Codificação de Seqüências de Imagens com Quantização Vetorial

Carlos Antonio Reinaldo Costa

Paulo Lício de Geus

Relatório Técnico DCC–09/93

Abril de 1993

Codificação de Seqüências de Imagens com Quantização Vetorial

Carlos Antonio Reinaldo Costa*

Paulo Lício de Geus†

Sumário

Avanços na tecnologia digital já permitem várias aplicações empregando a transmissão de imagens digitais de vídeo. Entretanto, existem alguns problemas que surgem quando se trata de transmissão de imagens digitais em tempo real. Devido a limitações impostas pela largura de banda da maioria das redes locais, normalmente é impossível transmitir toda a grande quantidade de informação necessária para representar as imagens. Portanto, é necessário que se elabore um método rápido de compressão de imagens, que seja capaz de atingir altas taxas de compressão, mantendo uma qualidade suficientemente boa nas imagens reproduzidas.

Neste artigo é proposto um método de compressão de seqüências de imagens que combina quantização vetorial com estratégias de detecção de movimento e transmissão progressiva. Também é introduzido um esquema de classificação que tem o objetivo de dar uma melhor representação às arestas de imagens reproduzidas por quantização vetorial. São descritas algumas variações do método e são listados e analisados alguns resultados experimentais.

*Departamento de Ciência da Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

†Departamento de Ciência da Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP. Pesquisa desenvolvida com suporte financeiro parcial do CNPq, FAPESP e FAEP/UNICAMP

1 Introdução

As técnicas de compressão de imagens surgem da necessidade de minimizar o número de bits necessários para representação de imagens, visando o armazenamento ou transmissão das mesmas. Os métodos podem ou não envolver perda de informação com respeito à imagem original. Evidentemente, quando existe perda, pode-se atingir taxas de compressão muito maiores; entretanto, em algumas aplicações a perda de informação não é admissível e nestes casos as taxas de compressão geralmente não ficam muito acima de 2:1 quando se lida com imagens de mundo real.

No caso específico deste trabalho a perda de informação é admissível. Determinar até que ponto esta perda é aceitável é algo que geralmente é feito de maneira experimental, usando-se de meios um tanto subjetivos, como a análise visual das images reproduzidas. A qualidade destas imagens e as taxas de compressão atingidas dependem de vários fatores, como:

- Método de compressão utilizado
- Quantidade e velocidade de movimento
- Aspectos de cada imagem em particular como: Freqüência espacial, definição de arestas, ruído, etc.

Uma grande quantidade de técnicas de compressão de imagens foi desenvolvida nos últimos anos, algumas das quais, voltadas para imagens isoladas e outras para seqüências de vídeo. O objetivo desta pesquisa foi a elaboração de um esquema combinando algumas destas técnicas de modo a permitir que se atinja altas taxas de compressão com a possibilidade de codificar e decodificar as imagens em tempo real. Para elaborar tal esquema, é necessário combinar métodos de codificação que reduzam tanto a redundância espacial (dentro de um mesmo quadro) quanto a redundância temporal (entre quadros contíguos). Para redução de redundância espacial a principal técnica utilizada foi a quantização vetorial, pois atende aos nossos objetivos de maneira bastante satisfatória.

Para redução de redundância temporal foram usadas estratégias de detecção de movimento e transmissão progressiva. A combinação destas estratégias foi inicialmente sugerida por Geus [Geu90] que elaborou um esquema baseado em quantização vetorial que foi posteriormente modificado em alguns aspectos por Costa [Cos93] visando a obtenção de melhores resultados.

Iniciamos este artigo com uma visão geral sobre as variações de quantização vetorial que foram utilizadas na elaboração do esquema; em seguida são descritas duas variantes do esquema propriamente dito e, por fim, é proposto um algoritmo para quantização vetorial classificada que foi adaptado a uma das variantes na tentativa de obter melhores resultados.

2 Quantização Vetorial

A quantização vetorial tem chamado a atenção de um grande número de pesquisadores nos últimos anos, dando origem a trabalhos como o de Gersho [Ger82], Gray [Gra84] e Nasrabadi e King [NK88], e muitos outros. Ela pode ser vista como uma generalização de um processo conhecido como quantização escalar que é usado, entre outras coisas, na digitalização de imagens, e que consiste em mapear números reais em um sub-conjuto finito e pré-definido do conjunto R dos números reais. No caso da quantização vetorial, ao invés de números são mapeados vetores k-dimensionais em um sub-conjunto finito e pré-definido do espaço euclidiano R^k.

