UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE SOFTWARE PARA INSERÇÃO E EXTRAÇÃO DE MENSAGENS EM ARQUIVO RASTER ATRAVÉS DE ESTEGANOGRAFIA

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

DANIEL ZANELLA

BLUMENAU, NOVEMBRO/2002

2002/2-12

PROTÓTIPO DE SOFTWARE PARA INSERÇÃO E EXTRAÇÃO DE MENSAGENS EM ARQUIVO RASTER ATRAVÉS DE ESTEGANOGRAFIA

DANIEL ZANELLA

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Paulo César Rodacki Gomes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Paulo César Rodacki Gomes

_____

Prof. Dalton Solano dos Reis

_____

Prof. Francisco Adell Péricas

DEDICATÓRIA

Dedico este trabalho, com carinho, àqueles que
o tornaram possível, mediante o apoio direto e
o constante estímulo que me motivaram a realizá-lo

AGRADECIMENTOS

Primeiramente a Deus, responsável por toda nossa existência.

Aos meus pais, irmãos e cunhado, que sempre me incentivaram e ajudaram, dando-me condições necessárias para que eu pudesse concluir este curso.

Em especial a meu pai, Prof. Fiorelo Zanella, Mestre em Letras, pela revisão gramatical do meu trabalho e por ter sempre acreditado em mim.

Ao meu grande amigo Alexandre, que na medida do possível me ajudou e incentivou na conclusão deste trabalho.

Ao professor e orientador Paulo César Rodacki Gomes, que com muita dedicação e sabedoria conduziu-me na realização deste trabalho de conclusão.

A todos os professores do Curso de Computação, construtores do conhecimento, tão importante na formação profissional do acadêmico.

Aos amigos acadêmicos, companheiros de caminhada, incentivadores indispensáveis no cumprimento de mais uma importante etapa em nossas vidas.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

RESUMO

Este trabalho consiste num estudo de métodos e conceitos de esteganografia, sendo sua aplicação voltada na inserção e extração de mensagens em arquivos *raster*. Com esse propósito foi feito um estudo também sobre tipos de imagens *raster* e cores. Ao final, um algoritmo foi desenvolvido para inserir e extrair uma mensagem em uma imagem utilizando-se da técnica *Least significant bit (LSB) insertion*.

ABSTRACT

The present work consist on a study of methods and concepts of steganography. Steganography is a set of techniques and methods to insert and extract hidden messages in raster files. Also, this work presents a study about raster images and colors. A software prototype was developed in order to insert and extract a text message inside image files, using the Least significant bit (LSB) insertion technique.

1 INTRODUÇÃO

Quando foram criados, os computadores eram utilizados para calcular e imprimir números, desde então a utilização se diversificou muito. Nos últimos anos, com o surgimento de novas tecnologias, a utilização dos recursos de computação gráfica se tornou indispensável para profissionais de diversas áreas (Brown, 1995).

Com a necessidade da utilização de funções gráficas, cresceu muito o estudo da Computação gráfica que é a área da ciência da computação que estuda a aquisição (síntese), manipulação (processamento) e interpretação (análise) de imagens por meio de computadores. Para ser possível manipular ou interpretar estas imagens é necessária a utilização de alguma técnica de armazenamento e especificação das figuras, que torne possível sua visualização em algum dispositivo computacional (Persiano, 1989) e (Banon, 1989).

Segundo Franco (1998), na antiga Grécia, tábuas recobertas de cera eram usadas como bloco de notas até que alguém descobriu que era possível utilizá-las para enviar mensagens secretas: bastava raspar a cera, escrever diretamente sobre a madeira e então recobrir a tábua novamente com cera. A utilização de tinta invisível para esconder uma mensagem secreta nas entrelinhas de uma carta aparentemente inocente é também muito antiga. A técnica do microponto, inventada pelos alemães na Segunda Guerra Mundial, permitia disfarçar num simples ponto uma microfotografia contendo textos e figuras suficientes para preencher uma página.

O que há de comum nesses exemplos, são mensagens que não são notadas por um observador desavisado. Esse é o domínio da esteganografia (do grego s*teganos* = escondido + *grafia* = escrita), segundo Moniz (2002), uma técnica que consiste em esconder (codificar) um texto dentro do código do arquivo de uma imagem. Através de um programa especial, é possível alterar a forma em que os *bytes* correspondentes a uma imagem são gravados de forma a incluir uma informação, aproveitando *bits* normalmente não utilizados.

Diferentemente da criptografia, cujo objetivo é esconder o conteúdo de mensagens, a esteganografia preocupa-se em esconder a própria existência da mensagem ou alguma característica associada (remetente, destinatário, local ou instante do envio, etc.), tirando vantagem do fato de o olho humano não conseguir distinguir alguns tipos de transformações.

O advento dos computadores e a crescente digitalização das informações trouxeram novos papéis para a esteganografia: a proteção de direitos autorais; a identificação de cópias de obras digitais como filmes, gravações de áudio, livros; e a inclusão confidencial de dados em imagens *raster*, arquivos de vídeo e arquivos de áudio.

A informação pode ser escondida de diferentes formas, dependendo do tipo de mídia e da técnica a ser utilizada. No que se refere a esteganografia em imagens, são citadas algumas técnicas descritas por Sellars (1999): *Least significant bit (LSB) insertion, Masking and filtering techniques* e *Algorithms and transformations.* Já na esteganografia em áudio, são citadas pelo mesmo autor as técnicas: *Low bit encoding, Phase encoding, Spread Spectrum e Echo data hiding.*

1.1 JUSTIFICATIVAS

A esteganografia utiliza-se de técnicas que atraem a atenção não só de cientistas na área da Computação Gráfica, mas também de profissionais de todas as áreas que necessitem enviar e receber mensagens com alto grau de confidencialidade. Essas técnicas ainda estão sendo aperfeiçoadas e encontram-se pouco divulgadas no meio científico. O trabalho pretende ser uma contribuição à comunidade acadêmica no que se refere à arte de ocultar informações.

1.2 OBJETIVO

O trabalho proposto tem como objetivo principal implementar um protótipo de software que permita carregar uma imagem *raster* em seu ambiente, para inserir uma mensagem utilizando-se da técnica de esteganografia *Least significant bit (LSB) insertion* e depois de concluída esta operação, extraí-la. Durante o processo de inserção será informada uma senha para que a operação seja bem sucedida. O mesmo acontecerá no processo de extração da mensagem, onde para visualizar a mensagem será preciso validar a operação com a senha referida.

1.3 ESTRUTURA

O trabalho é estruturado da forma a seguir.

No primeiro capítulo, é apresentada uma visão geral deste trabalho, justificativas, objetivos e a sua organização.

O segundo capítulo descreve conceitos sobre imagens *raster* e seus principais formatos, bem como conceitos sobre cores, destacando síntese e representação destas imagens.

No terceiro capítulo serão abordados conceitos sobre esteganografia e suas principais técnicas, características e aplicações.

No quarto capítulo, descreve-se a especificação, implementação e o funcionamento do protótipo.

No quinto capítulo, encontram-se relacionadas as considerações finais, análises, conclusões e sugestões para estudos futuros.

2 IMAGENS RASTER E CORES

2.1 IMAGENS RASTER

A computação gráfica é geralmente voltada a gerar, processar (tratar) ou interpretar informações visuais. Estas informações estão contidas dentro de uma imagem, a qual é uma representação gráfica, plástica ou fotográfica de pessoa ou de objeto (Velho, 1994).

Para entender melhor como as imagens são adquiridas e armazenadas, será visto como estas imagens são representadas digitalmente. Representar uma imagem consiste em modelar esta imagem de forma que os dados que a constituem possam ser interpretados pelo computador (França Neto, 1998).

Arquivos de armazenamento de imagens podem ser classificados em função do tipo de imagem armazenada e seu modo de exibição. Existem duas grandes categorias de imagens: as de formato *raster* e as de formato vetorial.

A imagem gráfica com formato *raster* (2D/3D) é a mais comum tecnologia de representação de imagens em uso hoje, pois torna possível representar os efeitos de cor, luz e sombra nos objetos realísticos e permite manipular o menor detalhe da figura. Um arquivo *raster* pode ser considerado como sendo uma matriz cujos índices de linhas e de colunas identificam um ponto na imagem, e o correspondente valor do elemento da matriz identifica a cor naquele ponto. As figuras normalmente especificadas por este meio são imagens capturadas através de dispositivos de entrada gráfica (*scanners*, câmeras de vídeo etc.) para serem modificadas através da aplicação de técnicas de manipulação de suas características originais. A esta manipulação chama-se de processamento de imagens (Persiano, 1989).

Enquanto imagens vetoriais são descritas em termos de equações matemáticas, as imagens *raster* consistem em uma matriz retangular de pontos coloridos (*pixels*) que quando observados a uma distância apropriada, apresentam uma aceitável representação da realidade, como numa fotografia. Para cada *pixel* são atribuídos um local e um valor de cor específicos.

As imagens *raster* são geralmente armazenadas sob a forma digital através de uma seqüência numérica que identifica a cor de cada um de seus pontos. O número de uns e zeros (*bits*) usados para criar cada *pixel* representa a profundidade da cor que se coloca na imagem.

A quantidade de *bits* usada para representar cada *pixel* é chamada de resolução de cor ou profundidade de *bits* (*bit depth* ou *pixel depth*). As resoluções de cor mais usadas atualmente são 1, 2, 4, 8,16, 24 e 32 *bits*.

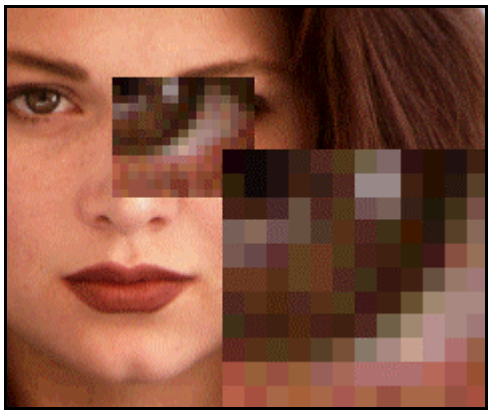
Dependendo da quantidade de cores da imagem ou do seu tamanho, a seqüência numérica de *bits* será maior ou menor. Se a imagem possuir apenas duas cores será necessário apenas um *bit* para identificação de cor. Entretanto, desejando-se definir *pixels* a partir de uma coletânea de 16.777.216 cores disponíveis, serão necessários 24 *bits* para a identificação da cor de cada *pixel*.

Quanto maior o número de pontos (*pixels*) da matriz, maior a quantidade de *bits* que comporão o arquivo, pois para cada um dos pontos haverá um certo número de *bits* representando a cor do mesmo. Uma imagem de 300x200 *pixels* com 256 cores (8 *bits*), por exemplo, terá 60.000 *pixels*, cada um representado por uma seqüência de 8 *bits* (Macedo, 1998).

A qualidade de uma imagem *raster*, ilustrado na fig. 1, é definida por duas variáveis:

a) densidade de pontos na matriz, normalmente definida em *pixels* por polegada (dpi);
b) resolução espectral ou número de cores, normalmente definido por um número de *bits* disponíveis para sua codificação (2,4,8,16 e 24).

Figura 1 – Qualidade de uma imagem *raster*



Fonte: (Macedo, 1998)

2.1.1 TIPOS DE IMAGENS RASTER

Quando imagens são capturadas eletronicamente por câmeras digitais, scanners, etc, ou geradas por programas, elas são sempre transferidas para um computador, onde ficam armazenadas em arquivos. Diferentes fabricantes de equipamentos digitais e programas de computador desenvolveram uma grande quantidade de formatos de arquivos. Os formatos de arquivos descrevem como as imagens *raster* são organizadas dentro do disco ou da memória do computador.

De acordo com Murray (1994), os arquivos de imagens *raster* variam em seus detalhes, mas todos possuem a mesma estrutura geral. Estes arquivos consistem de um cabeçalho, dados da imagem e outras informações.

O cabeçalho encontra-se no início do arquivo e contém as informações sobre os dados especificados em todos os outros lugares do arquivo. É iniciado com um identificador do arquivo, através de um número atribuído pelo criador do formato, identificando-o na aplicação. Normalmente esse número é único, independente da plataforma que está sendo utilizada. O cabeçalho está na estrutura de todos os arquivos *raster*, variando apenas a informação descrita nele de formato para formato. Normalmente, o cabeçalho é composto de campos fixos e possui informações como o mapa de cores[1], o identificador do arquivo, a versão do arquivo, o número de linhas por imagem, o número de *pixels* por linha, o número de *bits* por *pixel*, o tipo de compressão utilizado, e outras.

Nos muitos formatos existentes, os dados da imagem *raster* vêm logo após o cabeçalho. Mas isto nem sempre é estruturado assim, pois estes dados poderiam estar em qualquer outro lugar do arquivo, acomodando o mapa de cores ou outra estrutura de dados presente. Quando os dados não vêm logo após o cabeçalho, nele está indicada a posição inicial destes dados.

O rodapé é outra estrutura que geralmente compõem os arquivos *raster*. Ele é similar ao cabeçalho, só que se encontra no final do arquivo. É normalmente acrescentado ao formato

[1] Mapa de cores ou paleta de cores é uma estrutura que declara os componentes RGB das cores utilizadas pela imagem a que se encontram associadas, atribuindo a cada cor um índice.

quando se necessita de uma extensão para acomodar novos tipos de dados. Sua localização é determinada com a fixação de um *offset* no final do arquivo, em virtude do rodapé vir sempre após os dados.

Os arquivos de imagens *raster* são os mais apropriados para o armazenamento de imagens referentes ao mundo real. Segundo Murray (1994), eles possuem as seguintes vantagens:

a) podem ser criados sem dificuldades a partir de vetores de *pixels* existentes na memória;
b) a alteração dos dados pode ser feita de forma individual ou a partir de grandes grupos, ocasionando alterações na paleta de cores, se esta estiver presente na estrutura;
c) podem ser facilmente traduzidos para dispositivos de saída como tubos de raios catódicos e impressoras.

Como desvantagens, os arquivos de imagens *raster* apresentam as seguintes:

a) seu tamanho pode ser muito grande, relacionado com a quantidade de cores que a imagem possuir. Quanto maior a quantidade de cores que a imagem possui, maior será seu tamanho. A redução do tamanho dos dados a serem armazenados pode ser obtida através de técnicas de compressão de dados, mas como estes dados terão de ser descompactados para que possam ser usados, sua utilização se torna mais lenta;
b) como muitas vezes é difícil modificar o tamanho das imagens, torna-se conveniente imprimir a imagem na sua resolução original.

Dentre os vários tipos de formatos de arquivos para imagens *raster* destacam-se os do tipo BMP, GIF e JPEG.

2.1.1.1 BMP (*WINDOWS BITMAP*)

O formato BMP, também conhecido como formato DIB (*Device Independent Bitmap*), é segundo Lopes (2002), um formato proprietário da Microsoft sendo compatível ao sistema operacional Windows.

Este formato permite descrever imagens a cores com 1, 4, 8, 16, 24 ou 32 *bits* por *pixel*, representando assim imagens com 2, 16, 256, 2^{16}, 2^{24} ou 2^{32} cores, respectivamente, utilizando um mapa de cores em todos os casos, exceto nos casos com 16, 24 e 32 *bits*. Nos arquivos com 16, 24 ou 32 *bits* por *pixel*, existem máscaras para cada cor no lugar do mapa de cores. O valor da máscara é usado para fazer um AND lógico com o valor do *pixel*, conseguindo capturar assim a intensidade de uma cor específica no *pixel*. O formato BMP permite ainda a compressão opcional sem perdas do conteúdo de imagens com 16 ou 256 cores através do algoritmo RLE (*Run Length Encoding*) adaptado ao número de *bits* por *pixel* (4 ou 8) (Murray, 1996).

Segundo Lopes (2002), toda a informação pertencente a um arquivo de formato BMP está estruturada em três blocos, conforme fig. 2, podendo ser inseridos uma única vez, sendo obrigatoriamente colocados no arquivo na seguinte ordem:

a) Bitmap File Header, que descreve o arquivo;
b) Bitmap Info, que descreve o bitmap da imagem. É formado pelo sub bloco Bitmap Info Header, que descreve a imagem presente no arquivo, e pelo sub bloco Win3ColorTable que contém o mapa de cores da imagem (exceto com 24 *bits* por *pixel*);
c) dados da imagem, que são formados pelos *pixels* da imagem, comprimido ou não por meio do algoritmo RLE.

Figura 2 – Organização de um arquivo BMP

Fonte: (Agostini, 1999)

2.1.1.2 GIF (*GRAPHICS INTERCHANGE FORMAT*)

O formato *Graphics Interchange Format* (GIF) foi criado, segundo Brown (1995), pela Compuserve Inc. com o objetivo de transmitir imagens do tipo *raster* pela Internet. A primeira versão do GIF surgiu em 1987 (GIF87a). Dois anos após, a mesma empresa lançou a especificação GIF89a, que implementava o recurso da cor transparente. Segundo consta, este é provavelmente o formato de arquivos gráficos de maior popularidade.

De acordo com Lopes (2002), o formato GIF permite armazenar ou transmitir imagens com no máximo 256 cores, sempre definidas através de mapas de cor. A sua característica predominante é suportar apenas 8 *bits* por *pixel*, no máximo. Apesar desta limitação, o GIF ainda é o formato mais usado para armazenar imagens de baixa resolução. Cada *pixel* da imagem no formato GIF contém um índice correspondente ao número de ordem da sua cor no mapa de cores. Outro aspecto importante é que o conjunto dos índices que compõem cada imagem está comprimido pelo algoritmo de *Lempel-Ziv Welch* (LZW), que é um algoritmo de compressão sem perda de informação: uma imagem GIF pode ser lida e gravada, infinitas vezes, e sempre será idêntica à original.

Segundo Murray (1994), o formato GIF é diferenciado dos demais arquivos *raster*, pois é baseado sob o conceito de *Data Stream*, ou canal de dados. Estes consistem de uma série de pacotes de dados, chamados blocos, que armazenam informação adicional ao protocolo. Ou seja, o formato é na verdade um protocolo entre uma fonte emissora de imagens e uma aplicação de destino responsável pela apresentação das imagens.

