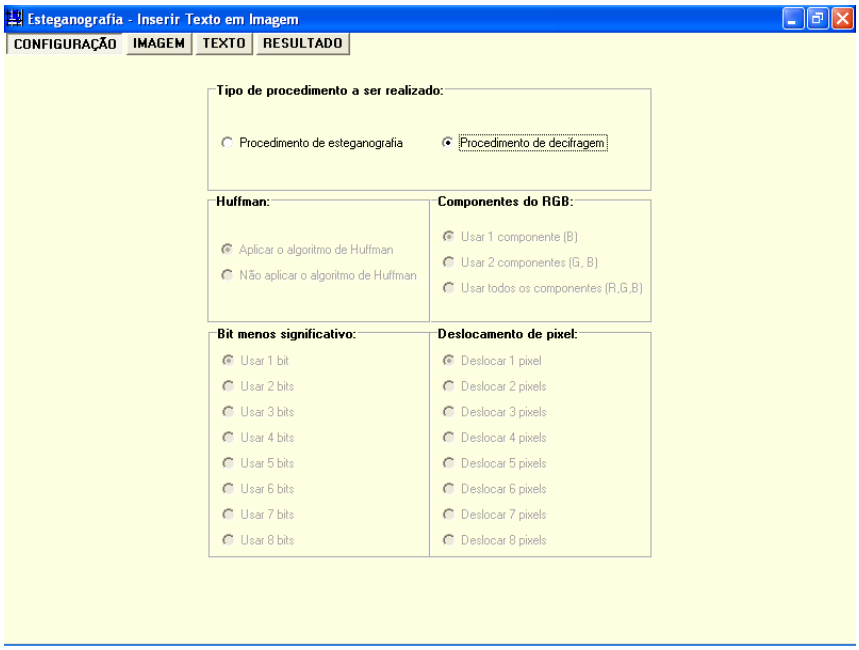


Figura 5.9 – Selecionando o Procedimento de decodificação no Menu Configurações

- Para realização do processo de decodificação, basta clicar em **Decifragem** e será apresentado o resultado (conforme mostra a figura 5.10). Onde o usuário pode definir a cor desejada no componente **Configuração da cor do pixels modificados**, mostrando assim, a quantidade de pixels alterados no processo de esteganografia;

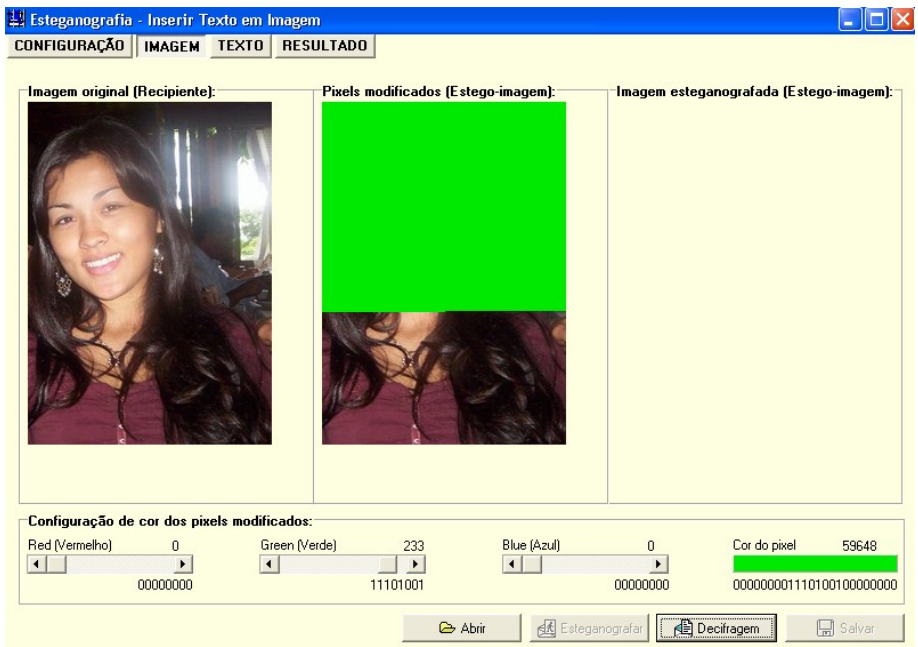


Figura 5.10 – Abrir Imagem para realizar o processo de decifragem no Menu Imagem

- No Menu Texto o usuário visualiza o texto recuperado no processo de esteganografia, conforme mostra figura 5.11.

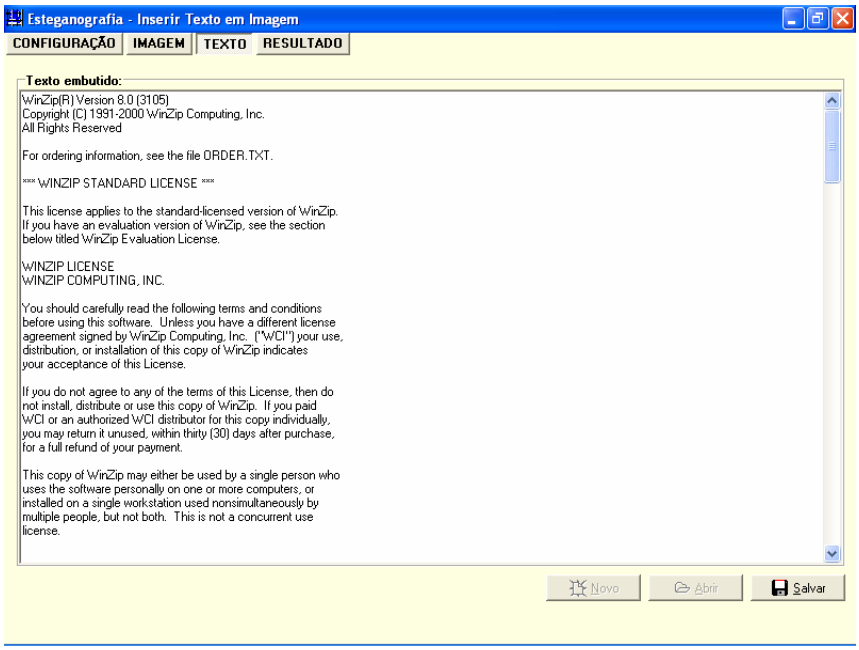


Figura 5.11 – Recuperação do texto codificado no Menu Texto

- O usuário pode salvar o texto recuperado no processo de esteganografia, lembrando que no processo salvar precisa inserir ao final do nome do arquivo a extensão **.txt**;

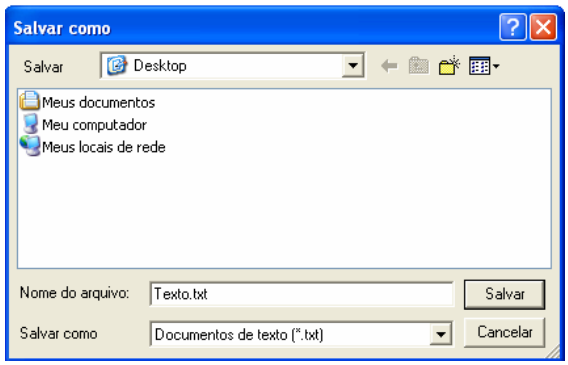


Figura 5.12 – Salvando o texto no formato txt

- Através do Menu Resultados obtém-se:

1. A configuração definida pelo usuário no processo de esteganografia;
2. O tempo de execução do processo de esteganografia;
3. A quantidade de caracteres do texto, inclusive em *"bits"*;

4. A quantidade de *"bits"* utilizados no processo com algoritmo de Huffman;
5. A exibição de um gráfico de desempenho, referente a compressão do texto utilizando o algoritmo de Huffman, que por sua vez utiliza a equação 5.1.

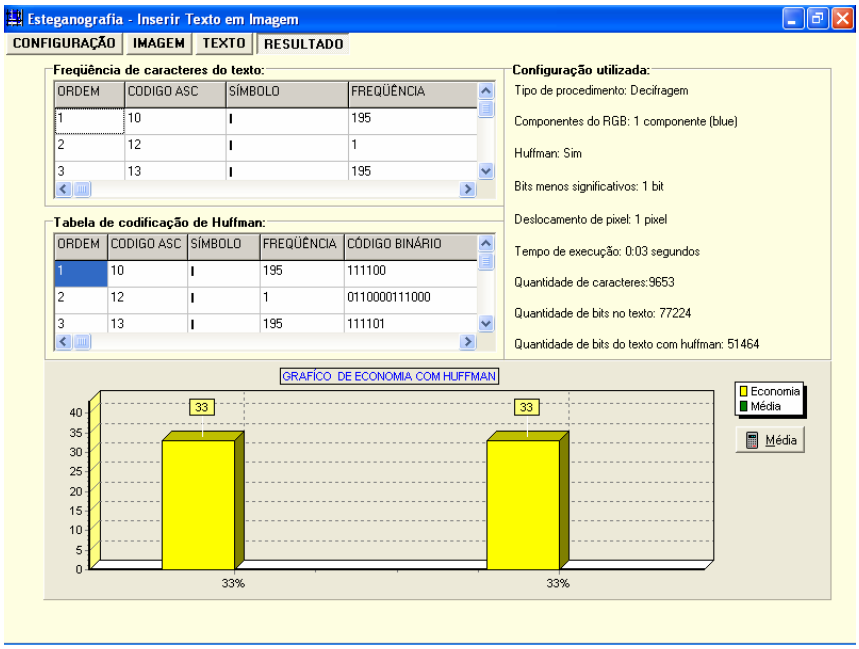


Figura 5.13 – Resultados do processo de decifragem no Menu Resultado

5.3 Descrição do Protótipo

5.3.1 Cabeçalho com Algoritmo de Huffman

A seguir descreve-se detalhadamente a realização da inserção dos *"bits"* (texto) na imagem quando é aplicado o Algoritmo de Huffman:

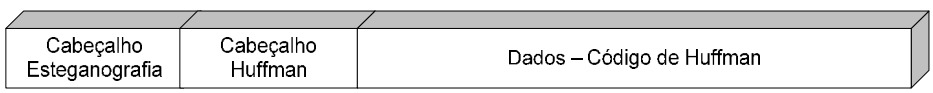


Figura 5.14 – A informação inserida (texto) na imagem com Algoritmo de Huffman

a) **Cabeçalho Esteganografia:** o cabeçalho que é inserido na imagem possui a seguinte estrutura como mostra a figura 5.15:

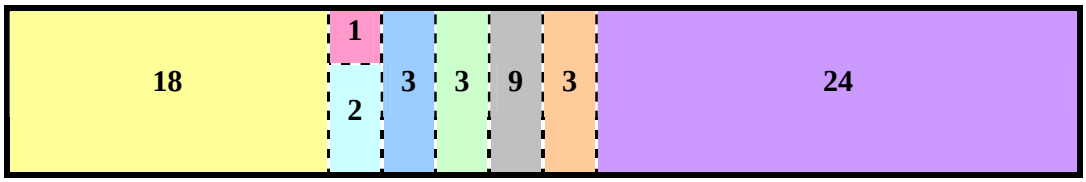


Figura 5.15 – Cabeçalho com Algoritmo de Huffman (Valores em *"Bits"*)

Tabela 5.1 – Descrição do cabeçalho com Algoritmo de Huffman

Campo	Pixels	Bits	Descrição
	6	18	Define se a imagem é esteganografada que possui a seqüência de 111111111111111111;
	1	1	Define se aplica Huffman: 0 não aplica e 1 aplica;
		2	Define a quantidade de componentes RGB.
	1	3	Define a quantidade de deslocamento de *"pixel"* utilizado;
	1	3	Define a quantidade de *"bits"* menos significativo utilizados variando de 1 a 8 *"bits"* menos significativos utilizados;
	3	9	Define a quantidade de símbolos utilizados;
	1	3	Define a quantidade de blocos utilizados na freqüência (campo somente utilizado quando é aplicado o algoritmo de Huffman);
	8	24	Define o tamanho do texto inserido na imagem.
Total	21	63	

b) **Cabeçalho Huffman:** o cabeçalho que é inserido na imagem possui o código de cada símbolo (em código *"ASCII"*) e a sua freqüência;

c) **Dados – Código de Huffman:** esse cabeçalho que é inserido na imagem possui e o texto codificado em código de Huffman de cada símbolo;

5.3.2 Cabeçalho sem Algoritmo de Huffman

A seguir descreve-se detalhadamente, como é realizada a inserção dos *"bits"* (texto) na imagem quando não é aplicado o Algoritmo de Huffman:

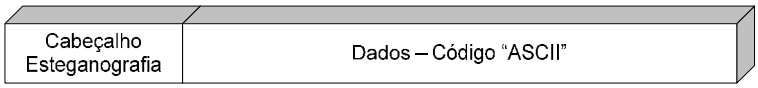


Figura 5.16 – A informação inserida (texto) na imagem

a) **Cabeçalho Esteganografia:** esse cabeçalho que é inserido na imagem possui a seguinte estrutura como mostra a figura 5.17:

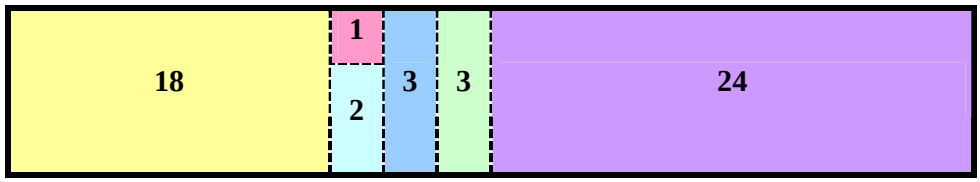


Figura 5.17 – Cabeçalho sem Algoritmo de Huffman (Valores em *"Bits"*)

Tabela 5.2 – Descrição do cabeçalho sem Algoritmo de Huffman

Campo	Pixels	Bits	Descrição
	6	18	Define se a imagem é esteganografada que possui a seqüência de 1111111111111111;
	1	1	Define se aplica Huffman: 0 não aplica e 1 aplica;
		2	Define a quantidade de componentes RGB.
	1	3	Define a quantidade de deslocamento de *"pixel"* utilizado;
	1	3	Define a quantidade de *"bits"* menos significativo utilizados variando de 1 a 8 *"bits"* menos significativos utilizados;
	8	24	Define o tamanho do texto inserido na imagem.
Total	17	51	

b) **Dados – Código *"ASCII"*:** esse cabeçalho que é inserido na imagem possui o texto codificado em código *"ASCII"* de cada símbolo;

5.3.3 Manipulação do *"Pixel"*

Aonde e criado um vetor unidimensional com o tamanho da imagem recipiente, armazenando a cor de cada *"pixel"* da imagem.

Na figura abaixo define como é realizado o processo, simulando a extração de um valor em decimal da cor de um determinado *"pixel"*:

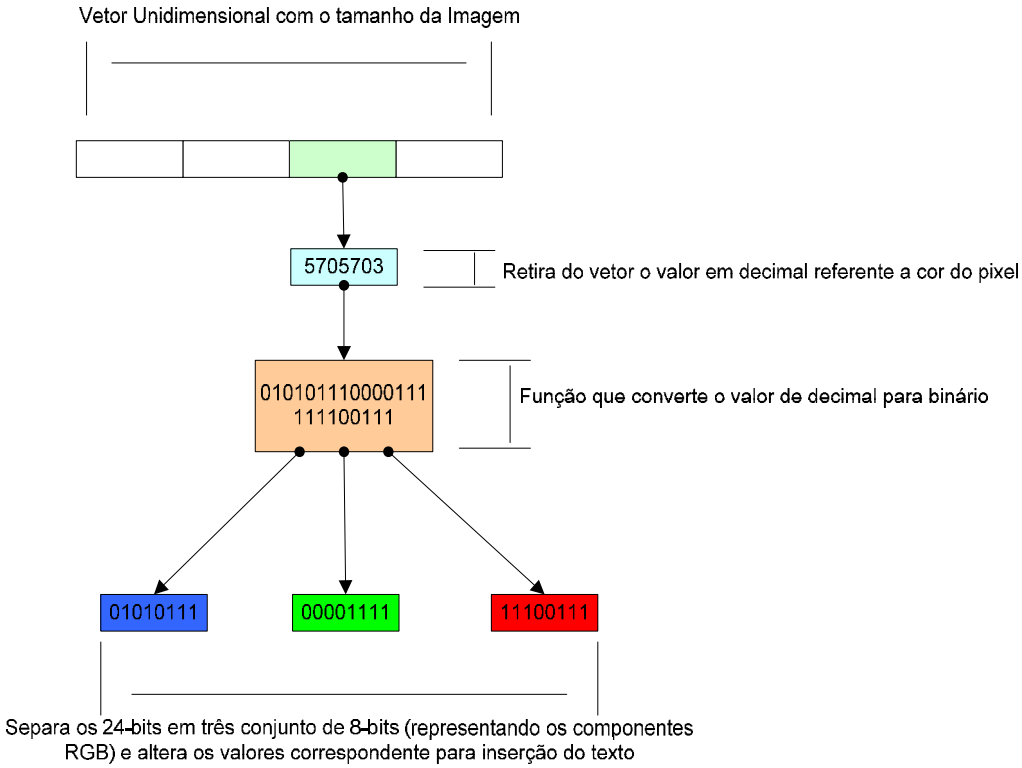


Figura 5.18 – Manipulação do valor do *"pixel"*

5.3.4 Diagrama do Processo de Codificação

A codificação consiste em inserir num texto em uma imagem recipiente, a descrição de cada etapa para codificação:

Etapa - 1:

- Inserção do texto por parte do usuário, o qual, pode ser digitado ou inserido com a utilização de um arquivo no formato TXT;

Etapa - 2:

- Inserção da imagem por parte do usuário, em formato *"Bitmap" "24-Bits"*;

Etapa - 3:

- Inicialização de um vetor tamanho da imagem recipiente para armazená-la;
- Inserção dos valores da cor de cada *"pixel"* (imagem recipiente) em cada posição do vetor;

Etapa - 4:

- Verificação das configurações realizadas pelo usuário, como: quantidade de *"bits"* menos significativos utilizados, componentes RGB, deslocamento e utilização de compressão de dados;

Etapa - 5:

- O Processo consiste em codificar texto para o código *"ASCII"* ou código de Huffman (detalhes no item 5.3.5) depende do que usuário definiu no Menu Configurações;

Etapa - 6:

- Modificação a partir do vetor utilizado na etapa - 3 para o valor um em binário, os seis primeiros *"pixels"* (os três *"bits"* menos significativos de cada *"pixel"* totalizando dezoito *"bits"*) da imagem para identificar que é uma estego-imagem;
- Utilização de um *"pixel"*, sendo um *"bit"* menos significativo destinado para identificar se o texto é comprimido (aplicando o algoritmo de Huffman), e dois *"bits*" menos significativo fornecidos para identificar a quantidade de componentes RGB utilizados para o processo de esteganografia;
- Modificação de três *"bits"* menos significativos de um *"pixel"* para destinar a quantidade de *"bits"* menos significativos utilizados no processo de esteganografia;
- Modificação de três *"bits"* menos significativos de um *"pixel"* para identificar o deslocamento utilizado no processo de esteganografia;
- Modificação de três *"bits"* menos significativos de um *"pixel"* que será destinado para identificar a quantidade de blocos para guardar a freqüência de caracteres. Esta informação só é utilizada quando o texto for comprimido;
- Definição do tamanho do cabeçalho;
- Inserção em um vetor as seguintes informações:

  - Tabela do texto: consiste no código do símbolo em *"ASCII"* com sua freqüência, sendo que essa tabela é ordenada de forma crescente de acordo com o código *"ASCII"* ou/;

- Texto codificado: consiste o texto codificado com o algoritmo de Huffman mais a tabela de Huffman;

- Verificar se o tamanho da imagem (recipiente) é suficiente para armazenar o texto;
- Armazenar a quantidade de *"bits"* esteganografados em três *"bits"* menos significativos de oito *"pixels"*;
- Aplicar o deslocamento desejado;

Etapa - 7:

- Extração das informações do vetor e inseri-las no vetor imagem recipiente;

Etapa - 8:

- Exibição das imagens resultantes, estego-imagens;

Etapa - 9:

- Exibição do resultado de todo o processo de codificação no Menu Resultado.

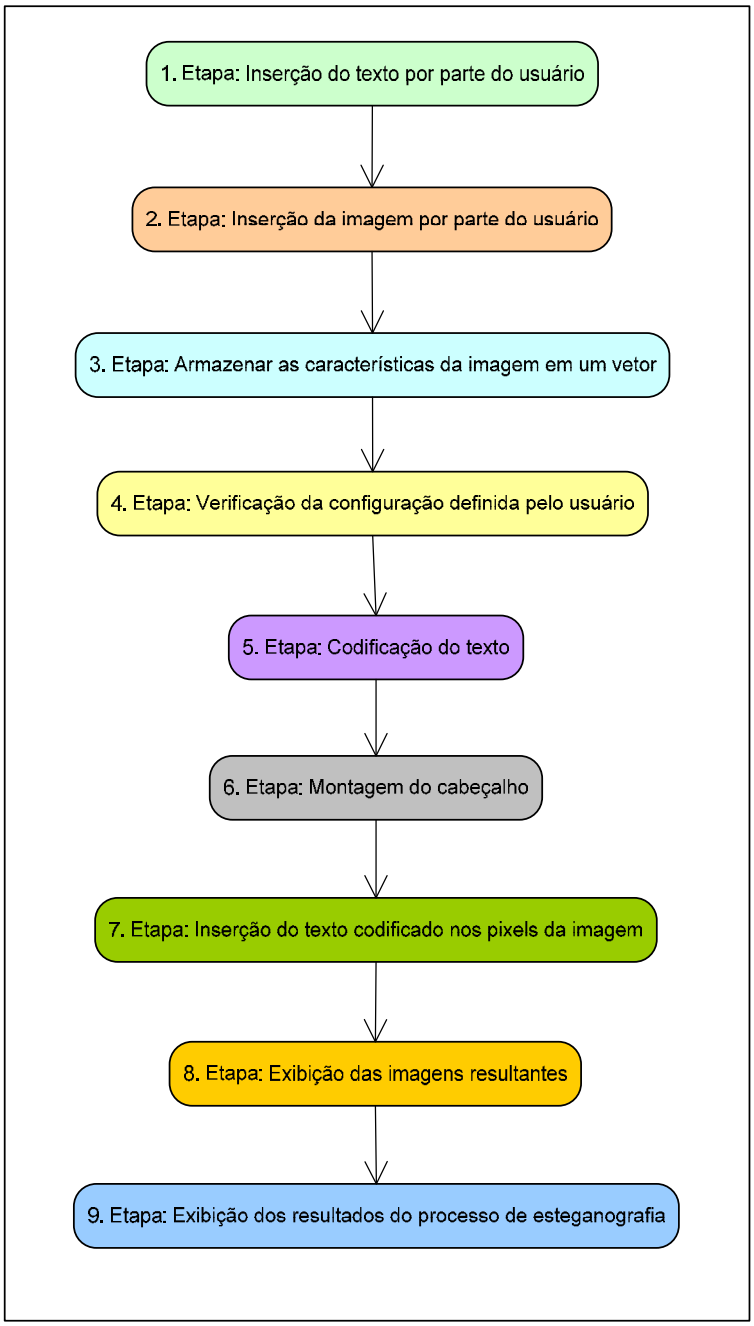


Figura 5.19 – Diagrama do processo de codificação

5.3.5 Diagrama do Processo do Algoritmo de Huffman na Codificação

A seguir estão descritos a seqüência de passos, realizados pelo protótipo, quando da utilização do algoritmo de Huffman:

Etapa - 1:

- Determinação da freqüência de ocorrências de cada símbolo;

Etapa - 2:

- Primeira ordenação crescente de acordo com freqüência de caracteres;

Etapa - 3:

- Inserção dos valores de símbolo (*"string"* [24]), da freqüência do símbolo e seu código *"ASCII"* na tabela de Huffman;

Etapa - 4:

- Montagem da Árvore Binária, definindo a localização de cada nó;
- Atribuição dos rótulos: 0 (zero) para cada aresta que conduz ao filho esquerdo de um nó interno e 1 (um) para cada aresta que conduz ao seu filho direito;

Etapa - 5:

- Atribuição do código correspondente a cada símbolo identificado pela trajetória obtida da raiz até a folha da árvore de Huffman que contém o símbolo. Gerar uma tabela de códigos com o código de cada símbolo.
- Depois da montagem da Árvore Binária, será realizada a busca dos códigos de Huffman de cada símbolo e inseri-los na tabela de Huffman;
- Realizar a ordenação crescente da tabela de Huffman de acordo com o código *"ASCII"*;

Etapa - 6:

Para a codificação pelo método de Huffman são necessários os seguintes passos:

1. Identificação, na tabela de códigos de Huffman, do código correspondente ao símbolo a ser codificado;
2. Substituição do símbolo a codificar pelo código correspondente;
3. Repetir os passos 1 e 2 para todos os símbolos a serem codificados, gerando, assim, uma seqüência de códigos.

[24] Forma de representar um ou conjunto de caracter na linguagem de programação

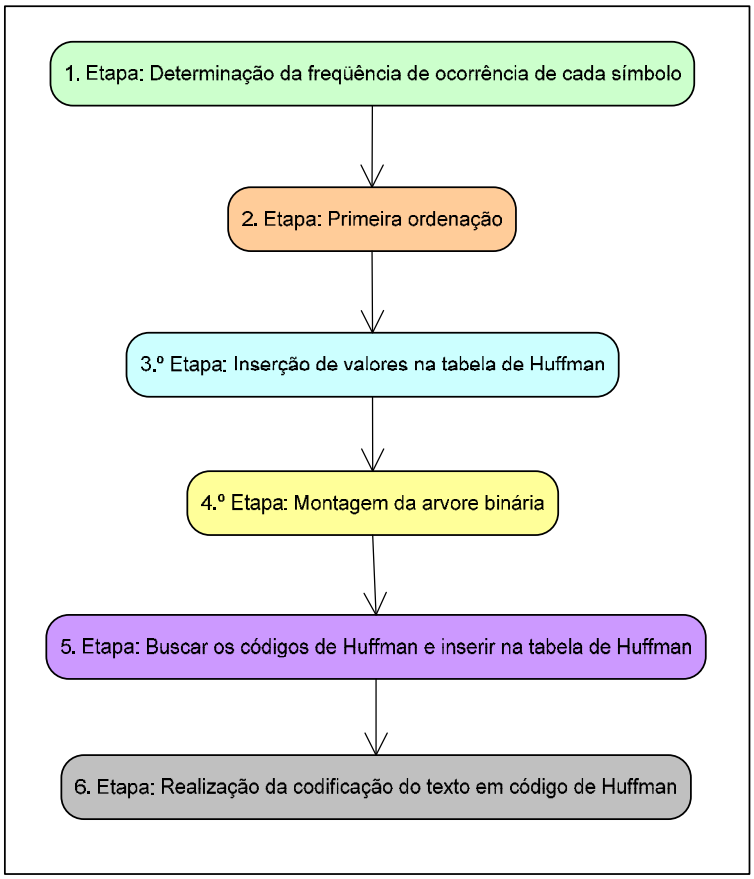


Figura 5.20 – Diagrama do processo de codificação do Algoritmo de Huffman

5.3.6 Diagrama do Processo de Decodificação

A decodificação consiste em retirar um texto dentro de uma estego-imagem, a descrição de cada etapa para decodificação:

Etapa - 1:

- Inserção de uma estego-imagem pelo usuário no formato BMP *"24-bits"* bmp;

Etapa - 2:

- Verificar se a imagem é esteganografada, isto é, se possui o valor um nos seis primeiros *"pixels"* (os três *"bits"* menos significativos de cada *"pixel"* totalizando dezoito *"bits"*);
- Verificar se utiliza o algoritmo de Huffman no texto;
- Verificar a quantidade de componentes RGB utilizados no processo de esteganografia;

- Verificar a quantidade de *"bits"* menos significativos utilizados no processo de esteganografia;
- Verificar o deslocamento utilizado no processo de esteganografia;
- Se utilizar o algoritmo de Huffman, verificar a quantidade de blocos para guardar a freqüência do caracter;

Etapa - 3:

- Extração dos *"bits"* que foram utilizados para inserção do texto codificado e inserção no Menu Texto;

Etapa - 4:

- Exibir o resultado do processo de codificação no Menu Resultado.