Na prática, um quantizador vetorial de dimensão k, consiste basicamente de dois mapeamentos: Um codificador α que, para cada vetor de entrada $\mathbf{x} = (x_0, x_1, ..., x_{k-1})$, associa um símbolo de canal $\alpha(\mathbf{x})$ de um conjunto finito M; e um decodificador β que para cada símbolo de canal v em M, associa um vetor $\beta(v)$ de um conjunto finito $D \subset R^k$. Chamaremos este conjunto de dicionário e os padrões nele contidos de vetores de código. O número de vetores de código em geral é de 2^l, onde l é o comprimento dos vetores binários que representam os símbolos de canal. No caso de compressão de imagens, os vetores são na verdade

blocos retangulares de tamanho fixo. A escolha do símbolo de canal a ser transmitido é feita por meio de comparações entre o bloco corrente e cada um dos vetores de código do dicionário. O codificador e o decodificador devem possuir dicionários iguais. A idéia é tentar encontrar o vetor de código que mais se aproxime do bloco a ser codificado e transmitir o símbolo de canal que o representa. A taxa de compressão atingida dependerá do número de vetores de código e, conseqüentemente, do tamanho dos símbolos de canal. A qualidade da imagem reproduzida dependerá não só do tamanho do dicionário, mas também de como ele foi gerado. Existe um algoritmo conhecido como LBG [LBG80] destinado à construção de quantizadores vetoriais que são ótimos no sentido de minimizar a distorção média para um determinado conjunto de vetores denominado seqüência de treinamento. Quanto maior e melhor escolhida for a seqüência de treinamento, maiores as chances de se produzir um quantizador que reproduza imagens com qualidade.

2.1 Variações de Quantização Vetorial

Pode-se observar que, em um esquema de quantização vetorial como o descrito acima, existe um sério problema de custo computacional. O número de comparações e cálculos de distorção a serem feitos, quando se percorre o dicionário linearmente, pode se tornar muito grande (considere por exemplo um dicionário de 1024 vetores de código representando blocos de 8x8). Existem soluções alternativas que evitam uma busca linear no dicionário. Em uma delas, conhecida como *tree-searched VQ* [BGGM80], ao invés de termos um dicionário linear, temos uma árvore binária onde o k-ésimo nível representa um dicionário intermediário com 2^{k-1} vetores de código. O quantizador fará somente duas comparações para cada nível da árvore (uma com cada um dos vetores de código para os quais o nó anterior aponta). Neste caso o número de cálculos de distorção que o quantizador fará será de $2\log_2 n$, onde n é o número de vetores de código do dicionário final. Este método não é ótimo, isto é, não há garantia de que o vetor de código escolhido é de fato a melhor opção para representar o bloco, mas em geral os resultados são bastante

satisfatórios, e com um significativo ganho em velocidade.

Existem também, estratégias que visam melhorar a qualidade da imagem reproduzida. Uma delas é utilizada por métodos conhecidos como *mean/shape VQ* e *gain/shape VQ* [Gra84], e consiste em dividir a informação que descreve o bloco em duas componentes: Uma dada por vetores normalizados, que descreve a "forma", e outra, dada por um escalar, que representa a quantidade de energia. Isto permite que dicionários menores produzam imagens de qualidade.

Uma outra estratégia, denominada quantização vetorial classificada, consiste em produzir vários dicionários ao invés de um único. Cada dicionário seria "especializado" em uma certa categoria de blocos, em geral privilegiando os blocos que contém arestas, pois normalmente estes são os mais difíceis de reproduzir.

Algumas destas estratégias são usadas nos passos de quantização vetorial do esquema que descrevemos a seguir.

3 MDPT/VQ

Nesta seção vamos examinar um esquema de compressão de seqüências de imagens que foi proposto por Geus [Geu90] e chamado de MDPT/VQ (*Moviment Detected, Progressive Transmission with Vector Quantization*). Além da quantização vetorial, o método se baseia nas seguintes estratégias:

Detecção de Movimento: é feita por blocos de imagem; um método adequado deve ser utilizado para minimizar o número de blocos a serem codificados sem afetar significativamente a qualidade da imagem reproduzida. Os blocos onde não é detectado movimento são simplesmente copiados do quadro anterior na decodificação. Um exemplo de método que dá bons resultados é medir a distorção entre blocos na mesma posição em quadros contíguos e verificar se estão abaixo ou acima de um limiar convenientemente escolhido. Este limiar é usado para evitar que blocos sejam detectados com movimento apenas por oscilação de ruído temporal. Um exemplo de medida de distorção que pode ser usada é o

erro quadrático, que é definido para dois vetores $\mathbf{v} = (v_0, \ldots, v_{k-1})$ e $\mathbf{w} = (w_0, \ldots, w_{k-1})$ como,

$$d(\mathbf{v}, \mathbf{w}) = \sum_{i=0}^{k-1} (v_i - w_i)^2$$

Descrição Prévia do Bloco: uma vez detectado movimento em um bloco, é utilizado um método eficiente de codificação para dar uma primeira descrição do mesmo. Várias técnicas poderiam ser utilizadas com este objetivo, entre elas, BTC (*Block Truncation Coding*), subamostragem com interpolação, quantização vetorial e codificação por transformadas [Geu90]; sendo que estas duas ultimas são as que produzem melhores resultados. O esquema que descrevemos aqui é baseado em quantização vetorial, mas também já foram desenvolvidos esquemas que utilizam estratégias semelhantes, mas substituindo-se a quantização vetorial por codificação por transformadas [Cos93].

