**Universidade de Brasília**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

# Mitigação de Erros utilizando Técnicas de Marca d'água e Halftoning

Matheus Lima da Rocha Pitta

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof.ª Dr.ª Mylène Christine Queiroz de Farias

Brasília
2011

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Marcus Vinicius Lamar

Banca examinadora composta por:

Prof.ª Dr.ª Mylène Christine Queiroz de Farias (Orientador) — CIC/UnB
Prof. Dr. Bruno Luiggi Macchiavello Espinoza — CIC/UnB
Prof. Dr. Alexandre Zaghetto — CIC/UnB

**CIP — Catalogação Internacional na Publicação**

Rocha Pitta, Matheus Lima da.

Mitigação de Erros utilizando Técnicas de Marca d'água e Halftoning / Matheus Lima da Rocha Pitta. Brasília : UnB, 2011.
113 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2011.

1. QIM, 2. Marca d'água, 3. Mitigação de Erros, 4. Halftoning

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

**Universidade de Brasília**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

# Mitigação de Erros utilizando Técnicas de Marca d'água e Halftoning

Matheus Lima da Rocha Pitta

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Mylène Christine Queiroz de Farias (Orientador)
CIC/UnB

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
CIC/UnB

Prof. Dr. Alexandre Zaghetto
CIC/UnB

Prof. Dr. Marcus Vinicius Lamar
Coordenador do Bacharelado em Ciência da Computação

Brasília, 1 de maio de 2011

# Dedicatória

Dedico este trabalho a minha família, pela paciência e tudo mais.

# Agradecimentos

Primeiramente tenho que agredecer a minha orientadora, a professora Mylène, por todo o trabalho (que não foi pouco) e pelas reuniões semanais. Em seguida agradeço a Ronaldo Rigoni e a Pedro Garcia, sem os quais esse trabalho não teria tido os resultados que obteve.

# Resumo

O interesse por técnicas de controle e mitigação de erros tem se tornado cada vez mais populares devido a crescente demanda por aplicações que utilizam transmissão de vídeo por canais com baixa confiabilidade, como por exemplo a Internet. Esse trabalho propõe um algoritmo para mitigação de erros que utiliza técnicas de marcas d'água e halftoning. No codificador, utilizamos um algoritmo inspirado no algoritmo de modulação do índice de quantização (QIM) para inserir uma cópia binária de cada quadro do vídeo nele mesmo. No decodificador, esta cópia (marca binária) é extraída e utilizada para reconstruir as partes do vídeo perdidas. O algoritmo apresenta bom desempenho conseguindo restaurar com boa qualidade vídeos com até 50% de perdas.

**Palavras-chave:** QIM, Marca d'água, Mitigação de Erros, Halftoning

# Abstract

Interest for error control and concealment techniques is becoming more and more popular due to increasing use of video transmission applications over low reliabilty networks, such as the Internet. In this work, an error concealment technique using watermarking and halftoning techniques is proposed. At the encoder we use an algorithm that was inspired from the quantization index modulation algorithm (QIM) to insert a binary copy of each frame in itself. At the decoder, this copy (binary mark) is extracted and used to reconstruct lost parts of the video. The algorithm is capable of restoring, with good visual quality, videos with up to 50% losses.

**Keywords:** QIM, Watermarking, Error Concealment, Halftoning

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

A forma mais comum de distribuição de mídia digital hoje em dia é através de redes, a mais famosa destas sendo a Internet. Como a Internet é uma rede de melhor esforço ela não possui garantias e erros acontecem ocasionalmente. No caso específico de vídeo este problema é agravado pelo fato dessas mídias serem normalmente comprimidas, o que significa que, uns poucos bits perdidos podem se propagar e tornar o resultado inutilizável.

Para se obter um vídeo resistente a erros diversas técnicas podem ser utillizadas. Estas podem ser divididas em duas categorias, controle de erros (*error control*) e mitigação de erros (*error concealment*). Peng Yin *et al.* [37] definem *error control* como a recuperação sem perdas. *Error concealment* é o processo de correção e mitigação de perdas em erros, utilizando informações espaciais e temporais do sinal para obter uma aproximação do sinal original de forma que estas perdas sejam pouco perceptiveis.

Uma das formas de implementação utilizadas pelas técnicas de mitigação de erros consiste em adicionar algum tipo de informação adicional ou redundante ao sinal original de forma a permitir que se perdas acontecerem, as partes do sinal perdido possam ser estimadas utilizando essa informação. Uma forma de se inserir essa informação adicional é utilizando marcas d'águas (*watermark*). O trabalho de Chowdary *et al.* [2] utiliza técnicas de inserção de marcas d'água para adicionar ao vídeo a informação necessária para a mitigação de erros ser possível.

Marca d'água é um método de inserir informações em um meio, muito utilizado, em aplicações de segurança. As marcas d'água podem ser classificadas como: visíveis, sendo que neste caso a marca inserida é normalmente um logo ou uma frase, ou invisíveis, na qual a informação inserida não é visualmente perceptível.

Marcas invisíveis podem ser usadas com vários propósitos, um dos quais seria esteganografia, onde o propósito da marca d'água é enviar informações secretas usando o sinal digital como hospedeiro. Uma possível aplicação seria, para fins de proteção de direitos autorais, permitir que um aparelho de cópia faça a decisão de copiar ou não um conteúdo verificando se a marca d'água está presente. Outra possível aplicação seria na solução de disputa sobre a propriedade de um arquivo que pode ser resolvida após a confirmação da presença da marca.

Existem vários métodos que podem ser utilizados para marcar digitalmente um arquivo, o mais simples deles é a modulação LSB (*least significant bit* ou bit menos significativo), na qual os bits menos significativos são alterados de forma a conter a marca desejada. Esse método possui uma fraqueza grande, que é a falta de robustez. Qualquer

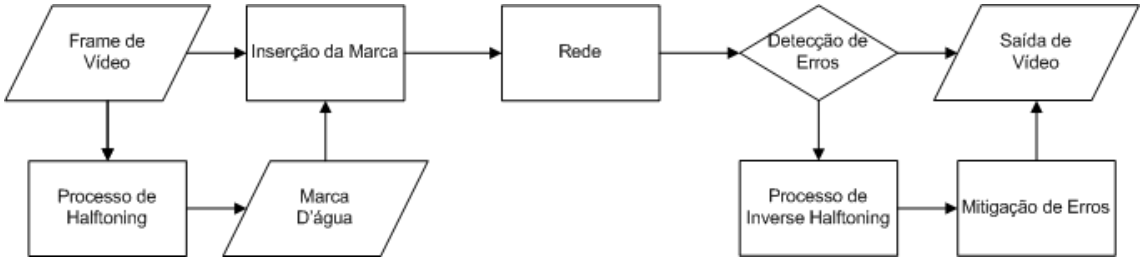



Figura 1.1: Diagrama de Blocos do Algoritmo Proposto.

ataque ao sinal, provavelmente, vai alterar o bit menos significativo, eliminando completamente a marca.

Em geral, quando se projeta um sistema de inserção de dados é desejável ter uma alta capacidade de inserção, uma alta robustez e um baixo nível de distorção visível (causada pela inserção). O problema é que essas características são conflitantes, logo um compromisso deve ser atingido.

Na última década o método de inserção por modulação do índice de quantização, (do inglês *quantization index modulation*, QIM) tem chamado bastante atenção da comunidade. Proposto por Chen e Wornell [6], QIM é um método robusto e com alta capacidade de inserção, mostrando melhor desempenho que outros métodos populares, como por exemplo o LSB e o espalhamento espectral. No nosso trabalho usamos um algoritmo inspirado no QIM, proposto nesse trabalho e descrito na Seção 4. Seguimos uma abordagem semelhante à proposta por Chowdary *et al.* [2], na qual a marca a ser inserida em cada quadro do vídeo é uma versão binária do próprio quadro, convertida usando técnicas de *halftoning*.

Na Figura 1.1 é apresentado o diagrama de blocos do algoritmo proposto nesse trabalho. Para cada quadro do vídeo original (isto é, sem erros e sem marca) obtém-se uma versão binária do próprio quadro usando a técnica de *halftoning*. Esta versão binária do quadro será a marca utilizada pelo algoritmo de inserção de marcas d'águas. Ou seja, uma versão de menor qualidade é inserida no próprio quadro. Esse processo provoca um certo nível de distorção no quadro original. Em seguida, este vídeo marcado é enviado por um canal potencialmente ruidoso. No receptor, o vídeo é decodificado. Caso algum erro seja detectado, extrai-se a marca inserida e aplica-se um processo de inversão do *halftoning* à marca extraída para recompor o sinal na região afetada.

O documento está dividido da seguinte forma. No Capítulo 2, descrevemos os algoritmos de marcas d'água e de criação de *halftoning*. No Capítulo 3, descrevemos as técnicas de mitigação de erros. No Capítulo 4, descrevemos o algoritmo proposto neste trabalho. No Capítulo 5, apresentamos os resultados obtidos. Finalmente no Capítulo 6, apresentamos as conclusões e trabalhos futuros.

# Capítulo 2

# Marcas D'água e Halftoning

Neste capítulo, descrevemos primeiramente o que são marcas d'água e algumas técnicas de inserção de marcas d'água. Em seguida, descrevemos a técnica que inspirou o algoritmo proposto, o QIM. Finalmente, descrevemos o processo de *halftoning* e seu processo inverso, o *inverse halftoning*.

## 2.1 Algoritmos de Inserção de Marcas D'água

O uso de técnicas para se enviar secretamente informação é antigo. Heródoto, em seu livro Histórias descreve um grego que tatuou a cabeça raspada de um escravo e enviou a um colega seu, com instruções de raspar novamente a sua cabeça para recuperar a informação. Hoje, o envio de informações se dá principalmente na forma digital. A distribuição digital tem várias vantagens bem conhecidas, como por exemplo transmissão sem ruídos, o processamento por software (em vez de hardware) e a reconfigurabilidade de sistemas. Já as desvantagens não são tão bem conhecidas. Do ponto de vista de produtores e distribuidores de conteúdo digital, a possibilidade de obtenção de uma cópia fiel sem a necessidade de comprar o produto novamente é indesejável. Métodos de proteção ou prevenção de cópia são de pouca utilidade pois uma vez que a um usuário legítimo, ou seja, que pagou pelo serviço, esteja com a mídia em mãos, este pode fazer quantas cópias quiser [12].

Uma das soluções para garantir os direitos autorais de conteúdos digitais consiste na utilização de marcas d'água. Marcas d'água são um código digital robusto, imperceptível e não-removível que é comumente inserido dentro dos dados de origem. Quando utilizados com o propósito de proteção de informação intelectual, a marca d'água contém dados sobre o conteúdo original, o estado ou destino dos dados [25]. Apesar de ser a aplicação mais popular das técnicas de marcas d'água, a proteção de direitos autorais não é a única possível aplicação. O método proposto nesse trabalho usa marcas d'água para mitigar erros ocorridos entre a transmissão (e geração) do sinal e a sua recepção e decodificação pelo usuário. Estes erros podem ocorrer por uma variedade de razões, entre as quais podemos citar ataques ou perdas na transmissão.

O projeto de marcas d'água envolve uma série de requisitos conflitantes. As marcas devem ser robustas contra manipulações comuns de dados, incluindo conversões analógico-digital e mudanças entre formatos. Também deve-se ter uma preocupação com segurança, ou seja, as marcas devem resistir a ataques de indivíduos "maliciosos" que tentem destruir

a marca. Por outro lado, marcas devem ser imperceptíveis visualmente e conter a maior quantidade de informação possível. Mais ainda, os métodos de inserção e recuperação das marcas devem ter baixa complexidade computacional, uma vez que muitas aplicações requerem processamento em tempo real.

Técnicas de inserção de marcas d'água podem ser aplicadas para proteger diversos tipos de mídias incluindo texto, áudio, imagens e vídeo. As técnicas podem ser classificadas como visíveis ou invisíveis. As marcas invisíveis são as mais usadas. Além da visibilidade, as marcas também podem ser classificadas de acordo com sua robustez nas seguintes categorias: robusta, semi-frágil e frágil. As marcas frágeis, por exemplo, são utilizadas para comprovar a ausência de adulterações em arquivos, dado que uma pequena alteração é suficiente para remover a marca. As marcas robustas, por outro lado, são projetadas para resistir a vários tipos de ataques, de forma a proteger a marca d'água. E as marcas semi-frágeis, são utilizadas para proteger contra apenas alguns tipos de ataque, falhando contra outros. Estas marcas servem para proteger o conteúdo contra tipos específicos de ataques. Outra forma de categorizar as técnicas de inserção de marca d'águas é pelo domínio no qual a marca é inserida. Ou seja, as marcas podem ser inserida no domínio espacial, no domínio da frequência, no domínio de compressão (*bitstream*) ou em um domínio híbrido. Por fim, podemos classificar as técnicas pelo método de extração da marca como público, privado e semi-privado. Neste trabalho, utilizaremos uma marca invisível, robusta, no domínio espacial e com chave pública. Nas próxima seção descrevemos algumas técnicas de inserção de marcas d'água, discutindo o seu desempenho.

### 2.1.1 Técnicas no Domínio Espacial

Nessas técnicas a inserção da marca é feita no domínio espacial, ou seja, alterando diretamente os valores dos pixels das imagens. Um dos exemplos mais simples deste tipo de abordagem é o algoritmo de inserção de marca d'água no LSB, já referenciado na introdução.

A Figura 2.1 mostra uma sequência de imagens nas quais um dos planos de bit foi completamente removido. A Figura 2.1a tem os bits menos significativos de cada cor zerados, a Figura 2.1b tem os bits do segundo plano de bits zerados e assim por diante, até a Figura 2.1h onde somente os bits mais significativos de cada cor foram zerados. Pode-se observar nestas imagens que os planos de bits de menor ordem podem ser alterados sem perda considerável da qualidade subjetiva.

Uma das técnicas de inserção de marcas d'água mais populares é a de espalhamento espectral [8]. Nesta técnica a informação é inserida combinando linearmente o sinal original com um ruído pseudo-aleatório, conforme vemos na equação a seguir:

$$S(x, y) = I(x, y) + k \times W(x, y), \tag{2.1}$$

em que $W(x, y)$ é o sinal ruidoso contendo a informação a ser inserida, $I(x, y)$ é o sinal original e $k$ é um fator de ganho. Aumentar este fator de ganho implica em maior robustez da marca d'água a um custo de menor qualidade do sinal resultante $S(x, y)$. Para recuperar a informação inserida gera-se novamente o sinal ruidoso e calcula-se a correlação entre o sinal marcado e o ruído. Se essa correlação for maior que um limiar $T$ a marca provavelmente está presente no sinal. Desta forma, recuperamos a informação binária, correspondente à existência ou à inexistência da marca no sinal. De forma a ser

(a) $1^a$ camada de bits (b) $2^a$ camada de bits (c) $3^a$ camada de bits (d) $4^a$ camada de bits

(e) $5^a$ camada de bits (f) $6^a$ camada de bits (g) $7^a$ camada de bits (h) $8^a$ camada de bits

Figura 2.1: Imagem mostrando a remoção de planos de bit.

possível inserir (e extrair) uma quantidade maior de dados, pode-se utilizar técnicas como a proposta por Chan Pik-Wah [26]. Nesta técnica, a imagem é dividida em blocos e o procedimento de espalhamento espectral é repetido individualmente para cada bloco.

Os métodos de inserção no domínio espacial possuem algumas características em comum. Entre estas, podemos citar

- A marca é aplicada diretamente nos pixels;
- Nenhuma trasformada é aplicada ao sinal;
- O sinal é combinado com a mensagem, em geral, com operações simples.

A maior vantagem deste tipo de método é a simplicidade, fundamental para aplicações em tempo real. A desvantagen, para o caso de aplicações em vídeo, são a baixa robustez (contra ataques de desincronização e processamento de vídeo) e uma dificuldade de otimizar a quantidade de informação inserível.

### 2.1.2 Técnicas no Domínio da Frequência

Uma outra categoria de métodos de inserção de marcas d'água são os métodos de inserção no domínio da frequência. Nesse grupo, insere-se a informação no domínio da frequência com o objetivo de dificultar a sua remoção.

Para se converter a imagem do domínio espacial para o domínio da frequência várias transformadas podem ser usadas. Podemos citar técnicas usando a transformada discreta de Fourier [4], a transformada discreta do cosseno (do inglês, *discrete cossine transform* DCT) [18] ou a transformada discreta de Wavelets (do inglês, *discrete wavelet transform* DWT) [32]. A principal vantagem dessa categoria de métodos são as características especiais dos domínios alternativos para combater as limitações dos métodos do domínio espacial. A grande desvantagem é a maior complexidade.

Uma das transformadas mais usadas pelas técnicas de inserção de marcas d'água é a DCT. A DCT permite quebrar a imagem em diferente bandas de frequência, tornando

muito mais fácil inserir informação na banda de frequências médias de uma imagem. Usam-se as frequências médias para evitar que as partes visualmente importantes da imagem (as baixas frequências) sejam degradadas. Ao mesmo tempo, também são preservadas as altas frequências que são regiões severamente afetadas pelos algoritmos de compressão[5]. Outra transformada frequentemente utilizada é a DWT. A vantagem da DWT em comparação com as outras transformadas, é que ela modela melhor vários aspectos do sistema visual humano, (do inglês, *human visual system* HVS). Isso nos permite inserir a marca em regiões onde o HVS tem menor sensibilidade, como por exemplo, nas componentes de alta resolução[26].

### 2.1.3 Técnicas de Inserção no Bitstream (MPEG)

Alguns métodos de inserção de marca d'água que se baseam fortemente nas estruturas de codificação (MPEG-1, -2 ou -4). Estes métodos têm como objetivo principal integrar a compressão com a mitigação de erros para minimizar a complexidade de processamento. Uma das maiores desvantagens da inserção de marcas no bitstream é que as marcas são altamente sensíveis a recompressão do material, seja com parâmetros diferentes ou com outro padrão ou formato.

