

## 4. DESENVOLVIMENTO

Existem incontáveis técnicas de esteganografia criadas, em especial, nos últimos 25 anos, tornando impraticável uma comparação minuciosa. A maior parte dos modelos e técnicas encontradas nas pesquisas realizadas são variações ou pequenas modificações de modelos mais antigos. Poucas variações e modificações encontradas apresentaram melhorias realmente significativas em relação aos trabalhos originais. Ademais, muitas técnicas propostas não eram suficientemente detalhadas ao ponto de serem reproduzidas em sua totalidade.

Alguns critérios foram aplicados na escolha das técnicas:

- Reconhecimento da técnica pela comunidade científica, isto é, ser citado ou utilizado por outros pesquisadores em artigos posteriores;
- Aplicável a imagens digitais, em especial, os formatos JPEG, PNG e BMP;
- Ser reproduzível, isto é, o trabalho deve ser suficientemente detalhado de tal forma que sua recodificação possa ser alcançada em plenitude;
- Uso de técnicas variadas. Procurou-se, aqui, exemplificar técnicas de diferentes características, tanto no domínio espacial (LSB), quanto no domínio da transformada (DCT e DWT);
- Alcançar uma escala mínima de robustez a ataques;
- Possuir uma capacidade de incorporação suficiente para suportar uma marca d'água em formato de imagem digital, onde esta deverá ter um tamanho mínimo que possa ser visualmente reconhecida após sua extração. Este aspecto é relevante pois muitas técnicas esteganográficas são aplicadas a textos escondidos em imagens, contudo uma imagem digital, para ser ocultada, irá ocupar um espaço consideravelmente maior que um texto plano.

### 4.1. MATERIAIS E MÉTODOS

#### 4.1.1. Materiais

Nas seções seguintes são descritos todos os recursos aplicados a este projeto.

#### *4.1.1.1.Máquina (Hardware)*

Foi utilizado como ferramenta de programação um Microcomputador portátil (notebook) marca ASUS modelo K45VM, dotado de um processador i7-3610QM e 8 GB de memória RAM.

Os softwares utilizados: Adobe Photoshop CS4 Extended (versão 11.0) e Matlab 2014a (versão 8.3.0.532). Todos os testes executados no sistema operacional Windows 7 Home Basic (versão 6.1, compilação 7601 SP1).

#### *4.1.1.2.Imagens de cobertura*

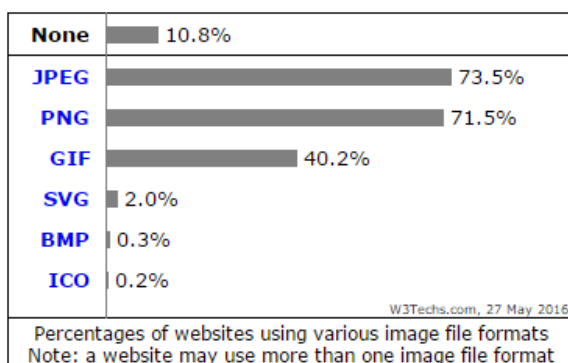
As imagens de cobertura escolhidas são clássicas para o uso da esteganografia, além de constarem como domínio público. O objetivo é facilitar comparações de desempenho com trabalhos existentes e futuros. As imagens são: Mandrill, também conhecida com Baboon; Lena e Pepper. As três imagens são coloridas e com uma dimensão de 1024 por 1024 pixels.

Todas as marcas d'água e imagens de cobertura estão reunidas no endereço: [https://drive.google.com/folderview?id=0B5H\\_thCYhxpZ21MMU80THRWTDg](https://drive.google.com/folderview?id=0B5H_thCYhxpZ21MMU80THRWTDg).

Os formatos de imagem escolhidos para este trabalho foram determinados, em especial, por sua utilização. A Figura 17 mostra a popularidade dos formatos mais comuns, exibindo o percentual de sites que utilizam cada formato de imagem. O formato JPEG é, o mais popular, presente em 73,5% dos endereços visitados, entretanto, o PNG vem galgando novas posições e pode ser considerado tecnicamente empatado com o formato JPEG. O GIF ainda é presente em 40% dos sites avaliados, contudo está sendo substituído gradativamente pelo PNG.

O formato BMP aparece com menos de 1% de uso, fator que contribuiu para o este tipo fosse deixado em segundo plano na realização deste trabalho.

Figura 17 - Percentual de uso de cada formato de imagem em sites da internet



Fonte: (W3TECHS, 2016)

A imagem Pepper (Figura 18) foi selecionada por possuir um número não grande de cores, largas áreas com poucos detalhes e de mesma cor, facilitando a verificação do impacto causado pela incorporação dos dados. Compreendida neste trabalho como uma imagem simples.

Figura 18 - Imagem “Pepper”



Fonte: (EL-EMAM; AL-ZUBIDY, 2013)

A imagem “Lena” (Figura 19) foi selecionada por ser a mais recorrente em trabalhos de esteganografia, possuir um número considerável de cores e mesclar grandes áreas de mesma intensidade com áreas detalhadas. Compreendida neste trabalho como uma imagem de média complexidade.

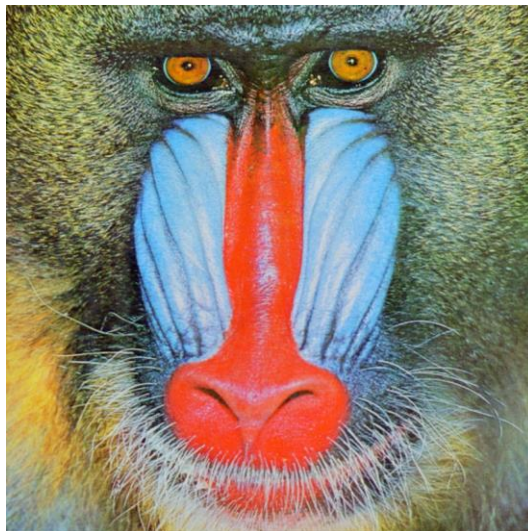
Figura 19 - Imagem “Lena”



Fonte: (EL-EMAM; AL-ZUBIDY, 2013)

A imagem “Mandrill”, mais conhecida como “Baboon” (Figura 20) foi selecionada por ser uma imagem complexa, com muitos detalhes e variações de cores. Útil na avaliação de impacto em áreas com muitos detalhes.

Figura 20 - Imagem “Baboon”



Fonte: (MOHD *et al.*, 2013)

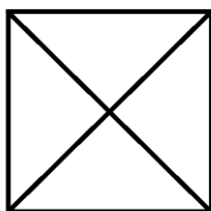
#### 4.1.1.1. Marcas d'água

As marcas d'água foram desenvolvidas com o intuito de simular um logotipo, marca pessoal ou assinatura. São compostas por imagens em tom de cinza e com baixa complexidade.

A recuperação da marca d'água não intenta ser perfeita, o objetivo principal é sua sobrevivência, ainda que em má qualidade, mas que permita reconhecer sua autenticidade. Mesmo quando a imagem recuperada é quase irreconhecível, certos padrões ou contornos podem ser observados, sendo, geralmente, suficientes para seu reconhecimento.

Assim, três marcas d'água foram criadas a partir do software Adobe Photoshop. O intuito é que estas tivessem uma complexidade gráfica crescente a fim de se observar seu comportamento quando submetido a diferentes tipos de imagem de cobertura. A Figura 21 mostra a marca mais simples, apenas um retângulo com um X em seu interior, sem curvas e com ângulos retos e agudos, nas cores preta e branca.

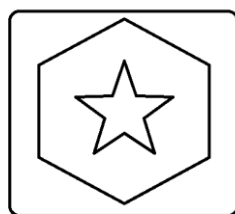
Figura 21 - Marca d'água simples



Fonte: Do Autor, 2015

A Figura 22, mescla três formas comuns: um retângulo de bordas arredondadas, um hexágono de contornos suaves e uma estrela de cinco pontas com ângulos agudos, toda a marca com apenas as cores branca e preta. É a representação de uma imagem de média complexidade.