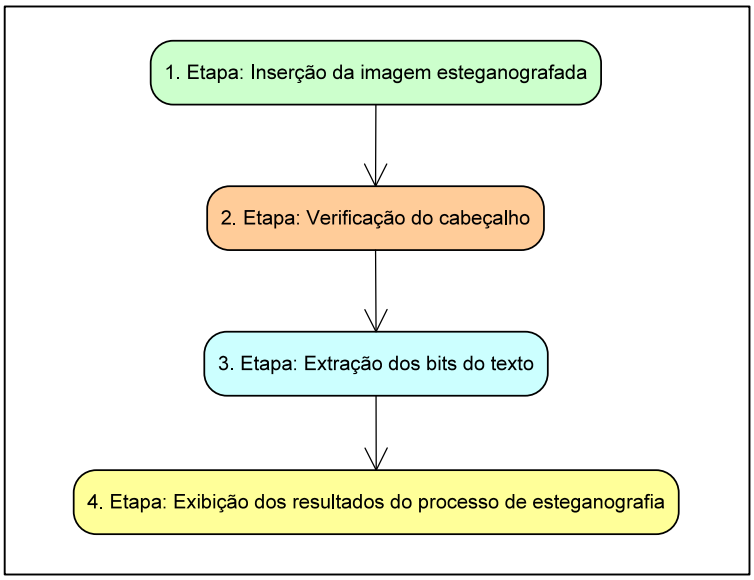


Figura 5.21– Diagrama do processo de decodificação

5.3.7 Diagrama do Processo de Decodificação do Algoritmo de Huffman

Abaixo estão listados os passos realizados pelo protótipo quando da utilização do algoritmo de Huffman:

Etapa - 1:

- Verificação, no cabeçalho, da quantidade de símbolos, três *"pixels"* (oito *"bits"*);

- Verificação, no cabeçalho, da quantidade de *"bits"* utilizados no processo de esteganografia, oito *"pixels"* (vinte e quatro *"bits"*);

Etapa - 2:

- Junção dos dados conforme o deslocamento utilizado;
- Armazenamento da tabela e do texto (em binário), em uma variável do tipo vetor;

Etapa - 3:

- Identificação da quantidade de símbolos utilizados no texto e inicialização da Árvore Binária;
- Armazenamento dos valores (freqüência do símbolo, o código *"ASCII"* do símbolo e a *"string"* do símbolo) em uma variável do tipo registro ou estrutura;
- Inserção dos dados na tabela de Huffman;
- Realização da primeira ordenação crescente de acordo com a freqüência;
- Montagem da Árvore Binária;
- Busca dos códigos de Huffman na Árvore;
- Acréscimo dos códigos na tabela de Huffman;

Etapa - 4:

Montagem do texto de acordo com o código de Huffman. Para a decodificação por meio do método de Huffman são necessários os seguintes passos:

1. Escolha do primeiro *"bit"* do código como *"bit"* corrente;
2. Definição de um nível da Árvore de Huffman, a partir da raiz, de acordo com o valor do *"bit"* corrente (0 => filho esquerdo ou 1 => filho direito) e atribuição ao *"bit"* corrente do próximo *"bit"* da *"string"* de *"bits"* do código a decodificar;
3. No caso de ser atingida uma folha da árvore de Huffman, deve-se fazer a substituição dos *"bits"* correspondentes à trajetória até a folha pelo caractere contido nessa folha;
4. Repetição dos passos 2 e 3 até que termine a "*string"* de *"bits"* do código a decodificar;

Etapa - 5:

- Exibição dos resultados nas tabelas e no gráfico do Menu Resultado, o resultado da compressão;
- Exibição dos dados definidos pelo usuário no processo de esteganografia no Menu Resultados;

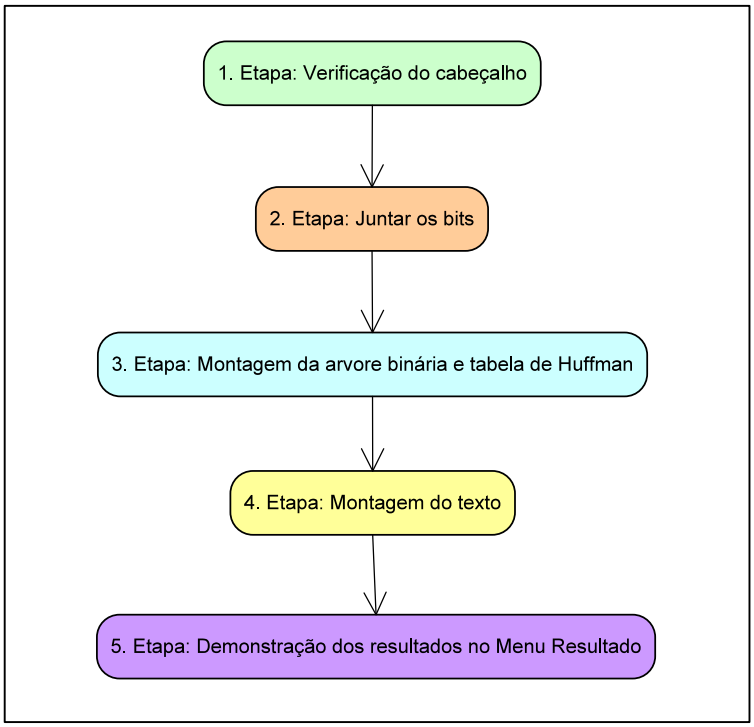


Figura 5.22 – Diagrama do processo de decodificação do Algoritmo de Huffman

Capítulo 6: Resultados e Simulações

6.1 Configuração do Ambiente de Homologação

As simulações foram realizadas em um computador com a seguinte configuração:

1. Hardware:

    - Processador AMD – Duron com freqüência de 1.800 MHz;
    - 256 MB[25] de memória RAM[26];
    - 40 GB[27] de disco rígido.

2. Softwares utilizados:

    - Sistema Operacional Microsoft Windows XP[28];
    - Microsoft Bloco de Notas;
    - Microsoft Paint;
    - Aplicativo de manipulação de imagens Adobe Photoshop 7.0.

6.2 Métodos Utilizados

Para ajudar na visualização dos resultados obtidos com as imagens esteganografadas, utilizou-se do método de histograma de cores, o gráfico de performance do Algoritmo de Huffman e os resultados das imagens esteganografadas.

6.2.1 Aplicativos Utilizados

Na realização dos testes com as figuras esteganografadas, utilizou-se do aplicativo de gerenciamento de imagem Adobe Photoshop para a visualização dos resultados em histograma de cores, o gráfico de performance do Algoritmo de Huffman do próprio

[25] Abreviação de MegaByte
[26] *"Radom-Access Memory"*
[27] Abreviação de GigaByte
[28] Versão do Windows lançado em 2001

Protótipo, as estego-imagens e imagens com detalhes adquiridas no processo de esteganografia do Protótipo.

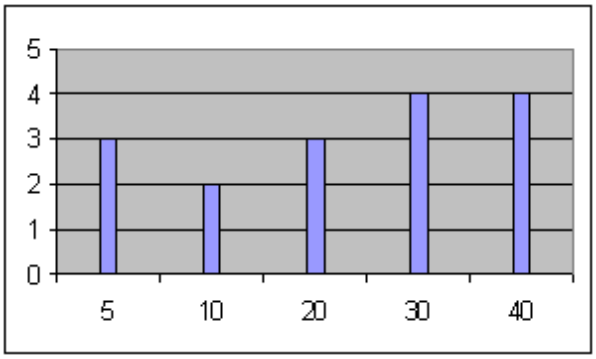
6.2.2 Histograma de Cores

O histograma de cores é um gráfico que determina a freqüência de cada cor em forma de barra, é uma informação importante para se analisar as características da imagem.

No caso de imagens coloridas, temos três histogramas, para R, G e B, porém a cor final percebida pelo sistema visual humano é uma combinação desses canais, o que dificulta uma análise imediata [Caversan, 2004].

20	30	30	20
20	40	40	5
5	40	40	5
10	30	30	10

(a)



(b)

Figura 6.1 : (a) Representação de uma imagem em tons de cinza com o valor da intensidade de cada pixel e (b) seu histograma [Caversan, 2004].

6.3 Resultados com Histograma de Cores

6.3.1 Amostra

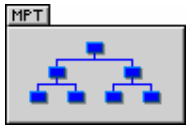


Figura 6.2 – Imagem utilizada para realizar os testes com histograma de cores

A imagem no formato BMP *"24-bits"* como mostra a figura 6.2 que possui 56.848 *"pixels"*, com tamanho de 189 KB[29].

