Análise de Métodos de Partição em Blocos em Codificação de Vídeo Baseada em Objeto

Flávia Magalhães Freitas
DEET/PUC Minas
30.535-610 Belo Horizonte-MG
pesquisa@pucminas.br

Abraham Alcaim
CETUC/PUC Rio
22.453-900 Rio de Janeiro-RJ
alcaim@cetuc.puc-rio.br

Resumo - ***Em codificação de vídeo baseada em objeto usando SA-DCT – "Shape Adaptive Discrete Cosine Transform", a informação sobre a segmentação pode ser usada para possibilitar alguns métodos alternativos de partição em blocos da região que contém o objeto, como proposto em [1], de forma a reduzir o número de blocos a serem codificados, melhorar a eficiência da SA-DCT e, consequentemente, reduzir a taxa de bits da codificação de textura. Neste trabalho propõe-se avaliar e eficiência desses métodos de partição sob dois critérios de otimização diferentes: primeiro, reduzindo-se o número de blocos que contêm pixels da borda do objeto segmentado e segundo, minimizando-se o número total de blocos a serem codificados.***

Palavras-chave – Codificação de vídeo, SA-DCT, partição em blocos.

1. INTRODUÇÃO

Para blocos inteiramente contidos no objeto segmentado, a SA-DCT fornece uma alta eficiência de codificação devido à elevada capacidade de compactação de energia e descorrelação de coeficientes [2]. Entretanto, isso não é necessariamente verdade nos blocos que contêm o contorno do objeto – os blocos de fronteira. Em diversos desses blocos, o número de pixels que efetivamente pertencem ao objeto pode ser muito pequeno, reduzindo o potencial da SA-DCT para compactação de energia [3]. Uma possível solução para este problema consiste em encontrar uma técnica eficiente de partição da região que contém o objeto, de modo a reduzir o número de blocos a serem codificados [1]. A conseqüência disso é que os blocos de fronteira terão uma maior potência média, a qual estará distribuída em um número menor de coeficientes DC e de baixa freqüência. Aliado a isso, pode-se conseguir ainda uma redução da carga computacional relativa à estimação e compensação de movimento baseadas em bloco, se o número de blocos que particionam o objeto é menor. Para uma dada qualidade requerida da imagem, é possível trabalhar com taxas de bits menores se utilizarmos os algoritmos de partição adaptativos à forma, ao invés do método tradicional de partição.

Nos métodos de compressão de imagem baseada em transformadas, o quadro é dividido em blocos de NxN pixels, sendo uma linha de blocos denominada camada de blocos. A dimensão N normalmente utilizada é igual a 8, ou seja, cada bloco possui 64 pixels. Definindo-se como pixel de referência da camada o pixel superior esquerdo do primeiro bloco dessa camada, tem-se que no método tradicional de partição, o pixel de referência da primeira camada de blocos é sempre o pixel (1,1), ou seja, o pixel superior esquerdo do quadro. Como os pixels de referência das demais camadas também estão localizados na primeira coluna e distanciados de N linhas um em relação ao outro, é necessário que se conheça apenas a referência da primeira camada para que se possa determinar as demais. A Fig.1 ilustra o método de partição convencional aplicado a um quadro de imagem sintética de tamanho 40x32 pixels. O único objeto, sobreposto ao fundo branco, consiste de uma pilha de quatro quadrados pretos de tamanho 8x8 pixels, deslocados. Sendo o pixel de referência (X_{ref} , Y_{ref}) igual a (1,1), o primeiro bloco da primeira camada corresponde exatamente ao bloco superior esquerdo do quadro. Na figura são mostrados todos os blocos 8x8 resultantes da partição.

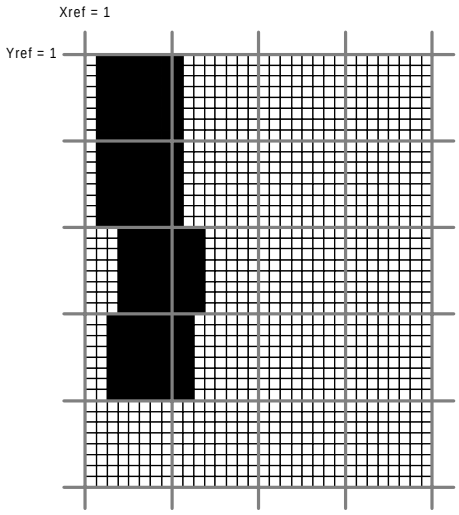

Fig.1: Quadro de imagem com partição convencional

Nos métodos de partição adaptativa à região segmentada, denominados métodos SARP - *Shape-adaptive region partitioning*, a redução do número de blocos que particionam o objeto é obtida a partir da modificação das posições horizontais das camadas de blocos. Em [1] foram apresentados dois métodos SARP: o método ortogonal e o método flexível. No método