Uma outra característica particular de um arquivo GIF, é que possibilitam conter mais do que uma imagem em sua estrutura. Para que isso seja possível, existem blocos de controle inseridos entre as imagens determinando o tempo que cada imagem persistirá na unidade gráfica de saída, permitindo assim a apresentação de várias imagens seqüencialmente. Quando as imagens forem uma seqüência animada, esta é designada usualmente por GIFs animados *(animated GIF*s).

Conforme descreve Lopes (2002), a informação contida num arquivo GIF está organizada em Blocos Lógicos, divididos em dois tipos:

a) Blocos de Controle (*Control Blocks*);

b) Blocos de Processamento de Imagem (*Image Rendering*).

2.1.1.2.1 BLOCOS DE CONTROLE

É nesta estrutura onde se encontra toda a informação necessária para que a aplicação possa processar os dados da imagem e controlar as unidades periféricas de representação gráfica. Os blocos de controle são divididos em:

a) *Header* (Cabeçalho), que identifica o início de um fluxo de dados em formato GIF. Tem presença obrigatória nos arquivos GIF, existindo um para cada arquivo;
b) *Logical Screen Descriptor*, que tem por finalidade definir as características e *configuração* que a unidade gráfica de saída deve possuir e os valores de parâmetros globais para todo o processamento. É obrigatoriamente colocado logo após do bloco *Header*;
c) *Graphic Control Extension*, que tem por finalidade controlar a seqüência de apresentação das imagens presentes no arquivo de dados GIF. Sua presença é opcional;
d) *Trailer*, que determina o final de um fluxo de dados em formato GIF.

2.1.1.2.2 BLOCOS DE PROCESSAMENTO DE IMAGEM

Os blocos de Processamento de Imagem *(Image Rendering*) são responsáveis pela declaração dos parâmetros de cada imagem e seus respectivos conteúdos. São divididos em:

a) *Image Descriptor*;
b) *Color Table*;
c) *Table Based Image Data*;
d) *Plain Text Extension*.

Conforme consta, cada imagem presente na estrutura do formato GIF é descrita por um bloco *Image Descriptor*, seguido, caso houver necessidade, por um bloco *Color Table* que contém o mapa local de cores dessa imagem, e por último, seguido por um bloco do tipo *Table Based Image Data* ou *Plain Text Extension* que descrevem o conteúdo da imagem. A definição ou não de um mapa local de cor para cada imagem é opcional. Se este for definido, a imagem será apresentada conforme as cores definidas pelo mapa. Quando não existir mapa local de cores definido, a imagem será apresentada com as cores determinadas pelo mapa global de cores, sendo de presença obrigatória (Lopes, 2002).

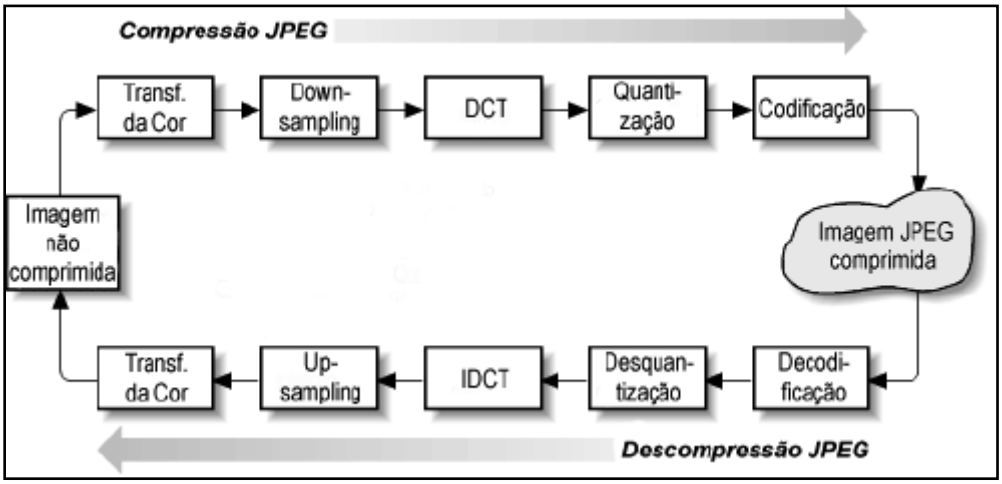
2.1.1.3 JPEG E JFIF (JPEG FILE INTERCHANGE FORMAT)

Conforme Lopes (2002), a norma internacional JPEG (ISO 10918-1) criada em 1990, define uma família de algoritmos de compressão e descompressão, com ou sem perda de informação, para imagens de sujeitos reais (de qualidade fotográfica), tanto coloridas como em escala de cinza. O JPEG permite graus de compressão de 10:1 a 20:1 sem perdas visíveis na qualidade da imagem. Graus de compressão de 30:1 a 50:1 podem ser atingidos com perda moderada de qualidade, enquanto imagens com qualidade baixa podem ser geradas permitindo uma compressão de 100:1. Segundo Miano (1999), uma foto de 1MB no formato BMP não comprimido pode chegar a menos que 50KB com o uso da compressão JPEG.

A sigla JPEG deriva, segundo Murray (1996), do nome da comissão internacional de normalização, o *Joint Photographic Experts Group*, sendo oficialmente nomeado de *Joint Committee* ISO/IEC JTC1 SC 29 *Working Group* 1. Ao contrário do que muitos imaginam, não existe nenhum formato JPEG, pois o mesmo não é definido como formato pela norma ISO 10918-1 e sim como algoritmos de compressão e descompressão.

A operação completa de compressão e descompressão da imagem definida no padrão JPEG está apresentada na fig. 3.

Figura 3 – Operação completa de compressão e descompressão JPEG



Fonte: (Agostini, 1999)

O primeiro passo para a compressão JPEG segundo Agostini (1999) é a transformação do espaço de cores usado na imagem não comprimida para o formato YCbCr. O padrão JPEG suporta qualquer tipo de espaço de cores. Na imagem no formato YCbCr é efetuada a

operação de *down-sampling*. Como a informação de luminância (Y) é a mais importante para a percepção do olho humano do que a informação de crominância (Cb e Cr), é possível diminuir o tamanho da imagem diminuindo o número de *bits* utilizados para representar a crominância, mantendo o mesmo número para a luminância. Com isso, um determinado *bit* de crominância pode estar associado a vários *bits* de luminância. Esta operação é chamada de *down-sampling* e a operação inversa é chamada de *up-sampling*. Após a operação de *down-sampling* as matrizes Y, Cb e Cr são divididas em blocos de 8x8 *pixels* e em cada bloco é aplicada a DCT (transformação discreta do cosseno) para transformar a imagem do domínio espacial para o domínio das freqüências. Em cada bloco, agora formado de 64 coeficientes da DCT, é efetuada a quantização, eliminando assim, aquelas freqüências menos perceptíveis ao olho humano. A última operação antes de obter a imagem comprimida é a codificação de entropia, que codifica os coeficientes quantizados utilizando os algoritmos de RLE e Huffman. Uma característica interessante do padrão JPEG é que todos os coeficientes DC quantizados dos blocos são codificados em conjunto por Huffman, enquanto que os coeficientes AC de cada bloco passam por outra codificação de Huffman.

A descompressão de uma imagem JPEG, segundo Agostini (1999), deve efetuar as operações inversas das realizadas na compressão, com a direção oposta. Primeiramente os dados devem ser decodificados por Huffman e RLE, a seguir devem ser dequantizados, devem passar pela IDCT (inverva da transformação discreta do cosseno), por um processo de *up-sampling* (os componentes Cb e Cr passam a ter o mesmo número de *bits* do componente Y) e, finalmente, o espaço de cores deve ser transformado para o espaço utilizado pelo usuário (normalmente RGB).

Com o vazio deixado pela norma ISO 10918-1, surgiu o formato JFIF (JPEG *File Interchange Format*), que é um formato de imagem simples e de fácil implementação, permitindo representar imagens a cores reais (2^{24} cores). A aceitação do formato JFIF foi surpreendente, justificada pela expansão da *World Wide Web* e da necessidade por parte de seus usuários de transmitir imagens com mais cores e de tamanho reduzido, do que o máximo de 256 cores permitido pelo formato GIF. A popularidade deste formato, associada ao nome extenso demais (JPEG *File Interchange Format*), levaram aos seus utilizadores a mencioná-lo simplesmente por JPEG estabelecendo assim à confusão entre o nome do formato JFIF e nome da norma JPEG (Lopes, 2002).

2.1.1.3.1 CARACTERÍSTICAS DO FORMATO JFIF

Uma das principais características do formato JFIF, é que ele, ao contrário da maioria dos formatos existentes, não utiliza o modelo de cor RGB, mas o modelo YCbCr (luminância, crominância azul e crominância vermelha) com quantização da cor, reduzindo assim o conteúdo da imagem em até metade do seu tamanho original (com perda de informação). O formato JFIF utiliza a compressão JPEG que se baseia da utilização de algoritmos de compressão e descompressão do tipo da transformada discreta do *co-seno*, derivada da transformada de *Fourier*. Os algoritmos são aplicados a grupos de 8x8 *pixels* de um mesmo componente de cor, sendo cada componente de cor processado independentemente dos outros. Os coeficientes da transformada, designados por coeficientes AC e DC, depois de alguns tratamentos são armazenados em segmentos de informação dos arquivos JFIF (Lopes, 2002).

2.1.1.3.2 FORMATO DOS ARQUIVOS JFIF

Os arquivos em formato JFIF, conforme Lopes (2002), estão organizados em blocos separados entre si por marcas com o tamanho de um *byte*. Cada marca é precedida por um *byte* contendo o valor hexadecimal FF que assinala que se segue uma marca. A tabela 1 apresenta as marcas mais comuns presentes em arquivos JFIF. Todos os arquivos contem um segmento SOI no seu início, seguido obrigatoriamente por um segmento do tipo APP0. Pode ser incluído um número variável de segmentos do tipo APPn, após o segmento APP0, caso seja necessário. Em seguida é incluído um ou mais segmentos DQT contendo cada um deles uma tabela de quantização, um segmento SOF0 descrevendo os parâmetros principais da imagem e um ou mais segmentos DHT contendo as tabelas de Huffman utilizadas na compressão e necessárias para a descompressão. Após estes segmentos vem um segmento do tipo SOS que descreve os componentes comprimidos da imagem. O arquivo termina com um segmento EOI que determina o fim do mesmo. Todas as marcas anteriormente citadas, exceto as marcas SOI e EOI, são imediatamente seguidas por dois *bytes* cujo valor corresponde ao tamanho do respectivo segmento.

A estrutura de um arquivo JFIF está apresentada na fig. 4 e é completamente compatível com as definições do padrão JPEG (Agostini, 1999). Isto significa que qualquer aplicativo preparado para abrir arquivos JPEG, poderá abrir arquivos JFIF.

Figura 4 – Estrutura do arquivo JFIF

Fonte: (Agostini, 1999)

Tabela 1 – Marcas de segmento mais comuns em arquivos de formato JFIF.

Marca	**Valor Hexadecimal**	**Tipo de marca**
SOI	D8	Início de uma imagem
APP0	E0	Segmento identificador JFIF
APPn	En	Outros segmentos identificadores opcionais
DQT	DB	Identificador de tabela de Quantização
SOF0	C0	Início de um quadro (frame) da imagem
DHT	C4	Identificador de uma tabela de Huffman
SOS	DA	Início dos dados (conteúdo) de uma imagem
EOI	D9	Fim da imagem

Fonte: (Lopes, 2002)

2.2 CORES

Cor é um fenômeno de percepção, e não um componente objetivo ou característica de uma substância. Cor é um aspecto da visão e consiste numa resposta psicofísica da reação do olho e uma resposta interpretativa automática do cérebro, da característica do tamanho da onda da luz sobre um certo nível de brilho (Farina, 1990).

Em 1666, Isaac Newton[2] demonstrou a origem da cor passando um raio de luz solar através de um prisma de vidro produzindo um arco-íris de coloração, de um espectro visível. Este fenômeno foi observado antes, mas sempre eram relatadas cores latentes que existiam no vidro do prisma. Newton, todavia, adicionou um simples passo ao experimento. Ele transferiu seu arco-íris miniatura para um segundo prisma que reconstituiu o original raio de luz branca. Sua conclusão foi revolucionária: cores estão na luz, não no vidro, e a luz que as pessoas vêem tão brancas são uma mistura de todas as cores de espectro visível (Gonçalves, 1997).

O uso da cor em Computação Gráfica apresenta várias vantagens. Além de tornar as imagens nos terminais de vídeos mais bonitas e agradáveis, auxilia a visualização de conexões em desenhos complexos, melhorando a legibilidade da informação possibilitando gerar imagens realistas. Enfim, o uso de cores torna o processo de comunicação mais eficiente.

A cor, elemento fundamental em qualquer processo de comunicação, merece uma atenção especial. É um componente com grande influência no dia a dia de uma pessoa, interferindo nos sentidos, emoções e intelecto; podendo, portanto, ser usada deliberadamente para se atingir objetivos específicos. Um projetista de interface deve lançar mão desse poder das cores de modo a utilizá-las adequadamente a tornar as interfaces mais poderosas (Farina, 1990).

A cor exerce uma ação tríplice: a de impressionar, a de expressar e a de construir. A cor é vista: impressiona a retina; é sentida: provoca uma emoção; e é construtiva, pois tendo um significado próprio, possui valor de símbolo, podendo assim, construir uma linguagem que comunique uma idéia (Farina, 1990).

---

[2] Físico, matemático e astrônomo inglês do século XVII

2.2.1 IMAGENS MONOCROMÁTICAS E COLORIDAS

As imagens podem ser classificadas em dois grupos: as imagens monocromáticas e as coloridas. Em um sistema monocromático basta que se tenha um único plano de *bits* no *buffer* de refrescamento. A cada *bit* se associa um estado digital (0 ou 1) e em decorrência simula-se nesta memória o mapa de *bits* da imagem a ser mostrada no monitor. Já os sistemas coloridos utilizam-se de monitores coloridos com três canhões de cores, produzindo feixes decompostos em três cores básicas: vermelho, verde e azul, que agem sobre grupos de três pontos de fósforo em cada posição do *pixel*. De uma forma geral, os computadores são capazes de armazenar e exibir imagens nas quantidades de cores de acordo com o quadro 1.

Quadro 1 - Relação Cores x Monitores

Número Total de Cores	Nome Comum
2	Preto-e-Branco, Monocromático (ou 1 *bit*)
4	Padrão EGA (ou 4 *bits*)
256	Padrão VGA (ou 8 *bits*)
65.536	VGA Avançado, *high color* (ou 16 *bits*)
16.777.216	*True Color* (ou 24 *bits*)

Fonte: (Reis, 1998)

É freqüente o caso em que a quantidade e o conjunto de cores definidos no arquivo a ser exibido num dispositivo é diferente do que pode ser de fato mostrada na superfície do dispositivo. A aplicação deve então, transformar as cores definidas no arquivo nas que são aceitas no dispositivo de saída. Se a quantidade de cores no arquivo é bem menor que a disponível no dispositivo de saída normalmente a conversão é simples, pois a aplicação pode escolher entre as cores disponíveis no dispositivo as mais adequadas para serem exibidas. Mas um problema ocorre quando a quantidade de cores definidas no arquivo excede a quantidade que pode ser de fato exibida. Neste caso, deve ser feito um processo para reduzir a quantidade de cores no arquivo fonte a ser exibido através da associação entre as cores usadas no arquivo e as disponíveis no destino. Este processo é chamado de quantização e pode provocar mudanças inaceitáveis no aspecto da imagem, como por exemplo, a introdução de cores que não existiam na imagem original (Murray, 1994).

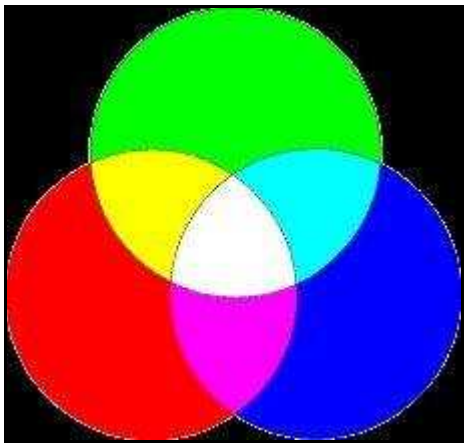
Independente de qual sistema utilizado, sempre se tem o zero indicando ausência de luz (preto) e o valor máximo é sempre associado à máxima intensidade de cor (branco – formado pela mistura de todas as cores) (Reis, 1998).

2.2.2 MISTURA ADITIVA DE CORES

Este processo é utilizado nos monitores de vídeo e televisões, através do qual, a cor é gerada através da mistura de vários comprimentos de onda da luz provocando uma alteração no comprimento de onda que atinge e sensibiliza o olho.

As cores primárias aditivas são o vermelho, o verde e o azul. No processo aditivo, o preto é gerado pela ausência de qualquer cor, indicando que nenhuma luz está sendo transmitida; e o branco, que é a mistura de todas as cores, indica que uma quantidade máxima de vermelho, verde e azul está sendo transmitida (fig. 5).

Figura 5 – Processo aditivo de formação de cores



Fonte: (Reis, 1998)

2.2.3 O MODELO RGB

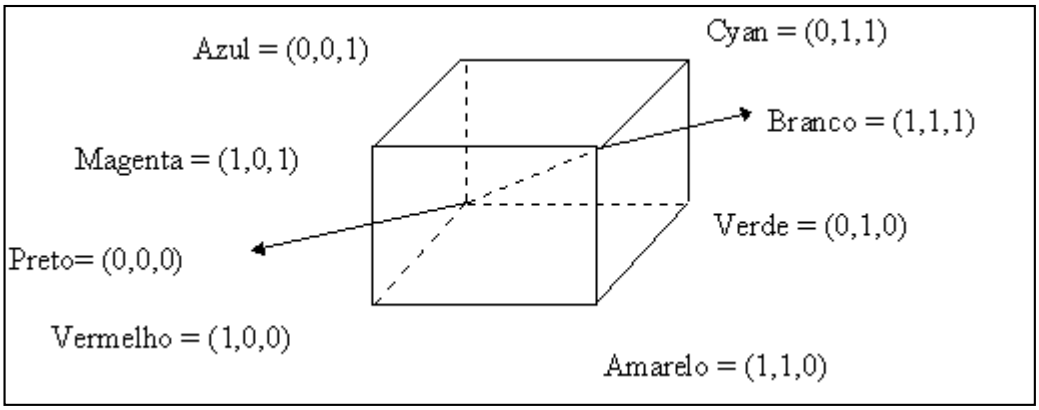
Segundo Hearn (1994), um modelo de cores é um método que explica as propriedades ou o comportamento das cores num contexto particular. Não existe um modelo que explique todos os aspectos relacionados à cor. Por isso, são utilizados modelos diferentes para ajudar a descrever as diferentes características da cor que são percebidas pelo ser humano. Existem vários modelos de cores, sendo que será apresentado apenas o RGB.