Figura 22 - Marca d'água média

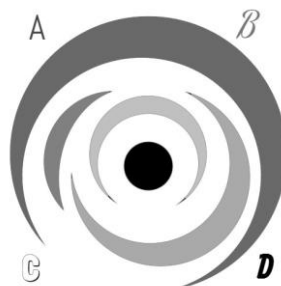


Fonte: Do Autor, 2015

A marca d'água mais complexa (Figura 23) mescla seis cores: branco, preto e quatro variações de cinza. Aplica várias curvas, ocupa mais o domínio espacial, faz uso de grandes e

pequenas áreas, também contém letras com fontes e cores e posições distintas para simular detalhes finos.

Figura 23 - Marca d'água complexa



Fonte: Do Autor, 2015

É oportuno observar que, mesmo quando a marca d'água é apresentada neste trabalho como “complexa”, esta parece simples quando comparada, por exemplo, a uma paisagem. Entretanto, a intenção é reconhecer a marca d'água extraída e associa-la a seu proprietário, o que é muito mais próximo de um logotipo ou assinatura do que de uma fotografia. Contudo, não existe impeditivo técnico à utilização de uma foto ou outra imagem complexa, mas detalhes finos podem ser perdidos ao ponto de impedir seu reconhecimento.

#### 4.1.2. Métodos

##### 4.1.2.1. Ataques

Uma série de ataques foi aplicado, tendo como guia principal o trabalho de Lusson *et al.* (2013). Os ataques foram selecionados para perturbar o domínio espacial e o domínio da transformada, sendo executados de maneira automática pelo software Matlab, com exceção do ataque Collage, que foi feito a partir do software Adobe Photoshop. O objetivo foi mostrar o nível de sobrevivência de cada marca d'água contra diferentes tipos de ataques, formatos e imagens de cobertura.

De fato, destruir a marca d'água é simples, bastando destruir a imagem portadora. Porém, a destruição da imagem portadora impede seu uso, assim, a intenção é manter viva a marca mesmo após os ataques, sendo que estes devem manter a imagem portadora intacta, ou muito próximo disso, conservando seu valor comercial.

Um ataque pode ter origem deliberada, uma tentativa direta de remover, falsificar ou substituir a marca d'água. Contudo, um ataque pode ser compreendido como qualquer tipo de ação que afete os dados da imagem, assim, mesmo que não intencional, um ataque pode ocorrer devido, por exemplo, ao *upload* de uma imagem a rede social, que por sua vez executa técnicas de compressão para mitigar o espaço ocupado pela mídia. Outro exemplo, é a aplicação de filtros, redimensionamento e mudanças de formato da imagem.

#### 4.1.2.1.1. Compressão JPEG

A necessidade de arquivos de tamanhos menores cresce à medida que seu volume aumenta. Técnicas para exibir e transferir imagens em um tempo aceitável foram sendo desenvolvidas para tornar factível sua utilização em larga escala.

Conforme abordado na seção sobre o formato JPEG, este é o formato mais amplamente utilizado e qualquer marca d'água deve ser resiliente a algum grau de compressão JPEG. A compressão com perda é definida pela tabela de quantização utilizada, mesmo com 0% de compressão, alguma informação é perdida no processo. Não há consenso do volume de compressão necessário para tornar uma imagem comercialmente inutilizável, pois dependerá de fatores como resolução, detalhes e utilização.

Para este trabalho, arbitrou-se compressões de 0%, 5%, 25%, 50% e 75%, isto é, um nível de qualidade de, respectivamente, 100%, 95%, 75%, 50% e 25%. O que mantém a qualidade geral da imagem com tamanhos de arquivo diminutos.

#### 4.1.2.1.2. Ruído

Segundo Solomon e Breckon (2013), o principal obstáculo ao eficiente processamento de imagens e de sinal é, em geral, o ruído. Ruído significa uma pequena variação (aleatória) sofrida pelo sinal em torno de seu valor verdadeiro, devido a fatores externos ou internos durante o processamento da imagem. Esses fatores não podem ser facilmente controlados e, portanto, introduzem elementos aleatórios ao processo.

Ainda segundo Solomon e Breckon (2013), ruído é o principal problema em 99% dos casos em que as técnicas de processamento de imagens falham ou processamento adicional se torna necessário para a obtenção do resultado desejado. Em consequência, uma grande parte do domínio do processamento de imagens e de qualquer sistema de processamento de imagens é

dedicada à redução e a remoção do ruído. Um sistema robusto deve ser capaz de lidar com ruído.

Neste trabalho, o ruído foi simulado pela técnica *salt and pepper* (sal e pimenta), que é causado pela introdução aleatória de pixels puramente branco ou preto (intensos/fracos) na imagem. Nos sensores modernos este tipo de ruído é menos comum, embora possa ser visto na forma de falhas em sensores de câmeras (pixels quentes, que sempre têm intensidade máxima ou pixels mortos, que são sempre pretos). Esse tipo de ruído também é conhecido como ruído impulsivo. A aplicação do ruído sal e pimenta foi realizado pelo software Matlab, através da função “imnoise”.

#### 4.1.2.1.3. Redimensionamento

Conforme Solomon e Breckon (2013), a translação (movimentação de um objeto de um lugar para o outro), a rotação e a mudança de escala (redimensionamento) de um objeto, são operações que mudam as coordenadas do vetor de forma do objeto, mas não mudam sua forma essencial.

O ataque de redimensionamento (*resizing*) é executado quando a imagem estego tem suas proporções geométricas aumentadas ou diminuídas. Este ataque pode ser deliberado ou não. Um exemplo de ataque deliberado é a redução da imagem em apenas 1 pixel, tanto em largura quanto altura, este processo não afeta visualmente a imagem, mas causa distorções no agrupamento dos pixels.

Este ataque foi realizado como segue:

- 1) A imagem é diminuída em relação percentualmente de maneira uniforme (10%, 25%, 50%, 75%, e 90%);
- 2) A imagem é salva em disco, garantindo que suas novas propriedades serão consideradas e aplicadas;
- 3) A imagem é lida do disco;
- 4) A imagem é redimensionada ao tamanho-alvo deste trabalho, isto é, 1024 x 1024 pixels.



#### 4.1.2.1.4. Rotação

Segundo Gonzalez e Woods (2010), as transformações geométricas (translação, rotação e mudança de escala), modificam a relação espacial entre os pixels da imagem. Essas transformações costumam ser chamadas de transformações do tipo *rubber sheet* (superfície de borracha), porque podem ser vistas de forma análoga à “impressão” de uma imagem em uma superfície de borracha que possa ser esticada de acordo com um conjunto de regras predefinidas.

Ainda conforme Gonzalez e Woods (2010), a rotação é uma das transformações mais exigente em termos de preservação das características de linhas retas. Lusson *et al.* (2013) afirma que uma rotação, no sentido horário ou anti-horário, mesmo que em um grau muito pequeno ( $0,1^\circ$ ), geralmente é suficiente para perturbar todos os pixels, de forma que o valor comercial da imagem pode ser mantido. As rotações aplicadas neste trabalho foram de  $0,1^\circ$ ,  $1^\circ$ ,  $-0,5^\circ$  e  $-1^\circ$ .

#### 4.1.2.1.5. Recorte

O ataque de recorte consiste em remover (*cropping*) partes da imagem a partir de suas bordas, mantendo o centro intacto. A Figura 24 mostra o resultado do ataque com uma intensidade de 50%.

Figura 24 - Image estego Lena após ataque do tipo Recorte (50%).



