UM MÉTODO DE OCULTAMENTO DE ERROS EM TRANSMISSÃO DE VÍDEO

Ana Luísa de Araujo Santos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Ricardo Lopes de Queiroz, Ph.D.

Prof. Gelson Vieira Mendonça, Ph.D.

Prof. Marco Antonio Grivet Mattoso Maia, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2006

SANTOS, ANA LUÍSA DE ARAUJO

Um Método de Ocultamento de Erros em Transmissão de Vídeo [Rio de Janeiro] 2006

X, 91 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1.Processamento de Sinais

2.H.264

3.MPEG-4

4.Ocultamento de Erros

I.COPPE/UFRJ II.Título (série)

Agradecimentos

Inicialmente, agradeço e dedico este trabalho ao meu pai, Sergio, por sua extrema generosidade e apoio às minhas decisões. À minha mãe, Ilka, por seu amor e dedicação. Às minhas avós Cecilia e Eunice, e aos meus padrinhos Cléia e Manuel, pela base de valores e apoio incondicional em toda a minha vida.

Ao meu namorado, Miguel, por compreender minhas ausências e angústias, e me amar apesar das minhas falhas.

Ao Prof. Eduardo A. B. da Silva, meu orientador, por me incentivar e acreditar que eu seria capaz. Agradeço imensamente sua paciência, amizade e atenção. Sua dedicação aos alunos, disciplina, empolgação e enorme disposição em compartilhar conhecimentos são um exemplo para mim.

Ao Prof. Ricardo Queiroz, meu orientador, pela compreensão e também pelas discussões que muito contribuíram para este trabalho e futuros desdobramentos.

Aos professores membros da banca examinadora, Marco Grivet e Gelson Mendonça, por sua paciência e disposição em avaliar este trabalho em um curto prazo.

Ao meu amigo e grande incentivador, Tadeu Ferreira, por suas explicações, idéias, conselhos, companheirismo, e principalmente por sua preocupação e ombro amigo nos momentos mais difíceis.

Ao amigo Leonardo Baltar, simplesmente por estar ao meu lado, me ajudando a acreditar que seria possível.

Aos amigos José Fernando Leite, pela paciência e importantes contribuições na programação; Alessandro J. S. Dutra, Lisandro Lovisolo e Nuno Rodrigues, por suas idéias e disposição em esclarecer muitas das minhas dúvidas.

Aos amigos do LPS, em especial aos que me acompanharam nos anos do mestrado: Marcello Artimos, Filipe Diniz, Michel Tcheou, Carlo Marcello Siqueira, Fábio Freeland, Leonardo Baltar e Tadeu Ferreira.

Aos professores, alunos e funcionários do Laboratório de Processamento de Sinais da COPPE/UFRJ, por proporcionarem um ambiente de trabalho agradável e produtivo.

Meu agradecimento também ao LPS, pela infra-estrutura, e à CAPES, pela bolsa de mestrado, sem os quais não teria sido possível realizar este trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM MÉTODO DE OCULTAMENTO DE ERROS EM TRANSMISSÃO DE VÍDEO

Ana Luísa de Araujo Santos

Março/2006

Orientadores: Eduardo Antônio Barros da Silva
Ricardo Lopes de Queiroz

Programa: Engenharia Elétrica

Esta dissertação apresenta e analisa técnicas usadas para minimizar o impacto de erros na transmissão de vídeo codificado. O foco é direcionado para os métodos de ocultamento de erros (*error concealment*) implementados na fase de decodificação.

Primeiramente é apresentada a demanda por compressão de vídeo robusta a erros. Os principais métodos de ocultamento de erros encontrados na literatura também são abordados, com ênfase nas técnicas usadas para minimizar o efeito visual de macroblocos corrompidos no sinal de vídeo decodificado.

Em seguida, é proposto um método de ocultamento de erros em macroblocos baseado na coerência de movimento de macroblocos vizinhos. O método proposto é aplicado ao padrão H.264/AVC, estado da arte em compressão de vídeo. Ao final são analisados resultados experimentais comparativos com respeito às técnicas de ocultamento de erros implementadas pelo *software* de referência do padrão.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AN ERROR CONCEALMENT METHOD FOR VIDEO TRANSMISSION

Ana Luísa de Araujo Santos

March/2006

Advisors: Eduardo Antônio Barros da Silva
Ricardo Lopes de Queiroz

Department: Electrical Engineering

This dissertation presents and analyses techniques to minimize the impact of errors on the transmission of coded video. Our focus is on error concealment methods at the decoder side.

The demand for error resilient video compression techniques is briefly presented. The most popular error concealment methods are also discussed, focusing on decoder-side techniques to reduce visibility of corrupted macroblocks.

A new error concealment method based on motion coherence is proposed for macroblock recovery. The method is applied to the H.264/AVC state-of-the-art video compression standard. Finally, experimental results are analysed, comparing the proposed method to the method in the H.264/AVC reference software.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Atualmente, diversas aplicações demandam a transmissão de grandes quantidades de informação. Dentre os diversos tipos de fonte de informação, o vídeo é provavelmente um dos que produz maior quantidade de dados [1]. Dessa forma, é muito importante utilizar técnicas de compressão para se transmitir vídeo.

No caso do vídeo, uma sucessão de imagens estáticas, apresentando poucas diferenças, é percebida como um movimento contínuo devido ao fenômeno da persistência visual [2]. A compressão de vídeo faz uso dessas correlações espaço-temporais para transmitir somente as diferenças entre as imagens, gerando uma quantidade menor de *bits* para representar o vídeo. Essa abordagem faz uso de técnicas de predição, além de estimação e compensação de movimentos, sendo bastante utilizada nos padrões de compressão atuais [3–6].

De forma geral, ao diminuir o volume de dados necessários para representar um sinal de vídeo, as técnicas de compressão acabam por reduzir a quantidade de redundância da informação. Contudo, a redução de redundâncias apresenta a desvantagem de tornar a transmissão de vídeo comprimido mais vulnerável às interferências de canal.

O controle de erros em comunicações de vídeo é um problema que apresenta diversos desafios. Sinais de vídeo comprimido são bastante sensíveis a erros de transmissão, principalmente devido ao uso de codificação preditiva e de códigos de comprimento variável. A predição espaço-temporal torna o vídeo codificado extremamente vulnerável aos erros, já que a alta correlação entre amostras gera a propagação do erro de uma única amostra para várias amostras seguintes. Os erros

em códigos de comprimento variável, por sua vez, podem levar à perda de sincronismo do decodificador, e à conseqüente inutilização de trechos consideráveis do sinal codificado [7, 8].

Apesar das dificuldades descritas acima, a robustez a erros é um requisito essencial para aplicações que fazem uso de comunicações de vídeo, tais como radiodifusão (*broadcasting*) de TV digital, transmissão em alta definição, *streaming* por rede de pacotes ou terminais móveis, além de comunicação bidirecional em vídeotelefonia ou vídeo-conferência. Portanto, um sistema de codificação de vídeo deve satisfazer aos critérios de fidelidade do vídeo decodificado e de limitações do canal, levando em consideração as restrições de atraso da comunicação e complexidade computacional para cada aplicação em questão [9].

Na literatura foram propostas diversas técnicas para atacar o problema de erros na transmissão de vídeo codificado, que podem ser classificadas como [7, 8]: detecção e correção de erros, ou detecção e ocultamento de erros (*error concealment*). As técnicas de ocultamento de erros, por sua vez, apresentam duas classes distintas: as que introduzem mudanças no fluxo de *bits* codificado (*bit stream*), e as que não o modificam.

Esta dissertação tem como foco técnicas de ocultamento de erros aplicadas ao H.264, o padrão internacional estado da arte em compressão de vídeo. No caso do H.264, a norma especifica somente sintaxe e semântica do fluxo de *bits* codificado.

O método de ocultamento de erros proposto nesta dissertação é perfeitamente compatível com o padrão H.264, pois modifica o processo de decodificação sem introduzir mudanças no fluxo de *bits* codificado, nem requerer o uso de retransmissões. Baseada em critérios espaciais e temporais, essa técnica se propõe a regenerar macroblocos perdidos no processo de transmissão, a fim de evitar a propagação desses erros e minimizar seus impactos visuais na seqüência decodificada. O método proposto se baseia em critérios de melhor casamento de conglomerados (*clusters*) de regiões vizinhas ao macrobloco perdido, com respeito ao quadro de referência. O conjunto de vetores de movimento candidatos a recuperar o macrobloco é definido pelo fecho convexo obtido do agrupamento dos vetores de movimento das regiões vizinhas. Esse método é descrito nesta dissertação, e os resultados da sua implementação são avaliados comparativamente com respeito ao método implementado

no decodificador do *software* de referência do padrão H.264 [10, 11].

No capítulo 2 é apresentada a problemática de compressão de vídeo robusta a erros, introduzindo os conceitos teóricos gerais do problema. São descritos brevemente a codificação de fonte, a robustez a erros em compressão de vídeo, o padrão H.264, as diferentes abordagens de técnicas de ocultamento de erros, além dos princípios de ocultamento de erros na fase de decodificação, foco desta dissertação.

O capítulo 3 apresenta uma revisão bibliográfica de técnicas de ocultamento de erros na decodificação, com ênfase nas técnicas experimentais implementadas no *software* de referência do padrão H.264. São apresentadas também as adaptações desse *software* usadas para se obter resultados comparativos com o método proposto nesta dissertação.

O capítulo 4 apresenta o método de ocultamento de erros na etapa de decodificação proposto para o padrão H.264, as justificativas de tratamento do problema, e os resultados experimentais obtidos.

O capítulo 5 apresenta as conclusões desta dissertação, com comentários a respeito dos resultados obtidos, e também propostas de trabalhos futuros.

Capítulo 2

Robustez a Erros em Transmissão de Vídeo

Neste capítulo são introduzidos os conceitos teóricos do problema de ocultamento de erros no padrão H.264.

Primeiramente são descritos os conceitos de codificação de fonte em um sistema de comunicação e o problema de robustez a erros (*error resilience*) na transmissão de sinais de vídeo comprimidos.

Em seguida, é apresentado o padrão H.264, estado da arte em compressão de vídeo. Seu funcionamento elementar, características e parâmetros comparativos com relação aos padrões anteriores são então descritos.

Por fim, são apresentadas as técnicas utilizadas no ocultamento de erros (*error concealment*) causados pela transmissão de vídeo comprimido por canais suscetíveis a erros.

2.1 Codificação de Fonte

Codificação de fonte consiste em um dos sub-sistemas que compõem um sistema de comunicação. Um sistema de comunicação digital envolve a transmissão de informação em forma digital gerada por uma fonte, para um ou mais destinatários [12, 13]. Os elementos que compõem um sistema de comunicação são apresentados na Figura 2.1, e descritos brevemente em seguida.

Na Figura 2.1, a saída da fonte de informação pode representar *a priori*

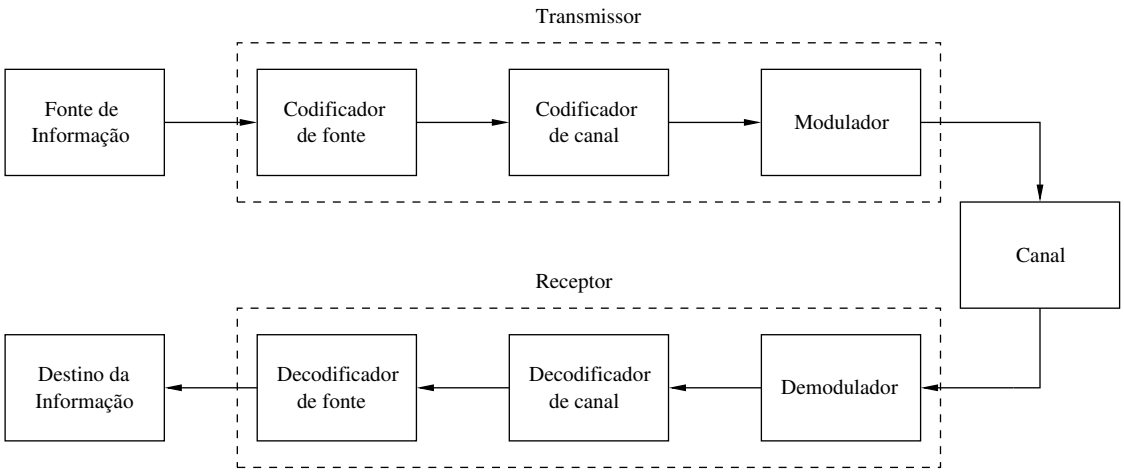


Figura 2.1: Elementos básicos de um sistema de comunicações.

um sinal digital ou analógico. Contudo, em um sistema de comunicação digital, o conteúdo produzido pela fonte é convertido em uma seqüência de dígitos binários (*bits*). Idealmente, busca-se representar esse conteúdo com o menor número possível de *bits*. Para tal são usadas técnicas de codificação de fonte, também denominadas de compressão de dados, que convertem a informação da fonte em uma seqüência contendo menos bits do que na fonte original.

Em seguida, o codificador de canal introduz redundâncias na seqüência de informação, que poderão ser usadas pelo receptor para corrigir efeitos de ruídos ou interferências durante a transmissão através do canal. Do ponto de vista do receptor, a adição de redundâncias aumenta a confiabilidade do sinal recebido, auxiliando o processo de decodificação.

Após a codificação de canal, o sinal é tratado pelo modulador digital, que converte a seqüência binária em formas de onda que serão de fato transmitidas. O canal de comunicação é o meio físico usado para enviar o sinal do transmissor para o receptor. Esse meio pode ser sem fio (*wireless*) ou com fio. Em ambos os casos, o canal afeta o sinal transmitido, introduzindo algum tipo de ruído na comunicação.

No receptor, as etapas ocorridas na transmissão ocorrem de forma inversa. O demodulador digital processa as formas de onda corrompidas pelo canal, recuperando os símbolos resultantes da codificação de canal. Esses, por sua vez, passam pelo decodificador de canal, que busca reconstruir a seqüência de informação original, de posse do código utilizado e das redundâncias contidas nos dados recebidos. Em seguida, o decodificador de fonte reconstrói uma aproximação do sinal originalmente

codificado no transmissor, que pode apresentar diferenças em relação ao sinal original em função de distorções introduzidas pelas etapas do sistema de comunicação digital.

Esta dissertação tem como foco as técnicas utilizadas nas etapas de codificação e decodificação de fonte. A fonte em questão se refere às informações que se deseja transmitir através de um sistema de comunicação. Esse conteúdo pode representar texto, imagens, voz, vídeo, ou outros tipos de informação.

Reiterando, a codificação objetiva a redução do número de *bits* necessários para se representar um conteúdo em formato digital. Desta forma, a compressão de dados é a arte ou ciência de representar informações de forma compacta [1]. Essas representações compactas são geradas, em geral, explorando-se as redundâncias presentes na informação original.

A compressão de dados está intimamente relacionada à predição [14]. No caso ótimo determinístico, o receptor seria capaz de predizer perfeitamente a informação a ser recebida. Assim, a máxima taxa possível de compressão seria atingida, já que a transmissão da informação redundante não seria necessária [13, 15]. Na prática, se o receptor da informação tem algum conhecimento *a priori* dos dados a serem recebidos, o transmissor pode reduzir a quantidade de informações que serão transmitidas, eliminando as redundâncias do conteúdo. O receptor, então, estima as informações eliminadas através de processos de predição.

As técnicas de compressão podem ser divididas basicamente em duas classes: com perdas ou sem perdas. A compressão sem perdas (*lossless*) possibilita a recuperação integral da informação original a partir da informação comprimida. Já a compressão com perdas (*lossy*) envolve a perda de parte da informação que está sendo comprimida, impossibilitando a recuperação exata da informação original após a descompressão. Contudo, as técnicas com perdas permitem maiores taxas de compressão, ou seja, são capazes de representar a informação original usando menos *bits*. Na prática, não existe uma técnica considerada ótima que se aplique à qualquer caso. Para cada aplicação deve-se avaliar a função taxa-distorção obtida, que representa um compromisso entre a taxa de compressão e as distorções com relação à informação original, dadas as restrições de capacidade do canal [13, 15].

Atualmente, diversas aplicações demandam a transmissão de grandes quan-

tidades de informação. As técnicas de compressão são utilizadas para diminuir o volume de dados transmitidos de fato e, consequentemente, a banda de transmissão necessária. Contudo, na prática a compressão reduz as redundâncias da informação, tornando-a mais vulnerável às interferências de canal. Para minimizar esses erros são usadas basicamente duas abordagens não-excludentes: a diminuição da probabilidade de erros usando códigos de canal eficientes; e as técnicas de robustez a erros na etapa de codificação de fonte, importantes quando não se consegue evitar os erros apenas com a codificação de canal.

Dentre os diversos tipos de fonte de informação, o vídeo é provavelmente um dos que produz maior quantidade de dados [1]. Teoricamente, o vídeo pode ser considerado como uma seqüência de imagens, ou seja, um sinal de 3 componentes: duas espaciais e uma temporal. Essas componentes espaciais e temporais apresentam correlações mútuas, que são exatamente as redundâncias do sinal de vídeo exploradas pelas técnicas de compressão.

Como vídeos são geralmente reproduzidos a taxas da ordem de 30 quadros por segundo, quadros subseqüentes tendem a apresentar poucas mudanças de conteúdo, mesmo em seqüências representando cenas com bastante movimento. No caso da compressão de vídeo, a correlação espaço-temporal permite que um quadro já reconstruído seja utilizado na predição do próximo quadro, gerando uma quantidade menor de *bits* para representar o vídeo. Esta abordagem é conhecida como estimação e compensação de movimentos, e é bastante aplicada a segmentos da imagem (*block-based motion compensation*) nos padrões de compressão atuais.

Além das redundâncias inerentes ao próprio sinal de vídeo, as limitações da capacidade de percepção do sistema visual humano são bastante exploradas pelas técnicas de compressão de vídeo. Na prática, a sensação de qualidade do vídeo decodificado é bastante subjetiva, já que muitas vezes não se consegue discernir a presença de pequenos erros ou artefatos [2]. Essas características do vídeo contribuem para que a maioria absoluta das aplicações que representam o estado da arte implementem alguma técnica de compressão para a transmissão através de canais de banda limitada.

Em um sistema de comunicação [7], o vídeo é primeiramente comprimido e depois segmentado em pacotes de dados. Estes são multiplexados juntamente com

outros conteúdos, tais como áudio ou informações de controle, e preparados para transmissão através do canal. O canal introduz erros e distorções na comunicação, que podem ser tratados de diversas maneiras. As técnicas para codificação de canal introduzem informações adicionais para proteger *a priori* o conteúdo durante a transmissão, tais como informações de sincronia, códigos corretores de erros e *interleaving*. Contudo, o tratamento de erros pela decodificação de canal nem sempre é suficiente para eliminar integralmente os erros da informação passada para o decodificador de fonte. Para isso, um outro conjunto de técnicas de robustez a erros (*error resilience*) pode ser utilizado para minimizar os erros remanescentes.

A robustez a erros é um requisito essencial para aplicações que fazem uso de comunicações de vídeo, tais como radiodifusão (*broadcasting*) de TV digital, transmissão em alta definição, *streaming* por rede de pacotes ou terminais móveis, além de comunicação bidirecional em vídeo-telefonia ou vídeo-conferência. Portanto, um sistema de codificação e de codificação de vídeo, ou *codec* (*coder/decoder*), deve equacionar os critérios de fidelidade do vídeo decodificado e as limitações do canal, levando em consideração as restrições de atraso da comunicação e complexidade computacional para cada aplicação em questão [9].

2.2 H.264

O H.264 proporciona robustez e taxas de compressão significativamente maiores que os padrões anteriores. Técnicas avançadas de codificação de fonte foram introduzidas no H.264/AVC, permitindo o uso mais eficiente dos recursos de transmissão e armazenamento, porém implicando em maior complexidade computacional. Comparado ao H.263 ou MPEG-2 [3, 4, 6], o H.264 é capaz de reproduzir o vídeo com a mesma qualidade, porém com taxas de compressão muito maiores. Essa característica constitui uma enorme vantagem competitiva, tanto em termos de transmissão quanto de armazenamento. Ademais, os custos de utilização da tecnologia do H.264 são bem mais baixos, tendo em vista que seu acordo de patentes e licença de uso [16] são mais vantajosos do que de padrões anteriores.

2.2.1 Histórico

O H.264 é o padrão internacional estado da arte em compressão de vídeo. Seu *draft* final foi aprovado em outubro de 2003 [17], com seguidas revisões da norma, sendo a mais recente de janeiro de 2005.

Também conhecido como MPEG-4 *Advanced Video Coding*, H.264/AVC, ou MPEG-4 Part 10, o padrão é resultado de esforços de desenvolvimento conjunto por 2 grupos de estudos de padrões: o *Moving Picture Experts Group* (MPEG), da *International Standards Organisation/International Electrotechnical Comission* (ISO/IEC) e o *Video Coding Experts Group* (VCEG), da *International Telecommunications Union* (ITU). Os grupos MPEG e VCEG são denominados oficialmente ISO/IEC JTC1/SC20/WG11 e ITU-T SG16 Q.6, respectivamente.

O grupo MPEG foi responsável pelo desenvolvimento de padrões de compressão de vídeo e áudio MPEG-1 e MPEG-2 [3, 4, 6], ainda largamente utilizados em aplicações de comunicações e armazenamento. O grupo VCEG, por sua vez, foi pioneiro no desenvolvimento de padrões de vídeo-telefonia com o H.261, e posteriormente com os padrões H.263 e H.26L.

O MPEG-4 havia sido proposto originalmente para oferecer maior flexibilidade que os padrões anteriores. Contudo, o padrão H.263 já apresentava melhores taxas de compressão. O MPEG-4, então, direcionou seu foco para a codificação orientada a objetos (MPEG-4 *Visual* [3]). Na época da conclusão do MPEG-4 Visual, o ITU-T começou a avaliar propostas de uma nova codificação de vídeo, denominada H.26L. Os avanços na capacidade de processamento e nas pesquisas em codificação de vídeo indicavam possibilidades de se alcançar patamares de eficiência em codificação de vídeo mais elevados com relação aos padrões anteriores [18].

Então, em 2001, os grupos MPEG e VCEG decidiram unir esforços, o que levou à criação de uma equipe conjunta denominada *Joint Video Team* (JVT). O grupo JVT identificou a demanda por técnicas de codificação mais avançadas para o MPEG, e decidiu utilizar o H.26L como base para a proposta do novo padrão. O MPEG-4 AVC é atualmente um padrão internacional, publicado conjuntamente pelo ISO/IEC como MPEG-4 Part 10, e pelo ITU-T como H.264. Atualmente vem sendo adotado por diversos países e tecnologias emergentes.

2.2.2 Estrutura

O H.264 suporta a codificação de seqüências de vídeo progressivo ou entrelaçado, com espaço de cores *YCbCr* e formatos de amostragem 4:2:0.

2.2.2.1 Formato da Codificação

De forma a atender aos requisitos de flexibilidade para diferentes aplicações, o formato do *stream* resultante da codificação H.264 é representado em uma estrutura de camadas, descrita pela Figura 2.2. A camada de codificação (*video coding layer* - VCL) representa o vídeo codificado, enquanto a camada de rede (*network abstraction layer* - NAL) formata o *stream* de saída do VCL, segmentando-o e acrescentando informações de cabeçalho para seu armazenamento ou transmissão pelo protocolo de transporte [5].

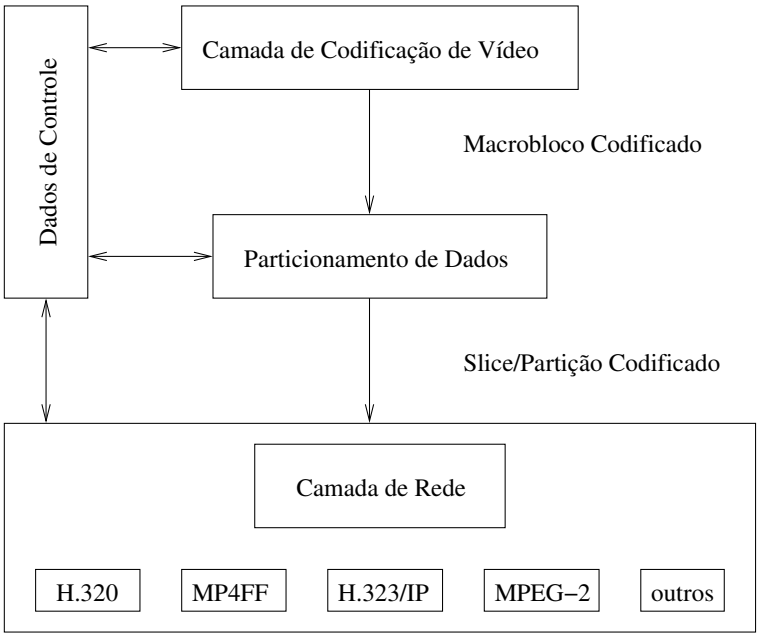


Figura 2.2: Estrutura da codificação H.264.

A camada de abstração de rede (NAL) é composta por 3 elementos principais:

- Unidades NAL (NAL *units*);
- Conjuntos de parâmetros;
- Unidades de acesso (*access units*).

O vídeo codificado é encapsulado em unidades NAL. Cada unidade contém os dados propriamente ditos, além de um cabeçalho indicando o tipo dos dados. As unidades NAL podem ser de 2 tipos: VCL, que contém dados que representam valores das amostras do vídeo; e não-VCL, que contém informações adicionais, como parâmetros de cabeçalhos aplicados a várias unidades NAL, sincronismos, dentre outras.

Os conjuntos de parâmetros abrangem cabeçalhos que se aplicam a várias unidades NAL do tipo VCL. Existem 2 tipos básicos de parâmetros: de seqüência e de imagem. Essa separação dos parâmetros permite desacoplar a transmissão de informações que mudam pouco frequentemente, de representações codificadas dos valores das amostras das imagens. Parâmetros da seqüência e da imagem podem ser transmitidos antes das unidades VCL NAL a que se referem, e podem ser repetidos a fim de prover robustez contra perdas.

As unidades de acesso, por sua vez, representam o conjunto de unidades NAL do tipo VCL e não-VCL associadas a uma mesma imagem decodificada. Uma unidade de acesso contém todos os macroblocos da imagem, informações adicionais da codificação, além de possivelmente algumas redundâncias de partes do conteúdo (*redundant slices*).

Assim como nos padrões anteriores, no H.264 o VCL segue uma abordagem de codificação híbrida baseada em blocos. As pequenas mudanças nos elementos da estrutura de codificação do H.264 são responsáveis, em conjunto, pela maior eficiência alcançada pelo H.264 com relação aos padrões anteriores.

As principais componentes da estrutura do VCL são:

- Macroblocos, *slices* e grupos de *slices*;
- Predição *intra* ou *inter*;
- Codificação por entropia.

Um sinal de vídeo codificado é composto por uma seqüência de imagens codificadas. Cada imagem é particionada em conjuntos de *pixels* de dimensão fixa, denominados macroblocos. Um macrobloco representa uma região de 16x16 pixels da componente de luminância Y, e 8x8 amostras para cada componente de cor C_b e C_r. O processo de codificação é todo orientado a macroblocos. As amostras

de um macrobloco são obtidas por predição temporal ou espacial, e o resíduo da predição é transmitido após os processos de transformada, quantização e codificação por entropia.

Os macroblocos de uma imagem são organizados em conjuntos denominados *slices*, que representam regiões da imagem que podem ser decodificadas de forma independente. A disposição espacial dos *slices* em uma imagem pode seguir diferentes tipos de organização: dispersa, *interleaved*, *raster scan*, dentre outras [18].