A denominação RGB vem do acrônico *Red* (vermelho), *Green* (Verde) e *Blue* (azul) usado em monitores coloridos que usam três sinais de entrada separados para controlar os feixes de imagens nas cores vermelha, verde e azul (Reis, 1998).

As cores primárias do modelo RGB são as 3 cores que o sistema utiliza para produzir outras cores. As cores podem ser produzidas a partir de uma combinação das primárias, ou então, da composição de 2 combinações. O universo de cores que podem ser reproduzidas pelo modelo RGB é chamado de espaço de cores (*color space* ou *color gamut*). Alternativamente, um espaço de cores pode ser definido como uma representação visual de um modelo de cores, como o cubo definido pelas componentes do modelo RGB. Esse modelo se baseia na sensibilidade do olho e usa um sistema de coordenadas cartesianas R, G, B, cujo subespaço de interesse é o cubo unitário mostrado na fig. 6.

Não existe um conjunto finito de cores primárias que produza todas as cores visíveis, mas sabe-se que uma grande parte delas pode ser produzida a partir das 3 primárias.

Figura 6 – Cubo RGB para terminais RGB



Fonte: (Ferreira, 1999)

As cores primárias RGB são aditivas, o que determina que as cores são produzidas através da adição das intensidades dessas cores. A diagonal principal do cubo, representada pela diagonal pontilhada, possui quantidades iguais de cores primárias e representa a escala de cinza. Cada ponto colorido, dentro dos limites do cubo, pode ser representado por (R, G, B), onde variam entre zero e um valor máximo.

3 ESTEGANOGRAFIA

3.1 HISTÓRIA

Ao contrário do que muitos pensam, o termo esteganografia é um conceito muito antigo, embora ainda pouco difundido em vários meios. É o que descreve Sellars (1999), sobre os primeiros registros encontrados sobre esteganografia na história. Estes relatos foram registrados pelo historiador grego Herodotus, durante o século V da Grécia antiga. O primeiro registro descreve o episódio da prisão do tirano grego Histiaeus em Susa, prisão esta decretada pelo rei Darius. Histiaeus precisava mandar uma mensagem de forma secreta para seu genro Aristagoras que se estava na cidade de Miletus. Ao ser visitado por um de seus escravos, Histiaeus raspou sua cabeça e tatuou uma mensagem no escalpo dele. Quando os cabelos do escravo tinham crescido e ocultado a mensagem, ele foi mandado para a cidade de Miletus, transmitindo assim a mensagem.

Outro registro de Herodotus, conforme Davis (2002), descreve a utilização das tábuas recobertas por cera, que na época serviam como bloco de notas, para escrever mensagens ocultas. Conforme conta a história, o grego Demeratus precisando notificar Sparta que Xermes tinha a intenção de invadir a Grécia, raspou a cera de uma das tábuas e escreveu a mensagem diretamente na própria madeira. Para que a mensagem ficasse encoberta, a tábua foi recoberta com cera, tornando a informação imperceptível sob qualquer inspeção.

Com o passar do tempo, muitas foram às maneiras na qual a esteganografia foi utilizada. O uso de tintas invisíveis, segundo Holmes (2002), era uma das formas mais simples que uma pessoa poderia utilizar para enviar uma mensagem sem que alguém percebesse. Esta era uma das formas utilizadas pelos antigos Romanos que escreviam entre linhas usando tintas invisíveis baseadas em substâncias como sucos de limão, urina e leite. Sobre o efeito do calor, as tintas invisíveis se tornavam legíveis.

Segundo Sellars (1999), o monge alemão Johannes Trithemius (1462-1526) foi um dos primeiros pesquisadores sobre esteganografia e criptografia. Seu primeiro trabalho, de nome Steganographia, relacionado a esteganografia, continha a descrição de sistemas de magia e profecias e também de um complexo sistema de criptografia. Sua publicação foi feita de

maneira póstuma, pois Trithemius temia pela reação das autoridades se o mesmo fosse oficialmente publicado.

Das pesquisas e fundamentações de Trithemius, surgiu o primeiro livro sobre esteganografia. Esta obra de quatrocentas páginas foi escrita por Gaspari Schotti no ano de 1665 e foi intitulada com o nome de Steganographica. Seu desenvolvimento teve continuidade durante os séculos XV e XVI. Nesse período, muitas obras tiveram sua existência ocultada pelos seus autores, justamente pelo medo da forte repressão imposta pelas facções da época. Para solucionar esse problema, o Bispo John Wilkins escreveu um tratado. Wilkins, além de tornar-se posteriormente mestre da Faculdade de Trinity, em Cambridge, destacou-se pela publicação de várias obras, dentre elas a invenção de vários esquemas para a codificação de mensagens em músicas e nós de fio para tintas invisíveis, descreveu os princípios de *cryptanalysis* por carta freqüências, e defendeu a esteganografia contra os que se opuseram a publicações no ramo (Sellars, 1999).

Em 1883, Auguste Kerckhoffs escreveu a obra *Cryptographie militaire*. Suas fundamentações eram principalmente sobre criptografia, mas apesar disso, Auguste descreveu importantes princípios que se tornam interessantes ao se projetar um sistema de esteganografia. Já em 1907, Charle Briquet, escreveu *Les Filigranes*, obra cujo conteúdo se tratava de um dicionário histórico sobre *watermarks*[3] (Sellars, 1999).

A evolução da esteganografia se tornou evidente durante o século XX. Já no início do século, durante a Guerra Bôer na África do Sul, o governo Britânico, segundo Sellars (1999), usava o Lord Robert Baden-Powell, para desenvolver a função de explorador. Sua missão era mapear as posições que se encontravam as bases de artilharia bôeres. Para dar garantia de que seu trabalho não fosse descoberto, seus mapas eram desenhados em forma de borboletas, sendo que as localizações das bases inimigas estavam estampadas em suas asas. Se por acaso seus desenhos fossem descobertos pelos bôeres, dificilmente decifrariam a principal informação, que seriam os mapas.

[3] *Watermark* é o processo de colocar uma marca “registrada” oculta nos produtos, permitindo manter controle sobre a pirataria.

A Segunda Guerra Mundial marcou uma fase importante de desenvolvimento em esteganografia, devido ao intenso período de experimentos de técnicas esteganográficas. A princípio a tecnologia mais utilizada foi a das tintas invisíveis, sendo em seguida utilizada as cifras nulas para esconder mensagens secretas. Essas cifras, de acordo com Davis (2002), nada mais eram do que simples mensagens de aparência inocente. O segredo para traduzir a mensagem era juntar determinadas letras formando então as palavras que compunham a mensagem secreta. Abaixo se tem como exemplo, citado por Sellars (1999), uma mensagem enviada por uma espiã alemã na Segunda Guerra:

"*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.*"

Levando em consideração apenas a segunda letra de cada palavra, tem-se a formação da seguinte mensagem:

"*Pershing sails from NY June 1.*"

Sellars (1999) cita que, ainda durante a Segunda Guerra Mundial, novas tecnologias foram desenvolvidas, aumentando assim a capacidade de transmissão de informações de forma secreta, dificultando ainda mais a descoberta das mensagens. Os nazistas introduziram um novo conceito em espionagem, recebendo o nome de microponto, da qual foi intitulada pelo diretor do FBI J. Edgar Hoover como "A obra prima da espionagem inimiga". Microponto trata-se de microfotografias do tamanho aproximado de um ponto, contendo informações de tamanho suficiente para preencher uma página datilografada, proporcionando a transmissão de dados em grande quantidade, bem como a transmissão de desenhos e fotografias.

O acontecimento mais marcante nos tempos atuais quanto ao uso de esteganografia segundo Sieberg (2001), acredita-se ter relação com o atentado terrorista no *World Trade Center*, ocorrido em 11 de setembro de 2001 nos EUA. Autoridades da segurança americana rastrearam inúmeras fotografias enviadas supostamente por membros da Al Qaeda, responsável pelo atentado, na tentativa de encontrar mensagens escondidas através do uso de técnicas esteganográficas. Os americanos acreditavam que o líder da Al Qaeda, Osama Bin Laden, enviava através de fotografias pela internet mensagens a seus seguidores no mundo,

relativas ao atentado. Quanto a esta busca, nada foi devidamente comprovado e divulgado até o momento.

Com o constante crescimento tecnológico dos computadores, a esteganografia sofreu um importante impulso. Métodos velhos, como o de esconder mensagens em imagens, foram implementados na forma de software, utilizando-se dos recursos disponibilizados pelas novas tecnologias do mundo eletrônico, levando a esteganografia para um nível totalmente diferente (Sellars, 1999). Nos próximos anos, novas técnicas e métodos de esteganografia surgirão, e com o poder da maioria dos computadores da atualidade, bem como o crescimento em popularidade da esteganografia, um avanço inédito é esperado.

3.2 DEFINIÇÃO

Até pouco tempo atrás, a esteganografia era considerada como o “primo pobre” da criptografia. Mas devido a crescente demanda da indústria por proteção dos direitos autorais, através de marcas d’água e impressões digitais, em áudio e vídeo, está se tornando cada vez mais popular (Sellars, 1999).

O termo esteganografia conforme já mencionado anteriormente vem da palavra grega s*teganos* que significa escondido, mais a palavra *grafia* que significa escrita, ou seja, escrita escondida ou escrita oculta. Atualmente é uma forma muito utilizada de esconder informações. Apesar da esteganografia ser na maioria das vezes confundida ou relacionada à criptografia, não dá para dizer que significa a mesma coisa, pois ambas possuem objetivos distintos. Enquanto o objetivo da esteganografia é de ocultar a existência de uma mensagem, o da criptografia é apenas embaralhar uma mensagem fazendo com que a mesma fique de forma incompreensível.

Uma definição mais precisa citada por Kuhn (1995) diz que: “*o objetivo da esteganografia é ocultar mensagens dentro de mensagens sem importância, de uma maneira que não permita nenhum inimigo detectar que existe uma segunda mensagem secreta presente*”.

Outra definição sugerida por Davis (2002), define que esteganografia é o processo que tem por objetivo esconder um arquivo dentro de outro, de tal maneira que impossibilite que

outras pessoas identifiquem o conteúdo do objeto escondido, bem como de reconhecer sua existência.

Já Moniz (2002), com uma definição mais detalhada diz que a esteganografia é uma técnica que permite esconder (codificar) uma informação dentro de uma mídia na qual ela será embutida. Dependendo do tipo de mídia (meios) utilizada, tem-se uma técnica relacionada através de um software apropriado, tornando possível alterar a forma em que os *bytes* correspondentes a uma mídia estão gravados de forma a incluir uma informação, aproveitando *bits* normalmente não utilizados ou de menor importância.

Conforme Sellars (1999), muitas são as técnicas e métodos de esteganografia para a ocultação de mensagens nas mais variadas mídias. Conforme mencionado anteriormente, alguns destes métodos são: tintas invisíveis, micropontos, assinaturas digitais, cobertura de canais e espectros de comunicação. Hoje, graças aos avanços tecnológicos, a esteganografia é utilizada em textos, imagens, sons, sinais e outros.

A principal vantagem da esteganografia para Sellars (1999) é a transmissão de informações de forma secreta, sem o conhecimento de sua existência. Um exemplo disto seria o envio de uma foto de um simples cão contendo informações de uma determinada organização referentes as suas últimas inovações tecnológicas.

Entretanto, Sellars (1999) diz que a esteganografia também possui algumas desvantagens. Ao contrário da criptografia, que requer pouco processamento computacional, a esteganografia exige um grande processamento e como resultado tem-se a ocultação de alguns poucos *bits* de informação. Outra desvantagem seria a descoberta do sistema que faz o processamento esteganográfico. Caso isso aconteça, ele ficará sem utilidade. Para contornar esse problema, é adotada uma senha nos processos de inserção e extração da mensagem, garantindo uma maior segurança e sigilo das informações. Muitas vezes, se houver necessidade, tem-se ainda a opção de criptografar a mensagem antes de inseri-la na mensagem envelope[4]. Deste modo, se a existência da mensagem oculta for detectada, torna-se difícil obter uma tradução da mesma em virtude da criptografia utilizada.

[4] Mensagem envelope é um termo designado no *Information Hiding Workshop* em Cambridge, Inglaterra, onde faz referência a mídia com dados já codificados.

3.3 ÁREAS DE APLICAÇÃO

Em se tratando de imagens *raster*, muitas são as áreas de aplicações para a esteganografia, dentre elas, a proteção de direitos autorais, classificação e comunicações secretas (Lin, 2002).

3.3.1 PROTEÇÃO DE DIREITOS AUTORAIS

Segundo Lin (2002), a proteção de direitos autorais em esteganografia refere-se ao processo de inserir uma informação ou assinatura com marca d'água (*watermarking*), em uma imagem com a finalidade de identificá-la como uma propriedade intelectual. Este é o contexto da marca d'água, inserir uma marca registrada oculta em imagens, vídeos, músicas e programas. A estrutura de uma marca d'água pode apresentar uma relativa complexidade. Além disso, no ato da venda ou distribuição de uma imagem, vídeo, música ou programas, ela facilita a identificação do comprador, bem como a autenticidade do produto. Outro benefício proporcionado pela marca d'água é que ela pode detectar se uma imagem foi modificada ou não. O processo para detectar uma marca d'água pode ser realizado através de estatística, comparação ou testes similares, ou também medindo a quantidade de outras características da marca d'água inserida em uma imagem esteganografada. Os processos de inserir e analisar as marcas d'água com o intuito de proteger os direitos autorais fez ressurgir o interesse pela esteganografia digital.

A importância da marca d'água ficou clara quando do lançamento do DVD (*Digital Versatile Disk*). A distribuição de filmes em DVD só foi concretizada com a aprovação por parte dos grandes estúdios de Hollywood de um esquema de proteção a ser utilizado. Nesse esquema, foram considerados e tratados, conforme Franco (1998), três mecanismos distintos e complementares:

a) o conteúdo analógico é criptografado logo após ser digitalizado, evitando assim a reprodução de uma copia digital num DVD *player* "normal";
b) o sinal NTSC gerado num DVD, sofre uma modificação, evitando uma gravação de boa qualidade num videocassete, mas sem afetar a qualidade de reprodução num aparelho de TV;

c) uma marca d'água digital é inserida no conteúdo analógico original; se o sinal NTSC for redigitalizado, a reprodução de seu conteúdo não poderá ser visualizada num DVD *player* "normal", que detectará a marca d'água (devido a capacidade da tecnologia usada na marca d'água, a mesma consegue suportar a conversões A/D e D/A, bem como a compressão MPEG-2).

3.3.2 CLASSIFICAÇÃO

Refere-se a títulos, anotações, datas e outras informações que podem ser inseridos em uma imagem ou outra mídia. O ato de copiar, por exemplo, uma imagem esteganografada também copia da mesma forma todas as descrições inseridas, permitindo o acesso a essas descrições a somente quem possuir a senha para poder extraí-las. Numa base de dados de imagens, por exemplo, palavras chave podem ser inseridas na imagem, facilitando uma eventual pesquisa através de mecanismos de busca. Se, por exemplo, uma imagem for um quadro pertencente a uma seqüência de vídeo, podem ser inseridos na imagem marcadores de tempo para que seja possível a sincronização com o áudio (Lin, 2002).

Outras duas aplicações interessantes da esteganografia citadas por Franco (1998) são o monitoramento automático da inserção de comerciais de rádio e a inclusão confidencial de dados de pacientes em imagens médicas, permitindo a recuperação dessas informações caso seja necessário.

3.3.3 COMUNICAÇÕES SECRETAS

Em determinadas situações, a transmissão de uma mensagem criptografada pode atrair atenção indesejada, ou até mesmo, o uso dessas tecnologias pode ser restrito ou proibido por lei. Isso não acontece ao se usar esteganografia. Essa tecnologia não chama a atenção no envio da mensagem nem sobre o remetente, a própria mensagem ou o destinatário. Um exemplo disso seria o envio de um segredo sem alertar possíveis intrusos ou curiosos (Lin, 2002).

3.4 TIPOS DE TÉCNICAS

Segundo Sellars (1999), existem vários tipos de técnicas de esteganografia, cada uma relacionada de acordo com as características de cada tipo de mídia utilizada. Nas seguintes

seções, será mostrado como a esteganografia pode e está sendo utilizada através de textos, imagens e áudio.

3.4.1 ESTEGANOGRAFIA EM TEXTO

O uso da esteganografia em texto, associada à popularidade da Internet, vem enfatizando cada vez mais aplicações nessa mídia. Essas aplicações diversificam-se desde o envio de mensagens embutidas no conteúdo do documento em texto, até controles para fins de autenticidade do mesmo. Num mundo globalizado e conectado a maior rede mundial de computadores (Internet), várias são as atividades transmitidas via rede, sendo muitas delas utilizadas de forma ilícita. A distribuição ilegal de documentos via *e-mail*, por exemplo, foi um dos problemas identificados por Brassil (1994) e outros. A conseqüência disto é a distribuição de cópias de documentos, sem o devido pagamento dos direitos legais ao autor do documento original. Visando obter um controle rigoroso para este tipo de pirataria foi discutido um método entre vários autores, onde através do uso da esteganografia em texto, seria embutida uma palavra-código para marcar um documento impresso, identificando o destinatário do documento quando o mesmo fosse recuperado. Em conjunto ao método proposto, métodos de criptografia também podem ser aplicados, aumentando a segurança.

A idéia principal dessas técnicas de marcação discutidas por Brassil (1994) consiste em inserir uma palavra-código (um número binário, por exemplo) no documento através alteração de características particulares do mesmo. Cada *bit* que compõe a palavra chave é aplicado a uma característica particular de um documento, possibilitando assim a codificação da palavra-código. O que determina qual método de codificação será utilizado é justamente o tipo de característica particular do documento a ser codificada a mensagem. São três as características identificadas por Brassil (1994).