Fonte: Do Autor, 2015

É observável que uma parte considerável da imagem é perdida, mas assume-se que sua porção mais importante, o centro, geralmente o foco da imagem, se mantém íntegro. A parte recortada foi colocada em cor escura para permitir a visualização. Apesar de parte da imagem ter sido removida, sua dimensão permanece a mesma (1024 x 1024 pixels). Este ataque pode ser realizado de maneira simples com softwares como Adobe Photoshop, Macromedia Fireworks, Gimp ou mesmo Microsoft PaintBrush. Para este trabalho, o ataque do tipo recorte, foi aplicado com intensidade de 10%, 25% e 50%, isto é, “removendo-se”, respectivamente, 10%, 25% e 50% dos pixels a partir das bordas.

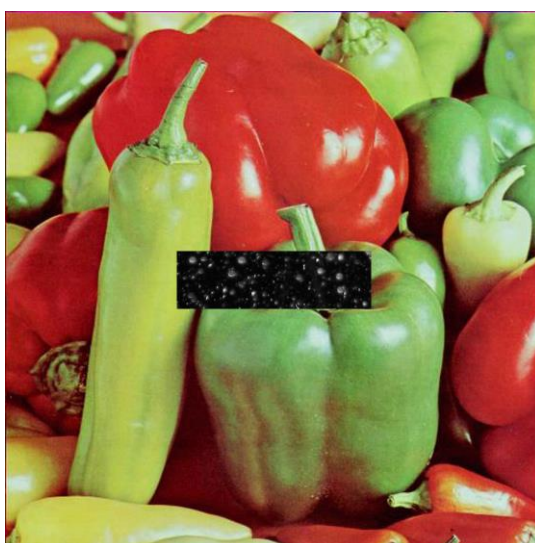
#### 4.1.2.1.6. Colagem

O ataque de colagem (*collage attack* ou *copy attack*) consiste em fazer uso de uma imagem externa “colando” seu conteúdo (ou uma parte de seu conteúdo) sobre a imagem estego.

A imagem utilizada foi a “*snowflakes.png*”, extraída do banco de imagens do software Matlab. Esta foi colada em três posições diferentes: seção superior, central e inferior da imagem estego. O processo foi realizado através do programa Adobe Photoshop, abrindo-se ambas imagens, selecionando e copiando o conteúdo completo da *snowflakes* e colando nas posições citadas por cima da imagem estego, posteriormente salvando-a no formato apropriado (JPEG, PNG ou BMP).

A Figura 25 mostra um exemplo do resultado.

Figura 25 - Uso do ataque de colagem na parte central da imagem Pepper

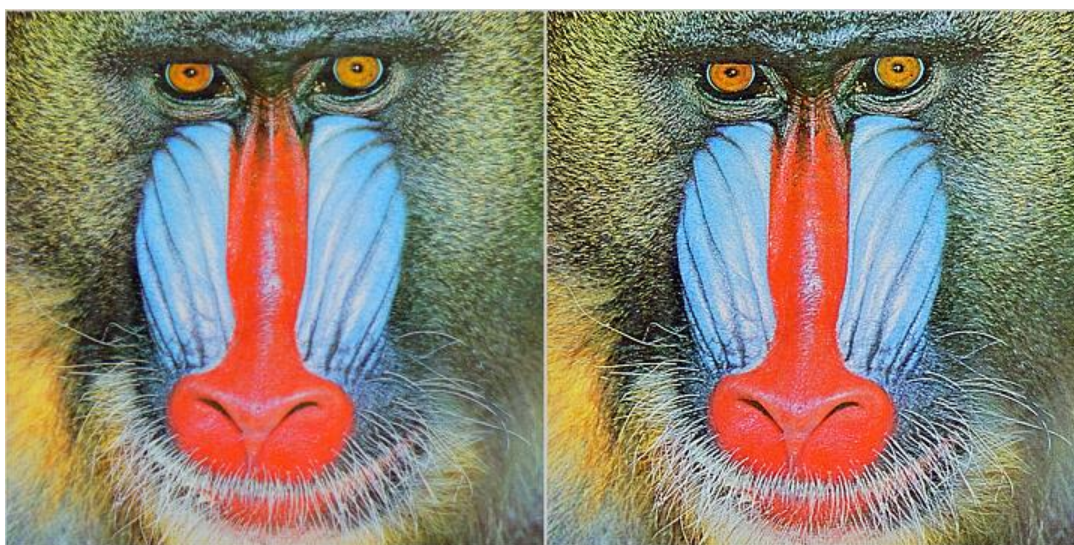


A imagem *snowflakes* foi utilizada em suas dimensões originais: 370 x 100 pixels. Isto é, 3,88% dos pixels da imagem estego foram destruídos. Contudo, a viabilidade comercial da imagem pode ser mantida a depender da localização e objeto da colagem. O ataque poderia ser utilizado, por exemplo, para esconder uma assinatura ou logomarca que identificasse o autor original da imagem.

#### 4.1.2.1.7. Nitidez

Segundo Petitcolas e Katzenbeisser (2000), o ataque de nitidez (*sharpen*) é uma melhoria de sinal. O comando utilizado foi *imsharpen* do software Matlab, com seus parâmetros sem alteração (padrão), isto é, não houve variações deste ataque. A Figura 26 ilustra a aplicação do comando de *imsharpen*.

Figura 26 - Imagem Baboon antes e depois do ataque sharpen



Fonte: Do Autor, 2015

O ataque de nitidez destaca certas características da imagem, em especial, as bordas. Seu funcionamento básico consiste na subtração da imagem turva (*blurred* ou *unsharp*) da imagem original, prática comum da indústria editorial.

#### 4.1.2.1.8. Corte

O ataque do tipo corte (*clip*) é semelhante ao ataque recorte (*crop*), contudo o *crop* literalmente recorta uma parte da imagem desprezando o restante, enquanto o *clip* remove uma seção específica da imagem original. A Figura 27 faz um paralelo entre as duas técnicas, que tem como objetivo final modificar a imagem estego.

Figura 27 - Diferença entre a técnica de corte (clipping) e recorte (cropping)



Fonte: Do Autor, 2015

Um total, aproximado, de 22% dos pixels da imagem foram destruídos. Testes foram realizados com o corte de mesma proporção da imagem estego nas porções superior e inferior.

#### 4.1.2.2. Qualidade da Imagem

Segundo Pedrini e Schwartz (2008), uma imagem pode sofrer degradações durante o processo de aquisição, transmissão ou processamento. Métricas de qualidade ou fidelidade podem ser utilizadas para avaliar a similaridade de uma imagem transformada em relação à original. Algumas medidas são voltadas a avaliações subjetivas, as quais se baseiam em análises realizadas por observadores humanos. Por outro lado, avaliações objetivas procuram medir a qualidade da imagem por meio de funções entre a imagem original e a imagem transformada. Os métodos de avaliação objetiva mais comuns são baseados em medidas de similaridade ou diferença entre as imagens.

Conforme Petitcolas e Katzenbeisser (2000), é um fato bem conhecido que as métricas de cálculo de distorção não são totalmente correlatas ao sistema visual humano (HSV), o que pode ser um problema, uma vez que muitas técnicas fazem uso da fraquezas no HSV.