A robustez a erros pode fazer uso da flexibilidade proporcionada pelo particionamento da imagem em *slices*. Cada *slice* estabelece um ponto de ressincronização, permitindo que a decodificação seja reinicializada. A independência na decodificação dos *slices* permite que esses sejam transmitidos em ordem arbitrária, usando a técnica (*arbitrary slice ordering* - ASO [18]). Os macroblocos também podem ser organizados de forma flexível, segundo diferentes padrões de organização, usando (*flexible macroblock ordering* - FMO [5]).

A robustez a erros também pode ser aprimorada com a separação dos conteúdos mais importantes do processo de codificação, como vetores de movimento e tipos de macroblocos, dos menos importantes, como coeficientes da transformada dos resíduos. Os conteúdos podem ser encapsulados em unidades NAL segundo seu nível de importância, funcionalidade denominada *data partitioning* [5].

Outra abordagem de robustez a erros consiste em transmitir trechos do vídeo codificado mais de uma vez. Esta redundância é denominada *slices* redundantes (*redundant slices*).

Existem 5 tipos fundamentais de *slices*:

- *Slice* **I** (*Intra*): Todos os macroblocos em um *slice* I são codificados usando predição *intra*;
- *Slice* **P** (*Predicted*): Os macroblocos em um *slice* P podem ser codificados usando predição *intra* ou *inter*. Cada predição *inter* pode usar somente uma imagem de referência;
- *Slice* **B** (*Bi-predictive*): Os macroblocos em um *slice* B podem ser codificados usando predição *intra* ou *inter*. Cada predição *inter* pode usar até duas imagens de referência;

- *Slice* **SP** (*Switching P*): Um *slice* SP é codificado de forma a permitir a transição entre 2 *streams* de vídeo que representam a mesma seqüência com qualidades diferentes, ou também avançar e retroceder em imagens de um mesmo *stream*, transmitindo menos informação que um *slice* I;

- *Slice* **SI** (*Switching I*): Um *slice* SI representa o ponto de sincronismo para a transição entre 2 *streams*, diferindo do *slice* SP por apresentar todos os macroblocos codificados usando predição *intra*.

2.2.2.2 Tipos de Predição

A predição *intra* se baseia nas correlações espaciais da imagem. Um bloco ou macrobloco utiliza as amostras de blocos vizinhos espacialmente para sua predição. O codificador seleciona o modo de predição que minimize a diferença entre o bloco original e sua predição.

No H.264, a predição *intra* suporta um total de 9 modos de predição para cada bloco de luminância de 4×4 *pixels*, 9 modos para blocos de luminância 8×8, 4 modos para macroblocos 16×16, e 4 modos para as componentes de cor C_b e C_r [18, 19].

Os 4 modos de predição *intra* para blocos de luminância de 16×16 *pixels* são apresentados na Figura 2.3, e são equivalentes aos modos usados para as componentes de cor. A Figura 2.4 apresenta os 9 modos de predição *intra* para blocos de luminância de 4×4 *pixels*, que são equivalentes aos dos blocos 8×8. Maiores detalhes nas referências [5, 18, 19].

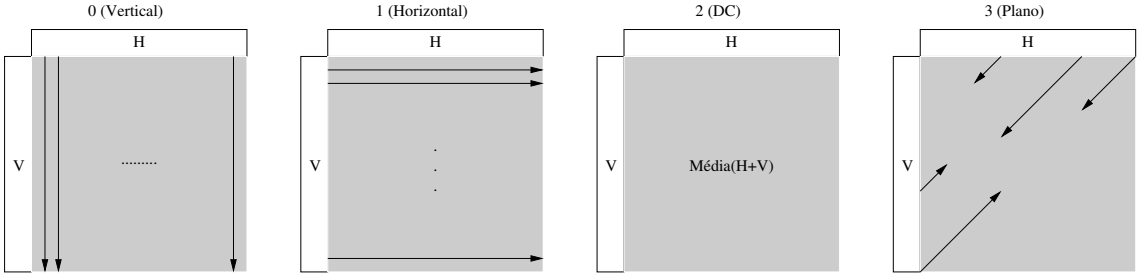


Figura 2.3: 4 modos de predição *intra* para macroblocos de luminância de 16×16 *pixels*.

A predição *inter* utiliza as correlações temporais da seqüência de vídeo, criando um modelo de predição de movimento de cada bloco. O H.264 suporta parti-

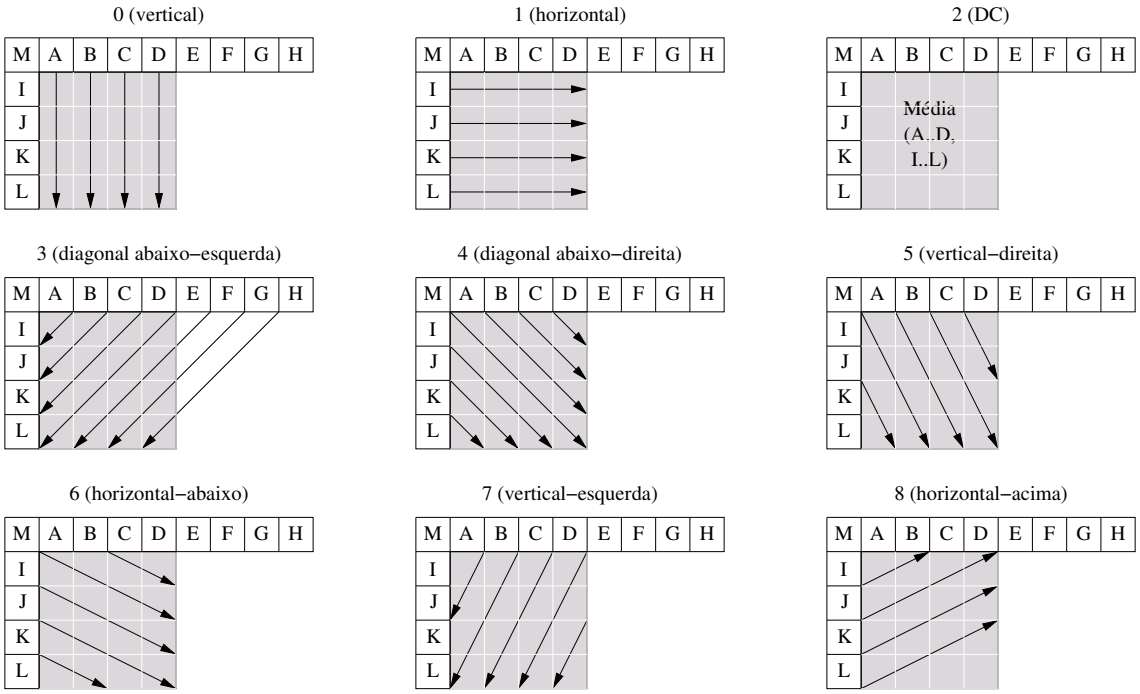
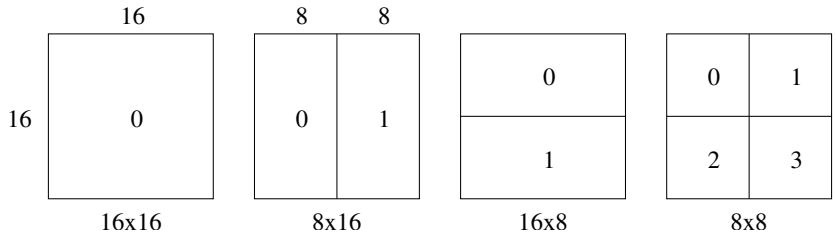


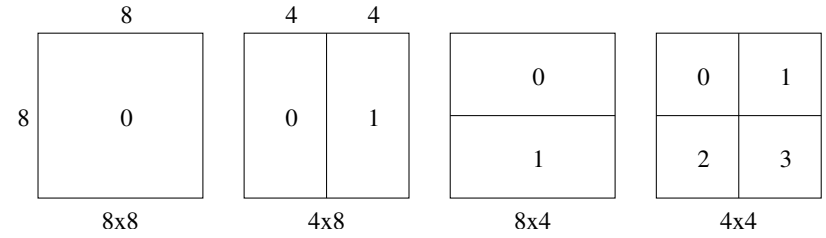
Figura 2.4: 9 modos de predição *intra* para blocos de luminância de 4×4 *pixels*.

cionamento de macroblocos em dimensões de até 4×4 *pixels*, e apresenta estimação e compensação de movimento com resolução de 1/4 de *pixel* para luminância, e 1/8 de *pixels* para crominância. Na codificação, os macroblocos são particionados em sub-blocos com as dimensões indicadas na Figura 2.5.

A predição de cada bloco de luminância de M×N *pixels* é obtida por compensação de movimento, que é especificada por um vetor de movimento, os resíduos da predição e o índice de uma imagem de referência dentre as já decodificadas [18]. O número de *bits* usado na representação do vetor de movimento e dos resíduos da predição é flexível. Portanto, a escolha do tamanho das partições de um macrobloco se dá ao minimizar o número de *bits* necessários para representar o vetor de movimento e os resíduos dos *pixels* da predição. Sua escolha depende também do nível de detalhes da região da imagem. No caso de regiões com muito movimento, partições grandes requerem menos *bits* para se representar o vetor de movimento, e mais *bits* para o resíduo; enquanto regiões pequenas necessitam menos *bits* para os resíduos, e maior quantidade para os vetores de movimento. A escolha do tamanho da partição exerce bastante influência no desempenho da compressão. Em geral, partições maiores são mais adequadas para regiões mais homogêneas da imagem, e partições menores podem ser mais eficientes em regiões com muito movimento.



(a) Macrobloco - 16×16 *pixels*



(b) Bloco - 8×8 *pixels*

Figura 2.5: Partições de um macrobloco para compensação de movimento.

Na predição *inter*, cada vetor de movimento é codificado a partir de sua predição, a qual usa vetores das regiões vizinhas previamente codificados. Isso é possível porque vetores de movimento de partições vizinhas são fortemente correlacionados. Na codificação, a predição de um vetor de movimento é formada a partir dos vetores previamente calculados. A diferença entre o vetor atual e o vetor resultante da predição é codificada e transmitida [9].

As imagens utilizadas como referência pela predição *inter* são armazenadas em uma estrutura de listas no *decoded picture buffer* (DPB), que contem imagens anteriormente decodificadas. Na predição *inter* em *slices* P, somente uma imagem de referência do DPB é utilizada na predição de um bloco de M×N *pixels*. No caso da predição *inter* em *slices* B, é usada a média ponderada da estimação e compensação de movimento obtida por duas imagens do DPB. Cabe ressaltar que na predição *inter* são usadas como referência imagens contidas no DPB, que não necessariamente representam imagens temporalmente próximas na seqüência de reprodução do vídeo. Assim, uma imagem pode ser obtida, por exemplo, através da predição com relação a outra imagem ocorrida 10 imagens antes ou depois na seqüência temporal do vídeo. Cabe ressaltar que o DPB é reinicializado ao receber um quadro IDR (*instantaneous decoder refresh*).

Um sinal de vídeo codificado usando H.264 é composto por uma seqüência periódica de imagens. O *software* de referência utiliza o termo GOP (*group of*

pictures) para se referir à menor "distância" entre quadros *intra*. Contudo o termo GOP não está definido na norma do H.264, apesar de ser um termo consagrado nos métodos de compressão.

2.2.2.3 Transformada, Quantização e Codificação por Entropia

Assim como os padrões anteriores, o H.264 usa transformadas espaciais para codificar os resíduos da predição. Além de usar uma aproximação inteira da DCT 8×8, o padrão também faz uso de uma transformada 4×4 similar à DCT. O uso de transformadas com coeficientes inteiros torna esta etapa do H.264 menos computacionalmente intensiva, e garante a precisão da transformada inversa no decodificador.

O parâmetro de quantização (*quantization parameter* - QP) controla a quantização dos coeficientes da transformada no H.264. Em conseqüência, o QP é utilizado para controlar o compromisso entre qualidade da imagem reconstruída e a taxa de bits de saída. Os coeficientes da quantização são geralmente reordenados, e passam pelo codificador por entropia.

O H.264 suporta duas classes de métodos de codificação por entropia [18], uma baseada em códigos de *Huffman*, o CAVLC (*context-adaptive variable-length coding*), e outra baseada em codificação aritmética, o CABAC (*context-adaptive binary arithmetic coding*). Ambas são adaptativas e baseadas em contexto, e freqüentemente utilizam o código *Exp-Golomb*, que possui uma estrutura simples e regular. O uso do CABAC ou do CAVLC aumenta o desempenho da codificação H.264 em relação aos padrões anteriores, sendo mais eficiente a codificação baseada no CABAC, porém com maior custo computacional.

Ainda no *loop* de decodificação, o H.264 implementa um filtro redutor de efeitos de bloco (*in-loop deblocking filter*). Esse filtro é utilizado para minimizar o efeito visual mais comum dos métodos de compressão atuais: os artefatos resultantes da descontinuidade das bordas dos blocos. Sua atuação adaptativa reduz os efeitos de bloco, mantendo as arestas reais das cenas representadas.

2.2.3 Perfis e Níveis

A flexibilidade do H.264 com respeito aos requisitos de diferentes aplicações é suportada por uma hierarquia de perfis (*profiles*) e níveis (*levels*), que definem a

sintaxe do fluxo de *bits* codificado (*bit stream*) e as restrições dos parâmetros de codificação. O padrão suporta 4 perfis: *Baseline*, *Main*, *Extended* e *High*. Maiores detalhes sobre as características dos perfis do H.264 são descritas nas referências [5, 9, 18, 19].

O perfil *Baseline* suporta *slices* I e P, código de comprimento variável baseado no contexto (CAVLC) e ordem flexível dos macroblocos (FMO). É utilizado preferencialmente em aplicações conversacionais, como vídeo-conferência e vídeo em estações móveis (telefones celulares).

O perfil *Main* suporta *slices* I, B e P, CAVLC, código aritmético binário adaptativo baseado no contexto (CABAC) e codificação de vídeo entrelaçado usando codificação quadro/campo adaptativa por imagem (PAFF) ou codificação quadro/campo adaptativa por macrobloco (MBAFF). É usado principalmente em aplicações de radiodifusão (*broadcasting*) de televisão digital.

O perfil *Extended* suporta as ferramentas do perfil *Baseline*, *slices* B, codificação de vídeo entrelaçado (PAFF e MBAFF), *slices* SI e *slices* SP. É usado principalmente em *streamings* de vídeo.

O perfil *High* suporta as ferramentas do perfil *Main*, formato YUV 4:2:0 com 8 bits por amostra, usa adaptativamente transformada 8×8 ou 4×4, matrizes de escalamento para quantização, controle separado do parâmetro de quantização (QP) de C_b (Croma Azul) e de C_r (Croma Vermelho), e formato de vídeo monocromático YUV 4:0:0. O perfil apresenta variações (High 10, High 4:2:2, High 4:4:4) [19]. No formato 4:4:4, usa espaço de cor YCgCo. É também usado em aplicações de radiodifusão (televisão digital).

O perfil *High* é uma extensão recente do padrão, denominada FREXt (*Fidelity Range Extensions*) [19], e atualmente tende a suplantar o perfil *Main*.

2.2.4 Diagrama de Blocos

O padrão H.264 não define o codificador/decodificador (*codec*), mas sim a sintaxe do fluxo de *bits* codificado (*bit stream*) e seu método de decodificação. Assim, para ser considerado compatível com o H.264, um *codec* deve implementar os elementos funcionais necessários para produzir um fluxo de *bits* corretamente decodificável usando o processo de decodificação definido pelo padrão.

Os elementos básicos que compõem um *codec* H.264 são apresentados nas Figuras 2.6 e 2.7. O funcionamento do *codec* será detalhado a seguir, apresentando a codificação passo a passo de um quadro [18]. O processo é equivalente para vídeo entrelaçado (aplicado a campos) ou progressivo (aplicado a quadros).

Na codificação do quadro $\mathbf{F}_n$, esse é segmentado em *slices*, que por sua vez são segmentados em macroblocos e blocos. Cada bloco é codificado usando os modos de predição *inter* ou *intra*. Para cada bloco é gerada uma predição $\mathbf{P}$, obtida através das amostras previamente reconstruídas. No modo *intra*, $\mathbf{P}$ é obtida por amostras do mesmo *slice* previamente reconstruídas $\mathbf{uF'}_n$. Já no modo *inter*, $\mathbf{P}$ é formada pela predição com compensação de movimento usando 1 ou 2 quadros de referência selecionados das listas 0 ou 1, dependendo se o *slice* é do tipo P ou B. Os blocos **ME** e **MC** representam, respectivamente, estimação e compensação de movimento usadas na predição *inter*.

Apesar de identificado como $\mathbf{F'}_{n-1}$, o quadro de referência não representa o quadro anteriormente codificado. Na prática, o quadro de referência pode ser selecionado entre os vários quadros já codificados, decodificados e reconstruídos, podendo representar quadros futuros ou passados na ordem temporal de visualização do vídeo.

A predição $\mathbf{P}$ é subtraída do bloco corrente, produzindo o resíduo $\mathbf{D}_n$. A $\mathbf{D}_n$ é aplicada uma transformada de bloco, seguida da quantização. Os coeficientes resultantes da quantização ($\mathbf{X}$) são reordenados e passam por um codificador por entropia. Ao fluxo de bits resultante são adicionadas informações adicionais necessárias ao processo de decodificação. Tudo é depois encapsulado pela camada de rede (NAL) para transmissão.

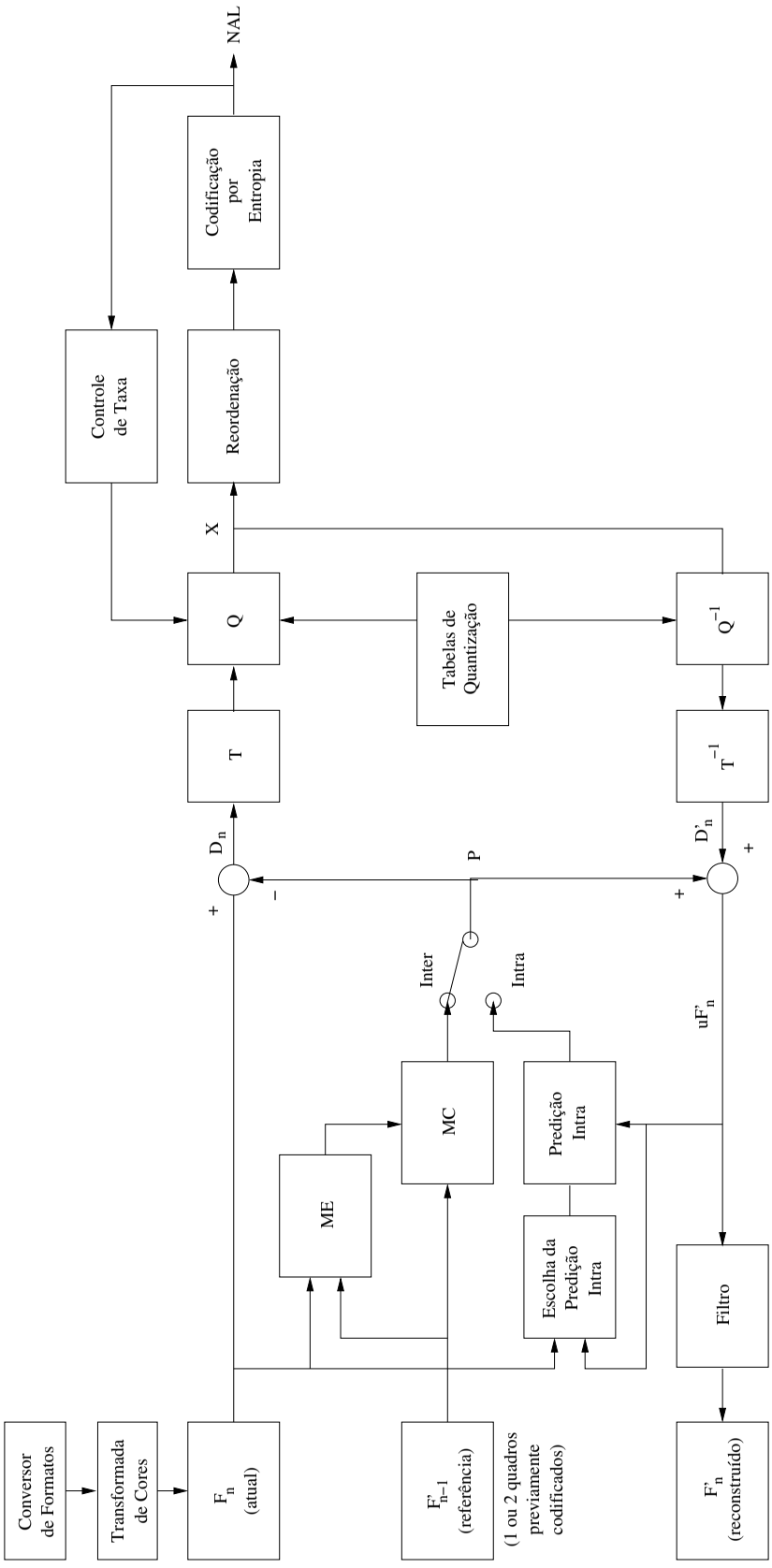


Figura 2.6: Codificador H.264.

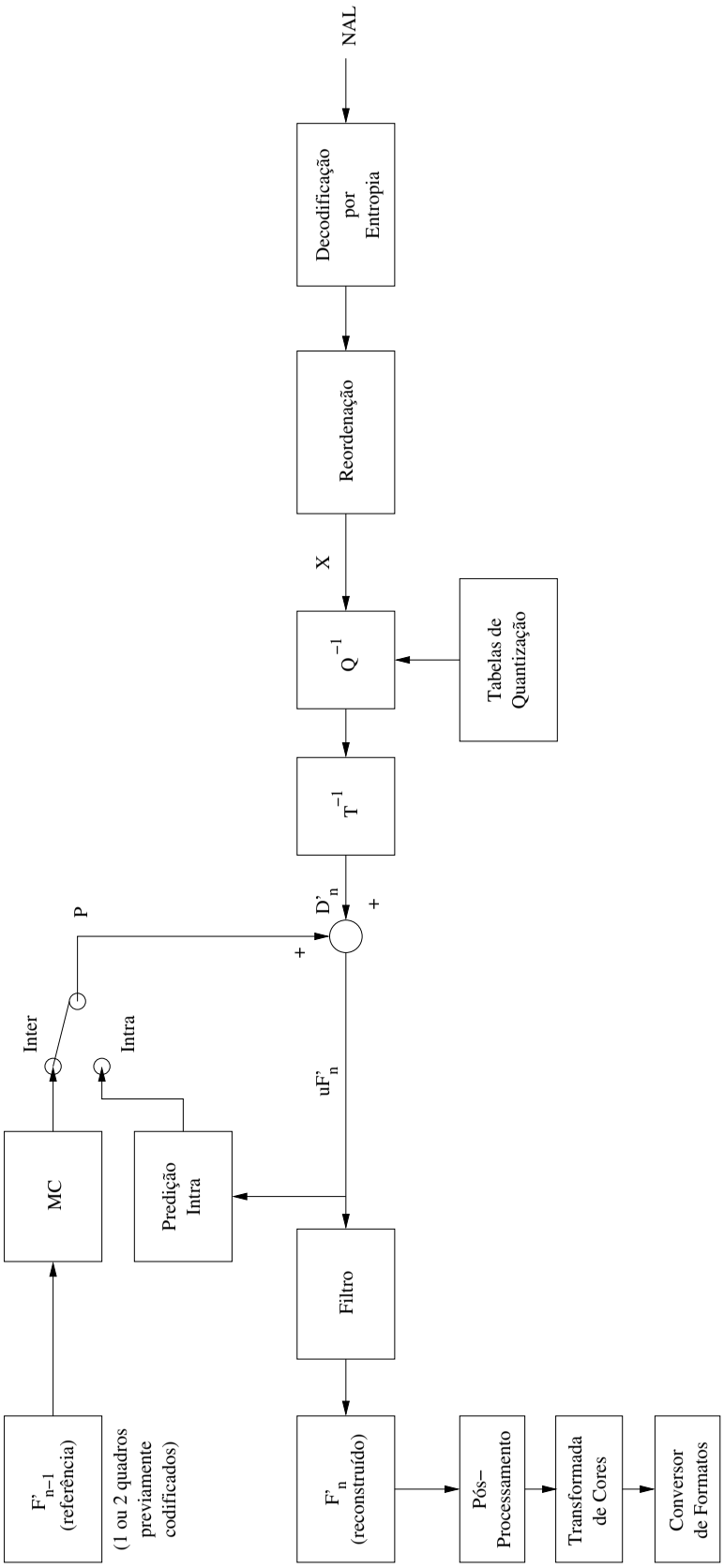


Figura 2.7: Decodificador H.264.

Na Figura 2.6, o conversor de formatos e a transformada de cores são etapas de pré-processamento do sinal de vídeo. O passo de quantização **Q** controla a taxa de saída (*bit rate*).

Parte do codificador da Figura 2.6 implementa as mesmas etapas do decodificador da Figura 2.7. Isso ocorre pois o codificador necessita reconstruir cada quadro codificado, a fim de utilizá-lo como referência para a codificação dos quadros seguintes.

O processo de decodificação consiste em re-escalar os coeficientes **X** com a função $\mathbf{Q}^{-1}$, e aplicar a transformada inversa $\mathbf{T}^{-1}$ para reproduzir os resíduos do bloco $\mathbf{D'}_n$. A predição **P** é então adicionada a $\mathbf{D'}_n$, gerando $\mathbf{uF'}_n$, uma versão reconstruída do bloco codificado originalmente. Ao final, um filtro de redução de efeitos de blocos é aplicado, resultando no quadro reconstruído.

2.3 Técnicas de Ocultamento de Erros

O controle de erros em comunicações de vídeo é um problema que apresenta diversos desafios. Seqüências de vídeo comprimido são bastante sensíveis a erros de transmissão, principalmente devido ao uso de codificação preditiva e de códigos de comprimento variável. A predição espaço-temporal torna o vídeo codificado extremamente vulnerável a erros, já que a alta correlação entre amostras gera a propagação do erro de uma única amostra para várias amostras seguintes. Os erros em códigos de comprimento variável, por sua vez, podem levar à perda de sincronismo do decodificador, e à conseqüente inutilização de trechos consideráveis do sinal codificado [7, 8].

A demanda pela transmissão de vídeo através de canais não confiáveis, como a internet e redes *wireless*, tem levado ao desenvolvimento de diferentes abordagens para minimizar os impactos de canais propensos a erros na qualidade de seqüências de vídeo codificado. Esses erros são tratados por diferentes técnicas de robustez a erros, incluindo os métodos de "ocultamento de erros" (*error concealment*).

As técnicas usadas para combater erros na transmissão de vídeo têm sido desenvolvidas segundo duas vertentes [8]:

- Controle de erros e esquemas de recuperação usados em comunicação de dados

em geral;

- Reconstruções e ocultamento de erros específicos para vídeo codificado.

Primeiramente, controle de erros e esquemas de recuperação em comunicações de dados foram aplicados à transmissão de vídeo. Essas técnicas incluem detecção e correção de erro *a priori* (*forward error correction*), códigos detectores de erros (CRCs - *cyclic redundancy check*), e retransmissões (*automatic retransmission requests*). Especificamente para vídeo, no caso em que um erro não é corrigido, mas apenas detectado, são usadas técnicas de reconstrução e ocultamento de erros buscando obter uma melhor aproximação possível do sinal original. Ademais, diferentemente da transmissão de outros tipos de dados, a codificação de vídeo não requer um esquema de codificação sem perdas, já que o olho humano tolera um certo nível de distorção em imagens e vídeos.

