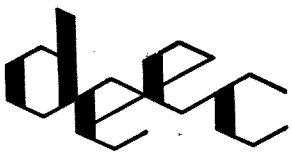


DEPARTAMENTO DE

ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

COMPRESSÃO DE VIDEO
DE REDUZIDA COMPLEXIDADE
PARA BAIXAS CADENCIAS

Maria Isabel de Castro Lopes Martins

COMPRESSÃO DE VIDEO DE REDUZIDA COMPLEXIDADE PARA BAIXAS CADENCIAS

Maria Isabel de Castro Lopes Martins

Licenciada em Engenharia Electrotécnica pela
Faculdade de Engenharia da Universidade do Porto

Tese submetida para satisfação parcial
dos requisitos do programa de Mestrado em
Engenharia Electrotécnica e de Computadores,
perfil Telecomunicações

Faculdade de Engenharia da Universidade do Porto

Departamento de Engenharia Electrotécnica
e de Computadores

Porto, Janeiro de 1992

Tese realizada sob a supervisão do Doutor

Artur Pimenta Alves

Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores

Faculdade de Engenharia da Universidade do Porto

AGRADECIMENTOS

Ao Professor Doutor Artur Pimenta Alves o meu agradecimento por ter tornado possível a realização desta tese, pelo incentivo e interesse sempre manifestados.

Ao Engenheiro Luís Corte-Real, desejo expressar a minha gratidão pela orientação e apoio prestados, quer na realização do trabalho quer na elaboração deste texto, pelos valiosos e imprescindíveis conhecimentos transmitidos, e pela sua constante disponibilidade.

Quero também agradecer a todos os colegas do INESC que, directa ou indirectamente, estiveram ligados ao processo de elaboração desta tese, em especial a:

- Engº Nuno Vasconcelos, pela sua colaboração;
- Engª Berta Baptista, pelo seu contributo na revisão de parte deste trabalho e pela ajuda e apoio sempre prontamente manifestados.

Ao Doutor Campos Neves agradeço a sua preciosa ajuda na revisão deste texto, bem como os conselhos, o ânimo e a presença amiga.

À minha irmã o meu obrigado pela paciência, tempo e cuidado que dedicou à elaboração de quase todas as figuras apresentadas.

Agradeço ainda ao INESC a bolsa de estudo concedida para a realização deste Curso de Mestrado, e as condições de trabalho oferecidas.

INDICE

Lista de Figuras

Capítulo 4

Lista de Tabelas

Lista de Quadros

Lista de Abreviaturas

ADC	Analog to Digital Converter (Conversor Analógico-Digital)
ALU	Arithmetic and Logic Unit (Unidade Aritmética e Lógica)
ARAU	Auxiliary Arithmetic Unit (Unidade Aritmética Auxiliar)
ASIC	Application Specific Integrated Circuit
CCIR	International Radio Consultative Committee
CCITT	International Telegraph and Telephone Consultative Committee
CIF	Common Intermediate Format
CODEC	Coder/Decoder (Codificador/Descodificador)
COST	Cooperation for Scientific and Technical Research
CPU	Central Processing Unit (Unidade Central de Processamento)
CVQ	Classified Vector Quantizer
C30	TMS320C30
DCT	Discrete Cosine Transform (Transforma Discreta de Cosseno)
DFT	Discrete Fourier Transform (Transformada Discreta de Fourier)
DMA	Direct Memory Access (Acesso Directo à Memória)
DPCM	Differential Pulse Code Modulation
DSP	Digital Signal Processor (Processador de Sinal Digital)
FFT	Fast Fourier Transform (Transformada Rápida de Fourier)
I/O	Input/Output (Entrada/Saída)
KLT	Karhunen-Loeve Transform (Transformada de Karhunen-Loeve)
LMS	Least Mean Squares
LOT	Lapped Orthogonal Transform
LPF	Low-Pass Filter (Filtro Passa-Baixo)
MAC	Multiply/Accumulate
MB	Macro Block (Macrobloco)
MC	Motion Compensation (Compensação de Movimento)
MFLOPS	Million of Floating-Point Operations Per Second (Milhões de Operações Vírgula-Flutuante Por Segundo)
MIPS	Million of Instructions Per Second (Milhões de Instruções Por Segundo)
PAL	Programmable Array Logic
PC	Personal Computer (Computador Pessoal)
PCM	Pulse Code Modulation
PLL	Phase Locked Loop
PVQ	Predictive Vector Quantization
QCIF	Quarter CIF

RAM	Random Access Memory (Memória de Acesso Aleatório)
RDIS	Rede Digital com Integração de Serviços
RM	Reference Model (Modelo de Referência)
SNR	Signal to Noise Ratio (Relação Sinal Ruído)
SPA	Significant Pixel Area
SSVQ	Sliding Search Vector Quantizer
TSVQ	Tree Searched Vector Quantizer
TVQ	Transform Vector Quantization
VCO	Voltage Controlled Oscilator (Oscilador Controlado por Tensão)
VGA	Video Graphics Array
VLC	Variable Lenght Code (Código de Comprimento Variável)
VLSI	Very Large Scale Integration
VSP	Video Signal Processor

Capítulo 1

INTRODUÇÃO

1.1 Compressão de Imagem

Em processamento digital de imagem, cada amostra da imagem digitalizada, também chamada *pixel*, é quantificada num número fixo de bits, para de seguida ser guardada ou transmitida digitalmente. A representação do sinal de imagem, discreta no tempo e em amplitude, constitui uma codificação PCM ("Pulse Code Modulation") [JAY84] da imagem. Se não for permitida uma degradação perceptível da qualidade da imagem, a codificação PCM de imagens monocromáticas de alta qualidade requer quantificação uniforme com 256 níveis de quantificação, no mínimo, correspondendo a $\log_2 256 = 8$ bits por pixel.

Surge pois o problema do grande volume de dados associado à representação digital de imagens. A utilidade da compressão de imagem verifica-se quer no armazenamento quer na transmissão de imagens, onde o objectivo é minimizar, respectivamente, a memória para armazenamento e a largura de banda necessária para transmissão em tempo-real. Assim, é importante considerar técnicas para representar uma imagem, ou a informação contida nesta, com o menor número possível de bits. Na terminologia da Teoria da Informação [GAL68] isto designa-se por *codificação da fonte*.

O processo de codificação com o objectivo de diminuir a quantidade de informação designa-se por *compressão*. Ao processo inverso dá-se o nome de *descompressão*. A razão entre a quantidade de informação existente na representação original e na representação codificada designa-se por *taxa* ou *factor de compressão*. Ao equipamento que efectua a compressão do sinal de vídeo a enviar e a descompressão do sinal recebido chama-se *CODEC de Vídeo*.

O desenvolvimento da tecnologia de vídeo digital na última década tornou possível a utilização da compressão de vídeo digital para uma grande variedade de aplicações de telecomunicações: difusão de televisão digital, teleconferência, vídeotelefone, etc. Devido à sua extensa aplicação, os esquemas de compressão e codificação têm sido de grande importância em processamento digital de imagem. Em aplicações de transmissão de sequências de imagens, as técnicas de compressão são fortemente condicionadas pelos requisitos de processamento em tempo-real, o que tende a impôr um severo limite na complexidade permitida.

1.2 Normalização

A normalização de algoritmos de compressão para vídeo foi iniciada pelo CCITT ("International Telegraph and Telephone Consultative Committee") para teleconferência e vídeotelefone. O CCITT Study Group XV é responsável pela normalização de codecs de vídeoconferência. Em 1984, o CCITT publicou as Recomendações H.120 e H.130 para codecs de videoconferência na taxa primária digital (1544 e 2048 Kbit/s). Considerando o rápido progresso da tecnologia de codificação de vídeo, o SGXV decidiu, em 1984, estabelecer o "Specialists Group on Coding for Visual Telephony" que ficou com a responsabilidade de recomendar uma norma para o algoritmo de codificação de sinais de vídeo para transmissão digital a n*384 Kbit/s (n=1, ..., 5) e p*64 Kbit/s (p=1, 2, ..., 30). Em meados de 1990, foi publicada a Recomendação H.261 [CCI90], que é actualmente a norma para codecs de vídeo para serviços audiovisuais na taxa p*64 Kbit/s. Estas taxas foram escolhidas para coincidir com as taxas de canal da Rede Digital com Integração de Serviços (RDIS), de modo a que a conectividade digital internacional "end-to-end" seja assegurada.

De facto, a RDIS vem oferecer o suporte para transmissão a estas taxas. Presentemente estão definidos dois tipos de acesso à rede RDIS:

- Acesso Básico, 2B+D, sendo a taxa de cada canal B 64Kbit/s e a do canal D 16 Kbit/s;
- Acesso Primário, 30B+D, sendo 64 Kbit/s a taxa de cada canal B e do canal D.

Resumindo, para aplicações RDIS, a taxa de transmissão para serviços audiovisuais está limitada a p*64 Kbit/s, variando p entre 1 e 30. O Acesso Básico é mais adequado para serviços do tipo vídeotelefone pelo facto de a taxa de transmissão disponível ser muito limitada. A taxa adicional disponível com o Acesso Primário permite a transmissão de imagens mais complexas e com melhor qualidade. Assim, este tipo de acesso é mais adequado para serviços do tipo vídeoconferência.

Após o estabelecimento da norma para codificação de vídeo a taxas iguais ou superiores a 64 Kbit/s, o esforço de investigação concentra-se agora na codificação de vídeo para transmissão a taxas inferiores a 64 Kbit/s, falando-se em taxas tão baixas como 8Kbit/s [COS91]. Embora a compatibilidade com H.261 seja desejável, do ponto de vista de serviço, os requisitos de qualidade a estas taxas podem beneficiar da utilização de outros métodos.

1.3 A Quantificação Vectorial

No final da década de 70 surgiram os primeiros trabalhos de quantificação vectorial aplicada à codificação de voz [BUZ80] [GER83]. A sua aplicação tem provado ser uma técnica capaz de atingir taxas de transmissão baixas com um nível de complexidade relativamente reduzido. Mais recentemente tem sido dedicada especial atenção à aplicação da quantificação vectorial à codificação de imagem [GRA84] [NAS88].

Nos últimos anos tem-se verificado uma intensa actividade de investigação na área da codificação de sequências de imagens para transmissão a taxas reduzidas. Actualmente, a solução mais comum é a codificação por transformada aplicada à imagem diferença . Recentemente, a Quantificação Vectorial tem aparecido como uma alternativa interessante aos métodos convencionais, especialmente na codificação de imagens com vista à transmissão a taxas muito baixas (64 Kbit/s ou inferiores).

Além de ser uma técnica com carácter assimptoticamente óptimo do ponto de vista da Teoria da Informação, a Quantificação Vectorial apresenta ainda uma vantagem muito importante, do ponto de vista prático, que é a extrema simplicidade da descodificação.

1.4 Implementação de Codecs de Vídeo

Devido à grande complexidade dos algoritmos de compressão, especialmente para factores de compressão muito elevados, a sua implementação requer circuitos electrónicos bastante sofisticados, o que os torna muito dispendiosos. De modo a alcançar o factor de compressão de vídeo desejado, é necessária uma enorme capacidade de processamento de sinal.

Para implementar o algoritmo de codificação de vídeo em tempo real, com custo razoável, é necessário usar tecnologia VLSI [LIO90]. Existem duas abordagens distintas para implementação de codecs de vídeo usando VLSI. Uma abordagem usa processadores de sinal e a outra usa ASICs ("Application Specific Integrated Circuits"). Cada uma tem vantagens e desvantagens. Genericamente, pode-se dizer que a abordagem usando processadores de sinal é mais flexível, requer menos esforço de desenvolvimento mas, provavelmente, irá resultar num codec mais dispendioso.

Um processador de sinal é um circuito electrónico VLSI, "single-chip", desenhado para aplicações de processamento de sinal. É programável e optimizado para tratar

grande quantidade de dados e efectuar cálculos aritméticos, como multiplicação e acumulação, rapidamente. Para operações em tempo real pode fazer-se a interface da sua entrada e saída com outros circuitos electrónicos.

Há vários tipos de processadores de sinal que podem ser usados em implementação de videocodecs. Estes incluem DSP's ("Digital Signal Processors"), Transputers e VSPs ("Vídeo Signal Processors"). Cada tipo tem a sua arquitectura e características distintas.

Como foi dito, é necessário um considerável processamento de sinal para comprimir a taxa de um sinal de vídeo para transmissão a taxas reduzidas, em tempo real. Actualmente, é vulgar utilizar-se um grande número de processadores de sinal, interligados num sistema multiprocessador, para atingir os requisitos de processamento de sinal vídeo [SA90].

De uma forma geral, pode considerar-se duas abordagens para a implementação de um codec de vídeo usando processadores de sinal: a abordagem funcional e a abordagem distribuída. A primeira substitui um módulo funcional, como a DCT/IDCT ou a estimação de movimento por um número apropriado de processadores de sinal, dependendo da sua complexidade computacional. Por outro lado, a abordagem distribuída atribui um processador de sinal a cada região da imagem, e usa o número de processadores de sinal necessário em paralelo de modo a conseguir a velocidade de processamento exigida.

1.5 Processadores de Sinal Digital (DSP)

Os DSP's têm evoluído de forma notável na última década, especialmente nos últimos anos [AHM91]. Os avanços na tecnologia de circuitos, arquitectura e algoritmos têm contribuído para a multiplicidade de novas aplicações dos DSP's.

As aplicações de processamento digital de sinal são muito exigentes em vários aspectos, nomeadamente em termos computacionais, requerendo que o processador tenha não só uma capacidade de efectuar cálculos aritméticos rapidamente, mas que esta facilidade seja utilizada de modo eficiente, minimizando as tarefas de controlo. A interface com o processador deve ser fácil, permitindo a sua integração num sistema com circuitos adicionais analógicos e digitais, e também permitindo a cooperação de vários DSP's na execução de certas tarefas mais pesadas. Outro aspecto relevante para

a vulgarização da sua utilização é a existência de um conjunto de ferramentas de suporte para desenvolvimento de "software" e simulação.

As características principais de um DSP incluem tamanho da palavra (16 ou 32 bits), cálculos com inteiros ou vírgula flutuante, memória interna, tempo de ciclo de instrução e velocidade medida por MIPS (Milhões de Instruções Por Segundo) ou por MACs (Multiplicação e Acumulação) por segundo.

Os DSP's têm sido vulgarmente usados para implementação de codecs de áudio. Geralmente, são necessários um ou dois DSP's, dependendo da complexidade da codificação e das capacidades do DSP usado. Devido à complexidade e exigência de velocidade dos algoritmos de vídeo, em geral, são necessários múltiplos DSP's em paralelo.

Tomando como referência a família de DSP's TMS320 da Texas Instruments, pode-se dizer que para aplicações de baixa velocidade, como controlo, a primeira geração (TMS32010) oferece melhor compromisso custo-desempenho enquanto para aplicações de elevadas taxas de amostragem, como processamento de vídeo/imagem, os DSP's da terceira geração (TMS320C30) com as suas capacidades de multiprocessamento, grande espaço de memória e alta velocidade são mais adequados.

De facto, os DSP's da terceira geração têm mostrado ser eficientes em grande número de aplicações de processamento de imagem [SIL91] [MAR90] [MAR91], permitindo criar sistemas bastante flexíveis e com custos razoáveis.

Com o surgimento de novas gerações de DSP's, TMS320C40 (vírgula-fixa) e TMS320C50 (vígula-flutuante) oferecendo cerca do dobro da capacidade de processamento em relação à geração anterior, pode-se esperar um desempenho superior destes processadores em aplicações de processamento de imagem.

1.6 Uma Aplicação - Codec de Baixa Resolução para Videovigilância

Com a implementação da Rede Digital com Integração de Serviços (RDIS) o assinante terá acesso a canais de 64 Kbit/s que tornam atractiva a ideia de transmitir sequências de imagens de qualidade reduzida e a baixa cadência, para aplicações do tipo da videovigilância.

A recente evolução nos domínios dos algoritmos de codificação de imagem e dos DSP's torna viável o desenvolvimento de codecs (codificador/descodificador) de baixa complexidade para aplicações que admitam qualidade pouco elevada.

Assim, foi efectuado um estudo [COR90a] para avaliação da viabilidade de aplicar a Quantificação Vectorial à codificação de sequências de imagens, a muito baixa cadência e com pouca exigência de qualidade, utilizando hardware de reduzida complexidade. Desse estudo concluiu-se ser viável implementar um codificador com um só DSP, desde que este tenha um ciclo de instrução de duração inferior a 83 ns. Relativamente ao descodificador é de realçar que este é substancialmente menos complexo do que o codificador. Este estudo conduziu à possibilidade do desenvolvimento de um serviço de vigilância remota suportada na rede RDIS.

O trabalho apresentado enquadra-se no desenvolvimento de um sistema demonstrativo de televigilância que poderá ser ensaiado na experiência piloto RDIS. O sistema será constituído por uma Estação Central de Televigilância e por uma Estação Remota, e permitirá a captação de imagens através de uma câmara vídeo acoplada à Estação Remota, efectuando-se a transmissão das imagens compactadas por dois modos alternativos: regime descontínuo imagem a imagem, ou imagem móvel.

No sistema proposto, a metodologia de transferência das imagens poderá ser de diverso tipo (por exemplo: transmissão de uma imagem a intervalos de tempo regulares; transmissão de uma imagem se detectado um movimento; transmissão de imagens móveis), sendo seleccionável pelo operador da Estação Central, de acordo com a natureza da acção de vigilância.

Embora se tome a videovigilância como aplicação motivadora deste trabalho, toda a infra-estrutura desenvolvida poderá servir como suporte de outras aplicações que necessitem de transmissão de imagem fixa ou móvel.

O projecto com vista ao desenvolvimento deste sistema inclui as seguintes tarefas:

- Desenvolvimento do subsistema de transmissão de imagem fixa, incluindo o suporte para a ligação à RDIS.

- Desenvolvimento do Codec de 64 Kbit/s de baixa resolução (180x144 pixels), permitindo a transmissão de imagem móvel.

- Implementação de uma interface de utilizador, integrando os diferentes serviços disponíveis. As funcionalidades a implementar serão, nomeadamente:
  a) Definição do regime de transferência de imagens: imagem fixa ou imagem móvel;
  b) "Congelamento" de uma imagem em regime de imagem móvel;
  c) Armazenamento e recuperação de uma imagem num banco de imagens, com indicação da origem, data e hora.

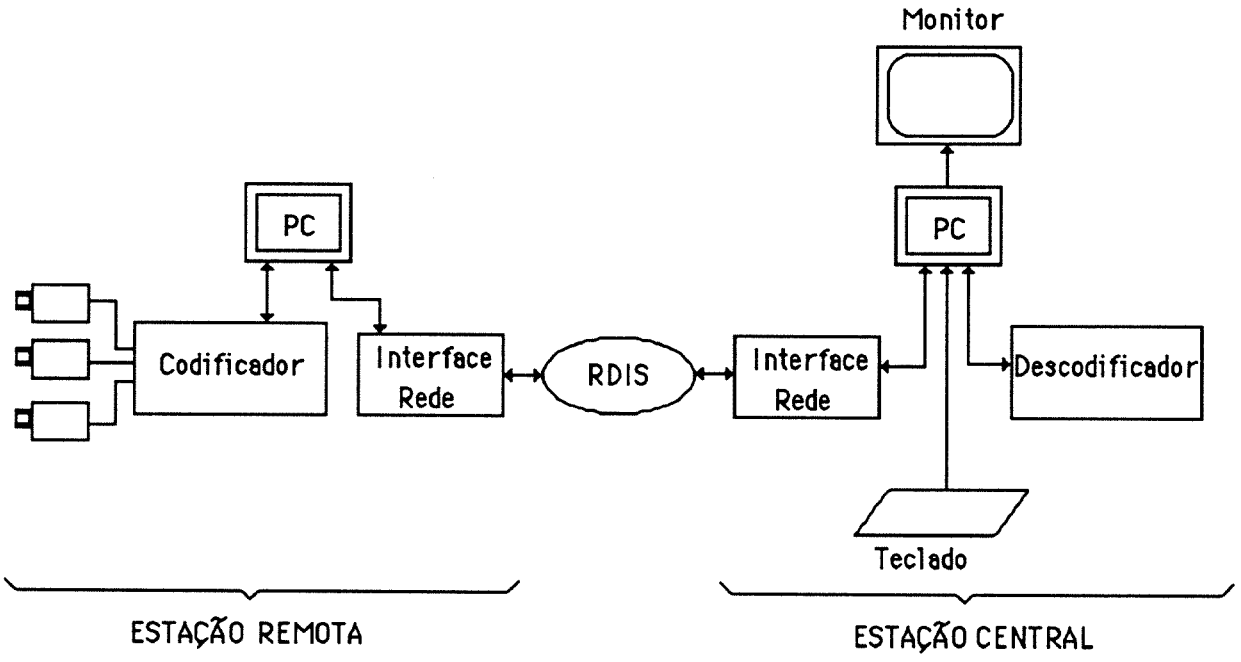


Figura 1.1 - Sistema de Televigilância.

O trabalho apresentado refere-se ao desenvolvimento do Codec de 64 Kbit/s de baixa resolução (180x144 pixels) e está a ser realizado no âmbito do projecto "Sistemas Integrados de Comunicação" (SIC), decorrente acualmente no INESC-Porto. Este trabalho foi iniciado em Fevereiro de 1991 e deverá estar concluído em Maio de 1992. Posteriormente, será feita a integração com os restantes subsistemas.

Este texto está estruturado em cinco capítulos:

No capítulo 2, intitulado "Técnicas de Compressão de Imagem", são apresentadas algumas das técnicas de compressão de imagem/vídeo mais divulgadas, sem pretensões de exaustividade. Este capítulo procura apenas introduzir os conceitos fundamentais associados às técnicas de compressão de vídeo referidas nos capítulos seguintes.

No capítulo 3, "Codificação de Sequências de Imagem para Transmissão a Taxas Reduzidas", é feita uma breve descrição do algoritmo de codificação de vídeo proposto na Recomendação H.261 do CCITT, para serviços audiovisuais na taxa p*64 Kbit/s, visto que este algoritmo é actualmente a norma para codificação a estas taxas. Em seguida, propõe-se um algoritmo baseado na quantificação vectorial adaptativa de blocos de dimensão e forma variável, para codificação a estas taxas com complexidade reduzida.

No capítulo 4, "Implementação de um Codec de Vídeoviglância", descreve-se a implementação de um codec de vídeo, de baixa resolução e qualidade reduzida, para transmissão de vídeo a 64 Kbit/s, em aplicações sem grande exigência de qualidade, como é o caso da vídeovigilância. Este codec é implementado sobre "hardware" de uso genérico, e utiliza o algoritmo de quantificação vectorial proposto na capítulo 3.

Finalmente, no capítulo 5 intitulado "Conclusões", são perspectivados desenvolvimentos futuros para o trabalho apresentado.

Capítulo 2

TÉCNICAS DE COMPRESSÃO DE IMAGEM

2.1 INTRODUÇÃO

A compressão de imagens digitalizadas tem vindo a ganhar uma importância crescente, sobretudo para transmissão, devido ao facto das capacidades dos canais serem, normalmente, limitadas e muito inferiores às exigidas pelo sinal vídeo. A compressão de imagem requer um poder e velocidade de processamento consideráveis, especialmente em aplicações de transmissão de sequências de imagens que exigem processamento em tempo-real, o que condiciona a complexidade das técnicas de compressão utilizadas.

Tipicamente, uma imagem comprimida, quando descodificada virá afectada de alguma distorção. No entanto, esta não deverá resultar numa perda de informação, ou de qualidade, apreciáveis. A eficiência de um algoritmo de compressão é medida pela sua capacidade de compactação, pela distorção resultante e também pela sua complexidade de implementação. A complexidade dos algoritmos de compressão de imagem é uma consideração particularmente importante quando se trata da sua implementação prática.

Os métodos de compressão de imagem convencionais podem ser classificados em duas grandes categorias: codificação preditiva [MUS79], em que é explorada a redundância existente nos dados, e codificação por transformada [TES79], em que a compressão é conseguida por uma transformação da imagem noutro vector, preservando a energia, de modo que a máxima informação seja compactada no menor número de amostras.

Entre as técnicas de compressão de imagem, as que usam predição têm a vantagem de serem facilmente implementáveis. Nos métodos que usam codificação por predição, como o DPCM ("Differential Pulse Code Modulation"), transmite-se a informação não em termos absolutos mas sim como a relação entre a informação actual e uma informação já existente. A compressão usando DPCM é, sem dúvida, um dos processos mais populares, devido à sua extrema simplicidade de implementação, embora não apresente um desempenho tão bom como os métodos de transformada. Como principais desvantagens apontam-se a grande sensibilidade aos erros de transmissão e às variações das propriedades estatísticas das imagens.

Os métodos de compressão por transformada, além de permitirem uma maior compressão, possuem uma maior imunidade a erros de transmissão e alteração das características da imagem. De um modo geral, a distorção devida à quantificação e a erros de transmissão, distribui-se pela imagem, durante a transformação inversa, e

torna-se menos importante visualmente. As desvantagens são, essencialmente, a complexidade de implementação e a necessidade de grande quantidade de memória.

Com o objectivo de combinar as vantagens destas duas técnicas surgiram os chamados sistemas híbridos [ROE79], Transformada/DPCM ou DPCM/Transformada, combinando as vantagens da simplicidade dos codificadores DPCM com o superior desempenho dos codificadores por transformada, particularmente para taxas de transmissão relativamente baixas. Em geral, o desempenho de um codificador híbrido situa-se entre o DPCM e a codificação por transformadas. É facilmente adaptável à codificação de imagens com ruído e às variações estatísticas dos dados, e é menos sensível a erros de transmissão do que DPCM, embora não seja tão robusto como os codificadores por transformada.

As técnicas referidas, de uma forma geral, exploram a redundância dos dados da imagem e também a percepção visual, para reduzir a taxa de transmissão. Um defeito de todas as técnicas convencionais resulta do facto de a quantificação ser efectuada em amostras individuais, ou seja, utilizarem quantificação escalar. Uma conclusão fundamental da "Rate-Distortion Theory" de Shannon [GAL68], o ramo da Teoria da Informação dedicado à compressão de dados, é que melhores resultados podem ser obtidos tratando vectores em vez de escalares, mesmo que os dados consistam numa sequência de variáveis aleatórias independentes. Assim, surgiu a quantificação vectorial [GRA84], inicialmente aplicada à codificação de voz, mostrando ser uma técnica capaz de atingir taxas de transmissão baixas com um nível de complexidade relativamente reduzido. Nos últimos anos, tem-se verificado uma intensa actividade de investigação nesta área, nomeadamente na sua aplicação à codificação de imagem. Relativamente às técnicas convencionais, DPCM e transformadas, a quantificação vectorial apresenta-se como uma alternativa vantajosa para aplicações a taxas reduzidas e como técnica complementar, nomeadamente no DPCM vectorial e na quantificação vectorial no domínio das transformadas.

Além das já citadas, existe ainda um número considerável de técnicas de compressão de imagem não referidas. De entre estas, destaca-se a codificação de contornos, em que a imagem é dividida em contornos e textura, fazendo a respectiva codificação e transmissão separadas. Por exemplo, os contornos podem ser codificados por técnicas de codificação de imagens binárias e a textura por transformadas. Como a textura tem, normalmente, pouca informação de alta frequência, pode-se desprezar grande número de coeficientes.

A compressão de imagem deu origem a imensa literatura nas últimas décadas, encontrando-se bons resumos do tema em [NET80] [JAI81] [MUS85] [NAS88].

2.2 DPCM - "Differential Pulse Code Modulation"

Em sistemas que usam predição é codificada e transmitida a diferença entre uma estimativa do valor de uma amostra e o seu valor real, em vez do valor dessa amostra representando o nível de luminância de um pixel a ser transmitido. Se esta diferença, denominada erro de predição, é quantificada e codificada, o esquema de codificação chama-se "Differential Pulse Code Modulation" (DPCM).

Em geral, os valores das amostras de pixels vizinhos estão espacialmente correlacionadas. Em sinais de vídeo, há ainda uma correlação temporal entre pixels em imagens consecutivas, também denominadas *frames*. A correlação, ou dependência estatística linear, indica que uma estimativa linear dos valores das amostras de pixels vizinhos resultará em erros de predição que têm uma menor variância do que os valores das amostras originais. Os algoritmos de predição unidimensionais usam a correlação de pixels adjacentes situados na mesma linha de varrimento. Outros esquemas mais complexos exploram ainda a correlação entre linhas e entre frames e são chamados predição bidimensional e tridimensional, respectivamente.

Num sistema DPCM, devido à menor variância do sinal a ser quantificado, codificado e transmitido, a gama dinâmica do quantificador pode ser diminuída e o número de níveis de quantificação pode ser reduzido, sendo pois necessários menos bits por pixel para a codificação, do que num sistema PCM, sem diminuir a relação sinal/ruído de quantificação. Adaptando a distribuição dos níveis de quantificação às características do sistema visual humano, consegue-se ainda maior redução da taxa de transmissão. Embora o ruído de quantificação seja aumentado por esta técnica, a qualidade da imagem não se degrada se o ruído adicional resultante não for perceptível ao olho humano.

O projecto e aplicação de técnicas que usam predição devem ter também em consideração outros factores como, por exemplo, as características do canal de transmissão. Um factor importante é a sensibilidade destes sistemas a erros de transmissão. Uma vez que as diferenças transmitidas são integradas no descodificador, um bit errado na transmissão gera uma sequência de amostras

reconstruídas erradamente, com um decaimento da amplitude do erro correspondendo à resposta impulsional do descodificador. O erro na imagem reconstruída pode propagar-se ao longo de uma linha, de linha para linha, ou de frame para frame, consoante o tipo de predição do sistema.

O factor de compressão conseguida com este tipo de codificação depende do tipo de imagens, dos requisitos de qualidade da imagem e do algoritmo de codificação.

Na maioria das imagens, os valores de pixels adjacentes estão correlacionados. Se o pixel u_{i-1} tem um determinado nível de cinzento, então o pixel adjacente u_i, na mesma linha de varrimento, provavelmente terá um valor muito próximo.

Nos sistemas de codificação que usam predição, uma estimativa do valor do pixel a ser codificado é feita a partir de informação previamente codificada e transmitida. O erro resultante da subtracção do valor estimado ao valor real do pixel é quantificado num conjunto discreto de níveis de amplitude. O elemento fundamental de um codificador DPCM é pois um filtro de predição, seguido do quantificador.

O codificador DPCM usa um filtro de predição linear que estima o valor do pixel de entrada baseado numa soma ponderada dos valores dos pixels adjacentes,

$$\hat{u}_i = \sum_{j=1}^{n} a_j \, u_{i-j} \tag{2.1}$$

sendo a_j os coeficientes do filtro.

Apenas valores de pixels previamente transmitidos são usados no cálculo da estimativa, de modo que o receptor também seja capaz de calcular $\hat{u}_i$. Os coeficientes a_j são optimizados de modo a minimizar a variância do erro de predição,

$$e_i = u_i - \hat{u}_i \tag{2.2}$$

Este é então quantificado com uma característica de quantificação não-uniforme, a qual é optimizada para produzir um erro quadrático médio de quantificação mínimo ou de acordo com a percepção visual do ruído de quantificação por parte do observador.

Na sua forma mais simples, DPCM usa o valor codificado do pixel anterior da mesma linha como estimativa. Assim, o sistema DPCM observa o valor do pixel u_{i-1} já quantificado, u'_{i-1}, e estima o valor do pixel seguinte u_i. Se $\hat{u}_i$ for o valor estimado de u_i, e se subtrair este valor ao valor real de u_i, para obter o erro de predição $e_i = u_i - \hat{u}_i$, assumindo que as estimativas são razoavelmente precisas, a diferença $u_i - \hat{u}_i$ será, em média, significativamente menor do que a amplitude do pixel u_i. Também a variância das diferenças será muito menor do que a variância dos valores originais dos pixels. Consequentemente, para codificar a sequência de diferenças, são necessários menos níveis de quantificação e menos bits do que para codificar a sequência de pixels.

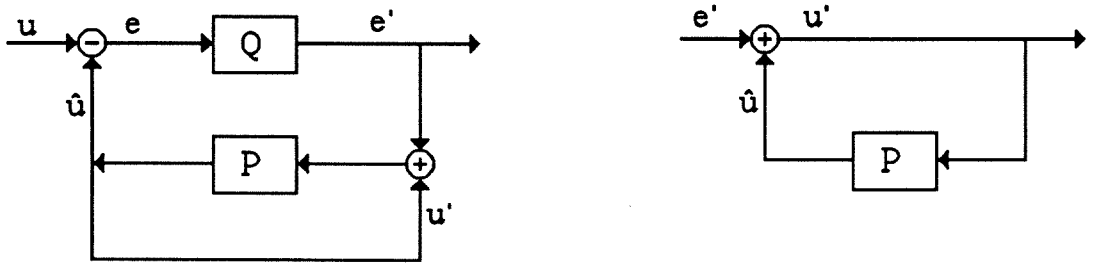


Fig. 2.1 - (a) Esquema do codificador DPCM. (b) Esquema do descodificador DPCM.

É assim removida a redundância mútua entre pixels sucessivos e quantificada apenas a informação "nova", não estimável a partir dos valores precedentes.

Na descodificação, para obter o valor reconstruído do pixel u'_i, adiciona-se à estimativa $\hat{u}_i$ o erro de predição descodificado e'_i.

Um aspecto importante deste esquema é o facto de, na codificação, a estimativa ser baseada nos valores anteriores da saída, ou seja, nos valores dos pixels anteriores afectados do erro de quantificação, em vez dos seus valores originais. Daí que o filtro de predição esteja na malha de realimentação do quantificador, como se pode ver na figura 2.1(a). Isto tem um efeito estabilizador que evita a acumulação de erros no sinal reconstruído. Na descodificação, a malha de realimentação do quantificador reconstrói o sinal. Se as saídas dos filtros de predição do codificador e do descodificador fossem diferentes, o resultado seria um erro que se iria acumulando no sinal descodificado. Um sistema DPCM permite um sincronismo perfeito entre o codificador e o descodificador, o que é muito importante para uma reconstrução correcta das imagens.

Esta característica é responsável por um dos principais problemas dos sistemas DPCM, nomeadamente a sua sensibilidade aos erros de transmissão. Um erro na transmissão da informação relativa a um pixel vai propagar-se pelos pixels seguintes.

No dimensionamento de um codificador DPCM, o filtro de predição e o quantificador são optimizados individualmente, ignorando os efeitos mútuos. Embora, em teoria, esta operação deva ser realizada conjuntamente, é feita esta aproximação devido à natureza não-linear do quantificador, o que torna a optimização conjunta impossível de realizar na prática. No entanto, quando o critério usado na optimização é o erro quadrático médio, esta aproximação não redunda numa significativa perda de qualidade.

FILTRO DE PREDIÇÃO

Filtros de predição para codificação DPCM podem ser classificados como lineares ou não-lineares, dependendo da estimativa ser uma função linear ou não dos valores das amostras previamente transmitidas. Outra divisão pode ser feita dependendo da localização dos elementos anteriores usados: filtros de predição unidimensionais usam elementos anteriores da mesma linha, filtros de predição bidimensionais usam também elementos das linhas anteriores, enquanto filtros de predição tridimensionais usam também pixels de frames previamente transmitidas.

No entanto, como foi referido, o filtro de predição é normalmente um filtro linear, sendo a estimativa do pixel u_i uma soma ponderada dos valores dos n pixels adjacentes, previamente codificados, sendo n a ordem do filtro, de acordo com a equação (2.1).

Os coeficientes a_j são escolhidos de modo a minimizar o erro quadrático médio de predição

$$E\{ | u_i - \hat{u}_i |^2 \} = E\{ | u_i - \sum_{j=1}^{n} a_j u_{i-j} |^2 \} \qquad (2.3)$$

Para encontrar os coeficientes de predição a_j que satisfazem esta condição, tomam-se as derivadas parciais do erro quadrático médio de predição em relação a cada a_j e igualam-se a zero. Assim, os valores óptimos dos coeficientes a_j são soluções das seguintes equações:

$$\sum_{j=1}^{n} a_j \, r_{k-j} = r_k \qquad k = 1, \ldots, n \tag{2.4}$$

sendo $r_{k-j} = E\{ u_{i-j} \, u_{i-k} \}$.

Os coeficientes de predição óptimos podem ser calculados de (2.4) se as autocorrelações da imagem, $E\{u_{i-j} \, u_{i-k}\}$, forem conhecidas [HAB71]. Quando $n \rightarrow \infty$ a sequência de amostras de erro torna-se completamente não-correlacionada [PAP65]. No entanto, é normalmente aceite que, se a sequência de amostras u_i puder ser modelizada por um processo autoregressivo de Markov de ordem n, então o filtro de predição óptimo será de ordem n [OPP78].

Note-se que esta análise assume o carácter estacionário do sinal e despreza os efeitos da quantificação, pois usa os valores originais dos pixels. Para codificadores que produzem imagens de elevada qualidade, os efeitos da quantificação são pequenos e podem ser desprezados.

O erro quadrático médio foi calculado por Habibi [HAB71] para filtros de predição de várias ordens. Os seus resultados mostram que, se os coeficientes do filtro estão bem adaptados às estatísticas de uma imagem, então, para essa imagem, o erro quadrático médio decresce significativamente aumentando a ordem do filtro até 3 e, a partir daí, a diminuição do erro é muito pequena. No entanto, se os coeficientes não estiverem bem adaptados, o que é mais realista, o decréscimo do erro devido ao uso de filtros de ordem superior a um não é tão significativo.

Os codificadores mais simples, e mais utilizados, usam apenas o valor do pixel anterior da linha para estimar o valor de u_i. Neste caso, verifica-se um deficiente tratamento dos contornos verticais. Com o uso de filtros bidimensionais, embora a melhoria em termos de erro quadrático médio seja pequena, o tratamento dos contornos verticais é significativamente melhorado. Além disso, com uma escolha adequada dos coeficientes, é possível melhorar a predição e tornar mais rápido o decaimento dos efeitos dos erros de transmissão na imagem reconstruída. De qualquer forma os contornos são sempre zonas de difícil tratamento, pois o dimensionamento do filtro de predição pressupõe o carácter estacionário do sinal, o que não se verifica nas regiões com contornos onde existem transições abruptas de intensidade.

QUANTIFICADOR

O critério vulgarmente usado na optimização do quantificador é a minimização do erro quadrático médio [MAX60]. No entanto têm sido propostas abordagens em que se pretende ter em consideração critérios subjectivos como, por exemplo, a manutenção do erro de quantificação abaixo de um limiar de visibilidade [SHA77] ou minimizar um erro quadrático médio de quantificação ponderado, sendo os pesos derivados de testes subjectivos [LIM67][NET77][NET77a].

Existem quatro tipos de defeitos comuns em imagens codificadas por sistemas DPCM, devido a um inadequado dimensionamento do quantificador:

1) Quando os níveis de quantificação para sinais de pequena amplitude estão muito espaçados pode ocorrer a oscilação entre dois níveis em zonas da imagem de amplitude constante, dando a aparência de ruído aleatório adicionado à imagem. Este defeito chama-se erro granular.

2) Quando a gama dinâmica do quantificador é insuficiente para acompanhar as variações do sinal observa-se uma redução da resolução da imagem. Para zonas de grande contraste, normalmente associadas a contornos, a saída do quantificador não consegue acompanhar as variações bruscas do sinal de entrada, resultando um efeito idêntico a uma filtragem passa-baixo da imagem.

3) No caso de os níveis de quantificação, para sinais de pequena amplitude, estarem muito afastados, em zonas de variação suave da amplitude, podem aparecer falsos contornos.

4) Quando ocorrem contornos em que a variação de amplitude é gradual, os pontos em que o quantificador muda de nível podem variar de linha para linha, ou de imagem para imagem, dando a falsa ideia de um contorno em oscilação. Este defeito é notório se os níveis de quantificação para sinais de grande amplitude estiverem muito afastados.