ortogonal, os pixels de referência de todas as camadas estão localizados na mesma coluna, exatamente como na partição convencional, sendo necessário que se conheça apenas a referência da primeira camada. Contudo, o universo de busca do pixel de referência desta camada é o conjunto dos N^2 pixels pertencentes ao primeiro bloco da primeira camada de blocos da partição convencional. O pixel escolhido será aquele que, ao ser utilizado como referência para o primeiro bloco da primeira camada da partição, produzir como resultado o menor número de blocos a serem codificados, segundo o critério de otimização adotado. Já no método flexível, as posições horizontais das camadas de blocos (colunas em que se localizam os pixels de referência) podem ser escolhidas de forma independente, sendo necessário que se conheça o pixel de referência de cada uma dessas camadas. Evidentemente, a distância vertical (número de linhas) entre os pixels de referência é igual a N.

Os métodos ortogonal e flexível são implementados em duas versões diferentes [1], as quais se diferenciam quanto ao procedimento de busca dos pixels de referência. A primeira versão é chamada de busca completa, ou procedimento ótimo, onde todos os N^2 pixels candidatos à referência são testados. A segunda versão é chamada de busca parcial, ou procedimento simplificado, onde se reduz o universo dos pixels a serem testados, visando à diminuição da carga computacional. Portanto, tratam-se de quatro algoritmos diferentes, denominados: ortogonal ótimo, ortogonal simplificado, flexível ótimo e flexível simplificado.

Neste trabalho, os quatro algoritmos foram implementados sob dois diferentes critérios de otimização para a escolha das coordenadas dos pixels de referência. O primeiro visa à minimização do número de blocos de contorno (NBC). No caso de ocorrer o mesmo número de blocos de contorno para dois ou mais pares de coordenadas de referência distintos, recorre-se à minimização do número total de blocos, que corresponde à soma do número de blocos de contorno com o número de blocos inteiramente contidos no objeto. O segundo critério minimiza inicialmente o número total de blocos (NTB) e, no caso de haver empate entre duas ou mais referências, utiliza também a informação do número de blocos de contorno.

Este artigo apresenta uma análise detalhada dos algoritmos existentes para o particionamento adaptativo à forma do objeto, testando-os com relação a dois diferentes critérios de otimização e comparando-os em termos de redução relativa do número de blocos. Para cumprir esses objetivos, na Seção 2 deste artigo será feita a descrição do método ortogonal nas versões ótima e simplificada e na Seção 3, será apresentado o detalhamento da implementação do método flexível, também em suas duas versões de busca. Os resultados experimentais dos algoritmos de partição aplicados a seqüências de imagens reais serão mostrados e analisados em tabelas na Seção 4, para os dois critérios de otimização. Finalmente, na Seção 5, serão apresentadas as conclusões finais deste trabalho.

2. OS MÉTODOS ORTOGONAL ÓTIMO E ORTOGONAL SIMPLIFICADO

Considerando-se que o objeto a ser particionado esteja inscrito em uma região retangular como mostra a Fig.2, definimos X_{min} como sendo a coluna coincidente com a borda esquerda dessa região (coluna 2) e Y_{min}, a linha coincidente com a sua borda superior (linha 1). Neste trabalho, a implementação dos métodos ortogonais preocupou-se em limitar a busca do pixel de referência entre aqueles que, pertencendo ao universo de busca definido na Seção 1, não estejam abaixo de Y_{min} e nem à direita de X_{min}. Dessa forma, garante-se que nenhum pixel pertencente ao objeto fique fora da região da imagem a ser particionada, com a vantagem adicional de limitar os pares de busca (X_{ref} , Y_{ref}) entre (1,1) e (X_{min} , Y_{min}).