Foi utilizado um texto no formato TXT com 16.000 caracteres, com tamanho 16 KB

Abaixo, é mostrado o histograma de cores da imagem utilizado na amostra:

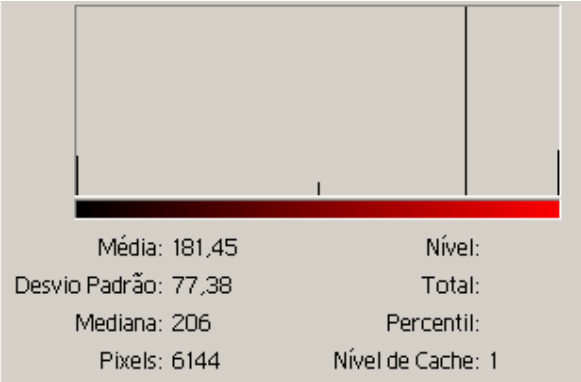


Figura 6.3 – Histograma de cores referente ao componente vermelho da figura 6.2

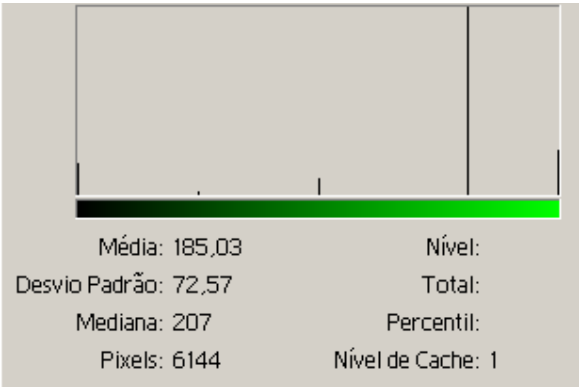


Figura 6.4 – Histograma de cores referente ao componente verde da figura 6.2

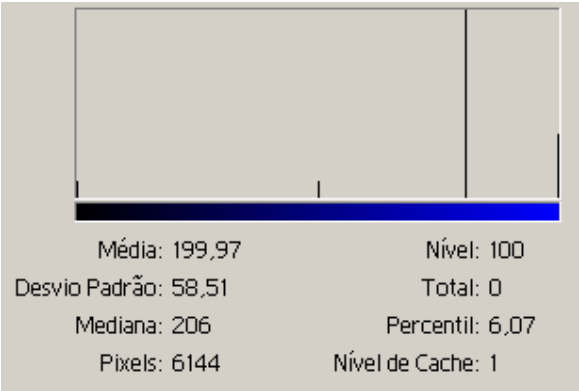


Figura 6.5 – Histograma de cores referente ao componente azul da figura 6.2

[29] Abreviação de KiloBytes

6.3.2 Tabelas

Os testes consistem em variar a quantidade de *"bits"* menos significativos, exemplo, se utilizarem dois *"bits"* menos significativos, são modificados dois *"bits"* menos significativo de cada componente RGB (na realidade são 6 *"bits"* alterados, dois de cada cor).

Nas tabelas abaixo (6.1 e 6.2) são mostrados os valores obtidos nos testes efetuados através das imagens esteganografadas com a inserção do texto sem compressão, utilizando as amostras citadas no item 6.3.1:

Tabela 6.1 – Comparativo sem compressão no texto na figura 6.2

	1 Bit			**2 Bit**			**3 Bit**			**4 Bit**		
	R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
Média	181,42	184,66	199,90	181,36	184,27	199,74	180,79	183,71	199,22	180,02	182,99	198,50
Desvio Padrão	77,30	72,27	58,49	76,90	71,88	58,03	76,40	71,34	57,51	75,50	70,47	56,51
Mediana	206	206	207	206	206	206	206	206	206	206	206	206

Tabela 6.2 – Comparativo sem compressão no texto na figura 6.2

	5 Bit			**6 Bit**			**7 Bit**			**8 Bit**		
	R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
Média	179,18	182,07	197,58	177,73	180,61	195,98	173,27	176,39	191,69	166,66	168,56	184,62
Desvio Padrão	74,12	69,06	55,24	72,74	67,70	53,67	71,36	66,39	53,23	76,53	72,30	62,14
Mediana	206	206	206	206	206	206	206	206	206	206	206	206

A partir, dos dados fornecidos pelas tabelas 6.1 e 6.2, pode se conclui que a degradação na imagem é maior, quanto mais *"bits"* menos significativos utilizados no processo de esteganografia.

Nas tabelas abaixo (6.3 e 6.4) são mostrados os valores obtidos nos testes efetuados através das imagens esteganografadas com a inserção do texto comprimido, utilizando as amostras citadas no item 6.3.1:

Tabela 6.3 – Comparativo com compressão no texto na figura 6.2

	1 Bit			2 Bit			3 Bit			4 Bit		
	R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
Média	181,43	184,77	199,96	181,42	184,50	199,85	181,05	184,13	199,48	180,60	183,62	198,93
Desvio Padrão	77,30	72,29	58,34	76,95	71,99	58,06	76,52	71,49	57,61	75,76	70,70	56,80
Mediana	206	207	206	206	207	206	206	207	206	206	207	206

Tabela 6.4 – Comparativo com compressão no texto na figura 6.2

	5 Bit			6 Bit			7 Bit			8 Bit		
	R	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**	**R**	**G**	**B**
Média	179,85	182,88	198,25	178,75	181,73	197,01	175,96	178,70	194,41	171,38	173,48	189,33
Desvio Padrão	74,92	69,82	55,81	73,90	68,76	54,77	72,95	67,69	54,48	75,35	72,40	60,04
Mediana	206	207	206	206	207	206	206	206	206	206	206	206

A partir, dos dados fornecidos pelas tabelas 6.3 e 6.4, pode se conclui que a degradação na imagem é maior, quanto mais *"bits"* menos significativos utilizados no processo de esteganografia, mesmo com o texto comprimido.

Depois de realizar esta simulação, verificou-se que a estego-imagem sofre menos degradação com o texto comprimido do que com o texto sem compressão, conforme mostra os parâmetros de média, mediana e desvio padrão, pois o texto comprimido utiliza menos *"bits"* para a inserção do texto na imagem.

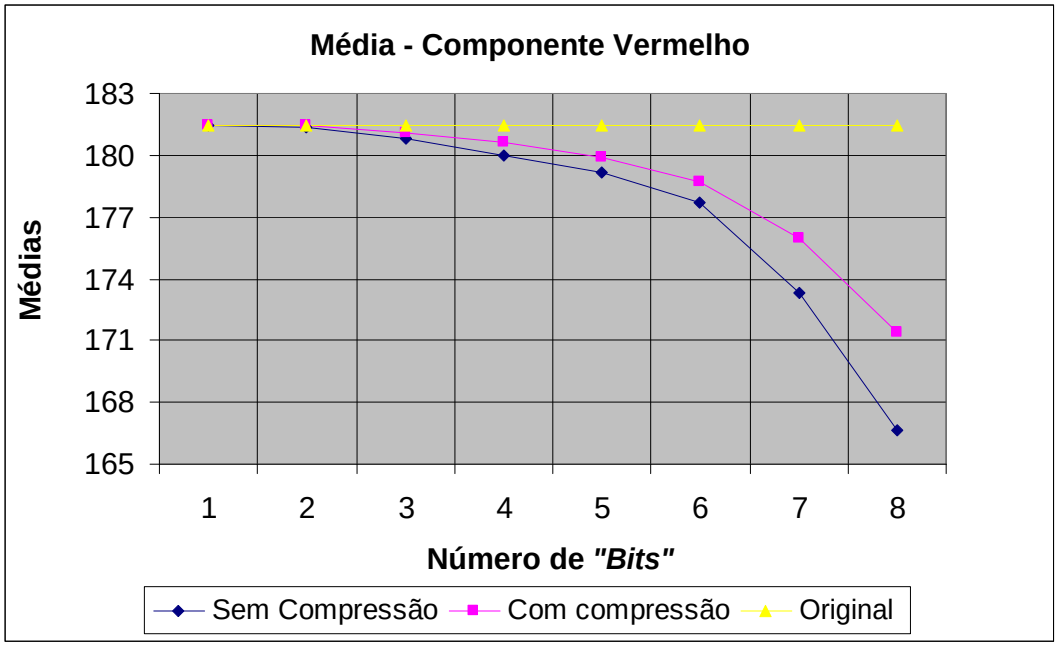


Figura 6.6 – Gráfico comparativo das médias dos histogramas do componente vermelho