Transmissão Progressiva: esta estratégia é baseada em um trabalho de Dreizen [Dre87], e consiste em transmitir a informação de atualização das imagens reconstruídas ao longo de mais de um quadro da seqüência. Após ser feita a descrição prévia do bloco, continua-se a transmitir informação adicional até que a distorção esteja abaixo de um certo limiar (a medida de distorção usada neste caso também foi o erro quadrático). Neste procedimento, além da quantização vetorial foi usado também uma técnica conhecida como DPCM (*Differential Pulse Code Modulation*) que consiste em codificar as diferenças de brilho entre píxels, ao invés de codificar os valores integralmente.

3.1 Variantes de MDPT/VQ

O MDPT/VQ possui duas variantes principais: *Block Eight*, baseada em blocos de 8×8 e *Block Four*, baseada em blocos de 4×4. A seguir damos uma breve descrição destas variantes.

3.1.1 *Block Eight*

Nesta estratégia a descrição prévia dos blocos é dada por um vetor de código que mapeia blocos de 8×8, entretanto a detecção de movimento é feita separadamente em cada um dos quadrantes dos blocos; se apenas um quadrante for detectado com movimento, a descrição prévia será feita por um vetor de código que mapeia blocos de 4×4. A seqüência de passos de transmissão progressiva, se não for detectado novo movimento em um bloco, é a seguinte:

1. No momento em que o movimento é detectado, o identificador (símbolo de canal) de um vetor de código para blocos de 8×8 é transmitido.

2. No quadro seguinte, os dois quadrantes correspondentes a uma das diagonais do bloco reconstruído são testados com respeito à distorção em relação ao quadro corrente; se a distorção nestes quadrantes é maior que um certo limiar, os identificadores de dois vetores de código representando os blocos de 4×4 envolvidos são transmitidos; se a distorção estiver abaixo do limiar, o algoritmo avança para o passo de DPCM.

3. No próximo quadro, os dois quadrantes correspondentes à outra diagonal são testados como no passo anterior, e segue-se o mesmo procedimento.

4. No quadro seguinte, cada quadrante é testado em ordem como nos passos anteriores; se a distorção estiver abaixo do limiar, o processo é encerrado; caso contrário é feita uma reconstrução residual baseada em DPCM.

5. O passo 4 é repetido até que a distorção fique abaixo do limiar (cada quadrante é tratado separadamente).

Para permitir busca rápida, o quantizador vetorial utilizado é do tipo *tree-searched VQ*, e também foram usadas estratégias do *mean/shape VQ*, visando uma melhor qualidade com dicionários menores. Calcula-se

a média aritmética de brilho de cada bloco e este valor é subtraído de cada píxel do bloco e codificado separadamente. Para codificar a média, também foi usado DPCM, isto é, são codificadas as diferenças entre médias de blocos na mesma posição em quadros contíguos. Para isto usamos um quantizador escalar de 16 níveis (4 bits) projetado baseando-se em um método de construção de quantizadores para DPCM descrito por Sharma e Netravali [SN77]. Os dicionários utilizados foram especialmente projetados para blocos com média zero; para isto, a média de brilho dos blocos da seqüência de treinamento foi subtraída durante a construção. Os dicionários para blocos de 8×8 tinham 8192 vetores de código (identificadores com 13 bits) e para blocos de 4×4 tinham 2048 vetores (identificadores com 11 bits). O passo de DPCM residual é feito a 1 bit por píxel, ou seja, 16 bits para cada bloco de 4×4 que indicam adição ou subtração, em cada píxel, de um fator de 6 níveis de brilho.

As taxas de compressão atingidas com o *Block Eight* dependem muito do tipo de cena e quantidade de movimento nas imagens. Em seqüências típicas de teleconferência (com movimentos de mãos, ombros e rosto) foi possível atingir taxas entre 20:1 e 40:1 que são relativamente altas e com uma qualidade nas imagens reproduzidas que pode ser considerada aceitável para algumas aplicações.

3.1.2 *Block Four*

O objetivo desta estratégia é reproduzir imagens com melhor qualidade, abrindo mão das altas taxas de compressão. Neste caso as imagens são divididas em blocos de 4×4 ao invés de 8×8. A descrição prévia do bloco é feita, portanto, por um vetor de código que mapeia blocos de 4×4. Se não for detectado novo movimento, a seqüência de passos de transmissão progressiva é a seguinte:

1. No momento em que o movimento é detectado o identificador de um vetor de código para o bloco de 4×4 correspondente é transmitido.

2. No quadro seguinte, é calculada a distorção do bloco reconstruído com respeito ao bloco corrente. Se estiver maior que um limiar

pré-estabelecido, passa-se para uma reconstrução residual baseada em DPCM; se a distorção estiver abaixo do limiar nenhuma ação é feita.