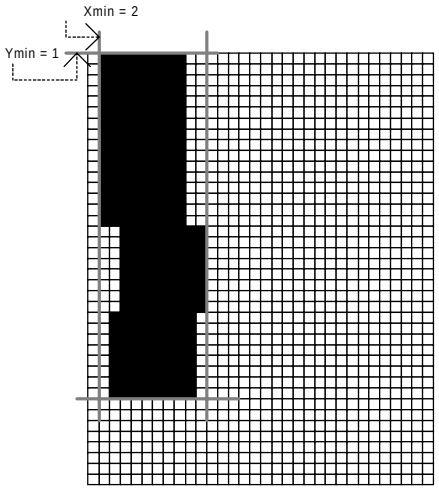

Fig.2: Objeto circunscrito na região retangular

No método ortogonal ótimo, para cada par ordenado de referência ($X_{candidata}$, $Y_{candidata}$), calcula-se o número de blocos resultante da partição com essa referência e escolhe-se o par que produzir o menor resultado, segundo o critério de otimização escolhido (minimização do número de blocos de contorno ou minimização do número total de blocos). Portanto, a busca será realizada entre N^2 pixels possíveis, no máximo. No método ortogonal simplificado, fixa-se a referência vertical candidata $Y_{candidata\_fixa}$ em 1 e varia-se a referência horizontal $X_{candidata}$. A referência horizontal candidata que produzir o menor número de blocos para $Y_{candidata\_fixa}$ igual a 1 é a escolhida e, uma vez fixa em X_{ref}, procede-se à determinação da referência vertical $Y_{ref} \mid X_{ref}$, selecionando-se aquela que, dentre as candidatas, proporcionar o menor número de blocos. Este método realiza a busca em apenas 2N pixels candidatos no máximo, reduzindo o tempo necessário à computação.

A Fig.3 ilustra o método de partição ortogonal aplicado a um quadro de imagem sintética, onde o pixel de referência (X_{ref} , Y_{ref}) foi igual a (2,1).

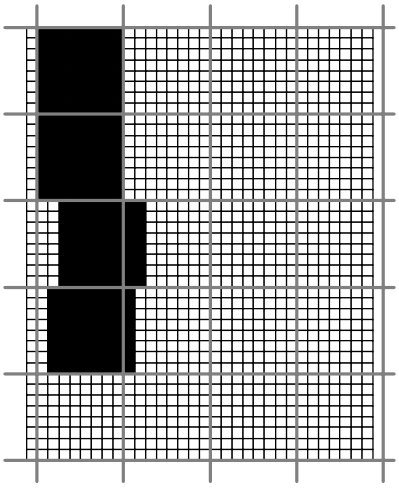

Fig.3: Quadro particionado utilizando-se os métodos Ortogonal Ótimo e Ortogonal Simplificado

3. OS MÉTODOS FLEXÍVEL ÓTIMO E FLEXÍVEL SIMPLIFICADO

Nos métodos flexível ótimo e flexível simplificado, de forma diferente dos métodos ortogonais, a referência horizontal de cada camada de blocos pode ser diferente das demais, permitindo com isso que as camadas de blocos fiquem deslocadas umas em relação às outras.

Se os pixels pertencentes ao objeto que estiverem localizados na i-ésima camada de blocos forem circunscritos na i-ésima região retangular da forma como foi descrito na seção 2, Xi_{min} corresponderá à coluna coincidente com a borda esquerda dessa região retangular e definirá o limite do intervalo de busca da referência horizontal para esta camada. Em Fig.4, são mostradas as regiões retangulares que circunscrevem os pixels de cada uma das camadas e os respectivos Xi_{min}.

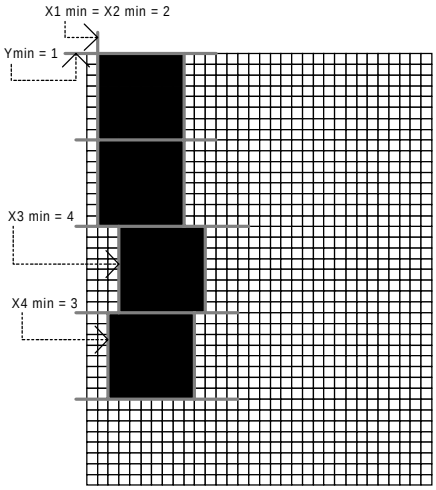

Fig.4: Regiões retangulares que circunscrevem os pixels pertencentes ao objeto de cada uma das camadas de blocos