De acordo com Pedrini e Schwartz (2008), o Erro Médio Quadrático (*Mean Square Root – MSE*) é a soma dos quadrados das diferenças de cada ponto da imagem original e da imagem aproximada, dividido pela multiplicação das dimensões da imagem, expresso pela equação (14):

$$MSE = \frac{1}{mn} \sum_0^{m-1} \sum_0^{n-1} \|f(i, j) - g(i, j)\|^2 \quad (13)$$

Quanto menor essa métrica, melhor a nova imagem se aproxima do original. Uma variação muito utilizada dessa métrica, e aplicada neste trabalho, é conhecida como Raiz do Erro Médio Quadrático (*Root Mean Square Error – RMSE*), expressa na equação (15):

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2} \quad (14)$$

Outra métrica aplicada a este trabalho é a Relação Sinal-ruído de Pico (*Peak Signal to Noise Ratio – PSNR*), utilizada para avaliar a diferença global entre duas imagens, expressa na equação (16):

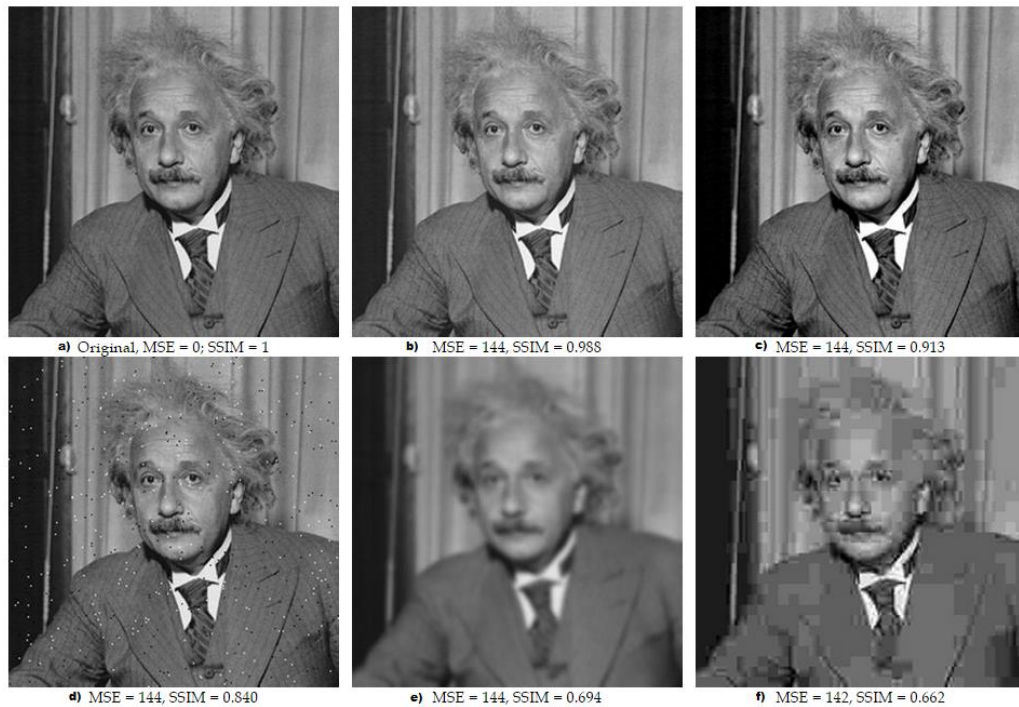
$$\begin{aligned} PSNR &= 10 \log_{10} \left( \frac{MAX_I}{\sqrt{MSE}} \right) \\ &= 20 \log_{10}(MAX_I) - 10 \log_{10}(MSE) \end{aligned} \quad (15)$$

Sendo MAX o valor máximo de intensidade de cinza. Tipicamente MAX = 255 para imagens representadas por 8 bits de profundidade. A métrica PSNR é expressa em decibel (dB), unidade originalmente definida para medir intensidade sonora em escala logarítmica. Valores típicos de PSNR variam entre 20 (para RMSE = 25,5) e 40 (para RMSE = 2,55). Quanto maior essa métrica, melhor a nova imagem se aproxima do original.

A última métrica aplicada neste trabalho é mais recente, chamada Similaridade Estrutural (*Structural Similarity – SSIM*). Nos últimos anos tem se mostrado uma alternativa viável a outras métricas e, em muitos casos, mais eficiente, entretanto, mais complexa (ZHOU *et al.*, 2004). O site de um dos criadores do SSIM, demonstra sua vantagem em uma série de fotografias, conforme Figura 28.



Figura 28 - Comparação entre as métricas MSE e SSIM



Fonte: (WANG, 2014)

A sequência mostra uma distorção severa da imagem, ainda que o MSE se mantenha o mesmo, neste caso, o SSIM apresenta um índice de qualidade da imagem mais confiável. Ademais, por ser expresso como índice, pode ser tratado como um percentual (%), sendo mais intuitivo e simples do que uma escala logarítmica, por exemplo.

Outro fato que não pode ser ignorado na qualidade da imagem é a dimensão. A dimensão diz ao tamanho da imagem e, apesar de por si só não representar qualidade, uma imagem de maior tamanho simplifica a observação de detalhes. Intrínseco a este conceito, está a capacidade de carga, isto é, o volume de dados possível a ser incorporado em uma imagem. Todas as imagens portadoras (Baboon, Lena e Pepper) foram trabalhadas na dimensão 1024 x 1024 pixels, enquanto as marcas d'água (simples, média e complexa), na dimensão de 512 x 512. Porém, o tamanho da marca d'água foi adaptado conforme o método utilizado, com esforço para que a proporção da marca d'água fosse mantida a mais próxima possível do original, sendo os resultados apresentados na Tabela 1.

Tabela 1 - Comparação da capacidade de incorporação de cada técnica

Algoritmo	Linhas em pixels	Colunas em pixels	Total em pixels
DCT AC	64	32	2048
DCT DC	96	64	6144
DCT AC DC	128	80	10240
LSB	768	512	393216
FUSION	256	128	32768
WDCT	48	32	1536

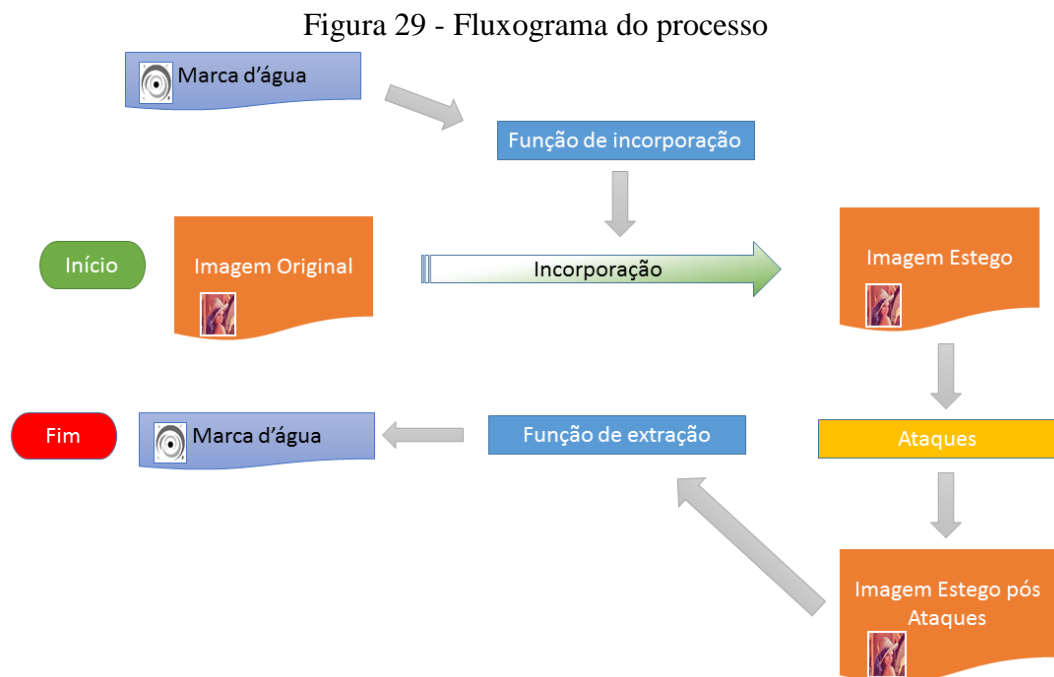
Fonte: Do Autor, 2015

O algoritmo de maior capacidade é o LSB, seguido por Fusion, DCT AC DC, DCT DC, DCT AC e WDCT, considerando a incorporação em uma imagem de 1024 x 1024 pixels.