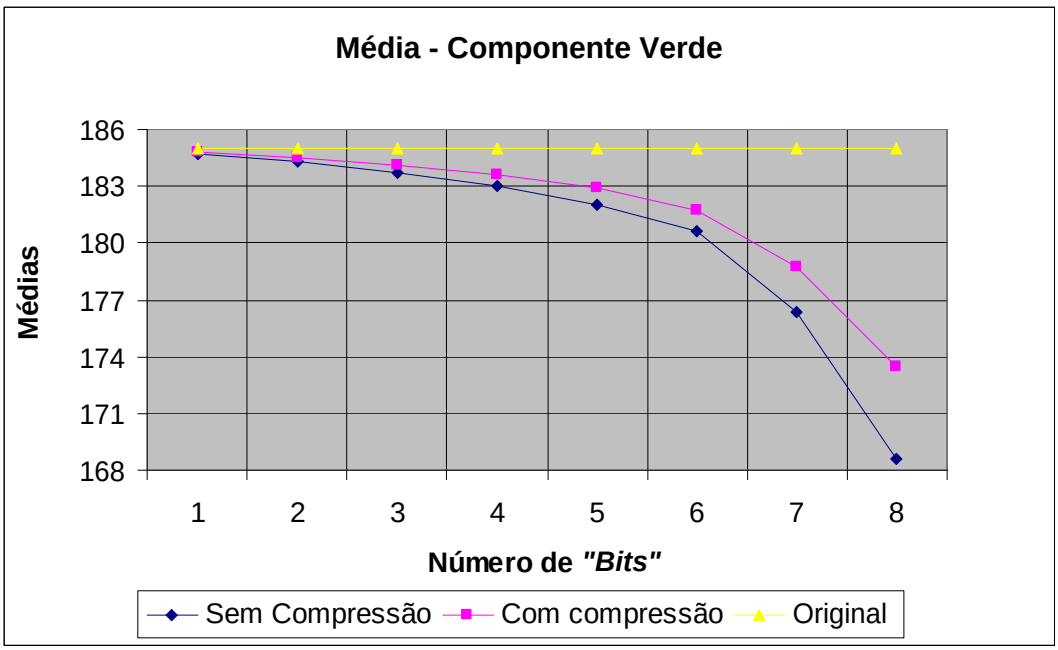


Figura 6.7 – Gráfico comparativo das médias dos histogramas do componente verde

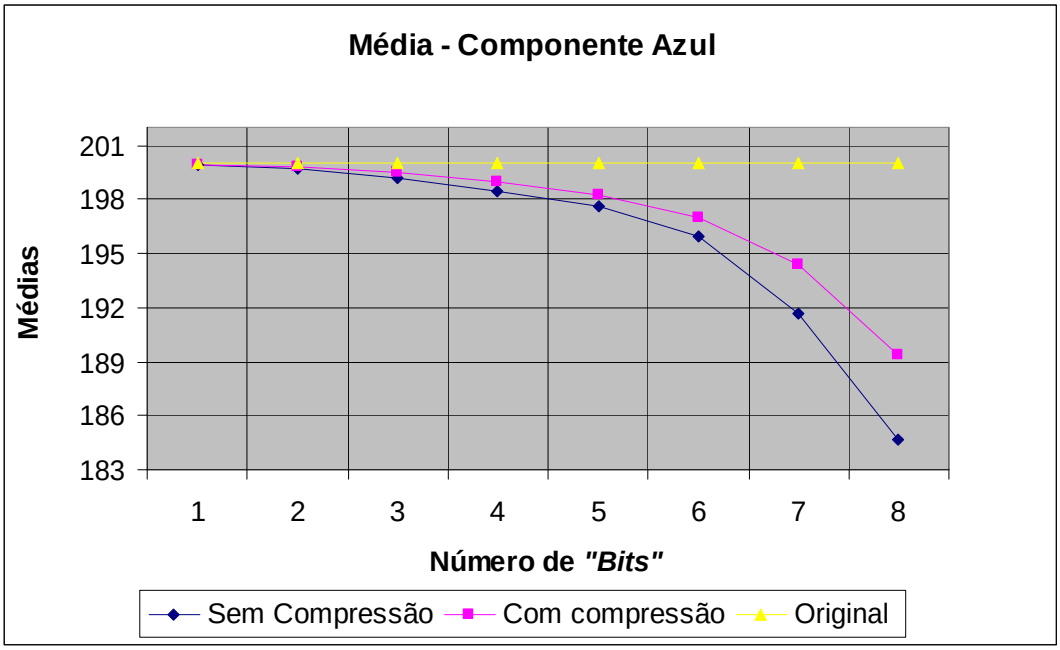


Figura 6.8 – Gráfico comparativo das médias dos histogramas do componente azul

Conforme mostram os gráficos acima, percebeu-se que as médias das estego-imagens decresceram, pois a imagem recipiente utilizada possui predominância nas cores branca e cinza claro, (que são cores com valores alto próximo de 255) e a tendência que esses valores diminuam, quanto maior a utilização de *"bits"* menos significativos, maior é a degradação da imagem, já com a compressão do texto, a degradação diminui.

Se for utilizada uma imagem recipiente que predomina a cor preta (valor baixo, isto é, próximo de 0), o processo é inverso, a tendência que as médias da estego-imagens, através do histograma de cores cresçam

6.4 Performance do Algoritmo de Huffman

6.4.1 Amostras

Para realização desta simulação foi utilizada uma amostra de imagem no formato BMP *"24-bits"*, conforme pode ser visto abaixo:

Figura 6.9 – Imagem de amostra

Foram utilizados cinco amostras de textos literários de Olavo Bilac[30] retiradas de [Rodrigues, 2000] no formato TXT com aproximadamente: 20, 500, 1.000, 5.000, e 10.000 caracteres.

6.4.2 Gráfico

O gráfico, abaixo, em forma de barras, mostra o desempenho do Algoritmo de Huffman, com os dados de entrada citado no item 6.4.1:

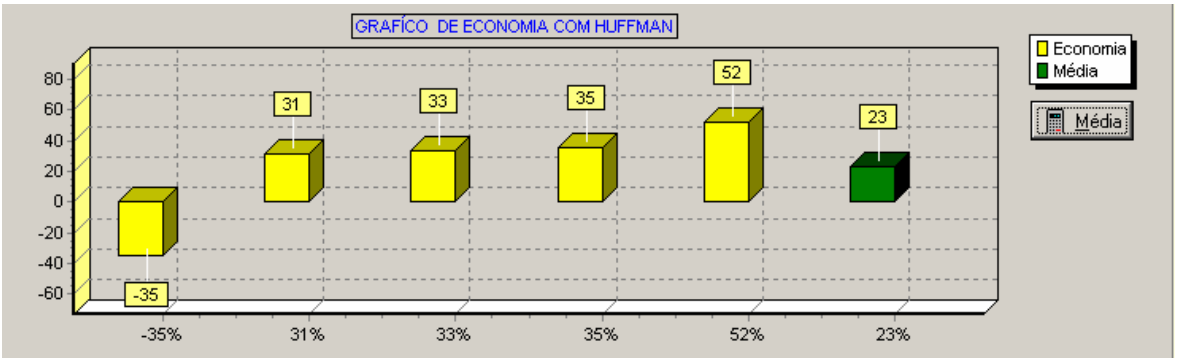


Figura 6.10 – Gráfico de desempenho do Algoritmo de Huffman

Após realização dos testes, verificou-se que o algoritmo de Huffman é mais eficiente para textos com maior quantidade caracteres, dependendo de suas características, como por exemplo, o texto que repete muito determinada palavra ou caracter. Como por exemplo, no teste realizando com 1.000 caracteres teve uma economia de 35%, ou seja, um texto de 1.000 caracteres ocupa um espaço na imagem como se fosse um texto de 650 caracteres.

6.5 Resultados das Imagens Esteganografadas

6.5.1 Amostra

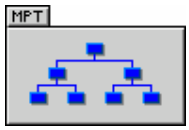


Figura 6.11 – Imagem utilizada para mostrar as estego-imagens

A imagem no formato BMP *"24-bits"* como mostra a figura 6.11 possui resolução de 96 X 64 *"pixels"*, com tamanho de 18 KB.

Foi utilizado um texto literário de Olavo Bilac em formato TXT com aproximadamente 2.500 caracteres, com tamanho 3 KB.

6.5.2 Estego-Imagens

As imagens a seguir, são as imagens resultantes do processo de esteganografia com: 1, 6 e 8 utilizados *"bits"* menos significativos, utilizando os dados de entrada conforme citado no item 6.5.1:

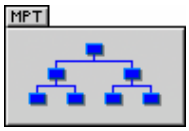


Figura 6.12 – Imagem esteganografada com 1 *"bit"* menos significativo com o texto comprimido

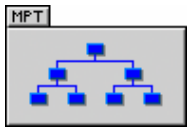


Figura 6.13 – Imagem esteganografada com 1 *"bit"* menos significativo com o texto sem compressão

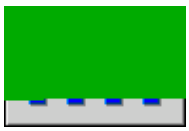


Figura 6.14 – Imagem mostrando os *"pixels"* utilizados para inserção do texto comprimido com 1 *"bit"* menos significativo

Figura 6.15 – Imagem mostrando os *"pixels"* utilizados para inserção do texto com 1 *"bit"* menos significativo sem compressão

[30] Olavo Bilac (Rio de Janeiro RJ, 1865-1918) escritor parnasiano

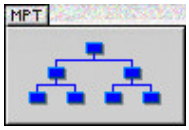


Figura 6.16 – Imagem esteganografada com 6 *"bits"* menos significativo com o texto comprimido

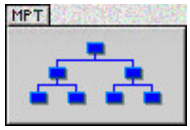


Figura 6.17 – Imagem esteganografada com 6 *"bits"* menos significativo com o texto sem compressão

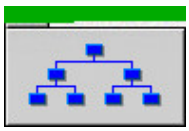


Figura 6.18 – Imagem mostrando os *"pixels"* utilizados para inserção do texto comprimido com 6 "bits" menos significativo