Na implementação do algoritmo flexível ótimo, inicialmente fixa-se uma referência vertical $Y_{candidata\_fixa}$ e calcula-se o número de blocos produzidos em cada camada de blocos, a partir da variação da referência horizontal em cada uma dessas camadas entre os valores 1 e o respectivo Xi_{min}. Esse procedimento possibilita a determinação da referência horizontal ótima $X_{ref}(i)$ da i-ésima camada, dado que a referência vertical seja aquela anteriormente fixada: $X_{ref}(i) \mid Y_{candidata\_fixa}$. O número de blocos no quadro, dado que a referência vertical é $Y_{candidata\_fixa}$, é igual à soma dos blocos ocorridos em cada uma das camadas, utilizando-se as referências horizontais ótimas $X_{ref}(i) \mid Y_{candidata\_fixa}$. A seguir, varia-se a referência vertical $Y_{candidata\_fixa}$ até o valor máximo igual a Y_{min} (borda superior da região retangular que circunscreve todo o objeto) e repete-se o procedimento descrito, até que tenha sido calculado o número de blocos ótimo para cada uma das possíveis referências verticais candidatas (1 a Y_{min}). A referência vertical Y_{ref} selecionada será aquela que produzir o menor número de blocos conforme o critério de otimização escolhido. As referências horizontais serão aquelas calculadas no passo anterior, para a referência vertical ótima: $X_{ref}(i) \mid Y_{ref}$.

O algoritmo flexível simplificado mescla procedimentos do algoritmo ortogonal simplificado com parte do algoritmo flexível ótimo. Inicialmente, fixa-se uma mesma referência horizontal $X_{candidata\_fixa}$ para todas as camadas de blocos e, variando-se $Y_{candidata}$ entre 1 e Y_{min}, calcula-se o número de blocos obtidos para cada par ($X_{candidata\_fixa}$, $Y_{candidata} \mid X_{candidata\_fixa}$). A referência vertical Y_{ref} escolhida será a $Y_{candidata}$ que produzir o menor número de blocos segundo o critério de otimização escolhido, exatamente como no algoritmo ortogonal simplificado. Fixando-se Y_{ref}, calculam-se referências horizontais ótimas distintas para cada uma das i-ésimas camadas de blocos: $X_{ref}(i) \mid Y_{ref}$, da forma como foi feito no algoritmo flexível ótimo.

A Fig.5 ilustra o método de partição flexível aplicado ao quadro de imagem sintética, onde as referências horizontais das diversas camadas de blocos coincidiram com os respectivos Xi_{min} mostrados em Fig.4.

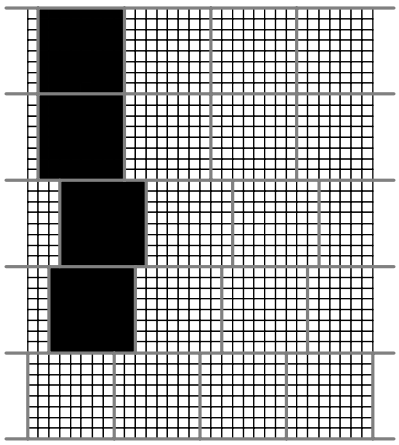

Fig.5: Quadro particionado utilizando-se os métodos Flexível Ótimo e Flexível Simplificado

4. RESULTADOS EXPERIMENTAIS

Foram implementados em ambiente MATLAB os dois métodos propostos em [1], em suas duas versões, gerando quatro algoritmos de partição: (1) Ortogonal Ótimo; (2) Ortogonal Simplificado; (3) Flexível Ótimo e (4) Flexível Simplificado. Nos algoritmos de partição implementados, utilizaram-se quadros segmentados das seqüências *"Children"* (quadro 1), *"Fish and Logo"* (quadro 10) e *"Weather"* (quadro 10), extraindo-se um objeto de cada quadro. As imagens foram representadas por matrizes de pixels indexados por: (coluna , linha) ou (posição horizontal , posição vertical), sendo que a coordenada (1,1) corresponde ao pixel superior esquerdo do quadro. Para a partição em blocos, os quadros de imagem foram divididos em camadas de blocos retangulares de dimensão 8.