#### 4.1.2.3. Algoritmos

Esta seção descreve os algoritmos utilizados, faz-se relevante informar que a maior parte destas técnicas não possui um nome formal, sendo, neste caso, arbitrado um nome ao procedimento apenas com o intuito de identificação dentro deste trabalho.

A Figura 29 ilustra graficamente o processo de incorporação, ataque e extração da marca d'água.



Fonte: Do Autor, 2015

Todos os algoritmos seguem a mesma estrutura básica, detalhada em alto nível a seguir:

- 1) Lê a imagem portadora (*cover*), nas dimensões 1024 x 1024 (em cores);
- 2) Lê o dado embutido (*embedded*), neste trabalho, uma imagem no formato (inicial) de 512 x 512. Caso a marca d'água seja em cores, é convertida em escala de cinza;
- 3) O dado embutido é redimensionado de forma a ocupar todo o espaço disponível da imagem portadora, com grande variação em cada procedimento;
- 4) Os parâmetros gerais são definidos, tais como: LSB, força de ataque, nível de compressão, canal RGB, etc.;
- 5) O dado embutido é convertido em formato binário sequencial e rearranjado em um array unidimensional;
- 6) A imagem original é armazenada tanto em variável quanto em disco para servir como parâmetro de comparação;
- 7) As informações são submetidas a função de incorporação (algoritmo esteganográfico);
- 8) A imagem estego é retornada com a informação secreta armazenada;
- 9) Compara-se a imagem estego com a imagem original a fim de determinar o grau de deturpação (RMSE, PSNR e SSIM), além disso também são criados os histogramas;
- 10) Aplica-se o ataque a imagem estego;
- 11) Escreve-se a imagem estego no disco;
- 12) Lê-se a imagem estego do disco;
- 13) A imagem estego é submetida ao algoritmo de extração. O retorno da função de extração é um array binário sequencial unidimensional;
- 14) O array é rearranjado de forma a tomar novamente as dimensões da imagem de entrada (*embedded*);
- 15) A marca d'água recuperada é salva como imagem no disco (em formato PNG para evitar perdas adicionais relacionadas a compressão).

#### 4.1.2.3.1. DCT AC

A primeira referência a este algoritmo é encontrada em Jian e Koch (1998), onde Zhao e Koch descrevem o uso da DCT como meio de utilização em marcas d'água digitais. Posteriormente, o algoritmo é aplicado, simplificado e apresentado por Petitcolas e



Katzenbeisser (2000), referência esta, utilizada neste trabalho. A seguir, a função de incorporação em alto nível do algoritmo DCT AC:

```

for i = 1,...,l(M) do
  choose one cover-block  $b_i$ 
   $B_i = D[b_i]$ 
  if  $m_i = 0$  then
    if  $B_i(u_1, v_1) > B_i(u_2, v_2)$  then
      swap  $B_i(u_1, v_1)$  and  $B_i(u_2, v_2)$ 
    end if
  else
    if  $B_i(u_1, v_1) < B_i(u_2, v_2)$  then
      swap  $B_i(u_1, v_1)$  and  $B_i(u_2, v_2)$ 
    end if
  end if
  adjust both values so that  $|B_i(u_1, v_1) - B_i(u_2, v_2)| > x$ 
   $b'_i = D^{-1}[B_i]$ 
end for
create stego-image out of all  $b'_i$ 

```

Esta função não recebe como parâmetro a imagem completa, mas apenas um canal RGB. Após vários testes em todos os canais e em múltiplas situações, o canal que apresentou o melhor equilíbrio entre robustez e baixa perturbação foi o azul. O primeiro passo, não mostrado no algoritmo anterior, é a divisão da imagem (apesar de utilizar o termo “imagem” neste contexto, de fato é apenas o canal *blue* da imagem) de cobertura em blocos de 8 por 8 pixels. Para cada bloco de pixels, é aplicada a DCT, denotada por  $B_i = D[b_i]$ . Na prática, a aplicação da DCT é realizada pela função “dct2” da ferramenta Matlab, que devolve os coeficientes em formato 8 x 8. O núcleo deste processo é a troca de coeficientes, regido pela seguinte diretiva: se o bit do dado a ser incorporado for 0, o coeficiente 2  $B_i(u_2, v_2)$  deve ser maior que o coeficiente 1  $B_i(u_1, v_1)$ , caso contrário os coeficientes  $c_2$  e  $c_1$  são trocados um pelo outro; se o bit do dado a ser incorporado for 1, o coeficiente 1  $B_i(u_1, v_1)$  deve ser maior que o coeficiente 2  $B_i(u_2, v_2)$ , caso contrário os coeficientes  $c_1$  e  $c_2$  são trocados um pelo outro.

A escolha dos coeficientes, que corresponde a funções cosseno, não é definida no trabalho de Zhao e Koch, mas Petitcolas e Katzenbeisser sugerem (4, 1) e (3, 2) ou (1, 2) e (3, 0). A escolha dos autores é baseada nas frequências médias, próximas em distância e de

valores similares entre si, em relação a tabela padrão de quantização do formato JPEG (Figura 30).

Figura 30 - Tabela padrão de quantização do formato JPEG

(u,v)	0	1	2	3	4	5	6	7
0	16	11	10	16	24	40	51	61
1	12	12	14	19	26	58	60	55
2	14	13	16	24	40	57	69	56
3	14	17	22	29	51	87	80	62
4	18	22	37	56	68	109	103	77
5	24	35	55	64	81	104	113	92
6	49	64	78	87	103	121	120	101
7	72	92	95	98	112	100	103	99

Fonte: (PETITCOLAS; KATZENBEISSER, 2000), adaptada pelo autor.

Após a etapa de trocas dos coeficientes, garante-se que  $|B_i(u_1, v_1) - B_i(u_2, v_2)| > x$ , para um  $x > 0$ , adicionando um valor determinado a ambos os coeficientes. Quanto maior o valor atribuído a  $x$ , mais robusto é o algoritmo contra ataques (incluindo a própria compressão JPEG), entretanto seu aumento gera deturpações maiores à imagem.

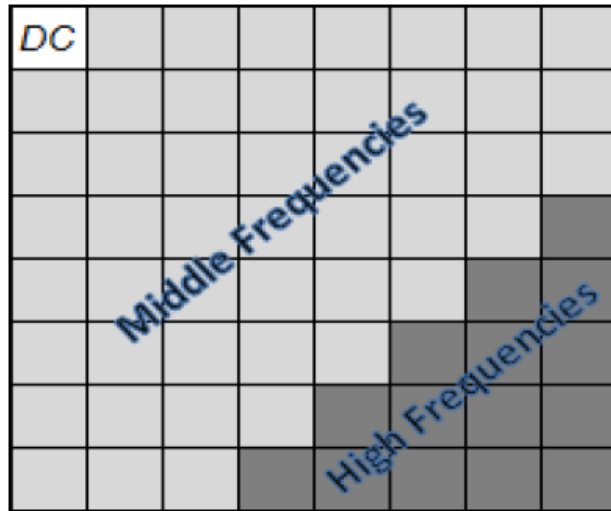
A última etapa é o retorno ao domínio espacial. O bloco com os coeficientes modificados é enviado como parâmetro para a função DCT Inversa. A função “idct2” do Matlab fica a cargo desta tarefa.

É importante observar que a escolha dos coeficientes e de  $x$  (doravante chamado de valor de persistência) não é necessariamente fixa, isto é, melhores resultados podem ser alcançados a depender da imagem de cobertura e o plano de cores. Assim, após vários testes e simulações, os termos mais equilibrados para os coeficientes foram os pares (4, 1) e (3, 2), em concordância com Petitcolas e Katzenbeisser, e 100 para o valor de persistência.