A priori, todas as técnicas de codificação robusta a erros (*error resilient*) se baseiam na premissa de tornar o processo de codificação menos eficiente, ao manter algum nível de redundância no *stream* codificado. Maior eficácia na detecção e ocultamento de erros no decodificador é alcançada através de redundâncias mantidas tanto na codificação de fonte, quanto na codificação de canal. As redundâncias são utilizadas para recuperar eventuais erros de canal, tal que *bits* com erros não representem perdas consideráveis de qualidade no vídeo reconstruído, e principamente, se propaguem o menos possível ao longo da seqüência.

O problema de robustez a erros em vídeo pode ser formulado, em linhas gerais, como o projeto de um par de codificador/decodificador de fonte e codificador/decodificador de canal que minimize a distorção do sinal no decodificador, dadas as características do vídeo e do modelo do canal. Este problema é muito difícil de ser resolvido, haja visto as inúmeras variáveis e suas características não-estacionárias.

Na literatura foram propostas diversas técnicas para atacar especificamente problemas de erros na transmissão de vídeo codificado, segundo diferentes abordagens. A seguir são apresentadas as técnicas em 3 grupos, categorizadas segundo o papel do codificador e do decodificador no processo de ocultamento de erros [7, 8]. As três categorias são:

- Codificação de fonte robusta a erros (*error resilient*), também denominada

forward error correction;

- Ocultamento interativo de erros;
- Ocultamento de erros no decodificador.

Dentre esses métodos, a codificação de fonte robusta a erros adiciona redundâncias à seqüência durante a etapa de codificação, de forma a tornar o fluxo de *bits* codificado menos suscetível a potenciais erros de canal. Já o ocultamento interativo de erros requer a troca de informações entre codificador e decodificador, para que o codificador se adapte às condições de perda detectadas no decodificador. O ocultamento de erros na fase de decodificação, por sua vez, busca ocultar o efeito de erros previamente detectados na transmissão, e que não tenham sido corrigidos nas outras etapas da decodificação.

Com respeito ao papel do codificador e do decodificador no ocultamento de erros, a codificação de fonte robusta a erros é implementada no codificador, o ocultamento de erros no decodificador, e a interativa distribui o processamento entre ambos.

A codificação de fonte robusta a erros busca aperfeiçoar a capacidade de recuperar erros, aumentando a redundância da informação codificada. Apesar de parecer contraditória com relação a premissa inicial da codificação de fonte em otimizar a taxa de compressão, a perda de eficiência na compressão é compensada pela robustez da codificação. Na prática, a codificação de fonte robusta a erros pode ser implementada tanto no codificador de fonte quanto no de canal. Assim, a capacidade do canal é utilizada de forma menos eficiente, em função do *overhead* introduzido na seqüência codificada. Contudo, ao agregar mais informação ao sinal codificado, os erros de transmissão afetam menos o sinal codificado. Além disso, sinais com maior redundância passam a estar disponíveis no decodificador, melhorando seu desempenho em termos de ocultamento de erros.

As técnicas de ocultamento interativas requerem a troca de informações adicionais entre codificador e decodificador, aumentando a complexidade da comunicação. Para muitas aplicações, a implementação de algoritmos de ocultamento interativo de erros não são apropriadas, devido a limitações práticas. Por exemplo, para aplicações em tempo real e *streamings* de vídeo, o atraso introduzido

pelas comunicações e eventuais retransmissões afetaria seriamente seu desempenho. Ademais, diversas aplicações podem não dispor de comunicação bidirecional (*full-duplex*). Exemplos de técnicas interativas são *automatic retransmission request* e predição seletiva baseada na realimentação do decodificador.

O ocultamento de erros no decodificador, por sua vez, inclui técnicas nas quais apenas o decodificador é responsável por implementar o ocultamento de erros. Essas técnicas são aplicadas ao final do processo de decodificação, e utilizam o conteúdo do vídeo recebido corretamente para estimar os elementos perdidos. A informação perdida é recuperada através de predições e interpolações, sem requerer mudanças no processo de codificação. Essas técnicas serão detalhadas mais adiante.

Os requisitos principais a serem considerados na escolha de uma dessas técnicas de ocultamento de erros para uma determinada aplicação são:

- Qualidade da imagem decodificada;
- Atrasos;
- Possibilidade do uso de retransmissões;
- *Overhead* em taxa de *bits*;
- Complexidade computacional.

Em geral, o ocultamento de erros é adequado à maior parte das aplicações. Seu fator crítico se concentra principalmente na complexidade do processamento no decodificador. Porém, apresenta vantagens ao não requerer retransmissões ou alterações no fluxo de bits codificado.

2.3.1 Ocultamento de Erros no Decodificador

O escopo desta dissertação engloba técnicas de ocultamento de erros no decodificador de vídeo. Erros causados pelo canal, que eventualmente incidam no fluxo de bits do vídeo codificado, são tratados inicialmente por técnicas de correção de erros na decodificação de canal. Quaisquer erros remanescentes, quando detectados, resultam em perdas de macroblocos, *slices* ou de parâmetros de codificação da seqüência ou da imagem [7, 8, 20].

O ocultamento de erros no decodificador não requer modificações do processo de codificação, nem o aumento do *overhead* da seqüência codificada. Ao atuarem após a decodificação, essas técnicas fazem uso somente das redundâncias já presentes no vídeo decodificado. Contudo, seu uso aumenta a complexidade computacional do decodificador de vídeo, o que restringe o escopo de aplicações em que pode ser utilizado.

Esses procedimentos de ocultamento assumem que os erros tenham sido detectados durante a decodificação, mas não corrigidos. A detecção de erros pode ser realizada no decodificador de fonte ou no de canal. Na decodificação de canal, a detecção se baseia em informações presentes no fluxo de bits, em códigos detetores de erros (CRCs), ou, no caso de redes de pacotes, nas informações de cabeçalho. Já a detecção de erros na decodificação de fonte explora as correlações características do vídeo natural.

As técnicas de ocultamento de erros no decodificador descritas neste texto atuam em erros que resultem na presença de artefatos na seqüência de vídeo resultante da decodificação. É válido assumir que erros na transmissão resultem em artefatos, mas que não afetem significativamente a integridade do fluxo de *bits* codificado, pois juntamente com as técnicas de ocultamento de erros também são implementadas técnicas de correção e outras funcionalidades que agregam maior robustez à transmissão do sinal de vídeo codificado. Na prática, a regeneração de artefatos se baseia nas correlações do sinal de vídeo já decodificado. Para tal, o decodificador de fonte busca estimar as amostras corrompidas usando suas correlações com as amostras adjacentes recebidas corretamente.

De fato, as técnicas usuais de ocultamento de erros no decodificador regeneram artefatos com base nas correlações do sinal de vídeo decodificado, sejam temporais, espaciais ou no domínio da freqüência. Em geral, implementam algum tipo de interpolação espaço-temporal, levando em consideração requisitos espaciais e temporais de suavidade.

O ocultamento de erros no decodificador busca a recuperação ou estimação das informações perdidas. Para codificações baseadas em blocos e macroblocos, as informações que precisam ser estimadas para se recuperar um macrobloco perdido são [7, 8] de textura ou de movimento.

As informações de textura abrangem os valores dos *pixels* ou dos coeficientes da transformada (DCT, em geral) do bloco da imagem original, ou do erro de predição. As informações de movimento, por sua vez, consistem nos vetores de movimento e também no modo de codificação, para macroblocos ou blocos codificados por predição *inter*.

A recuperação das informações de textura e de movimento são realizadas segundo abordagens temporais, espaciais, ou híbridas. A seguir são apresentadas algumas dessas técnicas, aplicadas à codificação baseada em blocos.

Cabe ressaltar que, neste texto, o termo "aresta" é aplicado a contornos de objetos da imagem, e o termo "borda", aos *pixels* que compõem a região externa de um bloco ou macrobloco.

2.3.1.1 Ocultamento Espacial

Os procedimentos espaciais de ocultamento de erros buscam recuperar as informações de textura do vídeo, usando somente as informações da imagem afetada pelos erros. As técnicas espaciais levam em conta que a imagem a ser tratada apresenta predominância de baixas freqüências e, conseqüentemente, os macroblocos corrompidos têm alta correlação com os seus vizinhos. Esse pressuposto se aplica a imagens naturais, que apresentam considerável predominância de baixas freqüências, com exceção de regiões com arestas. Dessa forma, é possível utilizar-se de algum algoritmo de interpolação espacial para se estimar os macroblocos perdidos através dos seus vizinhos.

Na literatura são encontradas diversas referências a técnicas de ocultamento espacial por interpolação de *pixels* dos macroblocos vizinhos. Muitos métodos se baseiam na regeneração de um bloco (ou macrobloco) perdido a partir de interpolações apenas dos *pixels* localizados nas suas bordas que pertençam a macroblocos vizinhos válidos (sem erros). Diversas variações dessa idéia podem ser observadas nas referências [11, 21, 22], usando filtros de interpolação ponderada ou transformadas. Dentre outras abordagens, temos a proposta de funções de suavidade [23], que modelam uma função de restrição para descrever os requisitos de suavidade entre o macrobloco perdido e seu vizinho.

As interpolações espaciais que consideram somente os *pixels* ou coeficientes da

transformada dos macroblocos vizinhos podem causar efeitos indesejáveis na imagem recuperada. Efeitos como uma imagem borrada ou embaçada são comuns. Esse problema é mais evidente no caso de imagens com a presença de muitas arestas. Para minimizar esses efeitos, são propostas técnicas direcionais de interpolação espacial, levando em consideração as direções das arestas [24–26]. Uma abordagem que utiliza somente as informações dos vizinhos acima e abaixo para determinar a presença de arestas e regenerar a região danificada [25], produz resultados muito bons para casos de perdas de macroblocos adjacentes na horizontal. Também ocorrem referências na literatura sobre algoritmos de ocultamento independentes da codificação, que usam propriedades de regularidade das *wavelets* [27].

2.3.1.2 Ocultamento Temporal

Os procedimentos temporais de ocultamento de erros buscam recuperar perdas de informação de movimento. Na prática, os métodos se baseiam na estimação dos vetores de movimento dos macroblocos perdidos, a partir dos vetores de movimento dos macroblocos vizinhos.

Na literatura, temos o uso de ocultamento temporal de erros em diversas técnicas de compressão que se utilizam de predição temporal baseada em blocos, como JPEG2000, MPEG-2, H.264, dentre outras.

As abordagens de ocultamento temporal costumam se basear no princípio comum de estimar o vetor de movimento do macrobloco perdido, usando os vetores dos macroblocos vizinhos. Cada um desses vetores é usado para obter uma estimativa do movimento do macrobloco perdido com respeito à imagem de referência. A escolha do vetor de movimento a ser usado para recuperar o macrobloco corrompido segue diferentes abordagens, que costumam levar em consideração o critério de suavidade espacial entre o macrobloco que irá substituir o macrobloco perdido, e seus vizinhos. BMAs (*boundary matching algorithms*) [11, 26, 28] são exemplos de métodos utilizados para escolher o vetor de movimento que resulte no melhor casamento entre as bordas do macrobloco que substituirá o macrobloco corrompido, com relação à sua vizinhança na imagem regenerada.

Na maioria dos métodos encontrados na literatura, os vetores de movimento das regiões vizinhas ao macrobloco perdido geram um conjunto de vetores de mo-

vimento candidatos. Nos métodos de casamento da região vizinha (BNM - *best neighborhood matching*) [29–31], todos esses vetores são verificados, testando-se o casamento de uma região vizinha ao macrobloco perdido, com respeito à imagem de referência.

Referências a outros métodos de ocultamento de erros são apresentadas no próximo capítulo.

Capítulo 3

Métodos Práticos de Ocultamento de Erros em Codificação de Video

Este capítulo apresenta diversas técnicas utilizadas atualmente no ocultamento de erros em codificação de video.

Em seguida, são descritos os métodos implementados no modelo de referência do decodificador H.264, introduzindo tanto os princípios de detecção de erros, quanto as técnicas de ocultamento de erros (*error concealment*).

3.1 Técnicas Modernas de Ocultamento de Erros

Todos os métodos de ocultamento de erros visam minimizar efeitos visuais causados por erros na decodificação do vídeo. As técnicas de ocultamento de erros apresentadas a seguir se aplicam a métodos atuais de codificação de fonte, baseados na segmentação da imagem em blocos ou macroblocos (*block-based*). Apesar das inúmeras variações, os métodos costumam usar informações dos macroblocos vizinhos ao macrobloco corrompido, sejam informações relativas à *pixels* ou ao movimento. Além disso, os macroblocos regenerados devem respeitar restrições de suavidade com respeito a macroblocos adjacentes. Assim, os métodos de ocultamento de erros apresentam, em geral, características semelhantes a de filtros passa-baixas.

A seguir são apresentados os principais métodos de ocultamento de erros encontrados na literatura. Primeiramente são apresentados alguns métodos baseados em critérios espaciais, seguidos de métodos baseados em critérios temporais.

No ocultamento espacial de erros, cada *pixel* do macrobloco corrompido pode ser obtido a partir da soma ponderada dos valores dos *pixels* mais próximos da borda dos 4 macroblocos espacialmente adjacentes, que também não estejam corrompidos [11]. Esse método foi proposto para o modelo de referência do decodificador H.264, e será descrito em detalhes mais adiante.

Na heurística de ocultamento de erros usada no *software* de referência, o ocultamento temporal é usado para regenerar macroblocos de *slices* codificados com predição *inter*, e o ocultamento espacial para *slices intra*. Esses critérios de decisão funcionam razoavelmente bem para seqüências de video com movimentos moderados. Contudo, a presença de erros em mudanças de cena ou em regiões de movimentos irregulares pode tornar bastante insatisfatório o resultado desses esquemas de ocultamento de erros.

Para minimizar esses problemas, alguns métodos propõem mudanças no critério de escolha de funções de ocultamento temporal ou espacial. A decisão pelo uso de ocultamento temporal ou espacial pode levar em consideração a detecção de mudanças de cena e nível de movimento da região afetada pelo erro [32]. Dessa forma, ambas abordagens espaciais e temporais de ocultamento são possíveis para macroblocos de *slices inter* ou *intra*. Uma heurística alternativa para a decisão de ocultamento temporal ou espacial é baseada no modo de codificação dos 8 macroblocos vizinhos ao macrobloco corrompido [26]. Caso haja predominância de codificação *intra*, são usados métodos espaciais de ocultamento. Caso contrário, são usados métodos temporais.

Em outros métodos espaciais, busca-se o melhor casamento de vizinhanças do macrobloco corrompido (*best neighborhood matching* - BNM). A regeneração de um macrobloco corrompido pode ser realizada através da busca de regiões semelhantes na mesma imagem [30]. O procedimento consiste em buscar o casamento (*matching*) de uma região, denominada vizinhança, composta por um conjunto de *pixels* pertencentes a partes de macroblocos localizados no entorno do macrobloco corrompido. A busca do melhor casamento da vizinhança é limitada por uma região de busca centralizada no próprio macrobloco corrompido. O critério de casamento utilizado é o erro médio quadrático (*mean square error* - MSE), calculado entre os *pixels* da vizinhança e os *pixels* da área testada na região de busca. Na região de

busca, a posição que resultar no menor MSE tem seu interior utilizado para substituir o macrobloco corrompido. O funcionamento do método BNM é representado pelas Figuras 3.1 e 3.2. A Figura 3.1 apresenta um macrobloco corrompido, sua vizinhança e região de busca a ser utilizada pelo método BNM. Já a figura 3.2 exemplifica alguns passos usados pelo método BNM para a escolha do macrobloco substituto.

O tamanho da região de busca no método BNM pode onerar computacionalmente a busca do melhor casamento da vizinhança do macrobloco corrompido. Para minimizar esse problema, uma proposta de modificação ao BNM utiliza uma região de busca em formato hexagonal ao redor do macrobloco corrompido [29]. Esse método propõe a busca centralizada em pontos específicos localizados dentro do hexágono, e o critério de melhor casamento é tomado pela média dos valores absolutos dos *pixels* (*mean absolute difference* - MAD).

Método espaciais de ocultamento de erros baseados em algoritmos de suavidade conhecidos como difusão anisotrópica espacial (*spatial anisotropic diffusion*) [31, 33] são propostos para suavizar gradientes entre *pixels* de macroblocos regenerados e suas vizinhanças. Visando diminuir o efeito de embaçamento (*blurring*) no casamento de bordas, a função iterativa de difusão é aplicada *pixel* a *pixel* na regeneração do macrobloco corrompido.

Equações lineares podem ser utilizadas para modelar a suavidade na conexão entre blocos corrompidos de 4×4 *pixels* e suas vizinhanças [23]. A equação de restrição de suavidade utiliza os *pixels* localizados nas bordas dos blocos vizinhos para recuperar as bordas do bloco corrompido, assumindo que *pixels* adjacentes devem apresentar pequenas diferenças de luminância. A região interna é então recuperada a partir de interpolações dos *pixels* de borda já regenerados. Sua implementação apresenta baixo custo computacional.

Alguns algoritmos de ocultamento espacial buscam a preservação de eventuais arestas da imagem, dado que a integridade das arestas é bastante importante na percepção visual de vídeos e imagens. O método *split match* [33], por exemplo, avalia as similaridades das regiões vizinhas ao macrobloco corrompido. O algoritmo busca identificar texturas, arestas ou detalhes espaciais em diversas direções dos macroblocos vizinhos, verificando a presença de padrões de objetos em regiões cada

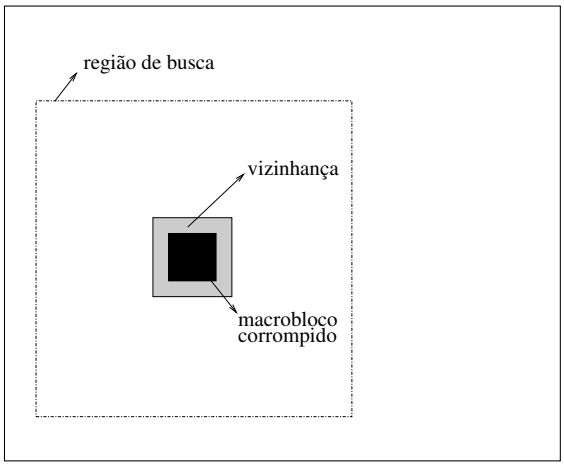
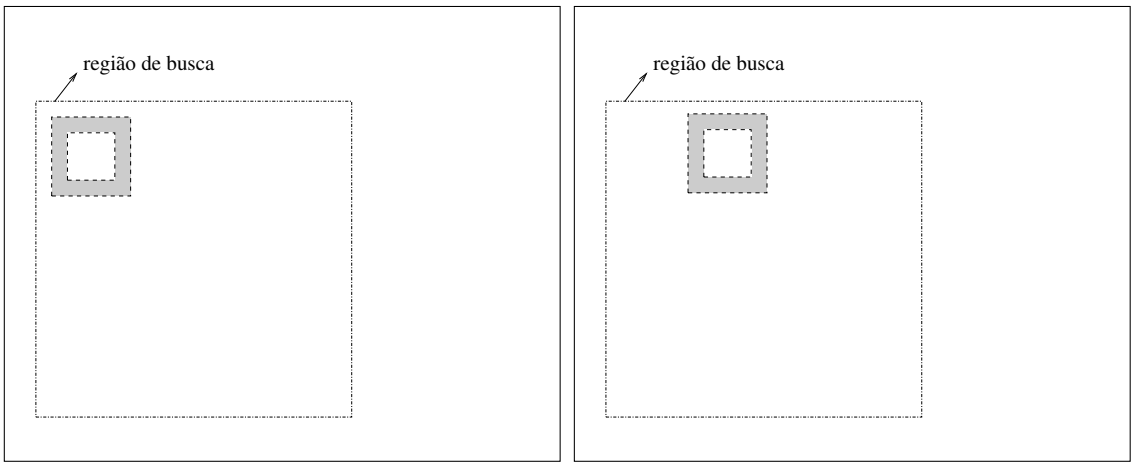


Figura 3.1: Representação de um macrobloco corrompido, sua vizinhança e região de busca usados no método BNM.



(a) Passo 1 (b) Passo 2

Figura 3.2: Procura do melhor casamento da vizinhança do macrobloco corrompido na região de busca, para o método BNM.

vez menores, até blocos de dimensão 4×4 pixels. A recuperação do macrobloco corrompido é feita através de funções de interpolação ou de difusão, aplicadas na direção das arestas, podendo também ser feita através da cópia de regiões vizinhas.

Outros algoritmos com foco na preservação de arestas são aplicados a imagens com perdas de macroblocos vizinhos na horizontal. Operadores matemáticos são utilizados para detectar a direção de eventuais arestas que atravessem as bordas superior e inferior do macrobloco corrompido [25]. Esses operadores são aplicados a *pixels* localizados nas bordas superior e inferior do macrobloco corrompido. Ao final, os *pixels* usados para substituir o macrobloco corrompido são obtidos por interpolação direcional, ponderada em função das arestas detectadas.

A detecção de arestas também pode ser realizada hierarquicamente, nos níveis de macroblocos, blocos e sub-blocos, a partir de transformadas no domínio *wavelet* [24]. Com a identificação das arestas, interpolações espaciais direcionais são então aplicadas na regeneração dos *pixels* dos macroblocos corrompidos.

Outro método de ocultamento espacial com foco na preservação de arestas visa aumentar a acurácia da estimação ao se buscar *pixels* espacialmente correlacionados em uma extensa área ao redor do macrobloco corrompido [26]. Nesse método, filtros de gradiente são usados para detectar a presença de arestas em 16 direções possíveis. Ao final, interpolações ponderadas levam em consideração a direção predominante das arestas.

Diferentemente das técnicas espaciais de ocultamento de erros, os procedimentos temporais buscam recuperar macroblocos corrompidos através da estimação de sua tendência de movimento. Conforme já mencionado, a maior parte das técnicas temporais utilizam os vetores de movimento dos macroblocos vizinhos para estimar a tendência de movimento do macrobloco corrompido com respeito à imagem de referência. A Figura 3.3 apresenta um exemplo do processo de compensação de movimento de um macrobloco, usando o vetor de movimento *VM*.

Através de técnicas de compensação de movimentos, os vetores de movimento dos macroblocos vizinhos ao macrobloco corrompido podem ser usados para se obter *pixels* candidatos a substituir o macrobloco corrompido [11, 28]. Com cada vetor de teste é obtida uma estimativa de *pixels*, a partir da imagem de referência. O critério utilizado para a escolha da melhor solução se baseia na máxima suavidade do casa-

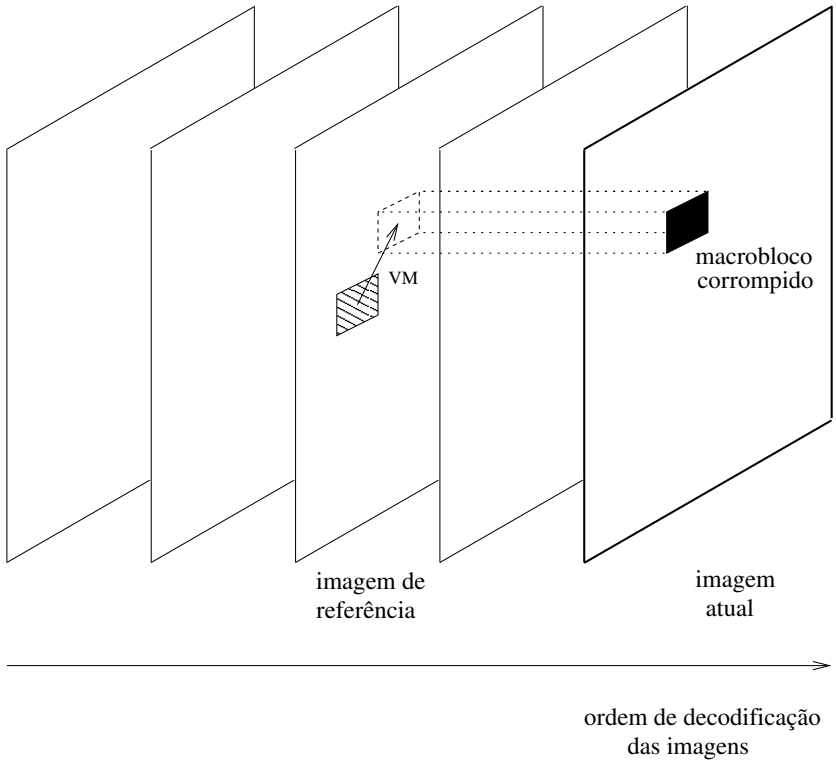


Figura 3.3: Processo de compensação de movimento de um macrobloco com respeito à imagem de referência.

mento desses *pixels* na imagem reconstruída, sendo então denominados algoritmos de casamento de bordas (*boundary matching algorithms* - BMA). Maiores detalhes são apresentados mais adiante, no detalhamento do ocultamento temporal de erros utilizado no *software* de referência do padrão H.264.

Os algoritmos BMA exploram o fato de que *pixels* adjacentes em uma imagem natural apresentam alta correlação espacial. A função custo do BMA é definida como a diferença absoluta entre os *pixels* na fronteira externa do macrobloco corrompido, e os *pixels* na fronteira interna do macrobloco candidato a substituí-lo. Conforme já mencionado, os macroblocos candidatos são obtidos pela compensação de movimento usando um conjunto de vetores candidatos. Dentre os macroblocos candidatos, é escolhido o que minimiza a função custo do algoritmo BMA.

Uma variação dos algoritmos BMA, denominada *overlapping* BMA, utiliza como critério de casamento do macrobloco substituto a diferença absoluta entre *pixels* na fronteira externa da posição do macrobloco corrompido na imagem de referência, e a posição equivalente na imagem afetada pelo erro [34]. Outras características desse método incluem a utilização dos modos de predição dos macroblocos

vizinhos, e a regeneração de macroblocos corrompidos de forma segmentada (bloco a bloco).

Outra possível abordagem temporal consiste em uma variação do método de casamento de vizinhanças (BNM) aplicado no domínio temporal, ou BNM com compensação de movimento (*motion-compensated* BNM) [29]. Esse método busca o melhor casamento da vizinhança do macrobloco corrompido na imagem de referência, usando como conjunto de vetores de teste os vetores de movimento válidos dos macroblocos vizinhos ao macrobloco corrompido. A Figura 3.4 ilustra o princípio de funcionamento do casamento da vizinhança do macrobloco corrompido, na imagem de referência.

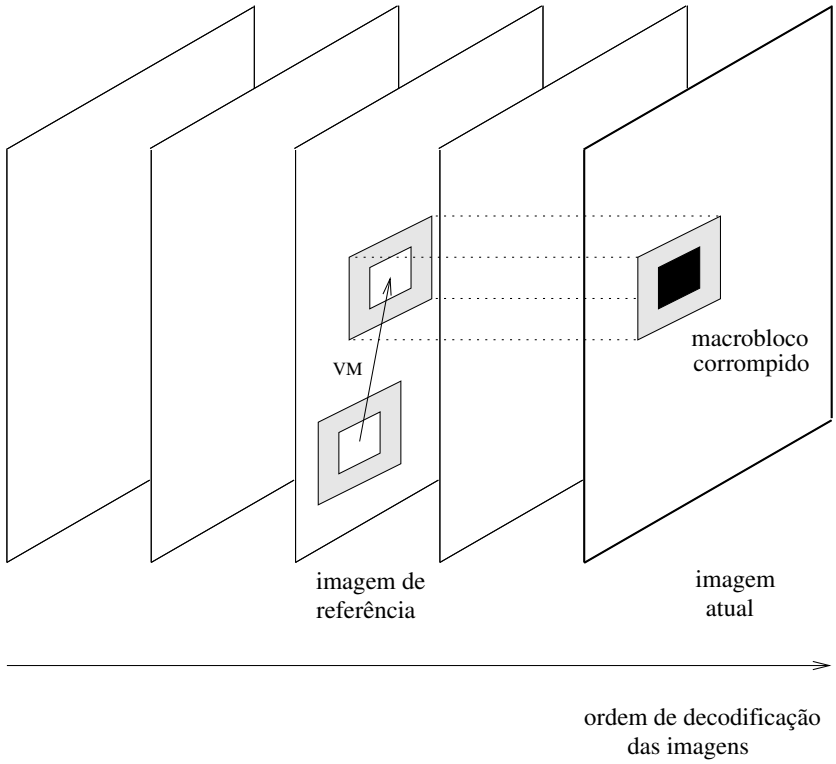


Figura 3.4: Método *best neighborhood matching* aplicado no domínio temporal.

