MÉTODO DE CALIBRAÇÃO PARA QUANTIZADORES VETORIAIS IMPLEMENTADOS NO PLANO FOCAL DE IMAGEADORES CMOS

Roberto de Moura Estevão Filho

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro
Janeiro de 2016

MÉTODO DE CALIBRAÇÃO PARA QUANTIZADORES VETORIAIS IMPLEMENTADOS NO PLANO FOCAL DE IMAGEADORES CMOS

Roberto de Moura Estevão Filho

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO.

Examinado por:

Prof. José Gabriel Rodríguez Carneiro Gomes, Ph.D.

Prof. Antonio Petraglia, Ph.D.

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

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2016

de Moura Estevão Filho, Roberto

Método de Calibração para Quantizadores Vetoriais Implementados no Plano Focal de Imageadores CMOS/Roberto de Moura Estevão Filho. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2016.

X, 38 p.: il.; 29, 7cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Eletrônica e de Computação, 2016.

Referências Bibliográficas: p. 36 – 38.

1. compressão de imagens. 2. imageadores CMOS. 3. plano focal. 4. quantização vetorial. I. Rodríguez Carneiro Gomes, José Gabriel. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia Eletrônica e de Computação. III. Título.

Agradecimentos

Agradeço à minha família, meu orientador, meus professores e meus colegas. Todos vocês contribuíram, direta ou indiretamente, para este trabalho.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletrônico e de Computação.

Método de Calibração para Quantizadores Vetoriais Implementados no Plano Focal de Imageadores CMOS

Roberto de Moura Estevão Filho

Janeiro/2016

Orientador: José Gabriel Rodríguez Carneiro Gomes

Curso: Engenharia Eletrônica e de Computação

Recentemente projetamos e testamos um imageador CMOS fabricado com tecnologia CMOS 0.35 μm, com compressão de imagens utilizando quantização vetorial para representação de texturas. Na teoria de projeto de quantizadores vetoriais de Lloyd-Max, a partição do espaço de entrada é casada ao dicionário e vice-versa, por meio das conhecidas condições da partição e do centróide. Apesar de ambos partição e dicionário serem projetados a partir de vetores extraídos da mesma base de imagens digital, apenas a partição é implementada de forma não-ideal, devido a descasamentos e erros produzidos pelo processo de fabricação. Visto que este trabalho não tem ênfase no *hardware* ou na arquitetura do sistema, alguns detalhes de *hardware* são descritos, de forma a ilustrar as fontes de erro mais importantes. Neste trabalho, é introduzido um método que permite a calibração do dicionário conforme os índices são gerados experimentalmente pela câmera, de forma a reconstruir texturas com erro quadrático médio consistentemente menor. O método é ilustrado por meio de experimentos feitos em imagens capturadas e aperfeiçoadas.

Palavras-chave: compressão de imagens, imageadores CMOS, plano focal, quantização vetorial.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

CODEBOOK CALIBRATION METHOD FOR VECTOR QUANTIZERS IMPLEMENTED AT THE FOCAL PLANE OF CMOS IMAGERS

Roberto de Moura Estevão Filho

January/2016

Advisor: José Gabriel Rodríguez Carneiro Gomes

Course: Electronic Engineering

We have recently designed and tested a CMOS imager fabricated in a 0.35 μm CMOS technology, with focal-plane image compression using vector quantization for texture representation. In Lloyd-Max vector quantization design theory, the input space partition is matched to the codebook and vice-versa, by means of well-known partition and centroid conditions. Although both the partition and codebook were designed with vectors extracted from the same digital image base, only the partition is non-ideally implemented, owing to mismatches and errors produced by the fabrication process. Whereas this work does not have a focus on hardware or system architecture, some hardware details are described, in order to illustrate the most important error sources. In this work, we introduce a method for updating the codebook as binary indices are experimentally generated by the camera, so that texture reconstruction with smaller mean squared error is consistently obtained.We illustrate the method by means of experiments carried out with captured and improved photographs.

Keywords: image compression, CMOS imagers, focal plane, vector quantization.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho é relacionado ao desenvolvimento de sensores de imagens CMOS com capacidade de processamento de imagens no plano focal, que tem sido um tópico de pesquisa muito ativo nos últimos 25 anos [1]. Tais sensores são importantes por conta da simplificação no caminho dos dados, isto é, processadores digitais e *buffers* são removidos e a taxa de quadros é aumentada [2]. Particularmente para algoritmos de compressão de imagens, processamento realizado não somente a nível do sensor [3–5], onde o processamento é feito em áreas adjacentes à matriz de pixels, mas também a nível do píxel é vantajoso, pois a redução de dados começa nas instâncias de processamento de dados que são mais próximas à sua captura. Como consequência, benefícios da redução de dados como largura de banda reduzida são obtidos imediatamente. De um ponto de vista teórico, a estrutura dos dados adquire propriedades distribuídas similares àquelas encontradas em sistemas de visão biológicos [6–9]. Apesar da riqueza de exemplos de processamento no plano focal CMOS na literatura, não foram encontrados exemplos de procedimentos, exceto por técnicas de supressão de ruído de padrão fixo como aquela relatada em [3].

Em [10], resultados experimentais de compressão de imagem obtidos por um imageador CMOS ilustram compressão de imagens no plano focal baseada em modulação por código de pulso diferencial (DPCM), para média de blocos de pixels 4×4, e quantização vetorial (VQ), para textura dos blocos de pixels. Entre vários outros exemplos de compressão de imagens no plano focal (uma extensa compara-

ção é apresentada na Tabela IV de [10]), somente a abordagem proposta permite processamento de dados a nível dos pixels e sem nenhum processamento de sinais adicional fora da matriz de pixels. Com esta idéia, a área de silício da matriz de pixels é inteiramente independente de outros circuitos que podem ser colocados ao redor da área do sensor, levando a um sensor extremamente simples. A aproximação de baixa complexidade da quantização vetorial de busca completa é uma parte muito importante deste projeto.

Por conveniência, detalhes teóricos sobre o algoritmo de compressão do imageador são apresentados no Capítulo 2. Em projetos comuns do estágio de VQ [11], o dicionário é casado com uma partição do espaço de entrada através de um algoritmo de minimização do erro médio quadrático, que aplica iterativamente as bem-conhecidas condições do centroide e da partição [11, 12], que são reproduzidas no Capítulo 3. Depois que o projeto está completo, o dicionário é fixado e usado para reconstrução de imagens em um computador digital. Apenas a partição é implementada em silício dentro do imageador. A partição difere da projetada idealmente, pois é sujeita a descasamentos e erros de fabricação típicos em tecnologia CMOS. Os erros de fabricação não são observáveis, pois apenas os índices comprimidos do VQ são lidos do imageador. Assim, o dicionário não está casado à partição, o que se traduz em degradação das texturas reconstruídas e, portanto, degradação da qualidade da imagem. No Capítulo 4, um algoritmo de calibração de dicionário baseado em dados experimentais capturados pelo imageador é apresentado. Este algoritmo é a contribuição principal deste trabalho. O algoritmo proposto realiza uma iteração adicional da condição do centroide [11], usando a partição de fato implementada e imagens-alvos ideais como recursos para o cálculo do dicionário atualizado. Para parar o algoritmo, é introduzido, na Seção 4.3, o cálculo da divergência de Kullback-Leibler [13] entre a densidade de dados de projeto do VQ e a densidade dos dados experimentais. O Capítulo 5 apresenta fotografias, para mostrar que o algoritmo proposto leva a um desempenho melhor em relação à reconstrução de texturas. Os resultados experimentais também mostram que o dicionário atualizado leva à redu-

ção do erro médio quadrático obtido na reconstrução dos vetores de coeficientes de textura, com relação ao erro médio quadrático obtido com o dicionário original.

Capítulo 2

Algoritmo de Compressão de Imagens

O algoritmo de compressão de imagens é aplicado a blocos de 4×4 pixels e é composto por três estágios principais: modulação por código de pulso diferencial (DPCM), transformação linear e quantização vetorial (VQ). O estágio de DPCM codifica a luminância média dos blocos de 4×4 pixels. Transformação linear e VQ são usados em conjunto para codificar a informação de textura dos blocos de 4×4 pixels após retirada a informação de luminância média. Este trabalho tem como foco melhorar o desempenho do VQ depois que a implementação em silício foi feita. O diagrama de blocos que descreve o algoritmo de compressão de imagens é mostrado na Fig. 2.1. Neste diagrama de blocos, o caminho de cima corresponde ao estágio do DPCM. Os estágios de transformação linear e VQ são indicados no caminho de baixo. As Seções 2.1, 2.2 e 2.3 apresentam uma visão geral destes três estágios. O decodificador é descrito brevemente na Seção 2.4.

Todas as operações de decodificação são feitas em um processador digital em um ambiente externo ao sensor de imagens. A caixa com linha tracejada 'Imageador' na Fig. 2.1 apresenta a situação experimental em que o codificador é não-ideal ($i'(n) = \alpha'(\mathbf{x}(n))$). O erro quadrático médio (MSE) entre $\tilde{\mathbf{x}}(n) = \beta(i'(n))$ e os vetores originais $\mathbf{x}(n)$ é maior do que o erro quadrático médio entre $\hat{\mathbf{x}}(n)$ e $\mathbf{x}(n)$,