Estes erros podem ser reduzidos aumentando o número de níveis de quantificação, mas isto faz com que seja necessário um maior número de bits para os codificar.

DPCM ADAPTATIVO

Em qualquer sistema adaptativo, os parâmetros são modificados de acordo com as variações nas estatísticas do sinal, optimizando o desempenho do sistema para sinais não-estacionários. Assim, devido ao carácter não-estacionário de certas imagens, é possível melhorar o desempenho dos codificadores DPCM introduzindo o conceito de adaptatividade no filtro de predição e no quantificador [HAB77].

O objectivo dos algoritmos de predição adaptativos é gerar um sinal de erro de predição estacionário e, assim, diminuir o erro de quantificação nos contornos da imagem, onde os filtros de predição fixos, geralmente, produzem os maiores erros de predição. Ao nível do filtro de predição a adaptatividade pode ser conseguida por comutação entre diferentes filtros com base numa regra de decisão pré-estabelecida [ZSC77]. O cálculo sistemático dos coeficientes do filtro baseado em algoritmos adaptativos (como o LMS [ALE85]) embora proposto não obteve grande sucesso devido, possivelmente, à sua complexidade. Como alternativa a adaptatividade pode ser aplicada à escolha dos pixels a usar na predição, com base em critérios que tomam em consideração aspectos locais da imagem, nomeadamente, a orientação de eventuais contornos [ZSC77].

Também na quantificação têm sido feitos esforços no sentido de adaptar as características do quantificador às características locais da imagem. Em geral, pretende-se segmentar a imagem em sub-imagens de modo que, em cada uma delas, quer a percepção do ruído de quantificação, quer as propriedades estatísticas do erro de predição, sejam uniformes e estacionárias. Com este objectivo têm sido desenvolvidos vários trabalhos, uns baseados em critérios puramente estatísticos e outros em critérios de percepção visual. Um método de conseguir adaptatividade consiste na aplicação de quantificadores diferentes conforme o tipo da zona de imagem em consideração [NET77].

Tanto no caso da predição como no da quantificação, o uso de técnicas que impliquem uma escolha prévia pode originar a necessidade de enviar informação complementar para o descodificador, excepto se a decisão for tomada exclusivamente com base em informação previamente transmitida. Em geral, as técnicas adaptativas levam a melhorias significativas da qualidade de imagem para a mesma taxa de transmissão.

2.3 CODIFICAÇÃO POR TRANSFORMADA

Como já foi referido, numa imagem existe uma elevada correlação entre pixels. Com a técnica das transformadas pretende-se, por uma transformação linear, transformar um vector de dados original noutro em que os elementos sejam o menos possível correlacionados.

Uma maneira simples de interpretar a transformação de uma imagem é considerar a transformação como uma rotação multidimensional de coordenadas. Para ilustrar este ponto considere-se o exemplo simplificado usando uma hipotética imagem com apenas 8 níveis de cinzento. Considere-se um codificador por transformada unidimensional em que a imagem é primeiro dividida em vectores de 2 elementos. O vector $x = (x_1, x_2)^T$ representa dois pixels adjacentes, e na figura 2.2(a) representa-se um gráfico do nível de cinzento de x_1 em função do nível de cinzento de x_2. Uma vez que cada pixel tem um de 8 níveis, há 64 combinações possíveis de x_1 e x_2. No entanto, nem todas são igualmente prováveis. É pouco provável que x_1 tenha um valor elevado e x_2 um valor baixo, e vice-versa. Uma vez que é mais provável que pixels adjacentes tenham níveis de cinzento próximos, as combinações mais prováveis são as que se situam na vizinhança de $x_1=x_2$, ou seja, na área sombreada da figura 2.2(a).

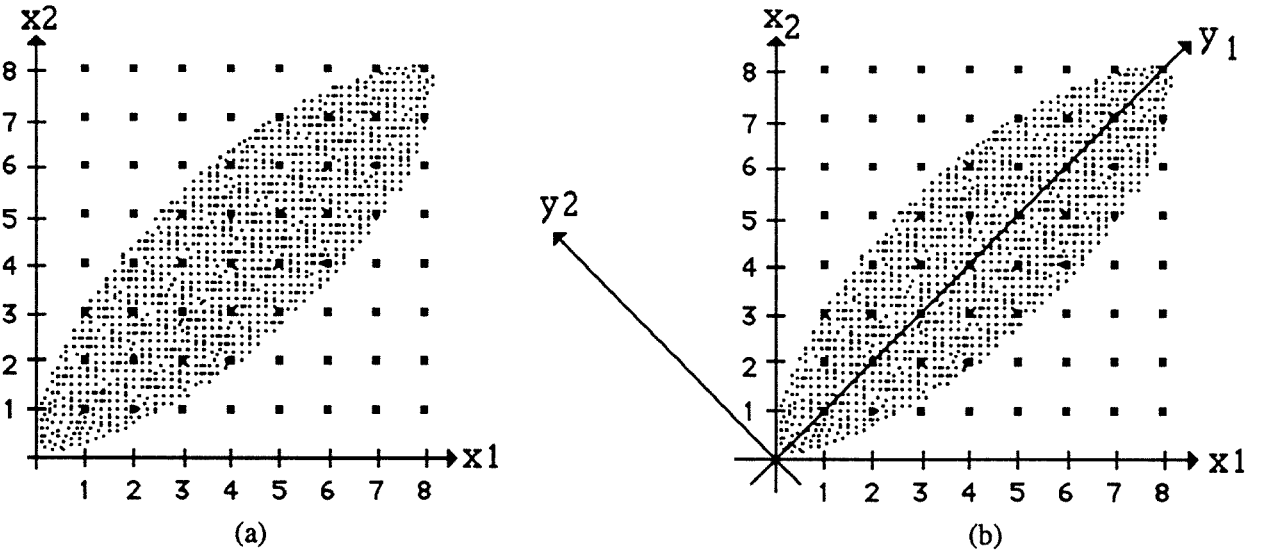


Figura 2.2 - (a) Pixels correlacionados. (b) Novo sistema de coordenadas para eliminar a correlação.

Suponha-se que o sistema de coordenadas é rodado, como se mostra na figura 2.2(b). No novo sistema de coordenadas os valores mais prováveis não estão na vizinhança de $y_1=y_2$ mas estão alinhados com o eixo y_1. Assim, as variáveis y_1 e y_2 são mais

independentes do que eram x_1 e x_2. A rotação do sistema de coordenadas também alterou a distribuição das variâncias. O total é o mesmo, $\sigma^2 y_1 + \sigma^2 y_2 = \sigma^2 x_1 + \sigma^2 x_2$, mas enquanto os dois elementos iniciais tinham a mesma variância ($\sigma^2 x_1 = \sigma^2 x_2$) a maior parte da variância total está agora na primeira componente do espaço transformado ($\sigma^2 y_1 > \sigma^2 y_2$). Finalmente note-se que, dados os coeficientes y_1 e y_2, pode efectuar-se a rotação inversa para obter os pixels x_1 e x_2.

O objectivo da transformada é reduzir a correlação entre pixels e melhorar a eficiência da codificação processando os coeficientes transformados independentemente uns dos outros. Na recepção, os bits recebidos são descodificados em coeficientes da transformada, aplicando-se em seguida uma transformação inversa para recuperar as intensidades dos elementos da imagem. Para comprimir podem-se desprezar, na transmissão, coeficientes que são pequenos e quantificar os outros do modo requerido pela qualidade da imagem.

A transformação de uma imagem exige um enorme esforço computacional. Habibi e Wintz [HAB71a] mostraram que, dividindo a imagem em blocos e aplicando a transformada a cada bloco, também se obtém compressão. Além disso, a transformação por blocos pode ser efectuada muito mais rapidamente pois exige menos esforço computacional e permite uma implementação com elevado grau de paralelismo. Assim, na codificação por transformada, a imagem é dividida em sub-imagens e depois cada uma dessas sub-imagens é transformada num conjunto de coeficientes mais independentes. Os coeficientes são então quantificados e codificados separadamente. A transformação e codificação isolada de cada sub-imagem despreza as redundâncias existentes entre estas, e assim, numa base puramente estatística, é vantajoso ter uma sub-imagem de grandes dimensões. No entanto, para simplificar a implementação e para explorar variações locais nas estatísticas da imagem, usa-se uma sub-imagem mais pequena.

Os parâmetros que determinam o desempenho de um codificador por transformada são a dimensão e forma das sub-imagens, o tipo de transformação usada, a selecção dos coeficientes a ser transmitidos e sua quantificação.

TRANSFORMAÇÕES

O objectivo da transformação é converter elementos da imagem estatisticamente dependentes em coeficientes, de algum modo, independentes. A maioria das transformações que são usadas são lineares e unitárias. Por razões práticas, uma

consideração importante é que, uma vez que a maior parte da compressão resulta de ignorar os coeficientes com pouca energia, a transformada compacte a maior percentagem da energia da imagem no menor número possível de coeficientes. Outro aspecto relevante, na prática, é a facilidade de executar a transformada.

TRANSFORMADA OPTIMA

Uma transformada verdadeiramente óptima resultará na melhor qualidade de imagem usando o menor número de bits, mas este critério é difícil de especificar quantitativamente. Um critério mais simples é requerer que os coeficientes da transformada sejam estatisticamente independentes, mas isto implica conhecimento de estatísticas das imagens de ordem superior a 2. Trata-se pois de procurar uma transformação que resulte em coeficientes não-correlacionados. Considerando os pixels numa imagem de dimensões $N \times N$ como um vector X de N^2 componentes, queremos uma transformada A na forma de uma matriz $N^2 \times N^2$ que resulte num vector de coeficientes Y não-correlacionado tendo N^2 componentes, isto é,

$$Y = A\ X \tag{2.5}$$

Uma vez que grande parte da compressão conseguida com a técnica das transformadas resulta de se ignorar alguns dos coeficientes, pretende-se que, na reconstrução de X, o erro quadrático médio devido a desprezar alguns coeficientes seja mínimo. Para isso procura-se que a energia da imagem seja compactada no menor número possível de coeficientes.

Existe uma transformada, conhecida como transformada óptima, que satisfaz todos estes critérios, e que é denominada transformada de Hotelling ou transformada de Karhunen-Loeve (KLT) [HAB71a]. Esta produz coeficientes não-correlacionados, minimiza o erro quadrático médio e concentra a máxima variância nos primeiros K coeficientes (para qualquer K).

A transformada óptima pode ser calculada a partir da covariância do vector de pixels X:

$$C_X = \mathrm{E}\ \{\ (X - \mathrm{E}(X))\ *\ (X - \mathrm{E}(X))^{\mathrm{T}}\ \}^2 \tag{2.6}$$

As linhas da matriz óptima A são os vectores próprios normalizados da matriz C_X, isto é, são soluções da equação

$$C_X X = \lambda_i X \tag{2.7}$$

Os coeficientes Y obtidos por esta transformação, têm uma matriz de covariância dada por

$$C_Y = \begin{bmatrix} \lambda_1 & 0 & 0 & \ldots & 0 \\ 0 & \lambda_2 & 0 & \ldots & 0 \\ 0 & 0 & \lambda_3 & \ldots & 0 \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot \\ \cdot & \cdot & & \ldots & \lambda_{N^2} \end{bmatrix} \tag{2.8}$$

onde λ_1, ..., λ_{N^2} são os valores próprios da matriz C_X. Sendo C_X uma matriz de covariância, todos os seus valores próprios são não-negativos e, se os ordenarmos de acordo com as suas grandezas, a maior percentagem de energia é compactada nos primeiros K coeficientes ($K < N^2$) que correspondem aos vectores próprios dos K maiores valores próprios. Se apenas estes K coeficientes forem enviados, então no receptor haverá um erro de reconstrução, cujo valor quadrático médio é dado por

$$e = \sum_{j=1}^{N^2} \lambda_j - \sum_{j=1}^{K} \lambda_j = \sum_{j=K+1}^{N^2} \lambda_j \tag{2.9}$$

Este é geralmente pequeno, uma vez que apenas inclui os valores próprios mais pequenos.

Embora a transformação óptima seja conhecida, o seu uso na prática não se torna viável uma vez que requer um conhecimento estatístico do sinal e exige um esforço de cálculo considerável. No entanto, esta é usada como referência para avaliação do desempenho de outras transformadas.

TRANSFORMADAS SUBOPTIMAS

Embora a KLT seja a melhor transformação quer em termos de erro quadrático médio quer do ponto de vista da qualidade subjectiva, normalmente usam-se transformadas ortonormais pois são mais fáceis de implementar além de terem a propriedade de preservar a energia e a informação do sinal original [AND68], produzindo

coeficientes menos correlacionados do que os pixels da imagem original. Têm ainda a vantagem de apresentarem uma transformação inversa facilmente definida.

Considerando os pixels numa imagem de dimensões NxN, estas transformadas podem ser definidas por

$$Y(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X(i,j)\ A(i,j,k,l) \qquad (2.10)$$

onde $X(i,j)$ é a matriz NxN dos pixels originais da imagem, $Y(k,l)$ a matriz NxN dos coeficientes transformados e $A(i,j,k,l)$ representa o núcleo da transformada directa.

A transformada inversa, que faz o mapeamento dos dados do domínio da transformada para o domínio espacial da imagem, é definida por

$$X(i,j) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} Y(k,l)\ B(i,j,k,l) \qquad (2.11)$$

onde $B(i,j,k,l)$ representa o núcleo da transformada inversa.

A transformação é dita separável se os seus núcleos podem ser escritos na forma

$$\begin{aligned} A(i,j,k,l) &= A_C(i,k)\ A_L(j,l) \\ B(i,j,k,l) &= B_C(i,k)\ B_L(j,l) \end{aligned} \qquad (2.12)$$

onde os índices dos núcleos indicam transformadas unidimensionais Linha e Coluna. Uma transformada bidimensional separável pode ser calculada em dois passos. Primeiro, é efectuada uma transformada unidimensional ao longo de cada coluna da imagem resultando

$$P(k,j) = \sum_{i=0}^{N-1} X(i,j)\ A_C(i,k) \qquad (2.13)$$

A seguir, é efectuada uma segunda transformada unidimensional ao longo de cada linha de P resultando

$$Y(k,l) = \sum_{j=0}^{N-1} P(k,j) \; A_L(j,l) \qquad (2.14)$$

Em notação matricial estas transformadas podem-se exprimir da seguinte forma

$$\boldsymbol{Y} = \boldsymbol{A}\,\boldsymbol{X} \qquad \text{transformada directa} \qquad (2.15)$$
$$\boldsymbol{X} = \boldsymbol{B}\,\boldsymbol{Y} \qquad \text{transformada inversa}$$

e

$$\boldsymbol{B} = \boldsymbol{A}^{-1} \qquad (2.16)$$

Sendo estas transformadas unitárias, $A^{T*} = A^{-1}$ (sendo A^{T*} a conjugada complexa de A^T) e, assim, a transformação inversa é tão fácil de implementar como a própria transformação.

Uma maneira de visualizar a transformação de uma imagem é considerar a equação (2.11) como um meio de sintetizar uma imagem a partir de um conjunto de funções matemáticas $B(i,j,k,l)$, para um sistema de coordenadas fixo no domínio da transformada (k,l). Nesta interpretação, o núcleo $B(i,j,k,l)$ é chamado uma função base bidimensional e os coeficientes da transformada $Y(k,l)$ são as amplitudes das funções base requeridas na síntese da imagem.

Uma das transformadas mais divulgadas, e a primeira a ser usada na codificação e transmissão de imagem, é a Transformada Discreta de Fourier (DFT) [AND68a]. A sua popularidade, em todas as áreas do processamento de sinal, advém não só das suas características mas também, e especialmente, do aparecimento da transformada rápida de Fourier (FFT). Consequentemente, surgiram numerosos algoritmos capazes de diminuir significativamente o seu grau de complexidade [BLA85].

Os núcleos das transformadas de Fourier directa e inversa são

$$A(i,j,k,l) = 1/N \quad \exp\{ -j2\pi(ki+lj)/N \} \qquad , j = \sqrt{-1} \qquad (2.17)$$
$$B(i,j,k,l) = 1/N \quad \exp\{ j2\pi(ki+lj)/N \}$$

Esta transformada também permite concentrar a energia do sinal em poucos coeficientes. No entanto, tem duas desvantagens: exige cálculos com aritmética complexa, pois conduz ao mapeamento de funções reais num espaço complexo, e tem uma taxa de convergência lenta. Esta última desvantagem, significativa em

aplicações de codificação de imagem, resulta das descontinuidades abruptas nas zonas de fronteira da imagem que provocam, no domínio de Fourier, componentes de grande amplitude às altas frequências (fenómeno de Gibbs). A este fenómeno dá-se o nome de "efeito de bloco", pois torna os contornos do bloco muito visíveis, especialmente quando o factor de compressão é elevado, isto é, quando se ignoram muitos dos coeficientes associados às altas frequências.

Posteriormente à aplicação da DFT na codificação de imagem, foram encontradas outras transformadas que apresentam vantagens do ponto de vista computacional ou de desempenho. Algumas dessas transformadas são: Slant, Haar, Hadamard e Transformada Discreta de Cosseno (DCT) [PRA78].

A Transformada Discreta de Cosseno [AHM74] é largamente aceite como a melhor transformação sub-óptima para codificação de imagem. Demonstrou-se [CLA81] que, em certas condições aceites na prática (coeficiente de correlação entre pixels adjacentes próximo de 1), a DCT aproxima a KLT. Há mesmo autores [TES79] que consideram ser, de entre todas as transformadas determinísticas, a que apresenta melhor desempenho.

O núcleo da Transformada Discreta de Cosseno directa pode ser definido como

$$\begin{aligned} A(i,j,0,0) &= 1/N \\ A(i,j,k,l) &= \cos[\pi(2i+1)k]\ \cos[\pi(2j+1)l] / 2N^3 \qquad k,\ l \neq 0 \end{aligned} \tag{2.18}$$

tendo o núcleo da transformada inversa a mesma forma.

A DCT reduz consideravelmente o ruído de bloco presente na DFT, forçando a simetria da imagem transformada [CLA85]. Esta simetria imposta pela DCT é também responsável pela sua superior capacidade de compactação. Para a maioria das imagens reais, de entre todas as transformadas, a DCT é a que se aproxima mais da KLT nas suas propriedades de compactação de energia.

Embora seja possível implementar a DCT com algoritmos FFT, foi demonstrado [CHE77] que também é possível escrever directamente um algoritmo rápido para a transformada de Cosseno (mais rápido do que os que usam FFT). Outros algoritmos rápidos para implementação da DCT têm sido propostos [LEE84] [HAQ85].

Muito recentemente surgiu um novo conceito de transformada: Transformada Ortogonal Estendida (LOT) [MAL89]. Esta nova famíla de transformadas superioriza-se em relação à DCT, especialmente na imunidade ao ruído de bloco.

DIMENSÃO DOS BLOCOS

Em codificação por transformada, inicialmente subdivide-se a imagem original num determinado número de sub-imagens (blocos), geralmente quadradas, e codifica-se cada sub-imagem como uma unidade, independentemente de todas as outras sub-imagens. Assim, a escolha da dimensão do bloco é uma consideração prática relevante.

A codificação por transformada é mais fácil de implementar se forem usados blocos de pequenas dimensões. No entanto, é importante que os coeficientes transformados sejam o menos correlacionados possível. Embora a transformação por blocos descorrelacione os coeficientes dentro dum mesmo bloco, há sempre alguma correlação entre os coeficientes dos blocos vizinhos pois, para blocos adjacentes, os pixels de cada bloco próximos da fronteira comum estarão correlacionados. Em geral, quanto maiores forem os blocos, menores são as correlações entre coeficientes interblocos. Por outro lado, para uma melhor adaptação à estrutura local da imagem, é preferível um bloco mais pequeno.

Simulações em computador sobre imagens reais [NET80] mostram que o erro quadrático médio resultante da codificação por transformada (para uma dada taxa de transmissão) melhora com o aumento do tamanho da sub-imagem. No entanto, esta melhoria não é significativa quando o tamanho da sub-imagem é aumentado para além de 16x16 pixels. A penalidade em erro quadrático médio usando um bloco de 8x8 pixels é insignificante comparada com blocos de 16x16 ou maiores. No entanto, a qualidade subjectiva das imagens parece não melhorar com o aumento do tamanho do bloco para além de 4x4 [WIN72], embora alguns autores [TES79] refiram ganhos significativos, em termos de desempenho, usando blocos de maiores dimensões (256x256).

Também, para imagens reais, o desempenho das transformadas KLT, Hadamard e Fourier é aproximadamente o mesmo para sub-imagens de 4x4; mas para blocos 8x8 e 16x16, a KLT é melhor do que a Fourier que, por sua vez, é melhor que a Hadamard. A DCT aproxima-se bastante da KLT, para qualquer tamanho de bloco [NET80]. Verificou-se que a DCT e a KLT apresentam erros idênticos para um modelo Gauss-Markov de primeira ordem [CAN86] de blocos de pequena dimensão (8x8 a 16x16 pixels).

A codificação por blocos tem a desvantagem de apresentar ruído de bloco, semelhante ao fenómeno de Gibbs observado com a DFT. Este ruído será tanto mais visível quanto maior for o factor de compressão, ou seja, quanto menor for o número de coeficientes transmitidos. Vários esquemas têm sido propostos para tentar ultrapassar este problema como, por exemplo, a sobreposição de blocos [CHE83].

SELECÇÃO E QUANTIFICAÇÃO DOS COEFICIENTES

Uma vez que, em codificação por transformada, a compressão é conseguida não transmitindo todos os coeficientes, após o cálculo da transformada, é necessário seleccionar, e quantificar, os coeficientes a transmitir. Na descodificação, os coeficientes não transmitidos são colocados a zero. Existem duas estratégias básicas para a selecção dos coeficientes: filtragem zonal ou "threshold" [WIN72].

Com a técnica de filtragem zonal, o critério para a escolha dos coeficientes é normalmente baseado na variância dos coeficientes. Estudos analíticos e experimentais [WIN72] mostraram que a zona óptima é a chamada "zona de variância máxima". Assim, estima-se a variância dos coeficientes, com base num conjunto de blocos típicos, e escolhem-se os K coeficientes de maior variância para serem transmitidos.

Com a técnica designada "threshold" em vez de transmitir os K coeficientes de maior variância seleccionam-se os K coeficientes de maior amplitude. Define-se um limiar e transmitem-se todos os coeficientes cuja amplitude excede esse limiar. Alterando o limiar pode controlar-se o número de coeficientes seleccionados e, consequentemente, a taxa de transmissão [JAI81]. Esta é uma técnica adaptativa enquanto a filtragem zonal é fixa.

Como os coeficientes associados às baixas frequências são, para a maioria das imagens, os que concentram mais energia, aplicando uma filtragem zonal são estes que são seleccionados. Assim, esta técnica é equivalente a uma filtragem passa-baixo da imagem. Por outro lado, a selecção por "threshold" é sensível aos coeficientes mais importantes, independentemente da sua posição. Consequentemente, esta técnica tem demonstrado um desempenho superior à filtragem zonal, em termos de qualidade da imagem reconstruída, para o mesmo número de coeficientes transmitidos. A selecção por "threshold" tem o inconveniente de requerer a transmissão adicional de informação, relativa à posição de cada coeficiente transmitido. No entanto, pode-se

minimizar este problema, fazendo um varrimento em zig-zag para seleccionar os coeficientes a enviar, iniciando-se na componente DC e prosseguindo pelas frequências sucessivamente superiores (figura 2.3). Uma vez que os coeficientes numa subdiagonal têm aproximadamente a mesma importância relativa, um varrimento em zig-zag resulta aproximadamente num conjunto de elementos com valores decrescentes. Aplicando uma codificação "Run-Length", consegue-se diminuir este "overhead" para proporções pouco significativas [COS89].

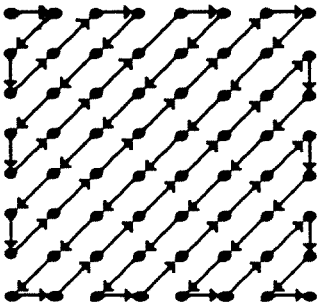


Figura 2.3 - Varrimento em zig-zag dos coeficientes resultantes da transformada.

Definidos os coeficientes a transmitir é necessário decidir o número de bits a atribuir a cada um. Tanto o erro quadrático médio como a qualidade subjectiva são bastante sensíveis à eficiência com que os bits são usados para codificar os coeficientes. A figura 2.4 ilustra uma atribuição de bits típica para codificar blocos de 16x16 pixels.

7	6	5	4	3	3	2	2	2	1	1	1	1	1	0	0
6	5	4	4	3	3	2	2	1	1	1	1	1	0	0	0
5	4	4	3	3	2	2	2	1	1	1	1	1	0	0	0
4	4	3	3	3	2	2	2	1	1	1	1	1	0	0	0
3	3	3	3	2	2	2	1	1	1	1	1	0	0	0	0
3	3	2	2	2	2	2	1	1	1	1	1	0	0	0	0
2	2	2	2	2	2	1	1	1	1	1	0	0	0	0	0
2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 2.4 - Atribuição de bits típica, usando filtragem zonal, em blocos de 16x16 pixels e usando DCT, para uma taxa de transmissão de aproximadamente 1 bit/pixel [JAI81].

Sendo as variâncias dos coeficientes bastante diferentes, seria pouco eficiente usar o mesmo quantificador para todos eles. Por outras palavras, se os níveis de quantificação forem ajustados para a gama dos coeficientes de maior variância, então os coeficientes de muito menor variância irão situar-se numa gama muito menor e, nesse caso, a maior parte dos níveis de quantificação não seria usada. Uma vez que, geralmente, os coeficientes com maiores variâncias têm uma contribuição significativamente maior para a imagem reconstruída, a distorção total devida à quantificação pode ser minorada atribuindo mais níveis de quantificação aos coeficientes com maiores variâncias e menos aos coeficientes com variâncias menores. Além disso, como cada coeficiente corresponde a uma determinada banda de frequências e a sensibilidade do sistema visual humano à distorção depende da frequência das distorções, em alguns casos pode obter-se uma melhor qualidade subjectiva atribuindo mais níveis de quantificação aos coeficientes correspondentes às frequências a que o olho humano é mais sensível.

O dimensionamento de um quantificador que minimize o erro quadrático médio na reconstrução da imagem, para um dado número de níveis de quantificação, exige o conhecimento da função densidade de probabilidade dos dados a quantificar [MAX60]. Várias distribuições diferentes têm sido assumidas para os coeficientes [PRA78] [TES79] [REI83]. Resultados de testes [REI83] mostraram que, para a maioria das imagens, e usando a DCT, as estatísticas dos coeficientes são melhor aproximadas por uma distribuição Gaussiana para o coeficiente DC e uma distribuição Laplaciana para os coeficientes AC. No entanto, salientam que algumas imagens (por exemplo, imagens aéreas) são melhor representadas por estatísticas Gaussianas.

EFEITOS DOS ERROS DE TRANSMISSÃO

Uma das vantagens dos codificadores de transformada não adaptativos, é que o efeito de bits errados na transmissão não se propaga para fora do bloco. Se, na recepção, um coeficiente é descodificado erradamente, devido a um erro de transmissão, ao calcular a transformada inversa só os pixels no mesmo bloco são afectados. A degradação resultante depende da transformada usada e do coeficiente errado. Em geral, erros em coeficientes de baixas frequências são mais visíveis do que em coeficientes de altas frequências, devido ao facto de, acima de certa frequência, a sensibilidade do olho humano decrescer. A probabilidade de erros nos coeficientes de baixas frequências é maior devido ao maior número de bits requeridos para os codificar. À medida que o tamanho do bloco decresce o espalhamento do erro pelo bloco decresce, e os erros de transmissão aparecem como pequenas manchas na imagem.

TRANSFORMADAS ADAPTATIVAS

A adaptatividade em codificadores por transformada pode melhorar significativamente a eficiência da codificação, uma vez que as estatísticas da imagem podem ser altamente não-estacionárias. Dois tipos de adaptação são possíveis: as alterações nos parâmetros são baseadas apenas em informação previamente transmitida e, como tal, disponível no descodificador, ou as alterações são baseadas também em dados não disponíveis no codificador. Este último caso, implica a transmissão de informação adicional.

Quase todos os parâmetros dos codificadores de transformada têm sido adaptados às estatísticas locais da imagem [HAB77]. Vários tipos de abordagens têm sido feitas para introduzir o conceito de adaptatividade nas transformadas, no entanto, a maioria revela-se demasiado complexa para ser implementada na prática. Além disso, não é fácil distinguir as vantagens decorrentes da transformada adaptativa das vantagens devidas à quantificação adaptativa. Parece que a maioria das vantagens da adaptação podem ser conseguidas usando uma transformada unitária simples, seguida de selecção adaptativa dos coeficientes a ser transmitidos e quantificação adaptativa. Esta abordagem é a mais usada [NET80].

A codificação por transformada adaptativa pode resultar na propagação dos efeitos de erros de transmissão, se não houver o cuidado de não usar informação de um bloco para o seguinte, quer para adaptação quer para descodificação. Em geral, se cada sub-imagem puder gerar um número fixo de bits e a adaptação for feita usando informação pertencente à sub-imagem, então os efeitos de erros de transmissão podem ser limitados às sub-imagens, mas isto pode comprometer a eficiência da codificação, especialmente para blocos de pequenas dimensões.

2.4 CODIFICAÇÃO HIBRIDA

Geralmente, entende-se por codificação híbrida aquela que utiliza simultaneamente DPCM e transformada. No entanto, outros métodos podem ser combinados, do que resulta também uma codificação híbrida. O objectivo é obter uma técnica pouco complexa mas que permita uma elevada taxa de compactação. Em termos de desempenho esta técnica situa-se entre o DPCM e as transformadas, em relação à complexidade e à compactação.

Nesta técnica usam-se blocos de pequenas dimensões, calculam-se os coeficientes aos quais se aplica DPCM, usando coeficientes dos blocos previamente transmitidos como estimativas. Têm sido considerados três esquemas [ROE77]:

a) Aplicar uma transformada a blocos unidimensionais compostos por elementos de uma linha e DPCM ao longo das colunas.

b) Aplicar uma transformada a blocos bidimensionais e DPCM entre coeficientes de blocos consecutivos.

c) Aplicar uma transformada a blocos bidimensionais e DPCM na dimensão temporal.

Os dois primeiros métodos, introduzidos pela primeira vez por Habibi em 1974 [HAB74], usam coeficientes de blocos relacionados espacialmente e são designados por métodos "intraframe". No primeiro, a correlação espacial na direcção horizontal é explorada usando a técnica das transformadas e na direcção vertical usando DPCM. O segundo, é essencialmente um sistema de codificação por transformada bidimensional onde a eficiência da codificação é melhorada explorando a correlação interbloco presente nos dados. Foi mostrado que se podem usar blocos de pequenas dimensões, reduzindo assim a complexidade de implementação associada às transformadas, sem afectar o desempenho do sistema.

O terceiro método, proposto por Roese [ROE77], usa blocos adjacentes no tempo, designando-se método "interframe". Pearlman [PEA84] demonstrou que esta técnica é superior à codificação por transformada tridimensional, quer em termos de relação sinal/ruído quer em termos de qualidade subjectiva das imagens reconstruídas.

Em codificação híbrida também se podem usar esquemas adaptativos que ofereçam um compromisso razoável entre desempenho e complexidade [JAI81] [MUS85].

2.5 QUANTIFICAÇÃO VECTORIAL

A quantificação vectorial [GRA84] consiste basicamente em fazer corresponder a um vector de dados um outro vector de uma tabela pré-definida e utilizar o respectivo endereço como código. O codificador por quantificação vectorial atribui a cada vector da fonte o vector da tabela de códigos mais próximo, baseado numa determinada medida de distorção. A descodificação consiste em ler, numa tabela idêntica à do codificador, o vector correspondente ao código especificado, e que representa os

dados originais. Assim, o ponto principal deste método é desenvolver uma "boa" tabela de códigos de vectores representativos, típicos dos dados que vão ser codificados.

Um codificador por quantificação vectorial consiste numa tabela $c = \{ y_i;\ i=1,2,...,N\}$ de N palavras de código em conjunto com uma função de mapeamento $q(.)$ que atribui a cada vector de entrada x, real, K-dimensional, uma palavra de código $y_i \in c$. As palavras de código y devem ser semelhantes aos vectores x da fonte, isto é, são também vectores reais, K-dimensionais. A dimensão K dos vectores e o tamanho N da tabela determinam o factor de compressão.

A quantificação vectorial pode ser vista como a combinação de duas funções (cf. figura 2.5): um codificador, que observa um vector de dados de entrada x e gera o endereço i da palavra de código y_i especificada por $q(x)$, e um descodificador que usa este endereço para gerar o vector reproduzido y_i. Quando x é quantificado como y_i, resulta um erro de quantificação em relação a uma medida de distorção $d(x, y_i)$. A medida de distorção representa o custo associado a reproduzir x como y_i. Assim, o melhor mapeamento $q(.)$ é aquele que minimiza $d(x, y_i)$ [GER82].

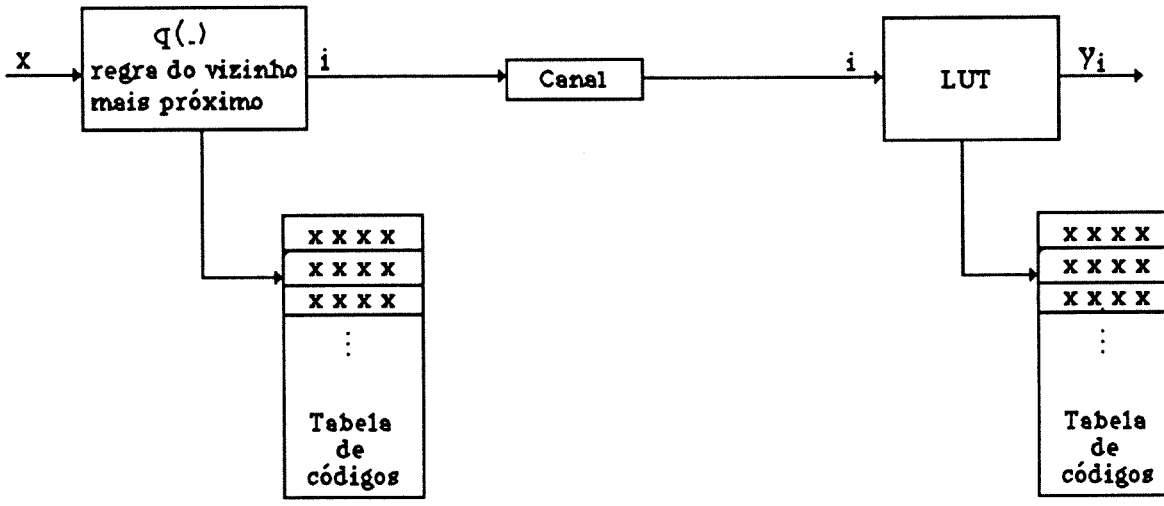


Figura 2.5 - Diagrama de blocos de um Quantificador Vectorial simples.

Definido um critério de distorção, o desempenho de um codificador por quantificação vectorial é medido pela distorção média por pixel. Para construir uma tabela de códigos é necessário realizar a partição do espaço K-dimensional do vector x em N regiões distintas, a que chamaremos células. Associado a cada célula s_i, o quantificador atribui um vector y_i como palavra de código se x pertence a s_i. A medida de distorção mais utilizada na codificação é o erro quadrático médio,

$$d(x, y_i) = (1/K) \sum_{j=1}^{K} (x_j - y_{ij})^2 \qquad (2.19)$$

em que x é o vector de entrada, y_i é o elemento i da tabela de códigos e K é a dimensão dos vectores. Outras medidas de distorção têm sido propostas, no entanto são, geralmente, demasiado pesadas computacionalmente para serem implementadas na prática.

O número de elementos da tabela de códigos e a dimensão dos vectores são parâmetros críticos que determinam a complexidade de codificação necessária para executar a busca através da tabela, a memória necessária para guardar a tabela, quer no codificador quer no descodificador, e a taxa de transmissão.

Se N for o número de vectores da tabela, K a dimensão dos vectores, e r a taxa de transmissão, em bit/amostra, estes três parâmetros relacionam-se por:

$$r = \log_2 N / K \qquad \text{(bit/amostra)} \qquad (2.20)$$

É evidente que a compactação cresce com a dimensão do vector.

Segundo a "Rate Distortion Theory" de Shannon, melhores resultados podem ser obtidos tratando vectores em vez de escalares [GAL68]. Apesar de algumas técnicas tradicionais de compressão, como por exemplo as transformadas, operarem em vectores, a quantificação é sempre efectuada sobre escalares. Neste sentido, estes sistemas são inerentemente subóptimos: melhores resultados podem ser sempre conseguidos, em teoria, codificando vectores em vez de escalares, mesmo que estes tenham sido previamente processados de modo a torná-los não-correlacionados ou independentes. Assim, em teoria, para uma dada distorção média admissível podemos atingir, com a quantificação vectorial, taxas tão próximas quanto se queira da taxa mínima, à custa do crescimento da dimensão do vector de dados.

O principal obstáculo ao uso de vectores de grande dimensão, é o crescimento exponencial da complexidade de codificação com a dimensão, para uma dada taxa. Um quantificador vectorial de dimensão K, para uma taxa de r bit/amostra, requer uma tabela de códigos de dimensão $N=2^{Kr}$ elementos e, consequentemente, uma memória de $K2^{Kr}$ palavras. De facto, quanto maior for a tabela e quanto maior for a dimensão do

vector, mais memória é necessária para o seu armazenamento e mais complexa se torna a operação de busca.

Esta limitação tem condicionado a dimensão dos vectores de dados tendo, no entanto, sido propostas diversas técnicas para ultrapassar este problema da complexidade. Por exemplo, em [BUZ80] é proposto um codificador com busca em árvore (TSVQ) para reduzir o esforço de busca num quantificador vectorial. Um TSVQ pode ser entendido como um codificador que busca uma sequência de pequenas tabelas de códigos, em vez de uma tabela mais extensa. A estrutura do quantificador pode ser vista como uma árvore onde cada busca e decisão corresponde a avançar um nível na árvore, começando pela raíz. As desvantagens destes codificadores são o requisito de mais memória para guardar as tabelas de códigos e, geralmente, as palavras de código seleccionadas não serem óptimas no sentido de minimizar a medida de distorção $d(x, y_i)$. Algumas outras técnicas que reduzem a complexidade podem ser encontradas em [JUA82] [FIS86] [TSE87] [KAO87] [EQU87] [KOH88] e [PAL89]. Em geral, permitem usar vectores de maior dimensão, para a mesma complexidade, mas, normalmente, levam a resultados inferiores aos que seriam conseguidos com uma tabela de códigos óptima e uma busca exaustiva, para as mesmas dimensão e taxa de transmissão.

Um problema central no projecto de um quantificador vectorial é o da obtenção da tabela de códigos. No caso geral, pretende-se determinar uma tabela para codificar imagens para as quais não temos qualquer modelo probabilístico. O método mais usado para gerar uma tabela de códigos tem sido o algoritmo conhecido por "algoritmo LBG" (Linde-Buzo-Gray) [LIN80]. Este algoritmo também é por vezes referido como "algoritmo de Lloyd generalizado" (GLA) uma vez que é uma generalização de um algoritmo devido a Lloyd [LLO82].

O algoritmo LBG é iterativo e pode ser descrito como se segue. Com base num conjunto de vectores de treino, de uma tabela de códigos inicial, de uma medida de distorção e de um critério de paragem, executam-se as seguintes operações:

1 - Atribuir cada um dos vectores de treino ao elemento da tabela de códigos mais próximo, com base na medida de distorção.