Dois tipos de coeficientes podem ser observados em cada bloco de 8 x 8, após a aplicação da DCT: DC e AC, (Figura 31). Os nomes DC e AC são originários da eletrônica. O valor superior esquerdo de bloco é o coeficiente DC, e contém o valor médio de todos coeficientes do bloco, ou seja, os coeficientes AC. O componente DC é muito importante, pois fornece uma estimativa apurada dos detalhes de cada bloco. Sua modificação causará

alterações em vários componentes AC, podendo criar uma discrepância no momento do retorno da imagem ao domínio espacial. Esta é a razão pela qual muitos algoritmos (como o popular JSteg) evitam sua modificação (SHEISI *et al.*, 2012). O nome DCT AC, aqui adotado, é em decorrência da modificação exclusiva dos componentes AC.

Figura 31 - Coeficientes DC e AC retornados da transformada DCT



Fonte: (SHEISI *et al.*, 2012)

A função de extração da informação é apresentada em alto nível abaixo. Recebe como parâmetro a imagem estego e, opcionalmente, o canal RGB utilizado na incorporação. Após a separação do canal, a imagem é segmentada em blocos de 8 x 8, cada bloco é submetido a aplicação da DCT (`dct2`). Os mesmos coeficientes aplicados na incorporação devem ser utilizados para extração, caso  $c1 B_i(u_1, v_1) < c2 B_i(u_2, v_2)$ , entende-se como bit 0, caso contrário, o valor do bit é 1. A sequência binária é armazenada em um array unidimensional, posteriormente, rearranjado e convertido para decimal, com intervalos de 0 a 255. Essa matriz é então convertida em imagem PNG e salva em disco.

```

for i = 1,...,l(M) do
  get cover-block bi associated with bit i
  Bi = D[bi]
  if Bi(u1, v1) ≤ Bi(u2, v2) then
    mi = 0
  else
    mi = 1
  end if
end for

```

#### 4.1.2.3.2. DCT DC

Este algoritmo guarda semelhanças com o modelo DCT AC, mudando o foco para o coeficiente DC em detrimento do coeficiente AC. Sua referência é encontrada no trabalho de Monika e Jasmine (2014). O algoritmo de incorporação DCT DC é apresentado a seguir.

Sender side:

Step-1: Write text message.(original message).

Step-2: Encrypt message using Blowfish algorithm.

Step-3: Select cover image.

Step-4: The cover image is broken into  $8 \times 8$  block of pixels.

Step-5: Use DCT to transform each block  $O_i$  into DCT coefficient matrix  $F_i[a,b]=DCT(O_i[a,b])$ .

Step-6: Calculate LSB of each DC coefficient and replace with each bit of secret message.

Step-7: Write stego image.

Este algoritmo foi originalmente apresentado para esteganografia e não marca d'água digital e, como foi proposto para trabalhar com texto, o modelo foi alterado para suportar imagens. Salienta-se que o presente trabalho não fez uso do algoritmo de criptografia Blowfish, pois o autor assumiu que este não era imprescindível ao processo, não afetando o resultado final, uma vez que o objetivo é a robustez e não a segurança. Ademais, não existem grandes empecilhos para que o processo seja alterado e passe a trabalhar com o algoritmo Blowfish, ou ainda outro procedimento criptográfico.

Diferentemente do algoritmo DCT AC, uma imagem completa, ou seja, com os três canais RGB, é enviada como parâmetro para a função. O próximo passo é a segmentação da imagem em blocos de  $8 \times 8$ , cada bloco é então separado por canais (RGB). Para cada bloco e canal é aplicada a DCT (`dct2` do Matlab), apresentando como retorno uma matriz de  $8 \times 8$  com os coeficientes da DCT. É selecionado o coeficiente DC e o LSB é substituído pelo bit do dado a ser incorporado, neste caso, a marca d'água. O último passo é a aplicação da DCT Inversa (`idct2` do Matlab) para retornar ao domínio espacial. A seguir, o pseudocódigo para extração da informação do algoritmo DCT DC (MONIKA; JASMINE, 2014):

Receiver side:

Step-1: Read the stego image.

Step-2: Divide the stego image into  $8 \times 8$  block of pixels.

Step-3: DCT is applied to each block.

Step-4: Calculate LSB of each DC coefficient.

Step-5: Decrypt message using Blowfish algorithm.

Step-6: Get original message.

Como anteriormente citado, o passo de decifração do algoritmo Blowfish não é aplicado, pois não foi utilizado no processo de incorporação. O trabalho de Monika e Jasmine (2014) não deixa explícito se o LSB aplicado foi efetivamente o bit menos significativo (mais à direita do octeto), assim, os primeiros testes foram realizados com o bit mais direita, com resultados insuficientes em relação a robustez. Dessa forma, entendendo que o valor do LSB é um parâmetro inerente ao processo, optou-se por aplicar testes com diferentes números, chegando ao ponto de equilíbrio para as imagens empregadas mediante o valor 5, isto é, o quinto bit menos significativo (ou o quarto MSB – *most significant bit*, 00000000). Este número apresentou-se permitindo melhor balanço entre robustez e perturbação da imagem.

A extração tem início com divisão da imagem em blocos  $8 \times 8$ . O segundo passo é separar os canais RGB. Para cada bloco e canal, é realizada a DCT (dct2 do Matlab), o coeficiente DC é lido da matriz e deste, o LSB é extraído (no caso, o LSB 5). O retorno é um array unidimensional de valores binários, que é colocado no formato de entrada da marca d'água, convertido para decimal e transformado em imagem.

#### 4.1.2.3.3. DCT AC DC

Os algoritmos apresentados anteriormente, DCT AC e DCT DC, se mostraram funcionais e com um grau de qualidade suficiente para permitir o reconhecimento das imagens extraídas. Durante a realização de testes, notou-se que o uso da DCT e a estrutura similar de ambos poderiam ser unidas formando uma única estrutura. Assim, o algoritmo DCT AC DC é a junção dos algoritmos DCT AC e DCT DC. Apesar de pesquisas recorrentes a respeito, não foram encontradas referências da existência deste tipo de técnica. O maior benefício dessa unificação foi o aumento da capacidade de incorporação dos dados.

Uma vez que a técnica não encontrou utilização na literatura, optou-se por aumentar o número de coeficientes de troca, afim de aumentar a capacidade de carga e, por consequência, a dimensão da marca d'água. Os coeficientes selecionados foram: (4, 1) e (3, 2); (2, 0) e (3, 0), respectivamente  $c_1$ ,  $c_2$ ,  $c_3$  e  $c_4$ . O algoritmo de incorporação segue a sequência:

- 1) Segmentação da imagem em blocos de 8 x 8;
- 2) Realização da DCT (dct2 do Matlab) para o canal especificado;
- 3) Troca dos coeficientes  $c_1$  e  $c_2$ , igualmente ao mesmo processo realizado no algoritmo DCT AC;
- 4) Garantir a diferença entre  $c_1$  e  $c_2$  por meio do valor de persistência (100);
- 5) Troca dos coeficientes  $c_3$  e  $c_4$ , igualmente ao mesmo processo realizado no algoritmo DCT AC com os coeficientes  $c_1$  e  $c_2$ ;
- 6) Aplicar a DCT Inversa (idct2 do Matlab) no bloco para o canal especificado;
- 7) Garantir a diferença entre  $c_3$  e  $c_4$  por meio do valor de persistência (100);
- 8) Para cada canal: realizar a DCT (dct2 do Matlab);
- 9) Extrair o componente DC;
- 10) Substituir o LSB (5) do componente DC pelo bit da marca d'água;
- 11) Aplicar a DCT Inversa (idct2 do Matlab) no bloco para todos os canais;

O processo de extração, não por acaso, trabalha da mesma forma que os algoritmos de extração DCT AC e DCT DC, seguindo as instruções a seguir:

- 1) Segmentação da imagem em blocos de 8 x 8;
- 2) Realização da DCT (dct2 do Matlab) para o canal especificado;
- 3) Se o valor do coeficiente  $c_1 > c_2$ , é assumido o bit 1, caso contrário, assume-se 0;
- 4) Se o valor do coeficiente  $c_3 > c_4$ , é assumido o bit 1, caso contrário, assume-se 0;
- 5) Para cada canal: realizar a DCT (dct2 do Matlab);
- 6) Extrair o componente DC;
- 7) Ler o LSB 5 do componente DC;

#### 4.1.2.3.4. LSB

O meio mais simples de ocultação de informação, por meio de uma sequência binária, é aplicando a substituição do bit menos significativo (LSB). Muitas variações deste algoritmo têm sido apresentadas ao longo dos anos, sendo que o processo clássico substitui o bit mais a direita pelo bit da informação a ser escondida. Variações aplicáveis: selecionar alguns bits com características específicas para uso, selecionar os bits a partir de um número pseudoaleatório, seleção dos bits em uma sequência pré-determinada, etc.