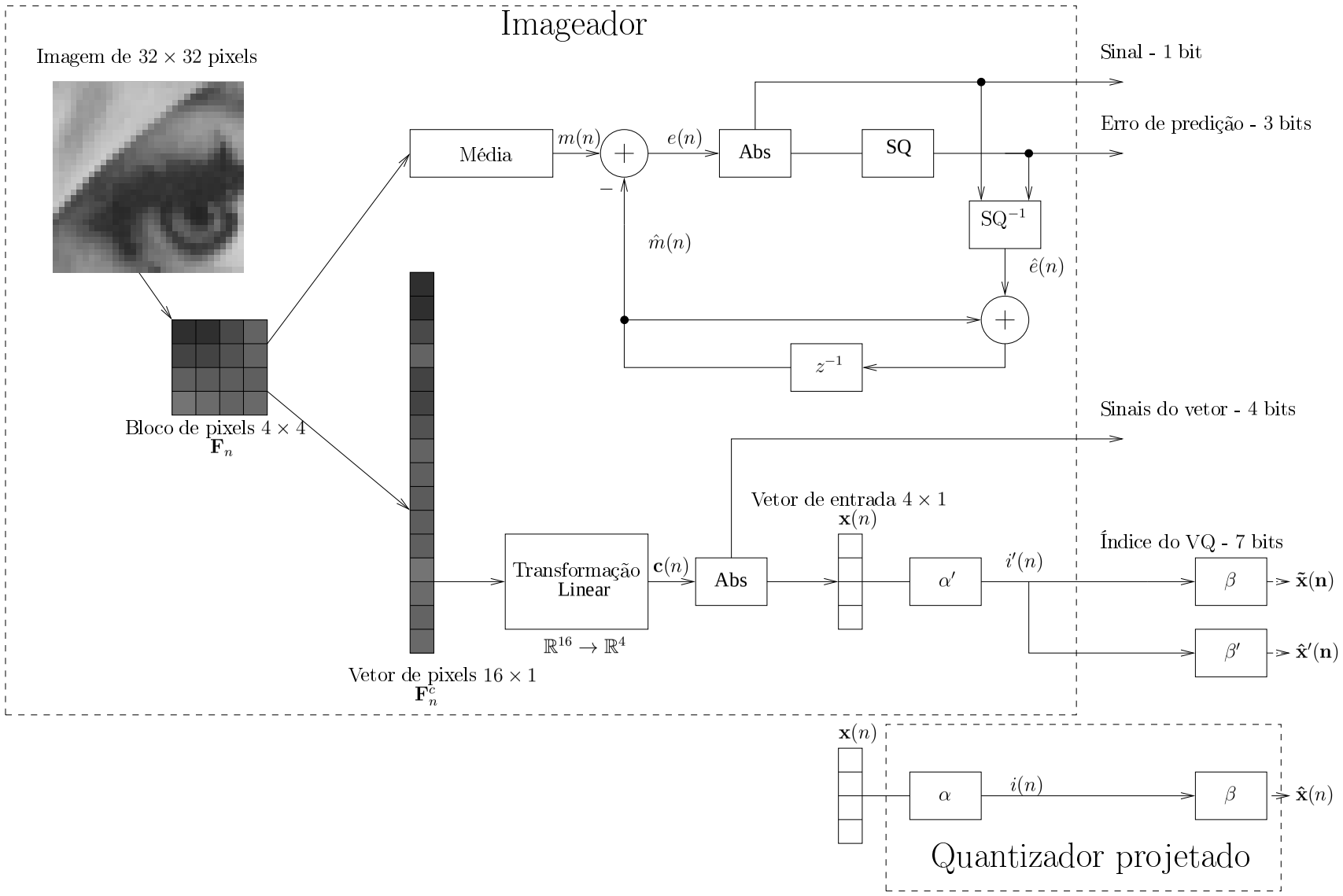

Figura 2.1: Algoritmo de compressão de imagens no plano focal. A maior caixa com linhas tracejadas indica a implementação em silício do imageador. A caixa com linhas tracejadas embaixo à direita indica o par codificador-decodificador do VQ teórico. A linha conectando α' a β' indica o uso de um dicionário calibrado que decodifica os índices do VQ $i'(n)$ obtidos experimentalmente.

pois β foi projetado para α (e não α'). O algoritmo proposto neste trabalho leva a um decodificador melhorado β' que é melhor casado com α' no sentido de que o MSE entre $\hat{\mathbf{x}}'(n)$ e $\mathbf{x}(n)$ é menor do que o MSE entre $\tilde{\mathbf{x}}(n)$ e $\mathbf{x}(n)$.

2.1 DPCM

O n-ésimo bloco de pixels da imagem 32×32 é denotado $\mathbf{F}_n$, como mostrado na Fig. 2.1. A média dos 16 valores dos pixels em $\mathbf{F}_n$ é computada e denotada $m(n)$. Um valor predito, denotado $\hat{m}(n)$, também é disponível no n-ésimo bloco. Para $n = 1$, o valor predito é $\hat{m}(1) = 0$. O erro de predição é calculado como $e(n) = m(n) - \hat{m}(n)$. O valor $e(n)$ é positivo ou negativo com igual probabilidade, portanto o sinal de $e(n)$ é transmitido diretamente para o decodificador, fora do imageador. O módulo de $e(n)$ é quantizado com três bits, que também são transmitidos para o decodificador. O módulo de $e(n)$ é reconstruído localmente, a partir dos mesmos três pixels, e, multiplicado pelo sinal de $e(n)$, forma $\hat{e}(n)$. O valor de predição para o próximo bloco é, então, gerado como $\hat{m}(n+1) = \hat{m}(n) + \hat{e}(n)$. Como tanto o codificador quanto o decodificador concordam a respeito de $\hat{e}(n)$, ambos são capazes de gerar a mesma sequência $m(n)$ exceto por erros no codificador por conta do processo de fabricação. O efeito de erros típicos nos processos de fabricação CMOS em algoritmos DPCM está fora do escopo deste trabalho.

2.2 Transformação Linear

O bloco de 4×4 pixels $\mathbf{F}_n$ é considerado como um vetor coluna 16×1, denotado como $\mathbf{F}_n^c$, como mostrado na Fig. 2.1. Uma transformação linear [14], definida pela

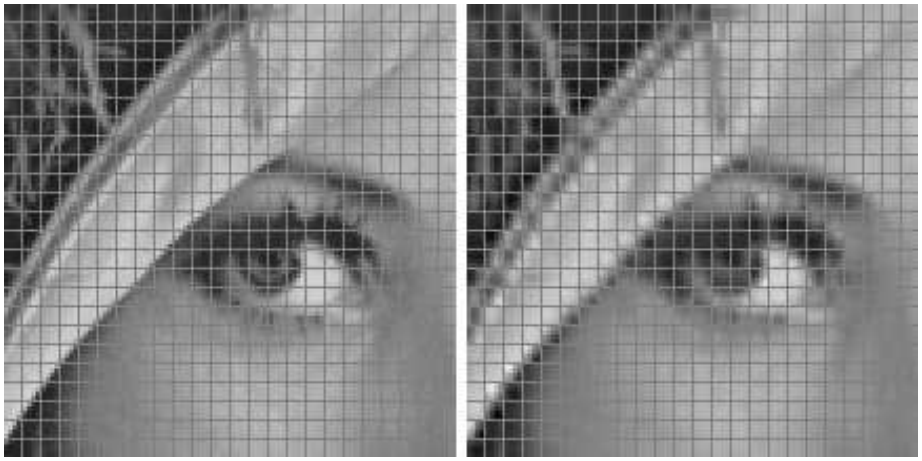

Figura 2.2: Imagem original (esquerda); exemplo de perda de dados (compressão de imagem) devida ao DPCM e supressão de onze componentes AC após a transformação linear dos blocos de 4×4 pixels.

matriz

$$\mathbf{H} = \begin{bmatrix} 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 & 2 & 1 & -1 & -2 \\ 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -2 & -2 & -2 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.1}$$

é aplicada aos pixels em $\mathbf{F}_n^c$. Para vetores de pixels $\mathbf{F}_n^c$ provenientes de fotografias naturais, os vetores 4-D $\mathbf{c}(n)$ obtidos possuem a maior parte da energia dos vetores de pixels originais. Para manter o projeto viável para implementação CMOS no plano focal, os 11 coeficientes restantes não são calculados. A expressão 'coeficientes AC' corresponde a coeficientes de corrente alternada, para indicar todos os coeficientes no domínio da transformada exceto o valor de média do bloco ou coeficiente de corrente contínua (DC), que é codificado pelo estágio de DPCM. Para ilustrar como a aplicação de DPCM para a média dos blocos e a manutenção de apenas quatro coeficientes AC de maior energia afetam a qualidade de uma imagem, a Fig. 2.2 mostra um exemplo de uma imagem original e sua reconstrução com o estágio de VQ ignorado, de forma que $\hat{\mathbf{x}}(n) = \mathbf{x}(n)$.

As normas das linhas de $\mathbf{H}$ na Eq. (2.1) são iguais a $\sqrt{40}$ para as linhas 1 e 2, e $\sqrt{16}$ para as linhas 3 e 4. Além disto, a energia de componentes AC de menor frequência (como as linhas 1 e 2) tende a ser maior do que a energia de

componentes AC de maior frequência (como as linhas 3 e 4). Para obter vetores $\mathbf{c}(n)$ cujas componentes têm aproximadamente a mesma variância, que é conveniente para implementação de VQ, o produto $\mathbf{HF}_n^c$ é pré-multiplicado por uma matriz diagonal $\mathbf{D} = \text{diag}(0.5; 0.5; 1.0; 1.0)$. A textura do n-ésimo bloco é, portanto, descrita por quatro números de acordo com $\mathbf{c}(n) = \mathbf{DHF}_n^c$. As componentes de $\mathbf{c}(n)$ são variáveis aleatórias conjuntamente Laplacianas. As probabilidades de cada uma destas variáveis serem positivas ou negativas são iguais. Portanto, os sinais $s_m(n) = \text{sign}(c_m(n))$, com $m = 1, \ldots, 4$, são enviados diretamente ao decodificador, fora do imageador. Os quatros valores absolutos, que são calculados como $x_m(n) = |c_m(n)|$, são enviados ao estágio de VQ para compressão. São definidos, então, os vetores $\mathbf{s}(n)$ e $\mathbf{x}(n)$.

2.3 Quantização Vetorial

Os vetores de entrada do VQ são denotados por $\mathbf{x}(n)$ ao longo deste texto. O índice n se refere ao n-ésimo bloco em uma imagem 32×32, como usado anteriormente, mas também pode se referir ao n-ésimo bloco de uma base de dados possivelmente grande que contém N blocos. Como visto na Fig. 2.1, $\mathbf{x}(n)$ é mapeado em $i(n) = \alpha(\mathbf{x}(n))$ e $i(n)$ é enviado ao decodificador. A função α corresponde a uma implementação ideal (uma simulação numérica) do VQ. A Fig. 2.1 também apresenta α', que representa a implementação não-ideal de α que é de fato obtida no plano focal do imageador CMOS. A implementação do VQ é ideal no sentido de que corresponde ao codificador originalmente projetado, o mesmo codificador usado no projeto no computador e nas simulações numéricas que levaram às predições teóricas de desempenho. Quando o codificador é fabricado, o que obtemos não é α, mas α'. A Seção 3.1.1 discute e ilustra, através da Fig. 3.2, a degradação causada por erros de fabricação que levam ao codificador α' ao invés de α.

2.4 Decodificador

Após uma implementação ideal do algoritmo na Fig. 2.1, o índice $i(n)$ é usado pelo decodificador, que é externo ao imageador CMOS, para gerar uma reconstrução de mínimo erro médio quadrático (MSE) de $\mathbf{x}(n)$ e, portanto, de $\mathbf{F}_n$. A reconstrução ideal de $\mathbf{x}(n)$ é denotada $\hat{\mathbf{x}}(n)$, de acordo com a caixa de linhas tracejadas 'Quantizador Projetado' na Fig. 2.1. Para implementar $\hat{\mathbf{x}} = \beta(i(n))$, o decodificador procura a $i(n)$-ésima coluna de uma matriz $4 \times K$ que é chamada dicionário. Com $\hat{\mathbf{x}}(n)$ disponível, o sinais dos coeficientes AC podem ser recuperados através de $\mathbf{s}(n)$, o que leva a $\hat{\mathbf{c}}(n)$. O mapeamento $\mathbf{c}(n) = \mathbf{DHF}_n^c$ pode ser invertido aproximadamente (exceto pela perda dos 11 coeficientes AC) de acordo com $\tilde{\mathbf{F}}_n^c = \mathbf{H}^T\mathbf{D}^{-1}\hat{\mathbf{c}}(n)$, que é a imagem, em um vetor coluna, sem a média. A luminância média do bloco é reconstruída de acordo com $\hat{m}(n+1) = \hat{m}(n) + \hat{e}(n)$, e a média reconstruída $\hat{m}(n)$ é adicionada a $\tilde{\mathbf{F}}_n$ para formar o bloco reconstruído $\hat{\mathbf{F}}_n$.