Existe uma grande variedade de algotimos propostos que usam MPEG, incluindo métodos baseados na modificação de grupo de imagens (do inglês, *group of pictures* GOP) [20] na manipulação dos coeficientes DCT de alta frequência [7], na classificação dos blocos DCT [13], na alteração de pares de coeficientes DCT do bloco de luminância do vídeo, na alteração dos vetores de movimento, etc [26].

## 2.2 Modulação do Índice de Quantização

Neste trabalho o método proposto utiliza um algoritmo inspirado no algoritmo de modulação do índice de quantização [6] (do inglês, *Quantization Index Modulation*) para a inserção de marcas d'água no vídeo. A escolha do QIM como técnica inspiradora tem como principal razão a sua baixa complexidade computacional, a sua robustez e a sua alta capacidade de inserção de dados. Nesta seção explicaremos o funcionamento do algoritmo do QIM e mostraremos uma possível implementação.

### 2.2.1 Funcionamento do Algoritmo

No algoritmo QIM, a função de inserção $s(x, m)$ é tratada como um conjunto de funções de $x$ (sinal original) indexadas por $m$. Para que a distorção causada pela inserção seja visualmente imperceptível, cada função do conjunto deve estar próxima a uma função identidade, ou seja:

$$s(x; m) \approx x, \forall m. \tag{2.2}$$

Para que o algoritmo seja robusto, os pontos de cada função devem estar distantes. No mínimo, devem ser não intersectantes, caso contrário, mesmo na ausência de perturbações, pode haver algum valor de $s$ para a qual não seja possível determinar um valor de $m$ único.

A propriedade de não-interseção, juntamente com a propriedade de proximidade da identidade da Eq. 2.2, sugerem que as funções sejam descontínuas. Quantizadores são

uma classe de funções próximas da identidade e descontínuas. Logo, o algoritmo QIM, insere informação modulando um índice ou uma sequência de índices com a informação a ser inserida e, em seguida, quantizando o sinal original com o quantizador ou sequência de quantizadores. Convenientemente, as propriedades da sequência de quantizadores podem ser relacionadas diretamente com alguns parâmetros de desempenho como taxa de inserção, distorção e robustez. Por exemplo, o número de quantizadores no conjunto determina a taxa de inserção. O tamanho e o formato da célula de quantização determina a distorção, oriunda de erros de quantização. Finalmente, a distância mínima entre os pontos de reconstrução de diferentes quantizadores no conjunto é dada pela expressão

$$d_{min} \triangleq \min_{(i,j);i\neq j} \min_{x_i,x_j} ||s(x_i;i) - s(x_j;j)||, \tag{2.3}$$

que determina a robustez da inserção. É importante enfatizar que, diferentemente do caso em que o sinal original é conhecido no destino, o decodificador de distância mínima precisa testar todos os pontos de reconstrução de todos os quantizadores, não só aqueles que realmente correspondem ao sinal original. Em particular, o decodificador deve fazer decisões de acordo com a equação:

$$m(y) = arg \min_{m} \min_{x} x||y - s(x;m)||. \tag{2.4}$$

Se, como é frequente, os quantizadores $s(x;m)$ mapearem $x$ ao ponto de reconstrução mais próximo, então a equacao acima pode ser reescrita como

$$m(y) = arg \min_{m} ||y - s(y;m)||. \tag{2.5}$$

### 2.2.2 Implementação

Uma decisão importante que deve ser tomada ao se projetar um algoritmo QIM é a escolha do conjunto de quantizadores. Uma estrutura conveniente são os quantizadores *dither*. Para contextos diferentes de marcas d'água, as mudanças tipicamente correspondem a vetores pseudo-aleatórios denominados de vetores de *dither*. Para inserção de marcas d'água o vetor de *dither* pode ser modulado com o sinal a ser inserido, isto é, cada valor $m$ do sinal é mapeado a um vetor de *dither* diferente $d(m)$. A função de inserção fica então

$$s(x;m) = q(x + d(m)) - d(m). \tag{2.6}$$

Onde $q\dot{()}$ é o quantizador. As técnicas de inserção que usam esses vetores damos o nome de modulação *dither*.

Uma possível implementação do QIM usando modulação *dither* é descrita no próprio artigo sobre QIM [6] e é conhecida (em inglês) como *Coded Binary Dither Modulation with Uniform Scalar Quantization*, ou em uma tradução livre, modulação binária codificada dither com quantização uniforme e escalar.

Nesta técnica, os bits de informação $\{b_1, b_2, \ldots, b_N\}$ que representam a mensagem a ser inserida, $m$, são passados por um código corretor de erros de taxa $k_u/k_c$ para se obter uma sequência codificada $\{z_1, z_2, \ldots, z_{N/L}\}$. No caso sem codificação, $z_i = b_i$ e $k_u/k_c = 1$. Dividimos o sinal original $x$ em $N/L$ blocos não intersectantes de tamanho $L$. Em seguida,

dois vetores dither de comprimento $L$, $d[k,0]$ e $d[k,1]$, e uma sequência de mesmo tamanho de quantizadores uniforme e escalar com tamanho de passo $\Delta_1, \ldots, \Delta_L$ são construídos de acordo com a expressão:

$$d[k,1] = \begin{cases} d[k,0] + \Delta_k/2, d[k,0] < 0 \\ d[k,0] - \Delta_k/2, d[k,0] \geq 0 \end{cases} \quad k = 1, \ldots, L. \tag{2.7}$$

Percorremos então os bits da mensagem e quantizamos o bit $x_k$ com o quantizador $d[k,0]$, se o bit for 0 ou com o quantizador $d[k,1]$, caso o bit seja 1. A quantização é feita de acordo com a Eq. 2.6.

## 2.3 Halftoning

*Halftoning* [16] é a técnica para se converter imagens de tons contínuos em imagens binárias usando padrões de pontos pretos e brancos. É uma técnica útil para se criar a ilusão de múltiplos níveis de intensidade em uma imagem binária, o que a torna uma boa escolha para aplicações onde um número reduzido de níveis é necessário, tais como jornais, *fax* e processos de impressão de documentos. Os algoritmos de *Halftoning* podem ser grosseiramente classificados em pontilhado ordenado (do inglês, *ordered dithering*) ou pontilhado com difusão de erros (do inglês *dithering with error diffusion*) [24].

Algoritmos pontilhado ordenado geram imagens em *halftone* comparando os valores de intensidade dos pixels da imagem original com limiares que são valores escalares ou matrizes, que podem ser gerados por vários métodos. Esses algoritmos podem ser subdivididos em 2 tipos: pontos dispersos (em inglês, *dispersed dot*) ou pontos aglomerados (em inglês, *clustered dot*). Com pontos dispersos os pontos menos relevantes da representação são espalhados em uma ordem aparentemente aleatória, mantendo uma distribuição local que mantém a correlação com os níveis de cinza da imagem original. Nos pontos algomerados, os pixels da imagem binária são aglomerados de forma a criar uma sensação visual de diferentes tons. Algoritmos de difusão de erro comparam os valores de intensidade dos pixels com um limiar fixo. O erro resultante entre o valor de saída e o valor original são distribuídos aos pixels vizinhos de acordo com pesos predefinidos. Ambas técnicas possuem vantagens e desvatagens para determinadas aplicações.

*Inverse Halftoning* é a técnica que permite a restauração da informação original de uma imagem (com vários níveis de cinza) a partir da sua representação binária. Pode ser usada em várias aplicações onde a única versão disponível de uma imagem é a binária. Por exemplo, é sabido que imagens em *halftone* degradam muito ao serem sujeitadas a operações simples, tais como filtragem, dizimação, interpolação, etc. Nesses casos, é necessário usar *inverse halftoning* para converter a imagem binária em uma imagem em escalas de cinza e, então, realizar as operações pretendidas.

Neste trabalho, usamos o método pontilhado ordenado de pontos dispersos para o processo de *halftoning* e uma técnica modificada de Freitas *et al.* [11] para o processos inverso.

# Capítulo 3

# Mitigação de Erros

Os métodos de mitigação de erros podem ser divididos em três categorias [35]. Mitigação de erros para frente (*Forward Error Concealment*), mitigação de erros por pós-processamento (*Error Concealment by post-processing*), e mitigação interativa de erros (*Interactive Error Concealment*), todas as quais serão melhor definidas mais a frente no texto.

A Figura 3.1 mostra um diagrama de um sistema de comunicação de vídeo em tempo real. A entrada é comprimida pelo codificador de fonte na taxa de bits desejada. Na figura, o codificador da camada de transporte refere-se a um conjunto de dispositivos que fazem a codificação de canal, o empacotamento e/ou modulação e o controle da camada de transporte (usando um protocolo de transporte). Esses codificador é para converter a saída do codificador de fonte para unidades de dados prontas para o envio.

O codificador de fonte pode ser particionado em dois componentes. O codificador de forma de onda e o codificador entrópico. O codificador de forma de onda é um dispositivo que reduz a taxa de bits do sinal de vídeo usando transformadas e quantização, portanto inserindo perdas irreversíveis na qualidade do sinal. Exemplos de codificadores de forma de onda incluem codificadores que usam a transformada DCT ou wavelet, assim como quantização vetorial. O codificador entrópico é um dispositivo que mapeia os símbolos do codificador de forma de onda em palavras de código binário de acordo com a distribuição estatística destes símbolos, sem introduzir perdas. Exemplos de codificadores entrópico incluem o código de Huffman [14] e a codificação aritmética [27].

No lado do receptor, as operações inversas são feitas para se obter o sinal de vídeo reconstruído. No diagrama, setas duplas são utilizadas para mostrar que algumas aplicações enviam informações do decodificador de volta ao codificador para controle e, em alguns casos, para mitigação de erros

Em geral, para se melhorar o desempenho do algoritmo de mitigação de erros no decodificador, uma certa redundância precisa ser adicionada no codificador de forma de onda, no codificador de entropia ou no codificador de camada de transporte. A esta redundância se dá o nome de redundância de mitigação. Quando a taxa de erro do canal aumenta, uma maior porcentagem da banda total deve ser alocada para a redundância de mitigação para se obter uma melhor qualidade do vídeo.

O problema da mitigação de erros pode ser sintetizado como o problema de projetar um par de codificador/decodificador de fonte e codificador/decodificador de camada de transporte de forma a minimizar a distorção do sinal para um dado modelo de vídeo, banda

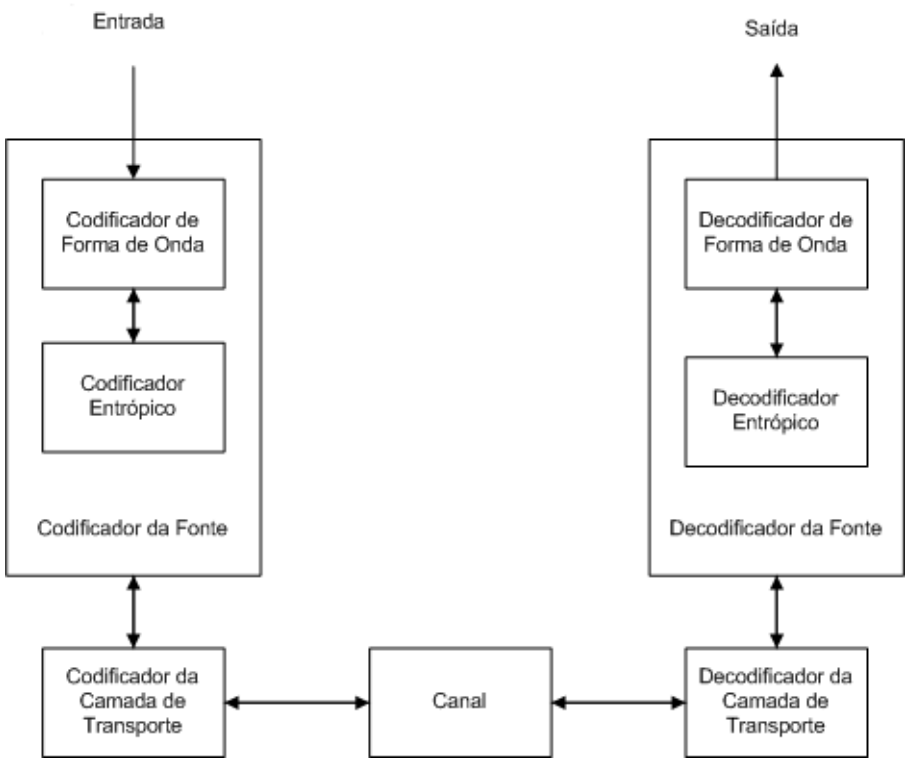



Figura 3.1: Diagrama mostrando um sistema de comunicação de vídeo.

total do canal e características de erro do canal. Esse problema é muito díficil (se não impossível) de se resolver, pois envolve muitas variáveis díficeis de modelar. Infelizmente, o projeto de um bom codificador de fonte requer um bom modelo para que se possa obter bons resultados com o algoritmo de mitigação de erros.

Os algoritmos de mitigação de erros disponíveis na literatura utilizam diversas abordagens. Na maioria delas, algumas variáveis são mantidas fixas e uma solução localmente ótima é obtida. Em suma, as abordagens podem ser divididas em três categorias.

- Mitigação de erros para frente (*forward error concealment*): O codificador é projetado para minimizar os efeitos de erros de transmissão, sem a necessidade de mitigação de erros no decodificador, o que torna o trabalho do decodificador mais simples.
- Mitigação de erros por pós-processamento (*error concealment by post-processing*): O decodificador é quem atua para a mitigação. Em geral, esses métodos tentam recuperar as perdas através de interpolação espacial e temporal sem que haja a necessidade de informações adicionais.
- Mitigação de erros interativa (*interactive error concealment*): O codificador e o decodificador trabalham em conjunto para minimizar os impactos na transmissão.

## 3.1 Detecção de Erros

Antes que se possa falar de métodos de mitigação de erros é necessário descobrir onde e quando os erros de perdas de qualidade acontecem. As técnicas de detecção se encaixam em duas categorias:

- Detecção na camada de transporte;
- Detecção no decodificador.

Uma maneira de se fazer a detecção de erros no codificador da camada de transporte é inserindo informação no cabeçalho. Por exemplo, na transmissão de vídeo baseada em pacotes, o *bitstream* de saída do codificador de vídeo é dividida em pacotes, cada um dos quais contém um cabeçalho e um campo de carga (*payload*) [23]. O cabeçalho contém ainda um campo de número de sequência, que identifica o pacote na sequência. No decodificador, esse número pode ser usado para detecção de perdas de pacote.

Um outro método de detecção na camada de transporte é a utilização de FEC. Ou seja, um código corretor de erros é aplicado em segmentos da saída do codificador. No receptor, é feita uma decodificação desses códigos para detecção e possível correção de possíveis erros de bit.

Para a detecção de erros no decodificador, as características dos sinais de vídeos são exploradas. Dois dos métodos utilizam as diferenças de pixels em duas linhas vizinhas para detectar erros de transmissão em vídeos codificados em PCM (do inglês, *pulse code modulation*) [19] ou DPCM (do inglês, *differential pulse code modulation*) [22]. Em um outro método, detecta-se o dano de um único coeficiente DCT se examinando a diferença entre pixels na vizinhança entre um bloco e quatro blocos vizinhos. No decodificador, quatro vetores de diferença são formados tomando-se as diferenças entre o bloco atual e seus blocos adjacentes nas quatro direções. Em seguida, uma DCT unidimensional é aplicada a esses vetores. Assumindo que a transição entre blocos seja suave, os valores dos vetores DCT unidimensionais devem ser relativamente pequenos na ausência de erros de transmissão. Logo, se esses vetores têm um coeficiente dominante, declara-se que um coeficiente foi danificado. Adicionalmente, a posição do coeficiente danificado também é estimada. Esse método assume que somente um coeficiente foi danificado. No caso de múltiplos coeficientes danificados, ele detecta e corrige apenas o coeficiente que possui o maior erro.

Esta técnica também pode ser utilizada no domínio da frequência. Nessa abordagem, uma palavra de código de sincronização é inserida ao final de cada linha de blocos. Quando essa palavra de sincronização é detectada, verifica-se se o número de blocos decodificados é igual a um número pré-determinado. Caso se encontre alguma diferença, é declarado que houve um erro. A posição do bloco onde o erro acontece é determinado pelo procedimento abaixo.

1. Uma média ponderada de erro quadrático é calculada entre os coeficientes na linha atual e na linha anterior para cada bloco 8x8.
2. Um peso maior é utilizado para coeficientes de baixa frequência e um peso menor para coeficientes de alta frequência, de forma que a medida de distorção seja mais próxima da distorção como percebida por observadores humanos.

3. O bloco com maior erro é dividido em dois blocos ou fundido com um bloco adjacente, dependendo se o número de blocos decodificados for maior ou menor que o número predeterminado.

4. Quando mais de um bloco for danificado o processo de divisão/fusão continua até que o número de blocos se alinhe com o número predeterminado.

Quando se utiliza códigos de comprimento variável (CCV) no codificador, erros em único bit do código podem causar erros de sincronização e, consequentemente, bits seguintes podem se tornar indecifráveis. No entanto, esta característica pode ser utilizada para se detectar erros de transmissão. Note que, na maioria dos casos, o código utilizado não é completo, isto é, nem todas as possíveis palavras são válidas. Assim, quando um decodificador detecta uma palavra que não esteja na tabela de decodificação, assume que houve um erro de transmissão. Ademais, a própria sintaxe utilizada no bitstream também pode ser usada para detecção. Por exemplo, se o número de coeficientes DCT for maior que o máximo (64 para um codificador 8 x 8) então um erro de transmissão é detectado.

Geralmente, detecção de erros com adição de informações no cabeçalho e/ou códigos FEC no nível da camada de transporte são mais confiáveis, ao custo de um aumento da banda de transmissão exigida. Os benefícios de técnicas de detecção de erros no decodificador que utilizam a caraterística de suavidade dos sinais de vídeo é que estas não aumentam o número de bits necessários para o codificador. O uso de palavras de sincronização e/ou CCV pode ser considerado um meio termo, onde é mantido um pequeno grau de redundância no processo de codificação, de forma a facilitar a detecção de erros no decodificador. Essas técnicas não são mutuamente exclusivas e podem ser usadas em conjunto.