3. No quadro seguinte, a distorção é mais uma vez medida com relação ao bloco corrente, mas desta vez a comparação é feita com um limiar menor; se a distorção estiver acima do limiar, passa-se para a reconstrução residual por DPCM; se estiver abaixo encerra-se o processo.

4. O passo anterior é repetido até que a distorção esteja abaixo do limiar.

O método de quantização vetorial usado é o mesmo adotado para o *Block Eight*, inclusive no que diz respeito à codificação da média. Entretanto, o dicionário para blocos de 4×4 usado no *Block Four* é menor, tendo apenas 256 vetores de código (identificadores de 8 bits). Usamos um dicionário menor neste método devido ao fato de que em alguns casos a taxa de compressão já fica bem próxima do limite mínimo para transmissão através de redes locais como *Ethernet* [Geu90], o que impede que aumentemos o tamanho dos identificadores dos vetores de código.

As taxas de compressão atingidas com este esquema estão entre 10:1 e 30:1; não são tão altas quanto as atingidas pelo *Block Eight*, mas a qualidade das imagens reproduzidas é significativamente superior, apesar de ainda ser visível alguma degradação, especialmente nas arestas.

3.2 Resultados

Nos testes aqui descritos utilizamos seqüências com 64 quadros de 256×256 píxels. Os quadros 31 e 63 de duas destas seqüências são mostrados na Figura 1. A seqüência *hand3* contém bastante movimento e foi usada para testar a reconstrução do fundo após a passagem de um objeto em movimento, e a seqüência *talk2* é uma cena típica de teleconferência, e contém uma quantidade moderada de movimento (rosto e ombros).

Os resultados que obtivemos, com respeito à relação sinal/ruído (SNR) e à taxa de compressão, são sumarizados nas Tabelas 1 e 2. A relação

Figura 1: Seqüências originais: *hand3* (acima) e *talk2* (abaixo).

sinal/ruído é um criterio quantitativo de avaliação da qualidade das imagens e é medida em decibéis (dB). Em geral, quanto maior for a SNR melhor será a qualidade visual da imagem (este fato nem sempre é verdadeiro).

Seqüência	Taxa Média		Taxa Mínima	
	Compressão	SNR (dB)	Compressão	SNR (dB)
hand3	23,49:1	28,28	21,54:1	27,52
talk2	34,33:1	30,02	27,20:1	29,00

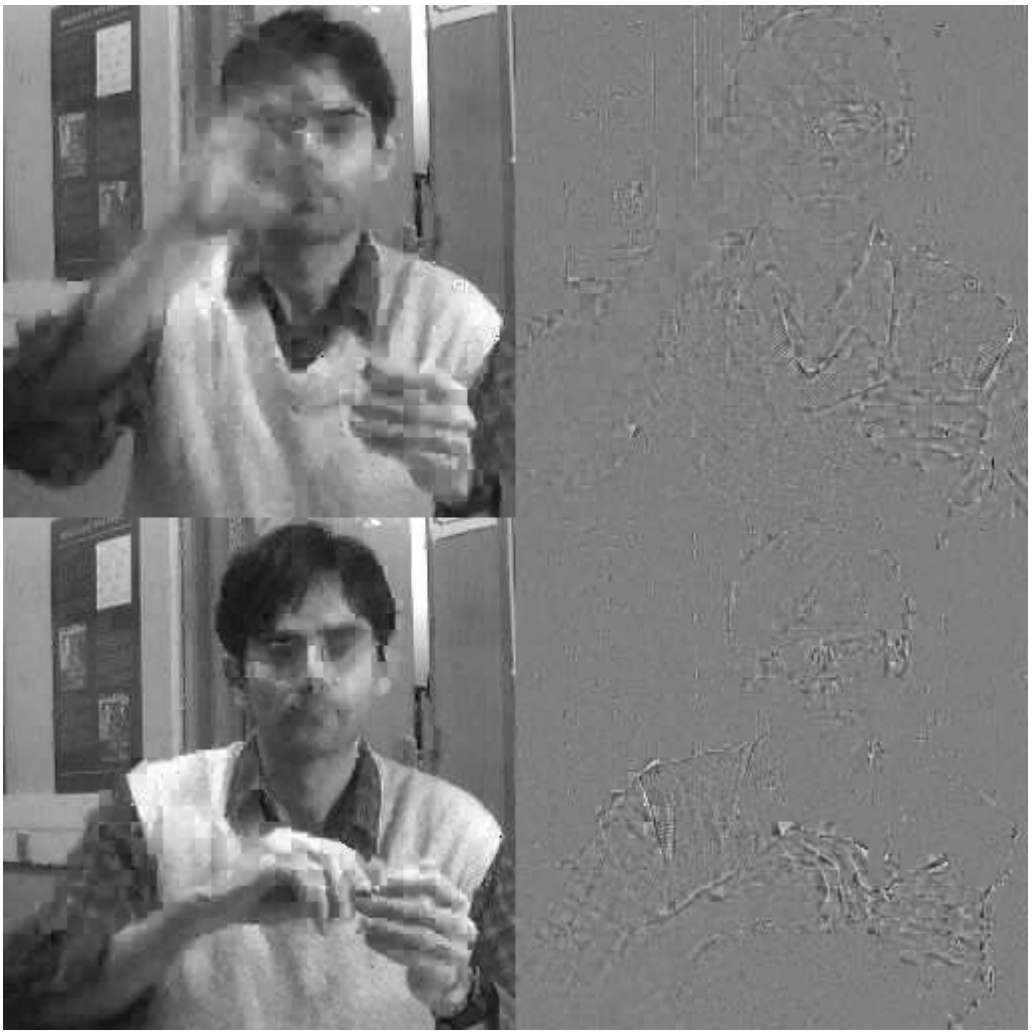
Tabela 1: Resultados obtidos com o *Block Eight*.