Capítulo 3

Projeto de Quantizadores Vetoriais e Implementação no Plano Focal

Neste capítulo, o algoritmo generalizado de Lloyd (GLA) [11], que é convencionalmente usado para projeto de quantizadores vetoriais, é descrito e sua extensão para o projeto de quantizadores vetoriais com restrição de entropia (ECVQ) [12] é brevemente mencionada. Define-se o dicionário $\mathbf{Y} \in \mathbb{R}^{M \times K}$, onde $M = 4$ é a dimensão dos vetores de entrada e $K \leq 128$ é o tamanho do dicionário [10]. O vetor $\mathbf{x}(n) \in \mathbb{R}^4$ é um vetor de entrada do VQ que representa a informação de textura do n-ésimo bloco 4×4 de uma imagem, ou base de dados de imagens, que contém N blocos. Considere $\alpha : \mathbb{R}^4 \to \{1, 2, \ldots, K\}$, $\alpha(\mathbf{x}(n)) = i(n)$, a função de codificação que mapeia $\mathbf{x}(n)$ no índice de dicionário $i(n)$, e $\mathbf{y}_{i(n)}$ (a i-ésima coluna de $\mathbf{Y}$) o centróide do dicionário que é decodificado. Para minimizar o MSE

$$D = \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{x}(n) - \mathbf{y}_{i(n)}\|^2, \tag{3.1}$$

aplica-se o GLA, que pode ser resumido como a iteração da condição do centróide

$$\mathbf{Y}|\mathbf{y}_k = \frac{1}{N_k} \sum_{\mathbf{x} \in \mathbf{y}_k} \mathbf{x}(n) \tag{3.2}$$

e da condição da partição

$$i(n) = \text{argmin}_k(\|\mathbf{x}(n) - \mathbf{y}_{k(n)}\|^2 + \lambda l_{k(n)}). \tag{3.3}$$

Na Eq. (3.2), a notação $\mathbf{x} \in \mathbf{y}_k$ indica os vetores $\mathbf{x}(n)$ para os quais $\alpha(\mathbf{x}(n)) = k$, e N_k indica a quantidade de tais vetores. Na Eq. (3.3), $l_{k(n)}$ é o comprimento da palavra-código binária que é usada para representar o inteiro $k(n)$, assumindo que um código de comprimento variável foi usado para a transmissão de dados. A taxa média de bits é definida como

$$R = \frac{1}{N} \sum_{n=1}^{N} l_{i(n)}, \tag{3.4}$$

onde $l_{i(n)}$ é o comprimento da palavra-código binária atribuída a $\mathbf{x}(n)$. Como R é limitado por baixo pela entropia

$$H = -\sum p_k \log_2 p_k, \tag{3.5}$$

onde os valores de p_k são probabilidades da distribuição massa de probabilidade associada ao conjunto $\{1, 2, \ldots, K\}$, a função custo (3.1) é modificada para

$$J = D + \lambda H, \tag{3.6}$$

onde o escalar λ é um multiplicador de Lagrange adequadamente escolhido. Minimizar J é equivalente a minimizar D com uma restrição de entropia, que é a essência do algoritmo de projeto de ECVQ [12]. Em projetos anteriores, utilizamos $l_{k(n)} = \log_2 p_k$, mas o uso de comprimentos inteiros, como do código de Huffman, é, na prática, equivalente. Iterando os passos (3.2) e (3.3), o projeto do ECVQ converge para um dicionário localmente ótimo $\mathbf{Y}^*$ e uma partição α^*.

Devido a restrições de complexidade [15], a partição ideal de busca completa do ECVQ não pode ser implementada no plano focal. Ao invés disto, preferiu-se uma versão aproximada de α^*, denotada por α [10], para a qual a complexidade

do circuito inteiro foi estimada em aproximadamente 600 transistores por bloco de pixels. A quantidade aproximada de transistores para cada estágio do diagrama de blocos da Fig. 2.1 é a seguinte: DPCM (120 transistores por bloco), leitura dos pixels (200), transformada linear (70), valor absoluto (50) e VQ (150). A solução de baixa complexidade do VQ usada neste projeto, que não é o VQ de busca completa, é responsável por aproximadamente 25% da complexidade do codificador. Uma solução de busca completa, no entanto, requer cerca de 5×80 (cálculo de distâncias Euclidianas) $+ 2 \times 80$ (*winner takes all*) transistores. Neste caso, a complexidade de todos os estágios seria cerca de 1000 transistores e o VQ seria responsável por cerca de 50% desta complexidade, sem mencionar a grande variedade de diferentes tamanhos de transistores. O decodificador realiza consultas à tabela e, para calibração, a operação do decodificador é repetida muitas vezes. O dicionário $\mathbf{Y}$ foi recalculado para operação ótima com a partição de baixa complexidade. Neste sentido, β é dito casado com α.

As propriedades fundamentais de uma partição com restrição de complexidade α, a respeito de sua implementação, são conforme é descrito a seguir. Considere o vetor de entrada $\mathbf{x}(n) \in \mathbb{R}^4$ e a matriz de pesos $\mathbf{W} \in \mathbb{R}^{4\times 4}$

$$\mathbf{W} = \begin{bmatrix} 0.5 & 0.5 & 0.0 & 0.5 \\ -0.5 & 0.5 & -0.5 & 0.5 \\ 0.0 & -0.5 & 0.5 & 1.0 \\ -0.5 & 0.0 & 1.0 & -0.5 \end{bmatrix}. \tag{3.7}$$

Nós aplicamos quantização escalar nas componentes de $\mathbf{f}(n) = \mathbf{W}\mathbf{x}(n)$, denotadas como $o_i(n) = \mathrm{SQ}\left(f_i(n)\right)$, onde $i = 1, \ldots, 4$. As saídas $o_i(n)$ representam a quantização escalar de $f_i(n)$, para $i = 1, 2, 3, 4$ usando, respectivamente, $3, 2, 1$ e 1 bits [10]. Isto, por sua vez, significa que há $7, 3, 1$ e 1 limiares t_{ij} que implementam a quantização escalar de $o_i(n)$. Como $\mathbf{o}(n)$, uma concatenação dos vetores binários, tem 7 bits, o dicionário pode ter até 128 células.

Decisões em cascata permitiriam implementação de linhas de decisão que não

são necessariamente paralelas e, com esta flexibilidade adicional, pode-se melhorar o desempenho (para redução de taxa ou de MSE). A implementação escolhida foi uma associação de $\mathbf{W}$ com SQ, conveniente com respeito a complexidade, em que cada limiar (comparador) é direta, independente e simultaneamente aplicado ao produto interno para o qual este aplica a quantização. Esta solução força os planos da partição a serem paralelos. Soluções alternativas baseadas em operações não-lineares aplicadas antes da SQ podem ser consideradas. Outros métodos de VQ de baixa complexidade que são adequados para implementação em *hardware* podem ser encontrados em [16], onde é introduzida uma abordagem do tipo *compressive-sensing* aplicada a VQ para a qual se alega 75% de economia de memória, para 30 dB de PSNR.

3.1 Detalhes de Implementação do *Hardware*

Os detalhes de implementação foram discutidos de forma aprofundada em [10], incluindo a aplicação de técnicas de projeto para os circuitos do pixel, transformada linear, valor absoluto, comparador e DPCM. A Fig. 3.1 ilustra a quantização, com dois bits, do resultado de um produto interno entre o vetor de textura $\mathbf{x}(n)$ e a segunda linha de $\mathbf{W}$. O circuito de valor absoluto na parte de cima da figura gera duas versões em modo de corrente de $x_1(n) = \text{abs}\,(c_1(n))$: a corrente que será usada com multiplicadores positivos é obtida a partir da tensão de *gate* $V_{x_1N}(n)$, e a corrente usada com multiplicadores negativos é obtida a partir da tensão de *gate* $V_{x_2P}(n)$. Para gerar os sinais de entrada $x_2(n)$, $x_3(n)$ e $x_4(n)$, são usados outros três circuitos de cálculo de valor absoluto, que não são mostrados na figura.

Na Fig. 3.1, a saída $o_{2\tau}(n)$ tem nível alto se $f_2(n) = -0.5x_1(n) + 0.5x_2(n) - 0.5x_3(n) + 0.5x_4(n)$ for maior do que o limiar $t_{2\tau}$, com $\tau = 1, 2, 3$, e tem nível baixo caso contrário. O produto interno é, então, quantizado com dois bits. O primeiro bit é igual a $o_{22}(n)$ e o segundo bit é obtido com uma operação de OR exclusivo (XOR) entre $o_{21}(n)$ e $o_{23}(n)$. Os erros de fabricação afetam M_2, M_3 e M_4 e, para modelar a imprecisão da implementação do circuito, os fatores de multiplicação são modelados

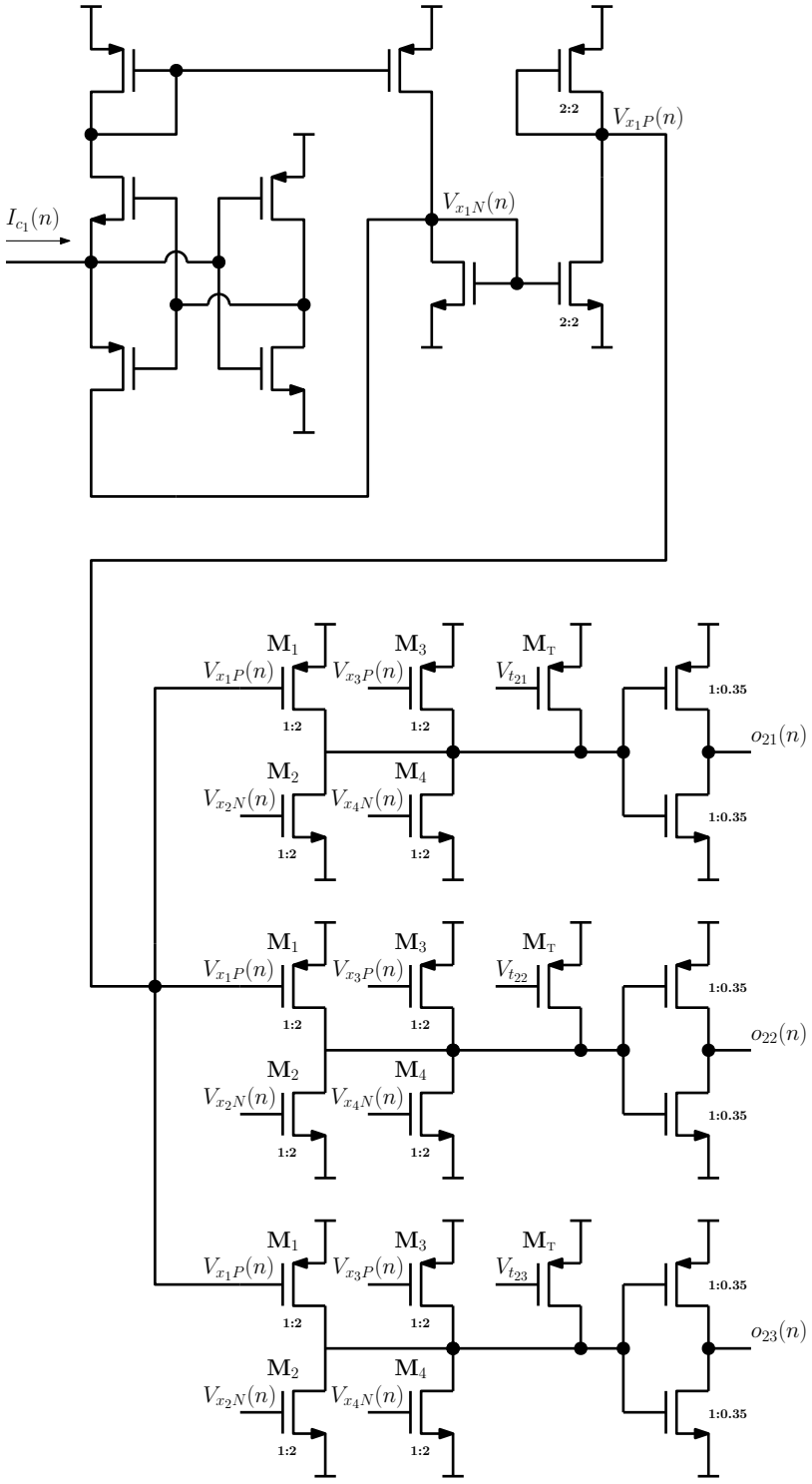

Figura 3.1: Circuito que implementa a quantização de $f_2(n) = -0.5x_1(n) + 0.5x_2(n) - 0.5x_3(n) + 0.5x_4(n)$. Apenas o circuito que calcula o módulo $x_1(n) = |c_1(n)|$ é mostrado. Os circuitos que calculam $x_2(n)$, $x_3(n)$ e $x_4(n)$ não estão na figura.