## 3.2 Mitigação de Erros para Frente

Nesta seção serão descritos alguns métodos de mitigação onde o codificador desempenha o papel principal na mitigação. Quando o canal de transporte não é sem perdas, existem dois tipos de distorções que são comummente observadas no decodificador:

- Ruído de Quantização introduzido pelo codificador de forma de onda e
- Distorção devido a erros na transmissão.

Uma dupla ideal de codificador da fonte e codificador da camada de transporte (incluindo FEC, empacotamento e protocolos de transporte) deve ser projetada tal que a distorção conjunta dos dois tipos de erros seja minimizada, dado a banda disponível e as características de erro do canal. Tipicamente, o codec de vídeo é projetado para minimizar erros de quantização.

### 3.2.1 Codificação em Camadas com Priorização de Transporte

Até agora o método mais popular e eficaz de prover resistência a erros em um sistema de transporte de vídeos é a codificação em camadas com priorização de transporte. O termo priorização de transporte aqui se refere a vários mecanismos de QoS [10], , incluindo

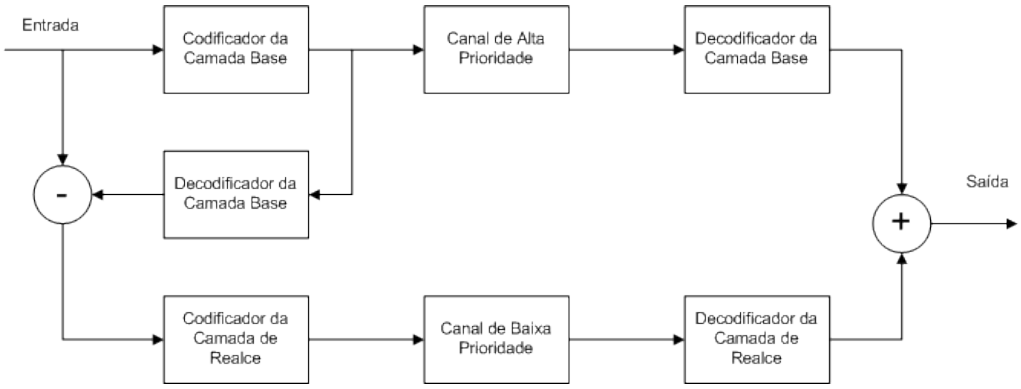



Figura 3.2: Diagrama de um sistema usando codificação em camadas e priorização de transporte.

proteção desigual de erros, que permitem taxas diferentes de perda/erro de canal, dando prioridades diferentes para dar suporte a taxas diferentes de atraso/perda.

Na codificação em camadas (do inglês *Layered coding*), a informação do vídeo é particionada em mais de um grupo ou camada. A Figura 3.2 mostra o diagrama em blocos de um sistema genérico de duas camadas. A camada base contém a informação essencial do vídeo fonte e pode ser usada para gerar a saída do sinal de vídeo com uma qualidade aceitável. Com camadas de realce, um sinal de vídeo de maior qualidade é obtido. Para se combater erros do canal, a codificação em camadas deve ser combinada com a priorização de trasporte para que a camada base seja entregue com um grau maior de proteção de erros. Em redes ATM existe um bit no cabeçalho da célula ATM que sinaliza sua prioridade. Quando um congestionamento do tráfego ocorre, um nó da rede pode escolher descartar células com menor prioridade primeiro. Priorização de transporte também pode ser implementada usando diferentes níveis de potência para se transmitir os subfluxos em um ambiente de tranmissão sem fio. Essa combinação de codificação em camadas com controle de potência desigual foi estudada para transmissão de vídeo em redes sem fio [15]. Adicionalmente, priorização pode ser feita usando diferentes técnicas de controle de erros para as diferentes camadas. Por exemplo, retransmissão e/ou FEC podem ser aplicados para a camada base, enquanto métodos de retransmissão/FEC mais fracos ou mesmo nenhum método é aplicado às camadas de realce [3].

Pode-se implementar codificação em camadas de várias formas, dependendo da forma na qual a informação de vídeo for particionada. Se a partição for feita no domínio do tempo, a camada base contém um bitstream com uma taxa de quadros menor e as camadas de realce contém informações incrementais para que seja obtida uma saída com uma taxa de quadros maior. Na codificação no domínio espacial, a camada base codifica uma versão subamostrada da sequência de vídeo original e as camadas de realce contém informação que permite se obter uma maior resolução espacial no decodificador. A camada base também pode codificar o sinal de entrada com uma quantização mais grosseira, deixando os detalhes para as camadas de realce. Em geral, pode ser aplicada diretamente nas amostras da entrada. As duas primeiras técnicas são conhecidas como refinamento de resolução temporal e refinamento de resolução espacial, respectivamente. A terceira é conhecida como refinamento de resolução de amplitude. Por fim, em codificadores baseados

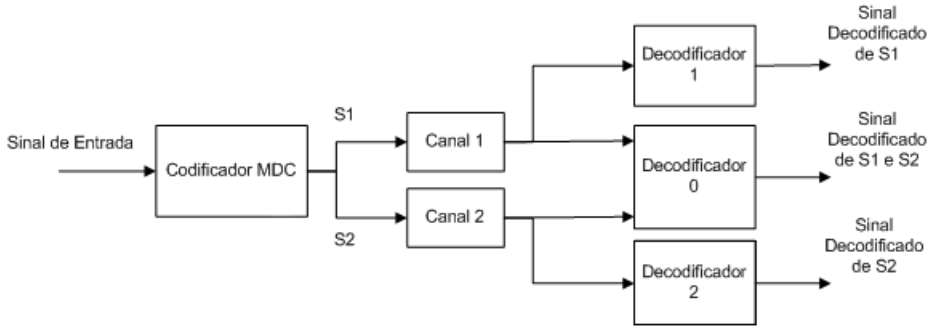



Figura 3.3: Diagrama de Codificação de Múltipla Descrição.

em transformadas, pode-se incluir coeficientes de baixa frequência na camada base e as altas frequências nas camadas de realce. Essa técnica é chamada de particionamento do domínio da frequência. Em um codificador usando predição compensada pelo movimento (*motion-compensated prediction*), o modo de codificação e os vetores de movimento são geralmente inseridos na camada base, dado que estas são as informações mais importantes. Note que os esquemas supracitados não precisam ser usados isoladamente, podendo ser combinados.

Apesar de nenhuma informação extra ser explicitamente inserida na codificação em camadas, o aumento de complexidade do sistema se traduz em ganho quando perdas acontecem. Em geral, tanto o codificador quanto o decodificador tem que ser implementados com uma estrutura multi-camadas mais complicada. Incluir camadas também aumenta a complexidade da codificação. Essa complexidade extra é influenciada por vários fatores, incluindo o método de codificação em camadas, a resolução temporal e espacial e a taxa de bits. Por exemplo, com o método de particionamento de dados, um processamento extra (relativamente menor) será necessário para obtermos uma taxa de bits mais alta.

Ao se projetar um codificador em camadas, deve-se levar em conta se a informação das camadas de realce será utilizada para a predição da camada base. Se forem utilizadas, o ganho na camada base será maior. Entretanto, quando informações da camada de realce forem perdidas, distorções também ocorrerão distorções na camada base. Logo, em alguns sistemas, a predição da camada base é feita usando somente informação da própria camada base para evitar esse problema.

### 3.2.2 Codificação de Múltipla Descrição

Conforme visto anteriormente, a codificação em camadas oferece resistência a erros quando a camada base é transmitida em um canal essencialmente sem erros. Em algumas aplicações no entanto, pode não ser viável garantir transmissão sem erros de uma porção dos dados enviado. Nesses casos, uma perda na camada base pode levar a efeitos desastrosos na qualidade do vídeo decodificado. Uma abordagem alternativa para se combater erros de transmissão no lado do codificador é utilizar codificação de múltipla descrição (do inglês *multiple-description coding, MDC*) [35]. Esse esquema de codificação assume que existem vários canais paralelos entre a fonte e o destino e que cada canal pode estar temporariamente parado ou sofrendo de uma rajada de erros. Os eventos de erros dos canais são independentes, logo a probabilidade que todos os canais estejam simultanea-

mente sofrendo erros é pequena. Esses canais podem ser caminhos fisicamente diferentes entre a fonte e o destino como, por exemplo, uma rede sem fio *multihop* ou uma rede de comutação de pacotes. Mesmo quando só existir um caminho físico, pode-se dividir o caminho em canais virtuais usando divisão de frequência, ou qualquer outra técnica de compartilhamento de canal.

Com MDC, vários fluxos de código (conhecidos como descrições) do mesmo sinal fonte são gerados e transmitidos por canais separados. No destino, dependendo de quais descrições chegam corretamente, diferentes esquemas de reconstrução (ou decodificadores) serão invocados. O codificador e o decodificador são projetados de tal forma que a qualidade do sinal reconstruído é aceitável com apenas uma descrição e que uma grande melhoria é possível com mais descrições. Um diagrama conceitual das duas descrições é visto na Figura 3.3. Nesse caso, existem três decodificadores no destino e apenas um funciona de cada vez. Para garantir uma qualidade aceitável com uma descrição, cada descrição deve carregar informação suficiente sobre o sinal original. Isso implica que as descrições terão sobreposições de informação. Obviamente, isto vai reduzir a eficiência da codificação, quando comparada com a eficiência do codificador convencional com única descrição (do inglês *single-description coder, SDC*) cujo foco é minimizar a distorção na ausência de erros de canal. Essa perda na eficiência representa um compromisso entre entre eficiência e robustez a longas rajadas de erros e/ou falhas de canal. Com SDC, muitos bits de controle de erros seriam gastos e/ou uma latência adicional seria preciso para se corrigir os erros de canal. Com MDC, uma rajada de erros, ou ainda a perda de um descritor inteiro, não tem um efeito catastrófico contanto que pelo menos uma descrição chegue ao receptor. Assim, pode-se usar menos bits de controle de erros para cada fluxo.

Uma forma simples de se obter descrições consiste em dividir as amostras adjacentes de sinal entre os vários canais, usando uma estrutura de subamostragem entrelaçada (em inglês *interleaving subsampling lattice*) [33] e codificar as subimagens resultantes de forma independente. Se alguma subimagem não chegar ao receptor, pode-se facilmente recuperá-la usando a correlação entre amostras adjacentes da imagem original. Essa abordagem causa uma sobrecarga na taxa de bits, pois o codificador não pode fazer uso da correlação entre amostras adjacentes.

### 3.2.3 Codificação Conjunta entre Fonte e Canal

Nos métodos citados anteriormente a interação entre os codificador de fonte e da camada de transporte é feita em alto nível. Na codificação em camadas, o codificador da fonte produz um fluxo em camadas assumindo que o codificador de transporte garantirá a entrega da camada mais importante. Por outro lado, com o MDC o codificador de fonte assume que todos os bits codificados serão tratados igualmente e que todos estão sujeitos a erros. Métodos que tratam a interação a mais baixo nível são conhecidos como codificação conjuntar entre fonte e canal (do inglês *joint source and channel coding*) [35].

Em geral, a codificação conjunta é feita projetando os codificadores de quantização e entropia para características de erro do canal de forma a minimizar os erros de transmissão. Algumas das técnicas estudadas para fontes genéricas são descritas a seguir:

- Spilker [30] notou que, quando o canal se torna muito ruidoso, um quantizador grosseiro tem melhor desempenho que um quantizador mais fino, para um codificador de fonte baseado em PCM.

- Kurtenbach e Wintz [17] projetaram quantizadores ótimos para minimizar o erro médio quadrático introduzido por erros tanto na quantização quanto no canal, considerando a distribuição de probabilidade da entrada e a matriz de erros do canal.
- Farvardin e Vaishampayan [9] extenderam o projeto dos quantizadores de Kurtenbach e propuseram um método para atribuir palavras chaves a características de erro do canal.

Para imagens, a aplicação de códigos de convolução foram propostos para proteger o sinal contra erros de canal, usando um codificador DPCM. Três técnicas usando códigos de convolução seguem:

1. A modulação e o código corretor de erros são os mesmos para todos os bits de todos os coeficientes quantizados da transformada.
2. A modulação e o código corretor de erros são iguais para os bits do mesmo coeficiente quantizado, mas podem ser diferentes para coeficientes diferentes.
3. A modulação e código corretor de erros variam entre bits do mesmo coeficiente.

Para o primeiro caso, em uma imagem típica de exterior, quando a razão sinal ruído do canal for menor que 10dB, a razão da imagem recebida é melhor com 50% de bits de código corretor de erro do que sem nenhum. A segunda e terceira opções melhoram ainda essa razão, descendo o limiar da razão sinal ruído do canal para 5dB.

## 3.3 Mitigação de Erros por Pós-Processamento

É sabido que imagens de cenas naturais tem como componentes de maior predominância os de baixa frequência, isto é, os valores dos pixels adjacentes variam suavemente, exceto em regiões com bordas [35]. Além disso, os olhos humanos toleram mais distorções em componentes de alta frequência do que nas componentes de baixa frequencia. Esse fatos podem ser usados para esconder os artefatos causados por erros de tranmissão. As técnicas citadas nesta seção realizam a mitigação de erros no decodificador. Alguma dessas técnicas podem ser usadas em conjunto com a informação auxiliar do codificador da fonte para melhorar a qualidade da reconstrução.

As técnicas de pós-processamento descritas a seguir fazem uso da correlação entre um macrobloco danificado e seus vizinhos no mesmo quadro e/ou no quadro anterior para mitigar erros. Algumas dessas técnicas só se aplicam a macroblocos codificados no modo intra, enquanto outras, embora aplicavéis ao modo inter, não fazem uso da informação temporal.

### 3.3.1 Predição Temporal por Compensação de Movimentos

Uma forma simples de explorar a correlação temporal em sinais de vídeo é a troca de um macrobloco danificado pelo seu macroboloco espacialmente correspondente do quadro anterior. Esse método, no entanto, pode produzir artefatos visuais na presença de muito movimento. Melhorias significativas são feitas quando se troca o bloco danificado

pelo bloco compensado de movimento. Esse método, conhecido em inglês como *Motion-Compensated Temporal Prediction* [35], é bastante efetivo quando combinado com uma codificação em camadas que inclui todas as informações de predição na camada base. Por causa de sua simplicidade esse método é bastante usado. De fato, o padrão MPEG-2 permite ao codificador enviar os vetores de movimento para macrobloco intracodificados, tal que esses blocos podem ser melhor recuperados se forem danificados na transmissão. Um problema dessa abordagem é que requer conhecimento da informação de movimento, algo que pode não ser disponível em todas as situações. Quando os vetores de movimento são danificados, eles precisam ser estimados dos vetores de movimento dos macroblocos vizinhos e estimativas incorretas podem levar a erros grandes em imagens reconstruídas. Outro problema que ocorre é quando o macrobloco é codificado no intramodo e a informação da codificação é perdida. Assim, mitigação com esse método pode levar a resultados ruins em situações como uma mudança de cena.

### 3.3.2 Recuperação Maximamente Suave

Esse método faz uso da característica de suavidade da maioria dos sinais de imagem e vídeo através de uma estratégia de minimização de energia. Essa minimização é feita bloco a bloco. Especificamente, para se estimar coeficientes DCT perdidos em um bloco, o método minimiza uma medida da variação espacial e temporal entre pixels vizinhos nesse bloco e seus blocos vizinhos no espaço e no tempo de tal forma que o sinal estimado seja tão suave quanto for possível .

Zhu *et al.* [39] propuseram um método de mitigação utilizando compensação de movimento, adicionando uma medida de suavidade temporal. No caso, a função a ser minimizada é uma soma ponderada de uma medida de diferença espacial e uma medida de diferença temporal. Para facilitar os cálculos, as medidas de diferença espacial e temporal são definidas como a soma das diferenças quadradas entre pixels adjacentes espacial e temporalmente, respectivamente. O bloco a ser reconstruído é representado em termos dos coeficientes recebidos, os coeficientes perdidos a serem estimados e o bloco de predição no quadro anterior (apenas para os blocos inter). A solução consiste em três interpolações lineares – nos domínios espacial, temporal e de frequência – dos pixels em blocos de imagens adjacentes que foram reconstruídos anteriormente, do bloco de predição do quadro anterior e dos coeficientes recebidos para este bloco, respectivamente. Quando todos os coeficientes de um bloco forem perdidos, a solução se reduz a interpolações espacial e temporal somente. Se o peso da diferença espacial for zero, a solução então é equivalente a repor um bloco danificado pelo bloco de predição, a mesma solução da seção anterior. Por outro lado, se o peso da diferença temporal for zero, somente a correlação espacial é usada e a solução é a interporalação linear entre os coeficientes recebidos e dados dos pixels vizinhos. Esse peso pode usar utilizado para blocos intra ou imagens. O operador de reconstrução depende dos pesos dados e da transformada associada aos coeficientes perdidos. Para um dado padrão de perdas, esse operador pode ser pré computado e a tarefa de reconstrução envolve um produto de matriz e vetor, com uma complexidade similar a de uma transformada de bloco.

Simulações feitas com a técnica descrita mostram que um bloco com os primeiros quinze coeficientes de baixa frequência pode ser recuperado com uma qualidade aceitável contanto que os blocos vizinhos estejam disponíveis para interpolação espacial/temporal.

Para aumentar a robustez, pode-se intercalar os coeficientes de blocos adjacentes de forma que um erro só afetará blocos espacialmente deslocados.

### 3.3.3 Projeção em Conjuntos Convexos

O método descrito na subseção anterior utiliza a característica de suavidade de sinais para minimizar a energia. Uma alternativa é se usar a projeção em conjuntos convexos (do inglês, *Projection onto Convext Sets*), ou POCS [35]. Os conjuntos convexos são adquiridos se determinando que os blocos recuperados tenham banda limitada isotropicamente (para um bloco em uma região suave) ou em uma direção particular (para um bloco contendo uma borda). Com essa técnica, um bloco combinado é formado ao se incluir oito blocos vizinhos com o bloco danificado.