Seqüência	Taxa Média		Taxa Mínima	
	Compressão	SNR (dB)	Compressão	SNR (dB)
hand3	12,91:1	30,40	10,58:1	29,74
talk2	23,07:1	31,72	16,29:1	31,05

Tabela 2: Resultados obtidos com o *Block Four*.

Os quadros 31 e 63 da seqüência *hand3* reconstruída são mostrados nas Figuras 2 e 3, juntamente com as imagens de erro. Os principais defeitos visíveis nas imagens concentram-se nas arestas em movimento. Nas imagens geradas pelo *Block Eight* esta degradação é bem mais visível; vários detalhes de regiões em movimento são perdidos. Entretanto, levando-se em conta que as imagens são mostradas em movimento, esta degradação não chega a ser excessiva. No caso do *Block Four*, a qualidade das imagens reproduzidas é significativamente superior, mas ainda é possível perceber algumas falhas, especialmente nas arestas diagonais.

Para melhorar a representação das arestas, sem que necessariamente haja uma queda na taxa de compressão, uma técnica que pode ser utili-



SNR: quadro 31 - 28,79 dB; quadro 63 - 28,26 dB.

Figura 2: Seqüência *hand3* codificada pelo *Block Eight*

SNR: quadro 31 - 30,63 dB; quadro 63 - 30,45 dB.

Figura 3: Seqüência *hand3* codificada pelo *Block Four*

zada é a quantização vetorial classificada. Na seção seguinte é descrita uma estratégia desenvolvida com este objetivo, e são mostrados alguns resultados juntamente com as conclusões obtidas sobre esta técnica.

4 Quantização Vetorial Classificada

O principal objetivo desta técnica é dar uma melhor representação para as arestas por meio da separação de diferentes tipos de blocos, dando maior ênfase àqueles que contêm arestas. Em uma abordagem feita por Ramamurthi e Gersho para blocos de 4×4 [RG86] foram definidas 31 classes diferentes, sendo que 28 delas foram dedicadas a arestas. Em nossa abordagem, preferimos definir um classificador mais simples, com um número menor de classes, pois precisamos manter a possibilidade de compressão e descompressão de seqüências de imagens em tempo real. A seguir é dada uma descrição da versão básica do classificador que foi desenvolvido.

4.1 O Classificador

Nós definimos um total de 12 classes que são ilustradas na Figura 4. A classe *plano* é destinada aos blocos que não possuem variação significativa no valor de brilho de seus píxels; a classe *variação suave* contém os blocos que possuem variação de baixa intensidade; na classe *textura*, os blocos são quase que totalmente constituídos de variações abruptas, mas não chegando a configurar nenhum tipo de aresta; a classe *aresta complexa* destina-se a blocos que contém arestas irregulares ou múltiplas; as classes de arestas simples são em um total de 8, divididas em horizontais, verticais, diagonais positivas e diagonais negativas com as diferentes polaridades. O algoritmo de classificação que desenvolvemos é bastante simples e não totalmente livre de falhas, entretanto baseando-se nos resultados obtidos, podemos garantir que o número de blocos classificados erradamente é muito pequeno e não chega a prejudicar o desempenho do método.

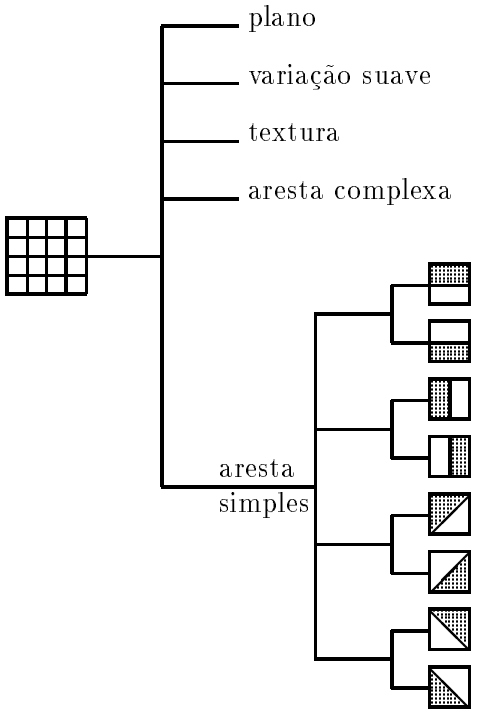


Figura 4: Classificação.