por $w'_{ij} = w_{ij} + \epsilon_{ij}$, com $j = 1, \ldots, 4$. Os limiares $t_{2\tau}$ também são afetados pelos erros de fabricação e, para levar isto em consideração, os limiares são modelados como $t'_{2\tau} = t_{2\tau} + \nu_{2\tau}$, com $\tau = 1, 2, 3$ no caso da quantização de dois bits da Fig. 3.1. Este modelo, portanto, leva em conta o fato de que a partição implementada é definida pela função $\alpha' \neq \alpha$.

Efeitos de descasamentos nos espelhos de corrente produzidos pelo processo de fabricação podem ser reduzidos determinando tamanhos de transistor adequados [17]. O uso de circuitos em modo de corrente permite uma implementação robusta de todas as operações aritméticas necessárias dentro da área do pixel da matriz de imageamento, para que a compressão de imagens seja feita a nível de pixels e que nenhum processamento seja feito fora da matriz. Isto foi realizado com *fill factor* de aproximadamente 7% e aproximadamente 38 transistores por pixel - uma descrição do imageador a nível de arquitetura pode ser encontrada nas Figs. 2 e 8 de [10]. O consumo máximo de potência é 37 mW, que é comparável ao consumo de potência de outros imageadores com compressão no plano focal. Se a tensão de alimentação for reduzida devido a evolução na teconologia, as mesmas soluções de circuito poderão ser utilizadas. As técnicas propostas podem ser prontamente adaptadas para outras tecnologias, como CMOS 0.18 μm.

3.1.1 Não-Idealidades Subjacentes do Circuito

Considerando que, dentro de um pixel, a corrente do fotodiodo é $I_{\rm ph}$ e que a corrente de saída é $I_{\rm out}$, a relação entrada-saída ($I_{\rm out} = f(I_{\rm ph})$) do pixel em modo de corrente que é usado neste trabalho é não-linear. Melhorias na linearidade de pixels foram estudadas por vários autores. Para um exemplo, pode-ser ver [18]. Idealmente, teríamos $f(I_{\rm ph}) = I_{\rm ph}$, mas um modelo mais realístico é baseado em $f(I_{\rm ph}) = a(I_{\rm ph})^2 + b(I_{\rm ph}) + c$. Apesar da correção de luminância poder ser aplicada em um ambiente digital após a decodificação da imagem, no plano focal a distorção quadrática faz com que os vetores de textura $\mathbf{x}(n)$ sejam deslocados no espaço de entrada do VQ. A saída do pixel é afetada por ruído aleatório (dos tipos kTC,

flicker, *reset* e *shot*) de forma que $I_{\text{out}}(u,v) = f(I_{\text{ph}}(u,v)) + w(u,v)$ para o pixel na posição (u,v) na matriz de imageamento ($u,v = 1,\ldots,32$) para o caso 32×32). Assumindo que a média de $w(u,v)$ é razoavelmente próxima de zero, a partição α só é afetada por erros constantes no tempo, e não é afetada por ruído temporal. O cálculo de produtos internos é afetado pelos efeitos de impedância de saída finita dos espelhos de corrente e também pelos efeitos de gradiente espacial que causam diferenças entre espelhos de corrente que têm a mesma função em blocos de pixel distintos. A distorção causada pela impedância de saída finita é similar à distorção quadrática e também causa deslocamento constante sobre os vetores de entrada do VQ. De forma similar ao que ocorre com a diferença de saída entre diferentes pixels, a alocação errada de vetores no VQ causada por efeitos de gradiente não é constante. Os circuitos de cálculo de valor absoluto são bastante precisos e sua relação entrada-saída é bem representada por $x_m(n) = |c_m(n)|$ (Fig. 2.1). Seguindo o caminho dos dados ao longo do codificador, a última fonte de erro constante é composta por erros nos limiares de quantização escalar: para VQ com sete bits aplicado sobre vetores de entrada de quatro dimensões $\mathbf{f}(n) = \mathbf{W}\mathbf{x}(n)$, os três bits com os quais $f_1(n)$ é codificado são calculados com sete limiares $t_{1j}, j = 1,\ldots,7$, que são, por sua vez, deslocados por sete erros constantes ν_{1j}. Erros similares ocorrem para $f_2(n)$, $f_3(n)$ e $f_4(n)$, que são codificados com dois, um e um bit, respectivamente. Os deslocamentos dos limiares fazem com que os limites da partição do VQ fiquem deslocados no espaço de entrada do VQ.

A Fig. 3.2 ilustra, em duas dimensões somente, a maneira como as bordas da partição do espaço de entrada são modificadas por conta de erros de fabricação. O decodificador β não é mais casado ao codificador, que é α' após a fabricação. Para a mesma sequência de entradas $\mathbf{x}_e(n)$, usada em uma simulação no computador ou capturada pelo sensor através de fotografia, a sequência de índices experimentais $i'(n)$ é diferente dos $i(n)$ que seriam obtidos através de simulações no computador. A diferença entre os índices $i'(n)$ e $i(n)$ pode, assim, ser usada como uma medida da degradação devida à presença de erros de fabricação e, se $i'(n)$ for usado no

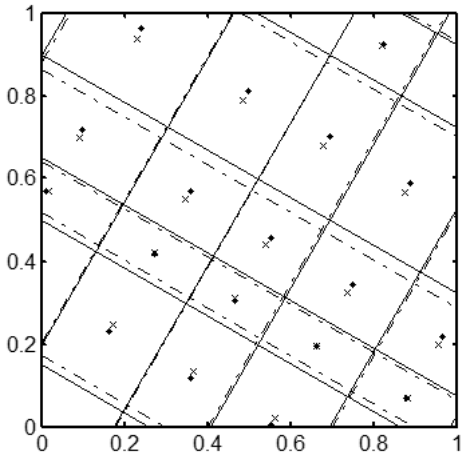

Figura 3.2: Ilustração conceitual mostrando a diferença entre a partição ideal (em linhas contínuas) e a partição experimental (em linhas tracejadas). A partição experimental não pode ser medida. A figura também ilustra o fato de que o dicionário original (pontos) não está mais casado com o codificador implementado. Um novo dicionário (indicado pelos "x") deve ser calculado.

decodificador ideal, um maior MSE ou taxa de bits (ou ambos) seria observado.

Diferentemente do ruído de padrão-fixo [19], que é um ruído constante no tempo, que forma padrões constantes na imagem, o ruído temporal presente em capturas de imagens experimentais pode ajudar no algoritmo de calibração. Isto acontece pois, após a adição de ruído temporal, os mesmos estímulos de entrada $\mathbf{x}_e(n)$ podem contribuir para duas células diferentes (vizinhas) do dicionário. Como o vetor é uma vez membro de uma célula e depois membro de outra célula, este acaba contribuindo para definir a borda entre estas células. O uso de ruído intencionalmente aplicado para tornar aleatório o erro de quantização, chamado *dithering*, é bem conhecido e minuciosamente explicado em outros artigos. Apesar da aplicação de ruído não ser intencional, é benéfica neste sentido. Apenas para fins de visualização, usa-se médias para suprimir o ruído temporal antes da imagem decodificada ser apresentada.

A combinação de todos os deslocamentos espúrios no espaço de entrada do VQ, que é consequência da distorção quadrática do pixel, impedância de saída finita nos espelhos de corrente que fazem o produto interno e erros de limiar, leva a células de partição que não têm os formatos idealmente projetados que são ótimos no sentido de minimização do erro quadrático médio. As posições dos centroides descritas no decodificador não são mais casadas à partição com respeito à minimização do erro

quadrático médio. Assumindo que a partição é estacionária (fixa para todo n) e usando, no decodificador, coordenadas de centróide recalculadas a partir das células da partição não ideal, a redução de erro quadrático médio é garantida.

3.2 Complexidade do Algoritmo de Calibração de Dicionário

A partição de busca completa do VQ não é adequada para implementação no plano focal, por conta de sua complexidade. A quantidade de cálculos de distâncias cresce linearmente com o tamanho do dicionário, que, por sua vez, cresce exponencialmente com a taxa de bits por vetor. Por exemplo, um VQ de busca completa com quatro dimensões e quantização de sete bits requer 128 cálculos de distância no $\mathbb{R}^4$ e, então, procurar a menor destas 128 distâncias. A partição implementada através de $\mathbf{W}$ e quantizadores escalares é, em contraste, restrita em complexidade e relativamente simples em termos de quantidade de operações aritméticas e quantidade de dispositivos CMOS. Esta apenas requer quatro produtos internos para o cálculo de $\mathbf{f}(n) = \mathbf{W}\mathbf{x}(n)$ e, então, 12 limiares para conversão A/D tipo *flash* das quatro "características" (*features*) de $\mathbf{f}(n)$ provenientes de $\mathbf{W}$ com 3, 2, 1 e 1 bit, respectivamente, como mencionado previamente.

O esquema de calibração proposto, do ponto de vista de complexidade, requer diversas operações aritméticas e posições de memória, o que torna o esquema inadequado para implementação no plano focal. Para o armazenamento da informação *raw*[1] dos "estímulos", ou seja, os vetores $\mathbf{x}(n)$ antes da compressão de dados, é necessária uma memória de tamanho 4×128 e conversores A/D devem estar disponíveis no plano focal para a estimação de valores *raw* dos pixels, o que não corresponde à situação prática. Ao invés disto, uma aproximação para a informação *raw* existe apenas no computador digital, onde o procedimento de calibração é executado usando

[1]Usaremos deste ponto em diante a palavra "*raw*" para denotar valores originais internos ao *hardware*. Por exemplo, no caso dos vetores de textura 4-D a palavra "*raw*" é usada no sentido de que estes vetores corresponderiam, aproximadamente, aos que poderiam ser calculados via transformada linear caso os valores dos pixels estivessem digitalizados

sequências de bits experimentalmente comprimidas geradas pelo imageador, para uma base de dados grande. Para recalcular as coordenadas dos centroides atualizados, o número de operações necessárias também é em torno de 4×128. É crucial notar que o valores dos centroides são necessários apenas para reconstrução e não para codificação. Portanto, apesar do alto custo computacional do algoritmo de calibração, este custo não é relevante já que o algoritmo não é executado no plano focal.