Primeiro, aplica-se um operador Sobel nesse bloco combinado para testar a existência de bordas. O bloco é, então, classificado como monótono (sem bordas) ou como um bloco de bordas. A orientação das bordas é quantizada para uma das oito direções igualmente espaçadas de $0 - 180^o$. Em seguida, se o bloco for monótono, um filtro passa-baixas isotrópico é aplicado. Se o bloco for um bloco de bordas, um filtro passa-bandas é aplicado ao longo da direção das bordas deste bloco. Esses filtros são implementado no domínio de Fourier. Os valores dessa filtragem são truncadas para $[0, 255]$. Essas duas operações, filtragem e truncamento dos valores, são aplicados até que o bloco não tenha mais mudanças. Nos testes feitos pelos autores, entre cinco e dez iterações são suficientes [31].

## 3.4 Mitigação de Erros Interativa

Nessa seção descreveremos técnicas que utilizam cooperação entre o codificador e o decodificador. Essa cooperação pode ser realizada no codificador de fonte ou no codificador da camada de transporte. No codificador de fonte, parâmetros podem ser adaptados de acordo com *feedback* do decodificador. Na camada de transporte, a informação de *feedback* pode ser usada para mudar o percentil utilizado para o FEC ou para retransmissão.

### 3.4.1 Codificação Seletiva

A mitigação de erros não seria tão importante em transmissões em tempo real se erros de bit ou perdas de pacotes não causassem propagação de erros. Se os erros se mantivessem por apenas um ou dois quadros, o olho humano não perceberia [35]. Predição temporal é uma parte indispensável de codificadores pela alta redundância entre quadros. Assim, se o decodificador puder prover informação sobre a localização das partes danificadas para o codificador, esse pode tratar as áreas de forma diferentes, de forma que a propagação de erros possa ser reduzida ou eliminada. Uma técnica simples consiste em enviar uma requisição ao codificador, sempre que um erro for detectado. Desta forma, o próximo quadro será enviado no modo intra. Dessa forma, a propagação de erros se limitará a uma RTT.

Entretanto, vale lembrar que a codificação no modo intra, tipicamente, reduz o ganho de compressão e degrada a qualidade do vídeo. Para reduzir o aumento da taxa de bits necessária causado pela codificação intra, Wada [34] propôs dois métodos para realizar recuperação seletiva para mitigação de erros. Quando uma perda de pacotes for detectada,

o decodificador envia a informação sobre os blocos perdidos ao codificador. Em um dos métodos, a área afetada pelo bloco perdido é calculada até o quadro atualmente sendo codificado. Em seguida, a codificação é feita sem o uso dessa área para predição.

### 3.4.2 Transporte Adaptativo

Retransmissão tem sido usada com sucesso em transmissões de dados que não são em tempo real. No entanto, para transmissões em tempo real esta técnica é considerada inaceitável devido ao grande atraso causado. Ainda assim, retransmissão, com controle adequado, pode ser usada para melhorar a qualidade dessas transmissões.

Marasli *et al.* [21] propuseram uma técnica na qual, diferentemente do TCP, as tentativas de retransmissão são determinadas por um atraso especificado. Já Smith [29] propôs um CUDP (do inglês, *cyclical user datagram protocol*, em uma tradução livre, protocolo de datagrama cíclico de usuário) que põe os pacotes da camada base de um codificador em camadas na frente da fila de transmissão, para aumentar o número de tentativas de transmissão da camada base.

### 3.4.3 Retransmissão Sem Espera

Para fazer uso de dados retransmitidos, uma implementação típica do decodificador terá que esperar pela chegada dos dados antes de processá-los. Isso introduz um atraso. Se o decodificador decodificar mais rápidamente, depois da chegada dos dados retransmitidos somente alguns quadros estarão atrasados. Esse atraso é conhecido como atraso de tráfego. Por outro lado, o decodificador pode mostrar todos os quadros com um atraso fixo, conhecido como atraso acumulado.

Em uma técnica proposta por Zhu [38], quando uma unidade de dado de vídeo for danificada, um pedido de retransmissão é enviado ao codificador para recuperar a informação danificada. Em vez de esperar a chegada dos dados, a parte danificada do vídeo é processada por outra técnica de mitigação de erros. A decodificação normal é continuada, enquanto um rastreamente dos pixels afetados e suas informações de codificação são gravadas. Os pixels afetados se referem aqueles que estão sujeitos a propagação de erros dos blocos danificados. Na chegada da retransmissão, os pixels afetados são corrigidos, de tal forma que eles são reproduzidos como se nenhuma perda tivesse ocorrido.

# Capítulo 4

# Algoritmo Proposto de Mitigação de Erros

Nesse capítulo descrevemos o algoritmo proposto de Mitigação de Erros. Dividimos o algoritmo em duas partes independentes: o codificador e o decodificador. Utilizamos as bibliotecas *scipy* e *numpy* [1] da linguagem Python para a maior parte do código e a linguagem Fortran para o processo de *halftoning*.

## 4.1 Codificador

No codificador, utilizamos um algoritmo inspirado no algoritmo de modulação do índice de quantização (QIM) para inserir uma cópia binária de cada canal de cor de cada quadro do vídeo nele mesmo. As cópias binárias são obtidas através de uma técnica de *halftoning* que foi projetada para que o processo de *inverse halftoning* gerasse uma imagem com a maior qualidade possível. O diagrama de blocos do codificador proposto é apresentado na Figura 4.1.

### 4.1.1 Realce de Bordas

O primeiro passo para o processo de geração da imagem binária consiste em aplicar um filtro de realce de bordas. Essa filtragem é realizada para melhorar a qualidade da imagem reconstruída no decodificador, uma vez que o processo de *inverse halftoning* gera uma imagem com qualidade menor que a original. Aplicamos um filtro *unsharp mask* na

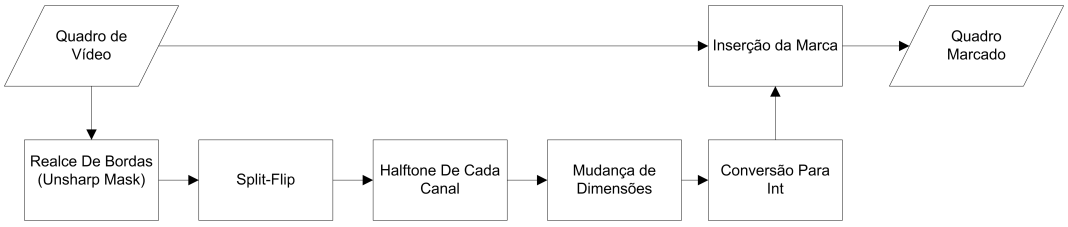



Figura 4.1: Diagrama do codificador.

Figura 4.2: Figura mostrando a imagem de referência (esquerda) e a imagem com realce de bordas (direita).

imagem original $I_o$, utilizando a equação:

$$I_{enh} = I_o - \bar{I} \tag{4.1}$$

onde, $\bar{I}$ é uma versão borrada de $I_o$ e geramos uma imagem com bordas realçadas $I_{enh}$. O algoritmo utilizado de *unsharp mask* é o utilizado pela biblioteca *scipy*. Na Figura 4.2 é apresentado um exemplo deste procedimento. A Figura 4.2a mostra a imagem original (Lena), enquanto que a Figura 4.2b apresenta esta imagem com as bordas realçadas.

### 4.1.2 Split-flip (Embaralhamento)

Separamos, então, $I_{enh}$ em seus três canais de cores. Em seguida, fazemos um embaralhamento das regiões de cada um desses canais. Mais precisamente, quebramos a imagem em 16 regiões e rotacionamos cada uma dessas regiões por $180^o$, embaralhando as regiões e gerando uma imagem espacialmente descorrelacionada com a original, ou seja, as regiões da imagem original não se encontram na mesma posição que na imagem embaralhada. Esse passo é necessário para garantir que a marca não seja armazenada na mesma posição que o conteúdo original a partir do qual ela foi gerada. A Figura 4.3 exemplifica o processo de embaralhamento, ao qual damos o nome de *split-flip*, aplicado na imagem Lena.

### 4.1.3 Halftoning

Conforme já descrito na Seção 2.3, para o processo de *halftoning* usamos uma técnica de pontos dispersos (*dispersed dot*). Mais especificamente, usamos padrões combinatoriais para gerar todas as combinações de bits necessárias para representar os níveis de intensidade de cinza. Conforme experimentos realizados, conseguimos inserir no máximo três bits por pixel com a marca d'água sem degradação. Assim, usando padrões combinatoriais podemos usar $2^3$ níveis de cinza, ou seja, oito.

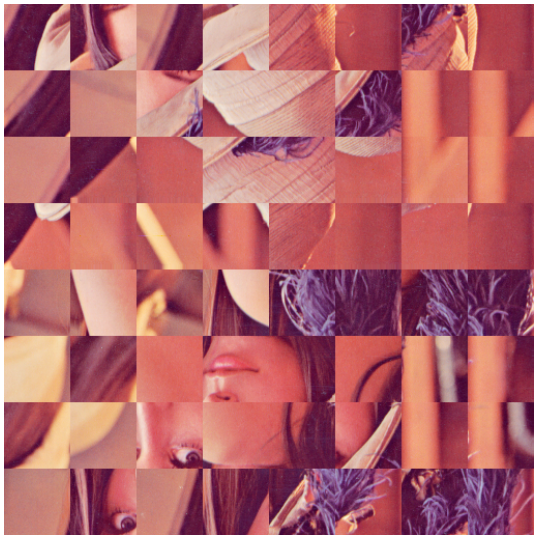


Figura 4.3: Imagem resultado da aplicação do processo *split-flip* na imagem Lena (Figura 4.2a).

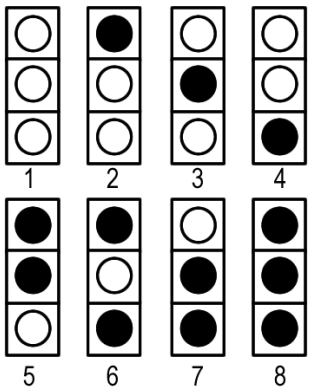



Figura 4.4: Padrões combinatoriais de pontos dispersos.

Aplicamos o processo de *halftoning* em cada canal de cor separadamente. Melhor explicando, tratamos cada canal de cor da imagem original como uma imagem em níveis de cinza e aplicamos o processo de *halftoning*. Os padrões combinatoriais de pontos dispersos utilizados para representar os níveis de intensidade dos canais de cores são apresentados na Figura 4.4.

Em seguida, quantizamos os níveis de intensidade da imagem em oito níveis distintos. A imagem resultante, $I_{quant}$, é obtida de acordo com a seguinte equação:

$$I_{quant}(x, y) = \left\lceil \frac{8}{255} I_{enh}(x, y) \right\rceil . \tag{4.2}$$

O método original de *inverse halftoning*, descrito em Freitas *et al.* [11], utiliza dez níveis de intensidade. Devido a limitações de capacidade de inserção do algoritmo, só podemos usar oito níveis neste trabalho. Essa diferença causa distorções na imagem reconstruída

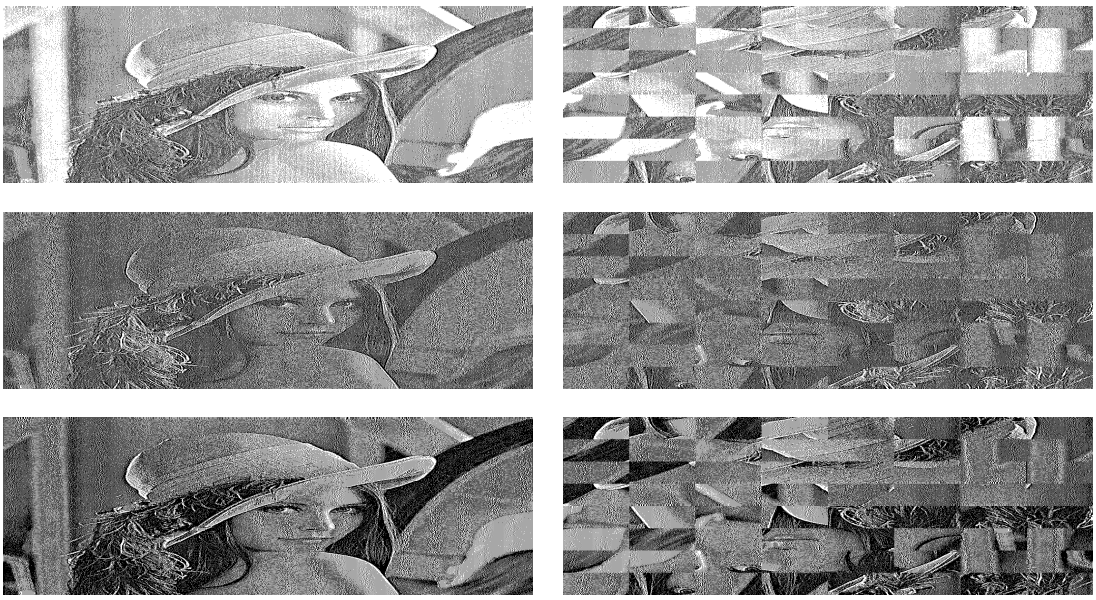


Figura 4.5: Halftones correspondentes aos canais de cores e suas respectivas versões embaralhadas utilizando o *split-flip*.

final. Para melhorar a qualidade do resultado final, levando em conta que algumas cores possuem maior área no espaço RGB, calculamos uma série de intervalos renormalizados ($\Phi$) do vetor original de intervalos utilizando a equação:

$$\Phi = \begin{bmatrix} 0.3000 & 0.0627 & 0.0000 \\ 0.0627 & 0.5900 & 0.0627 \\ 0.0000 & 0.0627 & 0.1100 \end{bmatrix} \Psi, \tag{4.3}$$

em que $\Psi$ é o vetor original de intervalos igualmente espaçados. Esses novos intervalos são mapeados pelos oito níveis representados pelos padrões de pontos da codificação em *halftone*.

Aplicamos esse processo de *halftoning* nos três canais de cor de cada quadro do vídeo. Ao final deste processo, obtemos três imagens em *halftone*, cada uma correspondendo a um canal de cor do quadro original. Cada canal, ao final do processo, tem três vezes a largura do original, conforme pode ser visto na Figura 4.5. Esse aumento do tamanho se deve ao algoritmo de *halftoning* utilizado, cujo efeito colateral é aumentar a imagem em algum dos eixos. A escolha desse algoritmo se deu pela necessidade de simplicidade, e também pelo fato desse algoritmo só precisar da distribuição local, em contrapartida com outros algortimos de *halftoning* onde a distribuição global dos pixels é necessária. Na primeira coluna dessa figura, vemos os *halftones* correspondentes aos 3 canais de cores. Na segunda coluna, vemos o resultado do *split-flip* aplicado a esses *halftones*.

### 4.1.4 Mudança de Dimensões e Conversão

Para cada quadro do vídeo, obtemos uma matriz $H$ de dimensões $(3M, N)$ onde $M$ é a largura do quadro original e $N$ é a sua altura. Convertemos essa matriz em uma nova matriz $H_3$ de dimensões $(M, N, 3)$. A partir dessa matriz, obtemos os valores dos pixels

Figura 4.6: Lena com marca d'água inserida.

de acordo com a seguinte equação:

$$bit_z = H_3(M, N, z), \text{ onde } z \in \{1, 2, 3\}. \tag{4.4}$$

Dessa equação, concatenamos os valores de $bit_1, bit_2, bit_3$, de maneira a formar um número binário. Finalmente, convertemos esse número binário para decimal, obtendo uma interpretação alternativa do *halftone*.

### 4.1.5 Inserção da Marca D'água

Para a inserção da marca d'água usamos um algoritmo inspirado no algoritmo QIM proposta nesse trabalho. Essa variante usa o valor obtido do estágio anterior como passo de quantização. Inserimos o valor decimal em um canal de cor de acordo com a seguinte equação:

$$I_{marked}(x, y) = \left\lfloor \left( \frac{I_o(x, y)}{\delta} \right) \right\rfloor \ \delta + I_{dither}(x, y), \tag{4.5}$$

na qual $I_{marked}$ é a imagem marcada, $I_o$ corresponde a imagem original, $I_{dither}$ é o valor em decimal da imagem binária e $\delta$ é a constante de quantização. Usamos 8 como valor de $\delta$. Em suma, o valor de cada pixel é discretizado de acordo com $\delta$ e, em seguida, este valor é somado ao valor em decimal da marca. Repetimos esse processo para cada canal de cor. A Figura 4.6 mostra a imagem Lena marcada com o algortimo proposto.

## 4.2 Decodificador

No decodificador, as cópias (marcas binárias) são extraídas, filtradas por um processo de *inverse halftoning*, desembaralhadas e utilizadas para reconstruir as partes do vídeo perdidas. O diagrama de blocos do decodificador proposto é apresentado na Figura 4.7.

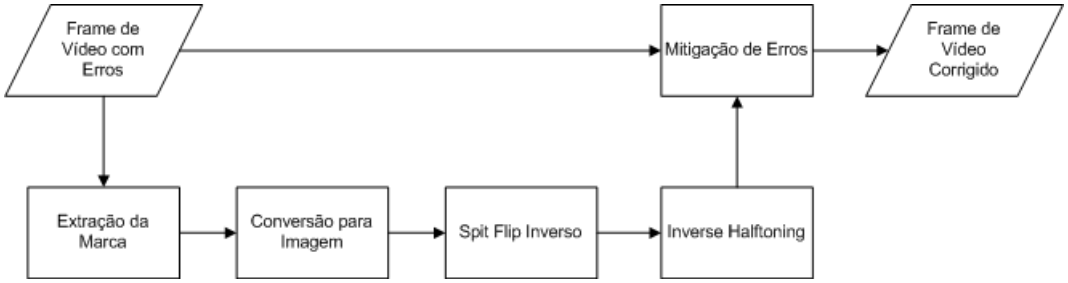



Figura 4.7: Diagrama do decodificador.