Para este trabalho, optou-se pelo uso do algoritmo LSB clássico. O processo de incorporação da mensagem é ilustrado nos passos a seguir:

- 1) Para cada canal RGB, linha e coluna;
- 2) Extraí-se o valor do pixel da imagem portadora;
- 3) Substitui-se o LSB 1 do pixel extraído da imagem portadora pelo bit da informação a ser ocultada;
- 4) O valor modificado substitui o original.

Destaca-se a simplicidade e objetividade da técnica, nota-se também que, por larga vantagem, o algoritmo LSB permite grande capacidade de carga, isto é, comparativamente as técnicas utilizadas neste trabalho, o LSB permite uma quantidade de dados muito maior.

A extração da informação é realizada com apenas um laço para cada canal, linha e coluna da imagem estego, extraíndo-se o bit menos significativo, neste caso específico, o LSB 1.

#### 4.1.2.3.5. Fusion

O algoritmo Fusion tem sua origem na transformada wavelet, e foi originalmente proposto por TOLBA *et al.* (2004). O conceito do algoritmo foi explicado pelo próprio autor, e segue livremente traduzido a seguir.

Um grande volume de técnicas foi proposto na literatura na última década com o objetivo de explorar características do Sistema Visual Humano (*Human Visual System – HVS*) para ocultação de dados. Deste ponto de vista, a transformada wavelet é uma ferramenta atrativa devida à similaridade intrínseca ao modelo teórico da percepção HSV. A ideia principal é chamada fusão baseada em wavelet (*wavelet based fusion*) e envolve a fusão da decomposição wavelet das versões normalizadas da imagem de cobertura e dado oculto em um resultado único. Em uma imagem normalizada, os componentes dos pixels residem no intervalo de 0 a 1, ao invés de 0 a 255. Os componentes wavelet correspondentes também apresentam variabilidade de 0 a 1.

O nome “Fusion”, aqui, não-oficialmente adotado, vem da referência a fusão, utilizada por Tolba. Segundo Kumar e Muttoo (2013), neste método, o processo de fusão acontece entre a transformada wavelet (DWT) dos dados a serem ocultados e a DWT da imagem de cobertura. Uma vez que os filtros wavelet regulares têm coeficientes em formato de ponto flutuante, um processo de normalização é aplicado na imagem portadora, convertendo-os ao intervalo de 0 a 1. A tão citada fusão acontece entre a decomposição wavelet da imagem de cobertura e o dado secreto por meio da equação:

$$f'(x,y) = f(x,y) + \alpha * g(xm,ym) \quad (16)$$

Sendo  $f'$  o coeficiente modificado da DWT;  $f$  é o coeficiente wavelet normalizado;  $g$  é o coeficiente normalizado da mensagem e o alfa é a força (fixador), que varia entre 0 e 1. O pré-processamento garante que os intervalos não sejam rompidos. A mensagem secreta é convertida ao formato binário, se o valor do bit é 1, soma-se o alfa, caso contrário, decrementa-se o alfa.

A função de incorporação segue descrita por Kumar e Muttoo (2013):

Algo: Fusion Embedding

Input: Cover image,I, original text,M, alpha, num, random-key,k

Output: the Stego-image, I'.

Step 1. Normalize the cover image,I. i.e., the pixel values made to lie between 0.0 and 1.0.

Step 2. Apply preprocessing on cover image: choose 'alpha' (preferably between 0 and 0.1) and reconstruct pixels to lie in the range [alpha, 1 – alpha]. This will ensure that pixels from the fused coefficients (during embedding) would not go out of range and hence the secret message will be recovered correctly.

Step 3. Apply 2D Haar transform/CDF97 transform on each color plane separately.

Step 4. Encode the original message,M, using the T-codes. The resulting secret message is a bit-stream of 0 and 1, denoted by (m1 m2.... mn), where n is the embedding message length and it generate an encoding key, K.

Step 5. Generate pseudorandom permutation, using a random-key,k, of the size equal to the length of cover image.

Step 6. Enter the number of times the message to be embedded, num.

for i = 1 to num do

6.1 Select wavelet coefficient of the transformed image randomly, say f(j, k)

6.2 Embed the secret message bit, m (i), into the transformed image in the following

way:

if m(i) = '1'

f (j,k) = f (j,k) + alpha;

else

f (j,k) = f (j,k) – alpha;

Step 7. Apply the inverse 2D Haar transform on each color plane separately.

Step 8. Denormalize the image, and obtain the stego-image, I'



O alfa pode variar entre 0 e 1, entretanto Kumar e Muttoo (2013) sugerem um intervalo menor, entre 0 e 0,1. De fato, durante os testes, valores acima de 0,1 originavam distorções visíveis na imagem e, chegou-se ao valor de 0,05 como melhor ponto de equilíbrio entre robustez e qualidade da imagem.

Esta técnica apresentou uma capacidade de carga bastante satisfatória para o uso de marca d'água digital. Na verdade, foi o algoritmo com maior capacidade de incorporação, perdendo apenas para o LSB. Considerando este fato, sugere que o algoritmo seja aplicado em todos os canais de cores, entretanto, além de aumentar consideravelmente a distorção da imagem, a quantidade de bits utilizável já é aceitável em apenas um canal (*blue*, o canal que se sobressaiu nos testes). Outro detalhe alterado é em relação a não utilização de chave criptográfica, ou seja, *random-key* e *T-code*.

A primeira etapa do processo é a normalização, que é aplicada pela divisão de cada pixel por 255 (valor máximo). A segunda etapa, chamada de pré-processamento, limita os valores normalizados ao intervalo de alfa a (1-alfa). A terceira etapa é aplicação da transformada wavelet 2D, tecnicamente falando, utilizando a função *dwt2* do Matlab, atribuindo como meio (*mode*) a técnica Haar ('*haar*'). O quarto passo é separar os coeficientes LL, LH, HL e HH. O quinto passo é percorrer cada coeficiente garantindo a seguinte regra: se o bit a ser inserido tem valor 1, o coeficiente será incrementado com alfa, caso contrário (bit 0), o alfa é subtraído do coeficiente. A última etapa é a aplicação da transformada wavelet 2d inversa (*idwt2*), mantendo os coeficientes inalterados (com exceção de LL).

O ponto fraco deste algoritmo se apresenta na recuperação da informação, pois exige a imagem original como entrada além da imagem estego. A função de extração do algoritmo Fusion segue a seguir:

Input: Stego-image,  $I'$ , random-key,  $k$ , encoding-key,  $K$

Output: Original message,  $M$ .

Step 1. Apply 2D Haar transform/CDF97 on each color plane of the stego-image,  $I'$

Step 2. Enter num, number of times message being embedded

Step 3. Initialize the hiddenmessage to zero.

for  $j= 1$  to num do

3.1 Select the embedded coefficients,  $i$ , using the PRNG based on the random-key,  $k$ , same as used in the embedding procedure.