O decodificador leva cerca de 12 ms para reconstruir uma imagem. O tempo realmente medido foi de 5.2 segundos para 500 imagens. O Algoritmo 1 (ver tabela na página 21) leva cerca de 20 minutos para processar todas as 412 imagens mencionadas na Seção 5.2 e na Fig. 5.2, mas uma quantidade diferente de fotografias experimentais está disponível para cada *bitmap*: 'Bike' (100 imagens), 'Bird' (25), 'Goldhill (100)', 'Lena (36)', 'Patterns' (1) [20], 'Peppers' (100), 'Steve' (25) e 'Vader' (25). Em média, 435 amostras foram geradas para cada fotografia, então cada iteração de 1 a 412 adiciona 64×435 $\mathbf{x}_e(n)$[2] vetores de dados e 64×435 índices $i(n)$ para o conjunto de dados experimentais. Como estas imagens são processadas somente para decodificação dos dados de VQ, temos que o estágio de VQ leva 6.7 ms por imagem na média. Por causa da influência de ruído temporal do pixel nos resultados de decodificação da imagem, índices diferentes do VQ podem ser gerados a partir do mesmos vetores de dados originais $\mathbf{x}(n)$. O tempo de calibração é, portanto, muito maior do que o tempo de decodificação, mas o processo de calibração é executado apenas uma vez. O tempo gasto com o procedimento no laboratório envolve a captura manual dos 412 bitmaps. Em trabalhos futuros, podem ser desenvolvidas soluções que aceleram a captura de padrões para calibração, mas isto não é relacionado ao tempo de processamento.

[2] O subscrito e será usado para indicar vetores de entrada para o VQ que estão sendo calculados, no computador, como aproximação aos vetores experimentais que estão sendo calculados dentro do imageador e que, por serem internos ao imageador, não são observáveis.

Capítulo 4

Algoritmo de Calibração de Dicionário

O algoritmo de calibração de dicionário consiste em usar a partição implementada (o imageador) de forma a obter experimentalmente os índices que serão usados para recalcular os centroides. A ideia proposta é descrita como pseudocódigo no Algoritmo 1. Como não é possível obter informação não comprimida dos pixels, a informação de textura que serviu de "estímulo" para a partição experimental α' será obtida através da codificação teórica no bitmap do mesmo bloco de pixels a partir do qual $\mathbf{x}(n)$ foi gerado. Enquanto a informação do codificador $i(n)$ é proveniente do circuito implementado, uma estimativa $\mathbf{x}_e(n)$ do vetor de entrada associado ao índice é calculada por *software* a partir de uma versão em bitmap do mesmo bloco de pixels, obtida através de uma base de dados armazenada no computador. A versão experimental de $\mathbf{x}(n)$ não pode ser observada, mas sua versão teórica pode ser calculada. Para distinguir entre ambas as versões de $\mathbf{x}(n)$, a palavra "estímulo" e o símbolo $\mathbf{x}_e(n)$ serão usados para denotar a versão de $\mathbf{x}(n)$ gerada pelo computador que é associada ao índice experimental $i(n)$. O algoritmo é essencialmente uma aplicação da condição do centroide (Eq. (3.2)) para minimização do MSE, considerando a partição α' fixa. Como tanto a partição como o dicionário estão otimizados conjuntamente, o dicionário recalculado deve representar texturas de forma mais

precisa do que o dicionário original, que foi projetado com a partição teórica.

O Algoritmo 1 otimiza o MSE da reconstrução das texturas de acordo com a distribuição dos vetores $\mathbf{x}_e(n)$ disponíveis na base de dados. Para estimar quão bem a distribuição $\mathbf{q_x}$ usada para calibração do VQ se aproxima da distribuição do projeto de VQ original $\mathbf{p_x}$, computa-se a divergência de Kullback-Leibler [13] entre $\mathbf{p_x}$ e $\mathbf{q_x}$. Resultados experimentais, no Capítulo 5, mostram que $\mathbf{q_x}$ converge rapidamente para $\mathbf{p_x}$, o que é favorável pois contruir uma base de dados tão grande quanto a base de projeto do VQ original é bastante difícil, já que a resolução do imageador é limitada a 32×32 e a obtenção experimental dos índices só é realizada através do imageador.

Algoritmo 1 Pseudocódigo do Algoritmo

$\mathbf{p_x} \leftarrow$ vetor de distribuição de probabilidades do conjunto de projeto do VQ;
$\mathbf{r} \leftarrow$ vetor 1×625 contendo zeros;
Base de dados de imagens disponíveis: bitmap(1) até bitmap(L);
nvetor $\leftarrow$ vetor 1×128 contendo zeros;
para $indice\_fotografia = 1$ a L **faça**
    Ler arquivo binário referente aos 64 blocos de pixels;
    para $n \leftarrow 1$ a 64 **faça**
        Ler n-ésimo vetor binário de 7 bits de VQ do arquivo;
        Converter o vetor de 7 bits para um inteiro $i(n)$;
        Ler bloco(n) do bitmap($indice\_fotografia$);
        Transformar bloco de pixels em vetor 16×1 $\mathbf{F}_{cn}$;
        Calcular estímulo de entrada $\mathbf{x}_e(n)$= abs($\mathbf{DHF}_{cn}$);
        Coluna $i(n)$ de $\mathbf{S} \leftarrow$ coluna $i(n)$ de $\mathbf{S}+ \mathbf{x}_e(n)$;
        nvetor$(i(n)) \leftarrow$ **nvetor**$(i(n)) + 1$;
        $j \leftarrow$ índice de $\mathbf{x}_e(n)$ em partição de 625 células no $\mathbb{R}^4$;
        $\mathbf{r}(j) \leftarrow \mathbf{r}(j) + 1$;
    $\mathbf{q_x} \leftarrow \mathbf{r}/\text{sum}(\mathbf{r})$;
    $\mathrm{D_{KL}}(indice\_fotografia) \leftarrow$ Kullback-Leibler $(\mathbf{p_x}, \mathbf{q_x})$;
para $k = 1$ a 128 **faça**
    se nvetor$(k) \neq 0$, **então**
        Coluna k de $\mathbf{Y} \leftarrow$ coluna k de $\mathbf{S}/$**nvetor**(k);
Retorne $\mathbf{Y}$;

4.1 Armazenamento dos Vetores de Estímulo e Calibração do Dicionário

Esta seção descreve o método usado para estimar os vetores de estímulo $\mathbf{x}_e(n)$ que são associados aos índices experimentais $i(n)$. Armazenar os vetores de estímulos $\mathbf{x}_e(n)$ em colunas de um acumulador $\mathbf{S}$ é o primeiro passo para calibração do dicionário. A matriz $\mathbf{S}$ tem tamanho $M \times K$, e sua k-ésima coluna é um acumulador de vetores da respectiva k-ésima célula do dicionário. Quando o imageador é exposto a uma fotografia, este gera uma sequência de índices $i(n)$. Para gerar os 64 vetores de entrada $\mathbf{x}_e(n)$ correspondentes aos índices experimentais $i(n)$, a versão em software do bitmap (ao qual o imageador é exposto) é dividida em 64 blocos 4×4. O vetor $\mathbf{x}_e(n) = \text{abs}(\mathbf{DHF}_{cn})$ é calculado como descrito no Algoritmo 1 e adicionado ao conteúdo da $i(n)$-ésima coluna de $\mathbf{S}$, que é definida como

$$\mathbf{s}_i = \sum_n \mathbf{x}_e(n) | \alpha'(\mathbf{x}_e(n)) = i. \tag{4.1}$$

Para calcular a população das células do dicionário, a $i(n)$-ésima componente do acumulador de frequência $\mathbf{n}$, um vetor linha 1×128, é incrementado de 1. Após $\mathbf{S}$ e $\mathbf{n}$ armazenarem dados suficientes, o novo dicionário $\mathbf{Y}'$ é calculado pela Eq. (3.2). A i-ésima coluna do dicionário aperfeiçoado é dada por

$$\mathbf{y}'_i = \begin{cases} \mathbf{s}_i/n_i, & n_i > 0 \\ \mathbf{0}, & n_i = 0 \end{cases}. \tag{4.2}$$

É possível que, ao final do processo de acumulação, existam células vazias. Isto pode ser causado pelo tamanho limitado da base de dados, com o qual é possível que nenhum vetor seja atribuído a uma célula de baixa probabilidade, ou pela restrição de entropia. Para evitar a divisão por zero que ocorreria quando o acumulador $\mathbf{s}_i$ é dividido por $n_i = 0$, esta divisão não é feita para células vazias. Neste caso, $\mathbf{y}'_i$ é definido como o vetor nulo. O procedimento completo é resumido no Algoritmo 1.

4.2 Melhoria Teórica do MSE

O dicionário como definido pela Eq. (3.2) minimiza o MSE na Eq. (3.1) para a partição ideal projetada α. Para minimizar o MSE para a partição não ideal implementada α', o dicionário $\mathbf{Y}$ é recalculado com dados experimentais. Para recalcular $\mathbf{Y}$, obtém-se o código binário de uma fotografia e então usa-se os índices gerados pelo VQ para particionar os estímulos. Após a partição dos estímulos, $\mathbf{Y}^h$ (na iteração $h = indice\_fotografia$) é recalculado como na Eq. (4.2). Isto minimiza o MSE para este conjunto específico de dados. Este cálculo é referido como iteração h. Então outra fotografia é adicionada aos acumuladores $\mathbf{S}$ e $\mathbf{n}$ e o dicionário $\mathbf{Y}^{h+1}$ é recalculado com este conjunto de amostras maior. Ao fim deste procedimento iterativo, obtêm-se um dicionário que minimiza o MSE para o conjunto de dados considerado, que é composto por todas as fotografias. Ou seja, o MSE minimizado depende do conjunto de imagens usado na calibração. Se amostras de imagens suficientes forem usadas, o MSE é minimizado para uma distribuição de dados que segue uma função de probabilidade adequada para a maioria das fotografias naturais.

4.3 Critério de Parada (Divergência de Kullback-Leibler)

Esta seção descreve o método usado para verificar a convergência da distribuição da base de dados baseado na divergência de Kullback-Leibler. A divergência de Kullback-Leibler [13] entre $\mathbf{p_x}$ e $\mathbf{q_x}$, definida por

$$D_{\mathrm{KL}}(\mathbf{p_x}||\mathbf{q_x}) = \sum_i \mathbf{p_x} \log_2 \left(\frac{\mathbf{p_x}}{\mathbf{q_x}} \right), \tag{4.3}$$

mede a diferença entre as funções massa de probabilidade $\mathbf{p_x}$ e $\mathbf{q_x}$. Ela representa a penalidade na taxa de bits que se recebe ao projetar um código de entropia para um VQ projetado com uma base de dados de treino com distribuição $\mathbf{q_x}$ quando, na realidade, a verdadeira distribuição dos dados que serão quantizados é $\mathbf{p_x}$.