### 4.2.1 Extração da Marca

A extração da marca se dá por um processo inverso ao de inserção. Usamos a seguinte equação para a extração:

$$I_{dither}(x, y) = I_{marked}(x, y) \ mod \ \delta, \tag{4.6}$$

em que $I_{marked}$ é o canal de cor marcado, $I_{dither}$ corresponde ao valor em decimal recuperado da equação e $\delta$ corresponde a mesma constante de quantização usada na inserção.

### 4.2.2 Conversão e Inversão Split-Flip

Convertemos $I_{dither}$ de decimal para binário e recuperamos a matriz tridimensional $(M, N, 3)$, correspondente a imagem em *halftone* (ver Eq. 4.4) Reconvertemos essa matriz tridimensional para uma matriz unidimensional $(3M, N)$, de forma a recuperar o *halftone* em seu formato original. O processo é repetido para cada canal de cor. Ao final desse processo, temos as imagens binárias para cada um dos canais. Em seguida, aplicamos novamente o *split-flip* para desembaralhar a imagem, retornando-a a seu estado original.

### 4.2.3 Inverse Halftoning

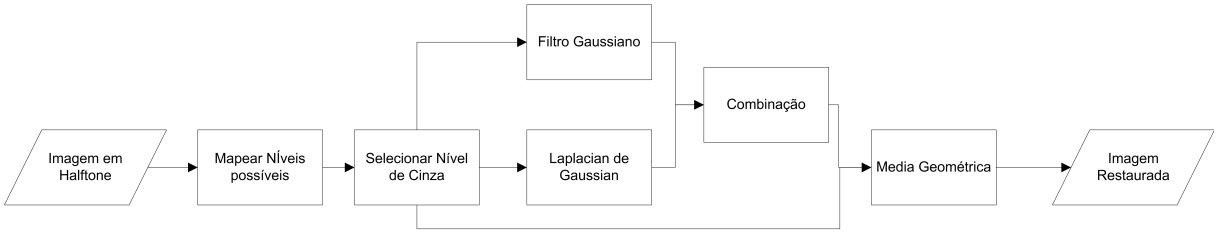



Figura 4.8: Diagrama do inverse halftoning.

Nesse passo, convertemos as imagens binárias recuperadas da extração da marca d'água em imagens com múltiplos níveis correspondentes ao 3 canais de cores. Em outras palavras, a partir da marca extraída recuperamos uma aproximação da imagem original.

Definamos $I_o$ como um canal de cor da imagem original, $I_{dither}$ como a sua representação em *halftone* e $D(p)$ como a distribuição da área ao redor do pixel $p$ de $I_{dither}$.

Figura 4.9: Inverse halftoning extraido de uma imagem marcada.

Para reconstruir a imagem de oito bits a partir da imagem de um único bit, calculamos a distribuição local $D(p)$ para todos os pixels em $I_{dither}$. Dessa distribuição, encontramos o intervalo correspondente que contém o valor de pixel mais provável da imagem original $I_o$. Desse intervalo, selecionamos aleatoriamente um valor dentro dele, gerando uma imagem levemente ruidosa $I_{col}$. Em seguida, filtramos $I_{col}$ com dois filtros: um passa-baixa Gaussiano e um Laplaciano-de-Gaussiano. A ideia é descorrelacionar espacialmente os pixels para poder processar cada um de forma independente, permitindo a reconstrução independente, mesmo quando os pixels vizinhos estejam faltando. As imagens resultantes, $I_{gauss}$ e $I_{log}$ são utilizadas para compor outra imagem $I_{blend}$, obtida pela seguinte equação:

$$I_{blend}(x,y) = \Upsilon\ I_{gauss}(x,y) + (1-\Upsilon)\ I_{log}(x,y), \tag{4.7}$$

onde $\Upsilon$ é a matriz de combinação que determina a proporção de cada imagem filtrada da entrada na saída.

Nos nossos experimentos, observamos que usando $\Upsilon = I_{col}$ preserva as bordas da imagem. Infelizmente, ao combinar os três canais recuperados para formar uma imagem colorida, várias distorções de cor são perceptíveis. Por outro lado, se usarmos $\Upsilon$ como uma matriz constante, $I_{blend}$ é uma versão borrada de $I_{dither}$. Realizamos então, outra composição utilizando as imagens $I_{col}$ e $I_{blend}$ com o objetivo de minimizar as distorções de cores e manter os detalhes da imagem original. A imagem gerada a partir da nova composição, $I_{rcv}$, é dada pela equação:

$$I_{rcv}(x,y) = \left\lfloor \sqrt{I_{col}(x,y)\ I_{blend}(x,y)} \right\rceil, \tag{4.8}$$

onde

$$I_{rcv} \approx I_o, \tag{4.9}$$

onde $I_{rcv}$ é a imagem recuperada, ou seja, uma aproximação da imagem original $I_o$. Vemos, na Figura 4.9 um exemplo de *inverse halftoning* obtido a partir da extração da marca inserida na imagem Lena (ver Figuras 4.2a e 4.6).

# Capítulo 5

# Resultados

Para testar o algoritmo proposto, utilizamos um conjunto de cinco vídeos no formato AVI, com resolução CIF 352x288. São eles *Cartoon*, *Rhinos*, *Mobile*, *Paris* e *Tempete*. Para testar o desempenho do algoritmo proposto, realizamos uma simulação de perdas de pacotes. Com este objetivo dividimos os quadros do vídeo em blocos de $8 \times 8$ pixels. Em seguida, selecionamos aleatoriamente uma certa quantidade de blocos para remoção. Foram gerados resultados com perdas de 30, 45, 60, 90 e 120 blocos por quadro. Na Figura 5.1 apresentamos exemplos de quadros dos vídeos *Rhinos e Tempete* com número crescente de perdas de blocos.

Os valores das posições das perdas é guardado como forma de detectar os erros. Tendo a informação das perdas em mãos, os erros são mitigados de acordo com o algoritmo proposto. Na Figura 5.2 e Figura 5.3 são apresentados exemplos dos resultados para os videos *Rhinos*, *Paris*, *Tempete* e *Mobile*. Na primeira linha das figuras são apresentados os quadros marcados, enquanto que na segunda linha são apresentados os quadros com blocos perdidos. A última linha das figuras apresenta restauração das imagens com as áreas afetadas pelo algoritmo.

Foram usadas cinco métricas de qualidade com referência diferentes para avaliar a qualidade dos vídeos gerados. São elas o erro médio quadrático (*mean square error*, MSE), a razão sinal ruído máxima (*peak signal-to-noise ratio*, PSNR), a razão sinal ruído (*signal-to-noise ratio*, SNR), a similaridade estrutural (*Structural Similartity*, SSIM) [36]

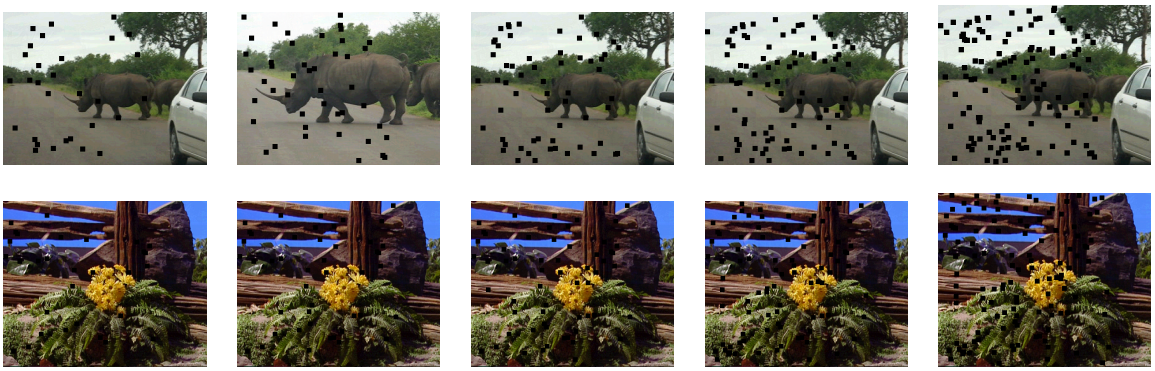


Figura 5.1: Quadros do video *Rhinos* e *Tempete* com perdas progressivas, de 30,45,60,90 e 120 blocos perdidos.

| | cartoon | rhinos | mobile | paris | tempete |
|---|---|---|---|---|---|
| MSE | 11,044 | 13,711 | 10,75 | 13,609 | 11,292 |
| PSNR | 37,701 | 36,76 | 37,821 | 36,793 | 37,603 |
| SNR | 31,797 | 27,233 | 32,102 | 32,535 | 30,924 |
| SSIM | 0,99412 | 0,97892 | 0,9563 | 0,98114 | 0,98046 |
| PBVIF | 0,80038 | 0,785 | 0,72257 | 0,79372 | 0,78988 |

Tabela 5.1: Tabela comparando o vídeo original versus o vídeo marcado.

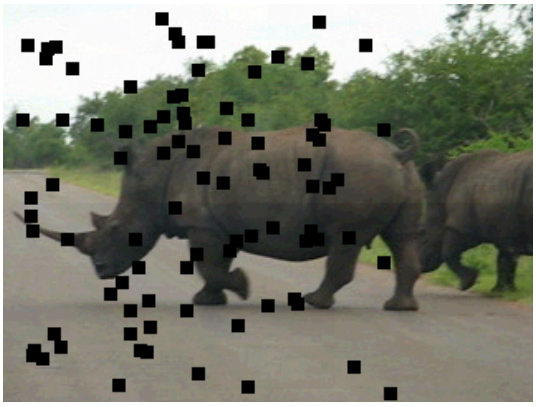
e a fidelidade de informação visual baseada em pixel (*Pixel-Based Visual Information Fidelity*, PB-VIF) [28].

Nos gráficos das Figuras 5.4-5.28, no eixo X, temos a quantidade de blocos perdidos, variando entre 30 e 120. No eixo Y, temos o valor da métrica. Em cada gráfico, é apresentada a comparação dos resultados obitdos para o vídeo original versus o vídeo marcado, o vídeo original versus o vídeo original com perdas e o vídeo original versus o vídeo marcado com perdas. Na Tabela 5.1 vemos os valores das métricas comparando o vídeo original com o vídeo com a marca d'água adicionada.

Conforme visto nos gráficos das métricas (Figuras 5.4 a 5.28), a qualidade dos quadros recuperados é visivelmente melhor que a qualidade dos quadros com perdas. Os resultados que obtivemos estão aquém do que esperávamos para quadros com perdas de poucos blocos, mas bem melhores do que esperávamos para quadros com perdas de muitos blocos. A recuperação é proporcionalmente melhor para vídeos com quadros com muitas perdas quando comparamos o vídeo com perdas e o restaurado.

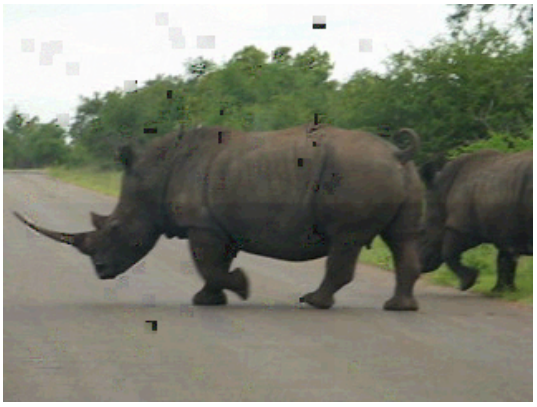
(a) Exemplos de quadros do vídeo *Rhinos* com marca

(b) Exemplos de quadros do vídeo *Paris* com marca



(c) Exemplos de quadros do vídeo *Rhinos* com perda

(d) Exemplos de quadros do vídeo *Paris* com perda



(e) Exemplos de quadros do vídeo *Rhinos* com erros mitigados

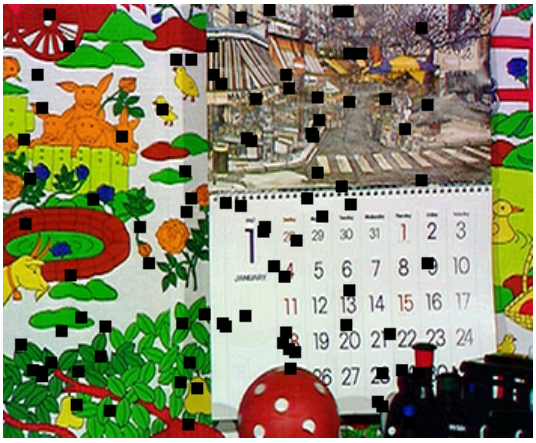
(f) Exemplos de quadros do vídeo *Paris* com erros mitigados

Figura 5.2: Quadros dos videos *Rhinos* e *Paris* marcados, com perdas e suas respectivas versões mitigadas.

(a) Exemplos de quadros do vídeo *Tempete* com marca

(b) Exemplos de quadros do vídeo *Mobile* com marca

(c) Exemplos de quadros do vídeo *Tempete* com perda



(d) Exemplos de quadros do vídeo *Mobile* com perda

(e) Exemplos de quadros do vídeo *Tempete* com erros mitigados

(f) Exemplos de quadros do vídeo *Mobile* com erros mitigados

Figura 5.3: Quadros dos videos *Tempete* e *Mobile* marcados, com perdas e suas respectivas versões mitigadas.

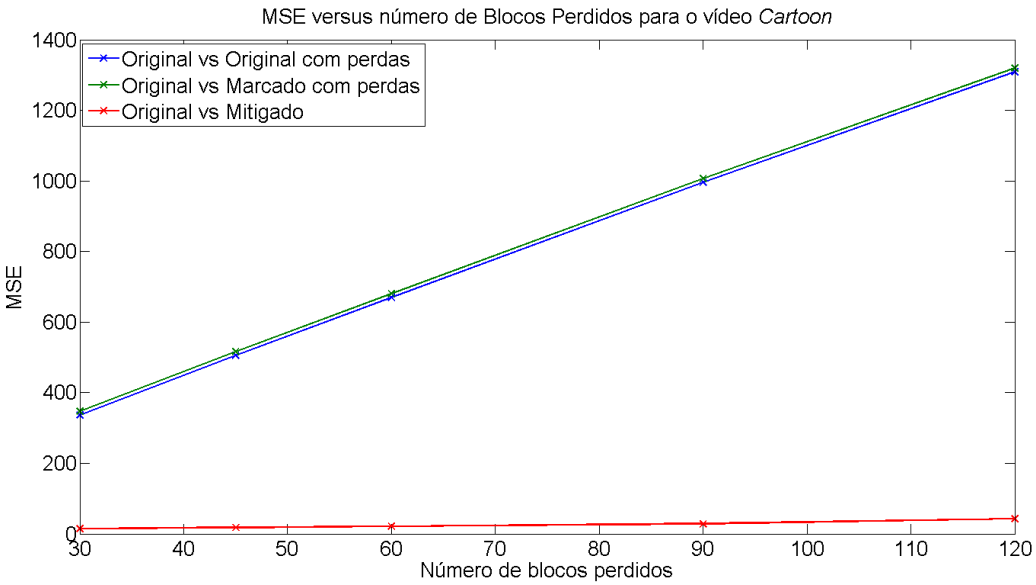



Figura 5.4: Gráfico dos valores do MSE versus número de blocos perdidos para o vídeo *Cartoon*.

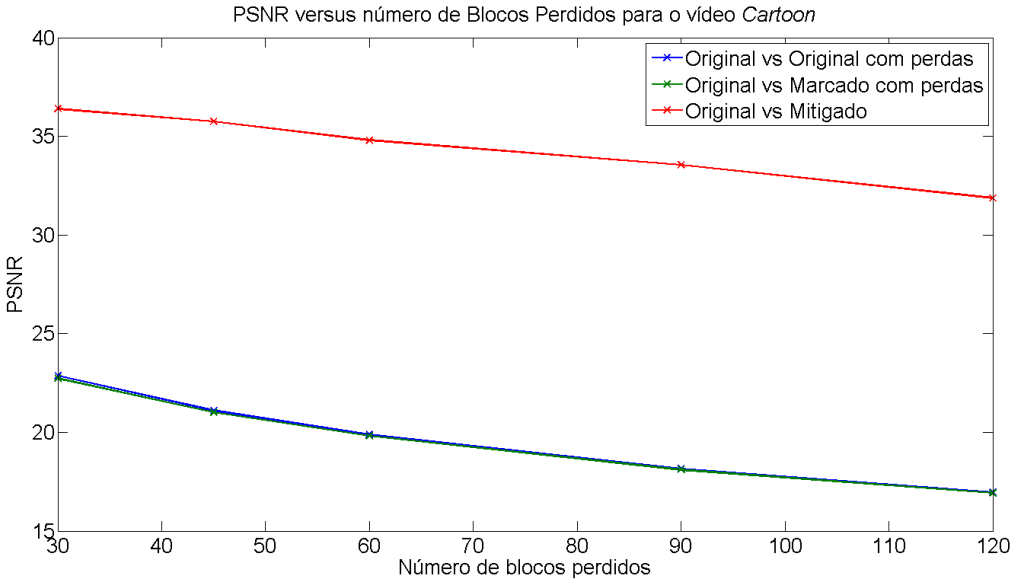



Figura 5.5: Gráfico dos valores do SSIM versus número de blocos perdidos para o vídeo *Cartoon*.