Uma variação do método BNM com compensação de movimento é obtida de forma equivalente ao método espacial, mas considerando uma região de busca retangular pré-definida na imagem de referência [31]. Os vetores de movimento usados para testar o melhor casamento são obtidos a partir das vizinhanças do macrobloco corrompido na imagem atual.

Outras soluções temporais implementam algoritmos com foco em uma maior eficiência computacional. Um desses métodos permite regenerar blocos de 8×8 *pixels*

usando no máximo 72 operações de multiplicação [21]. Para a regeneração de um bloco de M×N *pixels* são considerados apenas os (2(M+N+2)) *pixels* localizados na fronteira com seus blocos adjacentes. As interpolações implementadas se baseiam na transformada discreta cosseno (DCT), e em produtos de *kronecker* entre matrizes.

Outro algoritmo temporal de menor custo computacional pressupõe que os coeficientes de alta freqüência da transformada DCT aplicada aos *pixels* que compõem o macrobloco tendem a ser nulos [22]. Nesse método, ao se anular tantos coeficientes da DCT quanto for o número de *pixels* do macrobloco corrompido, é obtido um sistema de equações lineares cuja solução equivale a uma operação de interpolação para estimar os valores dos *pixels* do macrobloco corrompido. Nesse esquema de interpolação, somente 8 *pixels* de bordas são utilizados para se estimar um *pixel* do macrobloco corrompido.

A tendência dos vetores de movimento dos macroblocos vizinhos ao macrobloco corrompido também pode ser estimada através de polinômios obtidos por fórmulas de interpolação de Lagrange [35]. Esse algoritmo busca estimar iterativamente o movimento de cada sub-bloco de dimensão 4×4 *pixels* do macrobloco corrompido, a partir de sub-blocos de mesma dimensão pertencentes aos macroblocos vizinhos.

Outro métodos de ocultamento temporal são baseados em estimações de planos. Utilizando os vetores de movimento próximos a macrobloco corrompido, pode ser definido um plano de primeira ordem indicando a tendência de movimento dos vértices do macrobloco corrompido [36]. A partir da interpolação desses vetores são obtidas estimativas individuais de movimento para cada *pixel* do macrobloco corrompido.

Alguns métodos implementam técnicas de ocultamento de erros distintas em função da identificação do macrobloco corrompido como sendo parte do plano de fundo (*background*) ou frontal (*foreground*) da imagem [37]. Inicialmente, o algoritmo decide se o macrobloco corrompido pertence ao plano de fundo ou frontal, através da avaliação das diferenças entre os *pixels* de borda. Caso o macrobloco seja de fundo, uma função de substituição temporal é utilizada. Caso o macrobloco corrompido pertença ao plano frontal, o macrobloco será regenerado ao se ponderar estimativas obtidas de múltiplas imagens de referência.

Métodos mais sofisticados sugerem a recuperação de cada bloco de 8×8 *pixels* do macrobloco corrompido a partir de diversas informações de movimento obtidas dos macroblocos vizinhos [26]. A estimação de cada bloco é obtida por compensação de movimento. O conjunto de vetores de testes inclui vetores de movimento dos blocos vizinhos: na horizontal, vertical, diagonal, a mediana e a média de todos os vetores vizinhos, além do vetor nulo e do vetor de movimento do macrobloco na mesma posição na imagem decodificada anteriormente. O vetor que proporciona o melhor casamento é obtido através da minimização de uma função ponderada de distorção de bordas, aplicada entre o macrobloco candidato e a imagem regenerada.

Os algoritmos de ocultamento de erros já apresentados se aplicam às seqüências de video codificadas em blocos. Outras técnicas são propostas na literatura para codificação de video baseada em sub-bandas da transformada *wavelet*. Diferentemente da codificação baseada em blocos, regiões afetadas por erros não estão completamente perdidas, em se tratando de técnicas baseadas em sub-bandas. Para uma dada região afetada por erros, as sub-bandas recebidas corretamente são usadas de forma bastante eficiente na estimação e compensação de movimento, com base em imagens decodificadas anteriormente.

Técnicas de projeção em conjuntos convexos (*projection onto convex sets* - POCS) também são utilizadas no ocultamento de erros. Seu funcionamento, em linhas gerais, se baseia em projeções sucessivas da imagem distorcida aplicadas a conjuntos convexos de restrições definidos *a priori*. As restrições costumam se basear em características desejáveis para uma imagem, tais como suavidade, preservação de arestas, dentre outras.

Algoritmos baseados em POCS podem ser aplicados ao ocultamento de erros em sub-bandas *wavelets* de seqüências comprimidas usando-se JPEG2000 [38]. Outros métodos temporais de ocultamento de erros [39] são aplicados à técnica de compressão de vídeo motion-JPEG2000 [40].

Na literatura também são encontradas técnicas de ocultamento de erros independentes do método utilizado para a codificação. Por exemplo, métodos de ocultamento espaço-temporal podem ser baseados em transformadas *wavelets* [27]. Utilizando a propriedade de regularidade das transformadas *wavelets*, a correlação temporal entre imagens pode ser observada a partir do decaimento dos coeficientes

wavelet. Um modelo matemático descrevendo a evolução temporal dos coeficientes das imagens é utilizado para detectar erros, já que blocos corrompidos tendem a apresentar padrões diferentes na correlação cruzada de seus coeficientes.

3.2 Tratamento de Erros no Padrão H.264

Uma das principais conseqüências de erros no fluxo de *bits* codificado com o H.264 se deve à propagação de erros. Duas características particulares do padrão contribuem para a propagação de erros: o uso de códigos de comprimento variável e as predições. A codificação por entropia com códigos de comprimento variável permite que pequenos erros resultem em perdas de sincronismo do decodificador de entropia, resultando em perdas consideráveis de elementos da imagem. As predições *intra* e *inter*, por sua vez, propiciam a propagação de blocos ou macroblocos com erros ao longo da seqüência de video decodificado.

A codificação por entropia usada no padrão H.264 possibilita que erros de transmissão em uma palavra de código (*codeword*) afetem também as palavras de código subseqüentes, resultando na degradação do vídeo decodificado. Para atender às necessidades de sincronização do decodificador, cada um dos dois níveis da estrutura hierárquica do H.264, imagem e *slice*, utiliza uma palavra código para sinalizar seu início. Ao receber uma palavra código de início, o decodificador se ressincroniza. A recuperação do sincronismo no processo de decodificação evita que eventuais erros nas palavras de código recebidas anteriormente sejam propagados, o que afetaria a decodificação. Dessa forma, durante a decodificação, os erros permanecem confinados no escopo de uma imagem ou *slice*. Contudo, uma palavra código com erros ainda causa efeitos indesejáveis nas palavras subseqüentes do mesmo *slice*, por exemplo. Além disso, os efeitos dos erros de transmissão se traduzem em artefatos nas imagens, que apresentam o agravante de se propagar para as imagens decodificadas em seguida devido à codificação baseada em predições espaço-temporais [31, 41].

Conforme já mencionado anteriormente, são possíveis diferentes abordagens para minimizar o impacto de erros de transmissão em video codificado, atuando principalmente em:

- codificação de canal;

- codificação de fonte robusta a erros;
- detecção e correção de erros;
- detecção e ocultamento de erros (*error concealment*).

Nesta dissertação, o foco foi direcionado para as técnicas de ocultamento de erros na decodificação, aplicadas especificamente a erros que resultem em macroblocos corrompidos. Para tal, assumiu-se que os erros tenham sido detectados *a priori* pelo processo de decodificação, devendo ainda ser tratados por algum tipo de processamento, para torná-los menos perceptíveis e evitar sua propagação.

A detecção de erros no fluxo de *bits* codificado pelo H.264 não constitui o foco desta dissertação. Contudo, a caráter ilustrativo, são citadas a seguir algumas propostas de detecção de erros no escopo de um *slice*. Algumas condições de verificação de erros baseadas em restrições da sintaxe do fluxo de *bits* codificado se baseiam em [31]:

1. Detecção de erros em uma palavra de código da codificação VLC que represente, por exemplo, um coeficiente DCT, um vetor de movimento, o padrão de codificação (*coded block patterns* - CBP), o tipo de macrobloco, a imagem de referência, dentre outros;
2. Número total de macroblocos decodificados em um *slice* não correspondendo ao tamanho do *slice*;
3. Número de coeficientes DCT decodificados de um bloco de 4×4 *pixels* maior que 16;
4. Detecção de informações de vídeo inválidas.

O impacto dos erros na visualização do vídeo decodificado está fortemente relacionado ao efeito dos erros nas menores unidades que representam uma imagem de um sinal de vídeo: os blocos e macroblocos. No caso do padrão H.264 [41], os dados codificados de um macrobloco são divididos em duas partes, denominadas partições de dados (*data partitions* - DP) descritas abaixo:

- **DP1**: Contém o cabeçalho da imagem, o tipo de macrobloco, os modos de predição *intra* e os vetores de movimento;

- **DP2**: Contém CBPs e os coeficientes da DCT.

Para um macrobloco ser decodificado perfeitamente, o recebimento da partição **DP1** íntegra é condição necessária e suficiente [31], enquanto a integridade somente da partição **DP2** não é suficiente.

Neste texto, o não recebimento da partição **DP1** de um macrobloco pelo decodificador será considerado equivalente ao recebimento de um macrobloco corrompido. A técnica de ocultamento de erros implementada no modelo de referência, apresentada a seguir, trata justamente de macroblocos corrompidos.

3.3 Técnicas de Ocultamento de Erros Utilizadas no Modelo de Referência do Padrão H.264

O processo de definição de um padrão de codificação de vídeo envolve uma série de etapas. Primeiramente são estabelecidos requisitos funcionais e de desempenho. As funcionalidades básicas do padrão são geralmente definidas segundo a avaliação de propostas submetidas por instituições interessadas nos resultados dos esforços de padronização. A cada encontro do grupo de padronização, dentre as propostas submetidas são definidas as que serão adotadas ou descartadas. Através desse processo, os detalhes do padrão são gradualmente refinados [18].

No caso de codificação de imagens, o processo envolve a avaliação do desempenho de compressão e processamento de diferentes *codecs* (codificadores e decodificadores). Assim, um *software* de referência é desenvolvido de forma a implementar as funcionalidades definidas pelo processo de padronização. Juntamente com o *software* é também desenvolvido um documento descritivo denominado modelo de testes (*test model*). Ambos são atualizados gradativamente, de forma a incorporar as revisões do padrão. Quando o *software* e o documento alcançam maturidade suficiente, é gerada uma versão preliminar (*draft*) do padrão. Após revisões adicionais, o documento é publicado como padrão internacional.

No caso do H.264, o padrão não especifica o codificador de vídeo, definindo somente a sintaxe do fluxo de *bits* codificado, sua semântica e o processo pelo qual os elementos devem ser decodificados a fim de reconstruir o vídeo. Ao especificar sintaxe e semântica do fluxo de *bits* codificado, são definidos os elementos que o

caracterizam como compatível com o padrão. Para ser considerado compatível, um codificador deve ser capaz de produzir um fluxo de *bits* que seja decodificável pelo processo definido no padrão. Para completar a especificação, o padrão deve definir um *software* decodificador hipotético de referência (*hypothetical reference decoder*) [18].

Teoricamente, os padrões de codificação de vídeo são projetados para permitir interoperabilidade, e não necessariamente qualidade [9]. Essa característica do processo de padronização permite máxima liberdade para se otimizar o desempenho de um decodificador, por exemplo. Assim, as funções de ocultamento de erros na decodificação não se enquadram no escopo do padrão, podendo ser desenvolvidas livremente para otimizar a recuperação de erros, já que não interferem nas especificações do fluxo de *bits* codificado.

Nesta dissertação foram analisadas técnicas de ocultamento de erros do *software* de referência decodificador do padrão H.264, o *jm* (*joint model*) versão 9.6 [10, 11].

As técnicas de ocultamento de erros implementadas no *software* de referência do H.264 atuam no decodificador, sem implementar qualquer alteração na sintaxe de codificação. Na prática, a implementação de ocultamento de erros no decodificador busca compensar efeitos dos erros de transmissão a partir de informações correlacionadas presentes no vídeo decodificado. O algoritmo proposto para o *software* de referência implementa técnicas de ocultamento de erros baseadas em critérios espaciais e temporais, usando correlações entre os macroblocos corrompidos e os macroblocos adjacentes no mesmo quadro, ou também informações de quadros decodificados anteriormente.

Em linhas gerais, no método usado no *software* de referência [11], o ocultamento espacial de erros se baseia em interpolações ponderadas dos *pixels* dos macroblocos vizinhos (*weighted pixel value averaging*), e é aplicado a *slices* codificados usando predição *intra*. Já o ocultamento temporal é aplicado em *slices* codificados usando predição *inter*, e se baseia em estimar o vetor de movimento do macrobloco corrompido a partir dos macroblocos vizinhos, usando critérios de casamento de bordas (*boundary-matching-based motion vector recovery*). Assim, a recuperação dos macroblocos se baseia em estimações e interpolações do conteúdo recebido correta-

mente.

A estratégia de ocultamento de erros implementada assume que o decodificador de fonte descarta *slices* corrompidos ou incompletos. Portanto, algoritmos de detecção de erros devem atuar antes da entrega dos *slices* ao decodificador de fonte, identificando *slices* com erros. No *software* de referência [11], para cada imagem, todos os *slices* recebidos corretamente são decodificados primeiro, e então os *slices* corrompidos são tratados pelos algoritmos de ocultamento de erros. Assim, o processo de ocultamento de erros atua no *loop* da decodificação, reconstruindo cada imagem afetada por erros, de forma a evitar sua propagação para as imagens subseqüentes. Classificar essas técnicas como ocultamento de erros no pós-processamento da decodificação de video não seria apropriado, já que o ocultamento de erros não atua somente após o video ser decodificado por inteiro, mas sim imagem a imagem.

Na prática, todos os macroblocos de cada imagem são verificados durante a decodificação, sendo identificados os macroblocos pertencentes a *slices* corrompidos e os que foram recebidos corretamente. Após a decodificação de todos os *slices* recebidos corretamente para um quadro, caso existam macroblocos sinalizados como corrompidos, as funções de ocultamento de erros são iniciadas.

O algoritmo de ocultamento de erros atua no nível dos macroblocos, regenerando cada macrobloco corrompido. Após um macrobloco ser regenerado, seu estado passa a ser considerado "regenerado" (*error concealed*), ao invés de correto. Assim, não só os macroblocos recebidos corretamente, como também os "regenerados" podem ser usados no processo de ocultamento de erros em macroblocos vizinhos. No *software* de referência [11], macroblocos "regenerados" são usados na regeneração de seus vizinhos somente quando esses não tiverem nenhum macrobloco vizinho recebido corretamente. Nesses casos, uma estratégia de ocultamento de erros mal sucedida pode resultar na propagação de erros para vários macroblocos vizinhos.

A ordem em que os macroblocos corrompidos são regenerados também é importante. No *software* de referência, o ocultamento de erros em uma imagem se inicia com as colunas de macroblocos localizadas nas fronteiras da imagem, depois se movendo coluna a coluna para o interior da imagem. A ordem escolhida para esse processo se deve à importância de se preservar regiões com detalhes nas áreas centrais da imagem. A implementação pressupõe que as regiões externas da imagem

tendem a apresentar menos movimento que as regiões centrais. Já que o ocultamento de erros em regiões com muito detalhe tende a apresentar mais erros, ao se iniciar o processo de ocultamento por regiões externas da imagem, supostamente com menos movimento, ajudaria a prevenir a propagação de erros.

Ao final da decodificação, caso o *slice* seja do tipo I ou SI, são usados métodos espaciais de ocultamento de erros. Caso o *slice* seja do tipo P, são utilizados métodos temporais. *Slices* do tipo B ou SP não são recuperados na versão do *software* de referência utilizada nesta dissertação [10, 11]. Ao final do processo de ocultamento de erros em um macrobloco, este é sinalizado como "regenerado".

Maiores detalhes dos métodos de ocultamento de erros utilizados no *software* de referência são apresentados a seguir.

3.3.1 Ocultamento Espacial

No *software* de referência do padrão H.264 [11,20], o método de ocultamento de erros baseado em critérios espaciais é aplicado somente na regeneração de macroblocos codificados usando predição *intra*. Essa abordagem é coerente com o processo de codificação, já que na codificação a predição *intra* assume correlações espaciais entre os macroblocos da imagem. Com isso, o ocultamento de erros tende a obter bons resultados ao utilizar interpolações espaciais para regenerar macroblocos de *slices* codificados originalmente por predição *intra*.

Os procedimentos espaciais levam em conta que uma imagem natural tende a ter predominância de baixas freqüências, e conseqüentemente, *pixels* de macroblocos corrompidos apresentam poucas variações em relação aos seus vizinhos. As restrições de desempenho dos métodos espaciais incluem a dificuldade de regeneração de macroblocos localizados nas bordas da imagem, e também a possibilidade de perda de *slices* contendo vários macroblocos adjacentes.

O ocultamento espacial é implementado de forma independente para cada componente (Y, C_b e C_r). Para tal, devem ser primeiramente identificados os macroblocos corrompidos, e em seguida executadas funções de interpolação dos *pixels* perdidos, a fim de substituí-los. A regeneração de um macrobloco se inicia somente após a determinação de quais macroblocos adjacentes podem ser usados. Nesse ponto devem ser observadas as restrições quanto aos vizinhos, incluindo os casos de

macroblocos localizados nas bordas da imagem. *A priori* são desconsiderados os macroblocos adjacentes que também apresentem erros.

Com a informação dos macroblocos vizinhos elegíveis para uso no ocultamento de erros, um dado macrobloco corrompido pode ser regenerado. O valor de cada *pixel* do macrobloco corrompido é obtido através da interpolação ponderada de *pixels* de seus macroblocos vizinhos. Os *pixels* usados na interpolação são exatamente os que se encontram nas bordas dos macroblocos vizinhos, na fronteira à esquerda, direita, abaixo e acima do macrobloco corrompido. A Figura 3.5 apresenta o exemplo de *pixels* vizinhos usados para recuperar um determinado *pixel* de luminância de um macrobloco corrompido. O valor do *pixel* regenerado é obtido a partir da média ponderada dos valores desses *pixels* vizinhos, dada a disponibilidade dos macroblocos vizinhos correspondentes. O fator de ponderação utilizado leva em consideração exclusivamente a distância do *pixel* a ser regenerado para cada um desses 4 *pixels* vizinhos. Para cada *pixel* vizinho, o fator de ponderação é inversamente proporcional à sua distância até o *pixel* a ser regenerado.

A Equação 3.1 define a interpolação espacial que regenera um *pixel* de um macrobloco de 16×16 *pixels*.

$$Y_{pixel} = \frac{1}{2(d+1)}((d-y+1)\cdot Y_{acima} + y \cdot Y_{abaixo} + (d-x+1)\cdot Y_{esq.} + x \cdot Y_{dir.}) \tag{3.1}$$

onde d é igual a 16 para regenerar *pixels* de luminância, e 8 para as componentes e cor. As coordenadas x e y podem ser melhor entendidas a partir da Figura 3.5. A regeneração é equivalente para se recuperar as componentes de cor C_b e C_r, substituindo as distâncias de valor 16 por 8.

Cabe ressaltar que esse método de ocultamento espacial de erros não leva em consideração a presença de arestas na imagem, mas tão somente as correlações entre as bordas de macroblocos vizinhos. Isso pode ser considerado uma limitação desse método, já que a integridade de arestas influencia significativamente a percepção de qualidade visual em vídeos.

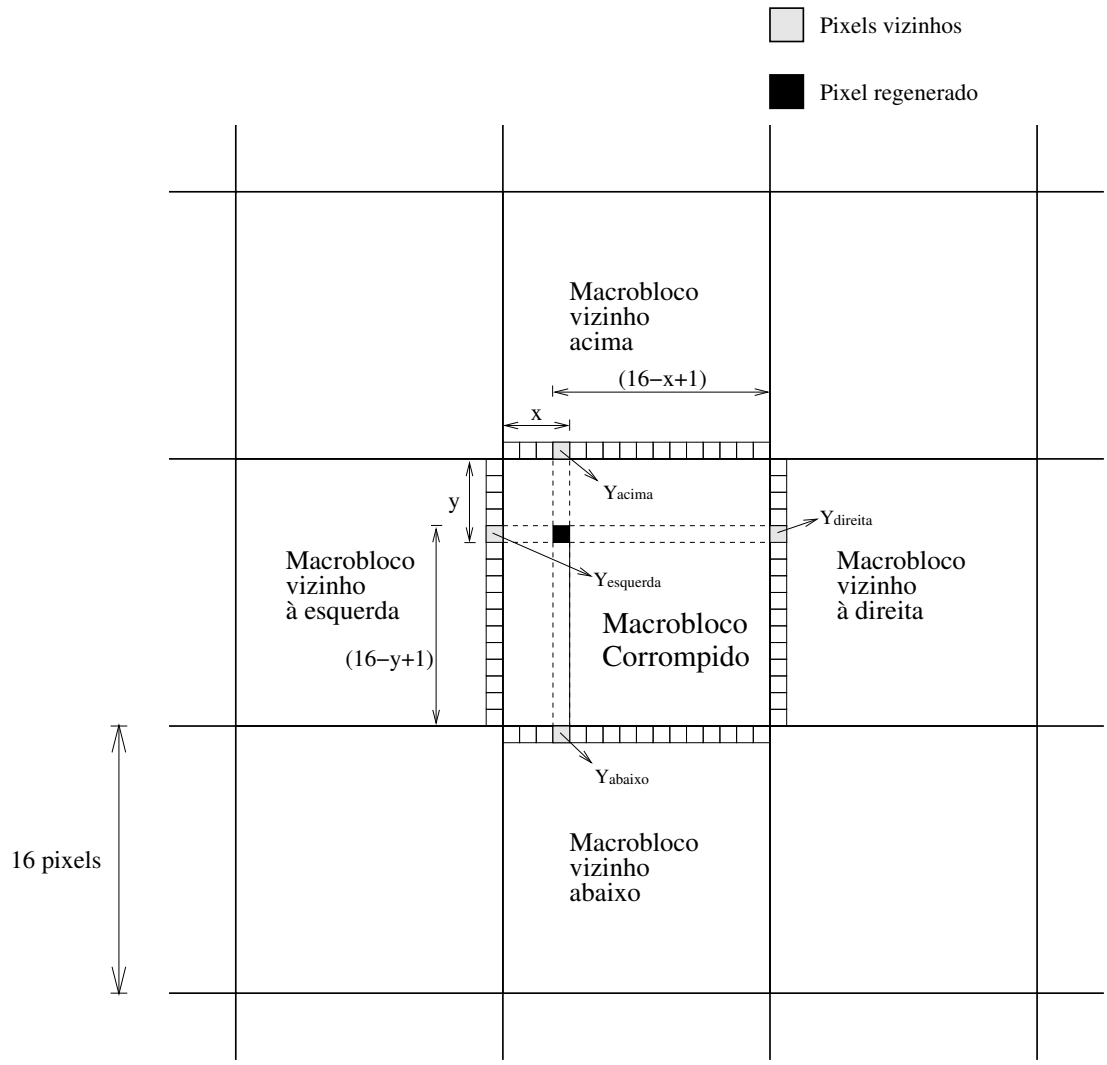


Figura 3.5: Regeneração de um *pixel* de luminância do macrobloco corrompido, usando a técnica de ocultamento espacial de erros implementada no *software* de referência do padrão H.264.

3.3.2 Ocultamento Temporal

Ao invés de buscar somente correlações espaciais dos *pixels*, uma abordagem de ocultamento de erros baseada em critérios temporais foi proposta. O ocultamento temporal envolve a estimação do movimento do macrobloco corrompido, a partir das informações de movimento de macroblocos vizinhos espacial ou temporalmente. O vetor de movimento estimado é então usado para compensar o movimento do macrobloco corrompido com respeito à imagem de referência. Assim, de posse de um vetor de movimento estimado e da imagem de referência, são obtidos os *pixels* que irão substituir o macrobloco corrompido, sem utilizar operações no domínio

espacial da imagem afetada pelo erro.

Primeiramente, o esquema implementado no *software* de referência do padrão H.264 [11] obtém as informações de movimento dos *slices* recebidos corretamente para a imagem dada. Dois métodos distintos de ocultamento de erros podem ser utilizados, sendo esta escolha dependente do movimento médio da imagem recebida com relação à imagem de referência. Na prática, se verifica se o movimento médio por *pixel* da imagem é menor ou maior que um determinado limiar (*threshold*), definido como 1/4 de *pixel* de magnitude. O movimento médio por *pixel* menor que esse limiar indica que a imagem apresenta, em média, pouco movimento com relação à imagem de referência. Então, nesses casos o macrobloco corrompido é subsituído pela cópia dos *pixels* do macrobloco localizado na mesma posição no quadro de referência. Contudo, caso o movimento médio por *pixel* seja maior que esse limiar, é utilizada uma função mais sofisticada de ocultamento, baseada em compensação de movimento. Dessa forma, a metodologia de ocultamento de erros do *software* de referência estima movimento nulo do macrobloco corrompido quando a imagem não apresenta, em média, muito movimento com respeito à imagem de referência. Essa abordagem visa economizar capacidade de processamento.

Por outro lado, a função utilizada no ocultamento de erros em imagens com maior movimento médio busca estimar o movimento do macrobloco corrompido a partir de blocos de 8×8 *pixels* adjacentes espacialmente. Os blocos adjacentes utilizados no ocultamento temporal de erros do *software* de referência são apresentados na Figura 3.6. Essa abordagem se baseia na suposição de que, em imagens naturais, regiões vizinhas espacialmente tendem a apresentar alta correlação de movimento. Esta suposição se justifica por observação, já que macroblocos vizinhos tendem a pertencer ao mesmo objeto, podendo ser representados por um campo contínuo de vetores de movimento.