3.4.1.1 CODIFICAÇÃO *LINE-SHIFT*

Segundo Sellars (1999), este método utiliza-se do deslocamento das linhas de texto no sentido vertical para codificar unicamente o documento. Os processos de codificação e decodificação podem geralmente ser aplicados ao arquivo do documento, bem como a uma imagem da página.

A codificação *line-shift* apresenta um bom funcionamento, com ótima recuperação dos dados, mesmo depois da décima fotocópia. Entretanto, é o método que apresenta maior visibilidade para quem lê o documento esteganografado. Ou seja, perceber-se com mais facilidade de que o documento tem alguma alteração. Outro fator negativo, é que esse método pode ser burlado pela medida manual ou automática do número de *pixels* entre as linhas das bases do texto (Sellars, 1999).

De qualquer maneira, quando um documento é marcado pela codificação *line-shift*, torna-se muito difícil o processo de decodificação do documento impresso. Para que a decodificação aconteça, cada página precisará ser reescaneada, alterada e reimpressa. Mas, mesmo assim, a decodificação pode ser comprometida, já que o documento pode sofrer efeitos prejudiciais como manchas ou borrões (Sellars, 1999).

3.4.1.2 CODIFICAÇÃO *WORD-SHIFT*

Na codificação *word-shift*, conforme descreve Sellars (1999), a mensagem é inserida no documento através do deslocamento das posições horizontais das palavras dentro das linhas de texto, mantendo uma aparência de um espaçamento natural. Assim como a codificação *line-shift*, a codificação *word-shift* também pode ser aplicada ao arquivo do documento bem como à imagem do texto. É importante ressaltar que este método somente pode ser aplicado em documentos com espaçamento variável entre as palavras adjacentes. É interessante também conhecer o espaçamento entre as palavras do documento original, o que se sugere para isso ter uma cópia do documento original.

A codificação de uma mensagem pelo método *word-shift* funciona da seguinte maneira. Para cada linha de texto do documento, os maiores e menores espaços entre as palavras são encontrados. Em seguida, para que uma linha seja codificada, o maior espaçamento é reduzido para uma certa quantidade, e o menor é extendido na mesma quantidade. Desta forma, o tamanho da linha é preservado, tendo alterações menos perceptíveis ao texto. Por esse motivo, para Sellars (1999), a codificação *word-shift* apresenta menos alteração e percepção a um observador do que a codificação *line-shift*, já que o espaçamento entre as palavras adjacentes em linhas é geralmente deslocado para permitir a justificação do texto. Embora a codificação *word-shift* é menos perceptível, ela também pode ser detectada e alterada.

Para que a codificação *word-shift* seja detectada, basta que alguém tenha conhecimento do algoritmo utilizado pelo formatador utilizado para justificação do texto. Através dele, é possível calcular o espaço atual entre as palavras e compará-lo ao espaçamento do documento original. Se houver diferença em espaços entre os dois documentos, é porque existem dados ocultos no documento. Outra maneira de identificação do método é para pegar dois ou mais documentos distintamente codificados e não corrompidos e executar operações de diferenciamento de *pixels* página por página das imagens. Dessa forma, poder-se-ia então rapidamente perceber o deslocamento das palavras e o tamanho do espaçamento entre elas (Sellars, 1999).

Quanto a uma possível alteração, a codificação poderia ser comprometida de duas maneiras. Através do reespaçamento das palavras codificadas para o espaçamento original produzido pelo formatador, ou simplesmente através da aplicação da rotação horizontal randômica em todas as palavras do documento que não foram encontradas nas bordas do mesmo.

3.4.1.3 CODIFICAÇÃO DE CARACTERÍSTICAS

O método que tem como nome codificação de características, conforme Sellars (1999) descreve, foi sugerido por Brassil (1994) e pode ser aplicada a uma imagem *bitmap* do documento ou a um arquivo do documento. As características do texto podem ser alteradas ou não, e o que determina essas alterações é a palavra código a ser inserida.

Basicamente, o funcionamento da codificação de características acontece da seguinte maneira. Para codificar *bits* em texto, pode-se estender ou encurtar os finais das linhas ascendentes verticais das letras, como por exemplo, b, d, m, n, etc. Durante o processo de codificação tem-se a preocupação de limitar as chances de uma futura decodificação visual, o que pode ser feito através de uma randomização da característica, que acontece antes da codificação. Esta randomização nada mais é do que alongar ou encurtar os comprimentos do final de cada linha, para em seguida serem alterados e só então a partir daí receberem a codificação dos dados. Assim os comprimentos originais do final de cada linha com dados codificados não seriam conhecidos, dificultando qualquer decodificação visual. Para que a decodificação seja bem sucedida é necessário que se tenha o documento original, para fins

comparativos da mudança nos *pixels* de uma característica, ou então uma especificação dessas mudanças (Sellars, 1999).

Uma característica importante desse método é a capacidade de suportar uma grande quantidade de dados a serem codificados, determinados pelo número elevado das características nos documentos originais que podem sofrer modificações. Assim como os outros dois métodos sugeridos por Brassil (1994), a codificação de característica normalmente é imperceptível ao leitor. Da mesma maneira, esse método pode ser detectado e alterado. A codificação da característica pode ser detectada com um ajustamento de cada comprimento de fim de linha para um determinado valor fixo, sendo um trabalho pesado e de muita atenção. Existem algumas maneiras para dificultar uma possível descoberta de codificação de dados com este método. Uma delas seria variando a característica particular a ser codificada, ou utilizar em conjunto o método *Word-shift*, por exemplo. Com certeza isto traria mais dificuldades para quem tentasse de alguma maneira detectar e alterar os dados codificados.

3.4.2 ESTEGANOGRAFIA EM IMAGENS

A esteganografia em imagens surgiu da evolução tecnológica dos computadores, sendo diretamente influenciada pelo crescente desenvolvimento do processamento de imagens digitalizadas, bem como a grande quantidade de imagens distribuídas na Internet, tornando-a assim, a mídia mais popular e utilizada na atualidade.

O uso da esteganografia em imagem aproveita-se da limitação do sistema visual humano (HVS) que é incapaz de perceber pequenas mudanças em padrões de cor e devido a esta fraqueza, arquivos de texto ou gráficos podem ser inseridos dentro de imagens sem serem detectados. Segundo Sellars (1999), qualquer texto, mensagem cifrada, ou imagens que pode ser inserido em um conjunto de *bits* pode ser escondido em uma imagem. Os dados podem ser escondidos de várias maneiras, dependendo da técnica utilizada e suas características. Normalmente cada *bit* de informação do dado é codificado e inserido nos setores (áreas “ruidosas”) da imagem onde não irão comprometer a aparência da mesma, não revelando a existência da informação. Muitas vezes, se for necessário, tem-se a possibilidade de inserir a informação de maneira dispersa por toda a imagem, o que esconderia de maneira mais eficaz os dados codificados.

Conforme comentado, existem várias maneiras para ocultar as informações em uma imagem, mas as mais conhecidas e utilizadas, citadas por Sellars (1999) são:

a) inserção nos *bits* menos significativos (LSB);
b) técnicas de máscara e filtragem;
c) algoritmos e transformações.

É importante ressaltar que todas essas técnicas para inserção de dados em imagens proporcionam diferentes graus de variação na formação da mesma, como por exemplo, diminuição da resolução, ou diminuição na profundidade das cores. Mas conforme citado anteriormente, essas mudanças, dependendo do grau de intensidade, passam despercebidas aos olhos humanos, tornando-as técnicas eficazes, evidenciando assim sua popularidade.

3.4.2.1 INSERÇÃO NOS *BITS* MENOS SIGNIFICATIVOS (LSB)

Uma das técnicas mais comuns de implementação de esteganografia em imagens é a inserção nos *bits* menos significativos, no qual o *bit* menos significativo de cada *byte* é alterado para formar o fluxo de *bits* que representam os dados embutidos. É sem dúvida a mais popular, mas segundo Sellars (1999), é uma técnica frágil quando se trata de manipulação da imagem. Um simples processo de conversão de uma imagem de formato BMP, por exemplo, para qualquer formato que utilize a compressão *lossy*[5] (como o utilizado na norma JPEG), pode comprometer ou até destruir toda a informação inserida na imagem.

Normalmente, as imagens utilizadas para inserção de dados possuem 8 *bits*/*pixel* ou 24 *bits*/*pixel*. No caso de uma imagem formada por 24-*bits/pixel*, tem-se a possibilidade de codificar três *bits* em cada *pixel* (cada *pixel* é representado por três *bytes*). Como a inserção é feita no primeiro *bit* de cada *byte*, que é o que tem menos influência na formação da cor, qualquer mudança que aconteça nos valores dos *bits* do *pixel* não será notada pelo olho humano. A letra A pode ser escondida em três *pixels*, conforme exemplo citado por Sellars (1999). Os três *pixels* originais da imagem estão representados pelos três *bytes* de 24-*bits/pixel* na fig. 7.

---

[5] Compressão *lossy* é um algoritmo de compressão de dados com perda, onde abandonam a precisão *bit* a *bit* para obter taxas maiores de compressão.

A codificação dos dados a serem inseridos, segundo Davis (2002) pode ser entendida da seguinte maneira. O primeiro passo da codificação é converter o texto ASCII do referido dado em seu equivalente em binário. Assim, cada caractere da mensagem é convertido em seu número ordinal. Este é então convertido em binário através do uso de um método denominado por Davis de rotina de divisão-quociente. Esse método consiste em dividir o número ordinal em dois através da função *mod*. Esta função retorna ou 1 ou 0, sendo os valores colocados em uma matriz. O processo só termina quando o dividendo for 0. Os números 1 e 0 na matriz são o equivalente binário do número ordinal do ASCII. Após todos os caracteres serem convertidos desta maneira, o dado binário é embutido na imagem na seqüência alterando o LSB do dado da imagem caso houver necessidade.

Figura 7 – Representação binária dos componentes de cor

(00100111 11101001 11001000) red green blue
(00100111 11001000 11101001) red green blue
(00100110 11001001 11101001) red green blue

O valor binário para a letra A é (10000011). Cada *bit* que compõe a letra A será introduzido no LSB de cada componente de cor dos três *pixels*, tendo início no *byte* esquerdo superior, resultando nos dados da fig. 8 abaixo.

Figura 8 – Resultado da inserção da letra A nos *pixels*

(00100111 1110100**0** 11001000) red green blue
(0010011**0** 11001000 1110100**0**) red green blue
(00100110 11001001 11101001) red green blue

Os *bits* em destaque foram os únicos que sofreram uma mudança de valor, mas não influenciando na mudança de cor do *pixel*. Cada *bit* menos significativo do *byte*, ou seja, o primeiro de cada componente de cor RGB, recebeu o valor correspondente de cada *bit* que compõem a letra A.

Em se tratando do uso de técnicas de LSB em imagens formadas por 8-*bits/pixel*, Sellars (1999) alerta que é extremamente importante uma atenção especial neste caso, pois imagens formadas por 8-*bits* não correspondem tão bem às mudanças dos dados como nos formatos de 24-*bits*. Em uma imagem de 8-*bits*, como GIF ou BMP, cada *pixel* é descrito com um número de 0 a 25, que serve como índice para uma cor específica na tabela de cores. Qualquer mudança no valor dos *bits* de LSB, os ponteiros (índices) são alterados e em conseqüência as cores também. A mudança de cor no *pixel* pode ser mínima ou até total, o que não é aceito nesse tipo de aplicação. Para amenizar esse tipo de problema, algumas recomendações são sugeridas e citadas por Sellars (1999). A primeira dela é tomar cuidado na escolha da imagem que será esteganografada, verificando através de testes o comportamento e aparência da imagem em relação a possíveis mudanças de cores, evitando assim que sejam futuramente percebidas. Outra maneira de contornar o problema seria evitar a escolha de imagens geralmente conhecidas, como pinturas famosas, por exemplo. Distribuir os dados a serem codificados de maneira dispersa por toda a imagem também é uma maneira eficaz de diminuir a percepção a mudanças de cores. Por fim, uma maneira mais fácil seria utilizar imagens que utilizam as paletas *grey-scale* (tons de cinza), onde as diferenças entre máscaras não são facilmente diferenciadas, ou imagens que consistem na maior parte em uma cor, tais como a paleta de Renoir. A fig. 9 demonstra a inserção de dados em uma imagem de 8-*bits*.

Figura 9 - Representação binária de uma imagem 8-*bits/pixel*

(00	01	10	11)
white	red	green	blue

Inserindo, por exemplo, na imagem o valor binário 0011, tem-se o seguinte resultado demonstrado na fig. 10 abaixo.

Figura 10 - Resultado da inserção do valor binário nos *pixels*

(00	0**0**	1**1**	11)
white	white	blue	blue

3.4.2.2 MÁSCARA E FILTROS

Segundo Bassia (1998), essa técnica consiste em esconder a informação através da marcação de uma imagem, de modo similar ao funcionamento das marcas d'água aplicadas em papel. Uma das vantagens dessa técnica é que pode ser aplicada em imagens sem sofrer alterações no caso da imagem passar pela compressão *lossy*, devido ao fato da marca d'água ser integrada na imagem. Outra vantagem é que as técnicas de máscara e filtragem também passam despercebidas pelo sistema visual humano que não consegue distinguir algumas mudanças na imagem.

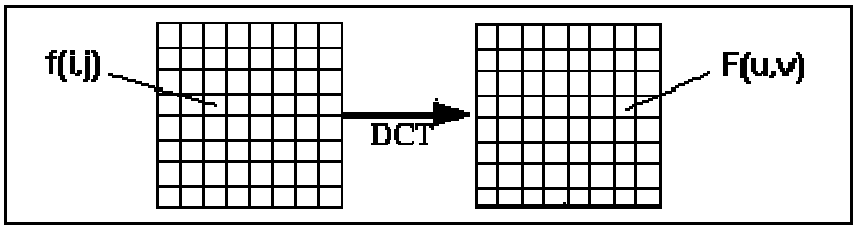
Sellars (1999) afirma que tecnicamente, as marcas d'água não são consideradas uma forma de esteganografia. Para ele, enquanto a esteganografia esconde uma informação na imagem, as marcas d'água atribuem uma informação à mesma, sendo considerada como um atributo. Mas outros autores, como Holmes (2002), defendem a idéia de que a marca d'água pode ser implementada como forma de esteganografia, sendo a maior parte de sua tecnologia desenvolvida para uso como proteção de *copyright* e ferramenta de licença. Outra afirmação de Holmes (2002), é que as marcas d'água podem ser implementadas de maneira oculta, o que é uma característica predominante da esteganografia.

Segundo Sellars (1999), as técnicas de máscara são mais adequadas a aplicações em imagens de formato que utilize o padrão JPEG, que utiliza compressão *lossy*, diferentemente da técnica de inserção dos *bits* menos significativos. A justificativa para isto é uma maior resistência ao processamento sofrido pela imagem, como por exemplo, compressão e cortes.

3.4.2.3 ALGORITMOS E TRANSFORMAÇÕES

Outro método de esteganografia em imagens citado por Sellars (1999) é ocultar um dado utilizando-se de funções matemáticas que são utilizadas nas compressões de algoritmos, representadas nas fig.11 e fig.12. São as duas funções: Transformação Discreta de Co-seno (DCT) e Transformação de Ondulação.

Figura 11 - Diagrama da função DCT



Fonte: (Li, 2000)

Figura 12 - Fórmula da função DCT

$$F(u,v) = \frac{\Lambda(u)\Lambda(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)\cdot u\pi}{16}\cdot\cos\frac{(2j+1)\cdot v\pi}{16}\cdot f(i,j)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & \end{cases}$$

Fonte: (Li, 2000)

As imagens com padrão JPEG utilizam a função da transformada discreta do cosseno (DCT) para comprimir seus dados. A DCT é uma transformação da compressão *lossy*, onde o arredondamento de erros á aplicado sobre valores do cosseno que não podem ser calculados de forma precisa. Existem algumas variações entre os dados originais e os dados recuperados e isto é determinado através dos valores e os métodos usados no cálculo da DCT. O objetivo da DCT e das funções de ondulação é transformar os dados pertencentes a um domínio em outro. Em seguida estes dados são transformados de um domínio espacial para um domínio de freqüência pela função DCT. Os dados de uma mensagem a ser inserida são codificados efetivamente nos coeficientes menos significativos.

As imagens podem também ser processadas, segundo Sellars (1999), através do uso da transformação rápida de *Fourier*, da transformação de pequenas ondas, ou também se utilizando outras propriedades, como por exemplo, a luminosidade. Sellars (1999) ainda cita que programas de esteganografia cujos sistemas são mais modernos utilizam as chamadas comunicações *spread-spectrum*[6] , que possibilitam transmitir um sinal *narrowband* (banda estreita) numa largura de faixa muito maior, fazendo com que a densidade *spectral* do sinal no

[6] Spread-spectrum é a utilização de ondas portadoras similares ao ruído e com a largura de banda muito maior do que a requerida para uma simples comunicação ponto a ponto com a mesma taxa de dados.

canal fique semelhante a um ruído. Esses programas de esteganografia utilizam dois tipos de técnicas de comunicações *spread-spectrum*. A primeira, de nome *hopping*, consegue esconder uma determinada informação modulando as fases do sinal dos dados (portador), utilizando-se para isso de uma seqüência de números pseudo-randômicos conhecidos pelo remetente e destinatário da mensagem, possibilitando a decodificação da informação. A segunda técnica utilizada, denominada seqüência-direta, divide a largura de faixa que se encontra disponível nos canais múltiplos e nos *hops* entre estes canais, utilizando da mesma forma uma seqüência de número pseudo-randômico.

Conforme Bender (1996), existe um método denominado *patchwork*, e é baseado em processos pseudo-randômicos estatísticos, imperceptíveis aos olhos humanos quanto à variação da luminosidade. O método *patchwork* utiliza o teste padrão redundante para codificar a informação a ser escondida, dispersando-a mais de uma vez por toda a imagem, possibilitando esconder uma mesma mensagem (de tamanho pequeno) repetidas vezes na imagem. A característica principal do funcionamento desse método é alterar a luminosidade dos pontos da imagem (a, b). Enquanto que o brilho do ponto a é diminuído por um, o brilho de b é somado por um. Outra característica é sua resistência quanto ao processamento de imagens como cortes e rotações. As técnicas como *patchwork* são ideais para aplicações de marcas d'água das imagens.