A seguir foram aplicados os quatro algoritmos de partição à região do objeto segmentado dos três quadros de imagem, utilizando-se, na implementação de cada um dos algoritmos, dois critérios distintos de otimização: critério 1 (minimização de NBC, o número de blocos de contorno) e critério 2 (minimização de NTB, o número total de blocos). Nas Tabelas 1 a 6 são apresentados os resultados da partição dos objetos extraídos dos quadros, utilizando-se cada um dos critérios de otimização: número de blocos de contorno (NBC), número total de blocos (NTB) e a constante TPR, que indica o tempo de processamento relativo desses algoritmos em relação ao método de partição convencional, que utiliza como referência o pixel (1,1) do quadro. É mostrada também a porcentagem da redução (valores negativos) ou do aumento (valores positivos) do número de blocos de contorno (NBC) e do número total de blocos (NTB), em relação à partição convencional.

Na Fig.6, é mostrado o objeto extraído do quadro 1 da seqüência *"Children"*, sendo os resultados da partição mostrados nas Tabelas 1 (critério de otimização 1) e 2 (critério de otimização 2), respectivamente.

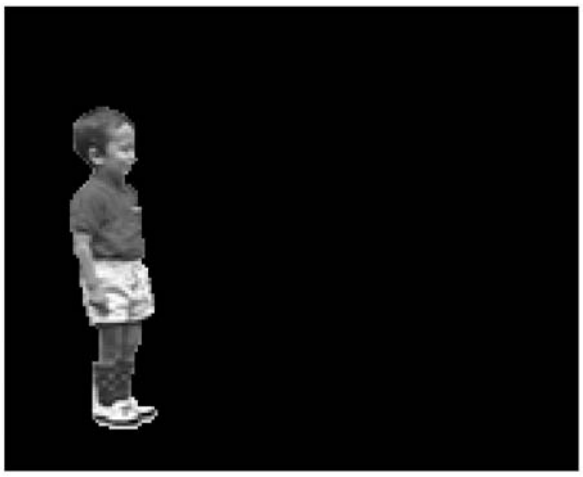

Fig.6: Objeto extraído do quadro 1 da seqüência "Children"

Tabela 1: Resultados para "Children" usando o critério de otimização 1 (NBC)

ALGORITMO	NBC	NTB	TPR
Convencional	30	43	1
Ortogonal Ótimo	28 (-6, 7%)	43 (-0,0%)	63,4
Ortogonal Simplificado	30 (-0,0%)	40 (-6,98%)	15,9
Flexível Ótimo	23 (-23,3%)	37 (-14,0%)	109,2
Flexível Simplificado	24 (-20,0%)	37 (-14,0%)	21,9

Tabela 2: Resultados para "Children" usando o critério de otimização 2 (NTB)

ALGORITMO	NBC	NTB	TPR
Convencional	30	43	1
Ortogonal Ótimo	29 (-3,3%)	39 (-9,3%)	63,5
Ortogonal Simplificado	30 (-0,0%)	40 (-7,0%)	16,1
Flexível Ótimo	23 (-23,3%)	37 (-14,0%)	110,1
Flexível Simplificado	24 (-20,0%)	37 (-14,0%)	21,7

No quadro 1 da seqüência *"Children"*, ao se utilizar o critério de otimização 1, a versão simplificada do algoritmo ortogonal mostrou-se ineficiente, ao passo que a versão ótima reduziu em quase 7% o número de blocos de contorno. Já as versões do algoritmo flexível apresentaram desempenhos semelhantes, reduzindo em cerca de 20 a 23% o número dos blocos de contorno. Em conseqüência da minimização do número de blocos de contorno, o número total de blocos também foi reduzido em quase todos os casos, chegando a 14% em ambas as versões do método flexível.

Ao optar-se pela minimização do número total de blocos, os resultados foram semelhantes, sendo exatamente iguais com a implementação dos algoritmos flexíveis.

Na Fig.7, é mostrado o objeto extraído do quadro 10 da seqüência *"Fish and Logo"*. Os resultados da partição deste objeto são apresentados nas Tabelas 3 e 4.

Para esse objeto, constata-se que o critério de otimização 1 produz resultados idênticos nas versões ótima e simplificada de um mesmo método. Em comparação à partição convencional, os métodos ortogonal e flexível, em quaisquer das versões, reduziram cerca de 26% e 35%, respectivamente, o número de blocos de contorno, e em 7 a 11%, o número total de blocos. Utilizando-se o critério de otimização 2, os algoritmos ótimos apresentaram o mesmo desempenho do caso anterior. Em contrapartida, os algoritmos

simplificados mostraram-se menos eficientes que as versões ótimas, produzindo 5 blocos de contorno a mais, além de fazer com que o número total de blocos fosse superior em 2 unidades.