4.1.1 O Algoritmo de Classificação

O algoritmo usa um mecanismo semelhante ao descrito por Ramamurthi e Gersho [RG86], mas os procedimentos são diferentes. Ele é dividido em duas fases: na primeira, é feita a atualização de uma série de contadores, baseando-se em duas tabelas que descrevem a configuração do bloco; a segunda é a árvore de decisão, que chega ao tipo do bloco baseando-se nos valores dos contadores. Nos ítens abaixo descrevemos cada uma destas fases para o caso de blocos de 4×4.

Atualização dos Contadores

As tabelas a que nos referimos são na verdade matrizes que chamaremos de D_h e D_v, de tamanhos 4×3 e 3×4 respectivamente, que são preenchidas com informações sobre as diferenças de brilho entre píxels vizinhos na horizontal e na vertical. São definidos dois limiares convenientes L_1 e L_2 de tal maneira que os valores nas matrizes são dados por:

$$D_h(i,j) = \begin{cases} -2 & \text{se} & & p(i,j) - p(i,j+1) & < -L_2 \\ -1 & \text{se} & -L_2 \leq & p(i,j) - p(i,j+1) & < -L_1 \\ 0 & \text{se} & -L_1 \leq & p(i,j) - p(i,j+1) & \leq L_1 \\ 1 & \text{se} & L_1 < & p(i,j) - p(i,j+1) & \leq L_2 \\ 2 & \text{se} & L_2 < & p(i,j) - p(i,j+1) & \end{cases} \quad (1)$$

para $i = 0, 1, 2, 3$; $j = 0, 1, 2$;

$$D_v(i,j) = \begin{cases} -2 & \text{se} & & p(i,j) - p(i+1,j) & < -L_2 \\ -1 & \text{se} & -L_2 \leq & p(i,j) - p(i+1,j) & < -L_1 \\ 0 & \text{se} & -L_1 \leq & p(i,j) - p(i+1,j) & \leq L_1 \\ 1 & \text{se} & L_1 < & p(i,j) - p(i+1,j) & \leq L_2 \\ 2 & \text{se} & L_2 < & p(i,j) - p(i+1,j) & \end{cases} \quad (2)$$

para $i = 0, 1, 2$; $j = 0, 1, 2, 3$; onde $p(i,j)$ é o valor de brilho do píxel na i-ésima linha e na j-ésima coluna do bloco. O limiar L_1 é usado para detectar pequenas variações e seu valor fica em torno de 3 para imagens de 256 tons de cinza; o limiar L_2 é usado para detectar arestas e seu valor fica em torno de 12.

Uma vez definidas as matrizes D_h e D_v, passa-se para a atualização dos contadores: h_{-2}, h_{-1}, h_1 e h_2, contam a quantidade de -2, -1, 1 e 2, respectivamente, em D_h; analogamente v_{-2}, v_{-1}, v_1 e v_2, contam estes valores em D_v.

Existe ainda um contador tx que é usado para contar pontos onde há grande diferença de brilho, mas que não chegam a definir arestas por estarem isolados; se o número de tais pontos for suficientemente grande

o bloco é classificado como *textura*, uma classe que existe para evitar que blocos com grande variação sejam classificados erradamente como arestas. Existem também os contadores c_h e c_v, que indicam as componentes horizontal e vertical de uma possível aresta. c_h e c_v são incrementados no máximo uma vez para cada linha de D_h ou coluna de D_v.

A Árvore de Decisão

As decisões são feitas baseando-se na comparação dos valores dos contadores com uma série de limiares. O primeiro teste verifica se o bloco possui algum tipo de acidente (componente de alta freqüência espacial). O bloco é considerado com acidente se:

$$c_h + c_v \geq L_{ar}. \tag{3}$$

Caso o bloco não tenha acidente, ele é considerado *variação suave* se:

$$h_{-1} + h_1 + v_{-1} + v_1 \geq L_{va}. \tag{4}$$

Caso contrário o bloco será considerado *plano*. Observe que os contadores h_{-1}, h_1, v_{-1} e v_1 poderiam, neste caso, ser substituídos por um único contador, entretanto a existência dos quatro contadores permite a possibilidade de, se for preciso, dividir a classe *variação suave* em diferentes subclasses representando diferentes orientações de variação. Isto pode ser útil no caso de lidarmos com blocos maiores.