2 - Aplicar o critério de paragem à variação da distorção média, relativamente à iteração anterior. Caso se verifique, termina.

3- Actualizar cada elemento da tabela de códigos, substituindo-o pelo centróide dos vectores de treino que lhe estão atribuídos. Voltar a 1.

O tempo de execução deste algoritmo é incerto, pois o número de iterações necessário não pode ser previsto "à priori". A experiência indica que o tempo de execução aumenta rapidamente com o crescimento da sequência de vectores de treino, com o aumento do número de palavras de código e com o crescimento da dimensão do vector.

Neste algoritmo a tarefa mais difícil é escolher uma tabela de códigos inicial aceitável. Uma vez que o algoritmo LBG só garante encontrar mínimos locais, é importante que se comece na vizinhança da solução correcta, senão o algoritmo pode convergir para um mínimo local distante do mínimo global. Neste caso, a tabela de códigos gerada pode oferecer um desempenho muito pobre [GRA82]. Uma possibilidade para inicializar o algoritmo LBG é usar uma tabela de códigos previamente desenvolvida para outros fins. Uma alternativa é inicializar com palavras de código igualmente espaçadas, no espaço dos vectores, ou usar a técnica chamada "splitting" [LIN80]. No entanto, a inicialização mais simples talvez seja escolher aleatoriamente uma amostragem da sequência de treino para ser usada como códigos iniciais.

Aplicada à codificação de imagens, a quantificação vectorial é efectuada sobre pequenos blocos rectangulares, isto é, os vectores são sub-blocos bidimensionais da imagem, tipicamente vectores de 16 elementos representando blocos de imagem de 4x4 pixels.

Como consequência da codificação por blocos, especialmente blocos de pequenas dimensões, resulta uma degradação significativa da qualidade subjectiva da imagem codificada, que apresenta o chamado "efeito de bloco", sendo este tanto maior quanto maior for a compactação. Outra limitação geralmente associada aos codificadores por quantificação vectorial é o deficiente tratamento dos contornos, que leva também a uma degradação da qualidade subjectiva da imagem. De facto, esta limitação não é uma característica do codificador, mas sim da medida de distorção normalmente usada — o erro quadrático médio. Têm sido propostos esquemas para solucionar este problema como, por exemplo, o CVQ - "Classified Vector Quantizer" [GER82a] [RAM84] [RAM86] onde cada vector da imagem é classificado numa de várias categorias e existe uma tabela de códigos diferente para codificar os vectores pertencentes a cada categoria.

A combinação da quantificação vectorial com codificadores convencionais resulta num codificador mais eficiente do que um codificador convencional usado com quantificação escalar. Assim, um quantificador vectorial preditivo (PVQ) interframe [HAN85] pode explorar a redundância existente no sinal de erro de predição obtido de

duas frames consecutivas. A quantificação vectorial no domínio das transformadas (TVQ) [KIN83] oferecerá um desempenho superior à de um codificador híbrido convencional (transformada/DPCM) pois os codificadores DPCM têm um fraco desempenho na presença de ruído e a correlação e dependência do sinal de erro não é totalmente explorada pela quantificação escalar. Além disso, os quantificadores escalares são forçados a atribuir um número inteiro de bits a cada coeficiente, enquanto que a quantificação vectorial permite uma alocação mais distribuída. Aliás, uma das vantagens da quantificação vectorial relativamente à quantificação escalar é permitir atribuir um número fraccionário de bits por amostra. Enquanto que a quantificação escalar necessita de, pelo menos, 1 bit/amostra, a quantificação vectorial K-dimensional permite uma taxa de apenas $1/K$ bit/amostra, no caso de ter apenas um símbolo binário para vectores de entrada K-dimensionais.

Nos últimos anos as técnicas híbridas, especialmente a quantificação vectorial no domínio das transformadas, têm merecido grande atenção. Algumas outras áreas importantes de investigação são a aplicação da quantificação vectorial à codificação de sequências de imagens [GOL86a] [GOL88] [SHE88], a utilização de técnicas adaptativas [GOL86] [AIZ86] e a incorporação de critérios de percepção visual [THY85] [RAM88].

Nasrabadi e King [NAS88] publicaram um excelente resumo das técnicas de Quantificação Vectorial aplicadas em codificação de imagem.

2.6 CODIFICAÇÃO INTERPOLATIVA

Na codificação por interpolação um subconjunto de pixels é transmitido e os restantes pixels são interpolados na recepção. A figura 2.6 mostra um exemplo em que existe uma subamostragem horizontal de pixels de 2:1, ao longo de cada linha. Os elementos subamostrados são alternados de uma linha para a seguinte e são interpolados por uma média de 4 vizinhos.

De um modo geral, as técnicas de codificação interpolativa podem ser divididas em duas classes: fixas e adaptativas. Em métodos de interpolação fixos, um conjunto fixo de elementos da imagem são seleccionados para transmissão e os restantes são interpolados. Vários tipos de amostras podem ser escolhidos para transmissão: por exemplo, pixels alternados, um pixel em cada quatro, linhas alternadas, frames

alternadas, etc.. A qualidade da imagem resultante é função do número e tipo de amostras que são ignoradas e do método de interpolação.

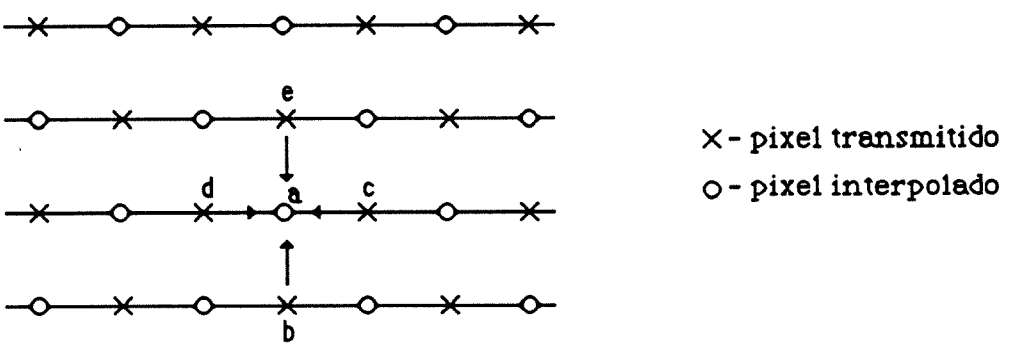


Figura 2.6 - Exemplo de interpolação.

A interpolação pode ser linear ou não. Pelos resultados conhecidos a interpolação linear parece ser eficiente, e o ganho obtido com o uso de polinómios de grau superior não é significativo.

A interpolação pode variar localmente com regras do tipo

$$\hat{a} = \begin{cases} 0.5\,(c + d) & \text{se } |c - d| \le |b - e| \\ 0.5\,(b + e) & \text{outros casos} \end{cases} \qquad (2.21)$$

Este tipo de interpolação varia entre médias horizontais, verticais, temporais ou noutra direcção, dependendo do tipo de correlação local existente nos dados.

A adaptatividade pode ser introduzida na escolha do número e da posição dos pixels a transmitir. A interpolação adaptativa consiste nos seguintes passos:

a) Escolha inicial dos pontos a transmitir;
b) Determinação do erro de interpolação;
c) Se o erro for inferior a um determinado limiar reduz-se o número de pixels a transmitir e volta-se a b); se o erro for superior a um determinado limiar aumenta-se o número de pixels a transmitir e volta-se a b).

O menor número de pixels necessários para manter o erro de interpolação abaixo do limiar é, geralmente, a característica desejada. Existem diversas variações dependendo do método de transmissão de pixels, da técnica de interpolação e do critério de qualidade.

O critério de qualidade usado tem sido, tradicionalmente, baseado no erro quadrático médio. No entanto, têm sido propostos critérios relacionados com a percepção visual [NET77b].

Também tem sido usada interpolação adaptativa bidimensional, o que permite uma maior redução do número de pixels transmitidos. Note-se, no entanto, que a adaptatividade exige a transmissão de informação complementar.

2.8 CODIFICAÇÃO DE SEQUENCIAS DE IMAGENS

Em sequências de imagens, como as encontradas na transmissão de vídeo, a informação "nova" está contida, normalmente, apenas num número relativamente pequeno de pixels. Frequentemente, uma grande percentagem de pixels em cada frame representam área de "fundo" e não variam significativamente de frame para frame. Tipicamente, os valores dos pixels de uma frame para a seguinte, numa posição (i, j) fixa, só diferem substancialmente nas áreas de movimento.

De um ponto de vista estatístico, a semelhança de valores dos pixels de frames consecutivas implica uma elevada correlação interframe entre os valores dos pixels temporalmente adjacentes. As técnicas de codificação referidas anteriormente, desenvolvidas para explorar a correlação espacial contida numa imagem, podem ser estendidas de modo a incluir a correlação interframe, conseguindo-se assim atingir taxas de compressão superiores. O único inconveniente da codificação interframe é a necessidade de armazenar sequências de frames.

Em codificação DPCM, a correlação temporal pode ser explorada usando filtros de predição tridimensionais, que estimam o valor de um pixel a partir de valores de pixels adjacentes espacialmente (na mesma frame) e temporalmente (na mesma posição espacial mas em frames anteriores).

A codificação por transformada tridimensional pode ser usada para descorrelacionar e compactar energia em cada frame e entre frames. Neste caso, a transformada é aplicada a blocos tridimensionais, compostos por pixels adjacentes numa frame e em frames anteriores.

No caso da codificação híbrida usa-se, vulgarmente, a técnica das transformadas para reduzir a correlação espacial e DPCM para reduzir a correlação temporal. Neste caso duas estruturas podem ser usadas:

1) Transformada/DPCM: Reduz a correlação espacial entre pixels e a correlação temporal entre coeficientes. Neste caso, aplica-se uma transformada a blocos bidimensionais da frame actual e DPCM na dimensão temporal, aos coeficientes resultantes.

2) DPCM/Transformada: Reduz a correlação temporal entre pixels e a correlação espacial do sinal de erro de predição. Neste caso, aplica-se DPCM na dimensão temporal aos pixels e uma transformada a blocos bidimensionais do sinal diferença.

Embora estas duas estruturas possam ser consideradas equivalentes, o uso de DPCM na dimensão temporal para reduzir a correlação entre pixels tem a grande vantagem de permitir o uso de técnicas de compensação de movimento que resultam num factor de compressão consideravelmente superior além de reduzirem significativamente o efeito de bloco [ERI85].

Note-se, no entanto, que em cenas com grande detalhe e muito movimento, filtros de predição baseados apenas na frame actual (intraframe) podem ser melhores do que filtros de predição baseados na diferença entre frames (interframe). Uma possível solução é seleccionar um filtro de predição interframe ou intraframe, dependendo de uma medida da quantidade de movimento.

A codificação interframe permite a utilização de técnicas que exploram a considerável redundância entre frames como, por exemplo, "Conditional Replenishment" e Compensação de Movimento.

"CONDITIONAL REPLENISHMENT"

Esta técnica [HAS79] é baseada num método simples de detecção e codificação das áreas de movimento que são actualizadas de uma frame para a seguinte. Só pixels cujo valor variou significativamente são processados. Se $u_{i,j,k}$ representar o pixel na posição (i,j) na frame k, então o sinal diferença interframe é

$$e_{i,j,k} = u_{i,j,k} - u'_{i,j,k-1} \tag{2.22}$$

em que $u'_{i,j,k-1}$ é o valor reproduzido do pixel $u_{i,j}$ na frame (k-1). Se a amplitude de $e_{i,j,k}$ é maior do que um pré-determinado limiar η, então esta é quantificada e

codificada para transmissão. No receptor, um pixel é reconstruído repetindo o valor do pixel correspondente da frame anterior, se estiver numa zona estacionária, ou somando a este o sinal diferença descodificado, se estiver numa área de movimento, isto é,

$$u'_{i,j,k} = \begin{cases} u'_{i,j,k-1} + e'_{i,j,k} & \text{se } |e'_{i,j,k}| > \eta \\ u'_{i,j,k-1} & \text{se } |e'_{i,j,k}| \leq \eta \end{cases} \qquad (2.23)$$

Para transmissão, são geradas palavras de código representando os valores quantificados e os seus endereços. Evidentemente, a taxa média conseguida dependerá da extensão e duração das áreas de movimento de modo que, para atingir uma taxa de transmissão estacionária, é necessário um "buffer" de saída razoavelmente dimensionado que absorva as variações. Para evitar o crescimento indiscriminado (overflow) do "buffer" terá que ser adoptada uma estratégia de controlo apropriada. Um esquema vulgar de controlo do "buffer" é aumentar o limiar η quando está iminente um "overflow".

Esta técnica pode ser aplicada a blocos de pixels. Assim, divide-se a imagem em blocos rectangulares de tamanho fixo, e só são processados blocos que variaram significativamente, em relação aos blocos correspondentes da frame anterior. Neste caso, também se pode chamar a esta técnica "block type descrimination".

COMPENSAÇÃO DE MOVIMENTO (MC)

As técnicas que utilizam predição têm uma estrutura local e, por isso, são bastante sensíveis a variações nos dados. O movimento em frames sucessivas torna as estatísticas interframe não-estacionárias. Podem-se conseguir ganhos consideráveis em termos de desempenho (melhoria da relação sinal/ruído, redução drástica da taxa de transmissão e melhoria significativa da qualidade subjectiva da imagem reconstruída, devido à redução considerável do efeito de bloco) fazendo adaptações no filtro de predição e no codificador para compensar as variações devidas ao movimento.

Os filtros de predição adaptativos mais bem sucedidos para codificação interframe são aqueles que têm em conta o movimento dos objectos. Estes são baseados na noção de que, se há objectos que se movem no campo de visão de uma câmara e se uma estimativa do seu movimento estiver disponível, então pode ser feita uma codificação preditiva mais eficiente tendo em conta diferenças de elementos em relação a

elementos da frame anterior que são apropriadamente deslocados espacialmente. Este tipo de predição chama-se predição com compensação de movimento. Os resultados obtidos com esta técnica dependem da quantidade e do tipo de movimento dos objectos em cena e da eficiência do algoritmo de estimação de movimento usado.

Um desenvolvimento importante em codificação de imagem foi a aplicação de modelos matemáticos descrevendo o movimento de objectos. Devido à complexidade computacional e aos requisitos de processamento em tempo-real, têm sido investigados essencialmente modelos simples descrevendo apenas a componente de translacção do movimento. Um movimento de translacção gera um deslocamento, de frame para frame, do objecto em movimento. Foram desenvolvidos vários métodos para estimar o deslocamento de um objecto em movimento, entre duas frames consecutivas [VEG89]. Conhecendo o vector de deslocamento, este pode ser usado para realizar codificação (DPCM, por transformada, híbrida, etc.) com compensação de movimento.

Os algoritmos de estimação de movimento usados em codificação de imagem, basicamente, podem ser classificados em dois grupos[MUS85]: algoritmos "pel-recursive" e algoritmos "block-matching".

Os algoritmos do tipo "block-matching" tornaram-se mais populares devido à sua menor complexidade. Nesta técnica segmenta-se a imagem em blocos rectangulares de tamanho fixo, e assume-se que todos os pixels pertencentes a um mesmo bloco sofreram igual deslocamento. Cada bloco terá uma translacção linear e o vector de deslocamento associado a cada bloco é codificado.

Utilizando um algoritmo do tipo "block-matching", para encontrar o deslocamento de um bloco de tamanho MxN pixels na imagem actual, este é comparado com os pixels numa área de busca SW da imagem anterior, para encontar o melhor "match". Assumindo um deslocamento máximo, horizontal ou vertical, de d_{max} pixels, a área de busca é dada por [MUS85]

$$SW = (M + 2\, d_{max}) \times (N + 2\, d_{max}) \tag{2.24}$$

como se mostra na figura 2.7. Efectuando uma busca exaustiva (Método de Força Bruta), o número de possíveis deslocamentos inteiros dentro desta janela de busca é

$$Q = (2\, d_{max} + 1)^2 \tag{2.25}$$

A utilização deste método, normalmente, resulta num número excessivo de cálculos. Para ultrapassar esta dificuldade, têm sido propostos vários métodos para simplificar o procedimento de busca.

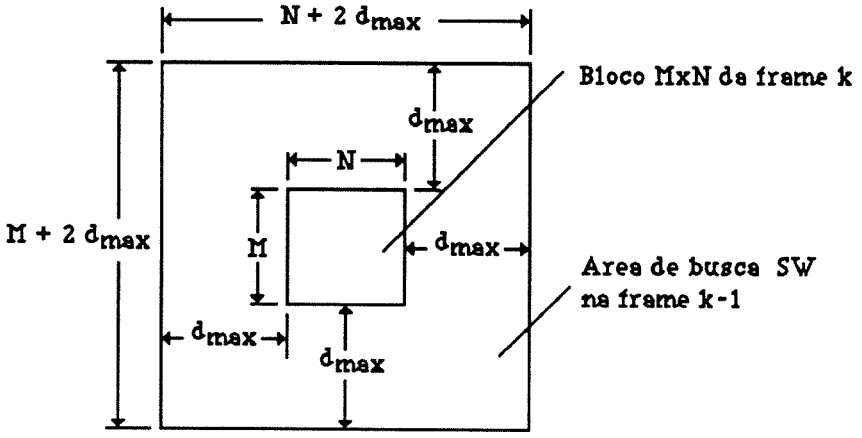


Figura 2.7 - Geometria do bloco MxN e área de busca SW.

O critério de "matching" pode ser uma função de correlação cruzada, o erro quadrático ou, simplesmente, o erro absoluto da diferença. Este último tem a vantagem de não necessitar de multiplicações. Também têm sido propostos vários métodos para simplificação da operação de busca do melhor "match". Geralmente, a busca é limitada a ± 7 pixels, aproximadamente. A busca pode ser ainda acelerada sendo a área de busca sucessivamente reduzida a metade ou menos.

Os algoritmos "pel-recursive" estimam o deslocamento de cada pixel da imagem, permitindo uma estimação mais correcta. A estimativa do deslocamento baseia-se, vulgarmente, na aplicação de um algoritmo de gradiente. Devido à complexidade e esforço computacional desta técnica, a sua aplicação tem sido restringida a problemas que não exigem processamento em tempo-real.

Capítulo 3

CODIFICAÇÃO DE SEQUENCIAS DE IMAGENS PARA TRANSMISSÃO A TAXAS REDUZIDAS

3.1 INTRODUÇÃO

O CCITT Study Group XV, responsável pela normalização de codecs de vídeoconferência, estabeleceu o "Specialists Group on Coding for Visual Telephony" com a finalidade de recomendar uma norma para o algoritmo de codificação de sinais de vídeo para transmissão digital a nx384 Kbit/s (n=1,..., 5) e px64 Kbit/s (p=1, 2,..., 30).

O trabalho desenvolvido no âmbito do projecto europeu COST 211bis foi fundamental para o estabelecimento de um standard para videoconferência a taxas de transmissão de 64 Kbit/s a 2Mbit/s. De facto, o trabalho de simulação desenvolvido por este grupo permitiu definir um algoritmo, referido por "Reference Model", para codificação de vídeo a taxas de px64 Kbit/s. Este algoritmo foi evoluindo sendo a sua versão final conhecida como "Reference Model 8" (RM8) [COS89]. Este documento foi submetido para normalização, em 1990, originando a Recomendação H.261 do CCITT [CCI90]. O algoritmo proposto baseia-se numa codificação híbrida DPCM/transformada, tendo sido adoptada a Transformada Discreta de Cosseno (DCT).

O CCITT "Specialists Group on Coding for Visual Telephony" foi cuidadoso ao normalizar ítems que são considerados fundamentais e ao deixar outros para inovação. Consequentemente, certa tecnologia particular pode ser incorporada em codecs de diferentes fabricantes para diferenciação do produto e competição pela partilha do mercado.

Após o estabelecimento da norma para codificação de vídeo a taxas iguais ou superiores a 64 Kbit/s, o esforço de investigação concentra-se na codificação de vídeo para transmissão a taxas inferiores a 64 Kbit/s, falando-se em taxas tão baixas como 8Kbit/s.

O projecto europeu COST 211bis deu lugar ao projecto COST 211ter que tem como um dos objectivos principais o estabelecimento de um "Reference Model" para codificação de imagens móveis a muito baixas taxas de transmissão ($\leq$ 32 Kbit/s) [COS91]. Uma das forças impulsionadoras deste trabalho é a introdução de redes móveis digitais, embora outras aplicações possam usufruir das vantagens de codificação a estas taxas. Tendo em conta que a taxas tão baixas a qualidade resultante para as imagens não será perfeita, e que aplicações diferentes podem tolerar diferentes tipos de degradação, coloca-se a hipótese de se chegar a mais de um modelo de referência. Embora a compatibilidade com H.261 seja desejável, do ponto de vista de serviço, os requisitos de qualidade a estas taxas podem pedir outros métodos.

Assim, continuam a ser estudados algoritmos baseados na codificação híbrida DPCM/transformada, proposta na H.261, mas paralelamente desenvolvem-se técnicas baseadas noutros métodos como, por exemplo, a codificação baseada em Modelos e a Quantificação Vectorial.

A Quantificação Vectorial apresenta-se como uma técnica capaz de atingir taxas de transmissão baixas com um nível de complexidade relativamente reduzido. Nos últimos anos, tem-se verificado uma intensa actividade de investigação nesta área [NAS88]. Relativamente às técnicas clássicas, a Quantificação Vectorial será uma alternativa vantajosa para aplicações a taxas reduzidas (compactações inferiores a 1:8). Além da vantagem de ser uma técnica com carácter assimptoticamente óptimo do ponto de vista da Teoria da Informação, a Quantificação Vectorial tem ainda uma vantagem muito importante, do ponto de vista prático, que é a extrema simplicidade da descodificação.

Assim, foi desenvolvido um algoritmo de codificação de sequências de imagens usando quantificação vectorial adaptativa, espacial e temporalmente, permitindo atingir taxas de compressão elevadas com uma reduzida complexidade de implementação [INE91] [MAR92]. A adaptatividade nas três dimensões do sinal de vídeo é conseguida aplicando dois mecanismos complementares: segmentação da imagem e quantificação vectorial adaptativa temporalmente. Explorando a redundância espacial do sinal de entrada, a segmentação da imagem resulta num algoritmo capaz de codificar as áreas mais relevantes da imagem com uma qualidade superior. Permitindo um algoritmo de alocação de bits extremamente flexível, a quantificação vectorial adaptativa temporalmente estende esta capacidade de acompanhar as variações do sinal de entrada ao domínio do tempo. A combinação destas duas técnicas leva a um codificador que pode distribuir de forma optimizada, em cada momento, a largura de banda de transmissão disponível entre as diferentes áreas da imagem de entrada, maximizando a qualidade global da codificação.

3.2 FORMATO DE VIDEO

Um sinal vídeo é composto por uma sequência de imagens ("frames"). Em alguns casos, cada imagem é composta por dois quadros interlaçados, com a finalidade de eliminar o efeito de "flickering". Um sinal vídeo a cores pode ser representado por três componentes de cor primárias (vermelho, verde e azul), ou por uma componente

de luminância e duas componentes de crominância. As duas componentes de crominância podem ser moduladas e combinadas com a componente de luminância para formar um sinal de vídeo composto.

Existem, presentemente, diferentes standards de televisão na Europa (625 linhas/50 Hz) e nos EUA e Japão (525 linhas/60 Hz), que são incompatíveis entre si [PRI80]. Por outro lado, a largura de banda destes sinais de televisão é demasiado grande para um serviço de vídeotelefone economicamente viável. A solução para este problema foi adoptar uma norma internacional para um formato de sínal vídeo, comum aos sistemas 625/50 e 525/60, com uma resolução espacial reduzida. O CCITT "Specialists Group" adoptou o "Common Intermediate Format" (CIF) e 1/4CIF como formatos de sinal vídeo para vídeotelefone. Na figura 3.1 apresentam-se os parâmetros para estes formatos. Os formatos locais são convertidos para este formato CIF antes de ser efectuada a codificação de vídeo de modo que o descodificador não necessita distinguir em que região está o codificador que com ele comunica. A conversão necessária é efectuada na transmissão e na recepção, não estando sujeita a recomendação.

As imagens são codificadas como luminância Y e crominância (duas componentes diferença de cor, U e V), C_B e C_R. Estas componentes e os códigos representativos dos seus valores amostrados são os definidos na Recomendação 601 do CCIR [CCI86].

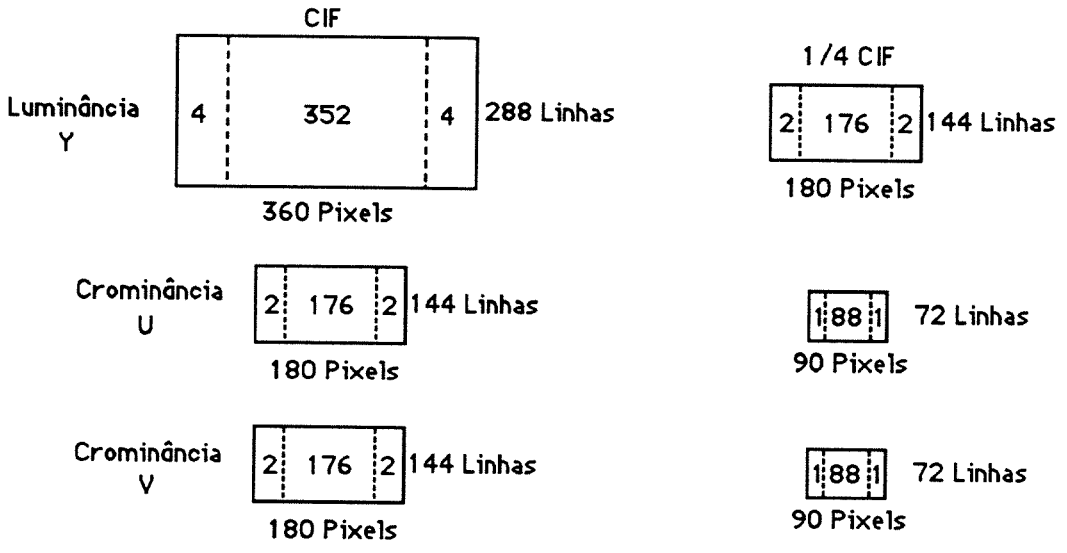


Figura 3.1 - Formatos de imagem para vídeo a baixas taxas de transmissão..

Foram definidos os seguintes parâmetros para o CIF:

- Número de pixels por linha
  Y: 360, C_R: 180, C_B: 180

- Número de linhas por imagem
  Y: 288, C_R: 144, C_B: 144

- Número de imagens por segundo: 30000/1001 ≈ 29.97

- Sem interlaçamento

- Estrutura de amostragem (cf. figura 3.2):
  Ortogonal, a posição das amostras de crominância partilha os mesmos limites de bloco com as amostras de luminância

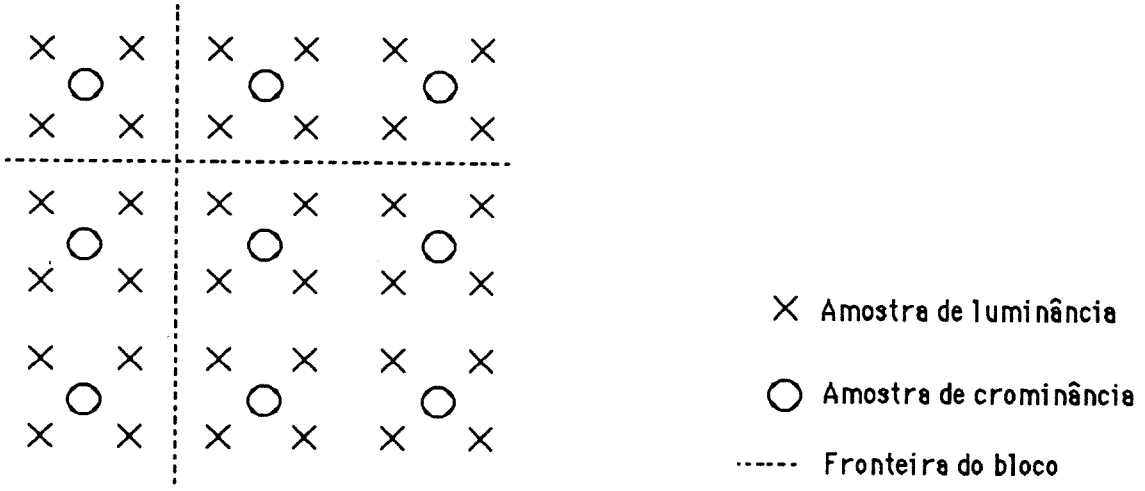


Figura 3.2 - Posição das amostras de luminância e crominância.

Note-se que as componentes de crominância são subamostradas 2:1 tanto horizontalmente, como verticalmente. Com o objectivo de ter um número de pixels por linha múltiplo de 16 (para facilidade de aplicação do algoritmo de codificação adoptado e descrito no ponto 3.3), o número de pixels codificados por linha foi reduzido, mantendo-se o número de linhas por imagem, obtendo-se a área codificada da imagem, chamada "Significant Pel Area" (SPA). Assim,

- Número de pixels por linha
  Y: 352, C_R: 176, C_B: 176

- Número de linhas por imagem
  Y: 288, C_R: 144, C_B: 144

Foi especificado um segundo formato, denominado "Quarter-CIF" (QCIF ou 1/4CIF), contendo metade do número de pixels por linha e metade do número de linhas do CIF.

Com 30 frame/s e 8 bit/pixel, sem compressão, a taxa de transmissão para CIF e QCIF é 36.5 e 9.12 Mbit/s, respectivamente. Consequentemente, é necessária uma taxa de compressão de, aproximadamente, 560:1 para um canal B da RDIS transportar um sinal vídeo CIF. Por esta razão, é usado como formato básico para aplicações de vídeotelefone o QCIF, sendo o formato CIF opcional. Uma redução da taxa de transmissão pode ser conseguida reduzindo a frequência de frame de 30 para 15, 10 ou 7.5 Hz. Consequentemente, para um canal B transportar um sinal QCIF, é necessária uma taxa de compressão que varia de 35:1 a 140:1, aproximadamente, ou por outras palavras, o número de bits por pixel disponíveis varia entre 0.06 e 0.23.

3.3 O CODEC DE VIDEO PROPOSTO NA RECOMENDAÇÃO H.261 DO CCITT

A Recomendação H.261, publicada em meados de 1990 [CCI90], é actualmente a norma para codecs de vídeo na taxa px64 Kbit/s. Esta recomendação descreve o algoritmo de codificação/descodificação de vídeo no âmbito dos serviços audiovisuais a taxas de px64 Kbit/s, variando p entre 1 e 30. Os elementos principais do algoritmo são: predição, estimação de movimento, DCT, quantificação e codificação de entropia.

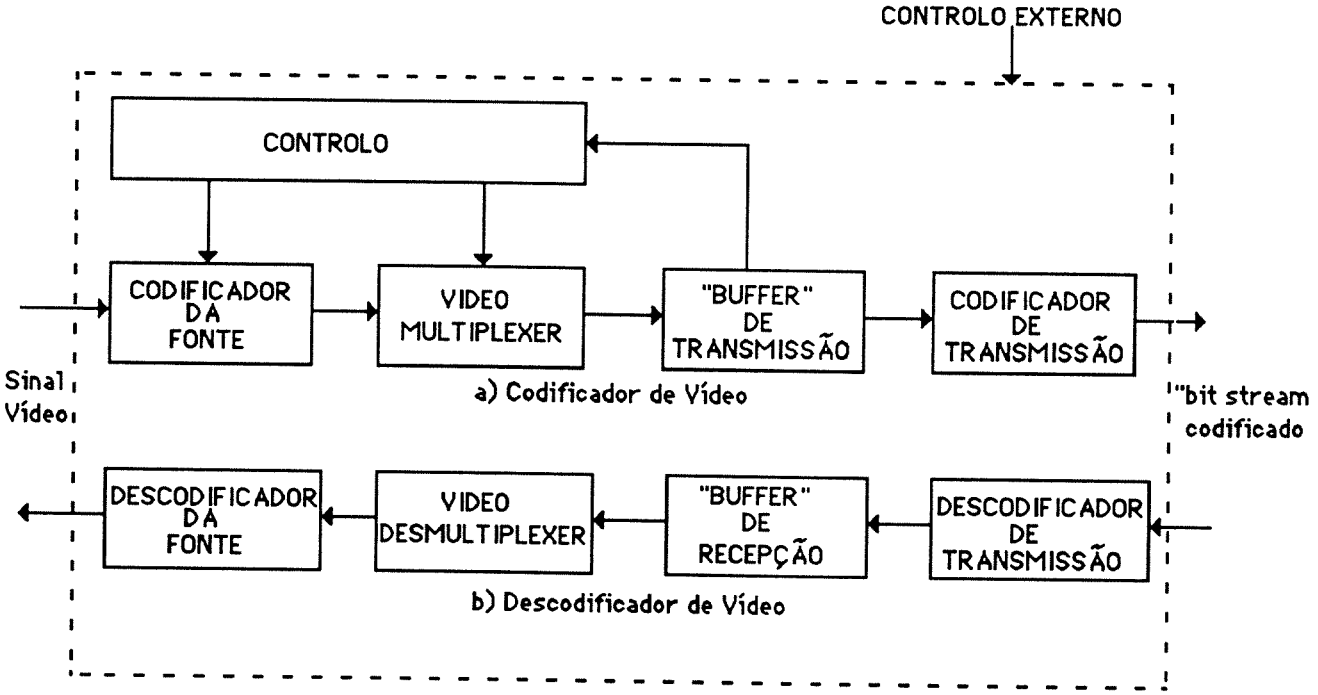


Figura 3.3 - Diagrama de blocos funcional do codec de vídeo.

Apresenta-se na figura 3.3 um diagrama de blocos funcional do codec de vídeo px64 Kbit/s. O sinal de vídeo é pré-processado para ser convertido no formato de vídeo pré-definido, CIF ou QCIF, sendo a taxa de frame máxima de 30 Hz. O codificador da fonte remove a informação redundante do sinal de entrada e quantifica o sinal resultante dentro de uma gama de distorção tolerável, de acordo com o algoritmo de codificação. O vídeo multiplexer atribui palavras de código de comprimento variável aos dados do codificador e multiplexa-os num "bit stream" digital com uma estrutura de dados pré-determinada. Um codificador de transmissão multiplexa sinais multi-media (vídeo, áudio, dados e controlo) num "bit stream" com a taxa e o formato do canal de transmissão, de acordo com a estrutura de trama e a especificação da interface.

3.3.1 Algoritmo de Codificação

Foi adoptado um codificador híbrido DPCM/transformada como algoritmo base de codificação. Apresenta-se na figura 3.4 a estrutura genérica do codificador.

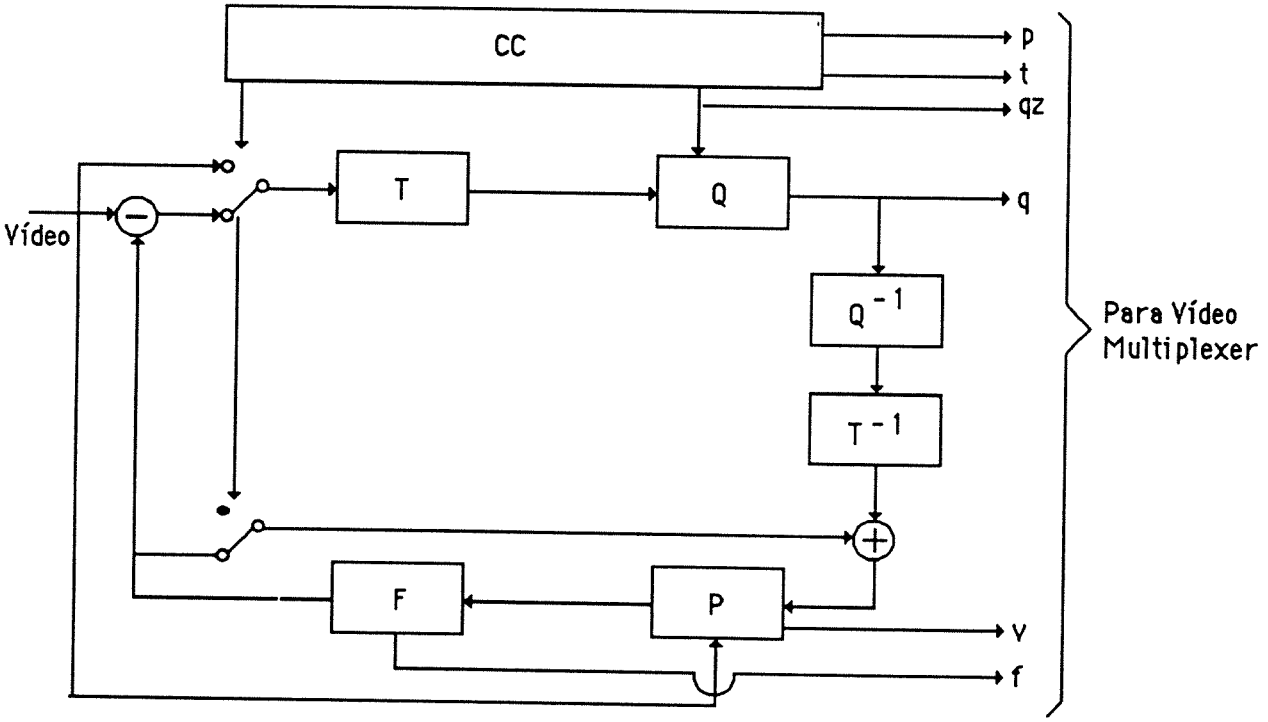


Figura 3.4 - Estrutura genérica do codificador.

Para codificação da fonte é usada uma técnica híbrida DPCM/transformada com compensação de movimento. A malha DPCM é aplicada na dimensão temporal, isto é, interframe. Para isso, é incluída nesta malha uma memória de frame, contendo a imagem previamente reconstruída. A malha DPCM não está activa para codificação intraframe.

O erro de predição (modo INTER), ou a imagem de entrada (modo INTRA), é primeiro dividida em macroblocos (cf. figura 3.5), cada um consistindo num bloco de luminância de 16x16 pixels e nos dois blocos de crominância de 8x8 pixels correspondentes (note-se que as cores foram subamostradas). O bloco de luminância 16x16 é ainda dividido em quatro sub-blocos 8x8. Um aspecto importante do algoritmo é que a transformada é efectuada sobre os blocos 8x8, quer para a luminância, quer para a crominância, para reduzir a dificuldade de processamento em tempo-real, sendo a estimação de movimento efectuada, opcionalmente, nos blocos de 16x16, só para a luminância [COS89].

Os blocos 8x8 são classificados em *transmitidos* e *não-transmitidos*. Só os blocos *transmitidos* são processados.

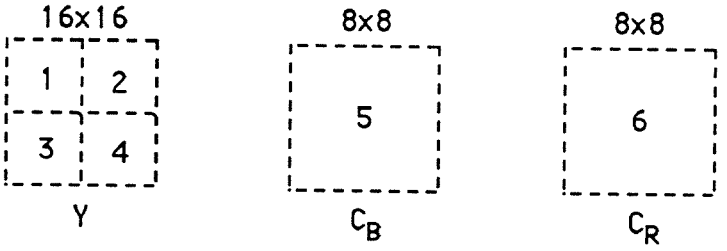


Figura 3.5 - Composição de um macrobloco.

O critério para escolha do modo (INTER/INTRA) e dos blocos a transmitir não está sujeito a recomendação e pode ser alterado dinamicamente.