Langelaar (1997) sugeriu alguns métodos que procuram marcar etiquetas nas imagens através da manipulação da intensidade do brilho de blocos do *pixel* que compõem a imagem, trocando-os por um determinado valor. Este método tem baixa resistência à compressão do JPEG, resistência esta influenciada pelo tamanho dos blocos do *pixel* utilizados, além de que, apresenta baixa visibilidade da etiqueta.

3.4.3 ESTEGANOGRAFIA EM ÁUDIO

Trabalhar com inserção de dados em sinais de áudio é um desafio com alto grau de complexidade em conseqüência dos fatores envolvidos nesta mídia, como o sistema auditivo humano (HAS) (Sellars, 1999). O HAS possui uma capacidade incrível de perceber faixas enormes de freqüências e também é muito sensível ao ruído. Segundo Bender (1996), enquanto o sistema auditivo tem uma faixa dinâmica relativamente grande, ele tem uma faixa de diferenciação de certa forma pequena - sons fortes tendem a abafar sons fracos. Ao tomar

conhecimento do sistema auditivo humano objetivando ocultar dados em áudio, é importante que suas fraquezas sejam levadas em consideração, assim como sua extrema sensibilidade.

3.4.3.1 AMBIENTES DE ÁUDIO

Quando se trabalha com a transmissão de sinais de áudio, Sellars (1999) menciona da importância de duas considerações principais. Primeiro, a maneira de como é armazenado o áudio, ou representação digital do áudio, e em seguida, o meio de transmissão do sinal.

3.4.3.1.1 REPRESENTAÇÃO DIGITAL

Segundo Sellars (1999), os arquivos de áudio digital possuem normalmente duas características preliminares:

a) método da quantização da amostra: é representado pela quantização linear de 16-*bits*, tendo como exemplos os formatos WAV (*Windows audio-Visual*) e AIFF (*Áudio Interchange File Format*). Distorções do sinal podem ser percebidas nestes formatos;
b) taxa temporal da amostragem: é representado pela amostragem temporal com freqüência de 8kHz (quilohertz, 9.6kHz, 10kHz, 12kHz, 16kHz, 22.05kHz e 44.1kHz).

Outra representação digital muito interessante que deve ser levada em consideração é o formato ISO MPEG-Audio. Como esse formato utiliza métodos de compressão de som, ele muda totalmente as estatísticas do sinal, pois codifica apenas as faixas de freqüência percebidas pelo sistema auditivo humano, conseguindo manter o som, mas mudando o sinal, devido à compressão (Sellars, 1999).

3.4.3.1.2 MEIOS DE TRANSMISSÃO

O meio de transmissão, ou ambiente da transmissão de um sinal de áudio consultam aos ambientes que o sinal pode ir completamente do codificador ao decodificador.

Conforme Bender (1996), quatro ambientes de transmissão são identificados:

a) ambiente *end-to-end* Digital: Quando um arquivo de áudio é copiado diretamente de uma máquina para outra sem ocorrer nenhuma mudança do mesmo, este por

conseqüência passará por todo o ambiente, sem apresentar muitas restrições na ocultação de dados;

b) ambiente de amostra Aumentada/Diminuída: o sinal é regravado em altas ou baixas taxas de reprodução, permanecendo no modo digital;
c) transmissão análoga e reamostragem: um sinal é convertido para modo analógico, transmitido em uma linha análoga e regravado;
d) ambiente “no ar”: o sinal é “jogado no ar” e “regravado com um microfone”. Deste modo, o sinal poderá sofrer algumas modificações não-lineares desconhecidas, podendo causar mudanças de fase, amplitude, ecos, etc.

3.4.3.2 MÉTODOS DE OCULTAÇÃO DE DADOS EM ÁUDIO

Existem vários métodos de ocultação de dados na mídia áudio, muitos com características semelhantes a outros métodos já mencionados, mas devidamente adequados a seu ambiente de utilização. Conforme Sellars (1999), os mais importantes e conhecidos serão descritos em seguida.

3.4.3.2.1 CODIFICAÇÃO DOS *BITS* BAIXOS

Quando o funcionamento da codificação dos *bits* baixos é analisado, logo se tem em mente o funcionamento da técnica de inserção no *bit* menos significativo na mídia imagem. Assim como a LSB nas imagens, a inserção dos dados da mensagem pode ser efetuada da mesma maneira nos *bits* menos significativos de arquivos de áudio.

Sua principal vantagem é ser um método com certa facilidade de implementação. Como desvantagem, assim como no método para imagens, possui baixa resistência a qualquer manipulação sobre a mídia. Alguns fatores diretamente relacionados a essa desvantagem como o ruído do canal e regravação, tem grandes possibilidades de destruir a mensagem.

Visando contornar esse problema, um método mais resistente à manipulação foi descrito por Bassia (1998). Para isto, é necessário alterar ligeiramente a amplitude de cada amostra, impedindo assim uma diferença perceptual. O resultado é a alta resistência na compressão MPEG, bem como a outras formas da manipulação do sinal, como filtrar ou regravar.

3.4.3.2.2 CODIFICAÇÃO DA FASE

O método de codificação da fase funciona a partir da substituição da fase de um segmento de áudio inicial por uma fase de referência que represente os dados. Segue abaixo o procedimento para codificação da fase:

a) a seqüência original do som é quebrada em n segmentos curtos;
b) é aplicada a cada segmento uma transformação de *Fourier* discreta (DFT), onde a cada quebra é criada uma matriz da fase e do valor;
c) a diferença de fase entre cada segmento adjacente é calculada;
d) para o primeiro segmento (S0), uma fase absoluta artificial p0 é criada;
e) para todos os outros segmentos, novos quadros de fase são criados;
f) uma fase nova e o valor original são combinados para gerar um novo segmento (Sn);
g) os novos segmentos são concatenados para criar assim a saída codificada.

Para que a decodificação seja processada, deve-se fazer a sincronização da seqüência antes de decodificar. O comprimento do segmento, os pontos de DFT, e o intervalo dos dados devem ser conhecidos por receptores. O valor da fase abaixo do primeiro segmento é detectado como 0 ou 1, que representam a seqüência binária codificada.

3.4.3.2.3 DIFUSÃO DO ESPECTRO (*SPREAD SPECTRUM*)

A maioria de canais de comunicação tentam concentrar dados de áudio dentro de uma região do espectro da freqüência mais estreita possível a fim conservar a largura e o poder de banda. Contudo, quando se utiliza a técnica de propagação do espectro, os dados codificados são difundidos transversalmente através do espectro da freqüência.

Direct Sequence Spread Spectrum (DSSS), um método discutido por Bender (1996), faz propagações do sinal multiplicando-o por uma seqüência pseudo-randômica de determinado comprimento máximo, conhecido como *chip*. A taxa da amostragem do sinal do *host* é usada como a taxa do *chip* para a codificação. Os cálculos da quantidade iniciais e finais para o fechamento da fase devem ser discretos, seguindo o sinal do *host*. Como resultado disso, são possíveis uma taxa do chip e uma taxa de dados associada mais elevada. Porém, diferente da codificação da fase, DSSS introduz um ruído aleatório ao som.

3.4.3.2.4 OCULTAÇÃO DE DADOS NO ECO

Ocultação de dados no eco insere dados em um sinal do *host* introduzindo um eco. Os dados são escondidos, variando três parâmetros do eco: a amplitude inicial, a taxa de deterioração, e o *offset* inicial, ou *delay*. Como o *offset* entre o original e o eco diminui, os dois sinais se misturam. Em um certo ponto, o ouvido humano não pode distinguir entre os dois sinais, e o eco é ouvido meramente como uma ressonância. Este ponto depende dos fatores tais como a qualidade da gravação original, do tipo de som, e do ouvinte.

Usando dois diferentes *delay*, ambos imperceptíveis ao ouvido humano, pode-se codificar um ou zero binário. A taxa da deterioração e a amplitude inicial podem também ser ajustadas abaixo da percepção do sistema auditivo humano, para assegurar-se de que a informação não seja percebida. Para codificar mais de um *bit*, o sinal original é dividido em parcelas menores, cada uma pode ser ecoada para codificar o *bit* desejado. O sinal codificado final é então apenas a recombinação de todas as parcelas independentemente codificadas do sinal. Enquanto um binário está representado por um certo *delay* y, e um zero binário está representado por um certo *delay* x, a detecção do sinal encaixado então envolve apenas a detecção de espaçamento entre os ecos.

3.5 CARACTERIZANDO TÉCNICAS DE OCULTAR DADOS

A técnica da esteganografia insere uma mensagem dentro de uma mensagem envelope, e segundo Lin (2002) vários fatores caracterizam os pontos fortes e fracos dos métodos existentes. A importância de cada fator depende principalmente da aplicação.

3.5.1 CAPACIDADE DE OCULTAR

O tamanho da informação a ser oculta está relacionado diretamente ao tamanho do envelope. A alta capacidade de ocultar é verificada através da utilização de um envelope pequeno para receber uma mensagem de tamanho fixo.

3.5.2 TRANSPARÊNCIA NA PERCEPÇÃO

Para a ocultação de uma mensagem no envelope são necessárias algumas alterações da mídia original. O importante é que após o processo de inserção da mensagem não ocorra degradação significante, bem como perda perceptível da qualidade do envelope. Em

aplicações com o objetivo de realizar comunicações secretas, se alguém perceber alguma diferença que deixe transparecer a presença de dados ocultos na mídia esteganografada, a técnica de esteganografia utilizada não foi eficiente o suficiente. Preservar a transparência de percepção em uma marca d'água inserida em uma imagem para efeitos de proteção de direitos autorais também é importante, porque a integridade do trabalho original precisa ser mantida. Para aplicações onde a transparência de percepção dos dados inseridos não é crítica, permite-se que uma maior distorção na mídia esteganografada, possa incrementar a capacidade de ocultar, a robustez ou ambos.

3.5.3 ROBUSTEZ

Robustez refere-se à resistência dos dados inseridos permanecerem intactos, mesmo que a mídia esteganografada sofra modificações, como filtragem linear ou não-linear, inserção de ruídos aleatórios, recortar ou redimensionar, etc. Robustez é crítica na proteção de direitos autorais, pois falsificadores podem tentar filtrar ou destruir quaisquer marcas d'água inseridas em imagens, por exemplo.

3.5.4 CUIDADO COM A RESISTÊNCIA

Além da robustez contra a destruição, o cuidado com a resistência refere-se à dificuldade que alguém terá para alterar ou forjar uma mensagem inserida em uma imagem esteganografada, como um falsificador substituindo uma marca d'água original pela sua, alterando o autor da imagem, por exemplo. Aplicações que necessitam de alta robustez geralmente necessitam também de um alto grau de cuidado com a resistência. Nas aplicações de proteção de direitos autorais, alcançar um alto cuidado com a resistência pode ser difícil porque o direito autoral é efetivo por muitos anos e a marca d'água precisa resistir mesmo que alguém tente modificá-la usando recursos computacionais avançados.

3.5.5 OUTRAS CARACTERÍSTICAS

Bender (1996) adverte sobre algumas restrições e características:

a) os dados originais não podem sofrer alterações significativas pelo processo de inserção dos dados e os dados inseridos devem ser os mais imperceptíveis possíveis;

b) os dados inseridos devem estar os mais imunes possíveis de modificações de ataques inteligentes ou manipulações;
c) um pouco de distorção ou degradação dos dados inseridos pode acontecer quando a mensagem esteganografada passa por modificações. Isto pode ser amenizado utilizando-se de códigos de correção de erros;
d) os dados inseridos devem ser reentrantes. Isto garante que os dados inseridos poderão ser extraídos quando apenas parte da mensagem esteganografada estiver disponível.

4 DESENVOLVIMENTO DO PROTÓTIPO

4.1 REQUISITOS DO SISTEMA

Com base nos conceitos apresentados nos capítulos anteriores, tornou-se possível o desenvolvimento do protótipo, sendo que o método utilizado foi o LSB (*Least Significant Bit*) *insertion*, que possibilitou a inserção e extração de mensagens em imagens de formato do tipo BMP. O protótipo possibilita ainda utilizar os demais tipos de formatos descritos no trabalho (GIF e JPEG), mas gerando no processo de inserção de mensagens, sempre imagens de formato do tipo BMP, justamente para contornar os problemas encontrados com os algoritmos de compressão e descompressão de imagens.

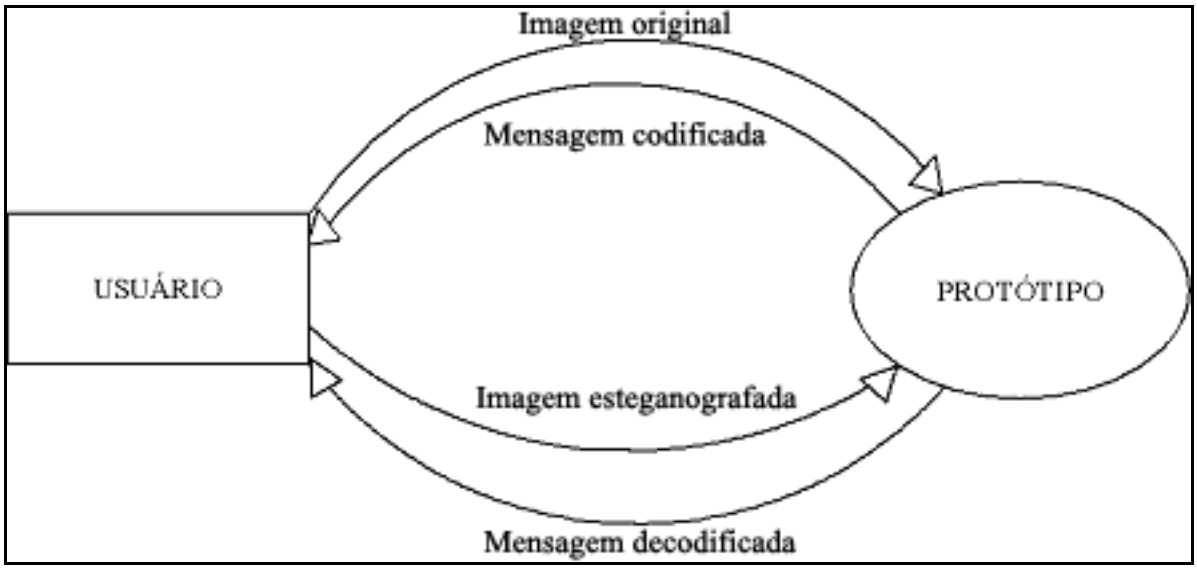
4.2 ESPECIFICAÇÃO DO PROTÓTIPO

Segundo Melendez (1990), para o desenvolvimento de sistemas de informação, a prototipação representa uma boa solução para a maioria dos problemas. A metodologia utilizada neste protótipo é a prototipação evolutiva. Através dela, o produto final será o próprio sistema, na sua forma mais aperfeiçoada. Sendo que o resultado deste processo evolutivo de prototipação tem-se o Diagrama de Contexto, Fluxograma e o Dicionário de Dados descritos a seguir.

4.2.1 DIAGRAMA DE CONTEXTO

O diagrama de contexto é uma representação gráfica do sistema como um todo e os seus relacionamentos (Melendez, 1990). Na fig.13, a imagem inserida receberá uma mensagem a ser codificada resultando em uma imagem esteganografada. Os dados da mensagem serão recuperados através do processo de decodificação.

Figura 13 – Diagrama de Contexto



4.2.2 FLUXOGRAMA

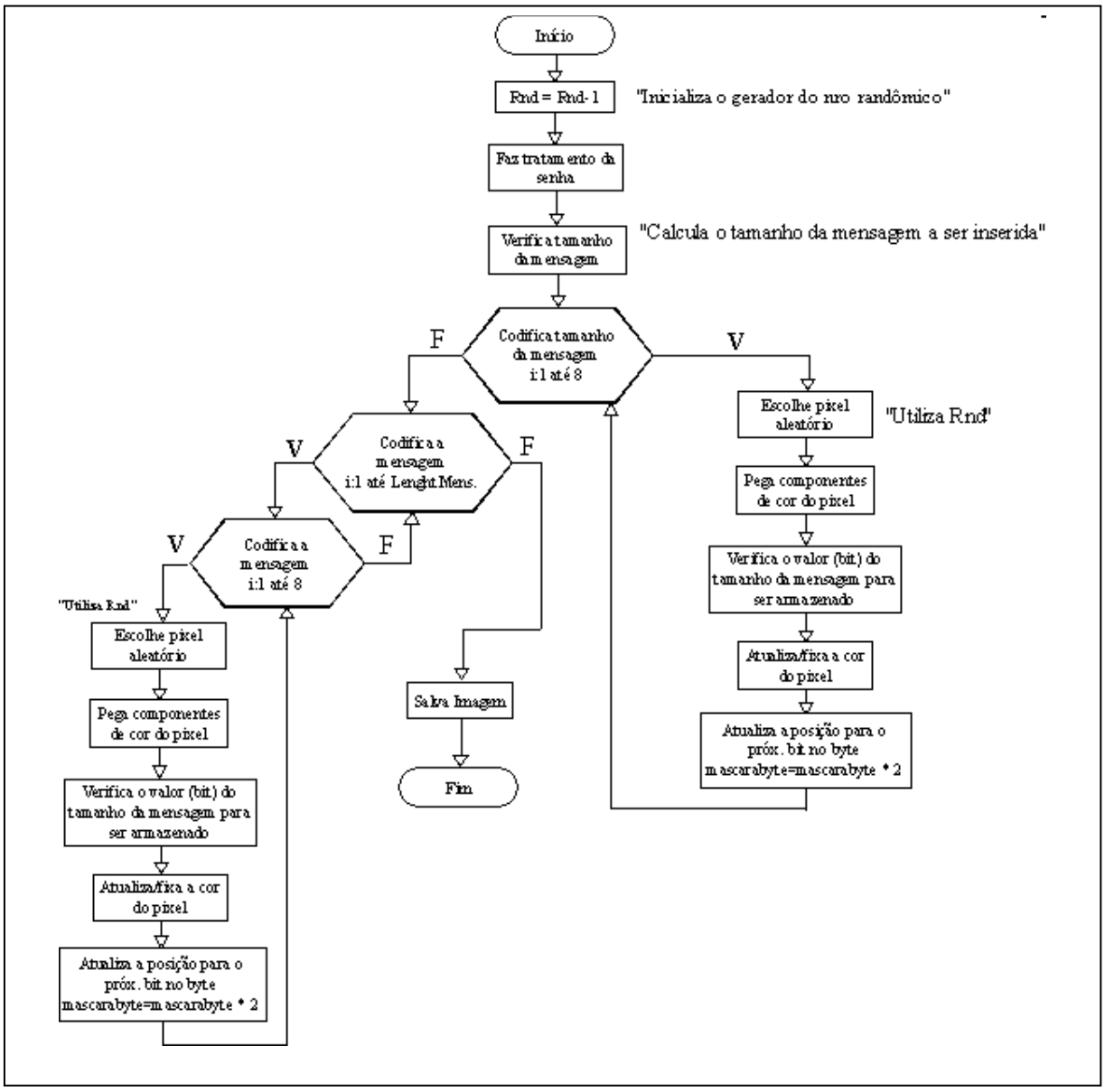
Na fig.14, encontra-se o Fluxograma genérico do algoritmo de codificação do protótipo, que representa todos os processos envolvidos nesta fase.