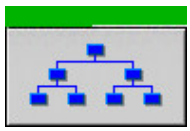


Figura 6.19 – Imagem mostrando os *"pixels"* utilizados para inserção do texto com 6 *"bits"* menos significativo sem compressão

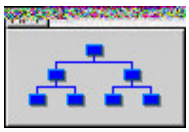


Figura 6.20 – Imagem esteganografada com 8 *"bits"* menos significativo com o texto comprimido

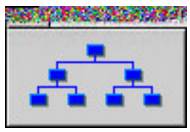


Figura 6.21 – Imagem esteganografada com 8 *"bits"* menos significativo com o texto sem compressão

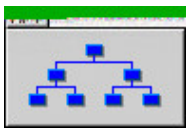


Figura 6.22 – Imagem mostrando os *"pixels"* utilizados para inserção do texto comprimido com 8 *"bits"* menos significativo

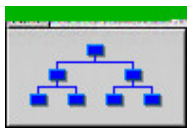


Figura 6.23 – Imagem mostrando os *"pixels"* utilizados para inserção do texto comprimido com 8 *"bits"* menos significativo sem compressão

Percebe-se após esta simulação, que quanto maior o número de *"bits"* menos significativos utilizados, maior o espaço para inserção de texto, porém, quanto maior o número de *"bits"* menos significativos utilizados maior será a degradação da imagem, de forma perceptível ao olho humano.

6.6 Analise dos Resultados Obtidos

6.6.1 Antes da Implementação do Protótipo

Antes do inicio da projeção e implementação do protótipo, foram feitas algumas considerações e alguns problemas foram discutidos junto ao orientador do projeto:

- Se ocorreria troca de letras utilizando o algoritmo de Huffman, pois é método estatístico;
- Qual seria influência no desempenho (tempo de processamento) do algoritmo de Huffman junto ao processo de esteganografia;
- O método de compressão por meio do Algoritmo de Huffman seria eficiente somente em grandes quantidades de caracteres;
- Quais seriam os métodos utilizados para verificar a degradação da imagem;
- Pouco material bibliográfico para realização do projeto a respeito de esteganografia.

6.6.2 Com o Decorrer da Implementação do Protótipo

Com o decorrer da implementação do protótipo, foram solucionados os problemas citados no item 6.6.1:

- Não ocorreram trocas de letras, pois na implementação não foi utilizado a probabilidade de cada letra, mas sim a freqüência de cada letra, possibilitando trabalhar com números inteiros que não precisem de arredondamento e o processamento mais rápido, pois não precisa realizar o cálculo de probabilidade de cada letra;
- Com a compressão de dados por meio do Algoritmo de Huffman foi possível verificar que o desempenho no processo de esteganografia foi agilizado, isto é, menos tempo para processar a imagem esteganografada, pois utiliza menos *"bits"* codificados do texto para a inserção na imagem recipiente;
- Realmente, como citado antes da implementação por parte do professor orientador, o método de compressão por meio do Algoritmo de Huffman seria eficiente somente em grandes quantidades de caracteres;

- O tempo foi suficiente para a implementação do protótipo e a redação da monografia;
- O método de histograma de cores foi definido junto ao orientador do projeto que seria utilizado para verificar a degradação da imagem, com os parâmetros de: média, mediana e desvio padrão;
- Foi realizada a pesquisa bibliográfica na Internet, utilizando principalmente artigos da IEEE e material fornecido pelos professores: MSc. Aderlon M. Queiroz e MSc. Claudio Penedo de Albuquerque

6.6.3 Comparando com Aplicativos Comerciais

Foram utilizados os aplicativos, testados por [Johnson, Jajodia, 1998], para realizar uma comparação com protótipo desenvolvido:

- StegoDos, software livre;
- White Noise Storm, aplicativo desenvolvido pela IBM[31];
- S-Tools, software livre.

A tabela abaixo, mostra o comparativo entre eles:

[31] *"Internacional Business Machines Corporation"*

Tabela 6.5 – Comparativo com Protótipo

Aplicativo	Ambiente Operacional	Imagem Recipiente	Vantagens	Desvantagens
Stego DOS	DOS[32]	Formato BMP 256 cores com resolução de 320x200 *"pixels"*.	- Software Livre;	- Aumenta o tamanho do arquivo;
White Noise Storm	DOS	Formato BMP 256 cores com resolução de 200x200 *"pixels"*.	- Aplica criptografia na mensagem antes de realizar o processo de esteganografia;	- Software pago; - Existe uma perda muito grande de espaço quando se aplica a criptografia na mensagem (overhead);
S-TOOLS	Windows	Formatos BMP e GIF até *"24-Bits"*.	- Software Livre; - Ambiente amigável; - Aplica criptografia na mensagem antes de realizar o processo de esteganografia;	- Utilização de apenas um "bit" menos significativo para realizar o processo de esteganografia; - Existe uma perda muito grande de espaço quando se aplica a criptografia na mensagem (overhead);
Protótipo	Windows	Formato BMP *"24-Bits"*	- Software Livre; - Ambiente amigável; - Aplica compressão na mensagem antes de realizar o processo de esteganografia; - Permiti que o usuário configure a quantidade de *"bits"* menos significativos utilizando no processo de esteganografia.	- Não aplica criptografia na mensagem antes de realizar o processo de esteganografia;

Lembrando, que esses dados a respeito dos programas utilizados, são retirados do artigo da IEEE [Johnson, Jajodia, 1998].

6.6.4 Observações Finais do Projeto

Segundo [Rocha, 2003] [Johnson, Jajodia, 1998], os seres humanos conseguem capturar mudanças em uma imagem quando estas ocorrem em fator acima de 3%.

Exemplo, se o protótipo trabalha apenas com um *"bit"* menos significativo, o conjunto total de mudanças no caso de uma mensagem que afete todos os LSBs é de apenas 0,78% ou (2/256) % — dado que cada componente de cor, tenha 8 *"bits"* a alteração no último *"bit"*. Caso o segundo *"bit"* menos significativo também seja alterado, a taxa de

[32] Sistema Operacional mono tarefa e monousuário desenvolvido pela IBM em 1981

alteração aumenta para 1,56% ou (4/256) %, ainda imperceptível à maioria dos seres humanos.

Conforme objetivo proposto foi pesquisado a maior quantidade de caracteres esteganografados na imagem sem que ocorra uma degradação perceptível ao olho humano, pode-se concluir que utilizando até 2 *"bits"* menos significativos de cada componente RGB é seguro, conforme cita [Rocha, 2003] [Johnson, Jajodia, 1998].

7. Conclusão

7.1 Considerações Finais

Este trabalho apresentou uma síntese da evolução da esteganografia, ao longo da história e algumas de suas aplicações modernas. Foram mostradas algumas das principais técnicas de mascaramento especialmente as de mascaramento em imagens.

Também foi mostrado o protótipo: Esteganografia - Inserção de texto comprimido em Imagem Digital desenvolvido durante o trabalho.

E finalmente, fez-se uma análise das estego-imagens produzidas pelas ferramentas definidas juntamente com orientador.

Associando a criptografia e a esteganografia, têm-se o poder de comunicação secreta pela rede mundial de computadores com o mais absoluto sigilo. Obviamente, a privacidade pode ser aproveitada para fins ilícitos conforme apresentado no capítulo 3, no entanto, o papel do autor deste trabalho, enquanto acadêmico, é atuar em prol da sociedade. A própria sociedade, deverá fazer as suas escolhas e aplicar o conhecimento da forma que lhe aprouver, ou seja da forma lícita.

O protótipo desenvolvido, ao ser disponibilizado em código aberto, adquire um imenso potencial didático, dando a qualquer interessado a possibilidade de aprofundar-se no assunto estudado que é a esteganografia.

7.2 Propostas Futuras

O campo de pesquisas em esteganografia digital está em constante evolução. Novas técnicas são inovadas a cada dia. Neste sentido, apresentam-se algumas melhorias que podem ser desenvolvidas e adequadas ao protótipo.

7.2.1 Códigos corretores de erros

Um dos grandes problemas da esteganografia consiste em recuperar a mensagem escondida após um ataque geométrico à estego-imagem.

Uma das saídas possíveis é aplicar códigos corretores de erro que possibilitem a recuperação da mensagem sem a necessidade de todos os *"bits"* estarem presentes no lado

receptor. Dado que alguns *"bits"* tenham sido perdidos durante o ataque geométrico[33], é possível tirar certas conclusões a partir dos códigos corretores.

7.2.2 Criptografia

Um sistema esteganográfico torna-se bastante seguro se associado à criptografia. Se a mensagem, antes de ser mascarada, for criptografada segundo uma chave ela estará bem segura.

7.2.3 Outros formatos de imagens

Atualmente, o protótipo produz estego-imagens de *"24-bits"* no formato BMP. Isto quer dizer que o mascaramento só é possível em imagens cuja compactação é do tipo "*lossless"* sem perda de dados. Conseqüentemente, as imagens são maiores que a maioria em circulação pela "*Internet"*. Como uma forma de reduzir as imagens produzidas, propõe-se a possibilidade de mascarar mensagens também em imagens passíveis de compactação "*lossy"* — com possível perda de dados

7.3.4 Mascaramento de sons

A técnica de mascaramento implementadas no protótipo trabalha com arquivo de imagem. Este mascaramento pode estender-se, no mesmo protótipo, mensagens em arquivos de sons. Considerando que o funcionamento de um arquivo de som é bastante parecido com um arquivo de imagem, para a implementação desta proposta não são necessários muitas alterações na versão atual.