O vetor de movimento de um macrobloco corrompido é obtido por predição com base nos vetores de movimento de seus macroblocos vizinhos. Um macrobloco vizinho, por sua vez, pode apresentar até 16 vetores de movimento, no caso extremo de ter sido codificado usando predição *inter* com 16 blocos de dimensão até 4×4 *pixels*. Com fins de simplificação, o *software* de referência é obtém um único vetor de movimento para cada bloco de dimensão 8×8 *pixels*. No caso de um bloco 8×8

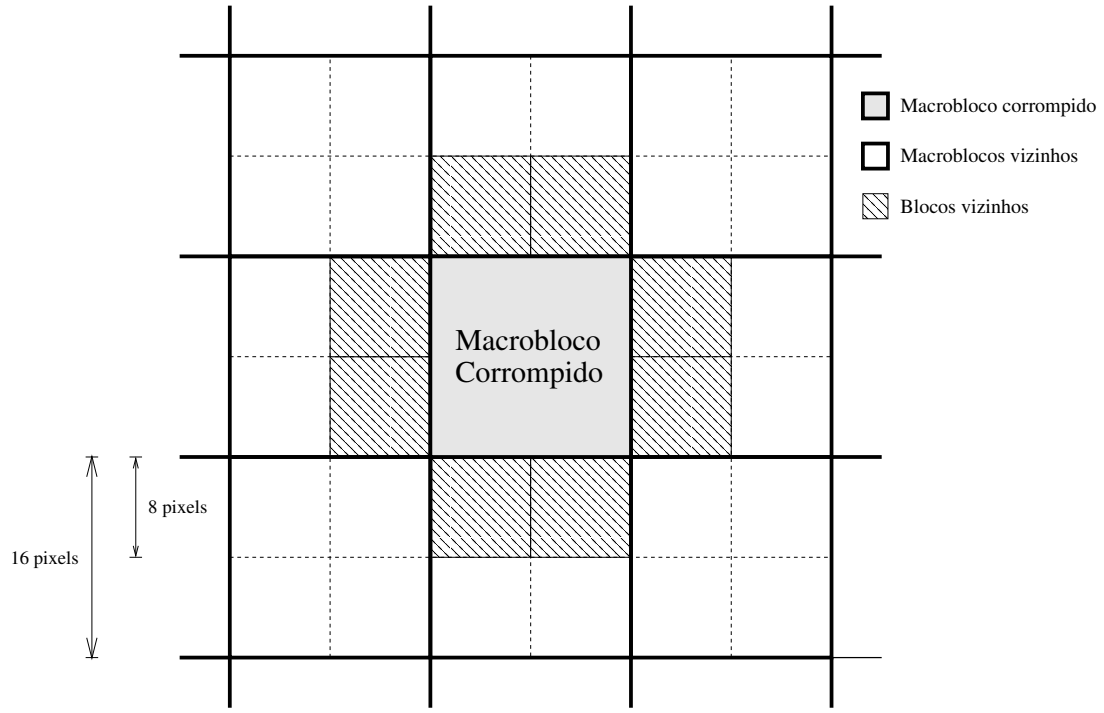


Figura 3.6: Blocos vizinhos ao macrobloco corrompido, cujos vetores de movimento são usados no ocultamento temporal de erros do *software* de referência.

apresentar informações de movimento em seus sub-blocos, o vetor de movimento usado para esse bloco será obtido através da média aritmética dos vetores de seus sub-blocos (dimensões 8×4, 4×8 ou 4×4 *pixels*).

A escolha do vetor de movimento dentre os macroblocos vizinhos é baseada em critérios de suavidade espacial (*spatial smoothness*) da imagem reconstruída. São considerados candidatos todos os vetores de movimento correspondentes a blocos de 8×8 *pixels* vizinhos ao macrobloco corrompido. Iterativamente, cada vetor é utilizado para se obter um macrobloco candidato a substituir o macrobloco corrompido, através da compensação de movimento com relação ao quadro de referência. Para cada estimativa de *pixels* candidatos a substituir o macrobloco corrompido, é analisada a suavidade do casamento com os macroblocos vizinhos. Ao final, é escolhido o vetor que apresentar a menor diferença de luminância através das bordas quando o macrobloco for substituído na imagem. Cabe ressaltar que o vetor de movimento nulo é sempre testado como candidato.

Assim, o vetor de movimento escolhido se refere ao macrobloco candidato que minimiza a distorção de bordas d_{sm} (*side match distortion*). A função de distorção

de bordas objetiva avaliar a continuidade dos *pixels* mais externos do macrobloco candidato a substituir o macrobloco corrompido, com respeito aos macroblocos vizinhos. A função d_{sm} é definida como a soma das diferenças absolutas de luminância entre *pixels* localizados na borda do macrobloco que está sendo estimado, e *pixels* adjacentes nos macroblocos vizinhos (à direita, esquerda, acima e abaixo), descrita pela Equação 3.2 e pela Figura 3.7. Com a avaliação da distorção de bordas, o algoritmo minimiza efeitos de "blocagem" e artefatos na seqüência regenerada.

A Equação 3.2 define a distorção de bordas como:

$$\min_{dir \in \{acima,abaixo,direita,esquerda\}} \arg \langle d_{sm} = \frac{1}{N}\sum_{j=1}^{N} |\tilde{Y}(mv^{dir})_j^{IN} - Y_j^{OUT}| \rangle \qquad (3.2)$$

onde $\tilde{Y}(mv^{dir})_j^{IN}$ é o *j-ésimo* valor de luminância Y da borda do macrobloco regenerado usando o vetor de movimento mv^{dir}; enquanto Y_j^{OUT} é o *j-ésimo* valor de luminância da borda do macrobloco vizinho, e N é o número total de *pixels* de borda calculados. O índice j se refere à posição de 2 *pixels* adjacentes, um pertencente à borda do macrobloco candidato, e o outro à borda do macrobloco vizinho.

A descrição do método de ocultamento temporal de erros realizada acima se aplica somente a macroblocos pertencentes a *slices* do tipo P. A versão 9.6 do *software* de referência [10,11], utilizada nesta dissertação, não suporta o ocultamento de erros em macroblocos pertencentes a *slices* do tipo B e SP.

O funcionamento desses métodos é severamente prejudicado quando há perda de *slices* contendo boa parte dos macroblocos da imagem, o que impossibilita a estimação dos vetores de movimento dos macroblocos corrompidos através dos vetores de movimento dos macroblocos vizinhos.

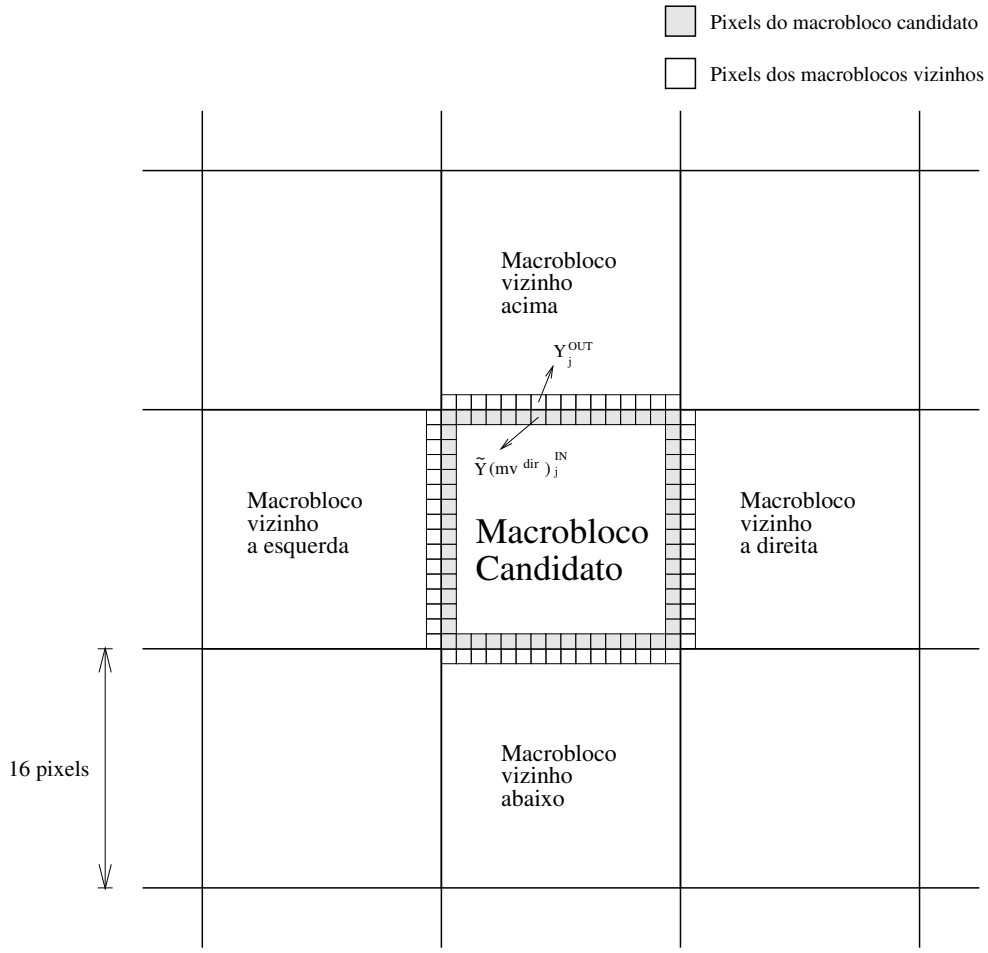


Figura 3.7: Pixels de luminância usados no cálculo da métrica de distorção de bordas de um macrobloco candidato, para o ocultamento temporal de erros implementado no *software* de referência.

Capítulo 4

O Método Proposto de Ocultamento de Erros

Este capítulo apresenta o método de ocultamento de erros proposto nesta dissertação, com a descrição de sua motivação e princípios de funcionamento.

Em seguida, são apresentados os resultados experimentais obtidos, incluindo sua avaliação de desempenho em comparação com o método utilizado no modelo de referência do padrão H.264.

4.1 O Método Proposto

O método proposto nesta dissertação objetiva minimizar o impacto de erros incidentes no fluxo de *bits* de seqüências de vídeo codificadas usando o padrão H.264. As técnicas de ocultamento de erros utilizadas atuam em erros ocorridos na transmissão, que resultem na presença de macroblocos corrompidos nas imagens da seqüência decodificada. O método proposto objetiva minimizar esses erros ao longo do processo de decodificação, com a substituição dos macroblocos corrompidos, minimizando assim os efeitos da propagação de erros. O método não é implementado no pós-processamento, mas sim no *loop* da decodificação.

As técnicas propostas para o ocultamento de erros em macroblocos levam em consideração alguns pressupostos. Assumiu-se que o algoritmo de ocultamento de erros atuará em erros que impactam somente na visualização incorreta dos macroblocos, desconsiderando-se erros severos no fluxo de *bits* que impossibilitem a

decodificação de partes maiores do vídeo, como quadros. Isso não impede que essa abordagem possa ser considerada para um sistema prático, já que a detecção de erros na transmissão, e conseqüentes tentativas de correção podem ser realizadas por outras etapas do sistema de comunicação.

A proposta deste método foi motivada pela busca de melhor desempenho no ocultamento de erros em macroblocos, tendo como parâmetro de comparação o método implementado no *software* de referência do padrão H.264 [10, 11]. Seu objetivo é aumentar a acurácia do processo de ocultamento de erros, principalmente em regiões com muitos detalhes de movimento. Analisando a literatura do assunto, decidiu-se que os requisitos de desempenho perseguidos seriam a melhoria na PSNR das imagens e, também, a diminuição dos efeitos de propagação de erros. A complexidade computacional não foi considerada foco das análises de desempenho do método proposto no escopo desta dissertação, contudo deve ser estudada em trabalhos futuros.

O método proposto se baseia no ocultamento temporal de erros, ou seja, na regeneração do sinal de vídeo com base nas correlações temporais presentes na própria seqüência de imagens. A escolha dessa abordagem está relacionada à predominância de imagens codificadas usando predição *inter* nos vídeos codificados com o H.264. O uso predominante desse tipo de predição produz taxas de compressão de vídeo mais eficientes, pois uma imagem codificada usando predição *inter* necessita, em média, de um número menor de *bits* para ser representada, comparada à codificação *intra*. Dessa forma, concluiu-se que melhorias no ocultamento temporal representariam maior impacto no desempenho do ocultamento de erros em macroblocos.

Como descrito na seção 3.3, o método temporal de ocultamento de erros do *software* de referência regenera um macrobloco corrompido a partir do levantamento da tendência de movimento de sua vizinhança. A implementação se baseia em estimação e compensação de movimentos do macrobloco corrompido com relação ao quadro de referência. Para tal, o método do *software* de referência utiliza somente os vetores de movimento dos blocos de 8×8 *pixels* adjacentes ao macrobloco corrompido. O critério de escolha do melhor candidato se baseia na avaliação da distorção de borda (*side match distortion*) dos *pixels* do macrobloco candidato e seus vizinhos

adjacentes na imagem afetada pelo erro.

Algumas limitações foram percebidas na heurística de ocultamento de erros do *software* de referência. Para minimizá-las, o método proposto implementou duas principais modificações: ampliou o conjunto de vetores de movimento vizinhos testados, e agregou outros critérios de decisão para a escolha do vetor de movimento capaz de melhor estimar o macrobloco corrompido.

Na codificação de vídeo usando o padrão H.264, os macroblocos são particionados em conjuntos de tamanhos flexíveis de *pixels* na estimação e compensação de movimentos da predição *inter*. Cabe ressaltar que, neste texto, os blocos e sub-blocos de um macrobloco serão referidos como partições. Em comparação com padrões anteriores, no H.264 cada macrobloco codificado "contém" uma maior quantidade de informação, devido ao uso de partições de tamanho variável e múltiplas imagens de referência. Assim, cada macrobloco pode apresentar até 16 vetores de movimento, 4 imagens de referência e um modo de predição. Apesar de até 16 vetores de movimento por cada macrobloco poderem ser transmitidos, os algoritmos de ocultamento de erros implementados no *software* de referência do H.264 [10, 11] não exploram totalmente essas informações de movimento. No *software* de referência, a estimação de movimento do macrobloco corrompido utiliza os vetores dos blocos de 8×8 *pixels* adjacentes ao macrobloco corrompido (Figura 3.6). Contudo, para cada bloco 8×8, caso haja vetores de movimento de suas partições (8×4, 4×8 e 4×4 *pixels*), é extraído somente o seu valor médio. Essa abordagem não permite testar o movimento real de partições vizinhas menores que 8×8 *pixels*.

Já o método proposto nesta dissertação amplia o conjunto de vetores de movimento vizinhos testados. Para isso, utiliza os vetores de todas as partições dos macroblocos vizinhos ao macrobloco corrompido, e não somente o movimento médio de partições adjacentes de tamanho 8×8 *pixels*. Essa característica do método proposto visa melhorar o desempenho do ocultamento de erros com respeito ao *software* de referência, buscando aumentar a acurácia da estimação de um campo de movimento das vizinhanças do macrobloco corrompido. Para tal, tentaremos escolher vetores dos macroblocos vizinhos que tenham a mesma tendência de movimento do macrobloco corrompido. A hipótese básica assume que, quanto maior a quantidade de partições formando um conjunto de blocos com movimento coerente, maior a pro-

babilidade que esse movimento seja o mesmo do macrobloco corrompido. Ademais, a coerência de movimento entre um conglomerado e o macrobloco corrompido pode indicar que ambas as regiões de *pixels* pertençam a um mesmo objeto da imagem, ou também a objetos com movimento correlacionado.

Associado ao uso desse maior conjunto de vetores de movimento, o método proposto os organiza em regiões de busca segundo um critério de proximidade baseado na distância euclidiana entre os vetores. Vetores de movimento vizinhos "similares" são classificados como pertencendo ao mesmo conjunto. Cada um desses conjuntos agrupa vetores correlacionados, que, por exemplo, pertençam a um mesmo objeto em movimento de translação. Neste texto, esses conjuntos de vetores de movimento e partições de *pixels* dos macroblocos vizinhos ao corrompido são denominados conglomerados, ou *clusters*. Como os movimentos não podem ser representados como sendo exatamente de translação, os vetores de um mesmo objeto não serão iguais. Assim sendo, é preciso estimar o movimento de um bloco perdido a partir do maior número possível de vetores de um objeto. Para isso é usado o fecho convexo, ou *convex hull*, que será melhor detalhado mais adiante.

A segunda principal modificação proposta para o método do *software* de referência está relacionada aos critérios de escolha do vetor capaz de melhor substituir, por compensação de movimento, o macrobloco corrompido. No *software* de referência, o único critério utilizado para escolher o macrobloco substituto do macrobloco corrompido é a distorção de bordas (*side match distortion*). Maiores detalhes sobre o critério de distorção de bordas podem ser obtidas pela Figura 3.7 e pela Equação 3.2.

De forma a garantir maior flexibilidade na escolha do vetor de movimento que melhor regenere o macrobloco corrompido, o método proposto apresenta duas possibilidades de métricas:

- Dentre todos os conglomerados, é selecionado o vetor que proporciona o melhor casamento *pixel* a *pixel* (SAD - *sum of absolute distortion*) das partições que compõem o conglomerado na imagem de referência;

- Para cada conglomerado, é primeiramente selecionado o vetor que proporciona o melhor casamento *pixel* a *pixel* (SAD) do conglomerado na imagem de referência. Tem-se então o vetor representativo do conglomerado. Para escolher

entre os vetores representativos de cada conglomerado, é usado o critério de distorção de bordas. Dessa forma, o vetor que proporciona o melhor casamento de bordas do macrobloco substituto na imagem afetada pelo erro é utilizado;

Cabe ressaltar que o método proposto também pode ser considerado como uma variante dos algoritmos temporais de ocultamento de erros baseados no melhor casamento de regiões vizinhas (*best neighborhood matching* [29–31]). Conforme já mencionado, o método proposto nesta dissertação classifica partições de macroblocos vizinhos segundo um critério de coerência do campo de movimento. Assim, partições dos macroblocos vizinhos com tendência de movimento similar são classificadas em conjuntos denominados conglomerados. A região de busca (*searching range*) do casamento de um conglomerado na imagem de referência é dada pelo fecho convexo obtido a partir dos vetores de movimento do próprio conglomerado.

A idéia de se buscar o melhor casamento dos conglomerados na imagem de referência se baseia na tentativa de estimar as regiões vizinhas que, juntamente com o macrobloco corrompido, pertenceriam a um mesmo objeto ou a objetos com mesma tendência de movimento. Essa abordagem é corroborada pelo fato de que, em seqüências de imagens naturais, um objeto em movimento geralmente consiste em uma área homogênea que cobre vários blocos pequenos. Na prática, blocos espacialmente vizinhos tendem a pertencer a um mesmo objeto, e a se mover na mesma direção, e com a mesma velocidade. Assim, sob o ponto de vista da codificação, vetores de movimento de blocos espacialmente vizinhos tendem a ser altamente correlacionados.

Maiores detalhes sobre o método proposto são apresentados a seguir.

4.2 Descrição da Implementação

O método de ocultamento de erros proposto nesta dissertação foi implementado através de modificações no *software* de referência decodificador do padrão H.264, o jm (*joint model*) versão 9.6 [10,11]. Primeiramente, o *software* de referência foi adaptado para simular erros incidentes em macroblocos, foco dos métodos de ocultamento estudados nesta dissertação. Essa adaptação do *software* foi usada para dar maior flexibilidade aos testes, permitindo que o usuário do decodificador

especifique as imagens e macroblocos onde devem ser "introduzidos" erros, por meio de parâmetros de entrada do *software* do método proposto.

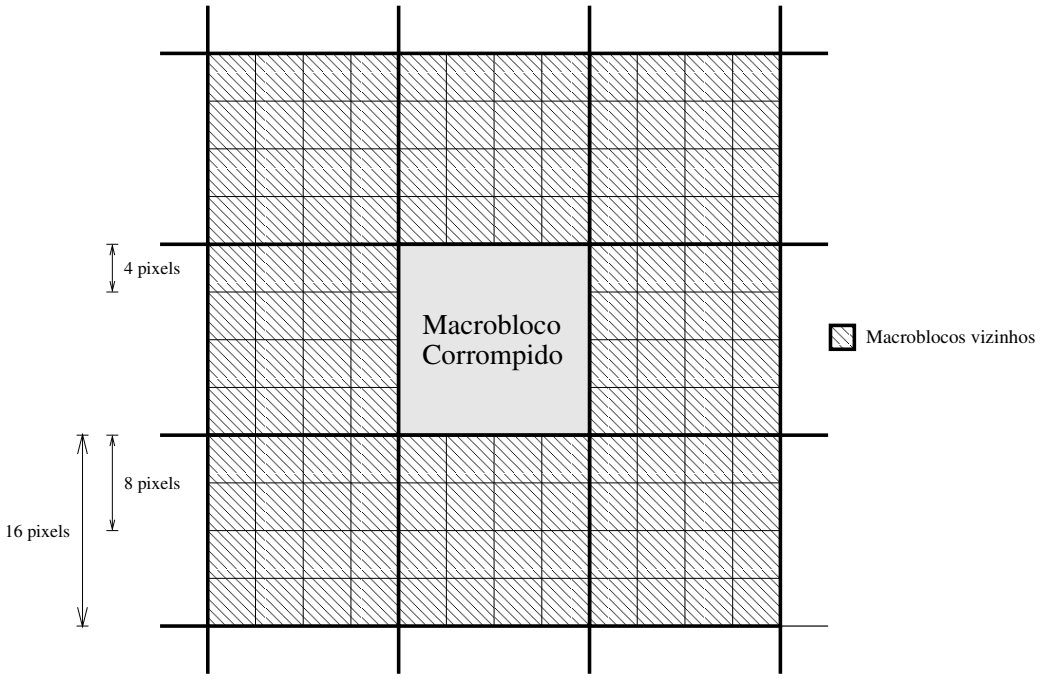
Essa simulação de erros foi implementada para avaliar especificamente o desempenho dos métodos de ocultamento de erros em macroblocos. Outras opções mais completas de simulação de erros no fluxo de *bits* H.264 vêm sendo implementadas por outros autores. O grupo de trabalho JVT, por exemplo, desenvolveu um *software* auxiliar que insere erros no fluxo de *bits* codificado. Contudo, esses erros são modelados especificamente segundo padrões experimentais de erros de transmissão de vídeo codificado via internet [42].

Após se adaptar o *software* de referência para simular erros em macroblocos, foi implementado o método de ocultamento de erros proposto. A heurística do método proposto é apresentada no Algoritmo 4.1, e também descrito nas etapas apresentadas a seguir.

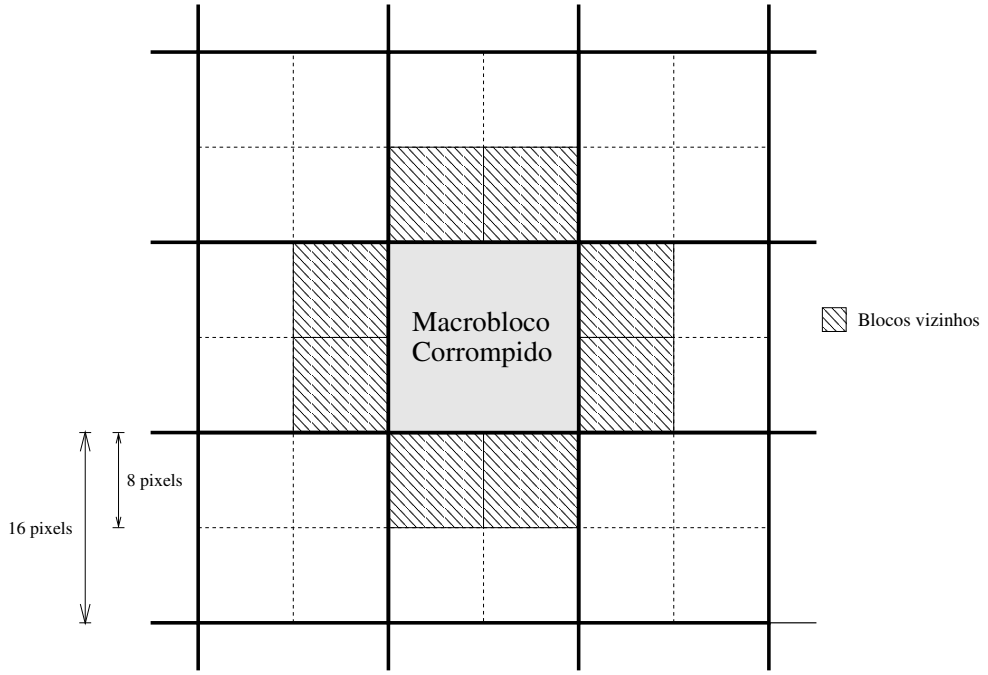
No passo 1, dado um macrobloco perdido o algoritmo verifica a disponibilidade de todos os macroblocos vizinhos, descartando os macroblocos vizinhos que também estiverem corrompidos. Somente quando um macrobloco corrompido não tiver nenhum macrobloco vizinho correto, seus vizinhos já regenerados são utilizados.

No passo 2 são selecionados os macroblocos vizinhos disponíveis que tenham sido codificados usando predição *inter*. Os vetores de movimento são obtidos de todas as partições existentes nos macroblocos vizinhos, sejam de dimensão 16×16, 16×8, 8×16, 8×8, 8×4, 4×8 ou 4×4 *pixels*. Comparado ao método do *software* de referência, o método proposto leva em consideração uma região vizinha maior, incluindo todos os 8 macroblocos vizinhos. A Figura 4.2 ilustra as diferentes regiões vizinhas utilizadas pelo método proposto e pelo *software* de referência.

No passo 3 do Algoritmo 4.1, os vetores de movimento são classificados em conjuntos segundo um critério de proximidade. A distância euclidiana máxima, parâmetro de entrada do *software*, é utilizada para classificar os vetores em conjuntos denominados conglomerados. Para um vetor pertencer a um conglomerado, sua distância euclidiana com relação a todos os outros vetores já pertencentes ao conjunto deve ser menor que a máxima distância especificada pelo usuário. Dessa forma, o algoritmo garante que os vetores pertencentes a um determinado conglomerado



(a) Método proposto



(b) *Software* de referência

Figura 4.1: Macroblocos e/ou partições vizinhas utilizados pelos diferentes métodos no ocultamento de erros.

são fortemente correlacionados. Contudo, esse método de classificação é fortemente influenciado pela ordem em que os vetores são classificados. A ordem utilizada está relacionada com a ordem em que os vetores são coletados dos macroblocos vizinhos.

Ao final do processo de classificação dos vetores em conglomerados, temos a indicação das partições dos macroblocos vizinhos que pertencem a cada conglomerado. A Figura 4.2 exemplifica um macrobloco corrompido, rodeado por macroblocos, blocos e sub-blocos pertencentes a 3 diferentes conglomerados. O macrobloco inválido indicado na Figura 4.2 se refere a um macrobloco que também tenha sido corrompido.

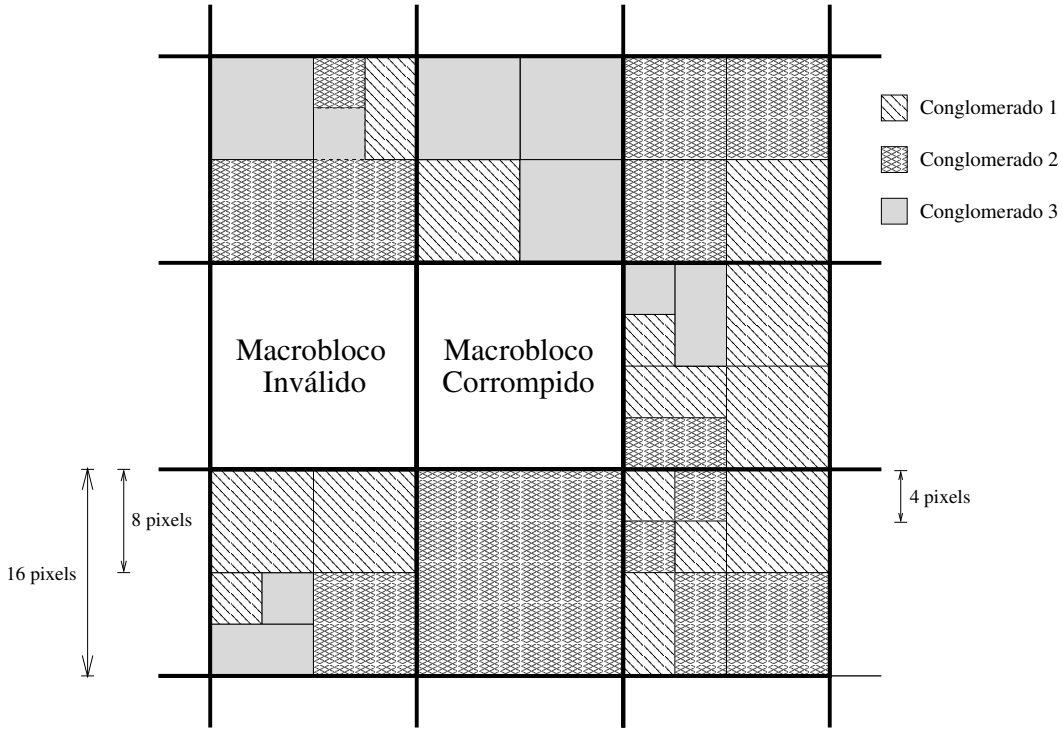


Figura 4.2: Exemplo de mapeamento de macroblocos, blocos e sub-blocos vizinhos em diferentes conglomerados.