O tipo de transformada usada nesta aplicação é a Transformada Discreta de Cosseno (DCT) [AHM74]. Esta é aplicada aos blocos 8x8 *transmitidos*, que podem ser o erro de predição, com ou sem compensação de movimento (modo INTER) ou os pixels da imagem de entrada (modo INTRA). Obviamente, se um bloco não contém movimento ou o seu valor estimado é exacto, a entrada para a DCT será uma matriz nula. Para imagens com pouco movimento, a matriz de entrada da DCT irá conter muitos zeros. A saída da DCT é uma matriz de coeficientes que representam energia no domínio

bidimensional das frequências. Em geral, a maior percentagem de energia está concentrada no canto superior esquerdo da matriz, que é a região das baixas frequências. Se for feito um varrimento zig-zag dos coeficientes, pela ordem apresentada na figura 3.6, a sequência resultante conterá longas "strings" de zeros, especialmente para o fim da sequência. É claro, neste ponto, que um dos principais objectivos deste algoritmo de compressão é criar zeros e agrupá-los, para uma codificação eficiente. Exactamente pela mesma razão, é também aplicado a esta sequência um limiar variável, antes da quantificação. Isto é conseguido incrementando o limiar (um coeficiente da DCT é colocado a zero se é menor ou igual ao limiar) quando é detectada uma "string" de zeros.

Aos coeficientes da transformada é aplicado um quantificador uniforme. O passo de quantificação pode ser ajustado pela taxa de transmissão, segundo indicação da ocupação do "buffer" de transmissão. Quando a taxa de transmissão atinge os seus limites, o passo de quantificação será incrementado de modo a que menos informação seja codificada, o que resultará numa imagem degradada. Por outro lado, o passo de quantificação será decrementado, para melhorar a qualidade da imagem, quando a taxa de transmissão estiver abaixo do seu limite.

1	2	6	7	15	16	28	29
3	5	8	14	17	27	30	43
4	9	13	18	26	31	42	44
10	12	19	25	32	41	45	54
11	20	24	33	40	46	53	55
21	23	34	39	47	52	56	61
22	35	38	48	51	57	60	62
36	37	49	50	58	59	63	64

Figura 3.6 - Ordem de transmissão dos coeficientes resultantes da DCT.

Para melhorar ainda mais a eficiência da codificação, é usado um esquema de codificação de comprimento variável (VLC), bidimensional, para as sequências de coeficientes da DCT quantificados. Numa dada sequência, o valor de um coeficiente não-nulo (LEVEL) é definido como uma dimensão e o número de zeros precedendo o coeficiente não-nulo (RUN) é definido como a outra dimensão. A combinação dos dois

é definida como um evento (EVENT), isto é,

EVENT = (RUN, LEVEL).

Um código de menor comprimento é atribuído a um EVENT que ocorre com maior frequência e vice-versa. É também usada uma marca de fim de bloco para indicar que não existem mais coeficientes não-nulos na sequência.

Os valores dos coeficientes codificados são então multiplexados juntamente com informação lateral, como por exemplo a classificação do bloco, informação de quantificação e vectores de deslocamento diferenciais. Finalmente, o "bit-stream" resultante é enviado para o "buffer", para transmissão. Note-se que alguma da informação lateral é também codificada com códigos de comprimento variável.

Na malha DPCM é usado, opcionalmente, um filtro passa-baixo. Este filtro, espacial bidimensional, opera em blocos 8x8 e pode ser usado para melhorar o processo de predição. Quando utilizado, é aplicado aos 6 blocos do mesmo macrobloco.

O descodificador efectua a operação inversa do codificador. A sua arquitectura é bastante mais simples do que a do codificador.

PREDIÇÃO

A predição é baseada num esquema de predição interframe, que explora a correlação temporal das imagens. Cada macrobloco (MB) é estimado usando o MB correspondente na frame anterior.

Se o MB tem uma pequena correlação temporal, pode ser usado um esquema intraframe, explorando a correlação espacial da própria imagem. Assim, consegue-se obter um melhor desempenho em cortes de cena, movimento muito rápido e em áreas de fundo descoberto.

A decisão entre o modo de predição a usar (INTER/INTRA) é feita com base no MB, e não está sujeita a recomendação.

A predição pode ser melhorada usando Compensação de Movimento e Filtragem Espacial.

COMPENSAÇÃO DE MOVIMENTO

O erro de predição pode ser minimizado usando compensação de movimento. O algoritmo de estimação de movimento a utilizar não está especificado na recomendação, estando no entanto, especificadas algumas regras a que deve obedecer.

A estimação de movimento é aplicada apenas aos blocos de luminância e é baseada no MB, ou seja, é efectuada com um tamanho de bloco de 16x16 pixels. Assim, há apenas um vector de movimento por MB que é usado para os 4 sub-blocos de luminância. A gama de busca é de ±7 pixels, calculada entre a frame actual e a frame previamente reconstruída. Cada vector de movimento tem uma componente vertical e uma componente horizontal, com valores inteiros não excedendo ±15.

O vector de movimento para os dois blocos de crominância é obtido dividindo por 2 os valores de ambas as componentes do vector de movimento, e truncando para obter componentes com valores inteiros.

Valores positivos das componentes do vector de movimento significam que a predição para esse MB é formada por pixels na imagem anterior, localizados espacialmente à direita e abaixo dos pixels que se estão a estimar. Todos os pixels referenciados por um vector de movimento devem estar localizados dentro da área codificada da imagem (SPA).

Para tornar a codificação mais eficiente, a informação que é realmente transmitida é o vector de movimento diferencial, ao qual é aplicado um código de comprimento variável. O vector de movimento diferencial é a diferença entre o próprio vector de movimento e o vector de movimento do MB precedente, O vector de movimento do MB anterior é assumido como zero se o último MB *transmitido* nao é contíguo ao actual ou se o MB anterior não era compensado em movimento.

FILTRO

A aplicação de vectores de deslocamento na imagem anterior pode originar ruído de alta frequência, chamado "efeito mosquito". Por seu lado, a quantificação também introduz ruído na malha de realimentação. Embora a predição possa ser óptima em valor médio, pode incluir diferenças muito grandes em alguns pixels. Procurando melhorar o processo de predição pode-se usar um filtro passa-baixo, espacial

bidimensional, operando em blocos 8x8, que diminui o ruído de alta frequência.

O filtro é separável em dois filtros não recursivos unidimensionais, um horizontal e outro vertical. Os seus coeficientes são 1/4, 1/2 e 1/4, excepto nas zonas limítrofes do bloco onde são alterados para 1, 0 e 1, para evitar que caia fora dos limites do bloco.

De acordo com o tipo do MB, compensado em movimento ou não, o filtro é aplicado ou não, a todos os 6 blocos do mesmo MB.

TRANSFORMADA

Os blocos *transmitidos* são processados, antes de serem realmente transmitidos, por uma DCT bidimensional, separável, de dimensão 8x8.

A saída da transformada directa é representada por 12 bit/coeficiente, na gama -2048 a +2047. A saída da transformada inversa é representada por 9 bits, cobrindo a gama -256 a +255.

As funções de transferência da DCT são dadas por:

DCT directa

$$F(u,v) = \frac{1}{4} C(u)\, C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos[\pi(2x+1)u/16] \cos[\pi(2y+1)v/16] \qquad (3.1)$$

DCT inversa

$$f(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)\, C(v) \; F(u,v) \cos[\pi(2x+1)]u/16] \cos[\pi(2y+1)v/16] \qquad (3.2)$$

sendo $x, y = 0, 1, ..., 7$; coordenadas espaciais da imagem

$u, v = 0, 1, ..., 7$; coordenadas no domínio da transformada

e $C(u), C(v) = 1/\sqrt{2}$; $u=0,\ v=0$

$= 1$; $u\neq 0,\ v\neq 0$

O procedimento para cálculo das transformadas não está sujeito a recomendação. No entanto, está especificada (Apêndice A da recomendação) uma tolerância de erro que a transformada inversa deve respeitar.

QUANTIFICAÇÃO

Antes da quantificação, e independentemente desta, é aplicado aos coeficientes um limiar variável a fim de incrementar o número de coeficientes nulos. O valor do limiar depende do comprimento da sequência de coeficientes nulos. Assume-se que foi feito um varrimento zig-zag dos coeficientes de modo a formar um vector, antes do processo de quantificação.

Existem 32 quantificadores diferentes, todos eles possuindo uma característica linear. Para o coeficiente DC, em modo INTRA, é usado sempre o mesmo quantificador com um passo de quantificação de 8. Dos restantes 31 quantificadores, é escolhido um, MB a MB, para ser usado em todos os outros coeficientes. O passo de quantificação pode variar entre 4 e 64, com degrau 2, controlado pelo nível de enchimento do "buffer" de transmissão.

Embora os níveis de decisão não tenham sido definidos nesta recomendação, é possível construir uma característica de quantificação com base nos níveis de reconstrução, os quais são especificados.

3.4 QUANTIFICAÇÃO VECTORIAL ADAPTATIVA DE SEQUENCIAS DE IMAGENS

3.4.1 Introdução

Nos últimos anos, tem-se verificado uma intensa actividade de investigação na área da codificação de sequências de imagens para transmissão a taxas reduzidas. A codificação por transformada aplicada à imagem diferença é, actualmente, a solução mais comum. Como a quantificação vectorial [GRA84] provou ser uma técnica bem sucedida para elevadas taxas de compressão, têm sido propostos codificadores [WAT89] [NAS89] em que o codificador por transformada é parcial ou totalmente substituído por um quantificador vectorial. O principal inconveniente da quantificação vectorial é o crescimento exponencial da tabela de códigos com o aumento da taxa de transmissão e da dimensão do vector, o que leva a dificuldades de implementação.

A quantificação vectorial tem sido usada, principalmente, na codificação de blocos de pequena dimensão, tipicamente 4x4 pixels, o que condiciona a qualidade da codificação. Com o objectivo de ultrapassar esta limitação têm sido propostos vários algoritmos, a maioria dos quais se baseia em quantificação vectorial adaptativa. Os

quantificadores vectoriais adaptativos podem-se agrupar em duas grandes classes: adaptativos espacialmente e adaptativos temporalmente.

Os quantificadores vectoriais adaptativos espacialmente [VAI87] [DAL88] [HO88] [NAS89] [BOX90] [COR90] baseiam-se, geralmente, na segmentação da imagem em blocos de dimensão variável, de modo a atribuir um maior número de bits por pixel às áreas mais activas da imagem. Usando esta técnica, o mesmo número de bits de codificação produz imagens com melhor qualidade subjectiva, uma vez que as regiões activas certamente incluem os contornos, que são importantes sobretudo em termos de avaliação subjectiva da qualidade da imagem.

Os quantificadores vectoriais adaptativos temporalmente [GOL86] [GOL88] baseiam-se, geralmente, numa substituição periódica da tabela de códigos de modo a acompanhar as variações temporais das características do sinal de entrada. Para cada frame, ou grupo de frames, é calculada uma nova tabela de códigos que é usada para quantificar os vectores de entrada. Deste modo, cada frame é codificada com a tabela de códigos mais adequada às suas características sendo a distorção minimizada. No entanto, a complexidade envolvida no cálculo da nova tabela de códigos, e o incremento na taxa de transmissão introduzido pela necessidade de transmitir a nova tabela de códigos, têm impedido a utilização prática deste tipo de quantificadores vectoriais.

Nesta secção apresenta-se um algoritmo de quantificação vectorial que inclui adaptatividade espacial e temporal, com reduzida complexidade de implementação. O algoritmo baseia-se na segmentação da imagem diferença, proposta em [COR90], em blocos de dimensão e forma variável, de acordo com uma medida de actividade local da imagem. Baseado nesta medida de actividade, os blocos resultantes são agrupados em três classes e codificados com três bancos de quantificadores vectoriais de taxa variável.

Controlando o número de bits atribuidos a cada classe, o codificador é capaz de atribuir mais bits às áreas que contêm mais informação sobre a imagem. Além disso, como os quantificadores têm taxa variável, a capacidade de canal disponível pode ser distribuída entre estas de forma optimizada. O resultado é um procedimento de alocação de bits muito flexível, capaz de cobrir uma vasta gama de taxas de transmissão e de qualidade de codificação. Embora flexível, este esquema é simples de implementar uma vez que, após a segmentação, o codificador sabe qual o número de blocos de cada classe, as suas características e pode, facilmente, calcular a taxa que deve atribuir aos blocos de cada classe.

3.4.2 Segmentação das Imagens em Blocos de Dimensão e Forma Variável

Como foi referido, com o objectivo de atribuir um maior número de bits por pixel às áreas mais activas da imagem, têm sido propostos codificadores baseados na segmentação da imagem em blocos de dimensão variável [VAI87] [BOX90]. Corte-Real e P. Alves [COR90] propuseram uma abordagem diferente para a codificação de sequências de imagens baseada na quantificação vectorial de blocos de dimensão e forma variável.

O algoritmo proposto segmenta uma imagem diferença usando um conjunto de blocos pré-definidos. A imagem diferença é inicialmente dividida em blocos de 4x4 pixels, formando uma matriz de blocos. Os blocos que têm um índice de actividade superior a um limiar pré-determinado são classificados em *Tipo 0* e os restantes em *Tipo 1*. Os blocos do *Tipo 0* são considerados muito activos e devem ser codificados individualmente. Os blocos *Tipo 1* são considerados pouco activos. Em seguida, a imagem é varrida sistematicamente ao longo de linhas e colunas de blocos 4x4, com o objectivo de agrupar os blocos *Tipo 1* de modo a formar macroblocos com uma área superior. Para cada bloco *Tipo 1*, agrupando-o com os vizinhos à direita e abaixo, procura-se obter blocos quadrados ou rectangulares de dimensão variável, pertencendo a um de 10 tipos previamente definidos (cf. tabela 1) e tendo um índice de actividade inferior ao limiar. Note-se que apenas blocos do *Tipo 1* são agrupados. A figura 3.7 mostra um exemplo de uma imagem diferença segmentada.

Tabela 1 - CARACTERISTICAS DOS BLOCOS

Tipo	Dimensão
0	4x4
1	4x4
2	8x4
3	4x8
4	8x8
5	16x8
6	8x16
7	16x16
8	32x16
9	16x32
10	32x32

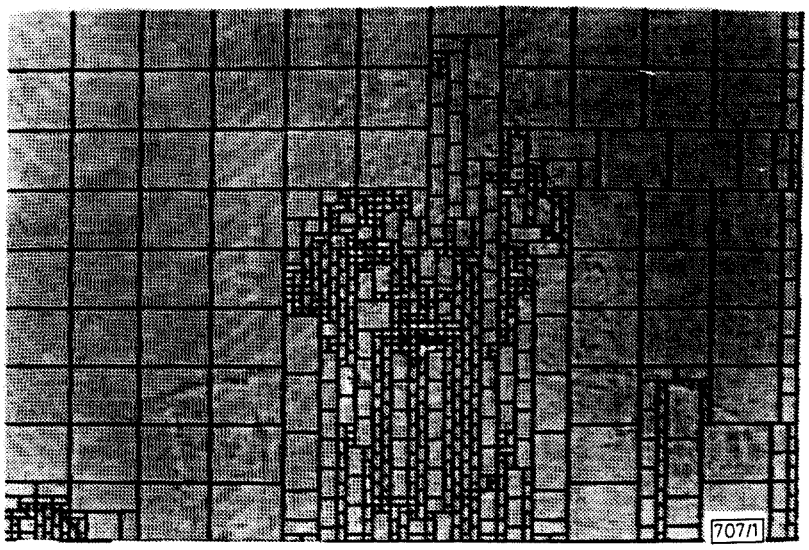


Figura 3.7 - Exemplo da segmentação da imagem diferença.

Após a segmentação, todos os blocos são convertidos para uma das três classes seguintes:

Classe 0: Blocos inicialmente considerados muito activos (*Tipo 0*). Dimensão 4x4.

Classe 1: Blocos quadrados considerados pouco activos. Dimensão 4x4.

Classe 2: Blocos rectangulares considerados pouco activos. Dimensão 8x4.

Esta conversão é feita por filtragem passa-baixo seguida de uma decimação apropriada. Para converter os blocos rectangulares verticais (*Tipos 3, 6 e 9*) para blocos *Classe 2* é necessário efectuar uma rotação de 90^{o}. No descodificador é efectuada a operação inversa em todos os blocos, a fim de reconstruir a imagem original.

Os blocos resultantes são codificados por três codificadores diferentes (um para cada classe de blocos) e, para cada bloco, a palavra de código necessita de ser complementada pela informação relativa à segmentação, para permitir a reconstrução da imagem original, no descodificador. Para reduzir o número de bits necessários para a descrição da segmentação, é utilizado um código de comprimento variável.

3.4.3 Codificação dos Blocos

Os blocos resultantes da segmentação apresentam características e requisitos, em termos de alocação de bits, muito diferentes. Uma vez que os blocos da *Classe 0* veiculam a maioria da informação subjectivamente relevante sobre a imagem, é

importante que possam ser codificados com elevada qualidade, pelo que estes devem ter um codificador específico. Os blocos da *Classe 1* e da *Classe 2*, geralmente associados a zonas de fundo, transportam pouca informação e não necessitam de uma codificação muito sofisticada.

A segmentação permite um algoritmo de codificação com uma excelente adaptatividade à entropia espacial da imagem, mas fraca no domínio temporal. Embora a subtracção entre imagens consecutivas reduza a redundância, este mecanismo não tem capacidade para se adaptar a variações do sinal de entrada, de modo a gerar a taxa mais adequada em cada instante. Esta limitação pode ser ultrapassada usando um quantificador vectorial adaptativo temporalmente, para codificar cada classe de blocos, facultando uma variação temporal do número de bits atribuidos a cada bloco.

O quantificador vectorial adaptativo temporalmente consiste num grupo de bancos de quantificadores, oferecendo uma taxa variável. Cada banco de quantificadores é composto por quantificadores vectoriais de diferentes dimensões e opera num modo de codificação particular. A escolha do modo de codificação a aplicar aos blocos a transmitir e a selecção do quantificador, de entre os disponíveis no banco de quantificadores associado a esse modo, são orientadas pelos seguintes critérios: distorção, taxa de transmissão e complexidade de implementação. Pretende-se maximizar a qualidade das imagens obtidas, controlando a taxa de transmissão gerada (limite superior, imposto pelo canal de transmissão, de 64 Kbit/s) e mantendo a complexidade de implementação dentro de limites impostos pela implementação prática em questão.

A combinação do algoritmo de segmentação com este esquema de bancos de quantificadores permite atingir facilmente um bom compromisso entre adaptatividade e "overhead" na taxa de transmissão, necessário para indicar ao descodificador qual o quantificador usado na codificação. Após a segmentação, toda a informação relativa aos blocos da imagem está disponível para o codificador e, assim, este pode estimar o número de bits que pode ser atribuído a cada classe de blocos e também o número de operações necessárias para os codificar. Desta forma, o número de bits atribuídos pela quantificação pode ser optimizado, imagem a imagem, de acordo com as características do sinal de entrada. Isto resulta num melhor desempenho, não só em termos de relação sinal/ruído mas também em termos de qualidade subjectiva. Escolhendo apenas um quantificador, de entre os disponíveis no banco de quantificadores, para codificar uma imagem, o "overhead" associado ao

processo adaptativo não é significativo. Deste modo, pode ser facilmente implementado um codificador, capaz de explorar a redundância temporal e espacial do sinal de entrada e de se adaptar à capacidade do canal.

Como é sabido, o olho humano apresenta menor sensibilidade à distorção em cenas de muito movimento. Com o algoritmo de segmentação apresentado consegue-se uma elevada qualidade codificando com número elevado de bits apenas as zonas da imagem onde existe grande actividade. Ou seja, cenas com pouco movimento geram muitos blocos de grandes dimensões que originam taxa reduzida. Os poucos blocos activos restantes, que transportam a informação subjectivamente mais valiosa, podem ser assim codificados com elevada resolução dando origem a uma qualidade subjectiva elevada. Por outro lado, nas cenas de muito movimento o algoritmo de segmentação gera muitos blocos activos, que têm de ser codificados com menor resolução. Mas como a sensibilidade do olho humano é, nestes casos, mais reduzida, a diminuição da qualidade da imagem obtida é, em termos subjectivos, atenuada. Consegue-se assim uma qualidade subjectiva bastante superior à obtida com um quantificador não adaptativo.

3.4.3.1 Codificação dos Blocos de Fundo

Após a obtenção da imagem diferença, as áreas de fundo são caracterizadas por uma função densidade de probabilidade Laplaciana, com um pico muito pronunciado centrado em zero, e uma variância reduzida. Após a segmentação, resultam em blocos de grandes dimensões.

Devido à pouca informação contida nestas áreas, a operação de subtracção é, geralmente, suficiente para eliminar a redundância existente, não se justificando qualquer tipo de processamento adicional. Assim, os blocos de fundo (*Tipo 1* a *Tipo 10*) são apenas convertidos para a *Classe 1* (blocos quadrados) e *Classe 2* (blocos rectangulares) e quantificados vectorialmente.

Sendo áreas de reduzida informação, o número de bits requerido para as codificar é também reduzido. Simulações [VAS91] permitiram concluir que 6 bit/bloco é suficiente para imagens de média/alta qualidade, sendo o ganho obtido incrementando este número pouco significativo.

3.4.3.2 Codificação dos Blocos Muito Activos

Geralmente, os blocos muito activos (*Classe 0*) são, uma reduzida área da imagem mas transportam a informação mais significativa em termos de qualidade subjectiva. A codificação destes blocos exige um processamento mais elaborado que o aplicado aos blocos de fundo. No algoritmo proposto em [COR90] a única diferença na codificação destes blocos consiste no uso de um quantificador vectorial com um número de bits significativamente superior ao usado pelos blocos de fundo. Nos pontos seguintes são propostas algumas alternativas com o objectivo de obter desempenhos superiores com taxas de transmissão equivalentes e reduzida complexidade de implementação.

A codificação dos blocos da *Classe 0* é efectuada por um codificador operando em diferentes modos, de acordo com as características do sinal de entrada. Estando estes blocos localizados em áreas de grande actividade, a estimativa do valor de um pixel baseada no pixel correspondente da imagem anterior, não é muito precisa. Assim, torna-se necessário aplicar métodos de predição mais elaborados, ou mesmo codificar o bloco com base apenas na imagem a codificar (modo intra).

3.4.3.2.1 Codificação Preditiva Interframe Com Compensação de Movimento

Usando este modo, os blocos da *Classe 0* são, após a segmentação e antes da quantificação, compensados em movimento, usando um algoritmo de "block-matching". Este método de predição resulta num sinal diferencial (ou erro de predição) com menor energia e, consequentemente, numa melhor qualidade do sinal codificado.

O uso da compensação de movimento origina dois tipos de "overhead":

- a transmissão dos vectores de deslocamento gera um aumento da taxa de transmissão,
- o peso computacional do algoritmo de "block-matching" introduz um aumento na complexidade de implementação.

Para minimizar este "overhead" foram adoptadas as seguintes soluções:

- codificação por entropia dos vectores de deslocamento [HUF52],
- utilização do algoritmo "three-step" [MUS85].

A estimativa do bloco com compensação de movimento é muito mais precisa e, assim, a codificação do bloco é mais fácil e são necessários menos bits (tabelas de códigos menos extensas) para quantificação. O ganho obtido é, geralmente, suficiente para justificar, em termos de complexidade de implementação e taxa de transmissão, a introdução da compensação de movimento.

Com o objectivo de reduzir a taxa de transmissão foi ainda introduzida a seguinte modificação: todos os blocos que, após serem compensados em movimento, apresentam um índice de actividade inferior ao limiar, são reclassificados e codificados de modo idêntico aos blocos de fundo. Como, nestes casos, a compensação de movimento permite uma eficiente predição do bloco, não é introduzida uma significativa degradação de qualidade codificando o bloco como bloco de fundo, isto é, com menos bits.

3.4.3.2.2 Codificação Intraframe por Média Prevista

Para sequências muito activas, o algoritmo de segmentação origina um elevado número de blocos *Classe 0*. Sendo a taxa de transmissão fixa, isto implica que cada bloco seja codificado com um reduzido número de bits. Usando compensação de movimento, alguns destes bits terão que ser, necessariamente, reservados para transmissão dos vectores de movimento. Consequentemente, o número de bits disponível para codificar o erro de predição é insuficiente. Nestas situações, o incremento na taxa de transmissão gerado pela compensação de movimento não pode ser suportado, pelo que, um melhor desempenho pode ser obtido usando um modo intraframe.

O método mais simples de realizar codificação de blocos intraframe é quantificá-los sem qualquer processamento prévio. No entanto, devido à gama dinâmica do sinal de vídeo, é necessário um elevado número de bits por bloco para se obter uma boa qualidade de codificação, mesmo quando a imagem pertence a uma sequência com reduzida actividade.

Um esquema de codificação simples que proporciona um significativo aumento de desempenho consiste em subtrair o valor médio do bloco aos seus pixels, antes da quantificação, e transmiti-la separadamente [BAK82]. O bloco a quantificar fica assim com características semelhantes às de um bloco obtido pela subtracção de imagens consecutivas (f.d.p. laplaciana), independentemente da quantidade de movimento da sequência.

O inconveniente deste tipo de codificação intraframe consiste no "overhead" na taxa de transmissão gerado pela transmissão da média do bloco quantificada escalarmente. Em [BAK83] são sugeridos alguns esquemas para uma codificação mais eficiente deste parâmetro, como por exemplo a utilização de DPCM. No caso presente estes métodos são pouco eficientes, dado que sendo este tipo de codificação utilizado apenas para os blocos de *Classe 0*, normalmente dispersos pela imagem e pertencentes a zonas de movimento desta, não se obtém grande vantagem utlizando informação de blocos anteriores na quantificação da sua média.

Este problema pode ser eliminado introduzindo uma alteração simples, proposta em [FEN88]. Neste caso, em vez de transmitir a média de cada bloco quantificada escalarmente, é efectuada uma estimativa da média no codificador e no descodificador. A média estimada μ é calculada usando os pixels circundantes do bloco, previamente codificados, como se representa na figura 3.8 (codificação por média prevista). A média estimada μ é dada por

$$\mu = \frac{1}{N} \sum_{i=1}^{9} x_i \tag{3.3}$$

onde os x_i's são os pixels circundantes do bloco já codificados. Sendo estes pixels conhecidos do descodificador, o cálculo da média pode ser efectuado sem a transmissão de qualquer informação adicional. Como a predição baseada na média do próprio bloco não é muito precisa, o desempenho com a predição por média prevista não é significativamente inferior. Consegue-se assim eliminar o aumento da taxa de transmissão originado pela transmissão da média quantificada escalarmente, mantendo a qualidade da codificação.

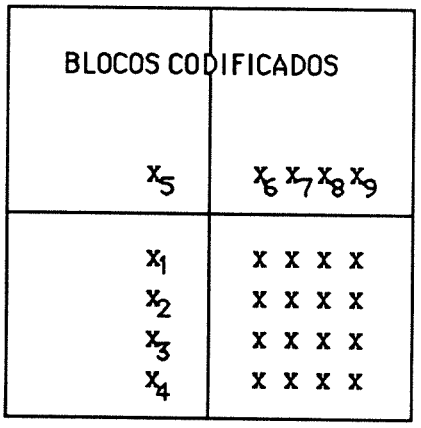


Figura 3.8 - Estimação da média.

Este método torna-se bastante atractivo, comparando com a predição interframe com compensação de movimento, para cenas muito activas que originam uma grande quantidade de blocos do *Classe 0*. Eliminando o "overhead" devido à transmissão dos vectores de deslocamento, podem ser atribuidos mais bits à codificação do bloco. Por outro lado, se os objectos em movimento tiverem elevado detalhe, e o movimento não for puramente translaccional, este método pode oferecer uma predição superior e, consequentemente, uma codificação de melhor qualidade. Finalmente, o número de operações extra-codificação é muito menor, pois o cálculo da média é uma operação de complexidade desprezável comparada com a estimação de movimento. Isto é especialmente importante quando existe um elevado número de blocos Classe *0* e, consequentemente, o tempo disponível para codificar cada um é reduzido.

Em termos de qualidade subjectiva, para cenas de baixa/média actividade e taxas de transmissão não muito reduzidas, o desempenho obtido com este tipo de codificação é inferior ao conseguido com predição interframe com compensação de movimento, devido à existência de efeito de bloco. No entanto, para taxas muito reduzidas onde a compensação de movimento se torna impraticável, apesar do referido efeito de bloco, este método apresenta um desempenho superior.

3.4.3.2.3 Quantificação Vectorial Adaptativa Temporalmente

O desempenho deste algoritmo de quantificação vectorial adaptativa depende essencialmente da selecção dos modos de codificação e dos bancos de quantificadores associados.

Para blocos da *Classe 1* e da *Classe 2* (blocos de fundo) a codificação preditiva interframe, sem compensação de movimento, é suficiente para garantir uma boa qualidade de codificação. No entanto, o mesmo não acontece com os blocos da *Classe 0* (blocos activos), para os quais o melhor modo de codificação depende da actividade da sequência de entrada: intraframe para cenas muito activas e interframe com compensação de movimento para cenas de baixa/média actividade.

Quando a sequência de entrada tem uma actividade reduzida, a codificação não está condicionada pela taxa de transmissão disponível mas pela complexidade computacional da quantificação vectorial, a qual impõe limites práticos no tamanho das tabelas de códigos. Assim, pode-se obter um melhor desempenho introduzindo

outros modos de codificação, usando blocos de menores dimensões, os quais, para o mesmo tamanho de quantificador, oferecem melhor qualidade de codificação gerando taxas de transmissão mais elevadas. Uma vez que o olho humano apresenta menor sensibilidade à distorção em áreas de movimento, uma codificação precisa durante períodos de muito movimento não é tão importante como a rápida recuperação da elevada qualidade quando a actividade diminui. Assim, a codificação de elevada qualidade durante períodos de reduzida actividade tem uma influência determinante no desempenho global da codificação.

Juntando todos os modos descritos, o codificador pode distribuir de uma forma optimizada a largura de banda de canal disponível entre as três classes de blocos originados pela segmentação. A escolha de um modo de operação determina o funcionamento dentro de uma dada gama de taxas de transmissão e a existência de um banco de quantificadores para cada modo possibilita um ajuste mais fino, permitindo que a adaptatividade seja levada até um nível de variação quase contínua do número de bits de quantificação.

3.4.4 Resultados de Simulação e Conclusões

Para avaliar a influência de cada modo de codificação no desempenho global, foi simulado um codificador operando nos seguintes modos:

Blocos de Fundo:

Dimensão do bloco - 4x4 (*Classe 1*) ou 8x4 (*Classe 2*);

Modo de codificação: preditiva interframe sem compensação de movimento.

Blocos Activos:

Modo 1:

Dimensão do bloco - 4x4;

Modo de codificação: intraframe por média prevista.

Modo 2:

Dimensão do bloco - 4x4;

Modo de codificação: preditiva interframe com compensação de movimento.

Modo 3:

Dimensão do bloco - 2x2;

Modo de codificação: intraframe por média prevista.

Todos os resultados apresentados referem-se a 8 imagens da sequência "Trevor" (uma sequência com média/elevada actividade) constituída por imagens de 180x144 pixels (QCIF), e uma frequência de 10 imagens/segundo. As tabelas de códigos foram obtidas a partir de uma sequência de treino constituída por 56 frames pertencentes a três outras sequências, usando o algoritmo LBG [LIN80]. O critério de actividade usado no algoritmo de segmentação foi a variância do bloco, de acordo com a equação (1), sendo σ^2 a variância do bloco, N a dimensão do bloco, x_j a amplitude do pixel j e μ a média do bloco.

$$\sigma^2 = \frac{1}{N} \sum_{j=0}^{N-1} (x_j - \mu)^2 \qquad (3.4)$$

O limiar de actividade foi definido com o valor de 80.

Em todas as simulações, os blocos *Classe 1* e *Classe 2* foram codificados com predição interframe, sem compensação de movimento, usando um banco de quantificadores vectoriais que cobre a gama de 5 a 9 bit/bloco. Para os blocos da *Classe 0*, as simulações foram efectuadas com bancos de quantificadores vectoriais cobrindo a gama de 5 a 10 bit/bloco para o modo 1, 6 a 11 bit/bloco para o modo 2 e 6 a 9 bit/bloco para o modo 3. Assume-se que a primeira imagem é conhecida.

A qualidade das sequências codificadas foi avaliada subjectivamente e por cálculo da relação sinal/ruído (SNR), de acordo com a equação (2), onde y_{ij} é a amplitude do pixel j após a quantificação,

$$\mathrm{SNR} = 10 \log \left[\frac{255^2}{\frac{1}{N} \sum_{j=0}^{N-1} (x_j - y_{ij})^2} \right] \quad (\mathrm{dB}) \qquad (3.5)$$

Em todas as legendas, o primeiro número identifica o número de bits da tabela de códigos dos blocos *Classe 0* e o segundo número identifica o número de bits da tabela de códigos dos blocos *Classe 1* e *Classe 2*.

As figuras 3.9 a 3.12 apresentam curvas típicas de relação sinal/ruído e taxa de transmissão geradas por codificação dos blocos da *Classe 0* nos modos 1 e 2 (blocos de dimensão 4x4).

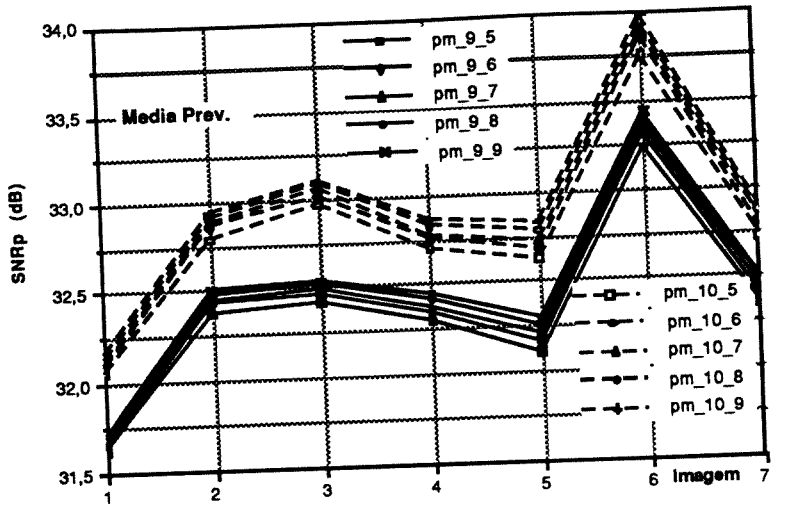


Figura 3.9 - SNR para a sequência Trevor, codificada no modo 1 (intraframe com média prevista), sendo os blocos *Classe 0* quantificados com tabelas de 9 e 10 bit/bloco.

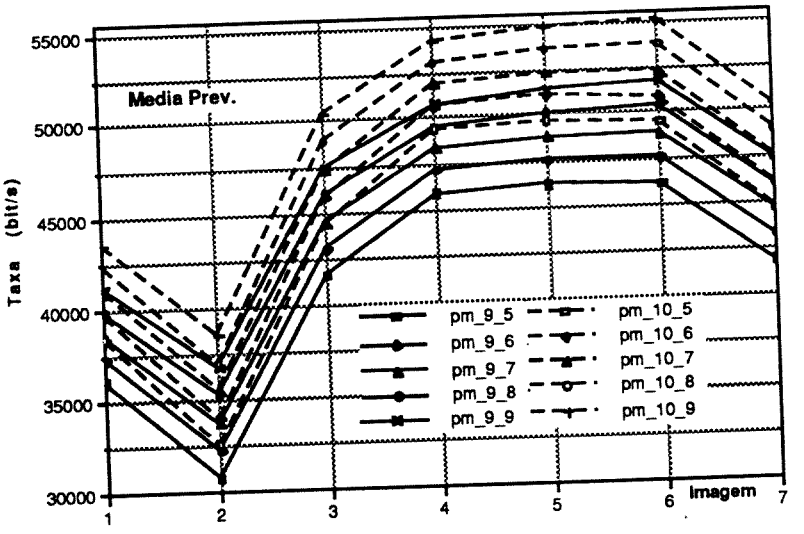


Figura 3.10 - Taxa de transmissão para a sequência Trevor, gerada por codificação no modo 1, sendo os blocos *Classe 0* quantificados com tabelas de 9 e 10 bit/bloco.

Observando as figuras 3.9 e 3.11 pode-se ver que, para estes modos de codificação, o incremento de um bit na quantificação dos blocos *Classe 0* resulta num aumento de qualidade de codificação superior ao obtido com igual incremento no número de bits de quantificação dos blocos de fundo e, simultaneamente, originando um menor incremento na taxa de transmissão global, como se pode constatar nas figuras 3.10 e 3.12.

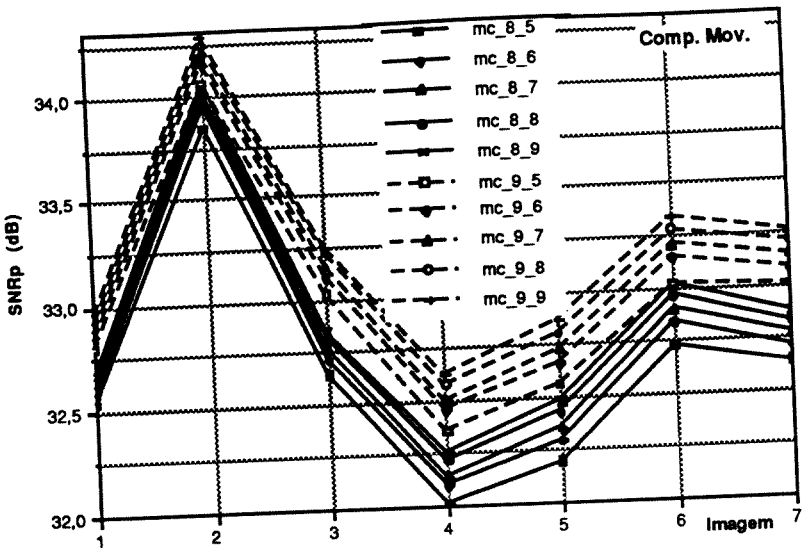


Figura 3.11 - SNR para a sequência Trevor, codificada no modo 2 (interframe com comp. de mov.), sendo os blocos *Classe 0* quantificados com tabelas de 8 e 9 bit/bloco.

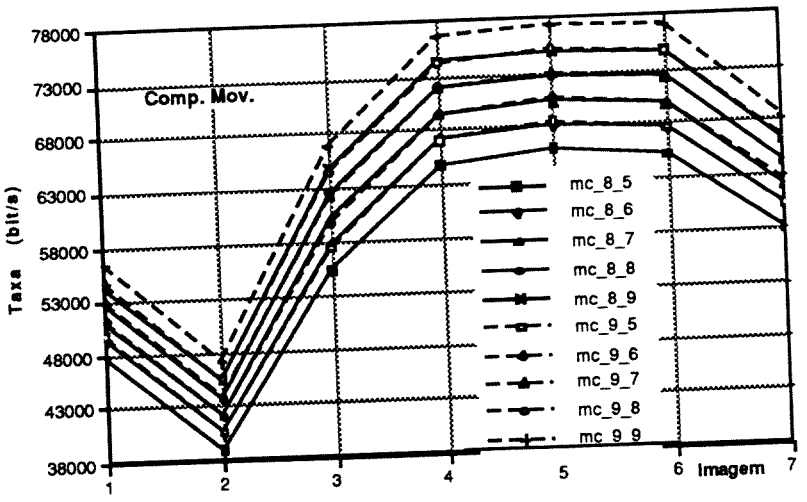


Figura 3.12 - Taxa de transmissão para a sequência Trevor, gerada por codificação no modo 2, sendo os blocos *Classe 0* quantificados com tabelas de 8 e 9 bit/bloco.

No entanto, existem diferenças quantitativas entre os dois modos de codificação. No modo 2 (interframe com compensação de movimento) o ganho em SNR devido a um aumento da taxa de transmissão alocada aos blocos da *Classe 0* é menor do que no modo 1 (intraframe com média prevista). Além disso, mesmo para imagens com muita actividade, o aumento da taxa de transmissão devido à codificação dos blocos da *Classe 0* com mais um bit é equivalente ao produzido pela codificação dos blocos de fundo (*Classe 1* e *Classe 2*) com mais um bit. No modo 2, este aumento é igual ao aumento

originado pela quantificação dos blocos de fundo com mais um, dois ou três bits. No entanto, com ambos os modos de codificação, o ganho em termos de SNR é bastante superior no caso do aumento de um bit na quantificação dos blocos da *Classe 0*, especialmente quando o número de bits por bloco da *Classe 0* é reduzido.