[33] Ataque que consiste em alterar as dimensões da imagem

Referências Bibliográficas

AMIM, M. M.; SALLEH, M.; IBRAHIM, S.; KATMIN, M. R.; SHAMSUDDIN, M. Z. I. et. al. **Information Hiding using Steganography**. Artigo IEEE Computer Science – 4. º National Conference on Telecommunication Technology Proceedings, p. 21-25, 2003.

BASIC, Flavia; AUGUSTA, Izabel; SPENCER, Renata et. al. **Formato de Arquivos Gráficos**. Disponível em: <*http://www.di.ufpe.br/~if291/documentos/formatos_graficos/formatos.htm*>. Acesso em: 8 de setembro de 2004.

BENTO, Ricardo J.; COELHO, Laura M. **Ferramentas de Esteganografia e seu uso na Infowar**. ICCyber'2004 – I Conferencia Internacional de Perícia em Crimes Cibernéticos. Setembro, 2004.

CASACURTA, Alexandre. **Computação Gráfica - Introdução**. Disponível em: <*http://www.inf.unisinos.br/~osorio/CG-Doc/cg.pdf* >. Acesso em: 4 de Setembro de 2004.

CAVERSAN, Fábio L. **Segmentação de Imagens**. Disponível em: <*http://www.facens.br/site/ensino/projetos/fabio/*>. Acesso em: 10 de Outubro de 2004.

CHAPMAN, Stephen. J. **Programação em MATLAB para Engenheiros**. Editoria Thompson. São Paulo, 2003.

CONCI, Aura. **Computação Visual e Processamento de Imagens**. Disponível em: <*http://easygrid.ic.uff.br/~aconci/CV.html*>. Acesso em: 4 de Setembro de 2004.

FARIA, Aloar M..; MOLINA, André L. **Esteganografia em Diferentes Meios**. Disponível em: <*http://www.ene.unb.br/~juliana/cursos/pimagens2/projetos/alunos/alaorandre/index.htm*>. Acesso em: 15 de agosto de 2004.

FRIDRICH, Jessica; GOLJAN, Miroslav.; DU, Rui et al. **Detecting LSB Steganography in Color and Gray Scale Images**. Artigo IEEE Multimedia, p. 22-28, Dezembro 2001.

GOIS, Mateus C. de A. **Mascaramento de Informações: histórico, definições e aplicações**. UFFE – Universidade Federal de Pernambuco. 2003.

HAYASHIDA, Ulisses K. **Projeto de Iniciação Cientifica**. Disponível em: <*http://www.linux.ime.usp.br/~cef/mac499-01/monografias/ulisses/*> . Acesso em: 16 de agosto de 2004.

JOHNSON, N.; JAJODIA, S. **Exploring steganography: seeing the unseen**. Artigo IEEE. Computing Practices, Vol. 31, p. 26-34, Fevereiro 1998.

MASTRONARDI, Giuseppe; CASTELLANO, Marcello; MARINO, Francescomaria et al.. **Steganography Effects *in* Various Formats of Images. A preliminary Study.** Artigo IEEE Computer Science – Internacional Workshop on Intelligent Data Acquisition Computing System: Technology and Applications, p. 116-119, Julho 2001.

MOURA, Milton G. C. **Códigos de Huffman**. Universidade dos Açores – Departamento de Matemática. Licenciatura Informática. 2004.

MULLER, Daniel N. **Compressão de Dados**. Disponível em: <*http://www.ulbra.tche.br/~danielnm/ed/E/polE.html*>. Acesso em: 26 de agosto de 2004.

NIIMI, Michiharu; NODA, Hideki; KAWAGUCHI, Eiji; EASON, Richard O. et. al. **High Capacity and Secure Digital Steganography to Pallete-Based Images**. Artigo IEEE, Vol. 2, p. 917-920, 2002.

OLIVEIRA, Marcelo V. **Formato de Arquivo BMP**. Disponível em: <*easygrid.ic.uff.br/~aconci/curso/bmp.pdf*>. Acesso em: 4 de Setembro de 2004.

PETICOLAS, R. J.; ANDERSON, F. A. P. et. al. **On the limits of steganography**. Artigo IEEE Journal of Selected Areas in Comunications, Vol. 16, p. 474-481, março 1998.

ROCHA, Anderson R. **Projetos de Iniciação Científica**. Disponível em: <*http://www.dcc.unicamp.br/~ra030014/ic/estego/index.php3*>. Acesso em: 2 de setembro 2004.

RODRIGUES, Luiz. **As Tormentas**. Disponível em: http://www.astormentas.com/din/poemas.asp?autor=Olavo+Bilac>. Acesso em: 14 de outubro 2004. São Paulo.

TENENBAUM, Aaron M.; AUGENSTEIN, Moshe J.. **Data Structures Using Pascal**. Prentice-Hall. New Jersey – 2. º Edição, EUA. 1986.

TENENBAUM, Aaron M.; AUGENSTEIN, Moshe J; LANGSAM, Wedidyah. **Estrutura de Dados Usando C**. Makron Books. São Paulo. 1995.

VENKATRAMAN, S.; Ajith, ABRAHAM; Marcin, PAPRZYCKI et. al. **Significance of Steganography on Data Security**. Artigo IEEE Computer Society – Proceedings of the Internacional Conference on Information Technology, 2004.

WANDERLEY, Juliana F. C. **Processamento de Imagem**. Disponível em: <*http://www.ene.unb.br/~juliana/cursos/pimagens2/index.html*>. Acesso em: 5 de Setembro de 2004. Universidade de Brasília.

Anexo A – Estrutura Detalhada do Formato *"Bitmap"*

A seguir descreve-se detalhadamente, as informações encontradas em cada *"byte"* de cada uma das partes do arquivo BMP [Oliveira, 2000]:

a) Cabeçalho de arquivo - informações do arquivo, tamanho de 14 *"bytes"*:

Tabela A.1 – Cabeçalho de Arquivo do Tipo *"Bitmap"* [Oliveira, 2000]

Campo	*"Byte"*	Descrição
BfType	2	Assinatura do arquivo: os caracteres ASCII "BM" ou $(42\ 4D)_h$,é a identificação de ser realmente BMP.
BfSize	4	Tamanho do arquivo em *"bytes"*.
BfReser1	2	Campo reservado 1: deve ser zero.
BfReser2	2	Campo reservado 2: deve ser zero.
BfOffSetBits	4	Especifica o deslocamento, em bytes, para o início da área de dados da imagem, a partir do início deste cabeçalho: - Se a imagem usa paleta, este campo tem tamanho = 14+40+ (4 x Número de Cores); - Se a imagem for *"true color"*, este campo tem tamanho = 14+40 = 54.

b) Cabeçalho de mapa de *"bits"* - informações da imagem, tamanho de 40 *"bytes"*:

Tabela A.2 – Cabeçalho mapa de *"Bits"* [Oliveira, 2000]

Campo	*"Byte"*	Descrição
BiSize	4	Tamanho deste cabeçalho (40 *"bytes"*), sendo sempre (28_h).
BiWidth	4	Largura da imagem em *"pixels"*.
BiHeight	4	Altura da imagem em *"pixels"*.
BiPlanes	2	Número de planos de imagem, sendo sempre 1.
BiBitCount	2	Quantidade de *"bits"* por *"pixel"* (n=1, n=4, n=8, n=24, n=32). Este campo indica, indiretamente, ainda o número máximo de cores que é determinado por 2^n
BiCompress	4	Compressão usada que pode ser: 0 = BI_RGB – sem compressão; 1 = BI_RLE8 – compressão RLE 8 *"bits"*; 2 = BI_RLE4 – compressão 4 *"bits"*.
BiSizeImag	4	Tamanho da imagem (dados) em *"byte"*: - se arquivo sem compressão, este campo pode ser zero; - se imagem em *"true color"*, será tamanho do arquivo (Bfsize) menos deslocamento (BfOffSetBits).
BiXPPMeter	4	Resolução horizontal em *"pixels"* por metro.
BiYPPMeter	4	Resolução vertical em *"pixels"* por metro.
BiClrUsed	4	Número de cores usadas na imagem. Quando zero indica o uso do máximo de cores possível pela quantidade de *"bits"* por *"pixel"*, que é determinado por 2^n
BiClrImpor	4	Número de cores importantes (realmente usadas) na imagem. Por exemplo das 256 cores, apenas 200 são efetivamente usadas. Se todas são importantes pode ser zero. É útil quando for exibir uma imagem em 1 dispositivo que suporte menos cores que a imagem possui.

c) Área de dados da imagem - cor que cada *"pixel"* deve ser ligado ou esses dados comprimidos, tamanho de campo BiSizeImg, do cabeçalho de informações da imagem.

Esta área do arquivo de imagens varia conforme existência de compressão [Oliveira, 2000].

No BMP *"true color"* (*"24-bits"*) cada seqüência de 3 *"bytes"* corresponde a uma seqüência *"Blue"* (azul), *"Green"* (verde) e *"Red"* (vermelho), isto é, a composição da cor do *"pixel"* diretamente [Oliveira, 2000].