O primeiro passo é a inicialização de um gerador de número randômico, que é o responsável pela dispersão da mensagem na imagem. Esses números são gerados de forma pré-definida para que possam ser recuperados no processo de decodificação. Após, ocorre um tratamento de senha, para que a mesma seja comparada com a senha a ser informada na fase de decodificação.

Em seguida, o tamanho da mensagem é verificado e armazenado para que a mesma quantidade de caracteres seja recuperada na próxima fase. Para ser armazenado, tem-se alguns processos. O primeiro é buscar um *pixel* na imagem utilizando-se do número responsável pela dispersão da mensagem. Em seguida é verificado o componente de cor do *pixel* escolhido que irá guardar o valor do *bit* correspondente a mensagem. O valor do componente de cor é atualizado após a inserção e é fixada a cor no *pixel*. Após, a posição do *bit* no *byte* é atualizada para que o próximo *bit* da mensagem seja inserido na imagem.

Depois que o tamanho da mensagem já foi inserido na imagem, o próximo passo é a codificação da própria mensagem. Enquanto existirem letras a serem codificadas, cada *bit* que compõem cada letra será inserido na imagem da mesma forma que o tamanho da mensagem foi inserido.

Figura 14 – Fluxograma genérico do algoritmo de codificação



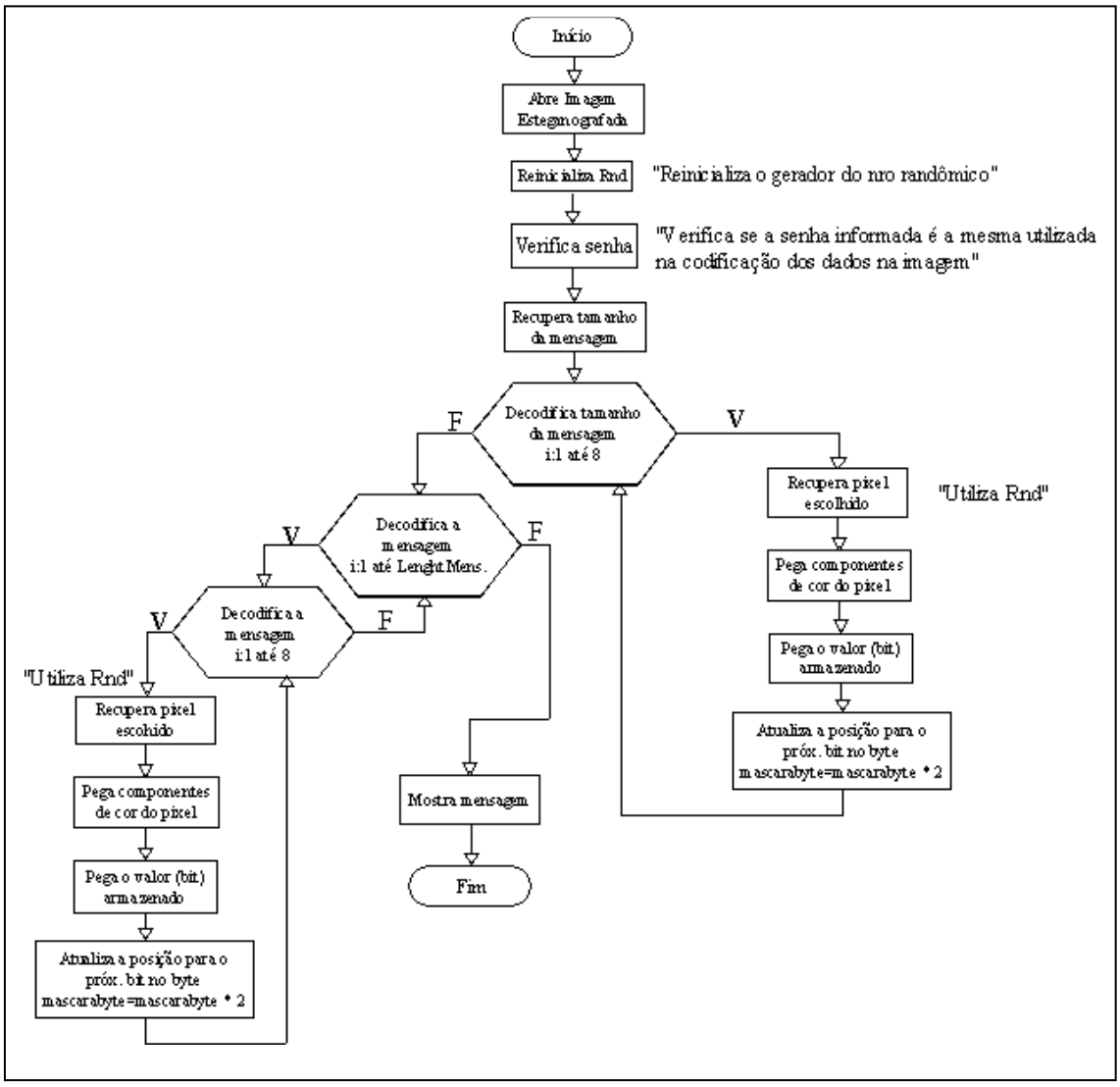
Na fig.15, encontra-se o Fluxograma genérico do algoritmo de decodificação do protótipo.

O primeiro passo da decodificação é a reinicialização do gerador de número randômico com a mesma semente utilizada na fase de codificação, para produzir a mesma série de números pseudo-randômicos, conseguindo assim recuperar os mesmos valores inseridos de forma dispersa na imagem. Após, ocorre a verificação da senha, para validar as demais operações do processo de decodificação.

Em seguida, o tamanho da mensagem inserida na imagem é recuperado. Para isso, é preciso buscar os *pixels* que contém esses valores na imagem utilizando-se do número responsável pela dispersão da mensagem. Em seguida é verificado o componente de cor do *pixel* que contém o valor do *bit* correspondente ao tamanho da mensagem. Após, a posição do *bit* no *byte* é atualizada para que o próximo *bit* do tamanho da mensagem seja recuperado.

Depois que o tamanho da mensagem foi recuperado, o próximo passo é a decodificação da própria mensagem. Enquanto existirem letras a serem decodificadas, cada *bit* que compõem cada letra será recuperado na imagem da mesma forma que o tamanho da mensagem foi. Para cada letra recuperada, a mesma é armazenada em uma variável que ao final exibirá a mensagem ao interessado.

Figura 15 - Fluxograma genérico do algoritmo de decodificação



4.2.3 DICIONÁRIO DE DADOS

Na tabela 2, encontra-se o Dicionário de Dados do Protótipo.

Tabela 2 – Dicionário de Dados

Nome	Tipo	Tamanho	Descrição
`Pixel`	Inteiro	1	Informa qual componente de cor do *pixel* foi escolhido
`R`	Inteiro	3	Sistema de coordenadas – indica a largura da imagem em *pixels*
`C`	Inteiro	3	Sistema de coordenadas – indica a altura da imagem em *pixels*
`Mascara_byte`	Inteiro	3	Posição do *bit* da mensagem no *byte*
`Mascara_bit`	Inteiro	3	Valor do *bit* que compõe a mensagem a ser inserida na imagem.
`Clrr`	Byte	3	Valor correspondente ao componente RGB de cor *Red*
`Clrg`	Byte	3	Valor correspondente ao componente RGB de cor *Green*
`Clrb`	Byte	3	Valor correspondente ao componente RGB de cor *Blue*
`Mostra_pixels`	Boolean	1	Valor para mostrar os pontos na imagem ocupados pela mensagem
`Posição_usada`	Coleção	Altura x Largura	Vetor para armazenar os dados inseridos na imagem
`Rnd`	Real	1	Número randômico para processar a distribuição dos *bits* da mensagem na imagem
`Wid`	Inteiro	3	Recebe o valor da largura da imagem
`Hgt`	Inteiro	3	Recebe o valor da altura da imagem
`Msg_length`	Byte	3	Recebe o tamanho da mensagem
`Msg`	String	3	Recebe o conteúdo da mensagem
`Ch`	Byte	3	Recebe valores em ASCI referentes à mensagem

4.2.4 ESTRUTURA DO PROTÓTIPO

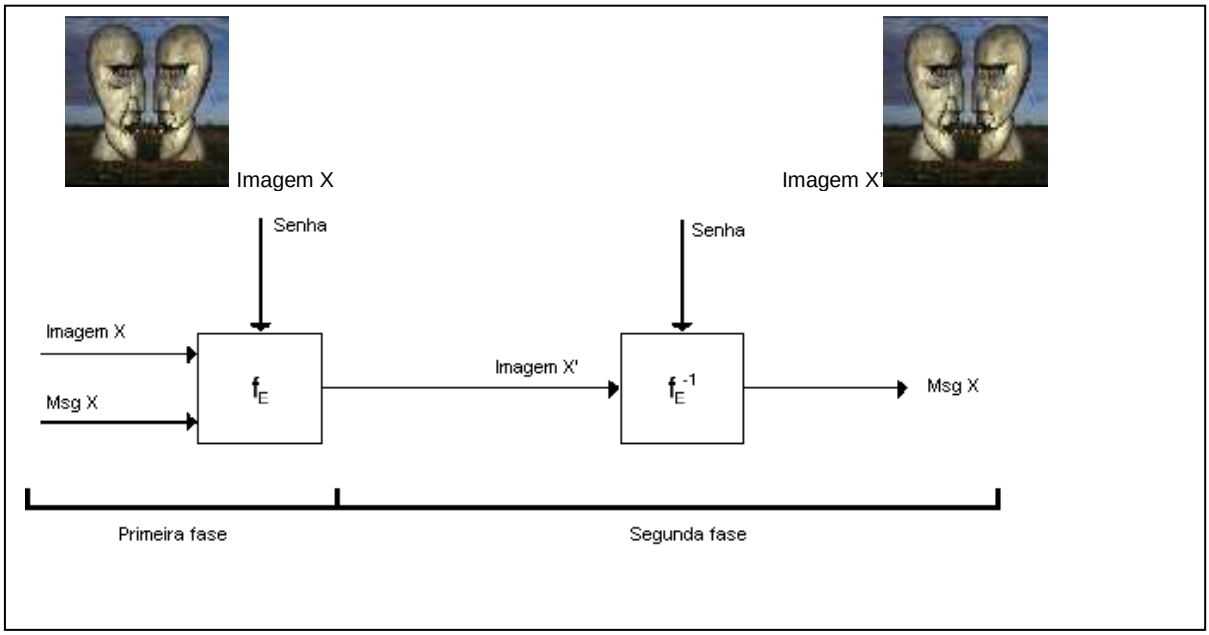
A estrutura do protótipo pode ser dividida em duas fases distintas.

A primeira fase pode ser denominada fase de inserção (codificação), onde uma imagem é selecionada para em seguida receber a codificação da mensagem mediante a informação da senha. Nessa fase, para cada *bit* que compõe uma letra da mensagem, é feita uma busca dispersa de um *pixel* na imagem utilizando-se de um número randômico pré-definido, bem como uma escolha entre os componentes de cor (RGB) que formam o *pixel*. Em

seguida, é verificado o valor do respectivo *bit* a ser inserido, e processado a atualização e fixação da cor.

A segunda e última fase é o processo inverso da primeira, onde os dados codificados na imagem são extraídos. Nela, os *pixels* escolhidos são recuperados utilizando-se o mesmo processo da fase de inserção. Para isto, utiliza-se como valor inicial do número randômico, o mesmo valor utilizado no processo de codificação. Para cada *pixel* recuperado, bem como o respectivo componente de cor com o *bit* inserido, ocorre a recuperação do valor do *bit* até que a letra da mensagem seja composta e conseqüentemente a mensagem. A fig.16 ilustra as duas fases da estrutura do protótipo.

Figura 16 – Estrutura do protótipo



Primeira fase: a Imagem X corresponde à imagem onde a mensagem (Msg X) será inserida. Nesta fase uma senha será informada. A função esteganográfica f_E será responsável pela codificação da mensagem na imagem, recebendo, para isto, como parâmetros Imagem X, Msg X e Senha.

Segunda fase: a Imagem X’ corresponde à imagem com mensagem inserida em seus dados. A função esteganográfica f_E^{-1} será responsável pela decodificação da mensagem na imagem, recebendo, para isto, como parâmetros Imagem X’e Senha.

4.3 IMPLEMENTAÇÃO DO PROTÓTIPO

Este protótipo teve sua implementação realizada no ambiente de programação *Visual Basic* 6.0, por possuir uma interface visual que atendesse às necessidades para o desenvolvimento do trabalho. Foi utilizado para manipulação de imagens *raster*, o componente *PictureBox* por possuir funções que permitem controlar, *pixel* a *pixel*, as informações necessárias para os processo de codificação e decodificação das mensagens. Das principais funções, destacam-se os métodos *Point* e *Pset*. O primeiro ativa os *pixels* da imagem, e o segundo faz o processo de leitura de seus valores.

O *Visual Basic* ainda disponibiliza algumas funções para a manipulação de cores, muito utilizadas no desenvolvimento deste protótipo. A função RGB permite construir a cor a partir dos valores vermelho, verde e azul de cada *pixel*.

4.3.1 ALGORITMO DE CODIFICAÇÃO DA MENSAGEM

Os quadros 2 e 3 apresentam as implementações destinadas a codificar a mensagem em uma imagem *raster*, transformando-a em uma imagem esteganografada.

Quadro 2 – Codificação da mensagem (Parte 1)

```
Private Sub cmdCodifica_Click()
Dim msg As String
Dim i As Integer
Dim posicao_usada As Collection
Dim wid As Integer
Dim hgt As Integer
Dim mostra_pixels As Boolean
   Screen.MousePointer = vbHourglass
   DoEvents
   ' Inicializa o gerador do número randômico. Esse nro distribui de forma dispersa os bits que compoem
   ' a mensagem na imagem. Esses nros são gerados de forma pré-definida para que possam ser recuperados.
   Rnd -1
   Randomize SenhaX(txtSenha.Text)
   wid = Imagem.ScaleWidth
   hgt = Imagem.ScaleHeight
   msg = Left$(txtMensagem.Text, 255)
   mostra_pixels = chkMostraPixels.Value
   Set posicao_usada = New Collection
   ' Codifica tamanho da mensagem
   CodificaByte CByte(Len(msg)), _
      posicao_usada, wid, hgt, mostra_pixels
   ' Codifica a mensagem
   For i = 1 To Len(msg)
      CodificaByte Asc(Mid$(msg, i, 1)), _
         posicao_usada, wid, hgt, mostra_pixels
   Next i
   Imagem.Picture = Imagem.Image
   Screen.MousePointer = vbDefault
End Sub
```

Quadro 3 – Codificação da mensagem (Parte 2)

```
Private Sub CodificaByte(ByVal Valor As Byte, ByVal posicao_usada As Collection, ByVal wid As Integer,
ByVal hgt As Integer, ByVal mostra_pixels As Boolean)
Dim i As Integer
Dim mascara_byte As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim mascara_bit As Integer

   mascara_byte = 1
   For i = 1 To 8
      ' Escolhe um pixel aleatório e componente RGB.
      EscolhePosicao posicao_usada, wid, hgt, r, c, pixel
      ' Pega os componentes de cor do pixel
      UnRGB Imagem.Point(r, c), clrr, clrg, clrb
      If mostra_pixels Then
         clrr = 255
         clrg = clrg And &H1
         clrb = clrb And &H1
      End If

      ' Adquire o valor que será armazenado.
      If Valor And mascara_byte Then
         mascara_bit = 1
      Else
         mascara_bit = 0
      End If
      ' Atualiza a cor.
      Select Case pixel
         Case 0
            clrr = (clrr And &HFE) Or mascara_bit
         Case 1
            clrg = (clrg And &HFE) Or mascara_bit
         Case 2
            clrb = (clrb And &HFE) Or mascara_bit
      End Select
      ' Fixa a cor do pixel.
      Imagem.PSet (r, c), RGB(clrr, clrg, clrb)

      mascara_byte = mascara_byte * 2
   Next i
End Sub
```

4.3.2 ALGORITMO DE DECODIFICAÇÃO DA MENSAGEM

Os quadros 4 e 5 apresentam as implementações destinadas a decodificar a mensagem em uma imagem *raster*, retornando o conteúdo da mensagem.