3.2 Extract the embedded value of alpha by subtracting the original cover image from the stego image in the wavelet domain

3.3 Obtain the secret message bit,  $m(i)$  as follows:

If  $\alpha > 0$

```

        m(i) = '1'
    else
        m(i)='0';
    3.4 hiddenmessage += m(i);
end; /*for(j)*/
Step 4. hiddenmessage /=num;
Step 5. Decode the hiddenmessage using T-decoding algorithm using the encoded-
key, K.

```

O primeiro passo é aplicar a transformada wavelet 2D (dwt2) em ambas as imagens: original e estego. Extraí-se os coeficientes desejados, neste caso, LL. A próxima etapa é subtrair a matriz LL da imagem estego da matriz LL da imagem original, armazenando-a em uma matriz de diferenças. A última etapa é percorrer a matriz de diferenças analisando se, caso o valor do coeficiente seja positivo, assume-se que o bit em oculto é 1, caso contrário (menor ou igual a zero), assume-se que o bit é 0.

#### 4.1.2.3.6. WDCT

O algoritmo nomeado WDCT é a utilização de duas técnicas no domínio da transformada: DCT e Wavelet. A iniciativa não é inédita, havendo, por exemplo, referências nos trabalhos de Kumar e Kumar (2010) e Vaghela *et al.* (2012).

A seção DCT segue princípio semelhante ao utilizado no algoritmo DCT AC, a diferença é o emprego dos três canais (RGB) em detrimento de apenas um, aumentando a capacidade de carga.

A função de incorporação segue os passos a seguir:

- 1) Aplicação da transformada wavelet 2D (dwt2) a imagem de cobertura;
- 2) Separação da banda LH;
- 3) Aplicação do algoritmo DCT AC nos três planos de cores da banda selecionada;
- 4) Aplicação da transformada wavelet 2D inversa (idwt2);

A banda LH apresentou o melhor equilíbrio entre robustez e qualidade da imagem. De fato, o uso desta técnica apresentou um ótimo resultado em relação a não degradação da imagem. Além disso, o índice de semelhança (SSIM) ficou acima de 99% quando comparada a imagem original contra a imagem estego.

A função de extração segue o mesmo princípio da função de extração do algoritmo DCT AC, com a diferença que esta é aplicada a banda wavelet LH e são utilizados os três planos de cores da imagem (RGB).

## 4.2. RESULTADOS







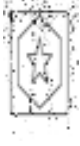





Para avaliar a robustez geral da marca d'água inserida, os algoritmos foram submetidos a oito tipos de ataques (e suas variâncias), em três tipos de formato de imagem, com três diferentes imagens de cobertura e três tipos de marcas d'água.

Os formatos utilizados foram JPEG, PNG e BMP. Entretanto, após testes e comparações, concluiu-se que os resultados obtidos com o formato PNG eram semelhantes aos obtidos com o formato BMP, para a melhor acomodação visual neste trabalho, optou-se por não incluir aqui os resultados do formato BMP. De maneira geral, os resultados obtidos com o formato PNG podem ser estendidos para o formato BMP.

### 4.2.1. Resultados da compressão JPEG

O Quadro 6 compara as imagens recuperadas para cada técnica, imagem de cobertura e marca d'água para o formato JPEG submetido a 25% de compressão. Este ataque não se aplica aos formatos PNG e BMP pois estes fazem uso de algoritmos de compressão sem perda (*lossless*).

Quadro 6 - Resultado da extração da marca d'água após a compressão JPEG em 25%

Compressão 25%		DCT AC	DCT DC	DCT ACDC	LSB	FUSION	WDCT
Baboon	Simple						
	Média						

Compressão 25%		DCT AC	DCT DC	DCT ACDC	LSB	FUSION	WDCT
	Complexa						
Lena	Simples						
	Média						
	Complexa						
Pepper	Simples						
	Média						
	Complexa						

Fonte: Do Autor, 2015

Os vários níveis de compressão JPEG se mostraram severos, causando danos consideráveis a estrutura da imagem, ainda que visualmente a correlação de degradação não seja evidente. Contudo, o Quadro 6 demonstra que, com exceção do algoritmo LSB, todas as técnicas, em algum momento, apresentaram traços da marca d'água.

Para os algoritmos DCT DC e DCT AC DC, a marca d'água complexa é a única que pode ser observada, ainda que seja apenas alguns resquícios, possivelmente esta sobreviveu por suas grandes áreas escuras, enquanto as marcas simples e média eram formadas apenas por linhas finas.

Entende-se que para o algoritmo DCT AC, Fusion e WDCT, em todas as situações (Quadro 6) a marca d'água sobreviveu. Para a DCT AC existe forte presença de ruído, em especial na imagem Pepper. Para o algoritmo Fusion, existe um forte ruído de fundo na incorporação da imagem Babbon, também apresenta uma certa repetição da marca d'água em decorrência do uso da transformada wavelet. Para a técnica WDCT o ruído geral foi menor, a imagem recuperada está mais clara e com contornos visíveis, contudo a imagem se repetiu três vezes e tanto a parte superior quanto a inferior foram cortadas, por exemplo, a marca d'água complexa perdeu as letras que ficavam no canto superior e inferior.

#### **4.2.2. Resultado do ataque ruído**




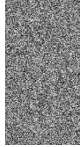


















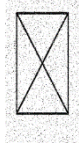

















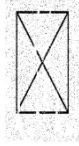













Várias intensidades foram aplicadas, sendo que as imagens se mostraram sensíveis a sua utilização. Entretanto, uma aplicação mais densa do ruído, pode causar artefatos visíveis na imagem, em especial, em largas áreas de mesma intensidade. Os Quadros 7 e 8 mostram o resultado da extração da marca d'água após o ataque, com uma intensidade de 2%, isto é, na dimensão em questão (1024 x 1024 pixels), aproximadamente 21.000 pixels foram afetados de maneira randômica. No formato JPEG, este tipo de ataque é particularmente agressivo, uma vez que seu algoritmo faz uso dos pixels vizinhos para determinados cálculos, o que termina por “espalhar” com mais rigor a deturpação inserida. O Quadro 7 compara as imagens recuperadas para cada técnica, imagem de cobertura e marca d'água para o formato JPEG submetido ao ataque sal e pimenta em uma intensidade geral de 2%.

O ataque com ruído foi severo para o formato JPEG, com exceção do algoritmo Fusion, todas as imagens recuperadas obtiveram resultados de baixa qualidade. Os algoritmos DCT AC e WDCT sofreram forte queda de qualidade em relação ao Quadro 6 (compressão JPEG), essa degradação conjunta é sempre esperada, uma vez que o WDCT faz uso da DCT AC. Também se observa que a marca de complexidade média é a mais difícil de ser reconhecida.

Mesmo causando danos severos, resquícios da marca d'água complexa são observados em todas as técnicas, inclusive, surpreendentemente, na LSB.

O Quadro 8 exhibe o resultado da aplicação do ruído em 2% de intensidade para o formato PNG.




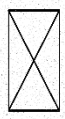

















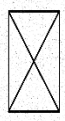

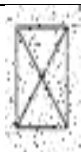















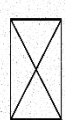














Quadro 7 - Resultado da extração da marca d'água em formato JPEG após ataque com ruído

JPEG/Ruído 2%		DCT AC	DCT DC	DCT AC DC	LSB	FUSION	WDCT
Baboon	Simple						
	Média						
	Complexa						
Lena	Simple						
	Média						
	Complexa						
Pepper	Simple						
	Média						
	Complexa						

Fonte: Do Autor, 2015



Quadro 8 - Resultado da extração da marca d'água em formato PNG após ataque com ruído

PNG/Ruído 2%		DCT AC	DCT DC	DCT AC DC	LSB	FUSION	WDCT
Baboon	Simple						
	Média						
	Complexa						
Lena	Simple						
	Média						
	Complexa						
Pepper	Simple						
	Média						
	Complexa						

Fonte: Do Autor, 2015