Estas diferenças são justificadas por dois factores: a percentagem de taxa de transmissão gerada pelos vectores de deslocamento é independente do número de bits utilizados na quantificação e a reclassificação dos blocos compensados em movimento, actuando predominantemente em imagens com um grande número de blocos *Classe 0*, reduz os picos que se iriam verificar na taxa de transmissão, mantendo o comportamento do codificador mais homogéneo de imagem para imagem.

A conclusão fundamental da análise dos gráficos é que o desempenho do algoritmo é determinado fundamentalmente pela codificação dos blocos da *Classe 0*. Sempre que a largura de banda disponível for limitada, deve ser dada prioridade a este tipo de blocos, codificando os blocos de fundo com o menor número de bits.

A análise subjectiva das sequências codificadas revela algumas particularidades destes dois modos de codificação. Com o modo de codificação intraframe com média prevista, as imagens exibem, por vezes, efeito de bloco e este é mais acentuado com o decréscimo do número de bits de quantificação, ou seja, quando são usados quantificadores vectoriais com tamanho reduzido para codificar os blocos da *Classe 0*. A qualidade global das imagens não é muito elevada, pelo que, este modo pode ser considerado de baixa qualidade.

Com o modo interframe com compensação de movimento, não se verifica a existência do efeito de bloco, sendo a qualidade subjectiva mais elevada. A avaliação subjectiva das sequências codificadas, com reclassificação dos blocos da *Classe 0*, permite a detecção de um aspecto peculiar deste modo de codificação: em algumas áreas de movimento, onde alguns dos blocos *Classe 0* foram reclassificados como blocos de fundo, se a codificação dos blocos de fundo for feita com um número reduzido de bits, é visível alguma perda de qualidade relativamente aos blocos *Classe 0* vizinhos, codificados com mais bits. Esta observação, não detectável a partir das curvas SNR, mostra como é importante, em modos de codificação de elevada qualidade, ter bancos de quantificadores com tabelas de código com um grande número de vectores para a codificação dos blocos de fundo.

A avaliação subjectiva das sequências codificadas permite ainda concluir que, com qualquer destes modos, não se consegue atingir codificação de elevada qualidade. Para tal, seria necessário quantificar com tabelas de códigos de maior dimensão, o que na prática não se torna viável.

Para evitar esta limitação e tornar possível a codificação de elevada qualidade, é necessário utilizar o modo 3 (codificação por média prevista de blocos de 2x2 pixels). Para isso, cada bloco da *Classe 0* (4x4) é decomposto em quatro blocos 2x2, antes de ser quantificado.

As figuras 3.13 e 3.14 apresentam curvas típicas de SNR e taxa de transmissão originados pela codificação dos blocos *Classe 0* no modo 3.

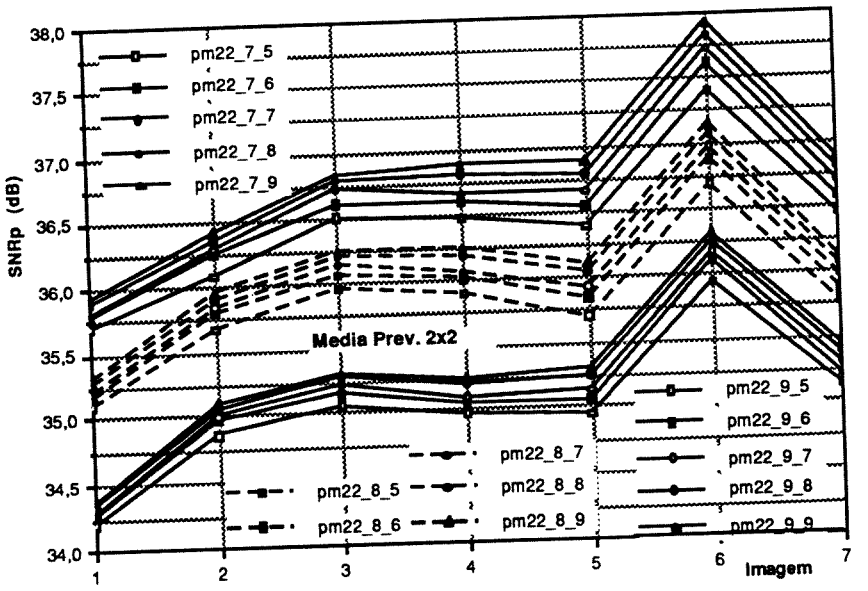


Figura 3.13 - SNR para a sequência Trevor, codificada no modo 3 (intra média prev. e blocos 2x2), sendo os blocos *Classe 0* quantificados com tabelas de 7, 8 e 9 bit/bloco.

Como era esperado, este modo oferece elevada SNR, mesmo durante períodos de grande actividade. Subjectivamente, a qualidade global da imagem é também muito boa, confirmando a importância deste modo para garantir uma rápida recuperação de qualidade após períodos de forte actividade os quais, para uma taxa de transmissão fixa, podem obrigar o codificador a entrar nos modos de mais baixa qualidade.

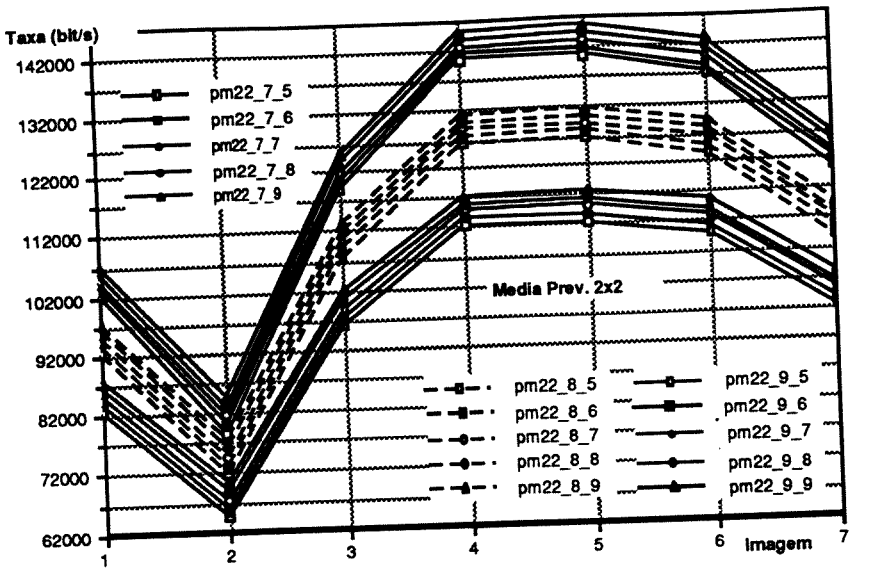


Figura 3.14 - Taxa de transmissão para a sequência Trevor, gerada por codificação no modo 3, sendo os blocos *Classe 0* quantificados com tabelas de 7, 8 e 9 bit/bloco.

As figuras 3.15 e 3.16 mostram as curvas de SNR e taxa de transmissão em função do número de bits dos blocos *Classe 0*, quando os blocos de fundo são codificados com 6 bits. Verifica-se que, com estes três modos, é possivel obter uma qualidade de codificação global que varia entre o fraco (29.5 dB) e o muito bom (37.5 dB), cobrindo um grande número de qualidades intermédias. É igualmente coberta uma vasta gama de taxas de transmissão, e também aqui com uma variação bastante fina.

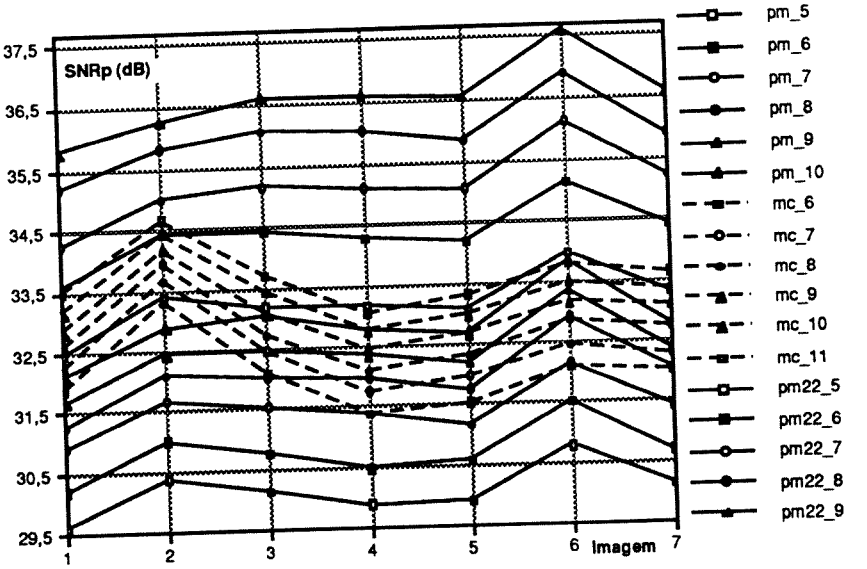


Figura 3.15 - SNR para a sequência Trevor.

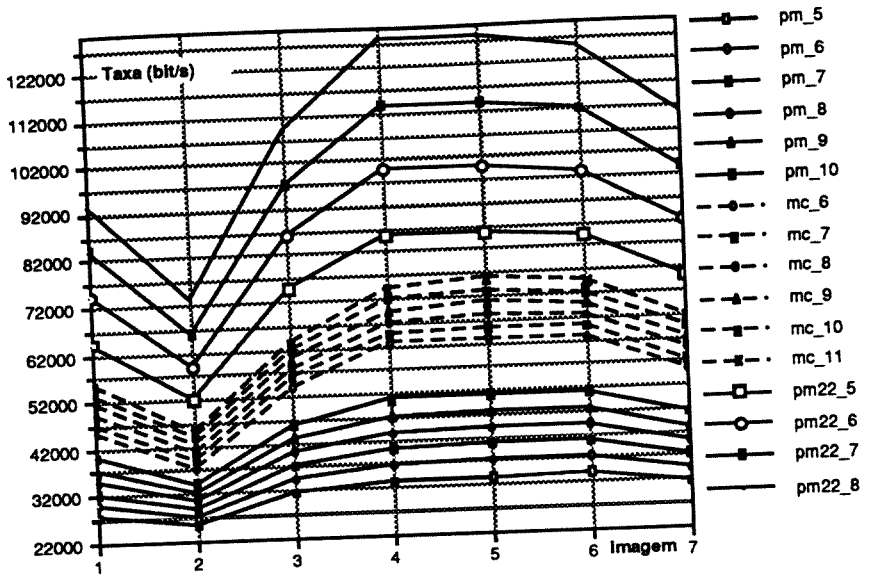


Figura 3.16 - Taxas de transmissão para a sequência Trevor.

A figura 3.17 apresenta uma curva da SNR média, para as 8 frames da sequência de entrada ("Trevor"), em função da taxa de transmissão média, para os três modos de codificação. Cada ponto corresponde a um tamanho específico (N) do quantificador vectorial usado para codificar os blocos da *Classe 0*:

- N de 2^5 a 2^{10} para o modo 1;
- N de 2^6 a 2^{11} para o modo 2;
- N de 2^6 a 2^9 para o modo 3.

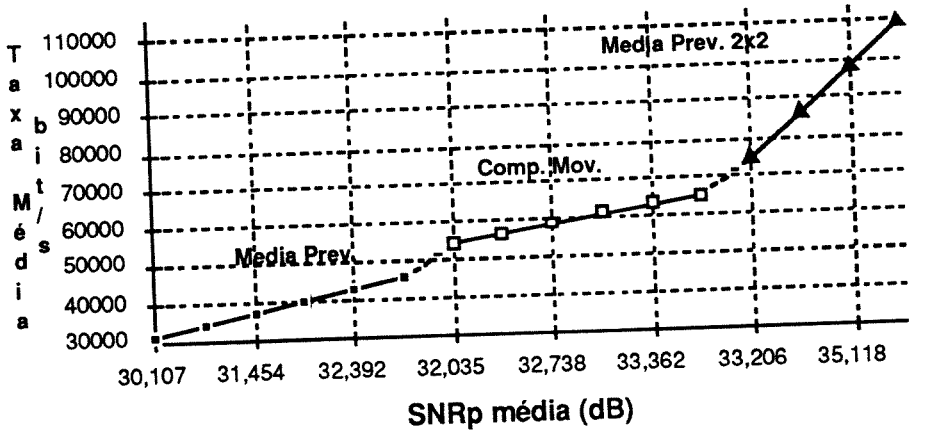


Figura 3.17- SNR média em função da taxa de transmissão média para os modos 1,2 e 3.

Em todos os casos, os blocos de fundo (*Classe 1* e *Classe 2*) foram codificados com um quantificador vectorial de comprimento 2^6. Esta curva permite concluir que, com uma selecção apropriada do modo de codificação, o codificador é capaz de cobrir uma vasta gama de taxas de transmissão, com incrementos relativamente pequenos, aumentando também a qualidade de codificação.

Capítulo 4

IMPLEMENTAÇÃO DE UM CODEC DE VIDEOVIGILANCIA

4.1 INTRODUÇÃO

Neste capítulo descreve-se a implementação de um codec de vídeo de baixa resolução (180x144 pixels) e qualidade reduzida, para transmissão de imagem móvel, à taxa de 64 Kbit/s, para aplicações de televigilância.

O codificador é implementado sobre um sistema de aquisição e processamento de imagem para PC-AT, ou compatível, ao qual poderão estar ligadas uma ou mais câmaras. O descodificador será também implementado sobre uma placa de processamento para PC-AT, com características idênticas à do codificador. O display das imagens será feito num monitor usando uma placa tipo VGA. Os sistemas usados não são específicos para compressão/descompressão de imagem, podendo ser usados em várias aplicações de processamento digital de sinal. Estes sistemas baseiam-se no processador digital de sinal TMS320C30 da Texas Instruments, e foram desenvolvidos pela Universidade de Coimbra [SIL90] [MEL91].

O algoritmo de codificação implementado é baseado na quantificação vectorial adaptativa de blocos de dimensão e forma variável, conforme descrito em 3.4. Note-se que uma das vantagens da quantificação vectorial é a extrema simplicidade do descodificador, o que a torna adequada para aplicações "single coder/multiple decoder", que poderá ser o caso do sistema de televigilância. A Estação Central (descodificador) poderá estar a monitorar mais do que uma Estação Remota (codificador) num mesmo terminal, equipado este apenas com um sistema de descompressão.

4.2 "HARDWARE"

4.2.1 Sistema de Aquisição e Compressão de Imagem

Este sistema [MEL91] é constituído por uma única placa compatível com o barramento PC-AT, com capacidade de aquisição, processamento e armazenamento de imagens a cores. O sistema pode ser dividido nos módulos de aquisição de imagem, sintetização de relógio (PLL), processamento (CPU), interface com o AT e ainda um módulo de memória. O diagrama de blocos da placa é apresentado na figura 4.1.

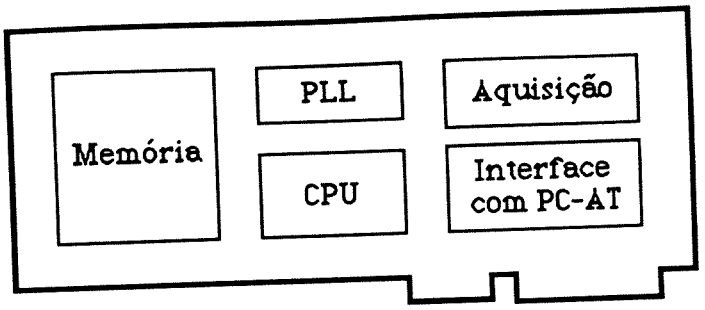


Figura 4.1 - Diagrama de blocos da placa de aquisição e compressão de imagem.

O módulo de processamento e interface com o computador hospedeiro tem as seguintes características:

- Processamento de imagens utilizando um processador digital de sinal de elevado desempenho, o TMS320C30 (que passamos a designar também por C30).
- Capacidade do computador hospedeiro controlar o sistema através da transmissão de comandos e leitura de resultados, após sinalização de fim de execução.

O módulo de aquisição tem as seguintes características:

- Aquisição e digitalização de imagens policromáticas (ou monocromáticas) provenientes de uma câmara cujo formato de saída seja RGB e com o formato semelhante ao CCIR, no que respeita aos sinais de sincronismo.
- Resolução programável da imagem a adquirir.
- Alteração dos limites superior e inferior da gama de digitalização do ADC por "software".
- Possibilidade de ligação simultânea de até duas câmaras policromáticas (ou seis câmaras monocromáticas).

O módulo de memória tem as seguintes características:

- Utilização alternativa de circuitos de memória de 128 Kbytes ou 32Kbytes.
- Capacidade máxima de armazenamento de 640 Kwords (640Kx4 bytes).
- Possibilidade do computador hospedeiro aceder à memória do sistema.

O módulo de selecção de relógio permite comutar o relógio do sistema entre um oscilador a cristal e uma PLL, permitindo operações síncronas com o sinal de vídeo.

A filosofia de funcionamento deste sistema assenta na partilha da memória local pelo processador (TMS320C30) e pelo computador hospedeiro (PC-AT). Escrevendo num registo de controlo do sistema, o PC pode parar o TMS320C30 (barramento em estado de alta impedância) ficando com acesso pleno à memória do sistema. Nesta situação o PC pode carregar na memória local os programas para o C30 executar logo após uma nova escrita no referido registo de controlo. Este registo em leitura funciona também como registo de estado sinalizando ao PC o fim de execução de um comando previamente pedido.

4.2.1.1 CPU

O processador utilizado é o processador digital de sinal da Texas Instruments, TMS320C30 [C30USG]. Este processador tem um elevado desempenho em aplicações de processamento de sinal, como é o caso da compressão e descompressão de imagem, sendo por isso indicado para o CPU desta placa de aquisição e compressão de imagem.

A sua grande flexibilidade provém da capacidade de processamento (33 MFLOPS, 16.7 MIPS), possível graças à implementação em "hardware" de funções que noutros processadores são implementadas em "software" e aos periféricos já internamente implementados (DMA, timers, portos série, etc).

O tempo de ciclo de 60 ns permite adquirir imagens com as resoluções especificadas para este sistema de televigilância.

No ponto 4.2.3 é feita uma descrição mais pormenorizada do TMS320C30.

4.2.1.2 Memória

O TMS320C30 possui um espaço total de endereçamento de 16 Mwords (cada word com 32 bits). Como cada pixel de imagem adquirida em formato RGB ocupa 24 bits (8 bits por cada cor) será guardado um pixel em cada word de memória. Como o C30 opera com instruções de 32 bits as memórias utilizadas para dados e programa terão também que ser do mesmo tipo.

Para garantir capacidade de armazenamento de imagens a cores com várias resoluções e permitir ainda o seu processamento e compressão, são utilizadas até 20 integrados de memória de 128 Kbytes, num total de 640 Kwords (640Kx4 bytes). Estes integrados têm um "pinout" compatível com os das memórias de 32 Kbytes, apesar de

possuirem mais quatro pinos que estas. Este facto permite a intermutação entre duas configurações de memória, consoante a resolução e o tipo de imagens que o utilizador pretenda adquirir e processar, sem exigir qualquer alteração física do circuito. A figura seguinte ilustra as duas configurações de memória possíveis.

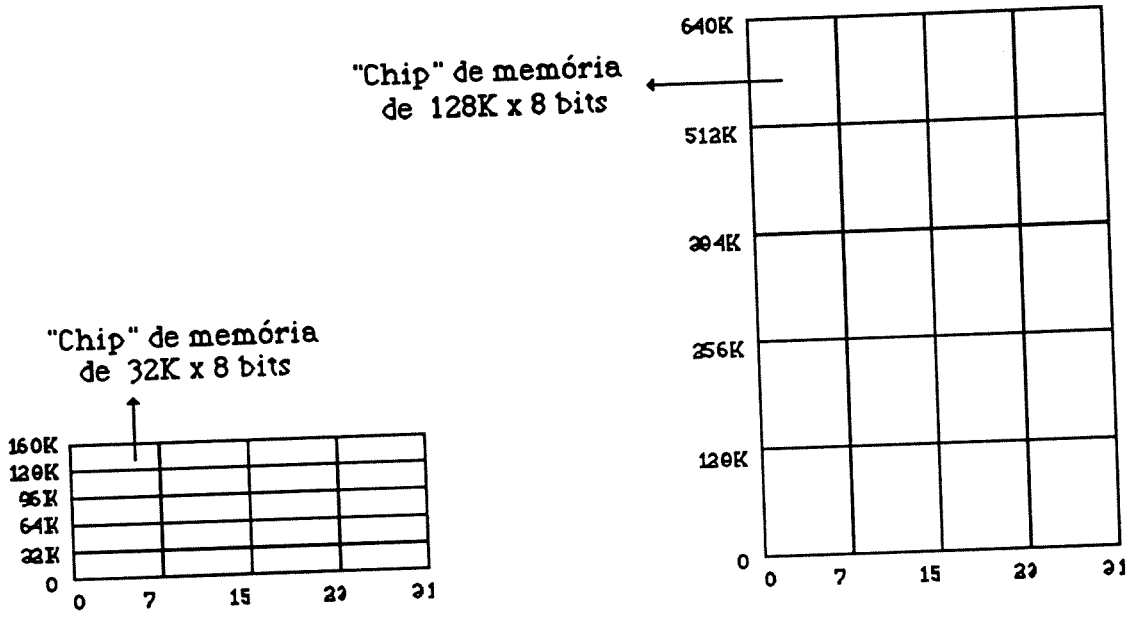


Figura 4.2 - Configurações alternativas da memória.

A compatibilidade entre os endereçamentos para estas duas configurações de memória é assegurada por um comutador físico que indica ao circuito de descodificação de endereços a configuração que está a ser utilizada.

4.2.1.2.1 Acesso à Memória pelo TMS320C30

Para aproveitar as capacidades deste processador são utilizadas memórias rápidas de forma a não introduzir ciclos de espera no processo de acesso à memória.

Estando o C30 configurado para trabalhar numa arquitectura de "Bank Switch" o processador mantém estáveis as linhas mais significativas de endereços, enquanto estão a ser feitos acessos ao mesmo banco de memória. Quando se dá uma mudança de banco de memória, o C30 insere um ciclo de espera compensando assim o tempo de descodificação das PAL's. O tamanho do banco de memória é definido no registo "Expansion Bus Control" [C30USG], mapeado na memória do C30, onde se deverá indicar se este é de 32 Kwords ou de 128 Kwords.

4.2.1.2.2 Acesso à Memória pelo PC-AT

O barramento de dados do PC-AT é de 16 bits, podendo aceder a bytes ou a palavras de 16 bits. Estando a memória vista pelo C30 organizada em palavras de 32 bits, cada palavra terá que ser partida em quatro bytes pois só assim é possível num acesso do AT, aceder a 16 bits de cada vez ou a cada byte individualmente.

Foi reservado 1 Mbyte do espaço de endereçamento do AT, que é de 16 Mbyte, o que equivale a 256 Kwords do espaço de endereçamento do C30, apresentando-se a memória do barramento principal do C30, vista pelo AT, representada na figura 4.3.

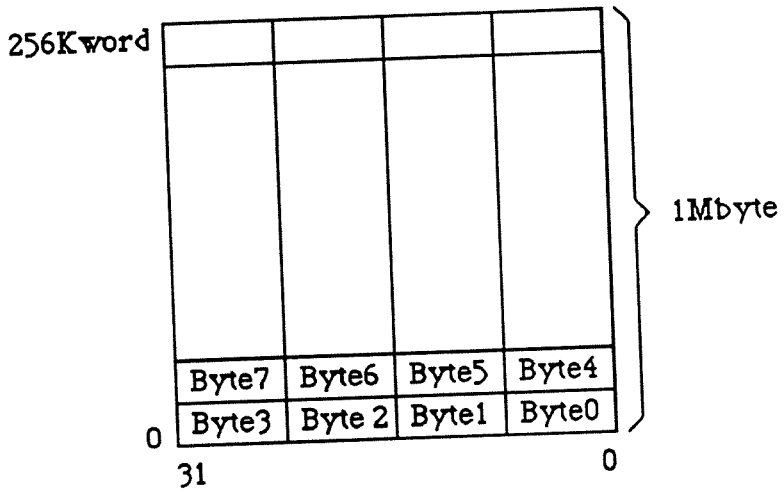


Figura 4.3 - Memória vista pelo PC-AT.

Para permitir o acesso a 8 Mwords do espaço de endereçamento do C30 é feito um endereçamento paginado. Considerando páginas de memória de 256 Kwords do C30 (isto é, 1 Mbyte do AT), são necessárias 32 páginas de memória.

O endereçamento do AT a uma página é feito através de um endereço base da placa, que selecciona um acesso à placa, ao qual se junta o endereço relativo na página. Para aceder a estas 32 páginas utiliza-se um Registo de Controlo, que contém os cinco bits mais significativos do endereço que, em conjunto com o endereço relativo na página, formam o endereço da memória. Este registo encontra-se mapeado como um porto do AT (ver 4.2.1.3).

Uma vez que a memória da placa está situada no barramento principal do C30, é feito um pedido de HOLD deste para que o AT possa aceder à memória. Como consequência

do pedido de HOLD o C30 liberta o barramento, colocando-o em alta impedância. Este pedido é feito através de um dos bits de controlo do Registo de Controlo ou quando é detectada uma selecção da placa.

4.2.1.2.3 Mapas de Memória

Apresentam-se a seguir os mapas de memória do barramento principal e do barramento auxiliar do sistema (isto é, do C30). Note-se que o barramento auxiliar com MEMSTRB activo está completamente desocupado.

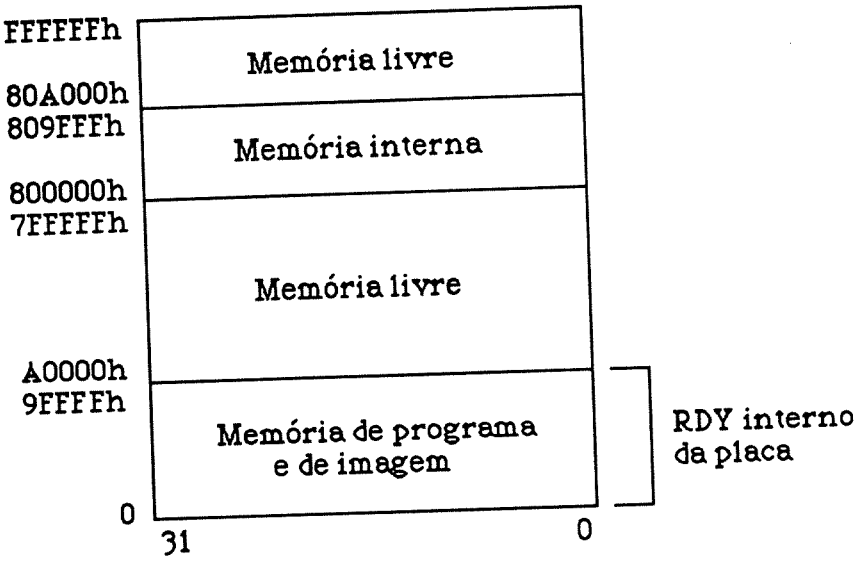


Figura 4.4 - Mapa de memória do barramento principal.

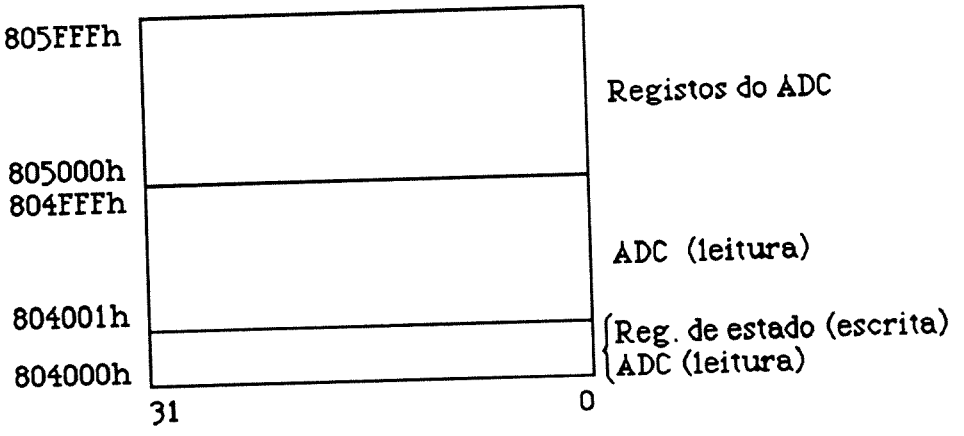


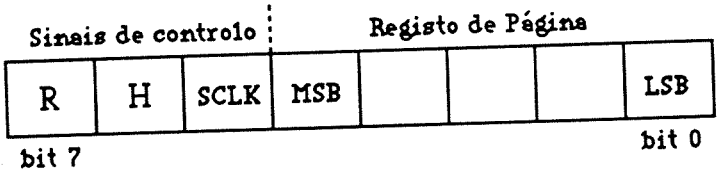
Figura 4.5 - Mapa de memória do barramento auxiliar com IOSTRB activo.

4.2.1.3 Interface com o PC-AT

Esta interface é responsável pelo acesso aos recursos da placa pelo PC-AT. Como foi referido, o acesso a esses recursos é feito de uma forma paginada, visto que o espaço de endereçamento do PC-AT é bastante limitado. Cada página vista pelo PC-AT ocupa o espaço de endereçamento de 1Mbyte, ou seja, 256 Kwords do espaço de endereçamento visto do C30, não sendo permitido o acesso acima de 8 Mwords, pois aí estão mapeados registos de controlo do C30 [C30USG].

Para controlo da placa existe um porto que é o registo CONTROLO/ESTADO (Registo de Controlo em escrita e Registo de Estado em leitura).

O Registo de Controlo só pode ser acedido pelo PC-AT em escrita e contém os sinais de controlo da placa, designados por RESET, HOLD, SCLK e ainda 5 bits para selecção da página de memória (em 32 possíveis), como se mostra na figura 4.6.



R : RESET
H : HOLD
SCLK : !SEL_CLOK (selecção de relógio)

Figura 4.6 - Registo de Controlo.

No arranque do sistema, é feito um "Reset" por "hardware" (pondo a zero todos os sinais de controlo do registo de controlo) ficando então o sistema adormecido para o AT. Para "acordar" o sistema e aceder à memória da placa, é necessário escrever no registo de controlo, seleccionando nesse momento a página de memória do C30 a que se quer aceder.

A interface com o PC-AT tem a capacidade de efectuar transferências bidireccionais de 8 ou 16 bits. Para aceder à memória da placa, mapeada no barramento principal do C30, é actuado o sinal "HOLD" do processador que liberta o barramento colocando-o em alta impedância, garantindo assim a não existência de contenção no barramento principal do sistema.

Situado no mesmo porto do registo de controlo, mas em leitura, está colocado o Registo de Estado que é responsável pela comunicação do estado corrente do sistema ao PC-AT, na sequência de um pedido de interrupção por parte desta interface.

Para permitir a troca de informação entre o computador hospedeiro e a placa, está implementada uma interrupção, que é desencadeada por uma escrita do C30 no registo de estado, a que deve estar associada uma rotina que permita ao PC saber a situação em que se encontra o sistema, por leitura do registo de estado. O nível da interrupção, assim como a linha de interrupção, são seleccionáveis por forma a não interferirem com outras tarefas do sistema.

Para maior flexibilidade do sistema, é possível controlar o primeiro endereço da placa de aquisição e o porto do PC-AT onde são mapeados o registo de controlo e o registo de estado, através de comutadores.

A placa é do tipo escravo, pelo que o PC-AT é responsável por todas as tarefas de comunicação com o sistema. Assim, a interacção entre o hospedeiro e o sistema faz-se da seguinte forma: se o hospedeiro deseja executar uma tarefa, deverá verificar se a rotina correspondente e, eventualmente, os dados para essa rotina, estão em memória; se isso não se verificar, deverá carregar a rotina e os dados e só depois mandar executar essa tarefa, através de uma escrita no registo de controlo. O C30 ao finalizar a execução deverá informar o PC-AT, através de um pedido de interrupção, de que terminou a tarefa e se a conseguiu executar com sucesso. Para isso, deverá escrever no registo de estado um código de estado. Após esta operação, o C30 pode entrar em ciclo de espera até novo comando.

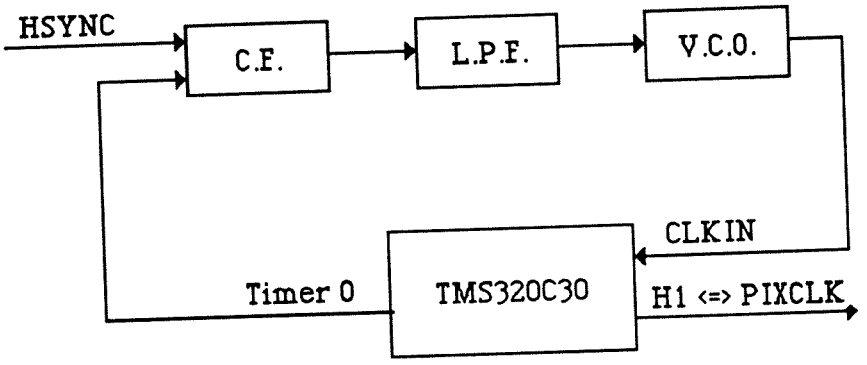
4.2.1.4 PLL

Devido à utilização desta placa tanto para a aquisição como para a compressão da imagem, existem duas necessidades distintas de relógio. Por um lado há necessidade de ter um relógio o mais rápido possível para aproveitar as potencialidades do processador nas operações de compressão de imagem, e por outro há necessidade de um relógio síncrono com um sinal exterior (sincronismo horizontal da câmara) e de frequência programável por forma a adquirir directamente pixels vindos do ADC, sendo possível ter várias resoluções de imagem. Para o primeiro caso é utilizado um oscilador fixo a cristal e para o segundo caso é utilizada uma malha de captura de fase ou PLL.

Para manter o sincronismo com a imagem durante a aquisição, uma malha de captura de fase (PLL) sintetiza um relógio síncrono para o ADC e para o CPU (C30), a partir da informação de sincronismo horizontal do sinal de vídeo.

4.2.1.4.1 Malha de Captura de Fase e Selecção de Relógio

Neste circuito precisamos de sintetizar uma frequência n vezes superior à frequência do sinal de sincronismo horizontal. Para isso, na malha de captura de fase, existe um divisor programável por n, que é implementado utilizando um dos recursos do C30 — o "Timer 0". Desta maneira o C30 fica incluído na malha de captura de fase, sendo o seu "Timer 0" o responsável pela divisão por n, tal como se apresenta na figura 4.7.



C. F. : Comparador de fase síncrono
L. P. F. : Filtro passa-baixo
V. C. O. : Oscilador controlado por tensão

Figura 4.7 - Malha de captura de fase.

Sendo utilizado um comparador de fase síncrono, o circuito reage de modo a que os dois sinais à entrada do comparador mantenham um desvio de fase nulo em relação ao bordo descendente destes sinais. Os pulsos de carga e descarga gerados por este comparador são aplicados a um filtro passa-baixo de forma a que à sua saída se tenha uma tensão contínua para controlar o VCO.

Para se seleccionar o relógio da PLL ou do oscilador a cristal existe um "multiplexer" que permite comutar o relógio quer pelo AT, quer pelo C30. Para comutar o relógio pelo AT é reservado o bit 5 do registo de controlo (SCLK). Para comutar o relógio pelo C30 é utlizado o sinal SELC30, originado no C30 através do porto 0, saída DX0. Dependendo do estado destes sinais o relógio do circuito vai ser o originado pela PLL ou por um oscilador fixo, como se apresenta a seguir:

SCLK	SELC30	RELOGIO
0	0	oscilador
0	1	oscilador
1	0	oscilador
1	1	PLL

4.2.1.4.1.1 Programação do "Timer 0"

Para configurar este "timer" como divisor programável por n, é necessário programar alguns registos de controlo deste dispositivo. Assim, existe o registo de período que terá que ser programado com o valor n/2, pois este registo só é incrementado de dois em dois ciclos de H1. Além deste, existe o registo de controlo global do funcionamento do "timer", que contém vários bits de controlo [C30USG]. Para que este dispositivo fique devidamente configurado o "Timer 0" deve ser configurado como saída pulsada, invertida para que a transição para zero do contador interno coincida com o impulso descendente da saída. Neste modo de funcionamento, a duração do impulso de saída é igual a um período de H1.

4.2.1.5 Aquisição de Imagem

Para digitalizar um sinal de vídeo é necessário extrair o sincronismo do sinal e utilizar uma malha de captura de fase para sincronizar o relógio do processador (C30) com o sinal de vídeo. Para além disso, há necessidade de um conversor analógico-digital (ADC).

O ADC está mapeado no barramento auxiliar do C30, o que permite fazer uma leitura directa dos pixels a adquirir pelo CPU. Assim, a transferência dos dados adquiridos para a memória é feita através do C30, pelo barramento auxiliar.

O circuito de extracção do sincronismo, a partir de um sinal de vídeo composto, retira a informação de sincronismo composto, sincronismo vertical, campo e "back-porch". Estes sinais são utilizados para sincronizar o "hardware" e "software", tornando assim possível a operação de digitalização de imagem.

O sinal de sincronismo horizontal, obtido do sincronismo composto por eliminação dos impulsos de serração, é utilizado para sintetizar o relógio que controla o sistema, como foi descrito no ponto anterior. O sinal de "back-porch" é utilizado para, durante

a parte invisível da imagem, fazer o "Clamping" e o "Zeroing" dos conversores A/D, permitindo assim a restauração da componente DC dos sinais de vídeo à saída do ADC. Os sinais de sincronismo vertical e de campo par/ímpar servem para sincronizar a rotina de aquisição de imagens, sendo fornecidos ao C30 através das "flags" XF0 e XF1, respectivamente.

Para conversão analógica-digital do sinal de vídeo é utilizado o circuito Bt253 da Brooktree [BT253]. Este permite a conversão simultânea de três sinais analógicos, neste caso os sinais R, G e B, para um formato digital programável, neste caso 8 bits por cor.

Pode-se assim seleccionar por "software" as entradas de vídeo a digitalizar e a entrada de onde será extraído o sincronismo, que pode ser qualquer uma das oito entradas analógicas, selecionando também o seu nível de detecção. Pode-se ainda seleccionar o formato de saída das cores. Este circuito permite ainda ajustar as referências de cor para os conversores A/D, pois possui seis fontes de corrente programáveis.

As saídas digitais do ADC estão mapeadas no barramento auxiliar do C30 ocupando em leitura as 4Kwords inferiores deste, com IOSTRB activo. Em escrita, a word inferior deste barramento é ocupada pelo Registo de Estado, como já foi referido. Quando se pretende digitalizar uma imagem basta fazer leituras na zona de endereçamento ocupada pelo ADC.

O sincronismo entre o ADC e o C30, assim como a garantia da presença de dados válidos no barramento quando há uma leitura do C30, são assegurados aplicando ao ADC o relógio de amostragem H3 gerado pelo C30.

Uma vez que as leituras do processador ocupam dois ciclos de relógio H1, este deverá possuir o dobro da frequência de amostragem do ADC.

O facto de haver duas entradas de vídeo RGB e Sincronismo, seleccionáveis por "software", permite que a placa esteja ligada a várias câmaras, o que tem particular interesse para um projecto de televigilância. Assim pode-se ligar à placa:

- 2 câmaras RGB
- 1 câmara RGB + 3 câmaras B/W
- 6 câmaras B/W

No caso de se pretender digitalizar uma imagem a cores, cada pixel de imagem ocupará uma "word" (32 bits), ficando as componentes R, G e B da imagem com a distribuição representada na figura 4.8.