A D_{KL} será usada como critério de parada para a iteração do algoritmo de calibração: se a distribuição de probabilidade dos dados experimentais for suficientemente próxima da distribuição teórica (com a qual o VQ original foi projetado), então o novo dicionário é assumido como ótimo para uma distribuição de probabilidades similar àquela de fotografias naturais em geral. Para calcular $\mathbf{p_x}$ e $\mathbf{q_x}$, o espaço de entrada, com quatro dimensões, dos vetores $\mathbf{x}_e(n)$ é particionado em uma grade regular. Um histograma conta a quantidade de vetores de entrada em cada célula desta partição regular. Este histograma é normalizado pelo tamanho da base de dados de forma a reproduzir a distribuição de probabilidade dos dados de entrada. Para implementação de entrada uma grade regular com $5^4 = 625$ células foi criada com centróides de quantização escalar 0.05, 0.15, 0.25, 0.35, 0.45 posicionados ao longo de todos os quatro eixos.

Para avaliar a convergência da distribuição experimental dos dados, a $D_{\mathrm{KL}}(\mathbf{p_x}||\mathbf{q_x}(h))$ é calculada depois que a h-ésima fotografia é acumulada, sendo a distribuição $\mathbf{q_x}$ calculada com todos os vetores $\mathbf{x}_e(n)$ acumulados até esta iteração do processo de calibração. O vetor $\mathbf{p_x}$ é a distribuição de probabilidades de todos os vetores $\mathbf{x}(n)$ da base de dados teórica e $\mathbf{q_x}$ é a distribuição de probabilidades dos vetores $\mathbf{x}_e(n)$ da parte da base de dados experimental usada até a h-ésima iteração de melhoria de dicionário que produz $\mathbf{Y}^h$. O inteiro h corresponde ao índice de fotografia do Algoritmo 1. À medida em que usamos uma parte cada vez maior da base de dados, as distribuições se tornam cada vez mais similares, de forma que a convergência pode ser assumida quando mudanças na $D_{\mathrm{KL}}(\mathbf{p_x}||\mathbf{q_x}(h))$ ficarem abaixo de um valor pequeno arbitrário. É interessante notar que as bases de dados teórica e experimental podem conter fotografias completamente diferentes. Se imagens naturais forem escolhidas, $\mathbf{q_x}(h)$ deve convergir para $\mathbf{p_x}$, já que os vetores $\mathbf{c}(n)$ de imagens naturais seguem uma distribuição conjuntamente Laplaciana.

Capítulo 5

Resultados Experimentais

Nesta seção, são apresentados dados experimentais, e os métodos utilizados para coletá-los, de forma a demonstrar como o novo dicionário se adapta à partição implementada conforme mais amostras são utilizadas. A seguir, descrevemos os arquivos binários que contêm dados experimentais (fotografias codificadas). Uma imagem capturada pelo imageador é salva em um arquivo como uma longa sequência binária. Esta sequência é particionada em 64 sequências de 15 bits, que representam a informação de um bloco da imagem. Como uma fotografia 32×32 contém 64 blocos, o comprimento total da sequência binária para uma imagem é 960. Para reduzir ruído temporal, cada imagem é capturada várias vezes. Além disto, para compensar o efeito de iluminação não-uniforme, para algumas fotografias ('Bike', 'Goldhill', 'Lena' , 'Peppers'), foram gerados três arquivos adicionais, que contêm fotografias das imagens rotacionadas de 90 graus entre cada uma das sessões de captura.

O algoritmo proposto é executado em um computador digital em um ambiente externo ao imageador. Para calibração, o imageador captura fotografias de várias imagens e os dados binários são lidos por um controlador integrado programável (PIC18LF4550, da Microchip [21]) e então enviados ao computador por meio da uma porta USB. O computador executa o algoritmo proposto e então atualiza seu próprio dicionário. O imageador não usa o dicionário na codificação da imagem e, portanto, não recebe nem precisa de informação alguma do computador.

Tabela 5.1: Comparação entre erro médio quadrático obtido com dicionários calibrados e dicionário original. O dicionário 'Completo' é calculado com toda a base de dados disponível. A classe de dicionários 'Exceto' é calculada com todos os vetores de entrada disponíveis exceto por aqueles obtidos da imagem a ser decodificada.

Imagem Decodificada	Exceto	Completo	Original
Bike	0.0871	0.0852	0.0989
Bird	0.0503	0.0501	0.0632
Goldhill	0.0288	0.0284	0.0371
Lena	0.0303	0.0295	0.0392
Peppers	0.0235	0.0224	0.0269
Steve	0.0663	0.0659	0.0744
Vader	0.1049	0.1022	0.1073

5.1 Melhoria experimental do MSE

A qualidade das imagens reconstruídas é medida calculando o erro quadrático médio entre os vetores originais $\mathbf{x}_e(n)$, obtidos através do *bitmap* da imagem original, e os vetores do dicionário $\hat{\mathbf{x}}(n)$, decodificados do arquivo binário proveniente do imageador, que são usados para representar os vetores originais. Ao obtermos o MSE de todos os vetores, que correspondem aos blocos da imagem, com diferentes dicionários, podemos determinar o dicionário que resulta no menor MSE. A Tabela 5.1 indica que o dicionário calculado com todas as fotografias disponíveis (coluna 'Completo') obtém MSE menor do que o dicionário originalmente projetado (coluna 'Original'). A coluna 'Exceto' corresponde a um dicionário calculado com todas as fotografias disponíveis exceto a utilizada na medição do MSE, de forma a tentar reproduzir as condições habituais de teste do imageador, já que, de forma geral, os vetores de teste não estarão presentes durante o projeto do dicionário. O dicionário 'Exceto', apesar de obter desempenho pior do que o 'Completo', ainda assim decodifica todas as fotografias com valores de MSE menores dos que os do dicionário 'Original'.

Figura 5.1: Fotografia 'Lena' após (da esquerda) decodificação, filtragem passa-baixas e processamento pixel a pixel com tangente hiperbólica. Este pós-processamento visa compensar erros de fabricação que levaram a níveis de sensibilidade abaixo do esperado.

5.2 Comparação Visual do Desempenho do Dicionário

Nesta seção, é apresentada uma comparação subjetiva do desempenho através da avaliação visual de imagens decodificadas com cada um dos três dicionários. As fotografias mostradas são composições de fotografias de tamanho 32×32, que é a resolução do imageador. Para obter a fotografia final, cada recorte 32×32 é decodificado individualmente e a imagem é então montada. A imagem montada, denotada por $\mathbf{F}$ passa por uma convolução

$$\mathbf{F}'(u,v) = \sum_{m=-1}^{1} \sum_{n=-1}^{1} \mathbf{G}(m,n)\mathbf{F}(u-m, v-n), \tag{5.1}$$

onde

$$\mathbf{G} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.2}$$

e $\mathbf{F}(u,v)$ é o pixel da u-ésima linha e v-ésima coluna. No caso da Fig. 5.1, o filtro passa-baixas $\mathbf{G}$ melhorou o MSE de 0.0198 para 0.0163 (assumindo faixa dinâmica de 0.0 a 1.0), o que corresponde a uma redução de 17%. O filtro passa-baixas também melhora a qualidade subjetiva das imagens reconstruídas, já que ruídos de

padrão fixo de alta frequência e artefatos de blocagem são reduzidos.

O pixel em modo de corrente se comporta de forma linear na faixa de entrada de 50 lux a 350 lux. A implementação do DPCM é sujeita a erros de fabricação, com os quais a faixa dinâmica não pode ser inteiramente aproveitada: problemas de saturação foram observados com níveis de iluminação abaixo dos valores máximos. Como consequência, as imagens obtidas sem pós-processamento são geralmente escuras, como exemplificado nas imagens da esquerda e do centro da Fig. 5.1. Para tornar as imagens mais claras, suas médias são suprimidas e substituídas por 0.5 (também assumindo faixa dinâmica de 0.0 a 1.0). Além disto, o contraste é aumentado por meio de uma tangente hiperbólica com ganho 10. Este valor foi verificado subjetivamente como o que levava aos melhores resultados de contraste. A imagem resultante $\mathbf{F}'$ é, portanto, processada, pixel a pixel, pela função tangente hiperbólica

$$\mathbf{F}''(u,v) = 0.5 + 0.5 \tanh 10 \left[\mathbf{F}'(u,v) - \frac{1}{Q^2} \sum_{u,v} \mathbf{F}(u,v) \right], \tag{5.3}$$

onde Q é o número de linhas (e colunas) da imagem. O processo é ilustrado na Fig. 5.1, que foi decodificada com o dicionário 'Original'. Os fatores 0.5 e 10 foram ajustados subjetivamente para preservar os detalhes originais da imagem mais à direita na Fig. 5.1. Este pós-processamento é necessário devido a erros de implementação do imageador e não por causa do algoritmo de calibração.

A partir da três colunas mais à esquerda na Fig. 5.2 pode-se observar que os novos dicionários apresentam texturas mais proeminentes em geral, o que resulta em bordas mais acentuadas. Em regiões mais suaves da imagem, no entanto, o resultado às vezes parece mais ruidoso. Comentários em relação a algumas características de algumas imagens específicas serão feitos a seguir de forma a ressaltar as diferenças entre os dicionários. Os dicionários 'Completo' e do tipo 'Exceto' podem ser considerados de forma geral como dicionários atualizados, pois ambos são resultados do algoritmo de calibração.

Para a fotografia 'Bike', as imagens obtidas com os dicionários atualizados, mostradas nas primeiras duas colunas, indicam melhor desempenho na representação das