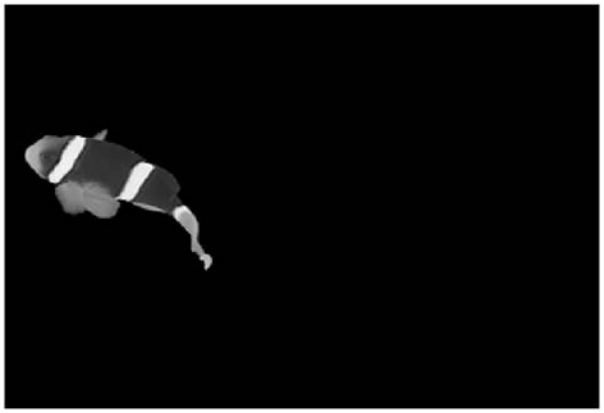

Fig.7: Objeto extraído do Quadro 10 da seqüência "Fish and Logo"

Tabela 3: Resultados para "Fish and Logo" usando o critério de otimização 1 (NBC)

ALGORITMO	NBC	NTB	TPR
Convencional	46	80	1
Ortogonal Ótimo	34 (-26,1%)	74 (-7,5%)	60,38
Ortogonal Simplificado	34 (-26,1%)	74 (-7,5%)	15,14
Flexível Ótimo	30 (-34,8%)	71 (-11,3%)	138,3
Flexível Simplificado	30 (-34,8%)	71 (-11,3%)	25,6

Tabela 4: Resultados para "Fish and Logo" usando o critério de otimização 2 (NTB)

ALGORITMO	NBC	NTB	TPR
Convencional	46	80	1
Ortogonal Ótimo	34 (-26,1%)	74 (-7,5%)	60,45
Ortogonal Simplificado	39 (-15,2%)	76 (-5,0%)	15,16
Flexível Ótimo	30 (-34,8%)	71 (-11,3%)	139,4
Flexível Simplificado	35 (-23,9%)	73 (-8,8%)	23,08

Na Fig.8, mostra-se o objeto extraído do quadro 10 da seqüência "*Weather*", cuja partição derivou os resultados apresentados nas Tabelas 5 e 6.

Fig.8: Objeto extraído do Quadro 10 da seqüência "Weather"

Tabela 5: Resultados para "Weather" usando o critério de otimização 1 (NBC)

ALGORITMO	NBC	NTB	TPR
Convencional	35	129	1
Ortogonal Ótimo	29 (-17,1%)	128 (-0,8%)	61,77
Ortogonal Simplificado	29 (-17,1%)	128 (-0,8%)	15,48
Flexível Ótimo	24 (-31,4%)	121 (-6,2%)	101,0
Flexível Simplificado	24 (-31,4%)	121 (-6,2%)	20,44

Tabela 6: Resultados para "Weather" usando o critério de otimização 2 (NTB)

ALGORITMO	NBC	NTB	TPR
Convencional	35	129	1
Ortogonal Ótimo	36 (+2,9%)	127 (-1,6%)	61,83
Ortogonal Simplificado	36 (+2,9%)	127 (-1,6%)	15,63
Flexível Ótimo	24 (-31,4%)	121 (-6,2%)	100,7
Flexível Simplificado	24 (-31,4%)	121 (-6,2%)	20,42

Para o objeto da seqüência "*Weather*", os resultados das versões ótima e simplificada de um mesmo método foram idênticos, independente do critério de otimização implementado. Ao se utilizar o critério de otimização 1, ocorreu uma redução significativa do número de blocos de contorno usando-se os algoritmos ortogonais (cerca de

17%). Nos algoritmos flexíveis, essa redução foi ainda mais relevante (cerca de 31%), reduzindo ainda o número total de blocos em aproximadamente 6%. Já ao se utilizar o critério 2, a redução dos blocos a serem codificados, bem como dos blocos de contorno, só foi significativa ao se utilizarem os algoritmos flexíveis. Quando os algoritmos ortogonais foram empregados, na tentativa de diminuir o número total de blocos, ocorreu inclusive o aumento do número de blocos de contorno.