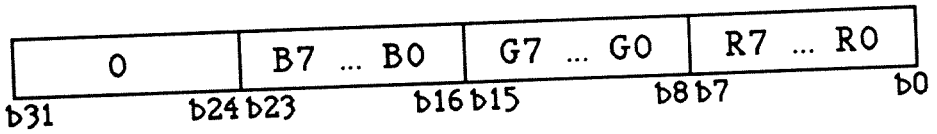


Figura 4.8 - Distribuição das componentes RGB numa "word".

No caso da digitalização de uma imagem proveniente de uma câmara monocromática cada pixel ocupa apenas 8 bits da "word" de 32. Dependendo da entrada a que a câmara estiver ligada (R1, G1, B1, R2, G2 ou B2) o pixel ocupará um byte diferente na "word", podendo ser necessário deslocá-lo por "software" para o byte menos significativo da "word". A figura 4.9 apresenta um exemplo que ilustra esta situação.

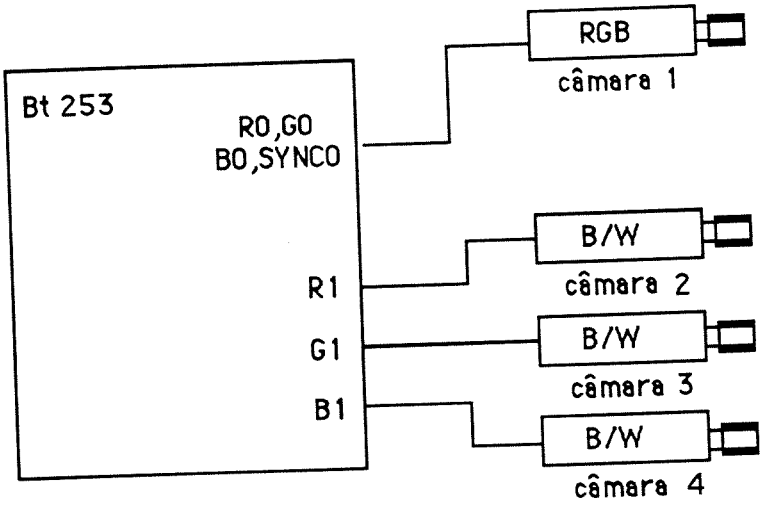


Figura 4.9 - Exemplo da ligação de uma câmara RGB e três câmaras B/W.

Se pretendermos seleccionar a câmara 3, de entre as câmaras ligadas, por "software" selecciona-se a entrada de vídeo 1 e a extracção do sincronismo de G1. Quando for feita a aquisição teremos a disposição de cada pixel na "word" representada na figura 4.10.

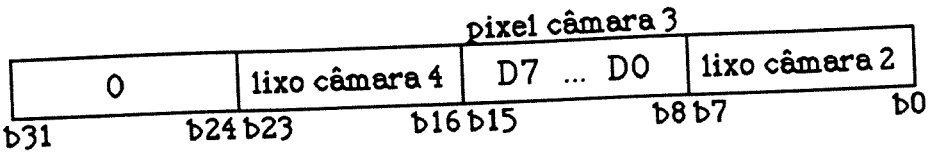


Figura 4.10 - Distribuição dos pixels aquiridos pela câmara 3 na "word".

Posteriormente, é necessário um tratamento dos dados por forma a deslocar o pixel adquirido para o byte menos significativo da "word" e mascarar os outros (figura 4.11).

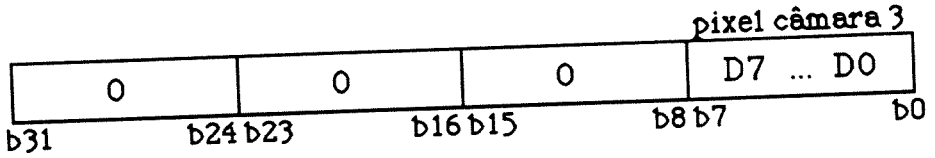


Figura 4.11 - Disposição do pixel aquirido após deslocamento e mascaramento.

Como a selecção das câmaras é comandada por "software", fazendo-se simultaneamente a selecção da linha para extracção de sincronismo, não é necessário a sincronização externa das câmaras.

A detecção de ausência de câmara pode ser feita por "software", lendo periodicamente o "Timer 1" do C30, que é o contador de sincronismo horizontal da câmara seleccionada, e observando se este é incrementado. Caso isso não se verifique, é porque a câmara seleccionada está desligada do sistema.

4.2.2 Sistema de Descompressão de Imagem

O diagrama de blocos da placa é o seguinte:

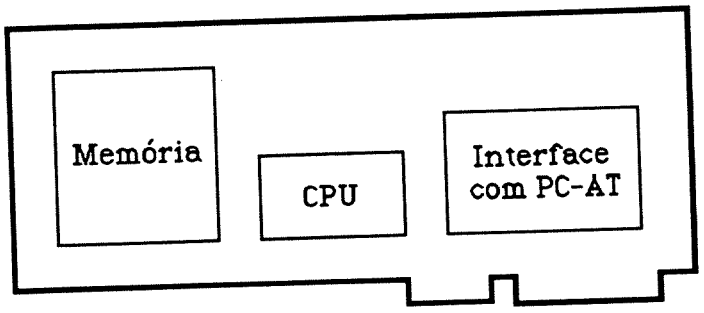


Figura 4.12 - Diagrama de blocos da placa de descompressão de imagem.

Tratando-se de uma versão reduzida da placa de aquisição e compressão de imagem, as suas especificações são em tudo idênticas às anteriores, excepto no que toca ao módulo de aquisição de imagem, que neste caso não existe. Sendo assim, o módulo da malha de captura de fase (PLL) não é necessário, pelo que também não existe.

Assim, fazem parte desta placa os seguintes ítems:

1) CPU
2) Memória
3) Interface com o PC-AT

Estes módulos são idênticos aos da placa de aquisição e compressão de imagem pelo que a sua descrição já se encontra no ponto anterior.

4.2.3 O DSP TMS320C30

O TMS320C30, da terceira geração da família TMS320, é um processador digital de sinal (DSP) de 32 bits, com um ciclo máquina de 60 ns, oferecendo uma capacidade de processamento de 33.3 MFLOPS (Milhões de Operações em Vírgula-Flutuante por Segundo) e 16.7 MIPS (Milhões de Instruções por Segundo) [C30USG], possíveis graças a um elevado grau de paralelismo interno e implementação, em "hardware", de funções que noutros processadores são realizadas por "software". Por outro lado, os periféricos implementados internamente (controlador de DMA, "Timers", Portos Série, etc.) contribuem também para a sua grande flexibilidade. No centro da sua arquitectura está a Unidade de Processamento Central (CPU).

A arquitectura geral (cf. figura 4.13), é do tipo Harvard no sentido em que, interna e externamente, tem múltiplos barramentos para aceder a instruções de programa, a dados ou executar transferências directas da memória (DMA). No entanto, também tem algo da arquitectura von Neumann, uma vez que o espaço de memória é unificado, e não há separação de espaços de programa e dados. Como resultado, o utilizador tem a livre escolha da colocação de programas e dados em qualquer posição de memória desejada.

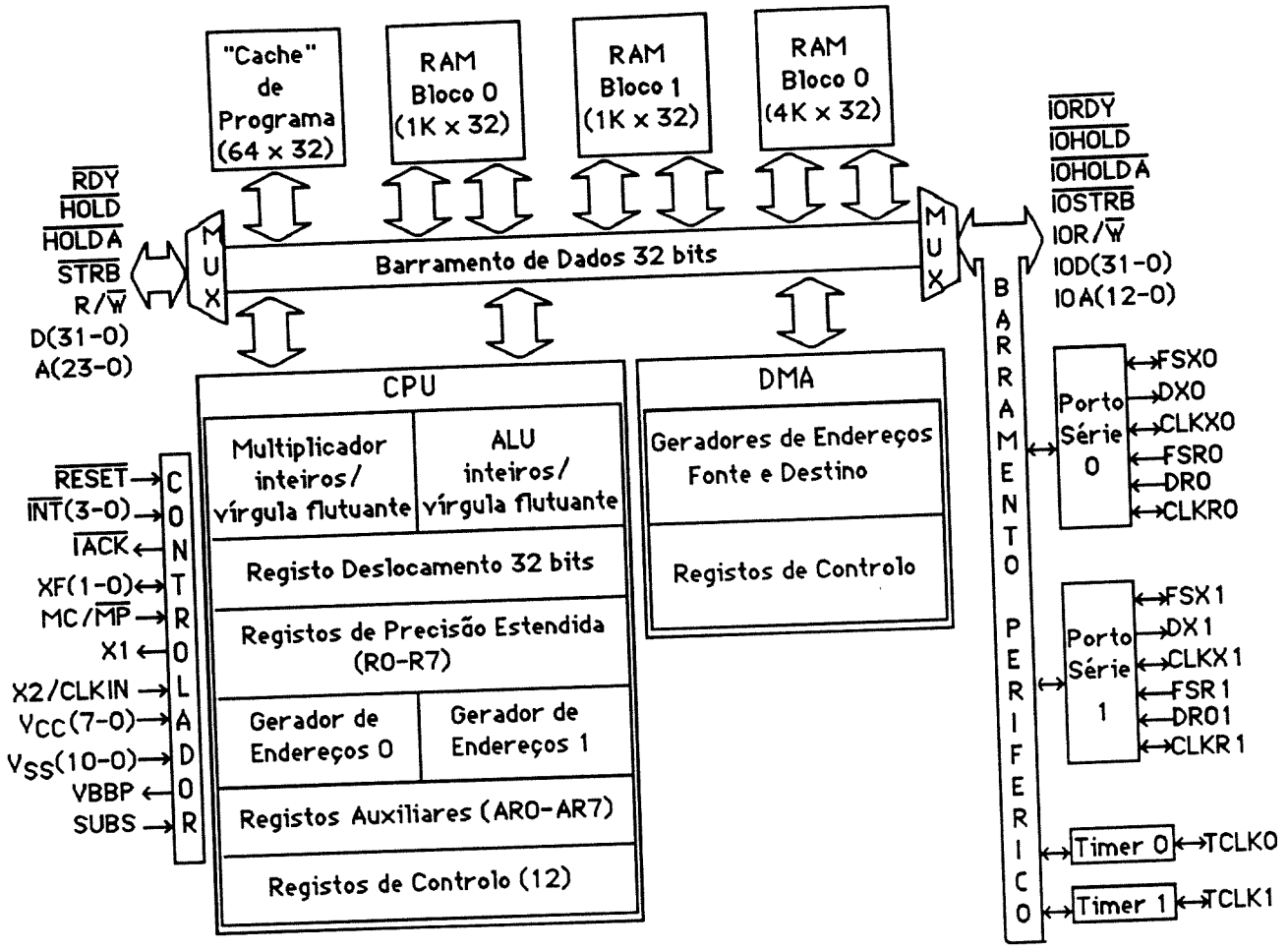


Figura 4.13 - Diagrama de blocos do TMS320C30.

Algumas das principais características da arquitectura do TMS320C30 são:

- Tempo de ciclo de 60 ns
- Dois barramentos de dados externos de 32 bits, podendo operar em paralelo
- Dois barramentos de endereço, para um espaço de memória de 16Mx32 bits
- Dois barramentos internos que permitem operações aritméticas e lógicas em paralelo
- Dois blocos de RAM interna de alta velocidade com 1Kx32 bits cada, de duplo acesso
- "Cache" interna de programa de 64x32 bits
- Unidade aritmética e lógica (ALU) e multiplicador de inteiros/vírgula-flutuante de 32/40 bits
- Registo de deslocamento de 32 bits permitindo efectuar operações de deslocamento de até 32 bits, num só ciclo
- Duas unidades aritméticas auxiliares (ARAUs) permitindo a geração de endereços sem utilização da ALU
- Vinte e oito registos, numa "Register File" multiporto

Das características relacionadas com o "hardware" destacam-se:

- Barramento principal de dados de 32 bits
- Barramento auxiliar de dados de 32 bits
- Barramento de endereçamento principal de 24 bits
- Barramento de endereçamento auxiliar de 13 bits
- Controlador interno de transferências por DMA
- Dois "timers"
- Dois portos série
- Quatro linhas de interrupção externas
- Duas "flags" configuráveis
- Operação simultânea do DMA e do CPU
- Package do tipo PGA de 180 pinos.

Das características relacionadas com o "software" destacam-se:

- Capacidade máxima de cálculo de 33.3 MFLOPS
- Velocidade máxima de execução de 16.7 MIPS
- Grande número de instruções paralelizáveis
- Instruções, geralmente, "single-cycle"
- Repetição de blocos de código
- Ciclos sem "overhead" associado ao seu controlo
- Multiplicação e operações da ALU paralelizáveis
- Instruções de dois e três operandos
- Instruções de salto standard (4 ciclos) e de salto atrasado (1 ciclo)
- Operações com inteiros, vírgula-flutuante e lógicas.

A arquitectura do TMS320C30 é baseada nos seguintes registos do CPU (cf. figura 4.14):

- Oito registos de precisão extendida, destinados a operações com inteiros (32 bits) ou em vírgula-flutuante (40 bits)
- Oito registos auxiliares, destinados à manipulação de endereços (24 bits) ou a operações com inteiros (32 bits)
- Doze registos de controlo e estado, com funções várias.

Os oito registos auxiliares e as duas ARAUs ("auxiliary register arithmetic units"), que operam em paralelo com o multiplicador e a ALU, permitem gerar dois endereços num único ciclo. Suportam endereçamento com deslocamentos, registos de indexação (IR0 e IR1), endereçamento circular e "bit-reversed".

REGISTO	FUNÇÃO
R0	Registo de precisão estendida
R1	Registo de precisão estendida
R2	Registo de precisão estendida
R3	Registo de precisão estendida
R4	Registo de precisão estendida
R5	Registo de precisão estendida
R6	Registo de precisão estendida
R7	Registo de precisão estendida
AR0	Registo Auxiliar
AR1	Registo Auxiliar
AR2	Registo Auxiliar
AR3	Registo Auxiliar
AR4	Registo Auxiliar
AR5	Registo Auxiliar
AR6	Registo Auxiliar
AR7	Registo Auxiliar
DP	Ponteiro de página de dados
IR0	Registo de índice 0
IR1	Registo de índice 1
BK	Tamanho de bloco
SP	Ponteiro de pilha
ST	Registo de Estado
IE	Desinibe interrupções CPU/DMA
IF	"Flags" de interrupções de CPU
IOF	"Flags" de I/O
RS	Endereço inicial de repetição
RE	Endereço final de repetição
RC	Contador de repetição
PC	Endereço da próx. inst. prog.

Figura 4.14 - Registos do CPU.

Além destes registos, existe ainda o "Program Counter" (PC) que contém o endereço da próxima instrução de programa a ser lida.

Muitas aplicações, como processamento de imagem, são difíceis de implementar num DSP das primeiras gerações, pois requerem um grande espaço de memória. Para satisfazer esta necessidade, o TMS320C30 oferece um espaço total de endereçamento de 16Mx32 bits, bastante superior ao dos DSPs de vírgula-fixa das gerações anteriores. Os espaços de programa, dados e I/O estão contidos neste espaço de endereçamento de 16Mword, divididas em várias zonas distintas. Os dois blocos de RAM internos têm cada

um 1Kx32 bits. Cada bloco RAM é capaz de suportar dois acessos de dados num único ciclo. Na figura 4.15 apresenta-se o mapa de memória do TMS320C30, no modo microprocessador.

O processador tem dois barramentos externos de dados, com palavras de 32 bits: o barramento principal e o barramento auxiliar. Associados a estes, estão os barramentos de endereçamento principal e auxiliar, de 24 e 13 linhas, respectivamente. Na figura 4.15 indicam-se as zonas de memória endereçadas por estes dois barramentos. O processador possui ainda dois barramentos internos. São possíveis operações simultâneas nos barramentos externos e internos.

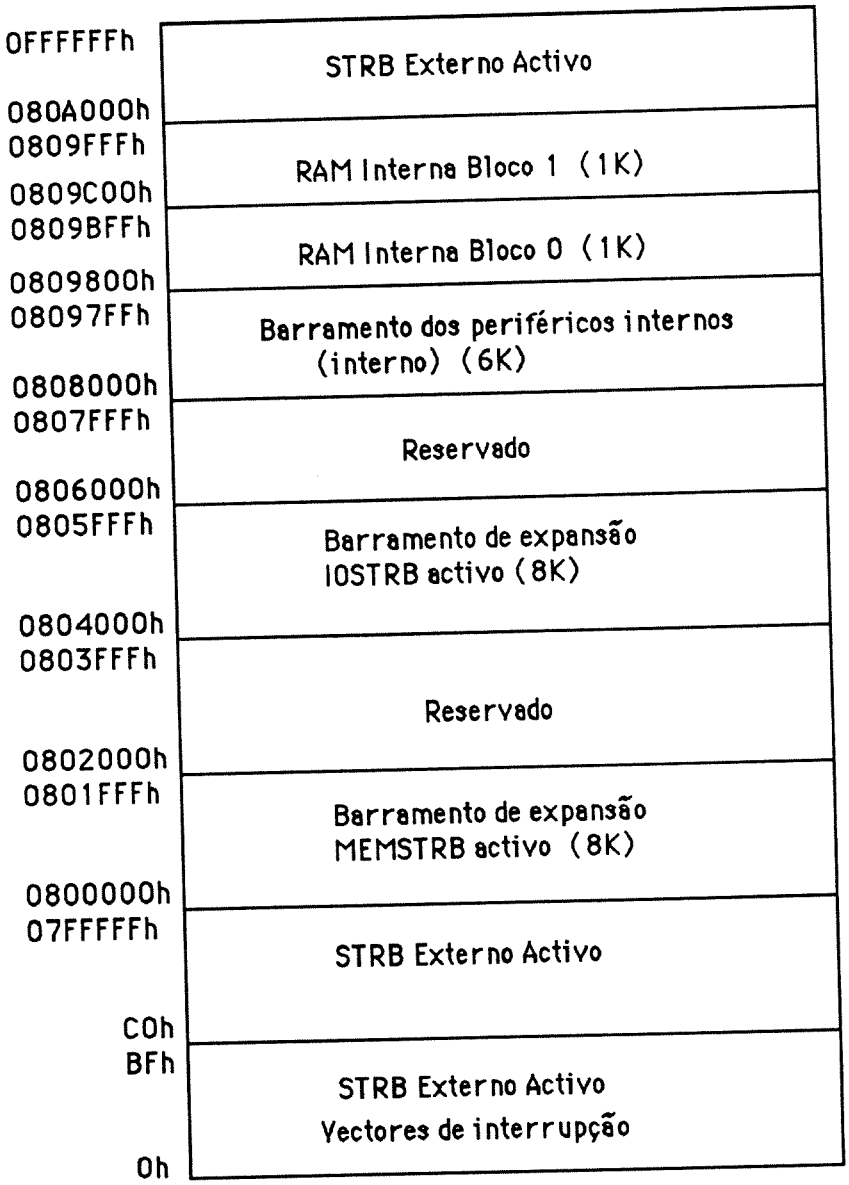


Figura 4.15 - Mapa de memória do TMS320C30.

Os periféricos internos do TMS320C30 são: dois "timers", 2 portos série, duas "flags" e um controlador de DMA. Estes periféricos são controlados por registos mapeados na memória. O controlador de DMA permite transferências de dados sem interferir com as operações do CPU.

Além das interrupções desencadeadas pelos periféricos internos existem quatro linhas de interrupção externas com uma hierarquia de prioridades (INT0 a INT3). Associadas às interrupções, bem como ao RESET, existem vectores mapeados no início da memória.

O conjunto de intruções está organizado nos seguintes grupos:

- "load and store instructions"
- "2-operand arithmetic instructions"
- "2-operand logical instructions"
- "3-operand arithmetic instructions"
- "3-operand logical instructions"
- "parallel operation instructions"
- "arithmetic/logical instructions with store instructions"
- "program control instructions"
- "interlocked operations instructions"

O TMS320C30 tem a capacidade de executar, num único ciclo, duas instruções em paralelo como, por exemplo, multiplicação e adição (ou subtracção), em dados do tipo inteiro ou vírgula-flutuante. As operações em vírgula-flutuante facilitam os cálculos e permitem manter a precisão. A aritmética de vírgula-flutuante é executada à mesma velocidade da aritmética inteira.

As instruções "load and store" incluem a possibilidade de carregar um registo condicionalmente. O conjunto de operações aritméticas e lógicas podem ter 2 operandos, um fonte e um destino, ou 3 operandos, dois fonte e um destino. Estas últimas permitem ler dois operandos da memória e/ou "register file" do CPU, num único ciclo. As instruções paralelizáveis oferecem um grande nível de paralelismo e flexibilidade. Incluem multiplicação/adição e a possibilidade de carregar e/ou guardar dois registos em paralelo. As instruções "arithmetic/logical with store" permitem executar uma instrução aritmética ou lógica entre um registo e um operando lido da memória, em paralelo com o "store" de um registo na memória. Também permitem operações rápidas em blocos de memória. Das instruções de

controlo de programa destacam-se dois tipos: "repeat modes" e "branching". Os modos de repetição permitem implementar ciclos sem o "overhead" associado ao controlo do ciclo por registos. As capacidades de "branching" incluem dois subconjuntos: "standard" e "delayed". Os primeiros, devido ao "pipeline" interno de quatro níveis, requerem quatro ciclos de execução. Os segundos garantem que é feito o "fetch" das três instruções seguintes antes de o PC ser modificado, logo estas são sempre executadas. O resultado é um "branch" que requer apenas um ciclo. As instruções "interlocked operations" suportam a comunicação entre múltiplos processadores, através do uso de sinais externos que permitem a implementação de mecanismos de sincronização.

Uma informação detalhada sobre o TMS320C30 pode ser obtida em [C30USG].

4.3 "SOFTWARE" do Codificador

O algoritmo implementado é baseado na quantificação vectorial adptativa de blocos de dimensão e forma variável, conforme descrito em 3.4. Neste ponto é feita uma apresentação na perspectiva de implementação. Na figura 4.16 apresenta-se o diagrama de blocos do codificador.

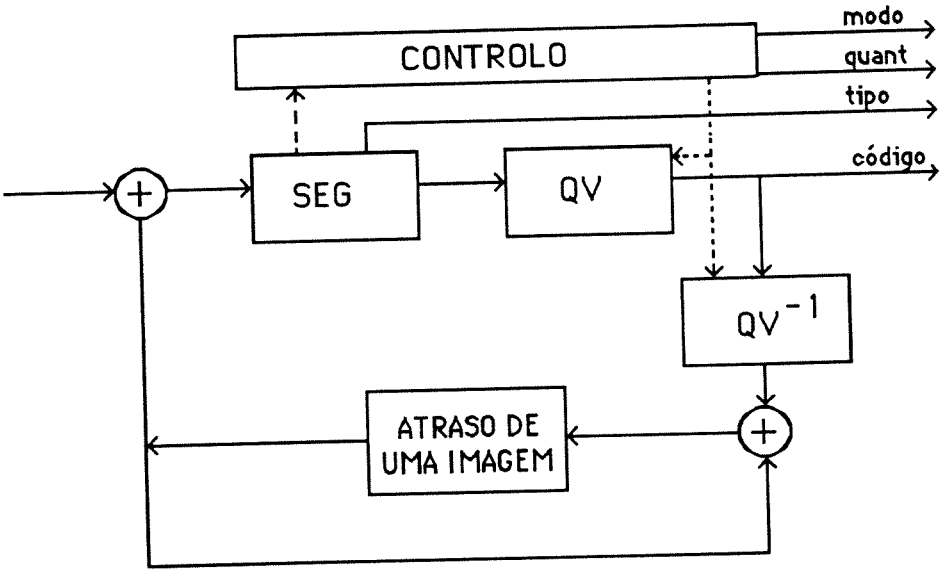


Figura 4.16 - Diagrama de blocos do codificador.

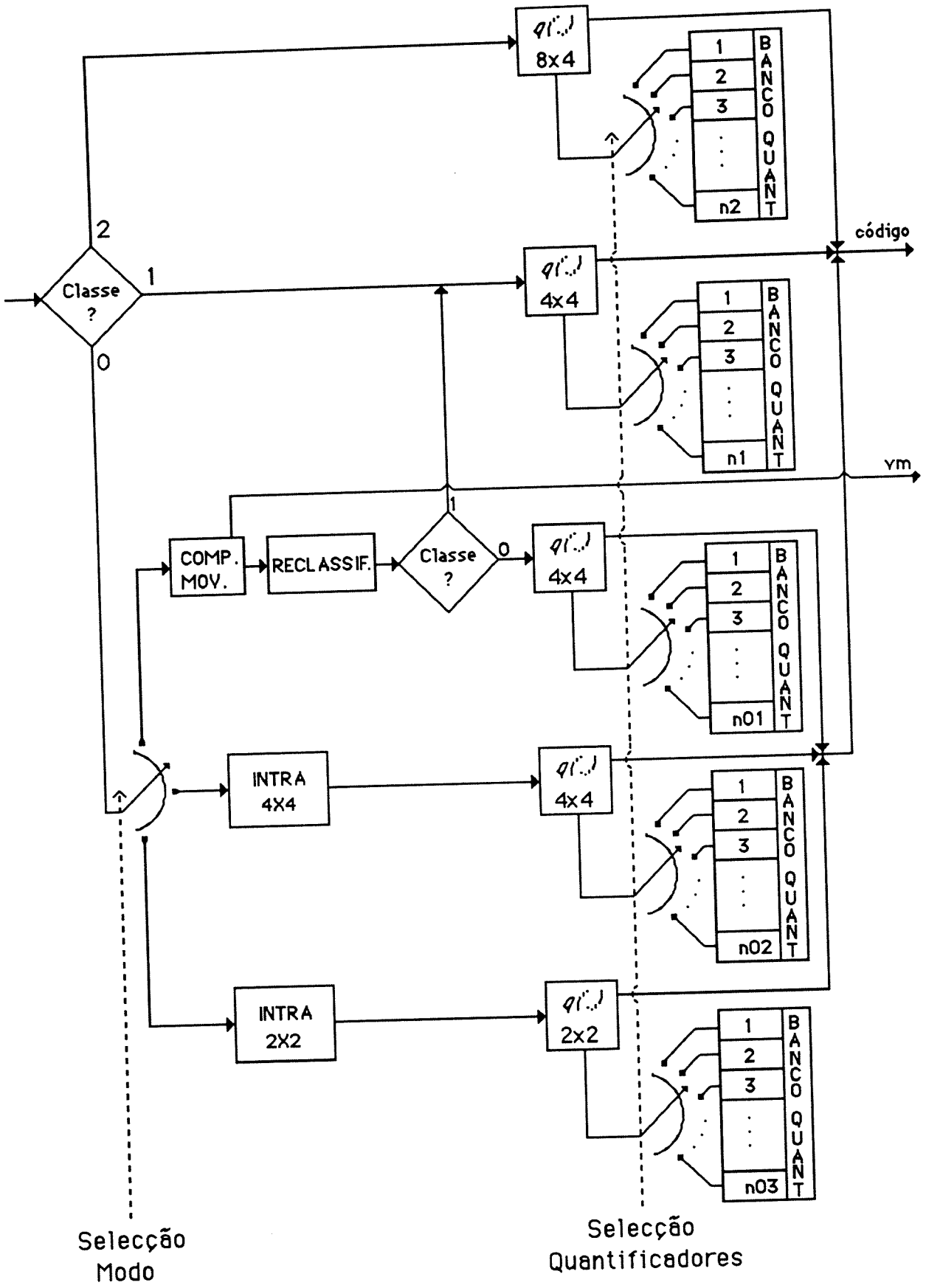


Figura 4.17 - Bloco QV.

O sinal de entrada é constituído por imagens QCIF (180x144 pixels), monocromáticas, a uma cadência de 8.3 Hz. Uma vez que a aquisição da imagem é feita pelo C30, o tempo disponível para aquisição e compressão de uma imagem é de 1/8.3Hz, ou seja, 120 milisegundos. Consequentemente, a codificação está limitada não só pela taxa de transmissão (64 Kbit/s) mas também pelo tempo de processamento.

Optou-se por fazer o controlo imagem a imagem da taxa gerada, utilizando um "buffer" dimensionado para uma imagem. Numa fase posterior, poderá pensar-se num método mais complexo de gestão do "buffer".

Por outro lado, devido à limitação imposta pelo tempo de processamento disponível, e para tirar partido de toda a capacidade do DSP, todas as rotinas implementadas foram escritas directamente em Assembly do C30 [C30ALT]. Para as testar foi utilizado o "TMS320C30 Emulator" [C30EMU], que é uma placa de controlo responsável pela interpretação de comandos enviados pelo PC e conversão destes em sequências de sinal apropriadas para controlar o TMS320C30 do sistema do utilizador, permitindo fazer o "debug" de software e hardware. A execução dos programas é feita à velocidade do DSP e a monitorização dá-nos acesso quer aos registos internos quer à memória, interna e externa, do DSP.

4.3.1 Segmentação da Imagem Diferença

O algoritmo de segmentação [COR90] é aplicado à imagem diferença, dividindo-a em 11 tipos de blocos, de forma quadrada ou rectangular, de diferentes dimensões (ver Tabela 1 da página 3.15). A segmentação é efectuada em duas fases que se passam a descrever.

PRIMEIRA FASE:
A imagem diferença é dividida em blocos de 4x4 pixels, formando uma matriz de blocos de dimensão 45x36, para uma imagem de 180x144 pixels. Estes blocos são classificados em Tipo 0 (muito activos) e Tipo 1 (pouco activos) consoante o valor do seu índice de actividade seja superior ou inferior a um limiar pré-determinado. A medida de actividade usada é a variância do bloco (VAR), dada pela diferença entre a média quadrática (MQ) e o quadrado da média (M) do bloco, de acordo com

$$VAR = MQ - M^2 \tag{4.1}$$

sendo

$$MQ = \frac{1}{16} \sum_{i=0}^{3} \sum_{j=0}^{3} (x_{ij})^2 \tag{4.2}$$

e

$$M = \frac{1}{16} \sum_{i=0}^{3} \sum_{j=0}^{3} x_{ij} \tag{4.3}$$

sendo x_{ij} a amplitude do pixel na linha i, coluna j do bloco. O valor do limiar de actividade foi colocado a 80.

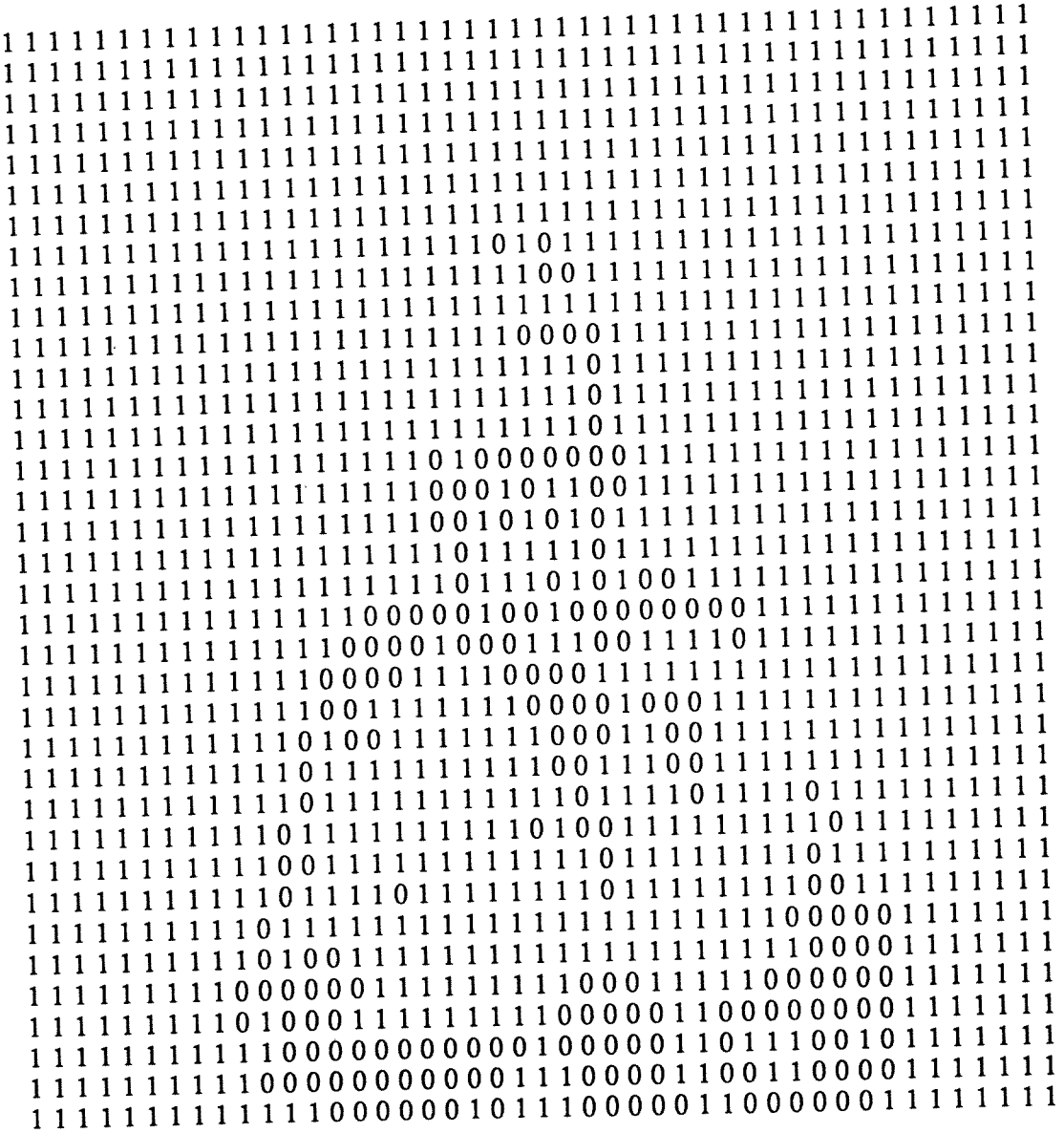


Figura 4.18 - Distribuição de blocos Tipo 0 e Tipo 1.

Para fazer a divisão em blocos 4x4, são processadas 4 linhas de cada vez. Como cada linha é constituída por 180 pixels, são processados 180x4=720 pixels. Tendo cada bloco de RAM interna 1Kwords, as 4 linhas são passadas para memória interna, o que vai permitir um processamento mais rápido. Assim, enquanto estão a ser processadas 4 linhas de um bloco de RAM interna, as 4 linhas seguintes são passadas para o outro bloco de RAM interna, por DMA. Desta forma, vão sendo processadas 4 linhas de cada vez, alternando o bloco de RAM interna em que estão guardadas.

Os valores da média e da média quadrática são também guardados nesta fase, para serem usados posteriormente. Na figura 4.18 apresenta-se a distribuição de blocos para uma imagem diferença da sequência "Trevor". Sendo esta sequência muito activa, o número de blocos do Tipo 0 é considerável.

SEGUNDA FASE:
Nesta fase, é feito um varrimento sistemático ao longo de linhas e colunas da matriz de blocos 4x4, com o objectivo de agrupar os blocos Tipo 1, considerados pouco activos, de modo a formar macroblocos com uma área superior. Note-se que todos os tipos de blocos considerados podem ser construídos a partir de blocos 4x4.

Assim, para cada bloco Tipo 1, agrupando-o com os vizinhos à direita e abaixo também de Tipo 1, procura-se obter blocos quadrados ou rectangulares de maior dimensão, tipos 2 a 10, tendo uma variância inferior ao limiar. Isto é feito calculando a variância para blocos de dimensão sucessivamente superior até ser ultrapassado o limiar, atingido o bloco de maior dimensão (Tipo 10), ou não existirem blocos do Tipo 1 contíguos que permitam formar um bloco de dimensão superior (isto é, se os blocos contíguos necessários para formar o bloco de ordem superior forem do Tipo 0 ou pertencerem já a um macrobloco criado nas linhas anteriores). Nos casos dos blocos rectangulares do tipo 2, 5 e 8, uma vez que estes não estão contidos nos blocos do tipo seguinte, 3, 6 e 9, é necessário testar também estes últimos antes de parar. Nestes casos, se ambos os blocos satisfizerem o critério de actividade escolhe-se o que tiver um índice de actividade inferior. Uma vez que este passo da segmentação é efectuado para cada bloco 4x4, todos os blocos que são agrupados são marcados com o tipo de bloco obtido. Na figura 4.19 mostra-se os blocos resultantes da aplicação deste procedimento à matriz de blocos apresentada na figura 4.18.

O cálculo da variância dos blocos é feito rápidamente usando os valores da média e média quadrática dos blocos 4x4, que foram calculadas e guardadas na primeira fase da segmentação. Para um macrobloco tipo X constituído por n blocos 4x4, a sua

variância VARX pode ser calculada da seguinte forma:

$$VARX = MQX - MX^2 = \frac{1}{n}\sum_{i=1}^{n} MQi - \left(\frac{1}{n}\sum_{i=1}^{n} Mi\right)^2 \qquad (4.4)$$

sendo MQi e Mi, respectivamente, a média quadrática e a média do bloco i, constituinte do macrobloco.

```
a a a a a a a a a a a a a a a 8 8 8 8 8 8 8 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 8 8 8 8 8 8 8 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 8 8 8 8 8 8 8 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 8 8 8 8 8 8 8 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 9 9 9 9 5 5 5 5 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 9 9 9 9 5 5 5 5 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 9 9 9 9 2 2 2 2 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 9 9 9 9 3 0 3 0 a a a a a a a a a a a a a a a 9 9 9 9 3
a a a a a a a a a a a a a a a 9 9 9 9 3 3 3 0 0 5 5 5 5 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 9 9 9 9 3 3 2 2 1 5 5 5 5 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 9 9 9 9 3 3 0 0 0 0 6 6 3 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 9 9 9 9 3 3 4 4 3 0 6 6 3 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 6 6 4 4 3 3 4 4 3 0 6 6 3 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 6 6 4 4 1 3 2 2 1 0 6 6 3 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 6 6 0 1 0 0 0 0 0 0 0 6 6 a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a 6 6 0 0 0 1 0 2 2 0 0 6 6 a a a a a a a a a a a a a a a a
a a a a a a a a 9 9 9 9 5 5 5 5 4 4 0 0 3 0 3 0 3 0 3 6 6 5 5 5 5 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 5 5 5 5 4 4 3 0 3 3 3 1 3 0 3 6 6 5 5 5 5 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 4 4 2 2 2 2 3 0 3 3 1 0 1 0 1 0 0 2 2 2 2 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 4 4 1 0 0 0 0 3 0 0 3 0 0 0 0 0 0 0 3 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 3 1 0 0 0 0 3 0 0 0 1 3 1 0 0 5 5 5 5 0 3 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 3 0 0 0 0 3 3 6 6 0 0 0 0 2 2 5 5 5 5 6 6 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 1 0 0 2 2 3 3 6 6 3 0 0 0 0 3 0 0 0 3 6 6 a a a a a a a a 9 9 9 9
a a a a a a a a 9 9 9 9 0 3 0 0 6 6 3 6 6 3 3 0 0 0 3 3 0 0 3 6 6 a a a a a a a a 9 9 9 9
a a a a a a a a 6 6 4 4 0 3 6 6 6 6 3 6 6 3 3 0 0 4 4 3 0 0 3 6 6 3 2 2 8 8 8 8 8 8 8 3
a a a a a a a a 6 6 4 4 0 3 6 6 6 6 3 6 6 3 2 2 0 4 4 6 6 0 3 6 6 3 0 1 8 8 8 8 8 8 8 3
a a a a a a a a 6 6 3 0 1 3 6 6 6 6 3 6 6 3 0 3 0 0 3 6 6 6 6 6 6 2 2 0 8 8 8 8 8 8 8 3
a a a a a a a a 6 6 3 0 0 3 6 6 2 2 3 6 6 3 3 3 3 0 3 6 6 6 6 6 6 3 0 1 8 8 8 8 8 8 8 3
a a a a a a a a 4 4 1 0 3 3 4 4 0 6 6 6 6 3 3 3 3 0 3 6 6 6 6 6 6 3 0 0 2 2 9 9 9 9 6 6 3
a a a a a a a a 4 4 0 3 3 1 4 4 3 6 6 6 6 3 3 3 4 4 3 4 4 6 6 4 4 0 0 0 0 0 9 9 9 9 6 6 3
a a a a a a a a 3 1 0 3 0 0 2 2 3 6 6 6 6 3 3 3 4 4 1 4 4 4 4 4 4 1 0 0 0 0 9 9 9 9 6 6 3
a a a a a a a a 3 0 0 0 0 0 0 4 4 6 6 6 6 3 3 3 0 0 0 2 2 4 4 1 0 0 0 0 0 0 0 9 9 9 9 6 6 3
8 8 8 8 8 8 8 8 3 0 3 0 0 0 1 4 4 2 2 6 6 1 3 0 0 0 0 0 4 4 0 0 0 0 0 0 0 0 0 9 9 9 9 6 6 3
8 8 8 8 8 8 8 8 3 3 3 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 4 4 0 2 2 3 0 0 1 0 9 9 9 9 6 6 3
8 8 8 8 8 8 8 8 3 3 0 0 0 0 0 0 0 0 0 0 0 3 3 3 0 0 0 0 2 2 0 0 1 3 0 0 0 0 9 9 9 9 6 6 3
8 8 8 8 8 8 8 8 3 2 2 2 2 0 0 0 0 0 0 1 0 3 1 3 0 0 0 0 0 2 2 0 0 0 0 0 0 1 9 9 9 9 6 6 3
```

Figura 4.19 - Matriz de blocos resultantes.