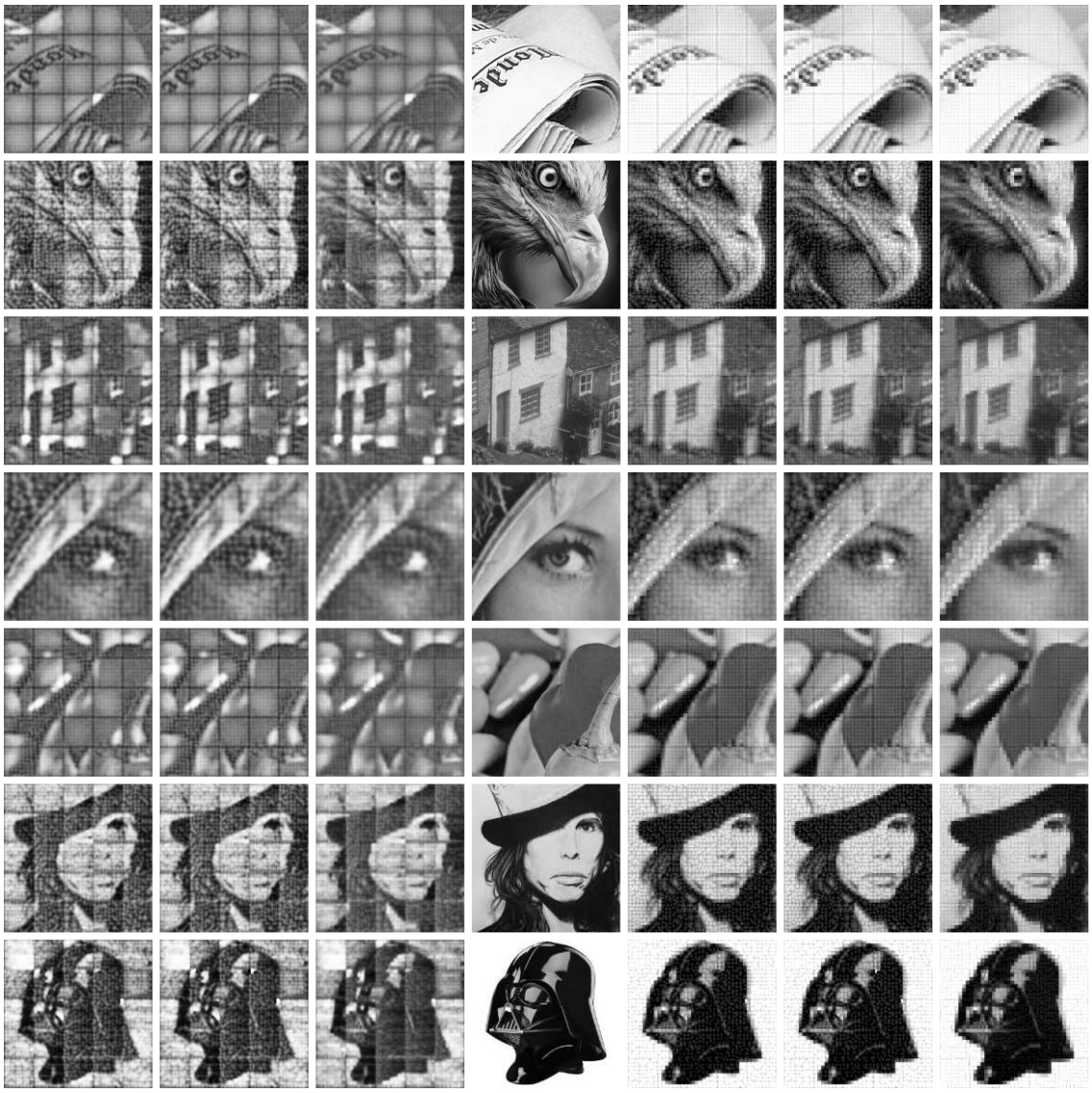

Figura 5.2: Três colunas mais à esquerda: imagem para comparação visual a respeito do desempenho dos diferentes dicionários. Cada imagem (de cima para baixo: 'Bike', 'Bird', 'Goldhill', 'Lena', 'Peppers', 'Steve', 'Vader') é apresentada decodificada pelos dicionários (da esquerda para a direita) 'Completo', 'Exceto' e 'Original'. A coluna do centro contém as imagens originais. Três colunas mais à direita: fotografias obtidas com os índices do VQ produzidos pelo imageador em experimentos reais, mas com informação de média dos blocos computada diretamente das imagens originais. Da esquerda para a direita: dicionários 'Completo', 'Exceto' e 'Original'.

letras do jornal quando comparadas às do dicionário 'Original'. Mesmo no caso do dicionário 'Exceto', que não contém informação de textura proveniente da fotografia 'Bike', as letras são bem representadas. Para a fotografia 'Goldhill', as texturas das janelas são mais claras e pode-se ter uma melhor ideia da estrutura da janela. As estruturas quase não são visíveis quando a imagem é decodificada com o dicionário 'Original', e ficam aparentes quando os dicionários atualizados são usados. Em relação à fotografia 'Lena', os dicionários atualizados representam bem as bordas do chapéu, já que não há o efeito serrilhado presente na imagem 'Original'. As plumas do chapéu também estão mais nítidas com os dicionários atualizados. No entanto, o maior contraste nos cílios dá um aspecto mais ruidoso ao redor dos olhos.

Para evitar os efeitos da implementação não-ideal do DPCM, de forma que a comparação entre as texturas reproduzidas fique facilitada, foram incluídas nas três últimas colunas, da esquerda para a direita na Fig. 5.2, os resultados obtidos com os índices do VQ provenientes do imageador, mas com a informação de média dos blocos diretamente da imagem original. Esta situação não é realista, porque só poderia ser obtida com uma codificação perfeita de média, mas permite uma comparação justa com foco nas texturas reconstruídas pelos três tipos de dicionário. As médias dos blocos reconstruídas nas três primeiras colunas da Fig. 5.2 possuem valores menores do que os valores exatos usados nas três últimas colunas, como indicado na Fig. 5.1. Portanto, para ajustar a energia relativa entre os sinais de DPCM e de VQ, a intensidade das texturas reconstruídas (saída dos VQs) nas três últimas colunas da Fig. 5.2 foi escalada por um fator experimental de 2 com respeito aos resultados das três primeiras colunas. Comparando as imagens das três primeiras com as três últimas colunas da Fig. 5.2, percebe-se que os erros na parte de DPCM afetam severamente a imagem decodificada, o que tornam necessárias as operações de pós-processamento ilustradas na Fig. 5.1. Problemas causados por iluminação irregular, como diferenças entre as médias de regiões com tonalidades similares ou manchas escuras nas bordas das imagens 32×32, também são eliminados nas três últimas colunas da Fig. 5.2. O comentários detalhados sobre os dicionários feitos

no parágrafo anterior também valem para estas três últimas colunas.

Aumentar a complexidade do DPCM reduziria o erro quadrático médio associado à representação dos níveis médios dos blocos, ao custo de redução do *fill factor*. Em estudos anteriores, foi observado que a representação de três bits para abs($e(n)$) é capaz de reconstrução de imagens a 30 dB [10], com taxa de bits por volta de 1.5 bits por bloco estimada pelo uso de codificação de entropia baseada em códigos de Huffman. Os erros associados ao processo de fabricação levaram a um DPCM degradado, cujo desempenho inferior é apresentado na Fig 5.2. Como comentado anteriormente, a calibração do DPCM não faz parte do escopo deste trabalho. A aplicação de codificação de Huffman ao DPCM não-calibrado foi considerada, mas, para manter a quantidade de transistores baixa, o circuito para codificação de entropia não foi implementado.

Para ilustrar os requerimentos de memória do algoritmo (uso de 4×128 posições de memória contendo números de ponto flutuante), assim como a informação do dicionário, que é disponível exclusivamente no decodificador, a Fig. 5.3 apresenta o dicionário teoricamente ótimo ('Original') e um exemplo de dicionário atualizado. Dependendo da implementação do algoritmo, o armazenamento de blocos de textura de 412 imagens (vetores $\mathbf{x}_e(n)$, assim como seus códigos binários) também deve ser realizado.

5.3 Análise do Critério de Parada

Esta seção apresenta os resultados obtidos com o cálculo da divergência de Kullback-Leibler. Como mostrado no Algoritmo 1, os vetores massa de probabilidade $\mathbf{p}_x$ e $\mathbf{q}_\mathbf{x}(h)$ são calculados por meio de um VQ das bases de dados. Como as bases experimentais são pequenas, existem células vazias (zeros) em $\mathbf{q}_\mathbf{x}(h)$, o que faz com que $\dfrac{\mathbf{p}_x}{\mathbf{q}_\mathbf{x}(h)} \to \infty$ e, portanto, tenhamos $D_{KL}(\mathbf{p}_\mathbf{x}||\mathbf{q}_\mathbf{x})$ indefinida. Para evitar este caso, definimos $\bar{\mathbf{q}}_\mathbf{x}(h) = \mathbf{q}_\mathbf{x}(h) + \varepsilon$, onde ε é um vetor de 256 componentes contendo termos muito pequenos, de forma que $\bar{\mathbf{q}}_\mathbf{x}(h)$ ainda pode ser considerado normalizado e evitando-se a divisão por zero. No experimentos descritos, $\varepsilon_i = 2.2204 \times 10^{-16}$

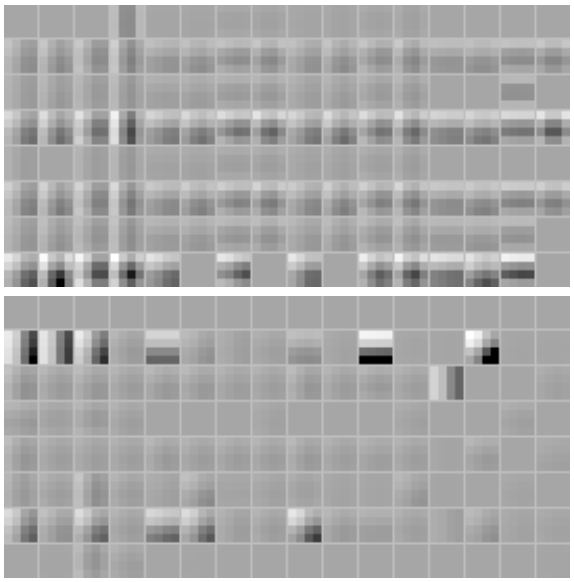

Figura 5.3: Em cima: dicionário teórico ('Original'). Como os centróides $\mathbf{y}_k$ pertencem ao $\mathbb{R}^4$, são mostradas as texturas 4×4 reconstruídas a partir de $\mathbf{y_k}$ considerando $\mathbf{s}(n) = [1,\ 1,\ 1,\ 1]$. Embaixo: dicionário 'Completo', correspondendo à estimativa do dicionário para α' contendo todos os dados experimentais disponíveis.

(um número pequeno padrão gerado no MATLAB). Para verificarmos a convergência da distribuição experimental $\mathbf{q_x}(h)$ para $\mathbf{p}_x$, obtivemos vários valores de $\mathbf{q_x}(h)$ com cada incremento de h correspondendo a uma imagem 32×32, isto é, cada vez que h aumenta uma unidade 64 vetores $\mathbf{x}_e(n)$ são adicionados ao cálculo de $\mathbf{q_x}(h)$. Imagens rotacionadas usadas no projeto do dicionário também foram adicionadas. A evolução de D_{KL} é mostrada na Fig. 5.4, que mostra que $\mathbf{q_x}(h)$ converge para $\mathbf{p_x}$ conforme o número de imagens na base de dados aumenta. A Fig. 5.5 mostra as distribuições massa de probabilidade $\mathbf{p_x}$ e $\mathbf{q_x}(420)$.

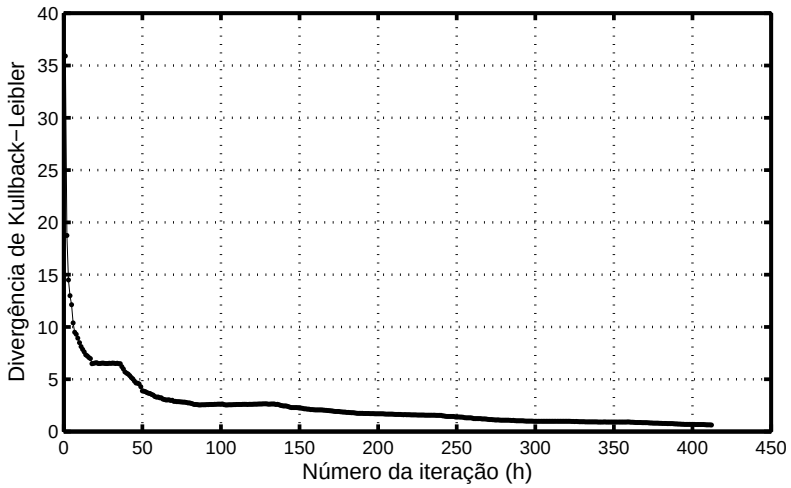

Figura 5.4: Divergência de Kullback-Leibler entre as distribuições $\mathbf{p_x}$ e $\mathbf{q_x}(h)$ em função do número de iterações do Algoritmo 1.