Quadro 4 – Decodificação da mensagem (Parte 1)

```
Private Sub cmdDecodifica_Click()
Dim msg_length As Byte
Dim msg As String
Dim ch As Byte
Dim i As Integer
Dim posicao_usada As Collection
Dim wid As Integer
Dim hgt As Integer
Dim mostra_pixels As Boolean
   Screen.MousePointer = vbHourglass
   DoEvents

   ' Reinicializa o gerador do número randômico com a mesma semente
   ' para produzir a mesma série de nros pseudo-randômicos, conseguindo
   ' recuperar os mesmos valores inseridos de forma dispersa na imagem.
   Rnd -1
   Randomize SenhaX(txtSenha.Text)

   wid = Imagem.ScaleWidth
   hgt = Imagem.ScaleHeight
   mostra_pixels = chkMostraPixels.Value
   Set posicao_usada = New Collection

   ' Decodifica tamanho da mensagem
   msg_length = DecodificaByte(posicao_usada, wid, hgt, mostra_pixels)

   ' Decodifica a mensagem
   For i = 1 To msg_length
      ch = DecodificaByte(posicao_usada, wid, hgt, mostra_pixels)
      msg = msg & Chr$(ch)
   Next i
   Imagem.Picture = Imagem.Image
   txtMensagem.Text = msg
   Screen.MousePointer = vbDefault
End Sub
```

Quadro 5 – Decodificação da mensagem (Parte 2)

```
Private Function DecodificaByte(ByVal posicao_usada As Collection, ByVal wid As
Integer, ByVal hgt As Integer, ByVal mostra_pixels As Boolean) As Byte
Dim Valor As Integer
Dim i As Integer
Dim mascara_byte As Integer
Dim r As Integer
Dim c As Integer
Dim pixel As Integer
Dim clrr As Byte
Dim clrg As Byte
Dim clrb As Byte
Dim mascara_bit As Integer

    mascara_byte = 1
    For i = 1 To 8
        ' Recupera o pixel aleatório e componente de RGB.
        EscolhePosicao posicao_usada, wid, hgt, r, c, pixel
        ' Pega os componentes de cor do pixel.
        UnRGB Imagem.Point(r, c), clrr, clrg, clrb
        ' Pega o valor armazenado.
        Select Case pixel
            Case 0
                mascara_bit = (clrr And &H1)
            Case 1
                mascara_bit = (clrg And &H1)
            Case 2
                mascara_bit = (clrb And &H1)
        End Select
        If mascara_bit Then
            Valor = Valor Or mascara_byte
        End If
        If mostra_pixels Then
            Imagem.PSet (r, c), RGB( _
                clrr And &H1, _
                clrg And &H1, _
                clrb And &H1)
        End If
        mascara_byte = mascara_byte * 2
    Next i
    DecodificaByte = CByte(Valor)
End Function
```

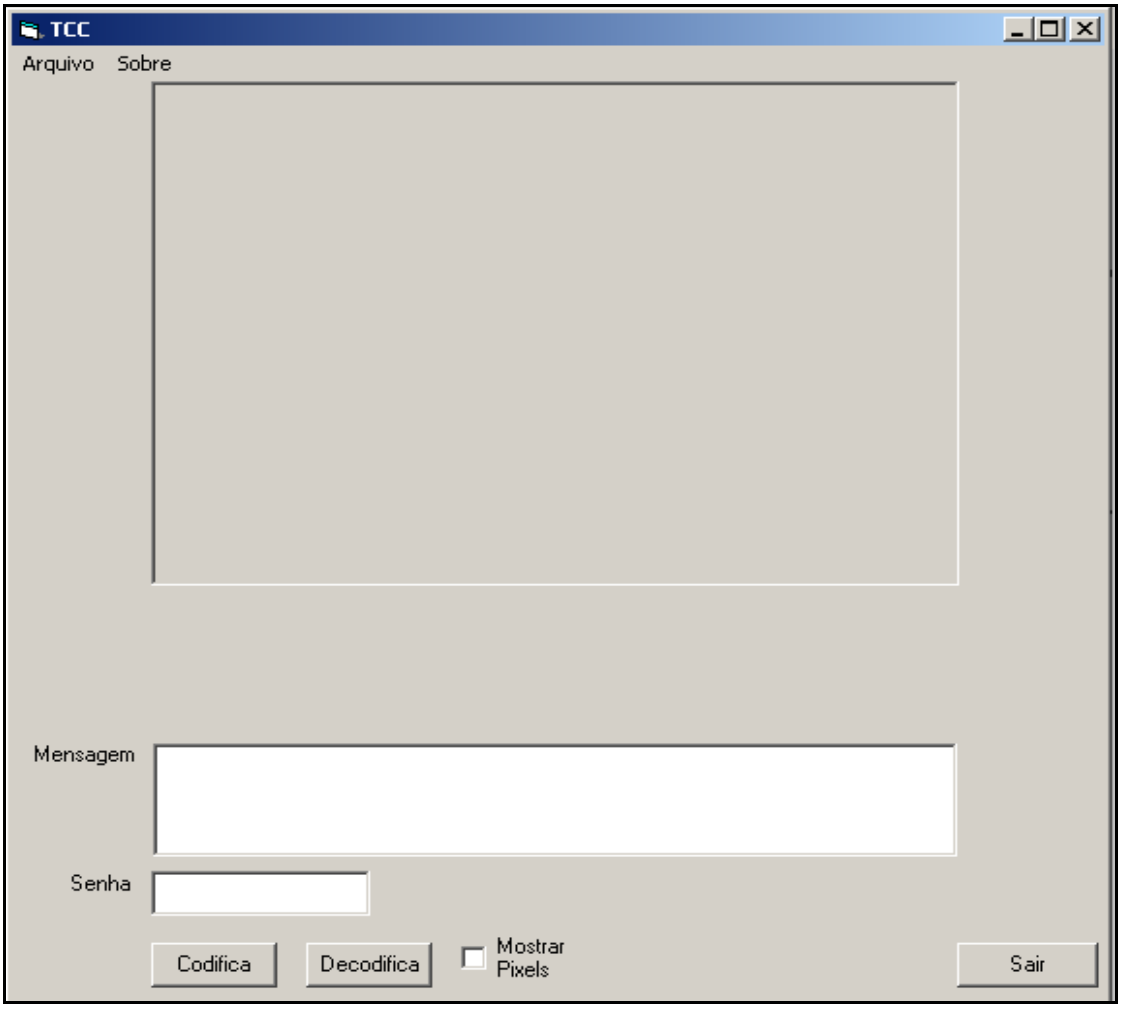
4.4 FUNCIONAMENTO DO PROTÓTIPO

Neste ponto será descrito o funcionamento do protótipo. A tela de apresentação pode ser vista na fig.17 abaixo.

Figura 17 – Tela de apresentação

A tela principal do protótipo, onde será feito os processamentos de codificação e decodificação das mensagens em imagens *raster* pode ser vista na fig.18.

Figura 18 – Tela principal do protótipo



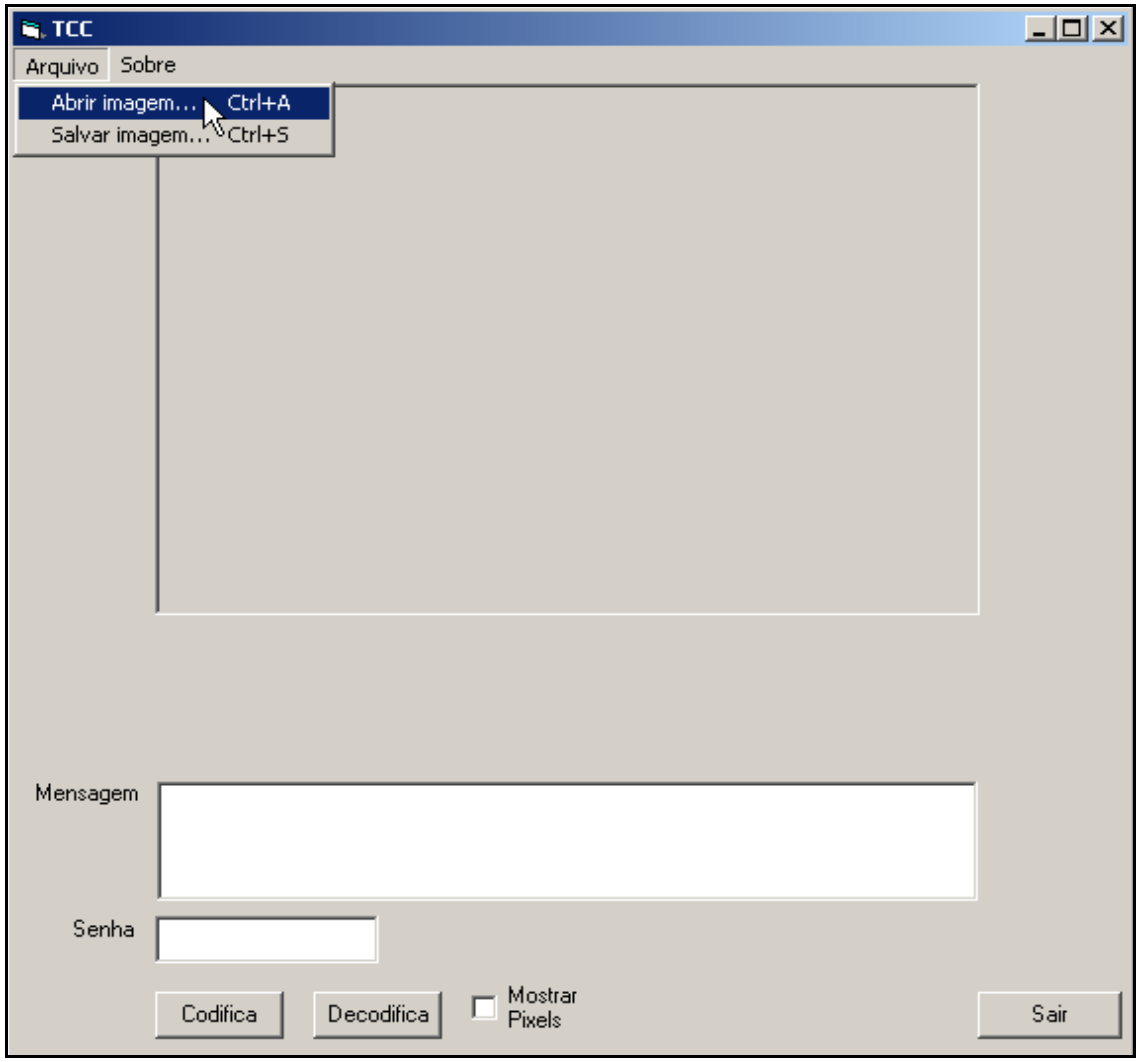
A tela Sobre contém informações sobre o desenvolvimento do protótipo (fig. 19).

Figura 19 – Tela Sobre

4.4.1 INSERINDO MENSAGEM EM IMAGENS

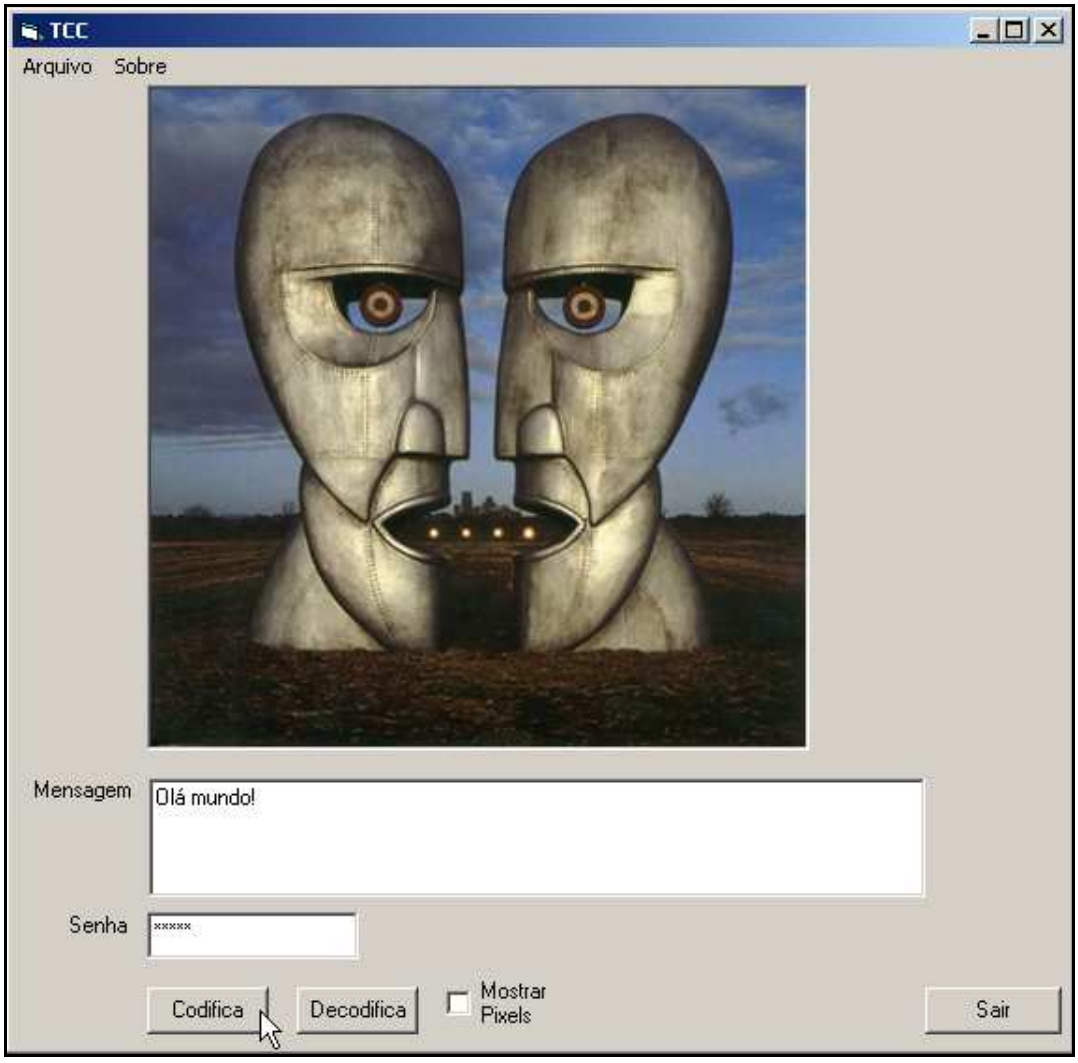
O primeiro passo na utilização do protótipo é a inclusão da imagem no ambiente. Basta clicar no menu Arquivo e em seguida selecionar a opção Abrir imagem (Ctrl + A), conforme mostra a fig.20.

Figura 20 – Abrindo a imagem



Em seguida basta escrever a mensagem a ser inserida na imagem. Para isto, a mensagem é digitada no campo "Mensagem", e a senha no campo "Senha" para posteriormente clicar no botão "Codifica". A fig.21 demonstra esse passo.

Figura 21 – Codificando a mensagem na imagem

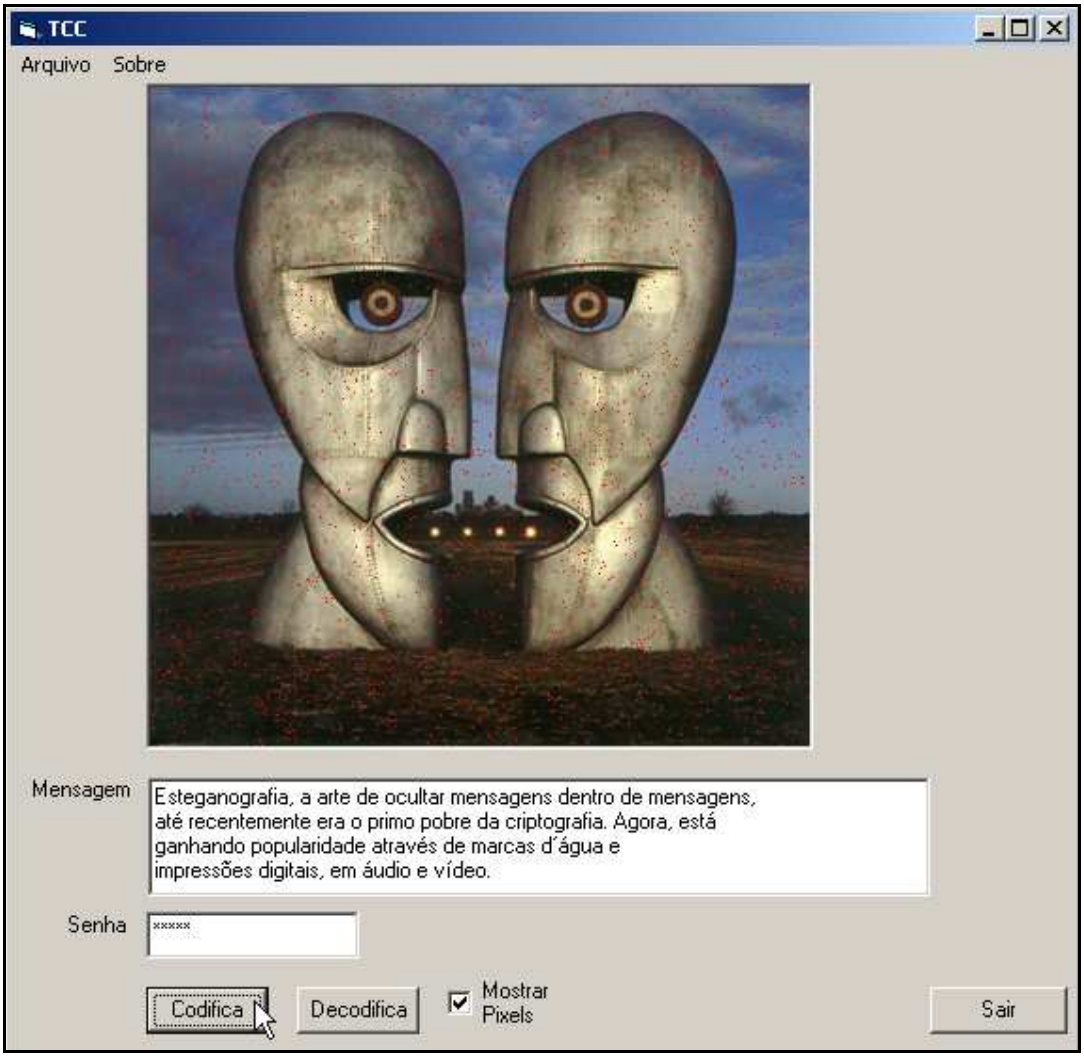


Após a mensagem ser inserida (codificada) na imagem, a mesma terá que ser salva. Para isto, basta clicar no menu Arquivo e selecionar a opção Salvar imagem (Ctrl + S), conforme é demonstrado na fig.22.

Figura 22 – Salvando a imagem

Ainda na etapa de codificação, pode-se visualizar as posições ocupadas pelos *bits* da mensagem inseridos nos *pixels* da imagem. Para isto, existe a opção “Mostrar Pixels” que, se selecionada, pinta os *pixels* com a cor vermelha, conforme a fig. 23 demonstra.