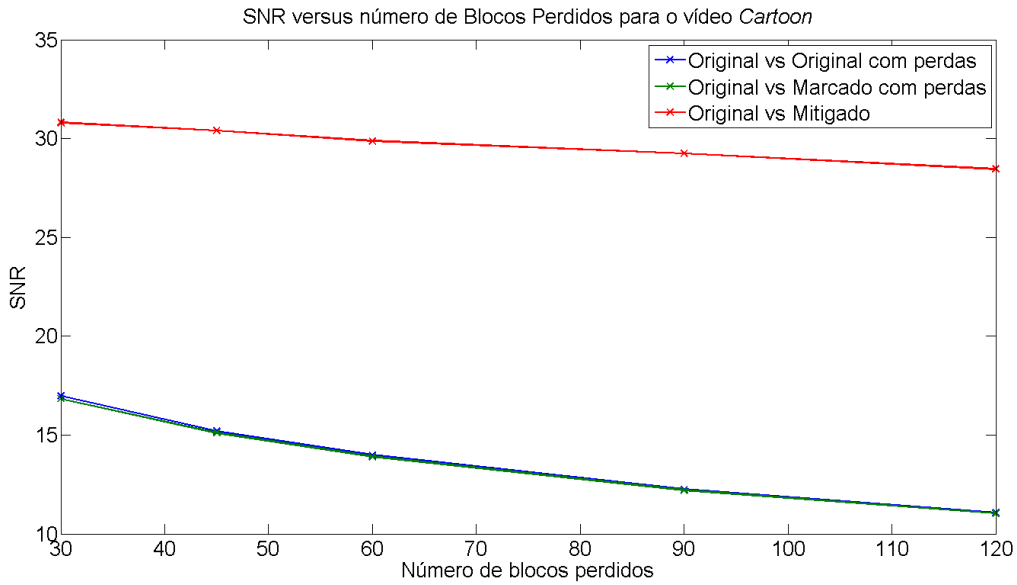



Figura 5.6: Gráfico dos valores do SNR versus número de blocos perdidos para o vídeo *Cartoon*.

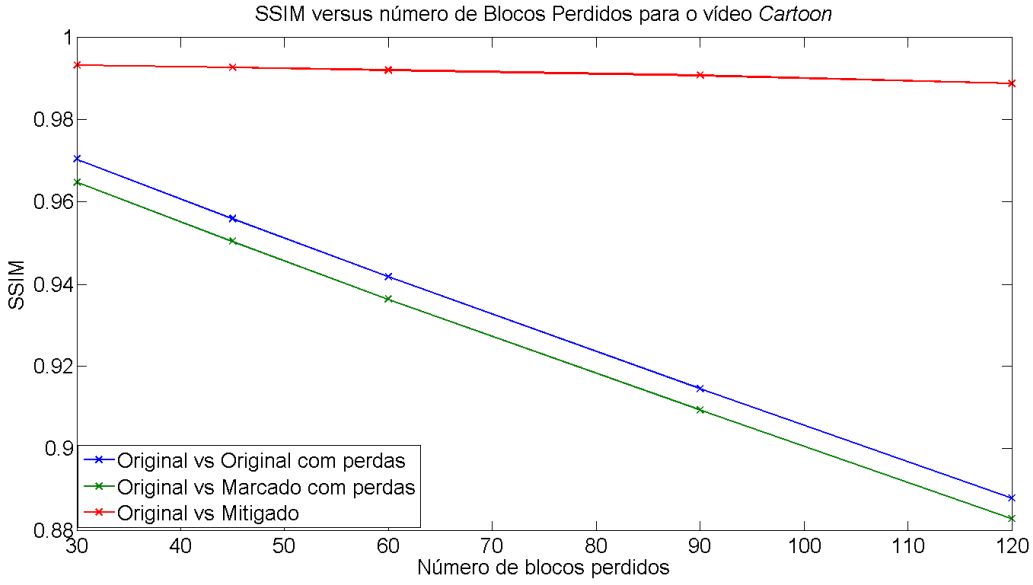



Figura 5.7: Gráfico dos valores do SSIM versus número de blocos perdidos para o vídeo *Cartoon*.

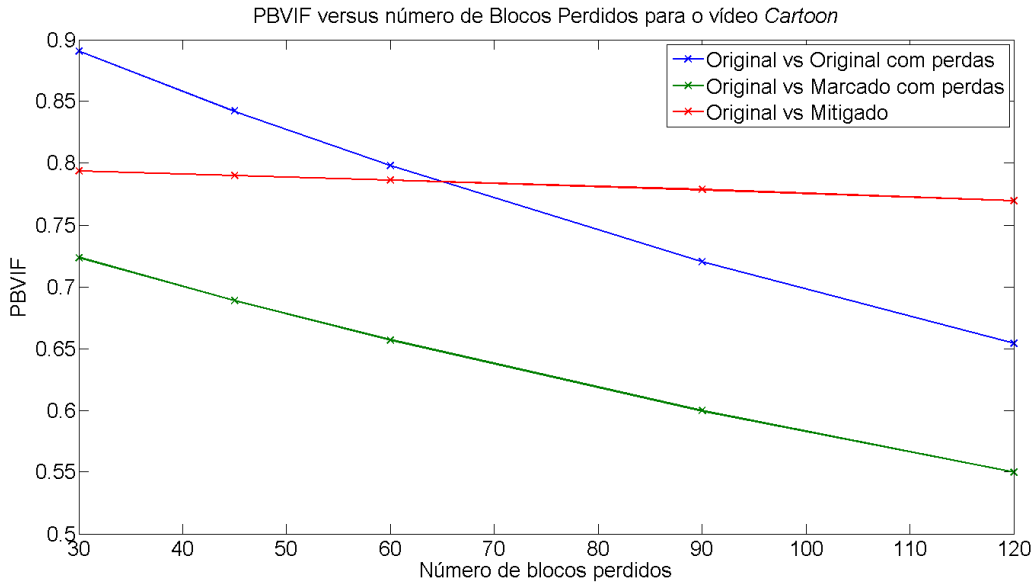



Figura 5.8: Gráfico dos valores do PBVIF versus número de blocos perdidos para o vídeo *Cartoon*.

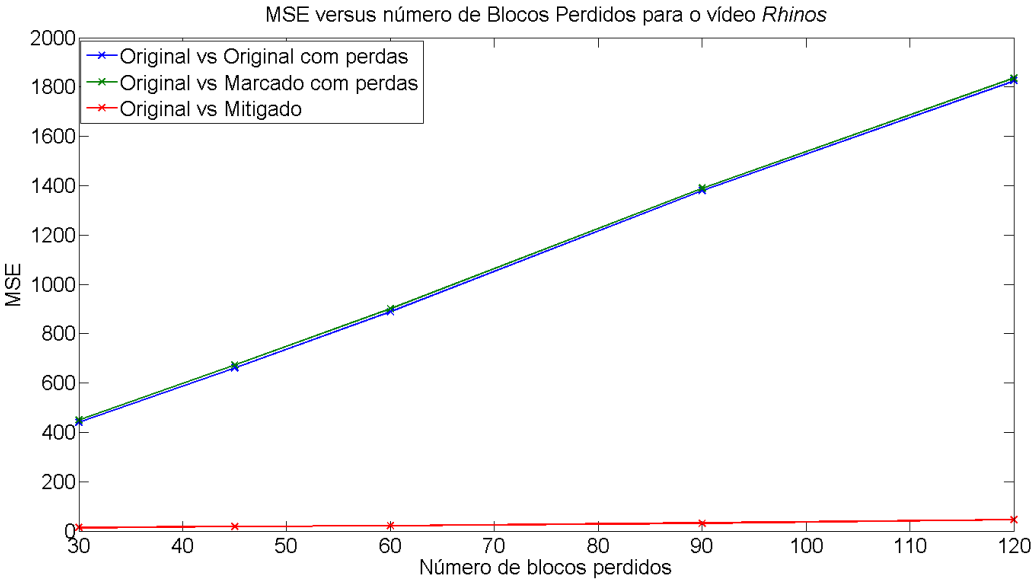



Figura 5.9: Gráfico dos valores do MSE versus número de blocos perdidos para o vídeo *Rhinos*.

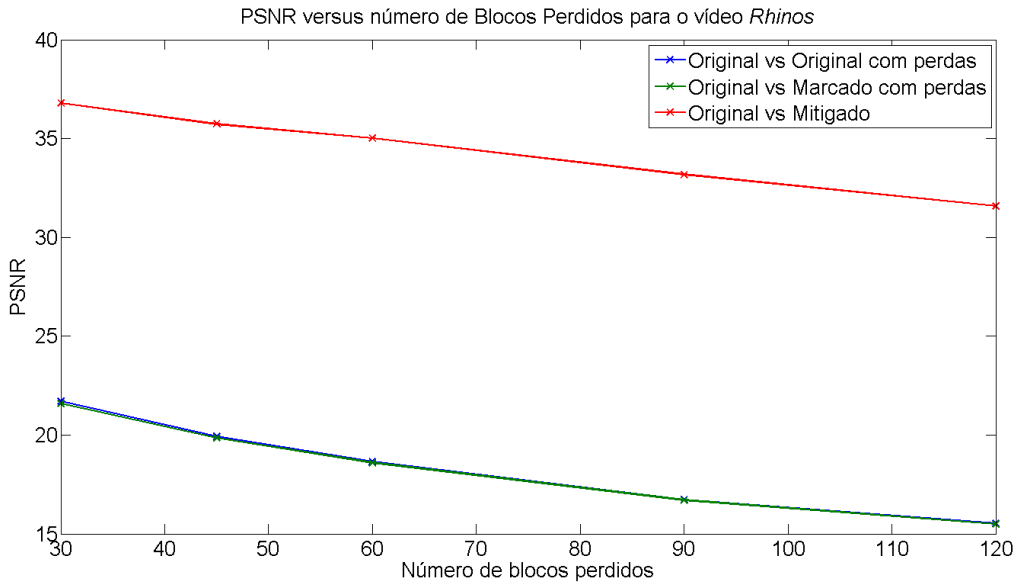



Figura 5.10: Gráfico dos valores do PSNR versus número de blocos perdidos para o vídeo *Rhinos*.

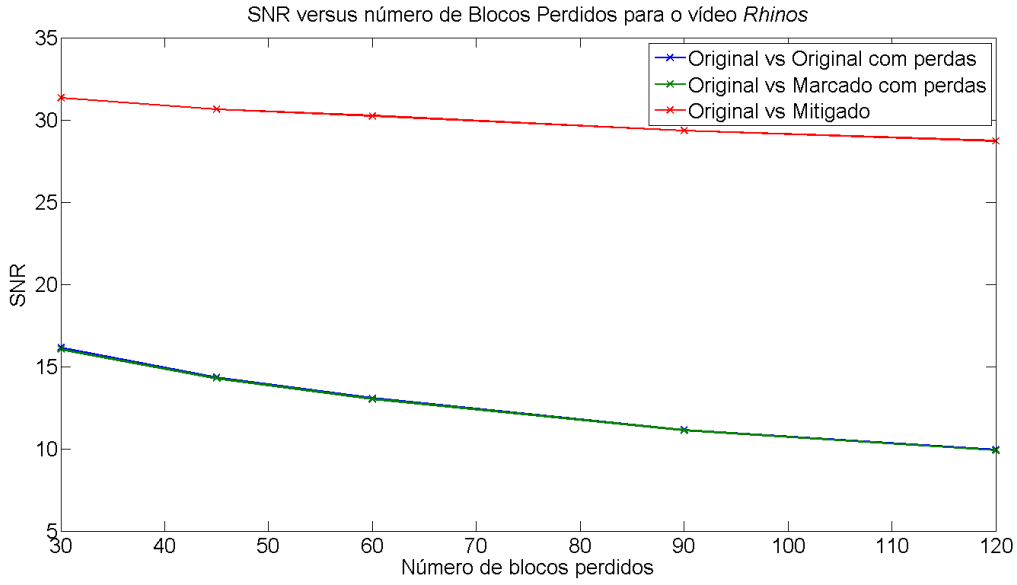



Figura 5.11: Gráfico dos valores do SNR versus número de blocos perdidos para o vídeo *Rhinos*.

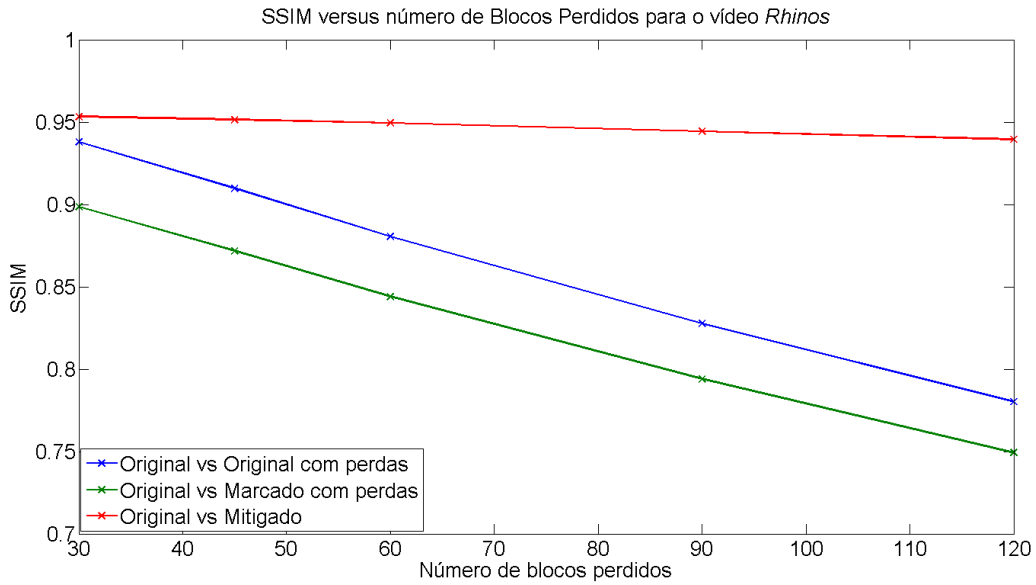



Figura 5.12: Gráfico dos valores do SSIM versus número de blocos perdidos para o vídeo *Rhinos*.

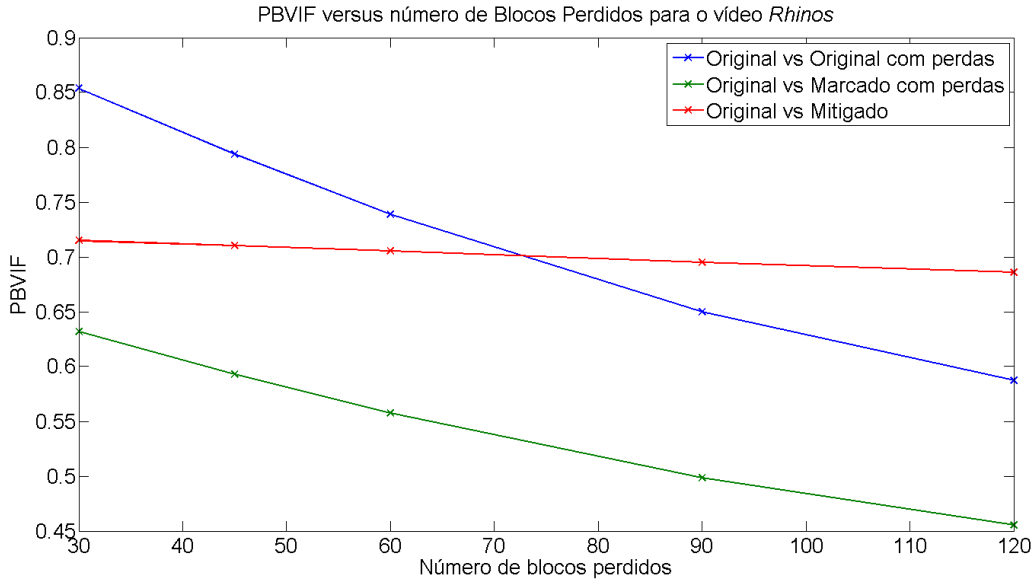



Figura 5.13: Gráfico dos valores do PBVIF versus número de blocos perdidos para o vídeo *Rhinos*.

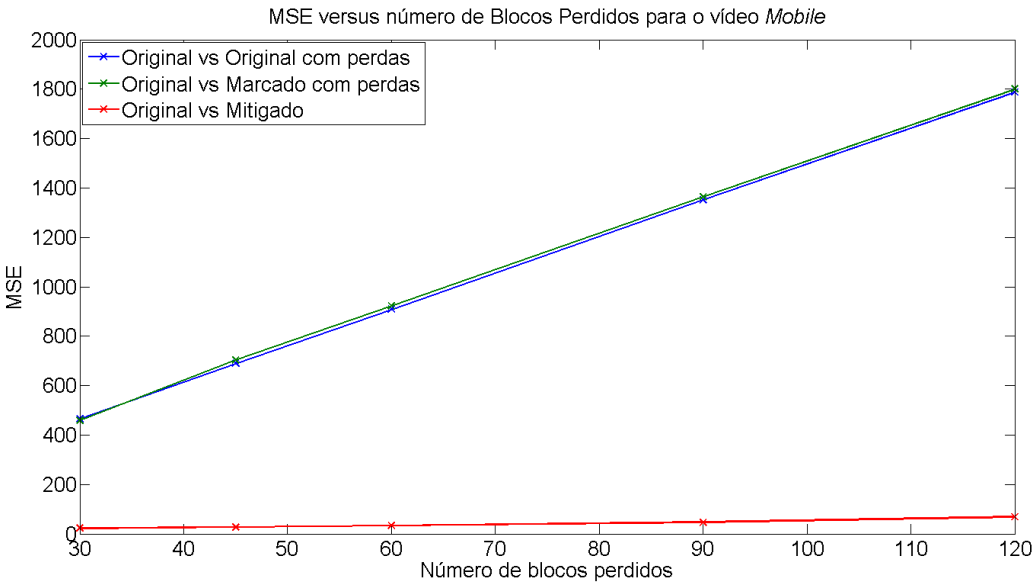



Figura 5.14: Gráfico dos valores do MSE versus número de blocos perdidos para o vídeo *Mobile*.