No caso do bloco ter acidente ele será considerado *textura* se:

$$tx \geq L_{tx}; \tag{5}$$

e aresta complexa se:

$$\begin{gathered} tx < L_{tx} \\ \text{e} \quad \max(v_{-2}, v_2) + \max(h_{-2}, h_2) - c_h - c_v \geq L_{ar} \\ \text{ou} \quad \min(v_{-2}, v_2) + \min(h_{-2}, h_2) \geq L_{ar}; \end{gathered} \tag{6}$$

onde L_{ar} é um valor mínimo para que se possa definir uma aresta, ou seja, se a expressão acima é verdadeira o bloco pravavelmente conterá mais de

uma aresta. Caso contrário, o bloco é considerado aresta simples e neste caso a orientação da aresta é determinada baseando-se na tangente do ângulo de inclinação, que é estimado por:

$$\tan(\alpha) = \frac{c_v}{c_h}. \tag{7}$$

Uma vez determinado o ângulo de inclinação, as polaridades horizontal e vertical, são consideradas positivas se, respectivamente:

$$v_2 > v_{-2} \quad \text{e} \quad h_2 > h_{-2}. \tag{8}$$

Caso contrário são consideradas negativas. Baseando-se nestes resultados o bloco será classificado em uma das oito classes de arestas simples.

Os valores dos limiares foram determinados experimentalmente por meio de exaustivos testes. Eles são diretamente relacionados com o tamanho dos blocos. Para blocos de 4×4, nós chegamos aos seguintes valores:

$$L_{ar} = 3, \quad L_{va} = 12 \quad \text{e} \quad L_{tx} = 6. \tag{9}$$

Com os primeiros testes que realizamos com este esquema observamos que blocos das classes *textura* e *aresta complexa* não estavam sendo reproduzidos de maneira satisfatória, e isto se devia ao fato de que os dicionários destas classes não eram capazes de abranger toda a grande variação de tipos de blocos presentes nestas classes. Para resolver este problema decidimos encontrar uma maneira de diminuir o número de blocos classificados nestas categorias. Isto foi feito com a ajuda de um filtro de média[1] usando vizinhanças de 2×2 píxels. Cada bloco detectado como sendo *textura* ou *aresta complexa* é filtrado e classificado novamente. O processo de filtragem suaviza os blocos sem chegar a desconfigurá-los. Com este procedimento houve uma queda substancial no número de blocos destas classes, o que permitiu que diminuíssemos os tamanhos dos respectivos dicionários; a maioria dos blocos passou a ser

[1] Tipo de filtro que substitui o valor de brilho de cada píxel pela média de brilho de uma vizinhança.

classificada como aresta simples ou variação suave e pode ser reproduzida com maior fidelidade.

Um outro problema que foi observado é que os dicionários para arestas simples não eram capazes de reproduzir bem a blocos com arestas de alta intensidade, o que nos levou a criar novas classes de arestas dirigidas a este tipo de bloco. Os dicionários usados para estas classes foram os mesmos usados para os outros blocos de arestas simples; entretanto, se o bloco for considerado aresta de alta intensidade, o vetor de código tem sua aresta acentuada no processo de decodificação.

A Tabela 3 mostra os tamanhos dos dicionários para cada uma das classes. O fato de estarmos usando *tree-searched VQ* tira bastante da flexibilidade quanto à escolha destes tamanhos (todos devem ser potências de 2), de modo que eles não são ótimos. Contudo procuramos adaptar da melhor maneira possível, levando em conta a probabilidade de um bloco estar em cada classe e a distorção média de cada uma delas.

4.2 Resultados

Quando comparamos imagens codificadas com ajuda deste esquema com imagens codificadas sem classificação, usando um dicionário de 256 vetores, pudemos observar que melhorou a representação das arestas mais abruptas, mas esta melhora nem sempre é observada em arestas suaves. Olhando mais atentamente pode-se até encontrar algumas regiões que pioraram com o uso de classificação; mas de uma maneira geral, a qualidade das imagens melhorou.

Ao introduzirmos classificação no MDPT/VQ também foi possível observar uma pequena melhora na qualidade das imagens, que também é indicada pela relação sinal/ruído. A Figura 5 mostra os quadros 31 e 63 da seqüência *hand3* codificada pelo *Block Four* com classificação e a Tabela 4 resume os resultados obtidos em termos de compressão e relação sinal/ruído. Houve uma pequena queda na taxa de compressão, mas vale lembrar que usamos 4 bits para identificar a classe de cada bloco; usando um método mais eficiente, como código de Huffman [Huf52], podemos codificar o identificador da classe com uma média em torno de 2,7 bits,

Classe	Polaridade	Tamanho do dicionário (blocos)
plano		4
variação suave		128
textura		4
aresta complexa		16
aresta 0°	+	16
	−	16
aresta 90°	+	16
	−	16
aresta 45°	+	16
	−	16
aresta −45°	+	16
	−	16
Total		280

Tabela 3: Tamanhos dos dicionários para cada classe.

o que provavelmente eliminaria a diferença.