A única informação relativa à segmentação que é necessário guardar, e transmitir de modo a permitir ao descodificador reconstruir a imagem original, é o tipo do bloco.

Uma vez que a imagem é varrida segundo uma ordem pré-definida (da esquerda para a direita e de cima para baixo), não é necessário transmitir os endereços dos blocos para descrever a segmentação, pois o descodificador fará a reconstrução da imagem segundo a mesma ordem.

Nesta fase, é também guardado o número de blocos que irão ser convertidos para cada uma das 3 classes, e ainda o número de ciclos que vão ser gastos na conversão Tipo→Classe e na reconstrução da imagem original, uma vez que este é fixo e conhecido, para cada tipo de bloco, excepto para os blocos do tipo 0 em que depende do modo de codificação seleccionado. Esta informação vai ser utilizada para a selecção do modo de codificação e do quantificador a usar para cada classe de blocos.

O número de ciclos de execução da rotina de cálculo da imagem diferença e segmentação é variável, dependendo do número e tipo dos blocos resultantes, no entanto com as imagens de teste verificou-se que este se situava entre 291835 ciclos (8l7.51 ms), para a imagem 6 da sequência "Miss America", e 293940 ciclos (17.64 ms), para a imagem 2 da sequência "Trevor".

4.3.2 Codificação dos Blocos

Após a segmentação, e antes da quantificação vectorial, os blocos resultantes são convertidos para uma das três classes seguintes:

Classe 0: Blocos inicialmente considerados muito activos (Tipo 0). Dimensão 4x4.

Classe 1: Blocos quadrados considerados pouco activos. Dimensão 4x4.

Classe 2: Blocos rectangulares considerados pouco activos. Dimensão 8x4.

Assim, temos a seguinte conversão:

Tipo	→	Classe
0		0
1		1
2		2
3		2
4		1
5		2
6		2
7		1
8		2
9		2
10		1

Esta conversão é feita, para cada bloco, imediatamente antes da quantificação. Os valores resultantes para os pixels de cada bloco são colocados num vector de dimensão 16 ou 32, sendo dispostos por linhas consecutivas. Este vector, que é o vector de dados de entrada para o quantificador vectorial, está guardado na memória interna para tornar mais rápida a operação de busca na tabela de códigos, durante a qual este vai ser acedido continuamente.

Apresenta-se no quadro 4.1 o número de blocos de cada classe para as sete imagens diferença da sequência de teste "Trevor" (sendo 80 o limiar de actividade usado na segmentação), bem como a percentagem de área de imagem coberta pelos blocos da classe 0.

QUADRO 4.1 - Número de blocos por classe, para a sequência de teste "Trevor".

Nº Img.	Nº total blocos	Nº blocos Classe 2	Nº blocos Classe 1	Nº blocos Classe 0	Area blocos Classe 0 (%)
1	450	93	61	296	18.3
2	393	126	53	214	13.2
3	524	109	70	345	21.3
4	562	80	61	421	26.0
5	575	103	52	420	25.9
6	572	100	66	406	25.1
7	523	101	69	353	21.8

Apresenta-se a seguir um quadro idêntico para a sequência "Miss America", uma sequência com actividade bastante inferior à anterior. Pode-se constatar o muito menor número de blocos da Classe 0 e, consequentemente, um número total de blocos consideravelmente inferior, visto que a segmentação das imagens resultou num elevado número de blocos de grandes dimensões.

QUADRO 4.2 - Número de blocos por classe, para a sequência de teste "Miss America".

Nº Img.	Nº total blocos	Nº blocos Classe 2	Nº blocos Classe 1	Nº blocos Classe 0	Area blocos Classe 0 (%)
1	233	103	36	94	5.8
2	198	114	36	48	3.0
3	249	120	44	85	5.2
4	285	102	39	144	8.9
5	168	105	32	31	1.9
6	171	87	28	56	3.5
7	175	104	30	41	2.5

4.3.2.1 Conversão dos Blocos Tipo 1...10 (Blocos de Fundo) para as Classes 1 e 2

Os blocos de fundo, tipo 1 a 10, são convertidos para as classes 1 e 2, consoante se trate de blocos quadrados ou rectangulares. A conversão é conseguida efectuando uma filtragem passa-baixo seguida de decimação.

Assim, os blocos tipos 4, 7 e 10, quadrados, são reduzidos a blocos Classe 1, com as dimensões do tipo 1, dividindo-os em sub-blocos de 2x2, 4x4 e 8x8, respectivamente, e substituindo cada sub-bloco pela sua média.

Os blocos tipos 3, 5, 6, 8 e 9 são reduzidos a blocos Classe 2, com as dimensões do tipo 2. Para os blocos tipos 5 e 8 é usado o mesmo procedimento, dividindo-os em sub-blocos de dimensões 2x2 e 4x4 e substituindo cada um destes pela sua média. Os blocos do tipo 3 necessitam de ser rodados para serem transformados em blocos Classe 2. Para os blocos tipos 6 e 9, é necessário dividi-los em sub-blocos de dimensões 2x2 e 4x4, substituir cada um destes pela sua média e rodá-los de modo a obter blocos da Classe 2. A rotação destes blocos, corresponde simplesmente a serem lidos por colunas em vez de linhas.

Apresenta-se no quadro seguinte o número de ciclos, e tempo de execução correspondente, para as rotinas de conversão implementadas no C30.

Quadro 4.3 - Tempos de execução das rotinas de conversão.

Tipo	Classe	Nº Ciclos	Tempo Execução (μs)
1	1	61	3.66
2	2	109	6.54
3	2	108	6.48
4	1	296	17.76
5	2	584	35.04
6	2	587	35.22
7	1	27	1.62
8	2	43	2.58
9	2	44	2.64
10	1	88	5.28

Note-se que para os blocos tipos 7, 8, 9 e 10, o número de ciclos gastos na conversão diminui drásticamente, pois estes são divididos em sub-blocos de dimensão 4x4, cuja média já foi calculada e guardada durante a segmentação. Repare-se também que para

os blocos Tipo 1 o número de ciclos é superior, pois ao serem copiados para o vector de dados é necessário aceder aos pixels da imagem diferença, guardados na memória externa, enquanto que as médias estão guardadas em memória interna.

4.3.2.2 Conversão dos Blocos Tipo 0 (Blocos Activos) para a Classe 0

Para converter um bloco do Tipo 0 para a Classe 0 não é necessário efectuar qualquer processamento, pois estes são equivalentes. No entanto, como após a segmentação o codificador dispõe da informação necessária para escolher o modo de codificação a aplicar a estes blocos:

1) codificação intra-imagem por média prevista (4x4),
2) codificação inter-imagem (4x4) com compensação de movimento, ou
3) codificação intra-imagem por média prevista (2x2),

é efectuado, nesta altura, o processamento necessário para os preparar para a quantificação.

No modo 1, é calculada uma estimativa da média do bloco com base nos valores dos pixels fronteira dos blocos adjacentes (ver 3.4.3.2.2). Este valor é, em seguida, subtraído aos valores dos pixels do bloco correspondente da imagem de entrada.

No modo 2, é necessário efectuar a operação de estimação de movimento, descrita em 4.3.2.2.1, para obter os valores dos pixels do bloco diferença compensado em movimento e os vectores de movimento.

O modo 3, é idêntico ao primeiro mas em seguida divide-se o bloco em quatro blocos de dimensões 2x2. Para facilitar a quantificação, os pixels são dispostos de outra forma no vector de dados de entrada para o quantificador, de modo a que os elementos de cada um dos quatro blocos fiquem em sequência.

Quadro 4.4 - Tempos de execução das rotinas de conversão dos blocos Classe 0.

Classe	Modo Codificação	Nº Ciclos	Tempo Execução (µs)
0	1	133	7.98
0	2	2740	164.40
0	3	137	8.22

4.3.2.2.1 Estimação do Movimento

O algoritmo de estimação de movimento adoptado foi o "three-step", que oferece um bom compromisso entre qualidade e complexidade. A busca é efectuada numa gama de ±6 pixels e o tamanho do bloco é de 4x4 pixels (bloco Classe 0). O critério de "match" é o de minimização da diferença absoluta, de acordo com (4.5), onde b_k representa o bloco na imagem actual e b_{k-1} representa o bloco na imagem anterior

$$M(i,j) = \sum_{m=1}^{4} \sum_{n=1}^{4} | b_k(m,n) - b_{k-1}(m+i,n+j) | \quad , -6 \le i, j \le +6 \qquad (4.5)$$

Assumindo um deslocamento máximo de 6 pixels, o algoritmo itera em três passos para o erro absoluto mínimo.

A área de busca tem as dimensões (4 + 2 x 6) x (4 + 2 x 6) = 16 x 16.

Passo 1 - O bloco actual é comparado com 9 blocos dentro da janela de busca (16 x 16) na imagem anterior, nas seguintes posições:

(-3, -3)	(0, -3)	(3, -3)
(-3, 0)	(0, 0)	(3, 0)
(-3, 3)	(0, 3)	(3, 3)

A posição da estimativa não deslocada é (0, 0).

Passo 2 - No segundo passo é usado um novo modelo de busca. O melhor "match" do passo 1 é o centro deste modelo:

(-2, -2)	(0, -2)	(2, -2)
(-2, 0)	(0, 0)	(2, 0)
(-2, 2)	(0, 2)	(2, 2)

Passo 3 - A posição do melhor "match" no passo 2 é a posição central do modelo de busca do terceiro e último passo:

(-1, -1)	(0, -1)	(1, -1)
(-1, 0)	(0, 0)	(1, 0)
(-1, 1)	(0, 1)	(1, 1)

O melhor "match" do passo 3 é o erro de "match" mínimo resultante.

Na figura 4.20 mostra-se um exemplo do processo de busca deste algoritmo de três passos.

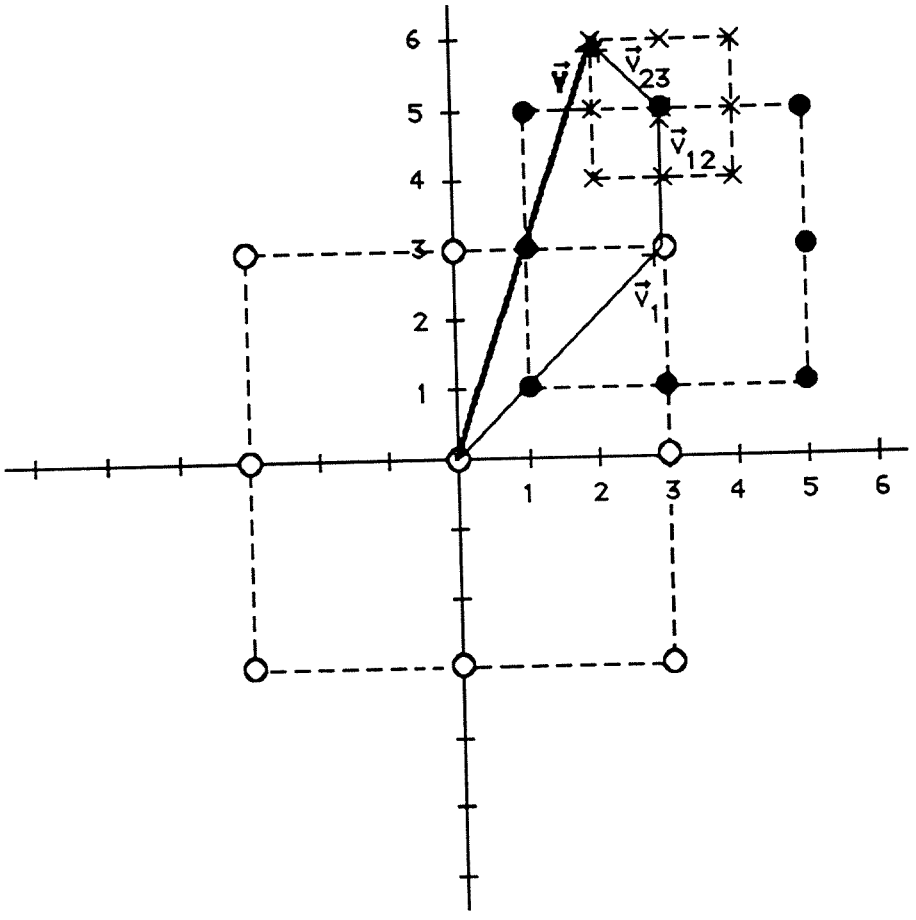


Figura 4.20 - Exemplo do processo de busca do algoritmo "three-step".

4.3.3 Quantificação Vectorial dos Blocos

A quantificação vectorial dos blocos consiste em fazer corresponder ao vector de dados, que contém os valores dos elementos do bloco que se quer quantificar, o vector da tabela de códigos mais próximo, de acordo com uma dada medida de distorção. Para

isso, é necessário criar previamente as tabelas de códigos apropriadas para a quantificação dos blocos de cada classe.

As tabelas de códigos usadas, foram obtidas a partir de uma sequência de treino constituída por 24 imagens da sequência "Miss America", 16 imagens da sequência "Clair" e 16 imagens da sequência "Split", usando o algoritmo LBG [LIN80].

4.3.3.1 Processo de Busca na Tabela de Códigos

A quantificação de um vector implica a busca exaustiva, na tabela de códigos, do vizinho mais próximo do vector a quantificar. Sendo o critério de distorção adoptado o erro quadrático médio, R a taxa de transmissão em bit/pixel, K a dimensão dos vectores, x o vector a quantificar, y_i o i-ésimo vector da tabela de códigos e $N=2^{RK}$ o número de vectores desta, o quantificador terá de efectuar o cálculo de N medidas de distorção do tipo:

$$d_i = \sum_{j=1}^{K} (x_j - y_{ij})^2 \qquad i = 1, ..., N \qquad (4.6)$$

e escolher o vector que dê origem à menor distorção.

Esta operação implica a realização de N*(K-1) somas, N*K subtracções, N*K multiplicações e N-1 comparações. Supondo que todas as operações têm o mesmo peso para o processador, a complexidade pode ser avaliada pelo número total, que neste caso é de 3KN-1. A memória necessária é a ocupada por N vectores, ou seja, se considerarmos como unidade de memória o espaço ocupado por uma componente do vector a quantificar, N*K unidades de memória.

A implementação da rotina de busca sequencial exaustiva na tabela de códigos, no C30, resulta num esforço de cálculo de 22 + (2 * K + 10) * N ciclos. Apresenta-se no quadro seguinte o número de ciclos e tempo de execução correspondentes, para o caso da busca numa tabela de 64, 128, 256, 512 e 1024 vectores, sendo a dimensão do vector 4 (blocos 2x2 Classe 0), 16 (blocos 4x4 da Classe 0 ou 1) ou 32 (blocos 8x4 da Classe 2).

Quadro 4.5 - Tempos de execução das rotinas de busca exaustiva.

K	Tempo de Busca (μs)				
	N = 64	N = 128	N = 256	N = 512	N = 1024
4	70.44	139.56	277.80	554.28	1107.40
16	162.60	323.90	646.44	1291.50	2581.80
32	285.48	569.64	1137.96	2274.60	4547.88

No quadro 4.6, apresenta-se o tempo necessário para quantificar as imagens da sequência de teste "Trevor", sendo os blocos de fundo quantificados com tabelas de 64 vectores (6 bit/bloco) e os blocos da Classe 0 com tabelas de 64 vectores (6 bit/bloco) e 128 vectores (7 bit/bloco).

Quadro 4.6 - Tempos (em milisegundos) gastos na quantificação dos blocos resultantes da segmentação das imagens da sequência de teste "Trevor".

Nº Img.	Blocos Fundo N = 64	Classe 0 N = 64	Total	Classe 0 N = 128	Total
1	36.47	48.13	84.6	95.87	132.3
2	44.59	34.80	79.4	69.31	113.9
3	42.50	56.10	98.6	111.75	154.2
4	32.76	68.45	101.2	136.36	169.1
5	37.86	68.29	106.2	136.04	173.9
6	39.28	66.02	105.3	131.50	170.8
7	40.05	57.40	97.5	114.34	154.4

Sendo 120 ms o tempo disponível para aquisição e compresssão de uma imagem, é claro que o tamanho das tabelas de códigos teria que ser muito limitado (igual ou inferior a 64), o que resultaria numa codificação de muito baixa qualidade. Assim, tornou-se necessário usar técnicas de redução da complexidade da busca na tabela de códigos.

4.3.3.1.1 Técnicas de Redução da Complexidade de Busca na Tabela de Códigos

A satisfação do critério de qualidade a atingir implica a necessidade da quantificação com um número mais elevado de bits/pixel. Sempre que o número de operações se torna, por este motivo, demasiadamente alto é necessário recorrer a técnicas de

redução de complexidade. Estas técnicas podem ser de dois tipos: técnicas óptimas (que não provocam degradação do desempenho do quantificador) e técnicas não óptimas (que introduzem degradação de qualidade na quantificação).

4.3.3.1.1.1 Técnicas Optimas

DESDOBRAMENTO DA FORMULA DO ERRO QUADRATICO MÉDIO

O desdobramento da fórmula do erro quadrático médio, permite reduzir o número de operações.

$$(x_j - y_{ij})^2 = x_j^2 + 2*x_j*y_{ij} + y_{ij}^2 \qquad (4.7)$$

O primeiro termo, sendo igual para as N medidas de distorção pode ser ignorado. A soma dos últimos termos (y_{ij}^2) para todas as componentes, dependendo apenas da tabela de códigos pode estar guardado sob a forma de uma tabela em memória.

A expressão pode assim ser calculada pela fórmula:

$$d_i = 2 * \sum_{j=1}^{K} (x_j * y_{ij}) + \sum_{j=1}^{K} (y_{ij}^2) \qquad i = 1, ..., N \qquad (4.8)$$

através de N*K somas, N*(K+1) multiplicações e N-1 comparações, ou seja, 2N(K+1)-1 operações. A economia assim obtida é de aproximadamente 33%:

$$\frac{3KN-1-2N*(K+1)+1}{3KN-1} = \frac{(K-2)*N}{3KN-1} \approx \frac{(K-2)}{3K} \approx \frac{1}{3}$$

Esta rotina, implementada no C30, requer um esforço de cálculo de 27+(K+13)*N ciclos. Apresenta-se no quadro 4.7 o número de ciclos e tempo de execução correspondentes, para o caso da busca numa tabela de 64, 128, 256, 512 e 1024 vectores, sendo a dimensão do vector 4, 16 ou 32. Comparando com os tempos apresentados no quadro 4.5, pode-se verificar que a redução do tempo de busca aproxima-se de 31% no caso de vectores de dimensão 16 e 39% no caso de vectores de dimensão 32. No entanto, no caso dos vectores de dimensão 4 (blocos 2x2) esta redução é apenas da ordem dos 5%.

Quadro 4.7 - Tempos de execução das rotinas de busca exaustiva simplificada.

K	Tempo de Busca (μs)				
	N = 64	N = 128	N = 256	N = 512	N = 1024
4	66.90	132.18	262.74	523.86	1046.10
16	112.98	224.34	447.06	892.50	1783.38
32	174.42	347.22	692.82	1384.02	2766.42

DISTORÇÃO PARCIAL

Uma outra técnica de redução de operações óptima é a técnica da distorção parcial. Esta técnica consiste em determinar a medida de distorção entre o vector a codificar e o vector da tabela de códigos (equação (4.6)) enquanto esta não ultrapassa o valor da mínima distorção encontrada até ao momento. Esta técnica pode ser utilizada conjuntamente com a anterior, minimizando o número de operações do cálculo do primeiro somatório da equação (4.8).

Verifica-se que para a maior parte dos vectores o cálculo do somatório é terminado após poucas parcelas, pelo que o número total de operações pode ser reduzido. No entanto, o número de comparações e a necessidade de tomar decisões aumenta. Consequentemente, a sua implementação no C30 não é muito eficiente, pois vai verificar-se uma sobrecarga associada ao controlo do ciclo por registos. Quando o número de repetições de um ciclo é fixo (como é o caso dos métodos anteriores), o C30 executa-o sem qualquer sobrecarga pois possui instruções de controlo que permitem a repetição de um bloco de instruções um número de vezes fixado à partida. No caso desta técnica, como o ciclo é controlado por uma condição, é necessário testar essa condição e em seguida usar uma instrução de salto condicional. As instruções de salto do C30, devido ao "pipeline" interno de 4 níveis, requerem 4 ciclos de execução. Sendo assim, para grande parte dos vectores, o facto de se efectuarem menos cálculos não compensa o acréscimo de ciclos de execução exigidos pelo controlo do ciclo. Assim, a economia resultante da utilização deste método não foi significativa. Este método tem ainda a desvantagem de não ter um tempo de execução fixo.

DESIGUALDADE TRIANGULAR

Uma terceira técnica óptima de redução de complexidade é baseada na desigualdade triangular [CHE89]. Sendo o critério de distorção utilizado na quantificação o erro

quadrático médio, a medida de distorção não é mais do que a norma l_2 de um espaço Euclidiano, ou seja, a distância entre os pontos deste espaço representados pelos dois vectores. Considerando três pontos do espaço (x, y_1 e y_2) e d(x,y) representar a distância entre os pontos x e y, sendo calculada de acordo com a equação (4.6), pela desigualdade triangular:

$$d(y_1,y_2) <= d(x,y_1) + d(x,y_2) \qquad (4.9)$$

quaisquer que sejam x, y_1 e y_2, verificando-se a igualdade apenas quando x se encontra sobre a recta que une y_1 e y_2.

Suponha-se que x é o vector a quantificar e y_1 e y_2 dois vectores da tabela de códigos sendo y_1 o vector de mínima distorção até ao momento e y_2 aquele com que a próxima medida de distorção vai ser calculada . Se $d(x,y_2) < d(x,y_1)$, então pela equação (4.9):

$$d(y_1,y_2) <= 2*d(x,y_1) \qquad (4.10)$$

A utilização desta propriedade pode permitir uma redução significativa da complexidade de codificação. Se existir no codificador uma tabela com as distorções entre todos os vectores da tabela de códigos, o quantificador pode, com base na equação (4.10), eliminar a operação de medida de distorção entre o vector de entrada e o vector da tabela de códigos em causa, gastando apenas o tempo necessário para efectuar uma comparação e tomar uma decisão. A eficiência deste método é dependente do tipo de vectores que constituem a tabela de códigos, tornando-se superior se estes estiverem pouco concentrados no espaço.

O seu único incoveniente é o de requerer a utilização das tabelas de distorção que ocuparão N^2 unidades de memória (de facto, como a matriz de distorções é simétrica, bastaria guardar metade dos elementos, $N^2/2$; no entanto, isto implica uma sobrecarga no cálculo dos endereços). Deste modo, para uma tabela de 1024 vectores seria necessário guardar uma tabela de distorções com 1024x1024=1M elementos. o que não é possível visto a memória do sistema estar limitada a 640 Kwords.

Esta técnica foi testada com tabelas de código de 256 vectores, tendo-se verificado uma redução média do número de ciclos de execução, em relação à busca exaustiva, de apenas 6%, aproximadamente. Este método, tal como o anterior, tem a desvantagem de necessitar do controlo de ciclos por registos, e decisão baseada numa condição. É provável que com tabelas de códigos de tamanho superior a sua eficiência aumente,

no entanto, os requisitos em termos de memória tornam-se impraticáveis.

4.3.3.1.1.2 Técnicas Não-Optimas

A utilização das técnicas óptimas analisadas pode não ser suficiente para reduzir o número de operações até ao limite suportável pelo codificador, sobretudo quando se utilizam tabelas de códigos grandes, pois como vimos a complexidade inerente à busca exaustiva aumenta exponencialmente com as dimensões da tabela de códigos.

Torna-se nesta situação necessário recorrer a métodos de redução de complexidade não óptimos, isto é, que originam uma degradação do desempenho do codificador. Existem diversas técnicas deste tipo [GRA84], podendo dividir-se em duas categorias:

- nos métodos pertencentes à primeira, determinados parâmetros do vector a codificar são extraídos e transmitidos separadamente. São designadas na literatura por "Product Code Vector Quantizers". Alguns exemplos de métodos desta categoria são os quantificadores do tipo "Ganho-Forma" [SABI84], os normalizados por média e variância [BAK83], os codificadores multi-andar [JUA82], etc..

- os métodos pertencentes à segunda categoria, geram tabelas de códigos com desempenho inferior ao obtido com a busca exaustiva, mas com uma determinada estrutura que permite a codificação com um número reduzido de operações. Os dois principais exemplos deste tipo de técnicas são a utilização de quantificadores do tipo "lattice" [GER79] e de quantificadores em árvore [GRA82a].

BUSCA EM ARVORES

Neste trabalho, foi dada especial atenção aos codificadores em árvore (TSVQ - "Tree Searched Vector Quantizer"). Neste tipo de quantificadores, a tabela de códigos é constituida por uma série de andares, estruturados em árvore de acordo com a figura 4.21 Em cada andar o vector é quantificado por uma sub-tabela de códigos, da tabela de códigos global desse nível da árvore, que constituem ramos do nó associado ao vector da tabela de códigos com que o vector foi codificado no andar anterior. A figura 4.21 ilustra a codificação numa árvore com três andares com 4 níveis no primeiro andar, 2 em cada sub-tabela de códigos do segundo e 4 em cada sub-tabela de códigos do terceiro, resultando assim uma tabela de códigos global de 32 níveis.

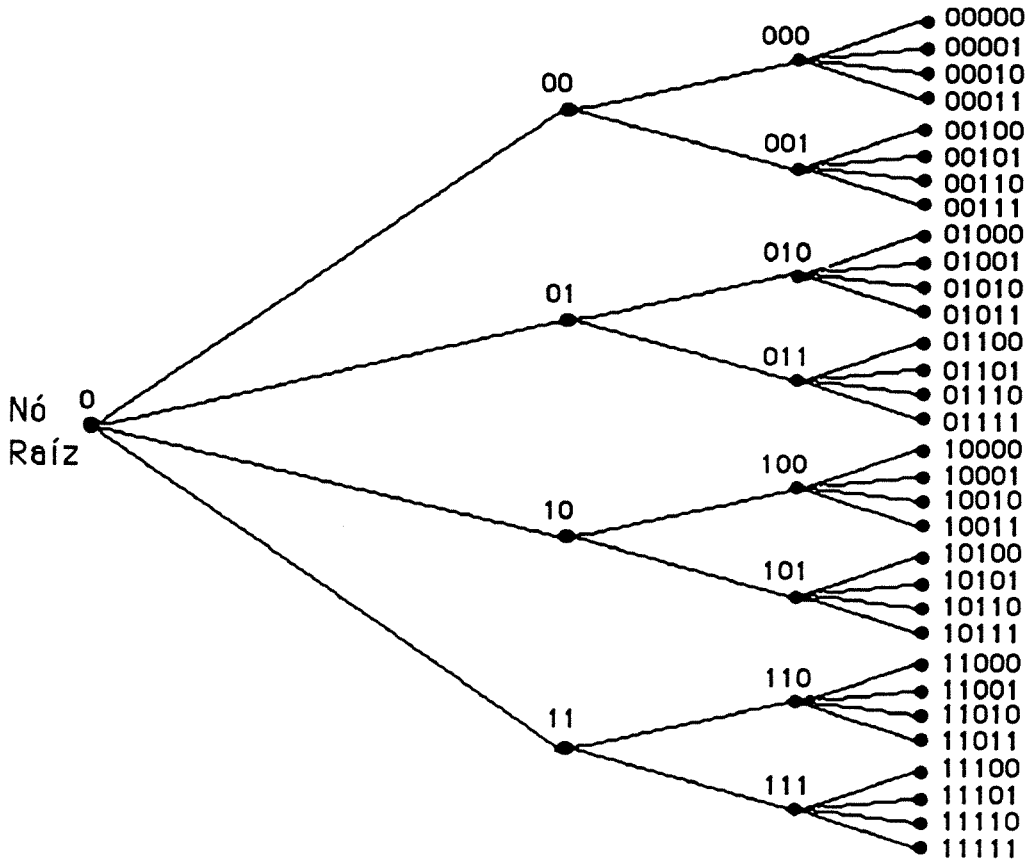


Figura 4.21 - Exemplo de uma tabela de códigos estruturada em árvore, com 3 andares e 4 níveis no primeiro andar, 2 níveis em cada sub-tabela do segundo andar e 4 níveis em cada sub-tabela do terceiro andar.

O desempenho destes quantificadores melhora com a diminuição do número de andares da árvore e o aumento do número de níveis por sub-tabela de códigos. O caso ideal da busca exaustiva pode ser visto como uma árvore de um andar e número de níveis dessa sub-tabela de códigos igual ao número de níveis total do quantificador. A complexidade aumenta, no entanto, da mesma forma, tornando-se necessário atingir uma solução de compromisso para a estrutura da árvore. O mínimo de complexidade pode ser atingido com uma árvore binária. Sendo $N=2^{RK}$ o número de níveis do quantificador a implementar, esta terá $\log_2 N$ andares e o quantificador, calculando duas medidas de distorção por andar calculará, no total, $2*\log_2 N$ medidas de distorção. A complexidade é assim proporcional às dimensões da tabela de códigos (2*RK), em vez de aumentar exponencialmente com estas (como acontece com a busca exaustiva). No quadro seguinte apresenta-se o número de cálculos do erro quadrático médio necessários para diversas estruturas de árvore, desde a busca binária até à busca exaustiva, para uma tabela de códigos final de 1024 vectores.

Quadro 4.8 - Estruturas em árvore possíveis, desde a busca binária à busca exaustiva, e correspondente número de avaliações do erro quadrático médio.

Nº de andares	Nº de níveis por andar	Nº de avaliações do e. q. m.
10	2,2,2,2,2,2,2,2,2,2	20 (busca binária)
5	4,4,4,4,4	20
4	8,8,8,2	26
3	16,16,4	36
2	32,32	64
1	1024	1024 (busca exaustiva)

O esforço de cálculo necessário para efectuar a busca numa árvore com m andares, sendo N_i o número de níveis de cada sub-tabela do andar i, é de

$$22*m + (2 * K + 10) \sum_{i=1}^{m} N_i \qquad \text{ciclos (de execução do C30)}$$

O número de níveis do quantificador a implementar é dado pelo produto do número de níveis de cada sub-tabela.

$$N = \prod_{i=1}^{m} N_i \tag{4.11}$$

Sendo $N_i = 2^{R_i}$, a taxa de um TSVQ com uma estrutura (R_1, R_2, ..., R_m) é

$$R = \sum_{i=1}^{m} R_i \qquad \text{bit/bloco} \tag{4.12}$$

Apresenta-se no quadro 4.9 os tempos de execução para tabelas de código em árvore com 2 andares, que correspondem a $44 + (2 * K + 10) (N_1 + N_2)$ ciclos.

Quadro 4.9 - Tempos de execução das rotinas de busca em árvore com 2 andares.

K	Tempo de Busca (μs)				
	N = 64 (8*8)	N = 128 (16*8)	N = 256 (16*16)	N = 512 (32*16)	N = 1024 (32*32)
4	19.92	28.56	37.20	54.48	71.76
16	42.96	63.12	83.28	123.60	163.92
32	73.68	109.20	144.72	215.76	286.80

A utilização de tabelas de códigos em árvore requer uma maior capacidade de memória no codificador pois é necessário armazenar as tabelas de códigos dos andares intermédios. O aumento máximo verifica-se para as árvores binárias sendo neste caso a memória necessária duplicada. No descodificador isto não acontece pois apenas é necessário guardar as tabelas de códigos do andar final.

Esta técnica pode ser conjugada com qualquer uma das técnicas óptimas apresentadas anteriormente, que podem ser utilizadas na codificação dentro de cada sub-tabela de códigos. No caso de se combinar esta técnica com o desdobramento da fórmula do erro quadrático médio, o esforço de cálculo é reduzido para

$$27*m + (K + 13) \sum_{i=1}^{m} N_i \qquad \text{ciclos (de execução do C30)}$$

Os tempos de execução correspondentes para tabelas de código em árvore com 2 andares, são apresentados no quadro seguinte.

Quadro 4.10 - Tempos de execução das rotinas de busca em árvore com 2 andares, utilizando a técnica do desdobramento da fórmula do erro quadrático médio.

K	Tempo de Busca (μs)				
	N = 64 (8*8)	N = 128 (16*8)	N = 256 (16*16)	N = 512 (32*16)	N = 1024 (32*32)
4	19.56	27.72	35.88	52.20	68.52
16	31.08	45.00	58.92	86.76	114.60
32	46.44	68.04	89.64	132.84	176.04

Comparando estes tempos com os do quadro 4.9, pode verificar-se uma redução de, aproximadamente, 30% para vectores de dimensão 16 e 38% para vectores de dimensão 32. No caso dos vectores de dimensão 4, esta redução situa-se entre 1.8% (N=64) e 4.5% (N=1024).

A introdução das árvores elimina o problema da complexidade de implementação. Com efeito, para uma árvore de 32x32 verificaram-se resultados muito próximos dos obtidos com a busca exaustiva (menos um bit ou até superiores). Sendo a degradação mínima é extremamente vantajosa a utilização desta técnica. Alterando a profundidade dos andares da árvore é possivel codificar com o número de bits que se

desejar, embora a qualidade decresça com o aumento do número de andares. Os resultados são sempre, no entanto, bastante superiores aos obtidos com a busca exaustiva que, para a capacidade máxima do processador utilizado, se limita a tabelas de códigos de 64 níveis.

BUSCA NUMA JANELA

Em [KOH88] é apresentada uma outra técnica subóptima baseada na ordenação da tabela de códigos pela média dos seus elementos. Associada à tabela de códigos existirá também uma tabela de médias, com as médias dos vectores de códigos ordenadas por ordem crescente. Na codificação começa-se por determinar o vector de código com média mais próxima da do vector de dados; em seguida, selecciona-se uma janela de w elementos, centrada nesse vector, e faz-se uma busca exaustiva dentro dessa janela.

Este algoritmo é chamado "Sliding Search Vector Quantizer" (SSVQ). Este método pode ser aplicado usando outra característica do bloco em vez da média, no entanto, os autores referem que resultados experimentais mostraram que a média é uma característica com boa capacidade de separação dos vectores de dados.

Note-se que este algoritmo mantém intacta a estrutura do quantificador vectorial de busca exaustiva. No entanto, a possibilidade de uma selecção errada da tabela parcial causa uma pequena degradação do desempenho.

A rotina que implementa no C30 este algoritmo, utilizando o método das bissecções sucessivas para encontrar a média mais próxima, requer um esforço de cálculo de $56+10*\ (\log_2 N - 1) + K + (2*K + 10) * w$ ciclos. Apresenta-se no quadro seguinte o número de ciclos e tempo de execução correspondentes, para o caso da busca numa tabela de 1024 vectores sendo a janela w de 16, 32, 64, 128 ou 256 vectores, e a dimensão do vector 4, 16 ou 32.

Quadro 4.11 - Tempos de execução da rotina de busca exaustiva numa janela.

K	Tempo de Busca (μs) para N = 1024				
	w = 16	w = 32	w = 64	w = 128	w = 256
4	26.28	43.56	78.12	147.24	285.48
16	50.04	90.36	171.00	332.28	654.84
32	81.72	152.76	294.84	579.00	1147.32

Assim, os blocos tipos 4, 7 e 10, quadrados, previamente reduzidos a blocos Classe 1, com as dimensões do tipo 1, dividindo-os em sub-blocos de 2x2, 4x4 e 8x8, respectivamente, e substituindo cada sub-bloco pela sua média, são recuperados substituindo todos os pixels de cada sub-bloco pelo valor codificado da sua média.

Os blocos Classe 2, com as dimensões do tipo 2, são convertidos para blocos tipos 3, 5, 6, 8 e 9. Para os blocos tipos 5 e 8 é usado apenas o mesmo procedimento, substituindo todos os pixels de um sub-bloco, de dimensões 2x2 e 4x4, respectivamente, pela sua média quantificada. Os blocos tipos 3, 6 e 9, necessitam ainda de ser rodados, o que significa processá-los por colunas em vez de linhas..

Após esta conversão, os pixels da imagem original são recuperados adicionando estes valores aos correspondentes dos pixels da imagem anteriormente reconstruída.

Apresenta-se no quadro seguinte o número de ciclos, e tempo de execução correspondente, para estas rotinas implementadas no C30.

Quadro 4.13 - Tempos de execução das rotinas de reconstituição dos blocos originais.

Classe	Tipo	Nº Ciclos	Tempo Execução (μs)
1	1	133	7.98
2	2	261	15.66
2	3	262	15.72
1	4	410	24.60
2	5	806	48.36
2	6	731	43.86
1	7	1462	87.72
2	8	2237	134.22
2	9	2237	134.22
1	10	5865	351.90

4.3.4.2 Blocos Tipo 0 (Blocos Activos)

Para reconstruir um bloco do Tipo 0 é necessário ter em consideração o modo de codificação utilizado.

1) Codificação intra-imagem por média prevista (4x4):
Neste modo, ao bloco codificado é adicionado o valor da média estimada. Este valor está guardado num registo não sendo necessário fazer novo cálculo. Os valores resultantes são guardados nos endereços correspondentes da imagem reconstruída.

Assim, os blocos tipos 4, 7 e 10, quadrados, previamente reduzidos a blocos Classe 1, com as dimensões do tipo 1, dividindo-os em sub-blocos de 2x2, 4x4 e 8x8, respectivamente, e substituindo cada sub-bloco pela sua média, são recuperados substituindo todos os pixels de cada sub-bloco pelo valor codificado da sua média.

Os blocos Classe 2, com as dimensões do tipo 2, são convertidos para blocos tipos 3, 5, 6, 8 e 9. Para os blocos tipos 5 e 8 é usado apenas o mesmo procedimento, substituindo todos os pixels de um sub-bloco, de dimensões 2x2 e 4x4, respectivamente, pela sua média quantificada. Os blocos tipos 3, 6 e 9, necessitam ainda de ser rodados, o que significa processá-los por colunas em vez de linhas..