Após a classificação dos vetores em conglomerados, no passo 4 se inicia o processo de estimação do vetor de movimento que será usado para substituir o macrobloco corrompido. Os vetores de movimento de cada conglomerado são utilizados para se obter um vetor de movimento candidato a substituir o macrobloco corrompido, através de compensação de movimento com respeito à imagem de referência. O conjunto de vetores de teste não se restringe somente aos vetores de movimento do conglomerado, mas sim a um conjunto ampliado de vetores obtidos a partir des-

tes. O aumento do conjunto de vetores é motivado pela demanda por se estimar um campo de movimento coerente a partir das regiões vizinhas ao macrobloco corrompido. Para isso, os vetores de movimento de cada conglomerado geram uma região de busca na forma de fecho convexo. Assim, o novo conjunto de vetores de teste passa a ser composto por todos os vetores pertencentes ao fecho convexo.

Um fecho convexo (*convex hull*) de um conjunto de pontos é definido como o menor conjunto convexo que inclua esses pontos [43]. Para um conjunto bi-dimensional finito, um fecho convexo é um polígono convexo. A Figura 4.3 apresenta exemplos da formação de fechos convexos em duas dimensões, gerados a partir dos pontos indicados.

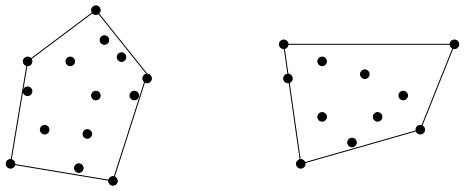


Figura 4.3: Formação de fechos convexos bi-dimensionais a partir de conjuntos de pontos.

Na implementação do método de ocultamento de erros proposto, temos o agrupamento de vetores em fechos convexos. A Figura 4.4 exemplifica um fecho convexo formado a partir de 4 vetores. A região hachurada indica o interior do fecho convexo, limitado pela linha tracejada.

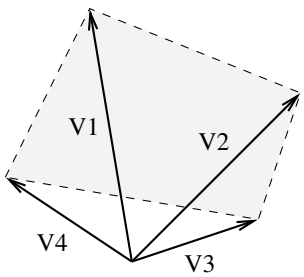


Figura 4.4: Exemplo de classificação de vetores em um fecho convexo.

O cálculo de um fecho convexo a partir de um conjunto de pontos é um problema já bastante explorado na literatura. Alguns algoritmos de cálculo de fechos convexos são: *QuickHull*, *Divide-and-Conquer*, e *Monotone Chain* [44]. Nesta implementação foi utilizado o algoritmo *Graham Scan* [45], criado em 1972 por R.

L. Graham. Esse foi o primeiro algoritmo publicado capaz de calcular um fecho convexo no plano, com complexidade $O(n \log(n))$ no pior caso. Em 1981, A. C. Yao provou que o algoritmo Graham Scan é ótimo em relação à complexidade no pior caso. No entanto, algoritmos *Graham Scan* não apresentam uma extensão trivial para 3 dimensões. A razão se deve ao fato do algoritmo depender de uma ordenação angular, que não tem equivalente para 3 dimensões.

Dado um conjunto de pontos no plano, o algoritmo *Graham Scan* [45] calcula o fecho convexo em 3 etapas, descritas com maiores detalhes no Algoritmo 4.2.

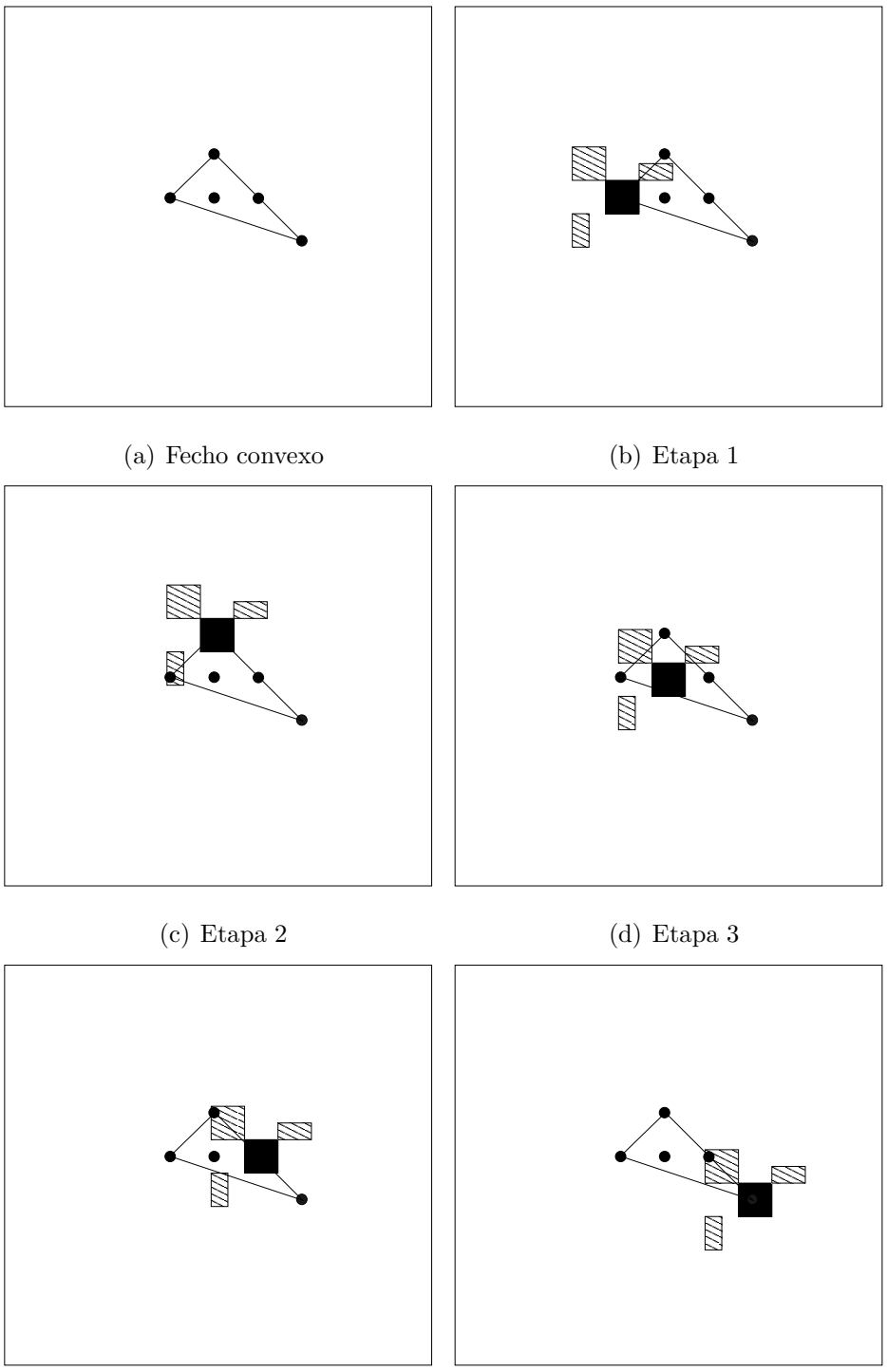
Conforme já mencionado, os vetores de um conglomerado passam a delimitar uma região convexa de vetores de testes. Estes vetores pertencentes ao fecho convexo são usados para testar o casamento do conglomerado na imagem de referência (*block matching*). O teste de casamento das partições de *pixels* de um conglomerado com respeito à imagem de referência é feito através do somatório da diferença absoluta (*sum of absolute differences* - SAD) entre os *pixels* das partições do conglomerado e a região correspondente na imagem de referência, obtida por compensação de movimento. O conglomerado representa uma região irregular de *pixels*, vizinha ao macrobloco corrompido. No teste de casamento, as distâncias relativas entre as partições do conglomerado se mantêm constantes.

A Figura 4.5(a) ilustra os vetores pertencentes ao fecho convexo relativo a um conglomerado. As figuras 4.5(b) a 4.5(f) apresentam as etapas do teste de casamento das partições de um conglomerado, para todos os vetores de movimento pertencentes ao fecho convexo relativo a esse conglomerado.

A Figura 4.6 também ilustra o teste de casamento de um conglomerado formado por 3 partições, com respeito à imagem de referência.

Cabe ressaltar que, ao se testar o casamento de um conglomerado na imagem de referência, se busca estimar o vetor de movimento mais adequado para regenerar o macrobloco corrompido. Esse vetor é denominado vetor representativo do conglomerado. No passo 5, o algoritmo deve escolher o vetor que será utilizado para substituir o macrobloco corrompido, dentre os vetores representativos de cada conglomerado. Para essa escolha são usadas duas heurísticas possíveis. A primeira considera somente o melhor resultado de SAD dentre os conglomerados, enquanto a segunda associa o critério de SAD com a continuidade do macrobloco substituto

com relação às suas vizinhanças. Essas métricas são apresentadas em mais detalhes a seguir.



(a) Fecho convexo (b) Etapa 1

(c) Etapa 2 (d) Etapa 3

(e) Etapa 4 (f) Etapa 5

Figura 4.5: Etapas da varredura de um fecho convexo na imagem de referência, usando as partições de um conglomerado.

A métrica de escolha do vetor representativo de cada conglomerado considera

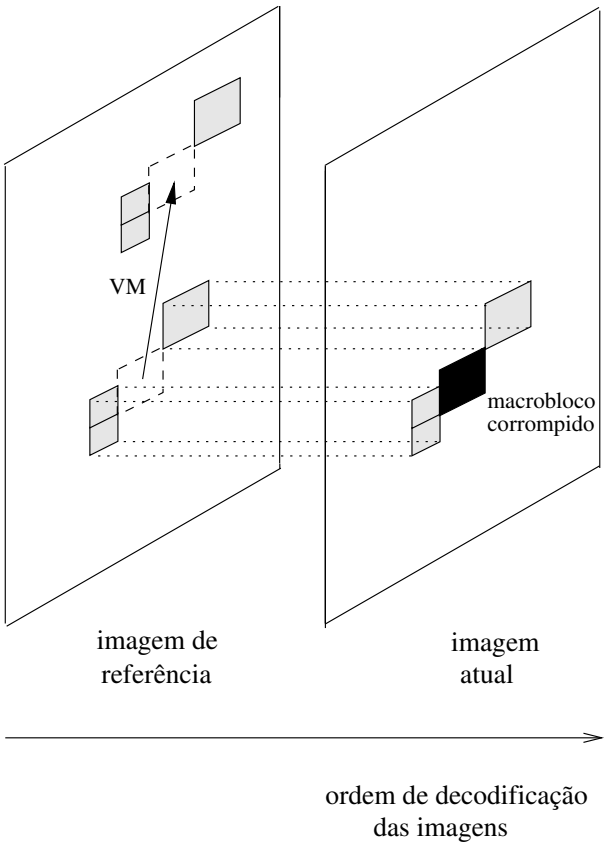


Figura 4.6: Casamento (*block matching*) de um conglomerado na imagem de referência

somente o critério de melhor casamento dos *pixels* do conglomerado na imagem de referência, usando a região de busca delimitada pelos vetores do fecho convexo do próprio conglomerado. Para cada conglomerado, é selecionado o vetor de movimento que resulte no melhor casamento de suas partições com respeito à imagem de referência. O critério de melhor casamento é definido pela menor somatório de diferenças absolutas (*sum of absolute differences* - SAD) entre os *pixels* das partições na imagem atual, e os *pixels* resultantes da sua compensação de movimento na imagem de referência.

Contudo, ao final deste processo, somente um vetor dentre os vetores representativos dos conglomerados deve ser utilizado para se estimar os *pixels* que irão substituir o macrobloco corrompido. Nesse passo foram implementadas duas métricas possíveis para a seleção desse vetor. A primeira escolhe, para cada conglomerado, o vetor que permite o melhor casamento das partições do conglomerado na imagem de referência. Já a segunda métrica, agrega a este critério a avaliação das

distorções de borda dos macroblocos candidatos a substituir o macrobloco corrompido. Dados os vetores de movimento representativos de cada conglomerado, *pixels* candidatos a substituir o macrobloco corrompido são obtidos por compensação de movimento, tal como indicado na Figura 3.3. Para cada vetor, é testado o critério de distorção de bordas.

Esse critério de seleção do vetor representativo, dentre os candidatos, tem como objetivo preservar a continuidade espacial do macrobloco substituto. Na prática, são implementados testes de distorção de bordas que identificam o melhor casamento do macrobloco substituto com respeito a imagem afetada pelo erro. A distorção obtida a partir de cada vetor de movimento testado é definida como a soma ponderada das diferenças de luminância dos *pixels* através da fronteira entre o macrobloco candidato e seus macroblocos adjacentes vertical e horizontalmente, tal como descrito na Figura 3.7 e na Equação 3.2.

Para qualquer das métricas propostas, no passo 6 do algoritmo 4.1 o vetor de movimento que levar ao resultado ótimo é utilizado para se obter um macrobloco substituto do macrobloco corrompido. A substituição é resultado da compensação de movimento a partir da imagem de referência, tal como indicado na Figura 3.3.

No passo 7 do algoritmo 4.1, o macrobloco originalmente corrompido já tendo seus *pixels* substituídos, é sinalizado como regenerado para o ocultamento dos macroblocos seguintes.

Cabe ressaltar que a estimação de movimento utiliza somente *pixels* de luminância. Contudo, o vetor escolhido é usado para regenerar *pixels* de luminância e crominância dos macroblocos corrompidos, por compensação de movimentos. O processo de compensação de movimento utiliza vetores e interpolações com acurácia de até 1/4 de pixel.

Maiores informações sobre a implementação do *software*, bem como instruções de uso são apresentadas no apêndice A do método proposto.

4.3 Resultados Experimentais

Vários parâmetros podem influenciar a percepção visual da perda de macroblocos no sinal de vídeo decodificado. Aspectos relevantes incluem a presença de

bastante movimento, a ocorrência de mudanças de cena, e também se uma imagem é muito usada como referência na predição de outras imagens. A quantidade de macroblocos codificados por predição *intra* ou *inter* numa dada imagem também influencia no impacto dos erros. Macroblocos codificados por predição *intra* tendem a ser melhor recuperados por técnicas espaciais de ocultamento de erros, o que se deve ao fato de a predição *intra* se basear em correlações espaciais. A taxa de compressão também pode ser relevante, já que influencia na qualidade do vídeo decodificado, e também nos produtos do processo de predição, tais como os valores dos vetores de movimento e dos resíduos de predição.

Os experimentos realizados tiveram como objetivo principal utilizar o método do *software* de referência como parâmetro de comparação para os métodos de ocultamento de erros propostos. Na apresentação dos resultados obtidos, a métrica que avalia somente a SAD dos conglomerados com respeito ao quadro de referência (ver seção 4.1) será referida como métrica **A**, enquanto a métrica que associa o critério de SAD com a continuidade do macrobloco substituto com relação às suas vizinhanças (ver seção 4.1) será referida como **B**.

A fim de que a comparação fosse realizada de forma criteriosa, alguns cenários representativos foram escolhidos como ambiente de simulação.

4.3.1 Medida Objetiva de Qualidade

A avaliação dos resultados experimentais foi baseada em uma medida objetiva, a PSNR (*peak signal-to-noise ratio*), dada pela Equação 4.1.

$$\mathrm{PSNR} = 10 \log_{10} \frac{M^2}{\frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|I(i,j) - K(i,j)\|^2} \ , \tag{4.1}$$

onde $I(i,j)$ e $K(i,j)$ representam valores dos *pixels* nas posições i e j das imagens I e K que estão sendo comparadas. A variável m representa o número de linhas das imagens, n o de colunas, e M o valor máximo que um *pixel* pode assumir. A PSNR pode ser aplicada tanto para comparar valores de luminância quanto de crominância.

Nas simulações analisadas, a PSNR é usada para comparar imagens correspondentes duas a duas, uma pertencente à seqüência original (não-codificada) e a outra à seqüência decodificada (afetada por perdas de macrobocos, e regenerada pe-

los métodos de ocultamento de erros). Os valores de PSNR apresentados se referem à PSNR da luminância.

4.3.2 Configurações das Simulações

As simulações foram realizadas com as seqüências de vídeo "foreman" e "silent" [46], com as sequintes configurações:

- formato QCIF (176×144 *pixels*);
- 300 quadros;
- vídeo progressivo (não entrelaçado);
- codificação usando um *slice* por quadro, do tipo I-P-B-P-B, com intervalo de 12 quadros entre *slices* tipo I;
- codificadas usando o H.264 perfil *main* a taxas constantes de 384kbps.

Por se tratarem de testes em seqüência não entrelaçadas, as imagens serão referidas como quadros.

Cabe ressaltar que a escolha de testes com seqüências de baixa resolução, como QCIF, se deve ao fato de serem adequados à transmissão *wireless* de vídeo, uma das aplicações que mais requer o uso de técnicas de ocultamento de erros.

4.3.3 Cenários das Simulações

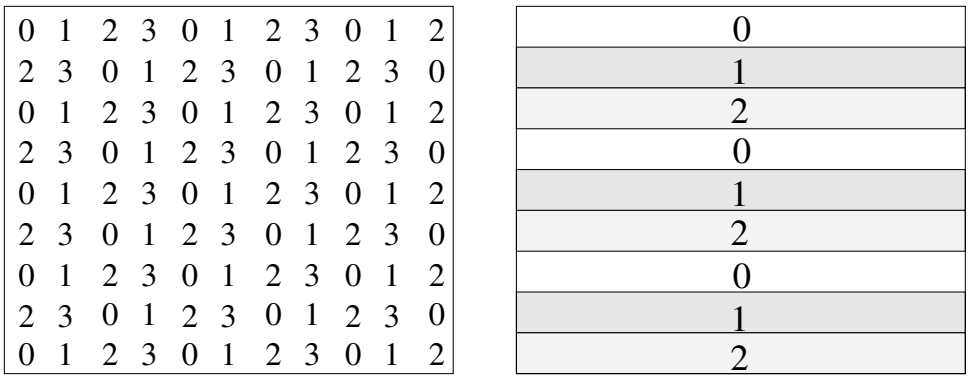
A eficiência do processo de ocultamento de erros está fortemente relacionada às características de movimento do vídeo decodificado, ou seja, um dado método de ocultamento de erros pode ser mais eficiente para regiões de muito movimento, e pouco eficiente para regiões de pouco movimento, ou vice-versa. A fim de comparar os diferentes métodos de forma mais adequada, buscou-se analisar o desempenho médio do ocultamento de erros em macroblocos.

Resultados médios foram extraídos de realizações dos seguintes casos de configurações de erros:

1. Erro em um único macrobloco por quadro, referenciado como **erro em macrobloco individual**;

2. Erro em uma linha de macroblocos adjacentes por quadro, referenciado como **erro em linha**.

Ambas as simulações são representativas de situações reais. A ocorrência de erros em macroblocos individuais representa a perda de *slices* do tipo *dispersed*, enquanto os erros em linhas de macroblocos adjacentes na horizontal representam a perda de *slice groups* do tipo *interleaved*. A Figura 4.7(a) ilustra uma imagem contendo macroblocos pertencentes a 4 diferentes *slices* na configuração *dispersed*, enquanto a Figura 4.7(b) representa a configuração *interleaved* para uma imagem com 3 *slices*.



(a) *Dispersed* (b) *Interleaved*

Figura 4.7: Tipos de agrupamento de macroblocos em *slices* (*slice groups*) avaliados na simulação de erros.

De forma a extrair mais conclusões sobre o comportamento médio das realizações desses erros, os resultados referentes à cada configuração foram compostos de duas maneiras distintas:

1. PSNR calculada a partir da média de realizações de erros no escopo de apenas um quadro da seqüência, denominada **média em um quadro**;

2. PSNR calculada a partir da média de realizações de erros em macroblocos pertencentes a *slices* P de todos os quadros da seqüência, denominada **média na seqüência**

Duas etapas de simulações foram implementadas. Primeiramente, foram gerados testes de **média em um quadro**, a fim de observar o comportamento do

método proposto em casos particulares de erros no escopo de um quadro. A partir destes testes, se verificou o desempenho comparativo entre as métricas A e B do método proposto, e também foi definido o parâmetro "distância euclidiana"das simulações posteriores.

Comparações mais criteriosas do comportamento do método proposto com o do *software* de referência foram obtidas com os testes de **média na seqüência**, usando os valores de "distância euclidiana" determinados previamente. O limiar máximo para distância euclidiana entre os vetores é usado na organização em conglomerados dos vetores de movimento vizinhos ao macrobloco corrompido. Diferentes valores para o limiar de distância euclidiana entre vetores de um mesmo conglomerado foram testados, verificando a eficiência do método proposto para regiões de busca mais restritas ou mais abrangentes.

As simulações de configurações de erros foram aplicadas somente a macroblocos pertencentes a *slices* do tipo P, restrição da implementação de ocultamento de erros do *software* de referência.

4.3.4 Cenário 1: Média em um Quadro

As simulações da **média em um quadro** tiveram como objetivo observar a eficiência do método proposto em evitar a propagação de erros ao longo da seqüência de vídeo, para casos particulares de erros no escopo de um quadro. Para tal foram obtidos resultados comparativos das métricas A e B do método proposto.

Para a simulação de erros em um quadro da seqüência de vídeo em questão, os resultados médios para cada valor de distância euclidiana foram extraídos da seguinte forma:

1. Simula-se erro em cada macrobloco (ou linha) do quadro. Cada simulação será denominada "realização do erro";

2. Para cada realização do erro no quadro, calcula-se o MSE (*mean square error*) entre os valores de luminância dos *pixels* de cada quadro da seqüência reconstruída e os *pixels* no quadro correspondente da seqüência original;

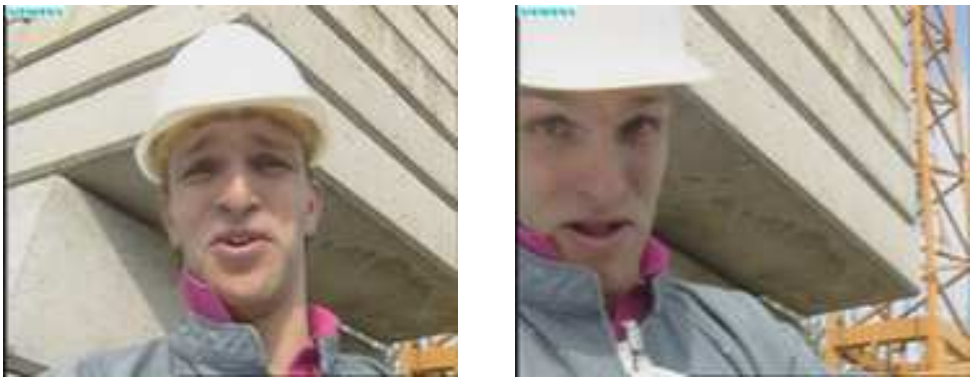
3. Calcula-se a média aritmética dos MSEs das realizações de erros para cada quadro da seqüência;

4. Calcula-se a PSNR sobre a média dos MSEs de cada quadro, segundo modificações da Equação 4.1.

Para essas simulações foram escolhidos dois quadros representativos da seqüência "foreman", por apresentarem as seguintes características: a predominância de macroblocos codificados por predição *inter*, e por pertencerem a trechos do vídeo com bastante movimento ou mudanças de cena. Os quadros escolhidos para a avaliação dos métodos de ocultamento são apresentados na Figura 4.8, e suas características são descritas a seguir:

- Quadro 62: Bastante movimento de translação em região rica em detalhes (face). Todos os macroblocos codificados originalmente por predição *inter*;
- Quadro 178: Mudança de cena com bastante movimento. Predominância de macroblocos codificados usando predição *inter* (8 macroblocos *intra* em um total de 99 macroblocos do quadro).

As duas configurações de erros em macrobloco individual e em linha foram simuladas para se observar o comportamento médio dos métodos propostos no escopo de um quadro.



(a) Quadro 62 (b) Quadro 178

Figura 4.8: Imagens da seqüência "foreman" usadas na simulação de erros.

Média em um quadro - erro em macrobloco individual As Figuras 4.9 e 4.10 apresentam os resultados das simulações de médias de erros em macroblocos individuais para os quadros 62 e 178 da seqüência "foreman", para distâncias euclidianas de 0, 4 e 16 *pixels*. Para todos esses casos, a métrica B do método proposto

apresentou desempenho melhor que o da métrica A. O limiar de distância euclidiana do agrupamento em conglomerados de vetores vizinhos ao macrobloco corrompido também resultou em diferenças no desempenho das métricas.

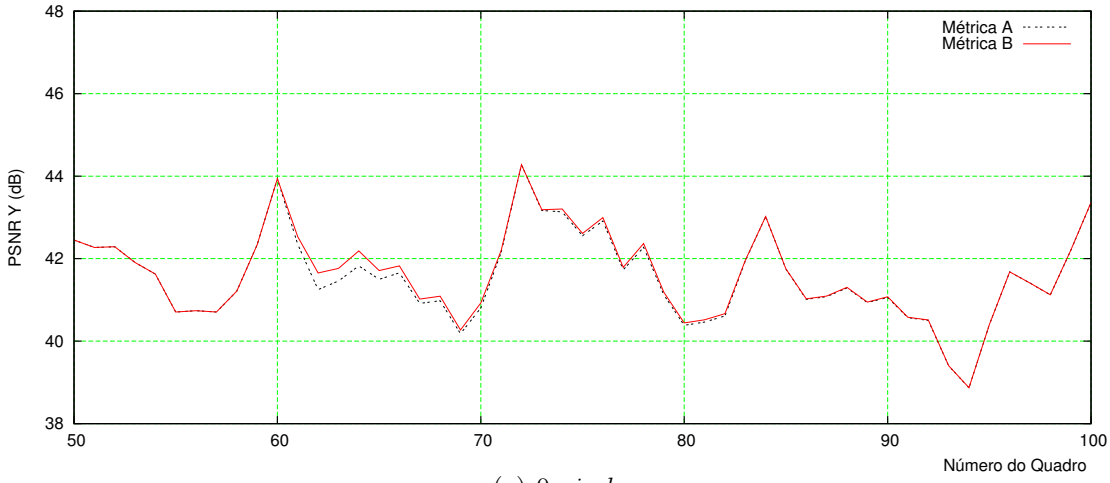
Média em um quadro - erro em linha As Figuras 4.11 e 4.12 apresentam os resultados das simulações de médias de erros em linhas para os quadros 62 e 178 da seqüência "foreman", para limiares de distância euclidiana de 0, 4 e 16 *pixels*. Assim como na configuração de erros em macroblocos individuais, a métrica B do método proposto mais uma vez apresentou desempenho melhor que o da métrica A.