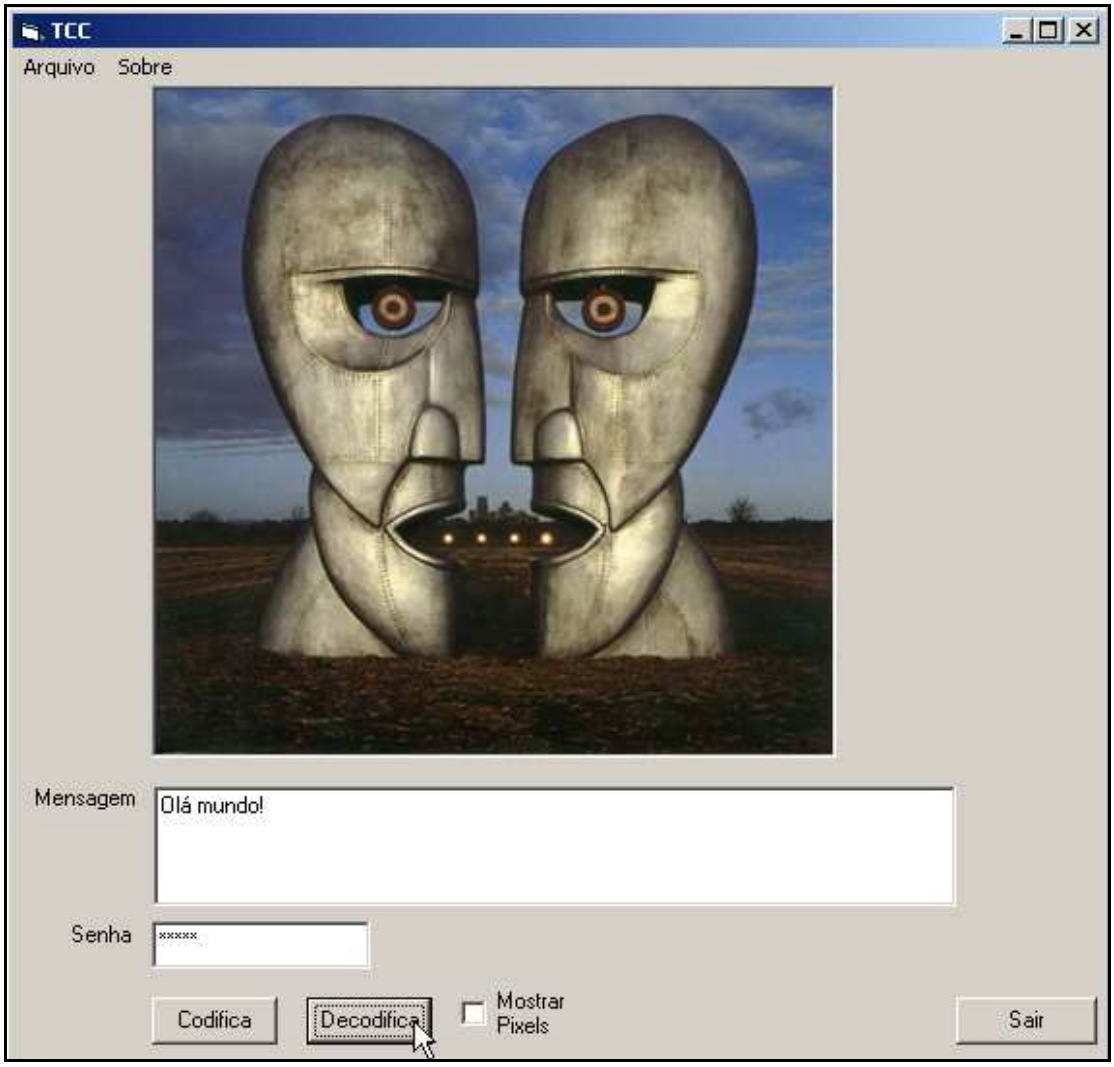
Figura 23 – Mostrando os *pixels* ocupados



4.4.2 DECODIFICANDO A MENSAGEM

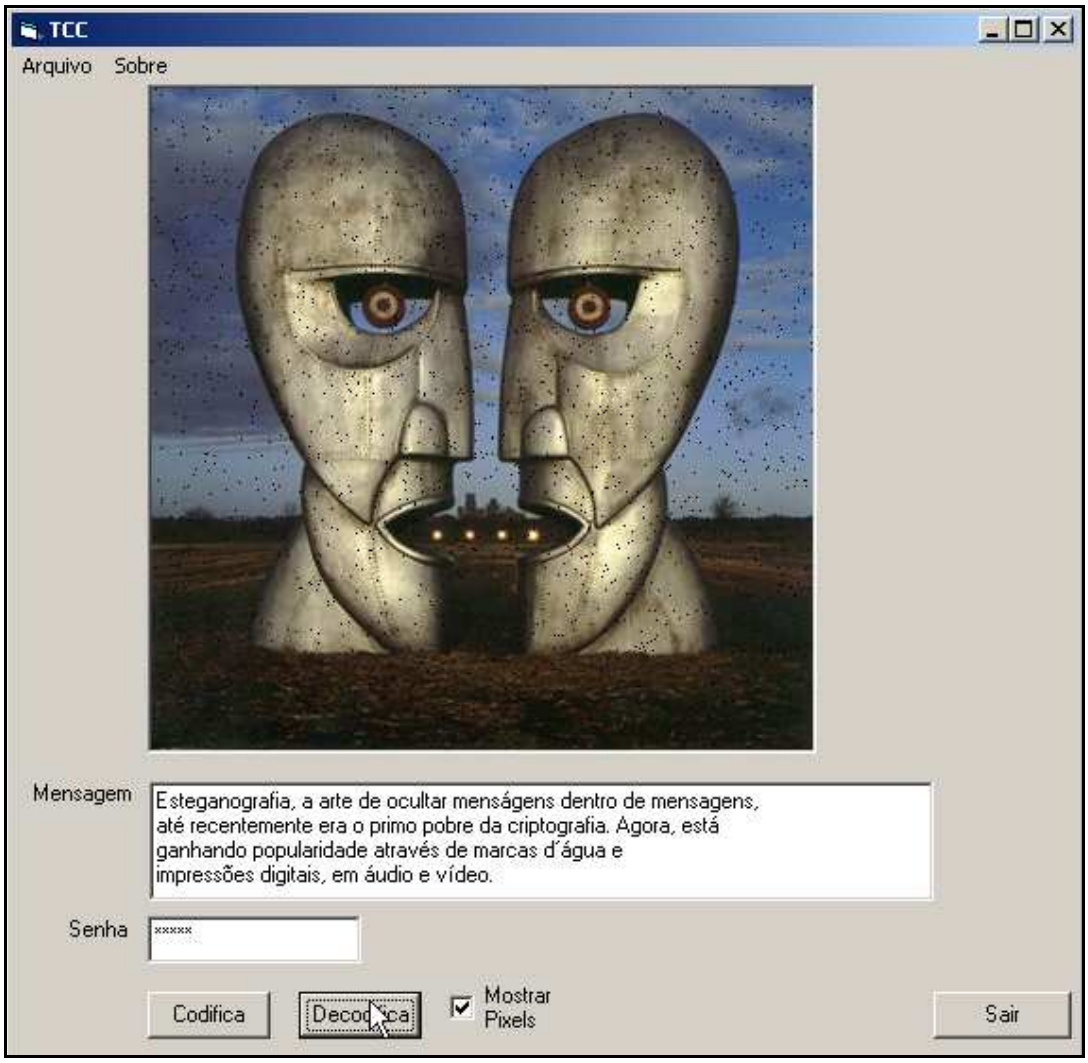
Para visualizar a mensagem codificada na imagem esteganografada, a mesma terá que ser inserida no ambiente do protótipo, bastando apenas selecionar a opção no arquivo Abrir imagem (Ctrl + A). Em seguida, digita-se a senha (a mesma utilizada para inserir a mensagem) para depois clicar no botão "Decodifica". Se a senha estiver correta, a mensagem aparecerá no campo "Mensagem", conforme a fig.24.

Figura 24 – Decodificando a mensagem em uma imagem esteganografada



Se a imagem a ser decodificada foi salva com a opção “Mostrar Pixels” setada, o protótipo tem a opção de “pintar” estes mesmos *pixels* com a cor preta. Para isto, deverá ser selecionada a opção “Mostrar Pixels” e clicar em “Decodifica”. A fig. 25 ilustra isso.

Figura 25 – Decodificando com a opção Mostrar Pixels setada



5 RESULTADOS FINAIS

Este capítulo apresenta as conclusões, limitações e sugestões referentes ao trabalho desenvolvido.

5.1 CONCLUSÕES

Baseando-se no levantamento bibliográfico realizado, o protótipo pôde ser construído com o intuito de alcançar os objetivos propostos. Ele está apto a inserir e extrair mensagens em imagens *raster* do tipo BMP, através da técnica *Least significant bit (LSB) insertion*, da qual ilustra o conceito de Esteganografia.

Os resultados de codificar uma mensagem sem haver alterações significativas nas imagens foram bem sucedidos. Visualmente, não se percebe nenhuma alteração de cor na imagem esteganografada, se comparada com a imagem original. Para isto optou-se por inserir os dados de forma dispersa na imagem, fazendo com que a percepção de alguma mudança de cor, caso houvesse, fosse amenizada. As fig. 26 e 28 correspondem a algumas das imagens originais utilizadas durante o desenvolvimento do protótipo. As fig. 27 e 29 correspondem às mesmas imagens (26 e 28) respectivamente, mas com dados codificados em seus *pixels*.

Figura 26 – Imagem original

Figura 27 – Imagem esteganografada

Figura 28 – Imagem original

Figura 29 – Imagem esteganografada

A recuperação das mensagens inseridas também obtiveram resultados precisos, já que todas as mensagens codificadas foram recuperadas durante a decodificação sem haver perda de informação, tornando o protótipo uma ferramenta íntegra e segura na transmissão de dados ocultos.

Durante a fase de desenvolvimento, percebeu-se que a técnica implementada é relativamente frágil, pois depende do *bit* menos significativo de uma imagem. Estes *bits* são os mais modificados pelos algoritmos de compressão de imagens, como o JPEG. Se uma mensagem for codificada em uma imagem que utilize algoritmo de compressão e salva com o mesmo formato, ocorrerá a perda de informação. Provavelmente o resultado não terá nada a não ser lixo. Para contornar esse problema foi padronizado no protótipo o formato das imagens esteganografadas como sendo do tipo BMP.

De acordo com os resultados obtidos, os objetivos propostos foram atingidos, tanto na implementação do algoritmo de codificação e decodificação de mensagens em imagens *raster*, quanto na introdução dos conceitos de uma nova tecnologia no meio acadêmico, abrindo portas para os conhecimentos de uma arte antiga, mas que ainda tem muito que se aprender.

5.2 EXTENSÕES

Como extensão deste trabalho, sugere-se aplicar outras técnicas de esteganografia, tanto em imagens, quanto nas demais mídias que a esteganografia pode ser empregada, abrangendo assim todos os conceitos de esteganografia e suas principais aplicações.

Propõe-se também desenvolver a mesma análise realizada neste trabalho, mas incorporando técnicas de criptografia na fase de codificação e decodificação dos dados, resultando em uma maior segurança das informações inseridas nas imagens.

Outra extensão com importante relevância seria a implementação de técnicas esteganográficas em imagens com algoritmos de compressão, para suprir a deficiência que possui no que diz respeito à recuperação de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

AGOSTINI, Luciano Volcan. **Estudo de padrões de compressão de imagens para aplicações VLSI**, Porto Alegre, 1999. Disponível em: <http://www.inf.ufrgs.br/~agostini/TI_Image.pdf >. Acesso em: 11 ago 2002.

BANON, Gerald Jean Francis. **Bases da computação gráfica**. Rio de Janeiro: Campus, 1989.

BASSIA, Peter; PITAS, Ioannis. **Robust audio watermarking in the time domain**, [S.1.], 1998. Disponível em: <http://poseidon.csd.auth.gr/ voyatzis/creus.zip>. Acesso em: 10 out. 2002.

BENDER , Walter; GRUHL, Daniel; LU, Anthony; MORIMOTO, Norishige. **Techniques for data hiding**, [S.1.], 1996. Disponível em: <http://www.almaden.ibm.com/journal/sj/mit/sectiona/bender.html>. Acesso em: 15 nov. 2002.

BRASSIL, Jack; LOW, Steven H.; MAXENCHUK, Nicolas F.; O'GARMAN, Lawrence. **Electronic marking and identification techniques to discourage document copying**, Toronto, 1994. Disponível em: < http://citeseer.nj.nec.com/brassil94electronic.html >. Acesso em: 05 ago. 2002.

BROWN, C. Wayne; SHEPHERD, Barry J. **Graphics file formats – reference and guide**. Greenwich: Manning Publications Company, 1995.

DAVIS, Ryan; GREGG, Tina; PARSON, Todd; RUPP, Andrew. **Aplication of steganographic methods to 8-bit color images**, Ohio, 2002. Disponível em: <http://www.deliriumjournal.org/writings/engineers/steganographic.htm>. Acesso em: 25 ago. 2002.

FARINA, Modesto. **Psicodinâmica das cores em comunicação.** São Paulo: Edgard Blücher, 1990.

FERREIRA, Simone Bacellar Leal. **Espaço de cor**, Rio de Janeiro, 1999. Disponível em: <http://www.inf.puc-rio.br/~bacellar/cores/espaco.htm>. Acesso em: 29 jul. 2002.

FRANÇA NETO, Leopoldo Rodrigues. **Um Ambiente para Processamento de Grandes Acervos de Imagens**, 1998. 137 f. Dissertação (Mestrado em Informática) – Departamento de Informática, Universidade Federal de Pernambuco, Recife.

FRANCO, João Henrique. **A arte de esconder a informação**, [S.1.], 1998. Disponível em: <http://www.uol.com.br/webworld/tecnologia/joaohenrique/joaohenrique12.htm>. Acesso em: 27 fev. 2002.

GONÇALVES FILHO, Aurélio. **Física e realidade**. São Paulo: Sciplione, 1997.

HEARN, Donald; BAKER, Pauline. **Computer graphics**, New Jersey, 1994. Disponível em: <http://gbdi.icmc.sc.usp.br/documentacao/apostilas/cg/ap11.html>. Acesso em: 15 nov. 2002.

HOLMES, David P. **Introduction to digital image steganography**, [S.1.], 2002. Disponível em: <http://www.giac.org/practical/David_P_Holmes_GSEC.doc>. Acesso em: 11 out. 2002.

KUHN, Markus. **Steganography mailing list**, Hamburg, 1995. Disponível em: <http://www.jjtc.com/Steganography/steglist.htm>. Acesso em: 05 ago. 2002.

LANGELAAR, Gerhard C.; VAN DER LUBBE, Jan; LAGENDIJK, Reginald L.. **Robust labelling methods for copy protection of images**, [S.1.], 1997. Disponível em: <http://www-it.et.tudelft.nl/ gerhard/spie97.zip.>. Acesso em: 25 ago. 2002.

LI, Ze-Nian. **Image compression -** JPEG, Canadá, 2000. Disponível em: <http://www.cs.sfu.ca/CourseCentral/365/li/material/notes/Chap4/Chap4.2/Chap4.2.html>. Acesso em: 16 dez. 2002.

LIN, Eugene T.; DELP Edward J.. **A review of data hiding in digital images**, Indiana, [2002?]. Disponível em: <http://debut.cis.nctu.edu.tw/~yklee/Research/Steganography/Edward_J_Delp/PICS99.pdf>. Acesso em: 14 out. 2002.

LOPES, João M. Brisson. **Computação Gráfica:** formatos de imagem. Portugal, 2002. Disponível em: <http://mega.ist.utl.pt/~ic-cg/programa/livro/FormatosdeImagem.pdf>. Acesso em: 15 nov. 2002.

MACEDO, Luiz Francisco de. **Computação Gráfica**: uma breve revisão, [S.l.], 1998. Disponível em: <http://www.boatsonly.com/macedo/textos/top_img.htm>. Acesso em: 20 set. 2002.

MELENDEZ FILHO, Rubem. **Prototipação de sistemas de informações:** fundamentos, técnicas e metodologias. Rio de Janeiro: Livros Técnicos e Científicos, 1990.

MIANO, John. **Compressed image file formats** – JPEG, PNG, GIF, XBM, BMP . USA: Addison Wesley Longman, 1999.

MONIZ, Felipe. **Esteganografia**, [S.1.], [2002?]. Disponível em: <http://www.csv.hpg.ig.com.br/entrevista.htm>. Acesso em: 27 fev. 2002.

MURRAY, James E. **Formatos de arquivos gráficos**, [S.l.], 1994. Disponível em: <http://www.di.ufpe.br/~if291/documentos/formatos_graficos/formatos.htm>. Acesso em: 15 set. 2002.

MURRAY, James E.; VANRYPER, Willian. **Encyclopedia of graphics file formats**. USA: O'Reilly &Associates, 1996.

PERSIANO, Ronaldo César Marinho; OLIVEIRA, Antônio Alberto Fernandes de. **Introdução à computação gráfica**. Rio de Janeiro: Livros Técnicos e científicos, 1989.

PETROUTSOS, Evangelos. **Dominando o Visual Basic 6:** a bíblia. São Paulo: Makron Books, 1999.

REIS, Dalton Solano dos. **Comparação de performance**, Blumenau, 1998. Disponível em: <http://www.inf.furb.rct-sc.br/~dalton/DiscipCG/ModuloA/moAtop02.htm>. Acesso em 15 set. 2002.

SELLARS, Duncan. **An introduction to steganography**, [S.1.], [1999?]. Disponível em: <http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/stego.html>. Acesso em: 23 jun. 2002.

SIEBERG, Daniel. **Bin Laden exploits technology to suit his needs**, [S.1.], 2001. Disponível em: <http://www.cnn.com/2001/US/09/20/inv.terrorist.search/ >. Acesso em: 04 mar. 2002.

VELHO, Luiz. **Computação Gráfica:** Imagem. Sociedade Brasileira de Matemática. Rio de Janeiro, 1994.