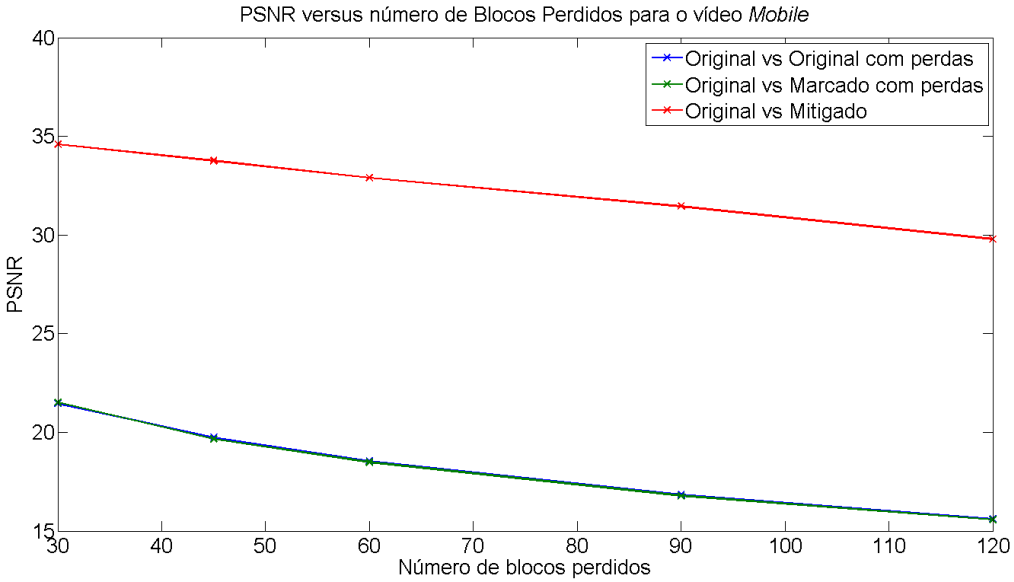



Figura 5.15: Gráfico dos valores do PSNR versus número de blocos perdidos para o vídeo *Mobile*.

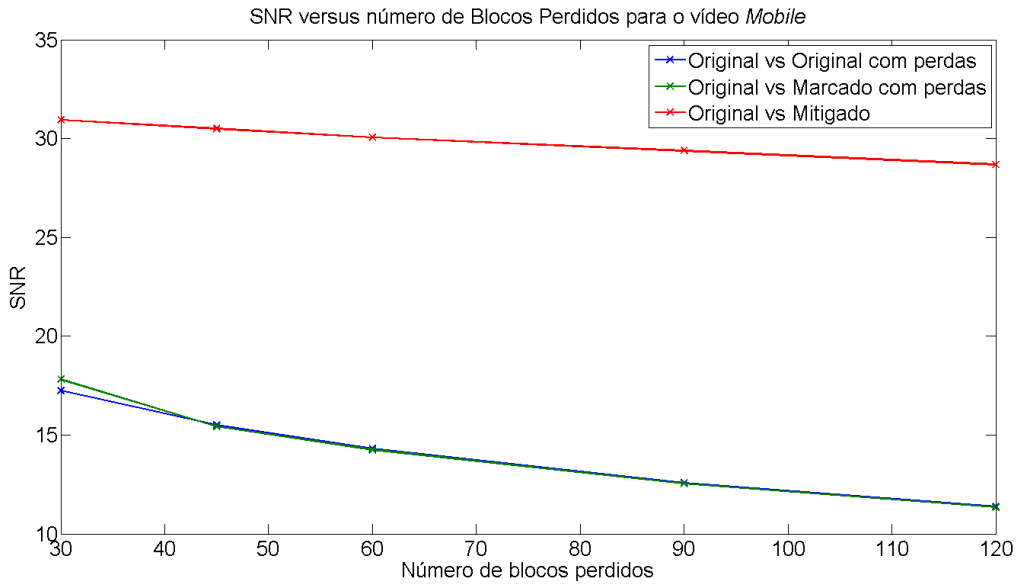



Figura 5.16: Gráfico dos valores do SNR versus número de blocos perdidos para o vídeo *Mobile*.

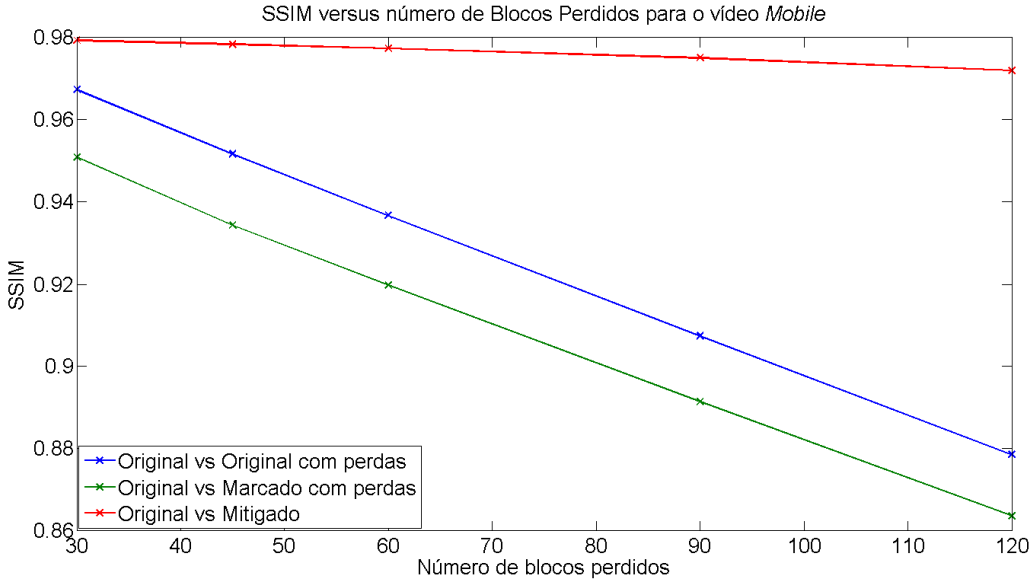



Figura 5.17: Gráfico dos valores do SSIM versus número de blocos perdidos para o vídeo *Mobile*.

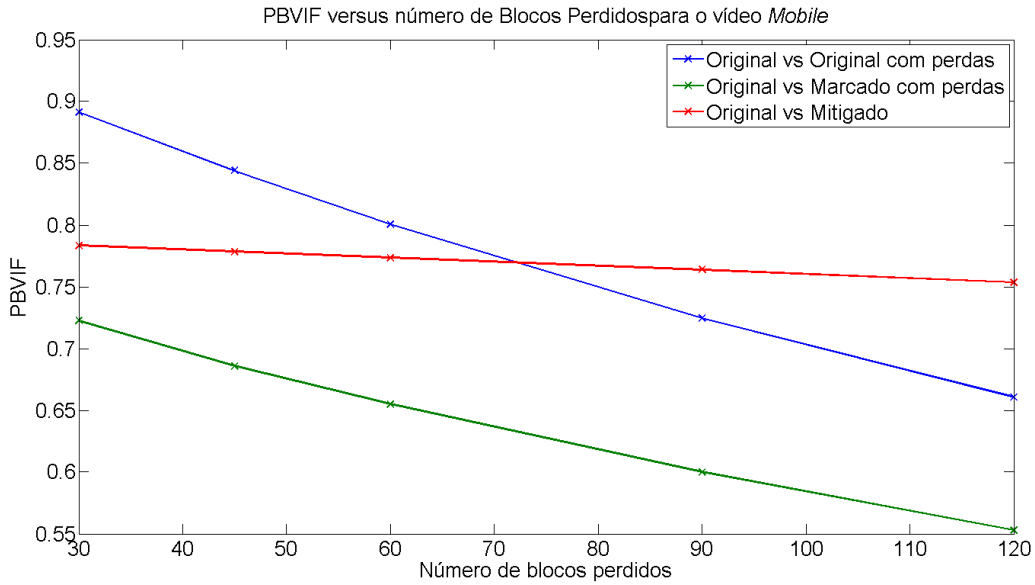



Figura 5.18: Gráfico dos valores do PBVIF versus número de blocos perdidos para o vídeo *Mobile*.

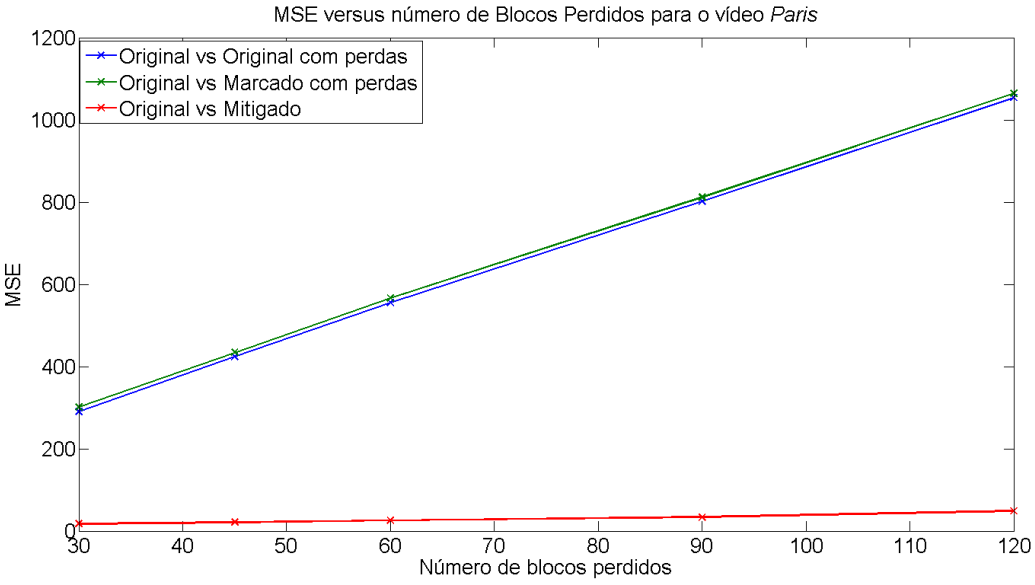



Figura 5.19: Gráfico dos valores do MSE versus número de blocos perdidos para o vídeo *Paris*.

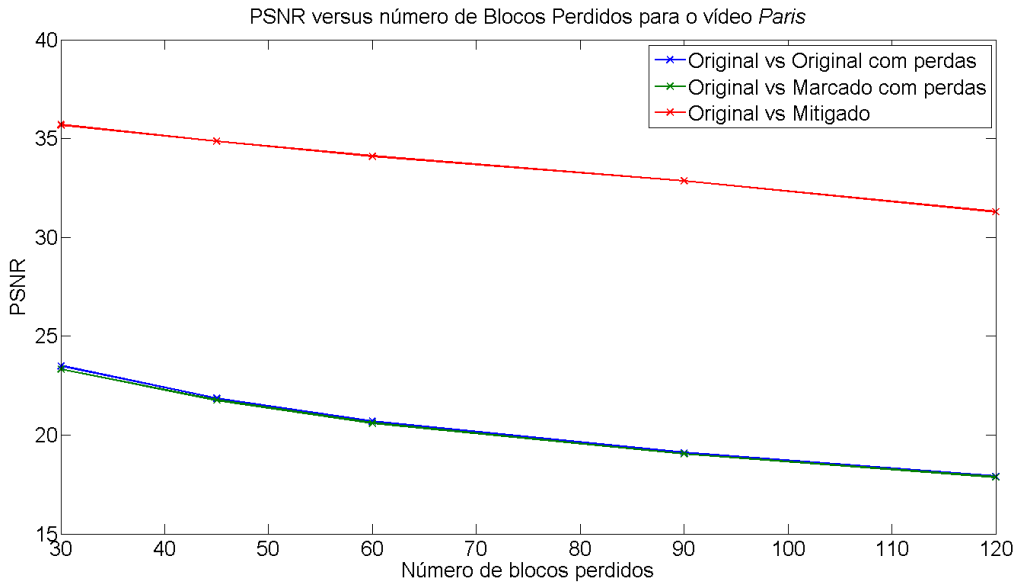



Figura 5.20: Gráfico dos valores do PSNR versus número de blocos perdidos para o vídeo *Paris*.

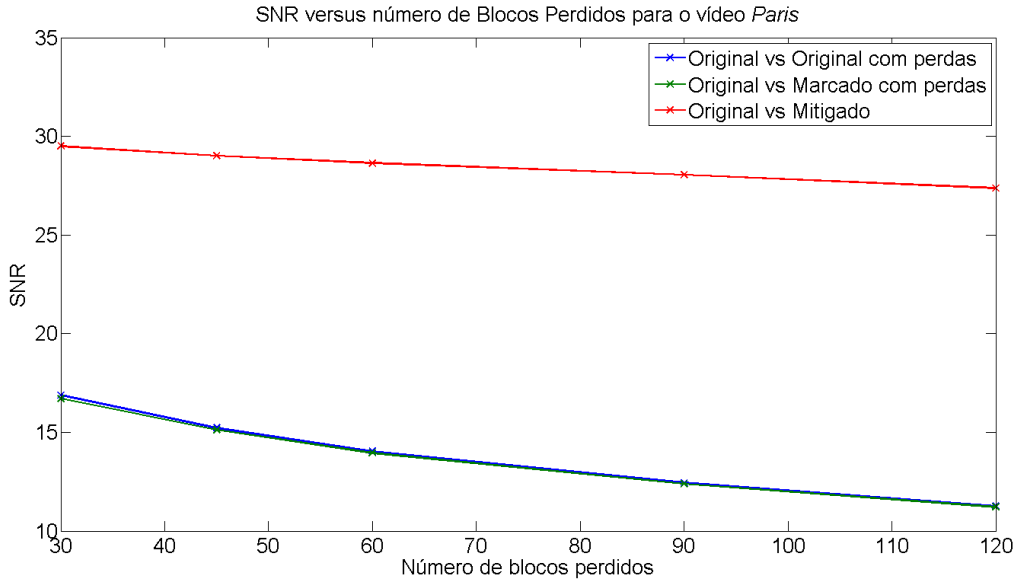



Figura 5.21: Gráfico dos valores do SNR versus número de blocos perdidos para o vídeo *Paris*.

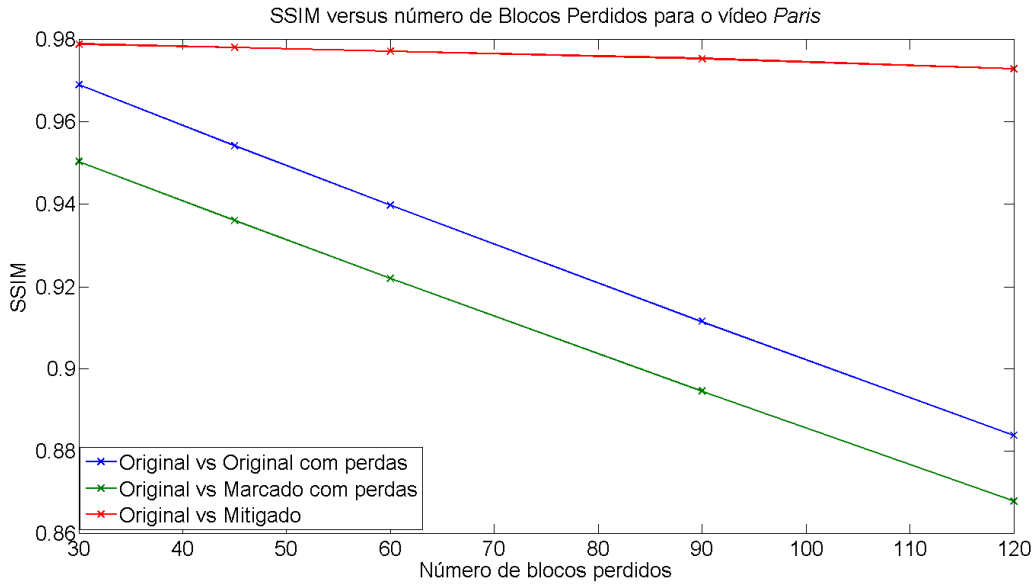



Figura 5.22: Gráfico dos valores do SSIM versus número de blocos perdidos para o vídeo *Paris*.

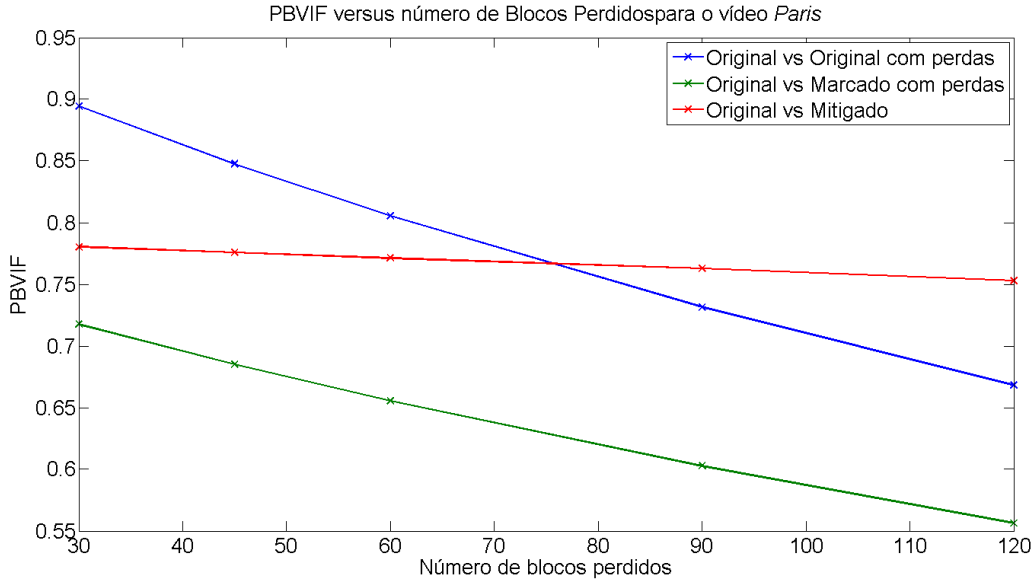



Figura 5.23: Gráfico dos valores do PBVIF versus número de blocos perdidos para o vídeo *Paris*.