Dos experimentos que obtêm o desempenho médio de erros no escopo de um quadro, foram extraídas as seguintes observações:

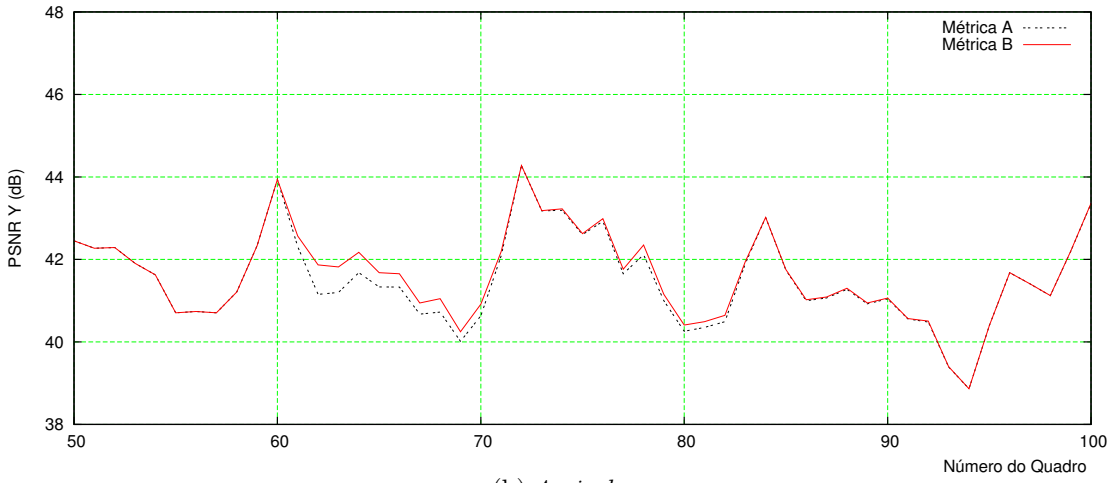
- Os resultados indicam que a métrica B apresenta desempenho melhor que o da métrica A, em termos da avaliação por PSNR. Esse resultado pode estar relacionado ao fato da métrica B levar em consideração critérios de suavidade espacial na escolha do macrobloco substituto;

- Os efeitos de propagação dos erros de um quadro corrompido para os quadros subseqüentes é perceptível. O número de quadros afetados pelos erros muda em função tanto do limiar de distância euclidiana do agrupamento de vetores, quanto da métrica utilizada;

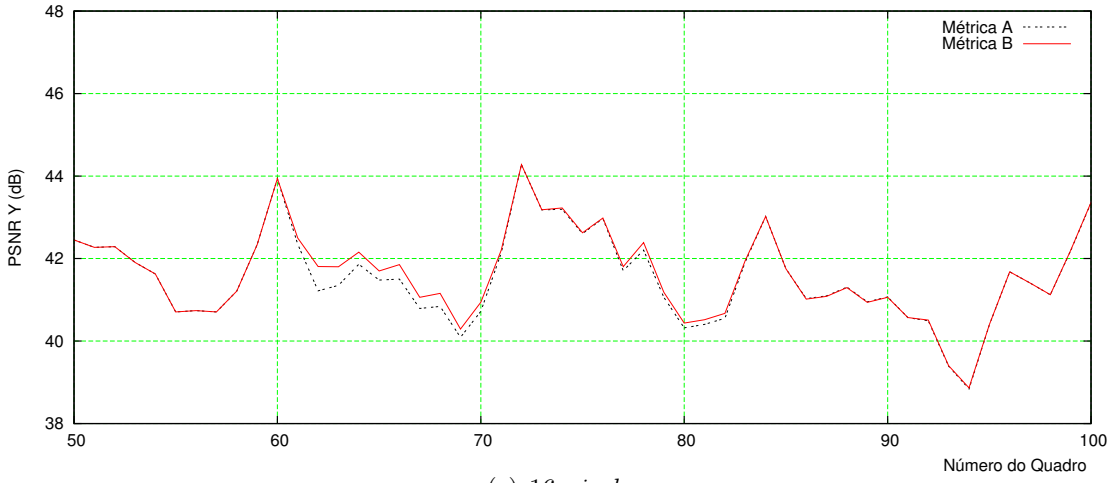
- Em média, os resultados das duas métricas do método proposto apresentaram variações pequenas para diferentes limiares de distância euclidiana do agrupamento de vetores. Contudo, com esses resultados ainda não foi possível extrair maiores conclusões sobre os efeitos da distância euclidiana, tornando necessário ampliar o escopo das simulações.



(a) 0 *pixels*

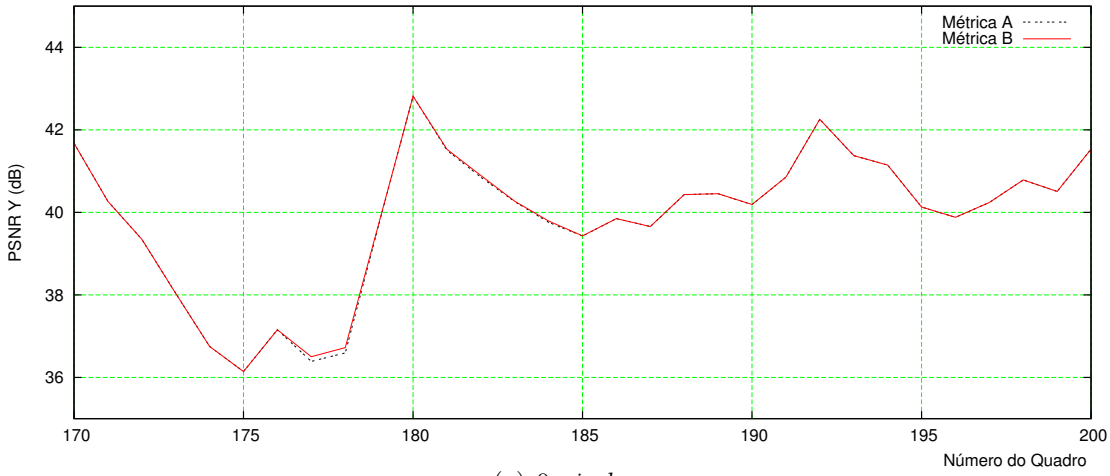


(b) 4 *pixels*

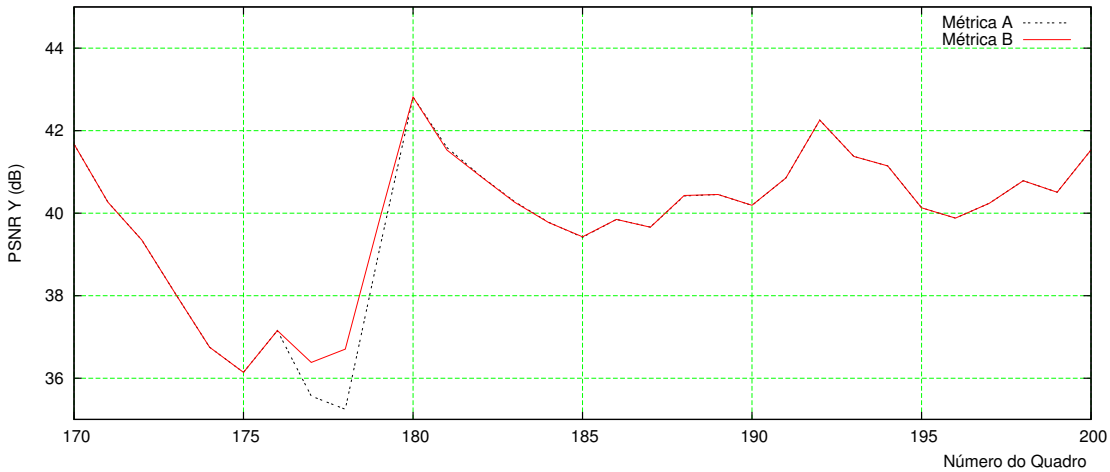


(c) 16 *pixels*

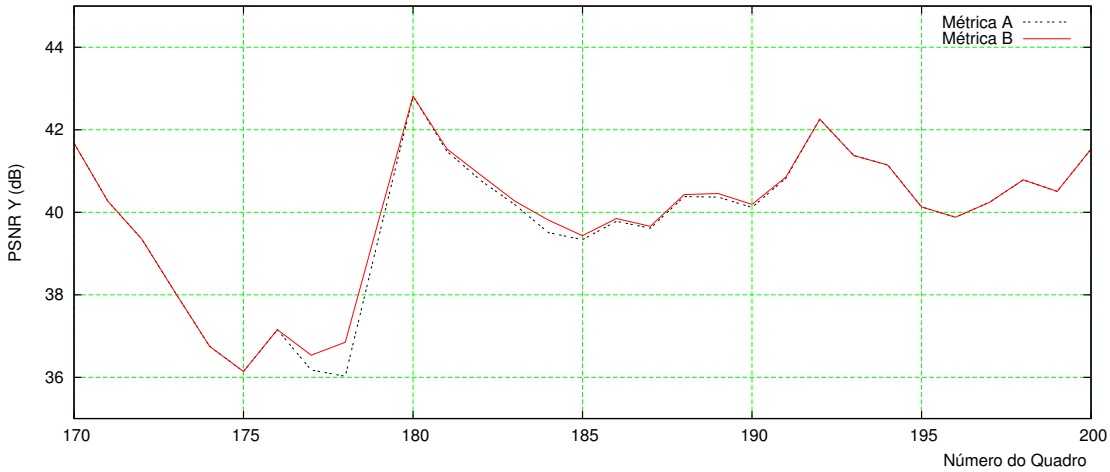
Figura 4.9: **Média no quadro**: evolução da PSNR para métricas A e B do método proposto. Configuração **erro em macrobloco individual** para o **quadro 62**. Seqüência "foreman" com taxa de 384kbps, para diferentes limiares de distância euclidiana.



(a) 0 *pixels*

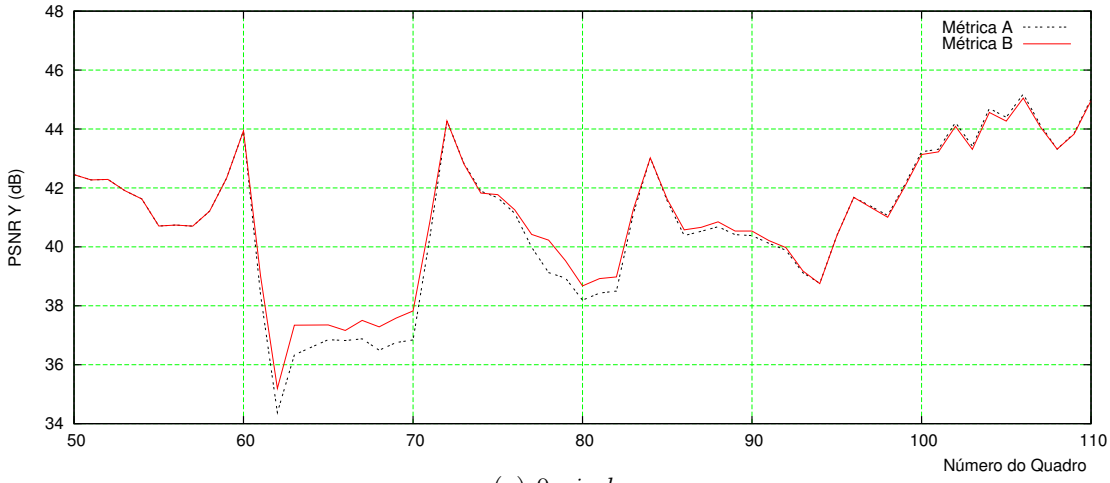


(b) 4 *pixels*

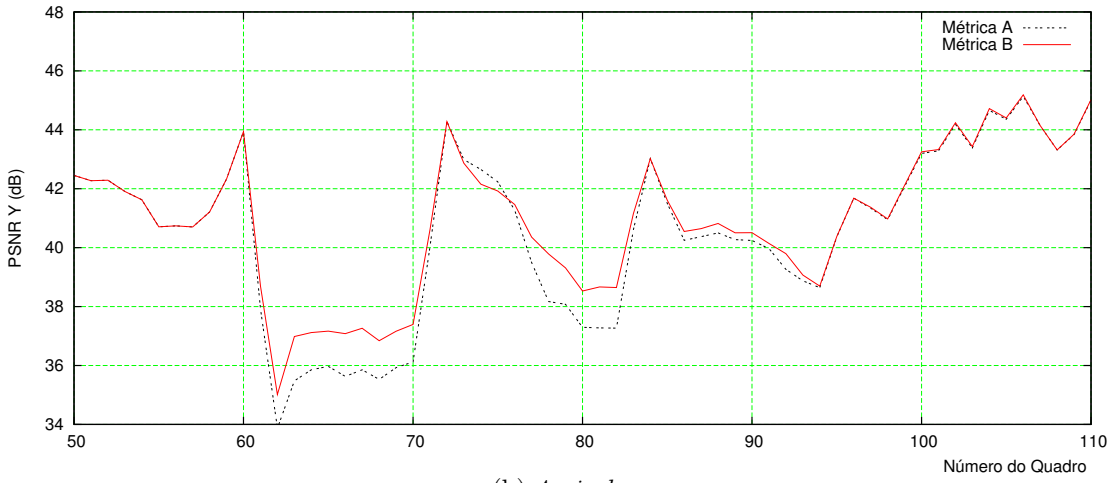


(c) 16 *pixels*

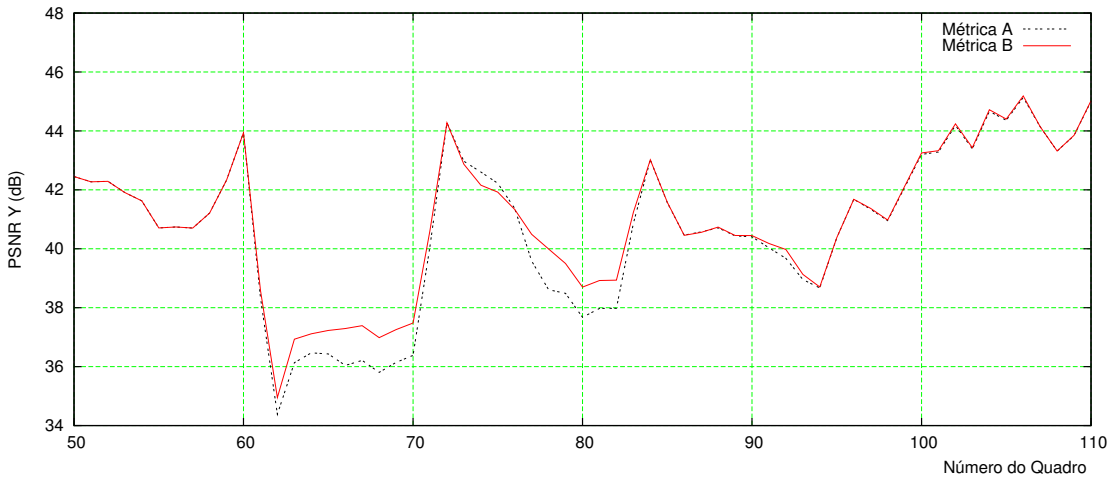
Figura 4.10: **Média no quadro**: evolução da PSNR para métricas A e B do método proposto. Configuração **erro em macrobloco individual** para o **quadro 178**. Seqüência "foreman" com taxa de 384kbps, para diferentes limiares de distância euclidiana.



(a) 0 *pixels*

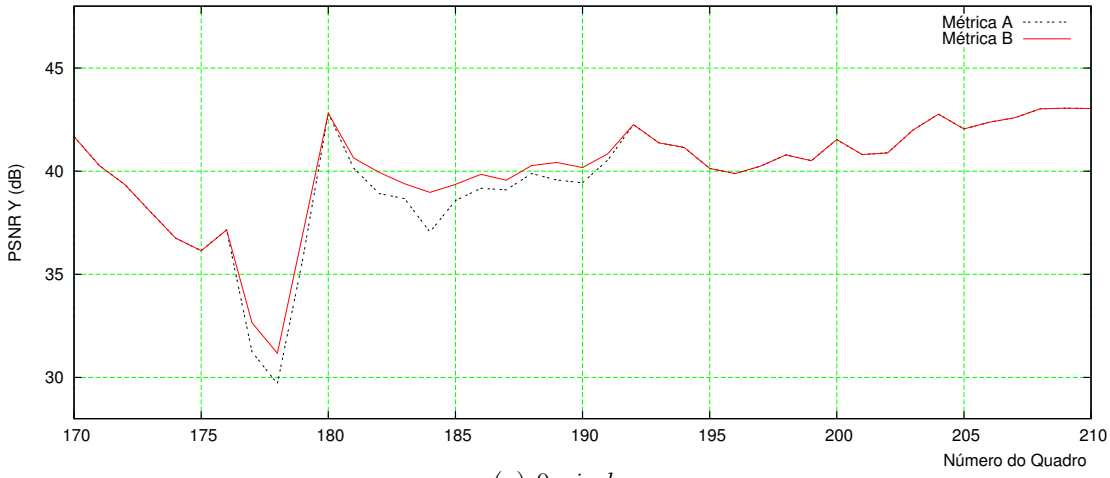


(b) 4 *pixels*

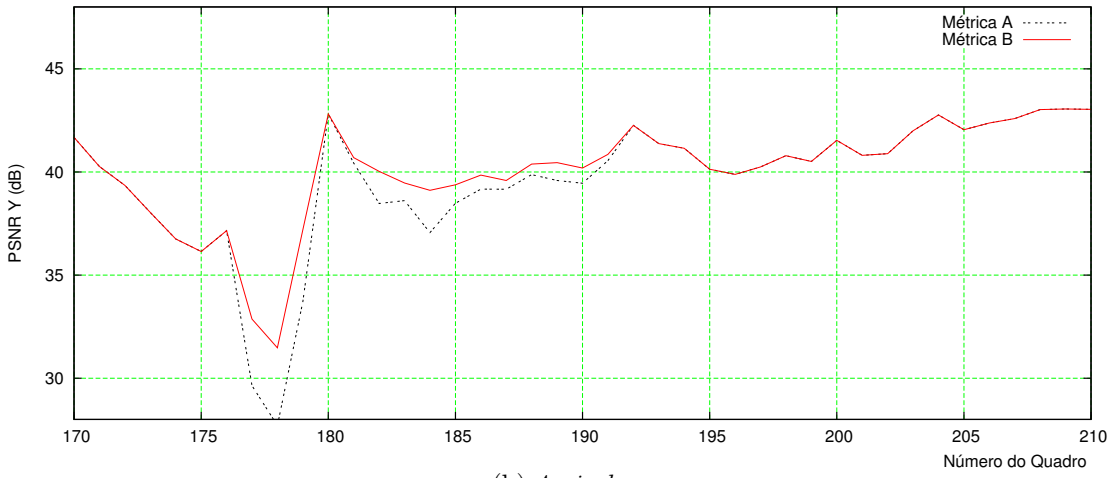


(c) 16 *pixels*

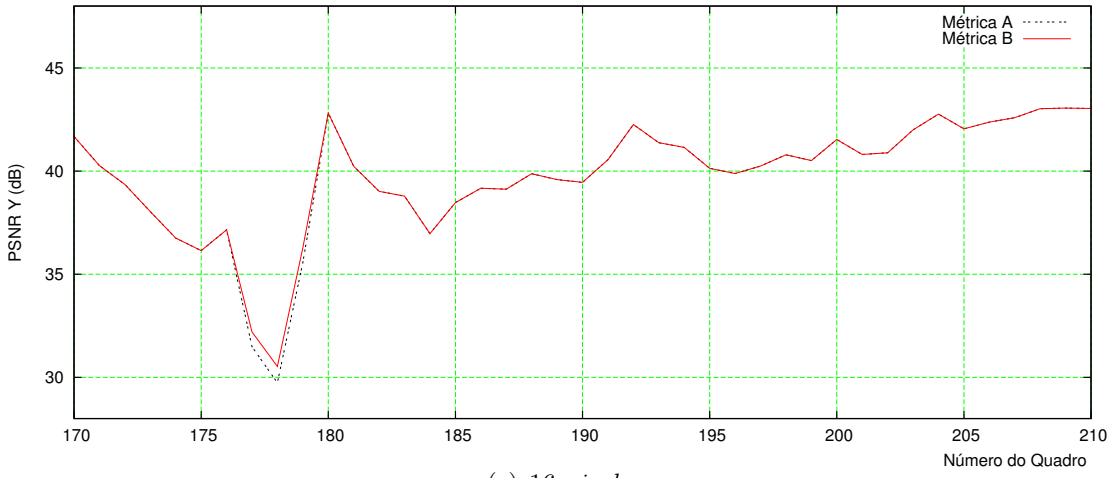
Figura 4.11: **Média no quadro**: evolução da PSNR para métricas A e B do método proposto. Configuração **erro em linha** para o **quadro 62**. Seqüência "foreman" com taxa de 384kbps, para diferentes limiares de distância euclidiana.



(a) 0 *pixels*



(b) 4 *pixels*



(c) 16 *pixels*

Figura 4.12: **Média no quadro**: evolução da PSNR para métricas A e B do método proposto. Configuração **erro em linha** para o **quadro 178**. Seqüência "foreman" com taxa de 384kbps, para diferentes limiares de distância euclidiana.

Os resultados das duas métricas do método proposto indicam que o uso do critério de distorção de bordas tende a gerar melhores resultados em termos de PSNR. Contudo, o critério de distorção de bordas é propenso a apresentar alguns problemas. Como esse critério se baseia somente nas diferenças de luminância dos *pixels* localizados nas bordas de macroblocos vizinhos, eventuais descontinuidades presentes nas regiões centrais do macrobloco candidato podem resultar na degradação do resultado, tanto em termos da PSNR quanto da percepção visual. No caso do *software* de referência, a escolha do vetor de movimento para substituir o macrobloco corrompido se baseia somente na distorção de borda. Ao usar o teste SAD de partições vizinhas ao macrobloco corrompido, o método proposto buscou agregar um critério de correlação espacial mais abrangente.

A partir das simulações no escopo dos quadro 62 e 178 da seqüência "foreman", optou-se por analisar com mais detalhes os resultados do método proposto para a métrica B, que apresentou melhor desempenho. Para tal, foram usadas simulações com escopo mais abrangente, os testes de **média na seqüência**, comparando resultados da métrica B e do *software* de referência.

4.3.5 Cenário 2: Média na Seqüência

As simulações de **média na seqüência** tiveram como propósito formar um conjunto mais amplo de resultados para se extrair maiores conclusões sobre o desempenho médio do método proposto, com relação ao do *software* de referência. No escopo da seqüência, decidiu-se por realizar comparações do *software* de referência apenas com a métrica B do método proposto, por essa ter indicado melhores resultados nas simulações no escopo de um quadro.

Para a simulação de erros no escopo da seqüência de vídeo, os resultados médios do método proposto, para cada limiar de distância euclidiana, foram extraídos da seguinte forma:

1. Simula-se erro em cada macrobloco (ou linha) para todos os *slices* do tipo P da seqüência em questão. Cada simulação será denominada "realização do erro";

2. Para cada realização do erro, calcula-se o MSE (*mean square error*) entre os valores de luminância dos *pixels* de quadros da seqüência reconstruída e os

pixels no quadro correspondente da seqüência original. O conjunto de quadros utilizados nessa média foi definido a partir de uma "janela" de quadros considerados mais propensos a serem afetados pelo erro. Essa "janela" é composta pelos 5 quadros anteriores e os 15 posteriores ao quadro em que ocorreu o erro, na ordem temporal de visualização do vídeo;

3. Calcula-se a média aritmética dos MSEs das realizações de erros;

4. Calcula-se a PSNR sobre a média dos MSEs das realizações de erros, segundo modificações da Equação 4.1.

As simulações no escopo da seqüência permitiram a obtenção de resultados médios a partir de um número maior de experimentos. Para diferentes limiares de distância euclidiana, o resultado nas seqüências de vídeo em questão foi avaliado pela PSNR do comportamento médio nos quadros mais propensos a sofrerem efeitos de propagação, para cada realização de erro. Assim como nas simulações no escopo de um quadro, as configurações de erros utilizadas foram: **erro em macrobloco individual** e **erro em linha**. Para essas simulações foram escolhidas as seqüências "foreman" e "silent".

Média na Seqüência - erro em macrobloco individual A Figura 4.13 apresenta os resultados do comportamento médio no escopo da seqüência para a métrica B do método proposto, e configuração de erros em macroblocos. As seqüências "foreman" e "silent" foram testadas para um conjunto de valores de distância euclidiana de 0 a 32 *pixels*, com passo de 4 *pixels*. Percebeu-se, então, que os melhores resultados de PSNR da métrica B se concentravam na faixa de valores de distância euclidiana menores que 8 *pixels*. Para analisar esse resultado com mais detalhes, novas simulações foram geradas para o intervalo de 0 a 8 *pixels*.

As simulações indicaram que a PSNR não é uma função monotônica da distância euclidiana. Para cada seqüência deve existir um valor de distância euclidiana que resulte na máxima PSNR, ou seja, a distância euclidiana ótima. Contudo, nessas simulações, mesmo o resultado para a distância euclidiana ótima da métrica B não chegou a ser melhor que o do *software* de referência.

A tabela 4.1 apresenta o resultado comparativo do *software* de referência com relação à métrica B do método proposto. O valor de PSNR obtido para a distância

euclidiana ótima da métrica B é apresentado juntamente com o resultado do *software* de referência. Os valores de distância euclidiana ótima da métrica B, medidos para as seqüências "foreman" e "silent", é de 3 e 4 *pixels*, respectivamente, como pode ser observado na Figura 4.13.