Nós chegamos também a tentar usar quantização vetorial classificada em blocos de 8×8, visando adaptar ao *Block Eight*, mas os resultados não foram nem um pouco animadores. A classificação só produz bons resultados para blocos pequenos (menores que 6×6 [RG86]); quando aumentamos o tamanho do bloco, o número de variações possíveis cresce exponencialmente e seria preciso usar um número muito grande de classes, o que poderia dar origem a um algoritmo de classificação extremamente complexo. Isto não é nem um pouco convidativo, levando-se em conta que temos que manter a possibilidade de codificação e decodificação em tempo real.

SNR: quadro 31 - 31,37 dB; quadro 63 - 30,70 dB.

Figura 5: Seqüência *hand3* codificada pelo *Block Four* com classificação.

Seqüência	Taxa Média		Taxa Mínima	
	Compressão	SNR (dB)	Compressão	SNR (dB)
hand3	12,13:1	30,92	9,71:1	30,28
talk2	22,10:1	32,00	15,21:1	31,52

Tabela 4: Resultados obtidos com o *Block Four* com classificação.

5 Conclusão

Neste artigo analisamos o desempenho de diferentes variações do MDPT/VQ. Com o *Block Eight* conseguimos obter taxas relativamente altas de compressão com um nível de qualidade razoável e que pode ser considerado satisfatório para algumas aplicações. Com o *Block Four* as taxas de compressão não foram tão altas, mas a qualidade das imagens reproduzidas foi significativamente superior, apesar de ainda ser possível verificar algumas falhas.

Como uma tentativa de melhorar a qualidade das imagens reproduzidas pelo *Block Four* sem uma conseqüente queda na taxa de compressão, foi desenvolvido um esquema de quantização vetorial classificada, que chegou a produzir resultados positivos, mas que em alguns casos ainda deixou um pouco a desejar. A melhora produzida pela classificação é bem mais evidente em imagens que possuem muitas arestas bem definidas e de alta intensidade. Em imagens muito ruidosas, ou constituídas, em sua maioria, por variações suaves, a diferença não chega a ser tão significativa. Talvez resultados melhores possam ser obtidos com um algoritmo de classificação mais sofisticado; entretanto, como necessitamos de processamento em tempo real, não podemos usar um método com custo computacional muito elevado. Vale lembrar que não esgotamos todas as possibilidades desta técnica; uma alternativa seria combiná-la com outros métodos: com o uso de codificação por transformadas, por exemplo, blocos dos tipos plano ou variação suave podem ser codificados de maneira bastante eficiente.

Métodos que usam transformadas também já foram estudados pelos autores, sendo inclusive elaborado um esquema que usa estratégias bastante semelhantes às do MDPT/VQ, mas substituindo a quantização vetorial por DCT, que é uma técnica que se adapta melhor à estratégia de transmissão progressiva [Cos93].

Referências

[BGGM80] Buzo, A., Gray Jr., A. H., Gray, R. M., and Markel, J. D. Speech coding based on vector quantization. *IEEE Trans. Acoust., Speech and Signal Proc.*, 28(5):562–574, October 1980.

[Cos93] Costa, Carlos A. R. *Técnicas de Compressão de Seqüências de Imagens Visando Transmissão em Tempo Real.* Tese de Mestrado, Universidade Estadual de Campinas, Campinas, 1993.

[Dre87] Dreizen, Howard M. Content-driven progressive transmission of grey-scale images. *IEEE Trans. Communications*, 35(3):289–296, March 1987.

[Ger82] Gersho, Allen. On the structure of vector quantizers. *IEEE Trans. Inf. Theory*, 28(2):157–166, March 1982.

[Geu90] Geus, Paulo L. *Approaches to Live Image Trasmission Between Workstations Over Limited-Bandwidth Networks.* PhD thesis, University of Manchester, Manchester, 1990.

[Gra84] Gray, Robert M. Vector quantization. *IEEE ASSP Magazine*, pages 4–29, April 1984.

[Huf52] Huffman, David A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, September 1952.

[LBG80] Linde, Y., Buzo, A., and Gray, R. M. An algorithm for vector quantizer design. *IEEE Trans. Communications*, 28(1):84–95, January 1980.

[NK88] Nasrabadi, N. M. and King, R. A. Image coding using vector quantization: A review. *IEEE Trans. Communications*, 36(8):957–971, August 1988.

[RG86] Ramamurthi, B. and Gersho, A. Classified vector quantization of images. *IEEE Trans. Communications*, 34(11):1105–1115, November 1986.

[SN77] Sharma, D. K. and Netravali, A. N. Design of quantizers for DPCM coding of picture signals. *IEEE Trans. Communications*, 25(11):1267–1274, November 1977.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

04/93 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

05/93 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

06/93 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

07/93 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

08/93 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br