Com relação à carga computacional, pode ser observado a partir das Tabelas 1 a 6 que o tempo de processamento relativo dos algoritmos de partição, em relação à partição convencional, não depende da imagem a ser codificada, tendo relação apenas com o tipo de algoritmo implementado. As versões simplificadas são cerca de 4 a 5 vezes mais rápidas que os versões ótimas e o método flexível é cerca de 1,5 vezes mais demorado que o método ortogonal, em ambas as versões.

Nas Tabelas 7 e 8, são mostradas as porcentagens médias do número de blocos de contorno e também do número total de blocos que foram reduzidas ao se aplicarem cada um dos algoritmos aos objetos extraídos dos quadros de imagem, para cada um dos dois critérios de otimização, comparando-os à forma convencional de partição. Por essas tabelas, percebe-se que os resultados proporcionados pelos algoritmos flexíveis são praticamente insensíveis ao critério de otimização escolhido. O algoritmo flexível ótimo produz exatamente os mesmos resultados, independente do critério de otimização.

Também se verifica das Tabelas 7 e 8, como já era esperado, que o critério de otimização 1 (NBC) mostrou-se mais eficiente que o critério de otimização 2 (NTB) na redução do número de blocos de contorno (à exceção do algoritmo flexível ótimo, onde os resultados são idênticos). Observa-se ainda que o critério de otimização 1 (NBC) foi mais eficiente que o critério de otimização 2 (NTB) também na minimização do número total de blocos, quando implementados quaisquer dos algoritmos simplificados. Portanto, levando-se em consideração a carga computacional exigida, a aplicação dos algoritmos simplificados, aliada ao critério de otimização 1 (NBC), parece ser a melhor estratégia.

Tabela 7: Redução média do número de blocos usando o critério de otimização 1 (NBC), comparativamente à partição convencional

ALGORITMO	NBC	NTB
Ortogonal Ótimo	-16,63%	-2,76%
Ortogonal Simplificado	-14,41%	-5,08%
Flexível Ótimo	-29,85%	-10,47%
Flexível Simplificado	-28,74%	-10,47%

Tabela 8: Redução média do número de blocos usando o critério de otimização 2 (NTB), comparativamente à partição convencional

ALGORITMO	NBC	NTB
Ortogonal Ótimo	-8,85%	-6,12%
Ortogonal Simplificado	-4,12%	-4,51%
Flexível Ótimo	-29,85%	-10,47%
Flexível Simplificado	-20,35%	-9,64%

Comparando-se os resultados das versões simplificadas dos métodos ótimo e flexível, implementadas com o critério de otimização 1 (NBC), a eficiência do algoritmo flexível simplificado foi cerca de duas vezes superior à eficiência do algoritmo ortogonal simplificado, às custas de um tempo de processamento relativo (TPR) aproximadamente 1,3 vezes maior.

5. CONCLUSÃO

Os algoritmos de partição adaptativa à região segmentada para o propósito de codificação de vídeo baseada em objeto mostraram-se eficientes no objetivo de reduzir o número de blocos, comparativamente à partição convencional.

Levando-se em consideração a eficiência na redução do número de blocos de contorno e do número total de blocos a serem codificados, bem como a carga computacional exigida, as versões simplificadas dos algoritmos mostraram ser a escolha mais acertada, quando implementados sob o critério de otimização 1 (NBC). Na versão simplificada, o algoritmo flexível é 1,3 vezes mais lento, porém, cerca de duas vezes mais eficiente que algoritmo ortogonal. A princípio, portanto, o algoritmo flexível simplificado seria a melhor opção. A escolha entre uma forma de implementação e a outra dependerá, basicamente, dos recursos de *hardware* disponíveis.

REFERÊNCIAS

[1] J.H.Moon at all, *"Shape-Adaptive Region Partitioning Method for Shape-Assisted Block-Based Texture Coding"*, IEEE Transactions on Circuits and Systems for Video Technology, Vol.7, No.1, pp.240-246, February 1997.

[2] T. Sikora, B.Makai, *"Shape-Adaptive DCT for Generic Coding of Video"*, IEEE Transactions os Circuits and Systems for Video Technology, Vol.5, No.1, pp.59-62, February 1995.

[3] M.Bi at all, *"Comment on Shape-Adaptive DCT for Generic Coding of Video"*, IEEE Transactions os Circuits and Systems for Video Technology, Vol.6, No.6, pp.686-688, December 1996.