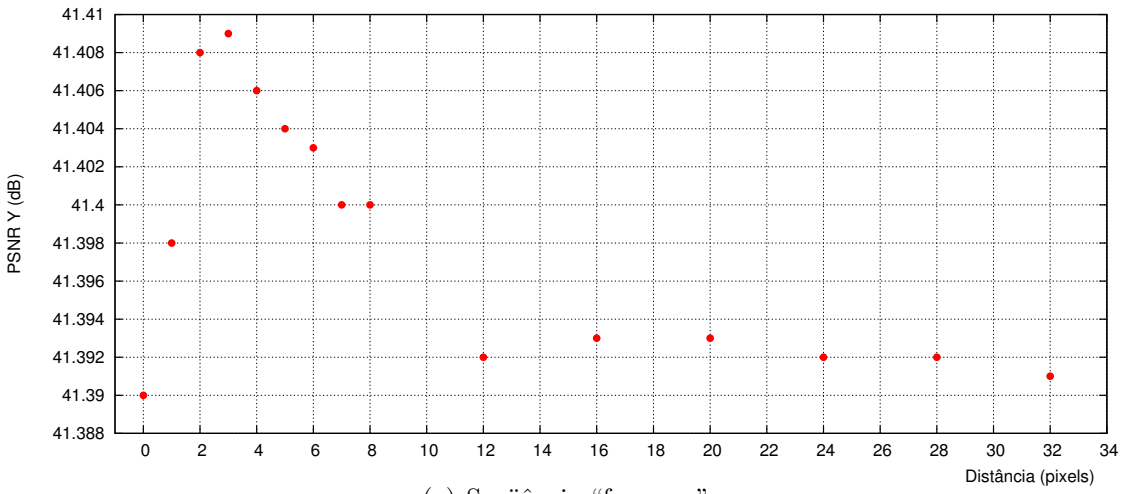
Tabela 4.1: Comparação do melhor desempenho da métrica B com o *software* de referência, para a configuração: **média na seqüência** e **erro em macroblocos individuais**

Seqüência	PSNR (dB)	
	Métrica B	*Software* de referência
"foreman"	41,409	41,413
"silent"	45,467	45,473

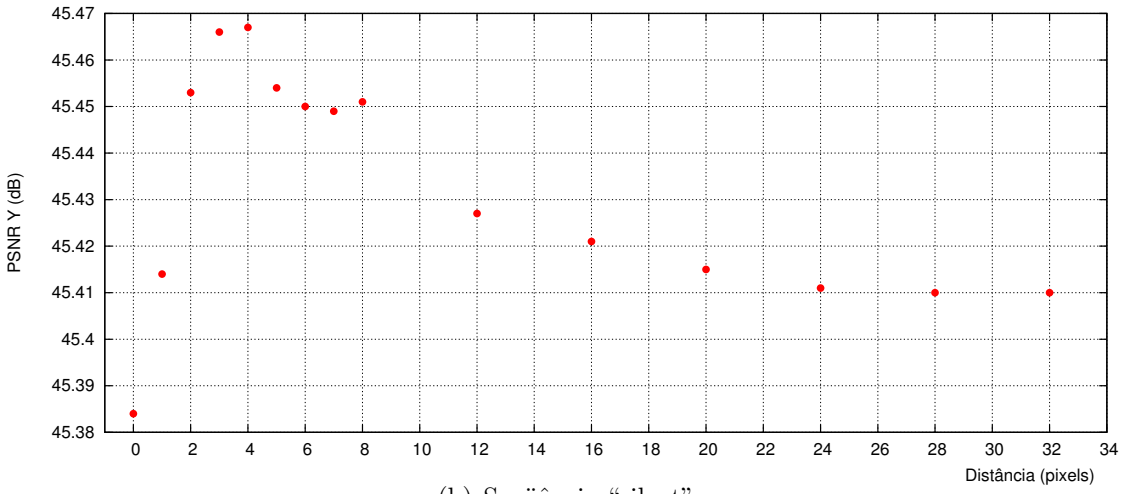
Média na Seqüência - erro em linha Os resultados médios da métrica B do método proposto são apresentados na Figura 4.14, para a configuração de erro em linhas no escopo da seqüência. As seqüências "foreman" e "silent" foram testadas para um conjunto de valores de distância euclidiana de 0 a 32 *pixels*, com passo de 2 *pixels*.

Com os resultados da Figura 4.14, percebeu-se então, que os melhores resultados de PSNR da métrica B se concentravam nos maiores valores de distância euclidiana. Com o aumento da distância euclidiana no agrupamento de vetores vizinhos ao macrobloco corrompido, os resultados da PSNR convergiram para um valor constante bastante próximo do valor máximo.

A tabela 4.2 apresenta os resultados comparativos do *software* de referência com relação à métrica B do método proposto. O melhor valor de PSNR obtido das simulações com diferentes valores de distância euclidiana para a métrica B não chegou a ser melhor que o método do *software* de referência, apesar de bem próximo.

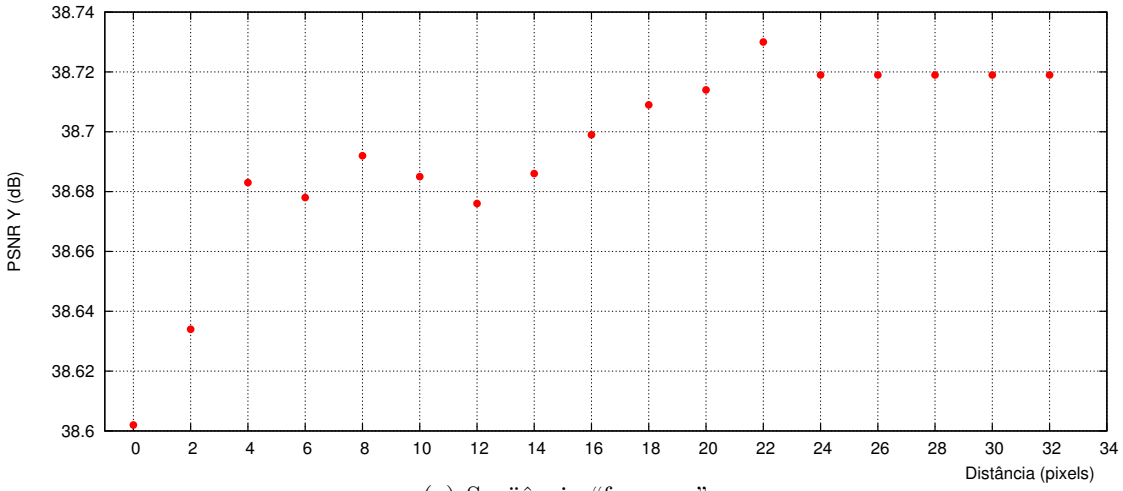


(a) Seqüência “foreman”

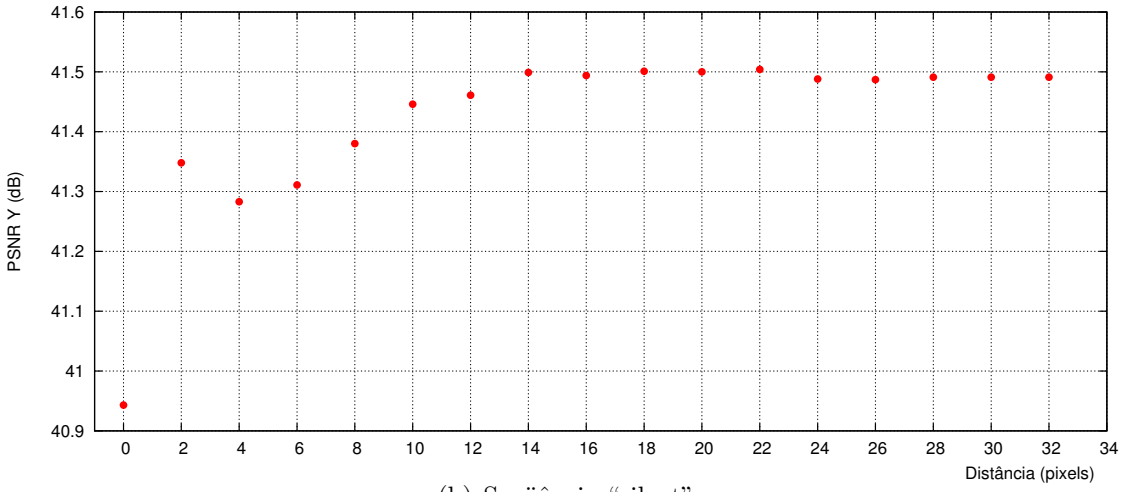


(b) Seqüência “silent”

Figura 4.13: **Média na seqüência**: PSNR da métrica B do método proposto para diferentes distâncias euclidianas. Configuração **erro em macroblocos individuais**, para as seqüências “foreman” e “silent” com taxa de 384kbps.



(a) Seqüência “foreman”



(b) Seqüência “silent”

Figura 4.14: **Média na seqüência**: PSNR da métrica B do método proposto para diferentes distâncias euclidianas. Configuração **erro em linha**, para as seqüências “foreman” e “silent” com taxa de 384kbps.

Tabela 4.2: Comparação do melhor desempenho da métrica B com o *software* de referência, para a configuração: **média na seqüência** e **erro em linhas**

Seqüência	PSNR (dB)	
	Métrica B	*Software* de referência
“foreman”	38,730	38,839
“silent”	41,504	41,832

4.3.6 Discussão

As simulações realizadas foram importantes para se obter resultados comparativos, tanto das métricas do método proposto entre si, quanto do método proposto com relação ao *software* de referência. A partir dessas simulações foi possível observar que:

- A métrica B apresenta indícios de alcançar melhor desempenho em termos de PSNR, em comparação com a métrica A;
- Os resultados comparativos do *software* de referência com a métrica B do método proposto apresentaram pequenas diferenças. Contudo, por terem sido gerados a partir de um grande número de testes, mesmo pequenas diferenças de PSNR são consideradas representativas de uma diferença de desempenho entre o método proposto e o do *software* de referência;
- O desempenho do método proposto varia consideravelmente em função da distância euclidiana. Esse fato indica que explorar a idéia do agrupamento de vetores em conglomerados pode levar a ganhos significativos de desempenho;

O desempenho inferior da métrica B do método proposto, com relação ao *software* de referência, motivou a formulação de algumas hipóteses. Buscou-se identificar características das metodologias de ocultamento de erros do método proposto que pudessem estar causando resultados piores com relação ao método do *software* de referência. Algumas hipóteses de características responsáveis pelo desempenho pior do método proposto são:

- **Compor conglomerados a partir de sub-blocos pertencentes aos macroblocos adjacentes de 16×16 *pixels*, ao invés de somente os blocos 8×8 adjacentes.** A ampliação do conjunto de vetores das vizinhanças do macrobloco corrompido usados nos testes de compensação de movimento poderia estar influenciando os resultados. No método proposto, tanto as partições adjacentes ao macrobloco corrompido, quanto as mais distantes, são avaliadas sem se atribuir qualquer prioridade para vetores de movimento de partições mais próximas ao macrobloco corrompido. Contudo, quanto mais afastada é uma partição de *pixels* do macrobloco corrompido, menos correlacionados tendem a ser os seus vetores de movimento. Assim, ao se permitir que os vetores

de teste sejam obtidos por partições vizinhas não adjacentes ao macrobloco perdido, pode-se escolher o vetor de movimento para substituir o macrobloco corrompido que, na prática, não traduza um resultado de PSNR condizente com a qualidade da recuperação do macrobloco corrompido;

- **Heurística de agrupamento de vetores em conglomerados.** O critério de SAD usado para determinar o vetor representativo de cada conglomerado requer que a própria organização dos vetores de movimento em conglomerados tenha sido eficiente. O melhor casamento de um conglomerado é "pesquisado" na imagem de referência, mantendo constantes as distâncias relativas entre as partições que compõem o conglomerado. Caso a organização dos vetores vizinhos em conglomerados não tenha sido eficiente, os resultados de SAD podem ser muito prejudicados. Uma heurística de classifição mais eficiente dos vetores de movimento em conglomerados pode ser desenvolvida a partir de algum parâmetro de confiabilidade dos próprios vetores, tal como a energia dos resíduos dos coeficientes da DCT, ou também usando diferentes técnicas de "clusterização".

- **Heurística de escolha do vetor representativo de cada conglomerado.** Como a métrica de escolha do vetor representativo de cada conglomerado se baseia nos *pixels* das partições dos conglomerado obtidos por compensação de movimento na imagem de referência, uma escolha sub-ótima para esse vetor de movimento pode resultar em impactos consideráveis na PSNR da imagem regenerada. Especialmente para imagens contendo arestas, a escolha de um vetor de movimento com coordenadas ligeiramente diferentes do vetor originalmente usado na predição do macrobloco perdido pode gerar grandes diferenças de MSE entre a imagem original e a imagem afetada pelos erros. Isso poderia levar a PSNR relativa ao método proposto de ocultamento a ser menor que a do *software* de referência.

Algoritmo 4.1 Método Proposto de Ocultamento de Erros.

1. Verificar os macroblocos vizinhos disponíveis;
2. Listar os vetores de movimento de todas as partições contidas nos macroblocos vizinhos;
3. Agrupar partições em conglomerados, segundo a proximidade euclidiana de seus vetores de movimento;
4. Para cada conglomerado:
   a) Definir a região de busca a partir da organização dos vetores de movimento em um fecho convexo;
   b) Para cada vetor pertencente ao fecho convexo, calcular o valor de uma função custo relativa ao uso desse vetor para obter o macrobloco corrompido por compensação de movimento. A função custo utiliza uma das seguintes métricas:
      i) Casamento *pixel* a *pixel* das partições do conglomerado em comparação com a imagem de referência (**Métrica A**);
      ii) A métrica (i) para se obter o vetor representativo de cada conglomerado, associada a uma métrica de distorção de bordas do macrobloco candidato a substituir o macrobloco corrompido (**Métrica B**).
5. Escolher o vetor que representa a melhor estimativa do macrobloco corrompido, segundo o critério escolhido;
6. Substituir o macrobloco corrompido;
7. Sinalizar o macrobloco corrompido como regenerado;

Algoritmo 4.2 *Graham Scan* para o cálculo de um fecho convexo bi-dimensional.

1. Busca do ponto extremo do conjunto. Esse ponto será denominado pivô, e pertencerá ao fecho convexo, garantidamente. O ponto escolhido como pivô apresenta a maior coordenada y (vertical). Caso haja mais de um ponto com essa coordenada, o ponto com menor coordenada x é preferencialmente selecionado como pivô;

2. Todos os pontos são então ordenados por ângulos crescentes com respeito ao pivô;

3. O fecho convexo é então construído testando-se cada um dos pontos, na ordem crescente de ângulo. Para cada ponto testado, é adicionada uma aresta ao fecho convexo para cada desvio à esquerda com respeito à aresta anterior. Caso haja um desvio à direita, o algoritmo elimina o ponto em questão, e retoma a iteração a partir do ponto válido anterior.

Capítulo 5

Conclusões

Esta dissertação teve como foco principal as técnicas de ocultamento de erros na fase de decodificação de um sinal digital de vídeo comprimido usando o padrão H.264/AVC. A principal aplicação das técnicas reside na transmissão *wireless* de vídeo.

O Capítulo 1 apresentou as motivações do uso de técnicas de compressão de vídeo, e os requisitos necessários para garantir a transmissão robusta do mesmo. O Capítulo 2 apresentou os conceitos teóricos que fundamentam a questão da robustez a erros na transmissão de vídeo, com ênfase nas técnicas usadas para minimizar o impacto de erros de transmissão na qualidade de sinais de vídeo codificados. Também foram abordados tópicos relacionados à compressão de vídeo, em especial sobre o padrão H.264. O Capítulo 3 aborda com maiores detalhes as técnicas de ocultamento de erros, incluindo a revisão bibliográfica de vários métodos atuais, seguido da descrição dos métodos de ocultamento de erros implementados no *software* de referência do padrão H.264. No Capítulo 4, é proposto um método alternativo de ocultamento de erros na decodificação. O método proposto busca estimar a coerência de movimento das partições vizinhas ao macrobloco corrompido. Ao final foram realizados estudos do desempenho em termos de PSNR, tanto das métricas propostas, quanto dos métodos do *software* de referência. Com base em análises de pontos favoráveis e desfavoráveis dos métodos estudados, são discutidas algumas propostas de modificações nas métricas usadas no ocultamento de erros.

Algumas das contribuições desta dissertação são:

- Revisão bibliográfica de métodos atuais de ocultamento de erros em codificação

de vídeo;

- Estudo do funcionamento do *software* de referência do padrão H.264, em um conjunto de cenários de testes de erros;
- Proposta, estudo e implementação de novos métodos de ocultamento de erros;
- Estudo comparativo do desempenho das métricas dos métodos propostos em relação ao *software* de referência;
- Identificação das características dos métodos existentes, sugerindo possíveis modificações.

A fim de obter melhores resultados para o método proposto, tanto em termos de PSNR quanto em evitar a propagação de erros ao longo da seqüência de vídeo, foram formuladas as seguintes propostas de trabalhos futuros:

- Análise da complexidade computacional do método proposto;
- Estudos de critérios mais sofisticados na classificação em conglomerados dos vetores de movimento vizinhos ao macrobloco corrompido, como por exemplo, testes de diferentes métricas de ordenação dos vetores, ou algoritmos de "clusterização";
- Estudos mais criteriosos dos impactos no desempenho do método proposto ao usar conglomerados formados por sub-blocos pertencentes aos macroblocos adjacentes 16×16 *pixels*, ao invés de somente os blocos 8×8 adjacentes como ocore no *software* de referência;
- Análise estatística do impacto de se considerar informação de movimento associada a partições menores que 8×8 *pixels*;
- Estudos da implementação de critérios de detecção e preservação de arestas para o método proposto;
- Implementação de mudancas nas métricas de ocultamento de erros. Adaptar o método proposto, analisando o resultado de cada modificação com relação ao *software* de referência, de forma a identificar os pontos fortes e fracos;

- Elaboração de cenários adicionais de simulações, e realização de testes mais detalhados, explorando diferentes modelos de perdas de macroblocos adequados aos padrões de erro de canal das aplicações a que se pretende utilizar o método;

Referências Bibliográficas

[1] SAYOOD, K., *Introduction to Data Compression*. 2 ed. San Francisco, Morgan Kaufmann, 2000.

[2] JAIN, A. K., *Fundamentals of Digital Image Processing*. New Jersey, Prentice Hall, 1989.

[3] HASKELL, B. G., HOWARD, P. G., LECHUN, Y. A., *et al.*, "Image and Video Coding - Emerging Standards and Beyond", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 8, n. 7, pp. 814–837, November 1998.

[4] MITCHELL, J. L., PENNEBAKER, W. B., FOGG, C. E., *et al.*, *MPEG Video Compression Standard*. New York, Chapman & Hall, 1997.

[5] WIEGAND, T., SULLIVAN, G. J., BJONTEGAARD, G., *et al.*, "Overview of the H.264/AVC Video Coding Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, pp. 560–576, July 2003.

[6] SIKORA, T., "MPEG Digital Video-Coding Standards", *IEEE Signal Processing Magazine*, v. 14, n. 5, pp. 82–100, September 1997.

[7] WANG, Y., WENGER, S., WEN, J., *et al.*, "Error resilient video coding techniques", *IEEE Signal Processing Magazine*, v. 17, pp. 61–82, July 2000.

[8] WANG, Y., ZHU, Q.-F., "Error control and concealment for video communication: A Review", *Proceedings of the IEEE*, v. 86, pp. 974–997, 1998.

[9] SULLIVAN, G. J., WIEGAND, T., "Video Compression - From Concepts to the H.264/AVC Standard", *Proceedings of the IEEE*, v. 93, n. 1, pp. 18–31, January 2005.

[10] "H.264/AVC Reference Software", http://iphome.hhi.de/suehring/tml/download/, Acessado em fevereiro de 2006.

[11] WANG, Y.-K., HANNUKSELA, M. M., VARSA, V., *et al.*, "The Error Concealment Feature in the H.26L Test Model". In: *Proc. IEEE ICIP*, v. II, pp. 729–732, Rochester, USA, 2002.

[12] PROAKIS, J. G., *Digital Communications*. 3 ed. San Francisco, McGraw-Hill, 1995.

[13] HAYKIN, S., *Communication Systems*. 4 ed. USA, John Wiley & Sons, 2001.

[14] BELL, T. C., CLEARY, J. G., WITTEN, I. H., *Text Compression*. New Jersey, Prentice Hall, 1990.

[15] COVER, T. M., THOMAS, J. A., *Elements of Information Theory*. USA, John Wiley & Sons, 1991.

[16] "MPEG License", http://www.mpegla.com/, Acessado em fevereiro de 2006.

[17] "Draft of Version 4 of H.264/AVC (ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 part 10) Advanced Video Coding)", Joint Video Team (JVT) of ISO/IEC MPEG ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6), Janeiro 2005.

[18] RICHARDSON, I. E. G., *H.264 and MPEG-4 Video Compression - Video Coding for Next-Generation Multimedia*. Chichester, John Wiley & Sons, 2003.

[19] SULLIVAN, G. J., TOPIWALA, P., LUTHRA, A., "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions". In: *Proceedings of SPIE Conference on Applications of Digital Image Processing*, pp. 454–474, August 2004.

[20] STOCKHAMMER, T., HANNUKSELA, M. M., WIEGAND, T., "H.264/AVC in Wireless Environments", *IEEE Transactions on Circuits and Systems for Videl Technology*, v. 13, n. 7, pp. 657–673, July 2003.

[21] KACHOUH, Z. A., BELLANGER, M. G., "Efficient Restoration Technique for Missing Blocks in Images", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, pp. 1182–1186, 2003.

[22] KACHOUH, Z. A., BELLANGER, M. G., "Fast-DCT-Based Spatial Domain Interpolation of Blocks in Images", *IEEE Transactions on Image Processing*, v. 9, pp. 729–732, 2000.

[23] ZHENG, J., CHAU, L.-P., "An Efficient Spatial Domain Error Concealment Method for H.264". In: *Proc. IEEE ICICS-PCM*, pp. 26–30, Singapore, 2003.

[24] COLLONESE, S., PANCI, G., SANSONE, C., *et al.*, "Hierarchical Image Analysis using Radon Transform: An Application to Error Concealment". In: *Proc. IEEE ICIP*, v. III, pp. 884–887, 2005.

[25] CHEN, J., LIU, J., WANG, X., *et al.*, "Modified Edge-oriented Spatial Interpolation for Consecutive Blocks Error Concealment". In: *Proc. IEEE ICIP*, v. III, pp. 904–907, 2005.

[26] XU, Y., ZHOU, Y., "H.264 Video Communication Based Refined Error Concealment Schemes", *IEEE Trans. Consumer Electronics*, v. 50, pp. 1135–1141, 2004.

[27] PENEDO, S. R., SEARA, R., "Contribuição para a Correção de Erro em Sinais de Vídeo Através da Propriedade de Regularidade das Wavelets". In: *Proc. SBT'05*, Campinas, SP, 2005.

[28] CHEN, M.-J., CHEN, L.-G., WENG, R.-M., "Error Concealment of Lost Motion Vectors with Overlapped Motion Compensation", *IEEE Tans. on Circuits and Systems for Video Technology*, v. 7, pp. 560–563, June 1997.

[29] KANG, L.-W., LEOU, J.-J., "A hybrid Error Concealment Scheme for MPEG-2 Video Transmission Based on Best Neighborhood Matching Algorithm". In: *Proc. IEEE ICME*, pp. 1355–1358, Sorrento, Italy, 2004.

[30] WANG, Z., YU, Y., ZHANG, D., "Best Neighborhood Matching: An Information Loss Restoration Technique for Block-Based Image Coding Systems", *IEEE Transactions on Image Processing*, v. 7, pp. 1056–1061, July 1998.

[31] TUNG, Y.-L., SHU, H.-C., LEOU, J.-J., "An Error Detection and Concealment Scheme for H.264 Video Transmission". In: *Proc. IEEE ICME*, pp. 1735–1738, 2004.

[32] SU, L., ZHANG, Y., GAO, W., *et al.*, "Improved Error Concealment Algorithms Based on H.264/AVC Non-normative Decoder". In: *Proc. IEEE ICME*, pp. 1671–1674, Sorrento, Italy, 2004.

[33] TSEKERIDOU, S., PITAS, I., "MPEG-2 Error Concealment Based on Block-Matching Principles", *IEEE Trans. Circuits and Systems for Video Technology*, v. 10, n. 4, pp. 646–658, June 2000.

[34] KIM, D., YANG, S., JEONG, J., "A New temporal Error Concealment Method for H.264 using Adaptive Block Sizes". In: *Proc. IEEE ICIP*, v. III, pp. 928–931, 2005.

[35] ZHENG, J., CHAU, L.-P., "A Temporal Error Concealment Algorithm for H.264 using Lagrange Interpolation". In: *Proc. IEEE ISCAS*, pp. II–133–II–136, Vancouver, Canada, 2004.

[36] ZHENG, J., CHAU, L.-P., "A Temporal Error Concealment Algorithm for H.264 Based on Plane Estimation". In: *Proc. IEEE ICICS-PCM*, pp. 253–257, Singapore, 2003.

[37] JUNG, B., JEON, B., KIM, M.-D., *et al.*, "Selective Temporal Error Concealment Algorithm for H.264/AVC". In: *Proc. IEEE ICME*, pp. 411–414, Sorrento, Italy, 2004.

[38] ATZORI, L., GINESU, G., RACCIS, A., "JPEG2000-Coded Image Error Concealment Exploiting Convex Set Projections", *IEEE Trans. Image Processing*, v. 14, n. 4, pp. 487–498, April 2005.

[39] ATZORI, L., BILGIN, A., MARCELLIN, M. W., "Error Concealment for Motion JPEG2000". v. I, pp. 781–784, 2005.

[40] FUKUHARA, T., KATOH, K., KIMURA, S., *et al.*, "Motion-JPEG2000 Standardization and Target Market". In: *Proc. IEEE ICIP*, pp. II–57–II–60, Vancouver, Canada, 2000.

[41] "Advanced video coding for generic audiovisual services (ITU-T Rec. H.264 ou ISO/IEC 14496-10 AVC)", Joint Video Team (JVT) do ITU-T e ISO/IEC JTC1, Maio, 2003.

[42] "*Software* Simulador de Erros Baseado na Proposta ITU-T VCEG Q15-I-16r1", http://ftp3.itu.ch/av-arch/video-site/9910_Red/, Acessado em fevereiro de 2006.

[43] BOYD, S., VANDENBERGHE, L., *Convex Optimization*. United Kingdom, Cambridge University Press, 2004.

[44] "Algoritmos Usados para Calcular Fechos Convexos", http://softsurfer.com/Archive/algorithm_0109/algorithm_0109.htm, Acessado em fevereiro de 2006.

[45] "Algoritmo Graham Scan para Calcular Fechos Convexos", http://www.cs.princeton.edu/∼ah/alg_anim/version1/GrahamScan.html, Acessado em fevereiro de 2006.

[46] "Projeto H-264 Brasil", http://h264brasil.ime.eb.br, Acessado em fevereiro de 2006.

Apêndice A

Implementação em *Software* do Método Proposto

O algoritmo de ocultamento de erros proposto é implementado em linguagem C, a partir de modificações do *software* de referência do decodificador (jm versão 9.6) [10, 11], e consiste em um aplicativo baseado em console. O ambiente de desenvolvimento utilizado é o mesmo do software de referência.

A aplicação é capaz de gerar diversos erros em macroblocos de uma seqüência de vídeo. Os macroblocos e quadros onde os erros devem ser inseridos são definidos pelo usuário nos parâmetros de entrada. O usuário também configura se as funções de ocultamento de erros devem ser utilizadas. Esta possibilidade permite a comparação dos resultados da decodificação de uma mesma seqüência contendo erros em determinados macroblocos, quando regenerada pelas funções de ocultamento de erros, e quando decodificada sem qualquer regeneração.

O programa principal do decodificador em questão é o arquivo *ldecod.exe*. Este programa requer a especificação de vários parâmetros de entrada para a execução da simulação. A configuração dos parâmetros de entrada pode ocorrer de duas formas distintas: usando um arquivo de configuração (ex. ldecod.exe <arquivo de configuração>), com o nome default decoder.cfg; ou especificando os parâmetros de entrada na própria linha de comando da execução (ex. ldecod.exe -<parâmetro> <valor especificado>).

As configurações dos parâmetros de entrada podem ser especificadas na linha de execução do programa, ou no arquivo de configuração. O parâmetro **-ne** indica

o número de pares quadro-macrobloco que receberão erros (valor maior que 0); o parâmetro **-fm** deve ter o formato "quadro:macrobloco", especificando a localização dos erros na seqüência; e **-ec** indica uso das funções de ocultamento de erros (ativado ou desativado com os valores 1 e 0, respectivamente); **-d** indica a distância euclidiana máxima para agrupar vetores em conglomerados; e **-m** indica o método de ocultamento de erros utilizado (1, 2 ou 3).

Os parâmetros de entrada são:

- Ocultamento de Erros (1: ativo; 0: inativo);
- Métrica de ocultamento de erros utilizada (descrição na seção 4.1):
  - 2: métrica A do método proposto;
  - 3: métrica B do método proposto;
- Imagens e macroblocos em que serão inseridos erros (Imagem:Macrobloco);
- Máxima distância euclidiana usada na classificação dos vetores de movimento em conglomerados.

A saída do programa retorna o valor da PSNR para cada imagem do vídeo decodificado, em comparação com a seqüência original.