Após esta conversão, os pixels da imagem original são recuperados adicionando estes valores aos correspondentes dos pixels da imagem anteriormente reconstruída.

Apresenta-se no quadro seguinte o número de ciclos, e tempo de execução correspondente, para estas rotinas implementadas no C30.

Quadro 4.13 - Tempos de execução das rotinas de reconstituição dos blocos originais.

Classe	Tipo	Nº Ciclos	Tempo Execução (μs)
1	1	133	7.98
2	2	261	15.66
2	3	262	15.72
1	4	410	24.60
2	5	806	48.36
2	6	731	43.86
1	7	1462	87.72
2	8	2237	134.22
2	9	2237	134.22
1	10	5865	351.90

4.3.4.2 Blocos Tipo 0 (Blocos Activos)

Para reconstruir um bloco do Tipo 0 é necessário ter em consideração o modo de codificação utilizado.

1) Codificação intra-imagem por média prevista (4x4):
Neste modo, ao bloco codificado é adicionado o valor da média estimada. Este valor está guardado num registo não sendo necessário fazer novo cálculo. Os valores resultantes são guardados nos endereços correspondentes da imagem reconstruída.

2) Codificação inter-imagem (4x4) com compensação de movimento:
O valor dos elementos do bloco codificado é adicionado aos valores dos pixels do bloco da imagem anterior deslocado de acordo com a informação dos vectores de movimento.
Os valores resultantes são guardados nos enderços da imagem reconstruída, correspondentes ao bloco actual.

3) Codificação intra-imagem por média prevista (2x2):
Este modo é idêntico ao modo 1. Para cada um dos quatro blocos codificados é adicionado o valor da média estimada anteriormente, e os valores resultantes são guardados nos endereços correspondentes da imagem reconstruída.

Quadro 4.14 - Tempos de execução das rotinas de reconstituição dos blocos Classe 0.

Classe	Modo Codificação	Nº Ciclos	Tempo Execução (μs)
0	1	132	7.92
0	2	180	10.80
0	3	143	8.58

4.3.5 Critério de Decisão

Presentemente, não está ainda implementada a codificação de comprimento variável a aplicar aos vectores de movimento, nem a reclassificação dos blocos Tipo 0 compensados em movimento. Consequentemente, é conhecido o número exacto de bits gerados por cada bloco, bem como o número de ciclos necessário ao processamento. Nesta fase, o critério de decisão usado baseia-se em dados determinísticos, fazendo a distribuição da taxa disponível e do tempo de processamento pelos blocos gerados, dando prioridade aos blocos do tipo 0.

Para satisfazer os requisitos de taxa de transmissão, são atribuidos 6 bits a cada bloco das classes 1 e 2, sendo o resto da taxa disponível distribuida pelos blocos da classe 0, através de uma simples divisão. Para satisfazer os requisitos de tempo de processamento, calcula-se o tempo necessário para quantificar os blocos das classes 1 e 2 com 6 bits, e distribui-se o tempo disponível pelos blocos da classe 0.

Sendo:

R = (65536 bit/s)/ 8.3 (imagens/s)= 7896 bits/imagem (uma vez que o controlo é feito imagem a imagem.

T = 120ms/60ns=2000000 ciclos, correspondente ao tempo de processamento total para cada imagem.

X = número de ciclos gastos na aquisição + diferenciação + segmentação da imagem.

Y = número de ciclos gastos na conversão Tipo-Classe + reconstrução dos blocos de fundo, calculado durante a segmentação.

y1 = número de ciclos gastos na conversão Tipo-Classe + reconstrução dos blocos do tipo 0, no modo 1.
y2 = número de ciclos gastos na conversão Tipo-Classe + reconstrução dos blocos do tipo 0, no modo 2.
y3 = número de ciclos gastos na conversão Tipo-Classe + reconstrução dos blocos do tipo 0, no modo 3.

rt = número de bits por bloco para codificar o tipo
rm = número de bits por bloco tipo 0 compensado em movimento, para codificar os vectores de deslocamento

B0 = número de blocos da classe 0.
B1 = número de blocos da classe 1.
B2 = número de blocos da classe 2.

Para selecção do modo, atribui-se 6 bits/bloco aos blocos de fundo, para calcular o número de ciclos (T') que resta para quantificar os blocos do Tipo 0, bem como a taxa (R') que fica disponível para este tipo de blocos:

$$T' = T - X - Y - B1 \times 518 - B2 \times 774$$

$$R' = R - (B1 + B2) \times (rt + 6)$$

Apresenta-se no quadro 4.15 o número de ciclos necessário para codificar os vectores de cada classe, variando o número de bits atribuído entre 5 (N=32) e 11 (N=2048).

Quadro 4.15 - Nº de ciclos de execução das rotinas de busca em árvore com 2 andares, utilizando a técnica do desdobramento da fórmula do erro quadrático médio.

K	Nº de Ciclos de Execução						
	N=32 (8*4)	N=64 (8*8)	N=128 (16*8)	N=256 (16*16)	N=512 (32*16)	N=1024 (32*32)	N=2048 (64*32)
4	258	326	462	598	870	1142	1720
16	402	518	750	982	1446	1910	2838
32	594	774	1134	1494	2214	2934	4374

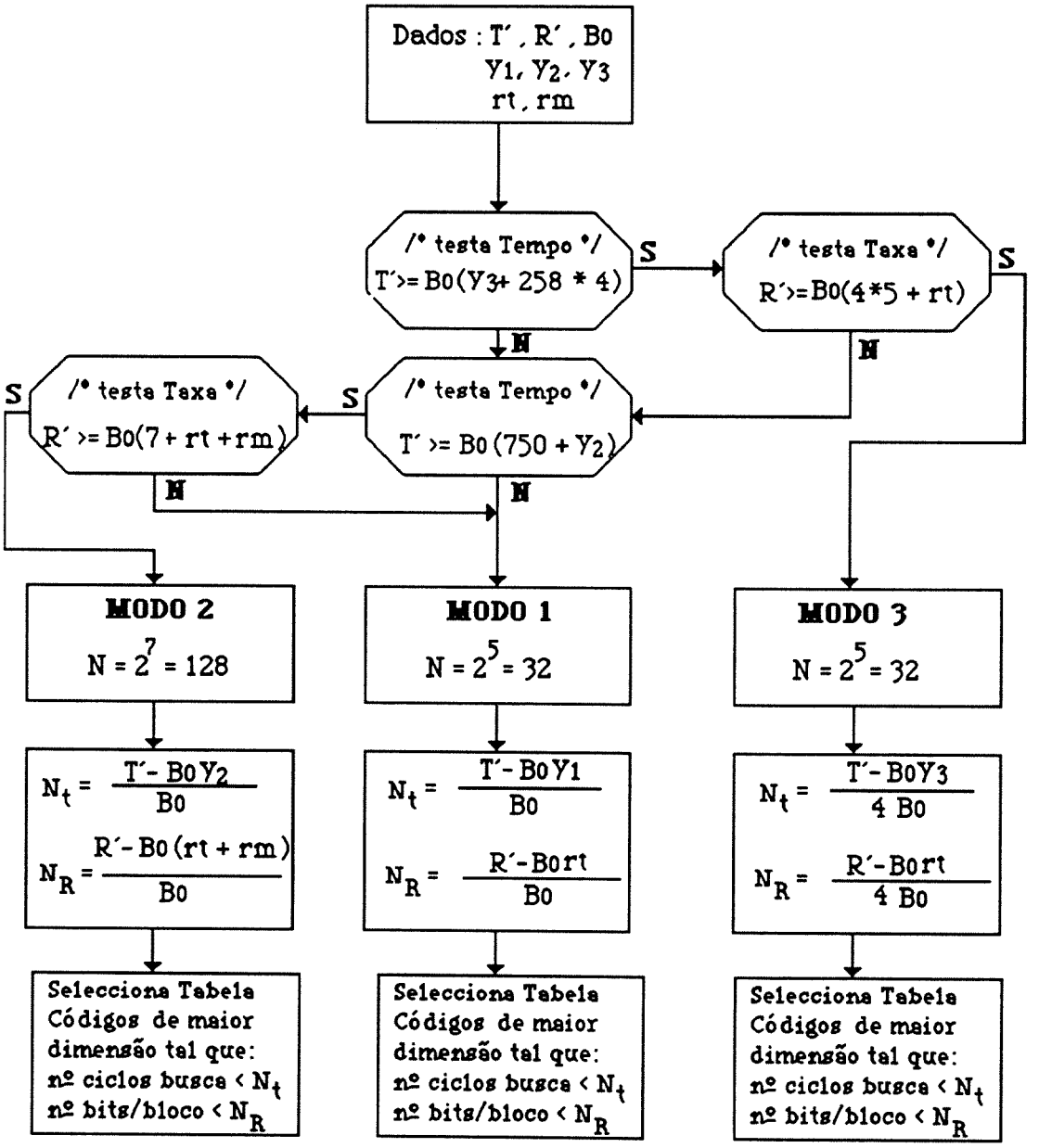


Figura 4.22 - Fluxograma de decisão.

Após a implementação da reclassificação dos blocos compensados em movimento e da codificação de comprimento variável, será necessário efectuar testes sistemáticos para obter estimativas da percentagem de blocos reclassificados e do número de bits gastos na codificação dos vectores de movimento.

4.3.6 Algoritmo Proposto

Sinal de entrada: QCIF (180x144 pixels) com 8.3 imagens/s.

Obtenção da imagem diferença.

Segmentação da imagem diferença.

Selecção do modo de codificação e dos quantificadores associados.

Conversão dos blocos de 11 tipos para 3 classes.

Quantificação dos blocos das 3 classes:

Modo 1- mais baixa qualidade:

Blocos da classe 0: codificação adaptativa por média prevista. O número de bits por bloco pode variar entre 5 e 10.

Blocos das classes 1 e 2: codificação inter-imagem com 6 bit/bloco.

Modo 2 - média qualidade:

Blocos da classe 0: codificação inter-imagem com compensação de movimento. Codificação adaptativa com 6 a 11 bits por bloco. Vectores de deslocamento codificados por entropia.

Blocos das classes 1 e 2: codificação inter-imagem com 6 bit/bloco.

Modo 3 - Alta Qualidade:

Blocos da classe 0: Codificação por média prevista de blocos 2x2. Codificação adaptativa com 5 a 9 bits por bloco.

Blocos das classes 1 e 2: codificação inter-imagem adaptativa com 6 a 9 bit/bloco.

Reconstrução da imagem original.

A escolha dos quantificadores associados a cada modo, feita com base nos resultados das simulações, pretende dar um incremento na SNR, aproximadamente contínuo. No entanto, na prática poderá verificar-se a necessidade de os alterar.

No caso de a imagem ser constituída na sua totalidade por blocos do tipo 0, o modo de mais baixa qualidade com 5 bit/bloco não garante que a taxa de 64 Kbit/s não é ultrapassada (se considerarmos 4 bits para codificação do Tipo e 5 bits para o código, obteríamos uma taxa de 1620*9*8.3 = 118.18 Kbit/s). Na prática, a probabilidade de ocorrência desta situação será pouco significativa, pois numa aplicação do tipo da videovigilância não deverão ocorrer cortes de cena, e mesmo que a câmara sofra uma rotação deverá manter-se uma área de fundo aproximadamente constante. Por este motivo não deverão existir problemas de "overflow do buffer de transmissão". No entanto, poderá prever-se a hipótese de a imagem não ser toda processada.

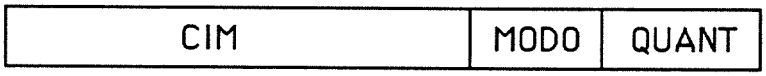
4.3.7 Vídeo Multiplex

A multiplexagem da informação de vídeo tem a seguinte estrutura:

1. Nível de Imagem
2. Nível de bloco

NIVEL DE IMAGEM

Cabeçalho de Imagem:



22 2 3 bits

O cabeçalho de imagem consiste em:

1. código de início de imagem	(CIM)	22 bits
2. informação do modo de codificação	(MODO)	2 bits
3. informação do quantificador seleccionado	(QUANT)	3 bits

Assumindo a utilização de códigos de comprimento fixo para todos os campos de informação, o código de início de imagem é a palavra 0000000000000000000000 (22 zeros consecutivos), uma vez que o número máximo de zeros consecutivos que poderá ocorrer será 21, no caso em que se usa o modo inter com compensação de movimento e o quantificador de 11 bits.

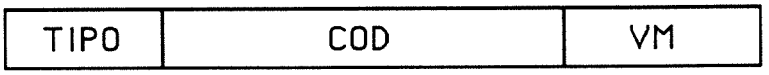
	TIPO	COD	VM	TIPO	COD	
...	1000	00000000000	000000	0100	00000000000	....
	3 +	11	+ 6	+ 1		

Como existem três modos de codificação são necessários 2 bits para a sua codificação.

Como o número máximo de quantificadores associado a um modo é 6, bastarão 3 bits para indicar qual o quantificador usado, pois o descodificador sabe quais os quantificadores associados a cada modo.

NIVEL DE BLOCO

Cabeçalho de Bloco:



4 5 - 11 0/6 bits

O cabeçalho de bloco consiste em:

1. tipo de bloco	(TIPO)	4 bits
2. código	(COD)	5 - 11 bits
3. vector de movimento	(VM)	0 ou 6 bits

Como existem 11 tipos de blocos são necessários 4 bits para identificação. No entanto, como não são necessárias as 16 palavras possíveis, podemos usar a informação do TIPO para reduzir as possíveis sequências de zeros consecutivos. Assim, em vez de a informação referente ao TIPO variar entre 0 e 11, adiciona-se um "offset" de 4 para forçar a ocorrência do bit 1. Logo, a informação do TIPO varia entre o valor 4 (Tipo 0) e o valor 15 (Tipo 11).

Para o código o número de bits é variável entre 5 e 11, pois depende do quantificador seleccionado.

Para os vectores de movimento (horizontal e vertical), variando os seus valores entre -6 e +6, são necessários 3 bits para cada um.

Note-se que, após a implementação da codificação de comprimento variável, para os vectores de movimento e também para o tipo, o código de início de imagem será certamente alterado, podendo talvez ser reduzido.

4.4 "Software" do Descodificador

A implementação do descodificador não deverá apresentar dificuldades, uma vez que o sistema usado para a descompressão tem capacidade idêntica à do sistema de compressão e o algoritmo de descodificação é extremamente simples, consistindo apenas na leitura do vector da tabela de códigos correspondente ao índice recebido.

Dado que o descodificador dispõe de grande capacidade de processamento, será recomendável a utilização de técnicas de pós-processamento, a fim de diminuir o efeito de bloco no modo de baixa qualidade. Foi já simulada uma técnica apresentada por Ramamurthi e Gersho [RAM86a], sendo os resultados obtidos bastante satisfatórios. No entanto, este tópico terá que ser ainda aprofundado.

4.5 Propagação de Erros de Transmissão

Os algoritmos baseados em codificação diferencial, ou que utilizam códigos de comprimento variável, são extremamente sensíveis à propagação dos erros de transmissão. Torna-se por isso necessária a transmissão de pixels no formato PCM que permitam fazer o reset do algoritmo de codificação nestas situações.

O algoritmo adaptativo proposto permite a transmissão desta informação sob a forma de "bit stuffing", ou seja, os pixels em PCM são transmitidos conforme a disponibilidade de taxa. Assim, em cenas de movimento reduzido, será transmitida muita informação em PCM, aumentando a qualidade das imagens descodificadas, com as vantagens subjectivas apontadas aquando da apresentação do algoritmo adaptativo.

O único inconveniente é a possibilidade de em cenas muito movimentadas, não existir taxa disponível para "bit stuffing" e o codificador ficar demasiado tempo sem transmitir informação PCM, criando-se grande susceptibilidade aos erros de transmissão. Este problema poderá ser resolvido atribuindo uma pequena fracção dos 64 Kbit/s para transmissão PCM, de modo a que o refrescamento de toda a área da imagem seja efectuado ao fim de um período de tempo pré-estabelecido. A própria técnica de efectuar este refrescamento pode ser importante. Por exemplo a transmissão dos pixels que constituem a imagem de uma forma entrelaçada (isto é, subamostrando a imagem) pode permitir a sensação de refrescamento ao fim de um varrimento, embora a cobertura total da área da imagem possa implicar vários varrimentos.

Capítulo 5

CONCLUSÕES

Foi desenvolvido um algoritmo de compressão de sequências de imagem para transmissão a taxas reduzidas e mantendo um baixo nível de complexidade. A sua implementação suportada por um "hardware" simples e de uso genérico mostrou que é possível implementar um codec de vídeo de baixa resolução, com complexidade reduzida, para aplicações sem grande exigência de qualidade.

O algoritmo implementado terá que ser ainda aperfeiçoado, em particular, o critério de selecção do modo de codificação e o tamanho das tabelas de códigos associadas a cada modo. Para isso será necessário efectuar testes sistemáticos, de modo a obter-se as estimativas necessárias.

Outro aspecto que não está ainda suficientemente estudado é a codificação de comprimento variável a aplicar aos vectores de movimento e, eventualmente, ao tipo do bloco. Neste momento, pensa-se que esta codificação, bem como a multiplexagem dos dados de modo a obter o "bit stream" pronto para enviar para a interface de rede, poderão ser efectuadas por um processador auxiliar, o CPU do PC. Também a gestão do "buffer" deverá merecer uma maior atenção. Um tópico que deverá merecer um estudo mais aprofundado é o pós-processamento a usar no descodificador, uma vez que este dispõe de grande capacidade de processamento, que não será totalmente utilizada na descodificação.

Por outro lado, pode perspectivar-se que, com algum aumento de complexidade, se poderá obter uma qualidade superior. Note-se que o codificador está, em muitas situações, mais condicionado pelo tempo de processamento do que pela taxa gerada. Assim, um melhor desempenho será certamento obtido se, mantendo o mesmo nível de complexidade, quer a nível de hardware quer do algoritmo de compressão, o TMS320C30 for substituído por um DSP das gerações mais recentes, por exemplo o TMS320C40, que tem um ciclo de instrução que é cerca de metade do ciclo do C30. Isto poderá, por um lado, permitir a utilização de tabelas de códigos de tamanho superior e, por outro, poderá permitir uma cadência de imagem superior, o que resultará numa melhor qualidade. Outra possibilidade a ter em consideração é o desenvolvimento de hardware um pouco mais complexo, usando mais de um DSP.

A nível do algoritmo de codificação, este poderá ser trabalhado com o objectivo de aperfeiçoar os modos implementados, ou mesmo criar novos modos. Algumas hipóteses a estudar serão a substituição da estimativa da média, usada nos modos 1 e 3, por uma interpolação bilinear, ou a criação de um modo inter-imagem compensado em movimento e aplicado a blocos de dimensão 2x2.

Outra hipótese que poderá ser testada é a substituição da quantificação vectorial por uma transformada, nomeadamente, a Transformada Discreta de Cosseno (DCT). Para isso é necessário um algoritmo eficiente para cálculo da DCT bidimensional. No âmbito de uma disciplina do Curso de Mestrado, foi implementado em Assembly do C30 um algoritmo rápido de cálculo da IDCT bidimensional, desenvolvido por Haque [HAQ85] e optimizado por A. Puga [PUG91], para o caso de blocos 8x8. A rotina que implementa o algoritmo optimizado, para blocos 8x8, tem um tempo de execução de 47 microsegundos [MAR91a].

No caso do codec de baixa resolução (180x144 pixels) e a baixa cadência (8.3 imagens/segundo), temos (180x144)/(8x8) = 405 blocos, o que resulta em 2x405x47 = 38070 microsegundos, isto é 38.07 ms, para execução da transformada directa e da transformada inversa. Sendo o tempo de aquisição e processamento de uma imagem 120 ms, e sendo o cálculo das transformadas a operação mais pesada em termos computacionais, pode-se pensar na sua utilização neste codec. Esta rotina está a ser utilizada noutro projecto actualmente em execução no INESC-Porto, tendo comprovado a sua eficiência.

Referências Bibliográficas

[AHM74] N. Ahmed, T. Natarajan and K. R. Rao: "Discrete Cosine Transform"
IEEE Trans. On Computers, vol. COM-33, nº 12, January 1974, pp. 90-93

[AHM91] H. M. Ahmed and R. B. Kline: "Recent Advances in DSP Systems"
IEEE Communications Magazine, May 1991, pp. 32-45

[AIZ86] K. Aizawa, H. Harashima and H. Miyakawa: "Adaptive Discrete Cosine Transform Coding with Vector Quantization for Color Images"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1986, pp. 985-988

[ALE85] S. T. Alexander and S. A. Rajala: "Image Compression Results Using the LMS Adaptive Algorithm"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. ASSP-33, nº 3, June 1985, pp. 712-714

[AND68] H. C. Andrews: "Entropy Considerations in the Frequency Domain"
Proc. of the IEEE, vol. 56, nº 1, January 1968, pp. 113-114

[AND68a] H. C. Andrews and W. K. Pratt: "Fourier Transform Coding of Images"
Proc. Hawaii Int. Conf. on System Science, January 1968, pp. 677-679

[BAK82] R. L. Baker and R. M. Gray: "Image Compression Using Non-Adaptive Spatial Vector Quantization"
Proc. of the 16th Asilomar Conf. on Circuits, Systems and Computers, October 1982, pp. 55-61

[BAK83] R. L. Baker and R. M. Gray: "Differential Vector Quantization of Achromatic Imagery"
Proc. of Int. Picture Coding Symposium, March 1983

[BLA85] Richard E. Blahut: "Fast Algorithms for Digital Signal Processing"
Addison-Wesley Publishing Company, 1985

[BOX90] J. L. Boxerman and H. J. Lee: "Variable Block-Sized Vector Quantization of Grayscale Images With Unconstrained Tiling"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1990, pp. 2277-2280

[BT253] Brooktree Semiconductors: "BT253 - 8 Bit Image Digitizer"

[BUZ80] A. Buzo, A. H. Gray, R. M. Gray and J. D. Markel: "Speech Coding Based Upon Vector Quantization"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. ASSP-28, Oct. 1980, pp. 562-574

[CAN86] J. V. Candy: "Signal Processing, The Model Based Approach"
MacGraw-Hill, 1986

[CCI86] CCIR, Recommendation 601: "Encoding Parameters of Digital Television for Studios"
1986

[CCI90] CCITT SG XV, Recommendation H.261: "Video Codec for Audiovisual Services at p*64 Kbit/s "
March 1990

[CHE77] W. Chen, C. H. Smith and S. Fralick: "A Fast Computational Algorithm for the Discrete Cosine Transform"
IEEE Trans. On Communications, vol. COM-25, nº , 1977, pp. 1285-1291

[CHE83] T. C. Chen and J. P. de Figueiredo: "An Image Coding Scheme Based on Spatial Domain Considerations"
IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. PAMI-5, May 1983, pp. 332-337

[CHE89] S. H. Chen, J. S. Pan: "Fast Search Algorithm for VQ-Based Recognition of Isolated Words"
IEE Proceedings, vol. 136, nº 6, December 1989, pp. 391-396

[CLA81] R. J. Clarke: "Relation Between the Karhunen-Loève and Cosine Transforms"
IEE Proceedings, vol. 128, nº 6, November 1981, pp. 359-360

[CLA85] R. J. Clarke: "Transform Coding of Images"
Microelectronics and Signal Processing, Academic Press, 1985

[COR90] L. Corte-Real and A. P. Alves: "Vector Quantisation of Image Sequences Using Variable Size and Variable Shape Blocks"
Electronics Letters, vol. 26, nº 18, 1990, pp. 1483-1484

[COR90a] L. Corte-Real, Isabel Martins e A. P. Alves: "Codificação de Sequências de Imagens a Baixa Cadência em Aplicações de Vigilância"
1º Workshop sobre I&D de Produtos para a RDIS, Lisboa, 1990

[COS89] COST 211-bis: "Description of the Reference Model 8 (RM8)"
COST 211BIS/SIM89/37, Paris 23/24, May 1989

[COS91] COST 211-ter Simulation Subgroup: "Status Report on Very Low Bit-Rate Coding of Moving Images", version 2
COST 211ter, October 1991

[C30ALT] Texas Instruments: "TMS320C30 Assembly Language Tools" User's Guide
Digital Signal Processor Products, 1988

[C30EMU] Texas Instruments: "TMS320C30 Emulator" User's Guide
Digital Signal Processor Products, 1989

[C30USG] Texas Instruments: "Third-Generation TMS320" User's Guide
Digital Signal Processor Products, 1988

[DAL88] E. Daly and T. R. Hsing: "Variable Bit Rate Vector Quantization of Video Images for Packet-Switched Networks"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1988, pp. 1160-1163

[EQU87] W. Equitz: "Fast Algorithms for Vector Quantization Picture Coding"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1987, pp. 725-728

[ERI85] Staffan Ericsson: "Fixed and Adaptive Predictors for Hybrid Predictive/Transform Coding"
IEEE Trans. On Communications, vol. COM-33, nº 12, Dec. 1985, pp. 1291-1302

[FEN88] Y. Feng and N. M. Nasrabadi: "A New Vector Quantization Scheme Using Inter-Block Correlation: Address-Vector Quantizer"
Proc. of Globcom, 1988, pp. 755-759

[FIS86] T. R. Fischer: "A Pyramid Vector Quantizer"
IEEE Trans. On Information Theory, vol. IT-32, July 1986, pp. 568-583

[GAL68] R. G. Gallager: "Information Theory and Reliable Communication"
John Wiley and Sons, 1968

[GER79] A. Gersho: "Assimptotically Optimal Block Quantization"
IEEE Trans. On Information Theory, vol. IT-25, July 1979, pp. 373-380

[GER82] A. Gersho: "On the Structure of Vector Quantizers"
IEEE Trans. On Information Theory, vol. IT-28, March 1982, pp. 157-166

[GER82a] A. Gersho and B. Ramamurthi: "Image Coding Using Vector Quantization"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 1, 1982, pp. 428-431

[GER83] A. Gersho and V. Cuperman: "Vector Quantization: A Pattern-Matching Technique for Speech Coding"
IEEE Communications Magazine, vol. 21, Dec. 1983, pp. 15-21

[GOL86] M. Goldberg, P. R. Boucher and S. Shlien: "Image Compression Using Adaptive Vector Quantization"
IEEE Trans. On Communications, vol. COM-34, Feb. 1986, pp. 180-187

[GOL86a] M. Goldberg, H. Sun: "Image Sequence Coding Using Vector Quantization"
IEEE Trans. On Communications, vol. COM-34, July 1986, pp. 703-710

[GOL88] M. Goldberg, H. Sun: "Frame Adaptive Vector Quantization for Image Sequence Coding"
IEEE Trans. On Communications, vol. 36, nº 5, May 1988, pp. 629-635

[GRA82] R. M. Gray and E. Karnin: "Multiple Local Optima in Vector Quantizers"
IEEE Trans. On Information Theory, vol. IT-28, March 1982, pp. 256-261

[GRA82a] R. M. Gray and H. Abut: "Full Search and Tree Searched Vector Quantization of Speech Waveforms"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 1, 1982, pp. 593-596

[GRA82b] R. M. Gray and Y. Linde: "Vector Quantizers and Predictive Quantizers for Gauss Markov Sources"
IEEE Trans. On Communications, vol. COM-30, Feb. 1982, pp. 381-389

[GRA84] R. M. Gray: "Vector Quantization"
IEEE Acoustics, Speech and Signal Processing Mag., vol. 1, April 1984, pp. 4-29

[HAB71] A. Habibi: "Comparison of *n*th-Order DPCM Encoder With Linear Transformations and Block Quantization Techniques"
IEEE Trans. On Comm. Technology, vol. COM-19, nº 6, Dec. 1971, pp. 948-956

[HAB71a] A. Habibi and P. Wintz: "Image Coding by Linear Transformation and Block Quantization"
IEEE Trans. On Comm. Technology, vol. COM-91, nº 1, Feb. 1971, pp. 50-62

[HAB74] A. Habibi: "Hybrid Coding of Pictorial Data"
IEEE Trans. On Communications, vol. COM-32, nº 5, May 1974. pp. 614-624

[HAB77] A. Habibi: "Survey of Adaptive Image Coding Techniques"
IEEE Trans. On Communications, vol. COM-25, nº 11, Nov.1977, pp. 1275-1284

[HAN85] H.-M. Hang and J. W. Woods: "Predictive Vector Quantization of Images"
IEEE Trans. On Communications, vol. COM-33, nº 11, Nov.1985, pp. 1208-1219

[HAQ85] M. A. Haque: "A Two-Dimensional Fast Cosine Transform"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. 33, nº 6, Dec. 1985, pp. 1532-1539

[HAS79] B. G. Haskell: "Frame Replenishment Coding of Television"
Advances in Electronics and Electron Physics, Suppl. 12, 1979, pp. 189-217

[HO88] Y. Ho and A. Gersho: "Variable Multi-Stage Vector Quantization for Image Coding"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1988, pp. 1156-1159

[HUF52] David A. Huffman: "A Method for the Construction of Minimum Redundancy Codes"
Proceedings of IRE, vol. 40, Sept. 1952, pp. 1098-1101

[INE91] INESC-Porto: "Vector Quantisation of Image Sequences with Low Bit Rate and Reduced Complexity"
N. Vasconcelos, L. Corte-Real, Isabel Martins and A. P. Alves
COST 211ter Simulation subgroup, doc. SIM(91)59, Munich, 12-13 Sept. 1991

[JAI81] A. K. Jain: "Image Data Compression: A Review"
Proc. of the IEEE, vol. 69, nº 3, March 1981, pp. 349-389

[JAY84] N. S. Jayant and Peter Noll: "Digital Coding of Waveforms; Principles and Applications to Speech and Video"
Prentice-Hall Signal Processing Series, 1984

[JUA82] B. H. Juang and A. H. Gray, Jr.: " Multiple Stage Vector Quantization for Speech Coding"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 1, 1982, pp. 597-600

[KAO87] H. A. Kaouri, J. V. McCanny and W. Millar: "Fast Search Techniques for Use in Vector Quantisation and Related Pattern Matching Applications"
Electronics Letters, vol. 23, nº 7, March 1987, pp. 311-312

[KIN83] R. A. King and N. M. Nasrabadi: "Image Coding Using Vector Quantization in the Transform Domain"
Pattern Recognition Letters 1 (1983), July 1983, pp. 323-329

[KOH88] J. S. Koh and J. K. Kim: "Fast Sliding Search Algorithm for Vector Quantization in Image Coding"
Electronics Letters, vol. , nº , August 1988, pp.

[LEE84] B. G. Lee: "A New Algorithm to Compute Discrete Cosine Transform"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. ASSP-32, nº 6, Dec. 1984, pp. 1243-1245

[LIM67] J. O. Limb: "Source Receiver Encoding of Television Signals"
Proc. of the IEEE, vol. 55, March 1967, pp. 364-380

[LIN80] Y. Linde, A. Buzo and R. M. Gray: "An Algorithm for Vector Quantizer Design"
IEEE Trans. On Communications, vol. COM-28, Jan. 1980, pp. 84-95

[LIO90] Ming L. Liou: "Visual Telephony as an ISDN Application"
IEEE Communications Magazine, February 1990, pp. 30-38

[LLO82] S. P. Lloyd: "Least Squares Quantization in PCM"
IEEE Trans. On Info. Theory, vol. IT-28, part II, March 1982, pp. 127-135
(Bell Lab. Tech. Note 1957)

[MAL89] H. S. Malvar and D. H. Staelin: "The LOT: Transform Coding Without Blocking Effects"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. 37, nº 4, April 1989, pp. 553-559

[MAR90] Maria Isabel Martins: "DSP's em Processamento de Imagem"
Relatório Interno - INESC, Julho 1990

[MAR91] M. Isabel Martins, J. A. Campos Neves e A. Pimenta Alves: "Estudo de um Sistema de Processamento de Imagem Baseado em DSP's"
RecPad91, pp. 3.5.1- 3.5.5, Fevereiro de 1991

[MAR91a] M. Isabel Martins: "Implementação da Transformada Rápida Inversa de Cosseno (8x8) pelo Algoritmo de Haque com o TMS320C30"
Relatório Interno - INESC, Outubro 1991

[MAR92] M. Isabel Martins, L. Corte-Real, N. Vasconcelos e A. Pimenta Alves: "Low Bit Rate Vector Quantisation of Image Sequences With Reduced Complexity"
a apresentar na conferência RECPAD92, Março de 1992

[MAX60] J. Max: "Quantizing for Minimum Distortion"
IRE Trans. On Information Theory, vol. IT-6, March 1960, pp. 7-12

[MEL91] A. G. Melo e M. S. Mota: "Sistema de Aquisição e Processamento de Imagens a Cores"
Relatório de Projecto, DEE - FCTUC, Outubro 1991

[MUS79] H. G. Musmann: "Predictive Image Coding"
Advances in Electronics and Electron Physics, Suppl. 12, 1979, pp. 73-112

[MUS85] H. G. Musmann, P. Pirsch and H. J. Grallert: "Advances in Picture Coding"
Proc. of the IEEE, vol. 73, nº 4, April 1985, pp. 523-548

[NAS88] N. M. Nasrabadi and R. A. King: "Image Coding Using Vector Quantization: A Review"
IEEE Trans. On Communications, vol. 36, nº 8, August 1988, pp. 957-971

[NAS89] N. M. Nasrabadi, S. E. Lin and Y. Feng: "Interframe Hierarchical Vector Quantization"
Optical Engineering, vol. 28, nº 7, July 1989, pp. 717-725

[NET77] A. N. Netravali and B. Prasada: "Adaptive Quantization of Picture Signals Using Spatial Masking"
Proc. of the IEEE, vol. 65, April 1977, pp. 536-548

[NET77a] A. N. Netravali: "On Quantizers for DPCM Encoding of Picture Signals"
IEEE Trans. On Information Theory, vol. IT-23, nº 3, May 1977, pp. 360-370

[NET77b] A. N. Netravali: "Interpolative Picture Coding Using a Subjective Criterion"
IEEE Trans. On Communications, vol. COM-25, nº 5, May 1977, pp. 503-508

[NET80] A. N. Netravali and J. O. Limb: "Picture Coding: A Review"
Proc. of the IEEE, vol. 68, nº 3, March 1980, pp. 366-406

[OPP78] A. V. Oppenheim: "Applications of Digital Signal Processing"
Prentice-Hall Signal Processing Series, 1978

[PAL89] K. K. Paliwal and V. Ramasubramanian: "Effect of Ordering the Codebook on the Efficiency of the Partial Distance Search Algorithm for Vector Quantization"
IEEE Trans. On Communications, vol. 37, nº 5, May 1989, pp. 538-540

[PAP65] A. Papoulis: "Probability, Random Variables and Stochastic Processes"
McGraw-Hill, 1965

[PEA84] W. Pearlman, P. Jakatdar: "The Effectiveness and Efficiency of Hybrid Transform/DPCM Interframe Image Coding"
IEEE Trans. On Communications, vol. COM-32, nº 7, July 1984, pp. 832-838

[PRA78] W. K. Pratt: "Digital Image Processing"
John Wiley & Sons, 1978

[PRI80] D. H. Pritchard and J. J. Gibson: "Worldwide Color Television Standards - Similarities and Differences"
SMPTE Journal, vol. 89, Feb. 1980, pp. 111-120

[PUG91] André T. Puga: "Compressão de Imagem Dinâmica por Transformada"
Tese de Mestrado, DEEC - FEUP, Fevereiro 1991

[RAM84] B. Ramamurthi and A. Gersho: "Image Vector Quantization With a Perceptually-Based Cell Classifier"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 2, 1984, pp. 32.10.1-32.10.4

[RAM86] B. Ramamurthi and A. Gersho: "Classified Vector Quantization of Images"
IEEE Trans. On Communications, vol. COM-34, Nov. 1986, pp. 1105-1115

[RAM86a] B. Ramamurthi and A. Gersho: "Nonlinear Space-Variant Postprocessing of Block Coded Images"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. ASSP-34, nº 5, October 1986, pp. 1258-1268

[RAM88] V. Ramamoorthy and N. S. Jayant: "High Quality Image Coding With a Model-Testing Vector Quantizer and a Human Visual System Model"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 2, 1988, pp. 1164-1167

[REI83] R. Reininger and J. Gibson: "Distributions of the Two-Dimensional DCT Coefficients for Images"
IEEE Trans. On Communications, vol. COM-31, nº 6, June 1983, pp. 835-839

[ROE77] J. A. Roese, W. K. Pratt and G. S. Robinson: "Interframe Cosine Transform Image Coding"
IEEE Trans. On Communications, vol. COM-25, nº 11, Nov. 1977, pp. 1329-1339

[ROE79] J. A. Roese: "Hybrid Transform/Predictive Image Coding"
Advances in Electronics and Electron Physics, Suppl. 12, 1979, pp. 157-187

[SA90] L. de Sá, V. Silva, F. Perdigão, S. Faria and P. Assumpção: "A Parallel Arquitecture for Real-Time Video Coding"
Microprocessing and Microprogramming, vol. 30, 1990, pp. 439-446

[SAB84] S. Sabri: "Movement Compensated Interframe Prediction for NTSC Color TV Signals"
IEEE Trans. On Communications, vol. COM-32, nº 8, August 1984, pp. 954-968

[SABI84] M. J. Sabin and R. M. Gray: "Product Code Vector Quantizers for Waveform and Voice Coding"
IEEE Trans. On Acoustics, Speech and Signal Processing, vol. ASSP-32, June 1984, pp. 474-488

[SHA77] D. K. Sharma and A. N. Netravali: "Design of Quantizers for DPCM Coding of Picture Signals"
IEEE Trans. On Communications, vol. COM-25, Nov.1977, pp. 1267-1274

[SHE88] H.-H. Shen and R. L. Baker: "A Finite State/Frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 2, 1988, pp. 1188-1191

[SIL90] J. C. G. Silvestre e A. J. S. Mendes: "Frame Grabber com Capacidade de Processamento para PC-AT"
Relatório de Projecto, DEE - FCTUC, Outubro 1990

[SIL91] J. Silvestre, A. Mendes, V. Silva e L. Sá: "Sistema de Processamento de Imagem Baseado em Microprocessador de Sinal"
RecPad 91, pp. 3.2.1- 3.2.5, Fevereiro de 1991

[TES79] A. G. Tescher: "Transform Image Coding"
Advances in Electronics and Electron Physics, Suppl. 12, 1979, pp. 113-155

[THY85] K. S. Thyagarajan, S. Parthasarathy and H. Abut: "A Matrix Quantizer Incorporating the Human Visual Model"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, vol. 1, 1985, pp. 141-144

[TSE87] H. Tseng and T. R. Fischer: "Transform and Hybrid Transform/DPCM Coding of Images Using Pyramid Vector Quantization"
IEEE Trans. On Communications, vol. COM-35, Jan. 1987, pp. 79-86

[VAI87] D. J. Vaisey and A. Gersho: "Variable Block-Size Image Coding"
Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1987, pp. 1051-1054

[VAS91] Nuno M. Vasconcelos: "Transmissão de Vídeo a 64 Kbit/s"
Relatório Interno - INESC, 1991

[VEG89] J. F. Vega-Riveros and K. Jabbour: "Review of Motion Analysis Techniques"
IEE Proceedings, vol. 136, nº 6, Dec. 1989, pp. 397-404

[WAT89] H. Watanabe and Y. Suzuki: "64 Kbit/s Video Coding Algorithm Using Adaptive Gain-Shape Vector Quantization"
Signal Processing: Image Communication, 1, 1989, pp. 87-102

[WIN72] P. A. Wintz: "Transform Picture Coding"
Proc. of the IEEE, vol. 60, nº 7, July 1972, pp. 809-820

[ZSC77] W. Zschunke: "DPCM Picture Coding With Adaptive Prediction"
IEEE Trans. On Communications, vol. COM-25, nº 11, Nov.1977, pp. 1295-1302