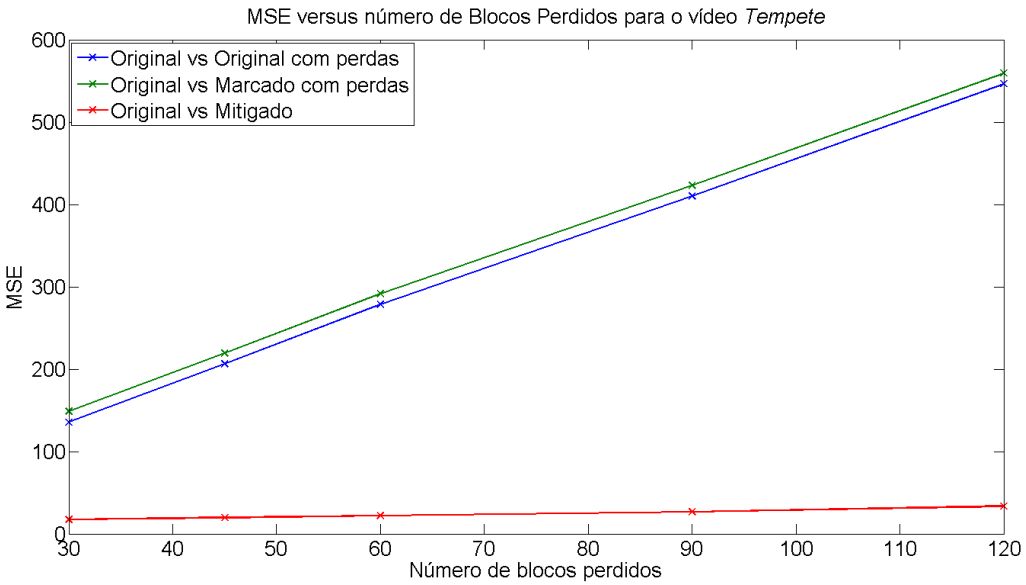



Figura 5.24: Gráfico dos valores do MSE versus número de blocos perdidos para o vídeo *Tempete*.

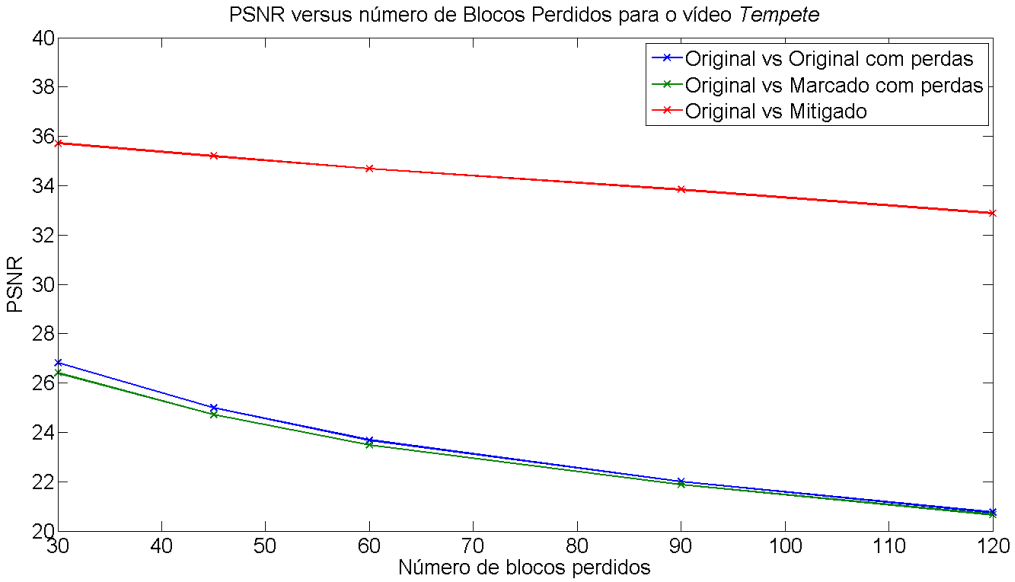



Figura 5.25: Gráfico dos valores do PSNR versus número de blocos perdidos para o vídeo *Tempete*.

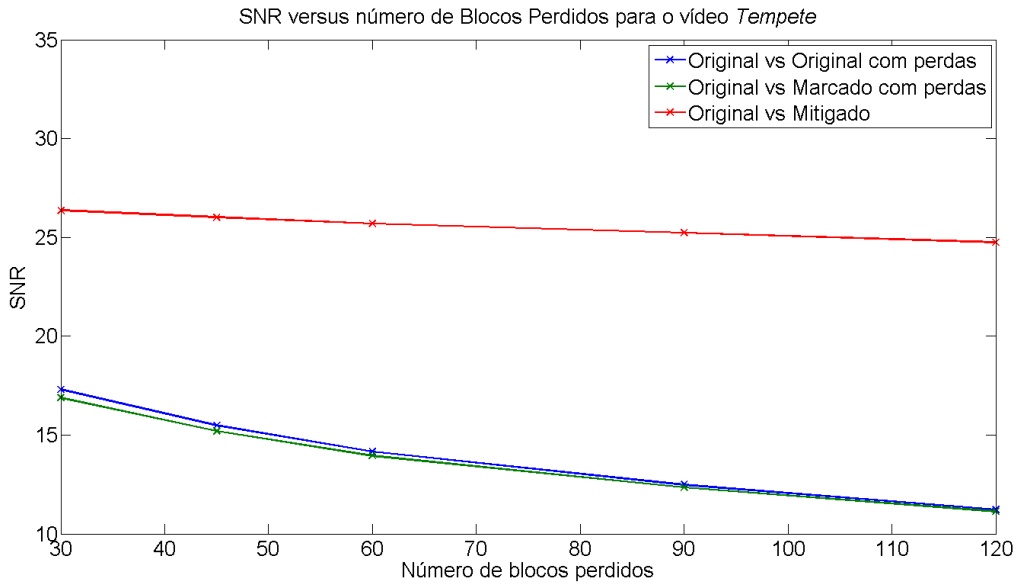



Figura 5.26: Gráfico dos valores do SNR versus número de blocos perdidos para o vídeo *Tempete*.

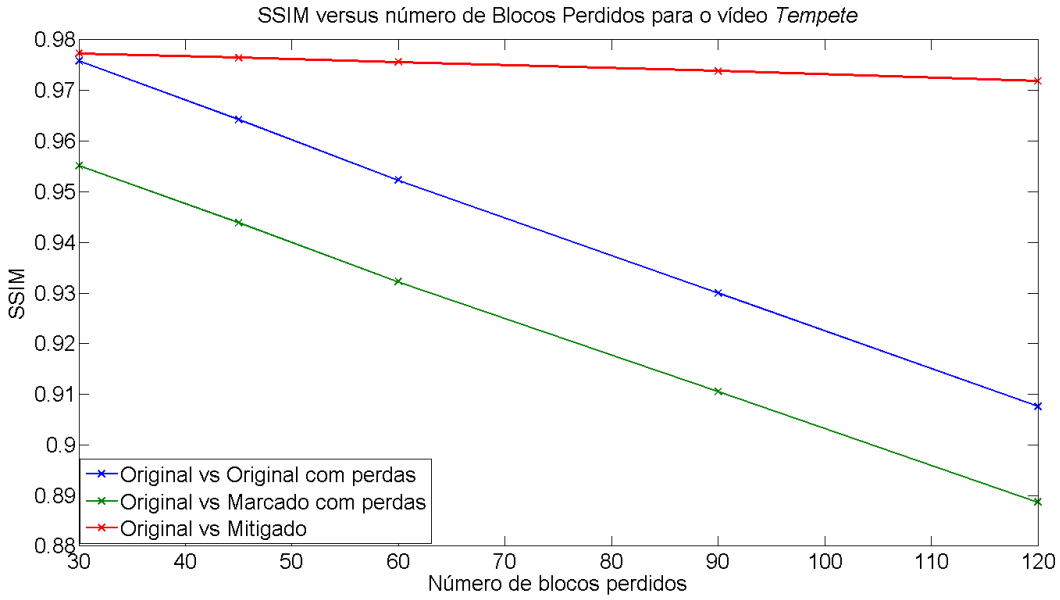



Figura 5.27: Gráfico dos valores do SSIM versus número de blocos perdidos para o vídeo *Tempete*.

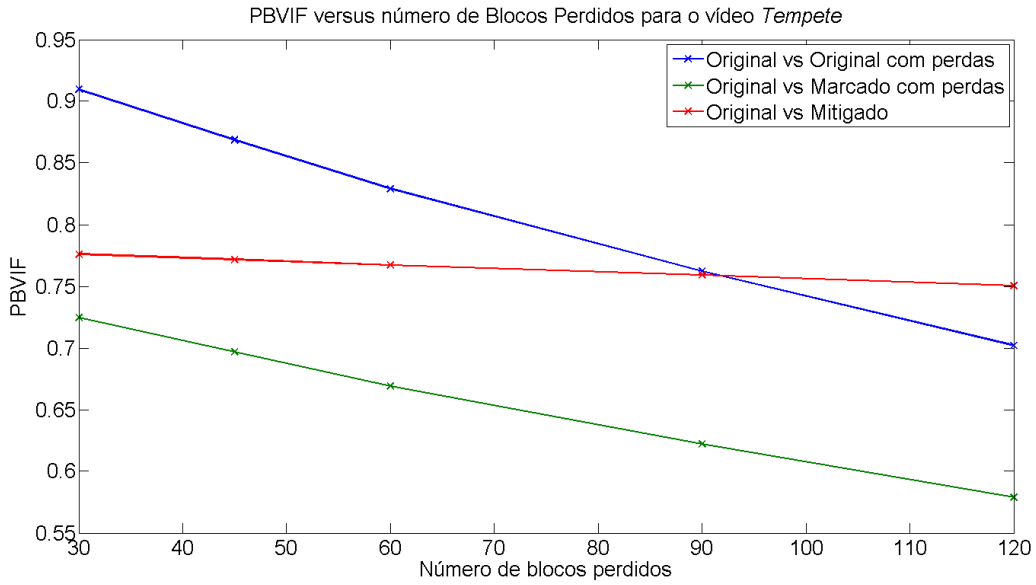



Figura 5.28: Gráfico dos valores do PBVIF versus número de blocos perdidos para o vídeo *Tempete*.

# Capítulo 6

# Conclusões e Trabalhos Futuros

O interesse por técnicas de controle e mitigação de erros tem se tornado cada vez mais populares devido a crescente demanda por aplicações que utilizam transmissão de vídeo por canais com baixa confiabilidade, como por exemplo a Internet. Como a Internet é uma rede de melhor esforço, ela não possui garantias e erros acontecem ocasionalmente. No caso específico de vídeo, este problema é agravado pelo fato dessas mídias serem normalmente comprimidas, sendo assim, uns poucos bits perdidos podem se propagar e tornar o resultado inutilizável. Esse trabalho propõe uma algoritmo para mitigação de erros que utiliza técnicas de marcas d'água e halftoning.

No codificador, utilizamos um algoritmo inspirado no algoritmo de modulação do índice de quantização (QIM) para inserir uma cópia binária de cada canal de cor, de cada quadro do vídeo, nele mesmo. A inserção da marca não causa degradação visível no vídeo. As cópias binárias são obtidas através de uma técnica de *halftoning* que foi projetada para que o processo de *inverse halftoning* gerasse uma imagem com maior qualidade possível. Estas cópias são embaralhadas antes de serem inseridas para aumentar a probabilidade de uma área perdida ter sua versão binária em outra localidade do quadro.

No decodificador, estas cópias (marcas binárias) são extraídas, filtradas por um processo de *inverse halftoning*, desembaralhadas e utilizadas para reconstruir as partes do vídeo perdidas. O algoritmo apresenta bom desempenho conseguindo restaurar com boa qualidade vídeos apresentando até 50% de perdas. Esse valor máximo de perdas se dá quando a região afetada possui suas cópias em regiões espacialmente opostas.

As técnicas do algoritmo só fazem uso de informações espaciais, porém vídeos possuem uma grande redundância temporal. Um possível trabalho futuro seria a utilização dessa informação temporal para uma melhor mitigação dos erros no vídeo. Em outras palavras, podemos usar a informação de quadros anteriores e posteriores para recuperar regiões afetadas do quadro em questão.

As técnicas propostas neste trabalho também podem ser utilizadas em outras aplicações. Uma possibilidade é a utilização da técnica em questão para detecção de *tampering*, ou seja a verificação da integridade do conteúdo. Especificamente, no codificador, uma marca seria inserida da mesma forma como é no algoritmo atual. No decodificador, a marca seria extraída e comparada ao o conteúdo. Caso diferenças sejam detectadas, o algoritmo seria capaz de identificar as regiões modificadas e reconstruir a informação original do conteúdo em questão.

# Referências


[1] Scientific tools for python. "[Online; accessed 12/01/201]". 20

[2] C.B. Adsumilli, M.C.Q. Farias, S.K. Mitra, and M. Carli. A robust error concealment technique using data hiding for image and video transmission over lossy channels. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(11):1394–1406, november 2005. 1, 2

[3] R. Aravind, M.R. Civanlar, and A.R. Reibman. Packet loss resilience of mpeg-2 scalable video coding algorithms. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(5):426–435, 1996. 13

[4] M. Barni, F. Bartolini, R. Caldelli, A. De Rosa, and A. Piva. A robust watermarking approach for raw video. In *Proceedings of the 10th international packet video workshop*, 2000. 5

[5] A.C. Bovik. *Handbook of image and video processing.* Academic Press, 2000. 6

[6] B. Chen and G.W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *Information Theory, IEEE Transactions on*, 47(4):1423–1443, May 2001. 2, 6, 7

[7] T.Y. Chung, M.S. Hong, Y.N. Oh, D.H. Shin, and S.H. Park. Digital watermarking for copyright protection of mpeg2 compressed video. *Consumer Electronics, IEEE Transactions on*, 44(3):895–901, 1998. 6

[8] I.J. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for images, audio and video. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 243–246, 1996. 4

[9] N. Farvardin and V. Vaishampayan. Optimal quantizer design for noisy channels: An approach to combined source-channel coding. *Information Theory, IEEE Transactions on*, 33(6):827–838, 1987. 16

[10] P. Ferguson and G. Huston. Quality of service: delivering qos on the internet and in corporate networks. *Recherche*, 67:02, 1998. 12

[11] P. Garcia Freitas, M.C.Q. de Farias, and A.P.F. de Araújo. Fast inverse halftoning algorithm for ordered dithered images. *SIBGRAPI*, 2011. 8, 22

[12] F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, July 1999. 3

[13] M.J. Holliman, N.D. Memon, B.L. Yeo, and M.M. Yeung. Adaptive public watermarking of dct-based compressed images. In *Proceedings of SPIE*, volume 3312, page 284, 1997. 6

[14] D.A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. 9

[15] M. Khansari, A. Jalali, E. Dubois, and P. Mermelstein. Low bit-rate video transmission over fading channels for wireless microcellular systems. *Circuits and Systems for Video Technology, IEEE Transactions on*, 6(1):1–11, 1996. 13

[16] D.E. Knuth. Digital halftones by dot diffusion. *ACM Transactions on Graphics (TOG)*, 6(4):245–273, 1987. 8

[17] A. Kurtenbach and P. Wintz. Quantizing for noisy channels. *Communication Technology, IEEE Transactions on*, 17(2):291–302, 1969. 16

[18] G.C. Langelaar, I. Setyawan, and R.L. Lagendijk. Watermarking digital image and video data. a state-of-the-art overview. *Signal Processing Magazine, IEEE*, 17(5):20–46, 2000. 5

[19] S. Lin and DJ Costello Jr. Error control coding: Fundamentals and applications, 1983. *Prentice-Hall, Inc. Englewood Cliffs*, 3:2–3, 1983. 11

[20] J. Linnartz and J. Talstra. Mpeg pty-marks: Cheap detection of embedded copyright data in dvd-video. *Computer Security ESORICS 98*, pages 221–240, 1998. 6

[21] R. Marasli, P.D. Amer, and P.T. Conrad. Retransmission-based partially reliable transport service: An analytic model. In *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, volume 2, pages 621–629, 1996. 19

[22] K. Ngan and R. Steele. Enhancement of pcm and dpcm images corrupted by transmission errors. *Communications, IEEE Transactions on*, 30(1):257–265, 1982. 11

[23] N. Ohta. *Packet video: modeling and signal processing.* Artech House, Inc., 1994. 11

[24] H. Pedrini and W.R. Schwartz. *Análise de imagens digitais: princípios, algoritmos e aplicações.* Thomson Learning, 2008. 8

[25] F. Perez-Gonzalez and J.R. Hernandez. A tutorial on digital watermarking. In *Security Technology, 1999. Proceedings. IEEE 33rd Annual 1999 International Carnahan Conference on*, pages 286–292, 1999. 3

[26] C. Pik-Wah. Digital video watermarking techniques for secure multimedia creation and delivery. Master's thesis, The Chinese University of Hong Kong, july 2004. 5, 6

[27] J. Rissanen and G.G. Langdon. Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162, 1979. 9

[28] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *Image Processing, IEEE Transactions on*, 15(2):430–444, 2006. 28

[29] B.C. Smith. *Implementation techniques for continuous media systems and applications*. PhD thesis, University of California at Berkeley, 1994. 19

[30] J.J. Spilker Jr. Digital communications by satellite. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1977. 685 p.*, 1, 1977. 15

[31] H. Sun and W. Kwok. Concealment of damaged block transform coded images using projections onto convex sets. *Image Processing, IEEE Transactions on*, 4(4):470–477, 1995. 18

[32] M.D. Swanson, B. Zhu, B. Chau, and A.H. Tewfik. Multiresolution video watermarking using perceptual models and scene segmentation. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, pages 558–561, 1997. 5

[33] AS Tom, CL Yeh, and F. Chu. Packet video for cell loss protection using deinterleaving and scrambling. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 2857–2860, 1991. 15

[34] M. Wada. Selective recovery of video packet loss using error concealment. *Selected Areas in Communications, IEEE Journal on*, 7(5):807–814, 1989. 18

[35] Y. Wang and Q.F. Zhu. Error control and concealment for video communication: A review. *Proceedings of the IEEE*, 86(5):974–997, 1998. 9, 14, 15, 16, 17, 18

[36] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing, IEEE Transactions on*, 13(4):600–612, 2004. 27

[37] P. Yin, B. Liu, and H.H. Yu. Error concealment using data hiding. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 3, pages 1453–1456, may 2001. 1

[38] Q.F. Zhu. Device and method of signal loss recovery for realtime and/or interactive communications, August 27 1996. US Patent 5,550,847. 19

[39] Q.F. Zhu, Y. Wang, and L. Shaw. Coding and cell-loss recovery in dct-based packet video. *Circuits and Systems for Video Technology, IEEE Transactions on*, 3(3):248–258, 1993. 17