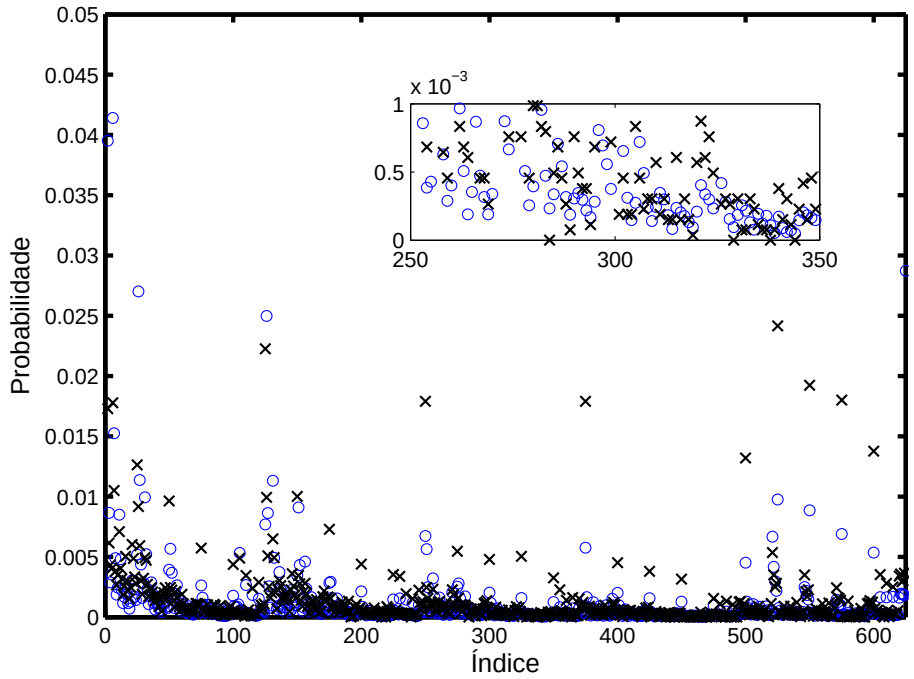

Figura 5.5: Exemplos de distribuições massa de probabilidade $\mathbf{p_x}$ e $\mathbf{q_x}(412)$. Enquanto $\mathbf{p_x}$ (indicada por 'o') é ideal e constante, $\mathbf{q_x}$ (indicada por 'x') é experimental e depende da iteração.

Capítulo 6

Conclusões

Neste trabalho, foi apresentado um algoritmo de calibração de dicionário que é adequado para um imageador CMOS usando VQ e leva a erro quadrático médio consistentemente menor que o obtido sem a aplicação do algoritmo. Visualmente, a Fig. 5.2 mostra que a representação das texturas se torna mais pronunciada. Para ilustrar as fontes de erro mais relevantes, foram apresentados alguns detalhes de implementação a respeito do imageador fabricado. A resiliência do sistema a erros na parte analógica havia sido avaliada por meio de simulações de Monte Carlo em um estudo anterior (página 64 de [22]). Este estudo anterior havia indicado que a perda de desempenho era suficientemente pequena para justificar a fabricação do protótipo. O principal objetivo do presente projeto é a melhoria do MSE depois de estabelecidos os erros de fabricação. Foi continuada a acumulação de dados experimentais, como sugerido em [20], e a D_{KL} convergiu para 0.6, que indica que a substituição da fonte de dados teórica pela experimental ocasionaria um aumento de 0.6 na taxa de bits em codificação direta (para um sistema hipotético de quantização com 256 vetores código). Podemos concluir, portanto, que a fonte de dados teórica é próxima à usada no projeto do VQ. A eficácia do método proposto para melhoria de MSE pode ser verificada na Tabela 5.1.

Imagens comprimidas com este imageador haviam sido reconstruídas com razão sinal-ruído de pico (PSNR) de cerca de 18 dB, para taxa de bits abaixo de 0.94 bit por pixel. O foco do presente trabalho é na melhoria do estágio de VQ do

imageador previamente apresentado. Uma comparação extensa, Tabela IV de [10], foi feita com 12 outros imageadores CMOS com diferentes algoritmos de compressão de imagens no plano focal, com PSNR variando de 13 dB a 48 dB. Enquanto o imageador proposto se encontra nesta faixa, ele corresponde à única abordagem cujo processamento de imagens é feito a nível dos píxels e na qual nenhum processamento é feito fora da matriz de pixels. Como não foram encontradas discussões sobre as vantagens de procedimentos de calibração para os outros imageadores, não foi apresentada nenhuma comparação a cerca dos possíveis ganhos em diferentes implementações. Um procedimento de calibração experimental é descrito em [23], mas este procedimento leva a correções em modelos para decomposição de imagens no plano focal em múltiplos níveis (não para codificação por blocos). Infelizmente, as conclusões e o aumento de desempenho relatados em [23] não podem ser comparados com os resultados obtidos com codificação por blocos.

Referências Bibliográficas

[1] OHTA, J. *Smart CMOS Image Sensors and Applications*. CRC Press, 2007.

[2] JENDERNALIK, W., BLAKIEWICZ, G., JAKUSZ, J., et al. "An Analog Sub-Miliwatt CMOS Image Sensor With Pixel-Level Convolution Processing", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 60, n. 2, pp. 279–289, Feb 2013. ISSN: 1549-8328. doi: 10.1109/TCSI.2012.2215803.

[3] LEON-SALAS, W., BALKIR, S., SAYOOD, K., et al. "A CMOS Imager With Focal Plane Compression Using Predictive Coding", *Solid-State Circuits, IEEE Journal of*, v. 42, n. 11, pp. 2555–2572, Nov 2007. ISSN: 0018-9200. doi: 10.1109/JSSC.2007.907191.

[4] CHEN, S., BERMAK, A., WANG, Y. "A CMOS Image Sensor With On-Chip Image Compression Based on Predictive Boundary Adaptation and Memoryless QTD Algorithm", *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, v. 19, n. 4, pp. 538–547, April 2011. ISSN: 1063-8210. doi: 10.1109/TVLSI.2009.2038388.

[5] ARTYOMOV, E., YADID-PECHT, O. "Adaptive Multiple-Resolution CMOS Active Pixel Sensor", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 53, n. 10, pp. 2178–2186, Oct 2006. ISSN: 1549-8328. doi: 10.1109/TCSI.2006.883161.

[6] PETROU, M., BHARATH, A. *Next Generation Artificial Vision Systems: Reverse Engineering the Human Visual System*. Artech House, 2008.

[7] RODRIGUEZ-VAZQUEZ, A., LINAN-CEMBRANO, G., CARRANZA, L., et al. "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 51, n. 5, pp. 851–863, May 2004. ISSN: 1549-8328. doi: 10.1109/TCSI.2004.827621.

[8] SARKAR, M., BELLO, D., VAN HOOF, C., et al. "Biologically Inspired CMOS Image Sensor for Fast Motion and Polarization Detection", *Sensors Journal, IEEE*, v. 13, n. 3, pp. 1065–1073, March 2013. ISSN: 1530-437X. doi: 10.1109/JSEN.2012.2234101.

[9] COTTINI, N., GOTTARDI, M., MASSARI, N., et al. "A 33 μ W 64×64 Pixel Vision Sensor Embedding Robust Dynamic Background Subtraction for Event Detection and Scene Interpretation", *Solid-State Circuits, IEEE Journal of*, v. 48, n. 3, pp. 850–863, March 2013. ISSN: 0018-9200. doi: 10.1109/JSSC.2012.2235031.

[10] OLIVEIRA, F., HAAS, H., GOMES, J., et al. "CMOS Imager With Focal-Plane Analog Image Compression Combining DPCM and VQ", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 60, n. 5, pp. 1331–1344, May 2013. ISSN: 1549-8328. doi: 10.1109/TCSI.2012.2226505.

[11] GERSHO, A., GRAY, R. M. *Vector Quantization and Signal Compression*. Boston, Springer, 1992.

[12] CHOU, P., LOOKABAUGH, T., GRAY, R. "Entropy-constrained vector quantization", *Acoustics, Speech and Signal Processing, IEEE Transactions on*, v. 37, n. 1, pp. 31–42, Jan 1989. ISSN: 0096-3518. doi: 10.1109/29.17498.

[13] COVER, T. M., THOMAS, J. A. *Elements of Information Theory*. New York, Wiley-Interscience, 1991.

[14] MALVAR, H., HALLAPURO, A., KARCZEWICZ, M., et al. "Low-complexity transform and quantization in H.264/AVC", *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 13, n. 7, pp. 598–603, July 2003. ISSN: 1051-8215. doi: 10.1109/TCSVT.2003.814964.

[15] SERACO, E. P., GOMES, J. G. R. C. "Computation of the complexity of vector quantizers by affine modeling", *Signal Processing*, v. 91, n. 5, pp. 1134–1142, maio 2011. ISSN: 0165-1684. doi: 10.1016/j.sigpro.2010.10.015.

[16] WANG, Y., BERMAK, A., BOUSSAID, F. "Reduced dimension Vector Quantization encoding method for image compression". In: *Design and Test Workshop (IDT), 2011 IEEE 6th International*, pp. 110–113, Dec 2011. doi: 10.1109/IDT.2011.6123112.

[17] KINGET, P. "Device mismatch and tradeoffs in the design of analog circuits", *Solid-State Circuits, IEEE Journal of*, v. 40, n. 6, pp. 1212–1224, June 2005. ISSN: 0018-9200. doi: 10.1109/JSSC.2005.848021.

[18] WU, X., ZHANG, M., VAN DER SPIEGEL, J. "High linearity current mode image sensor". In: *Electron Devices and Solid State Circuit (EDSSC), 2012 IEEE International Conference on*, pp. 1–4, Dec 2012. doi: 10.1109/EDSSC.2012.6482792.

[19] OTIM, S., CHOUBEY, B., JOSEPH, D., et al. "Characterization and Simple Fixed Pattern Noise Correction in Wide Dynamic Range "Logarithmic" Imagers", *Instrumentation and Measurement, IEEE Transactions on*, v. 56, n. 5, pp. 1910–1916, Oct 2007. ISSN: 0018-9456. doi: 10.1109/TIM.2007.903581.

[20] DE M ESTEVAO FILHO, R., GOMES, J., PETRAGLIA, A. "Codebook improvements for a CMOS imager with focal-plane vector quantization". In: *Circuits and Systems (LASCAS), 2013 IEEE Fourth Latin American Symposium on*, pp. 1–4, Feb 2013. doi: 10.1109/LASCAS.2013.6519052.

[21] *PIC18F4550 Data Sheet*. Microchip, 2006. Disponível em: <`http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf`>.

[22] HAAS, H. L. *Circuit Design for Focal Plane Image Compression in CMOS Cameras,*. Ph.D. dissertation, COPPE/PEE, Universidade Federal do Rio de Janeiro, Feb 2012. Disponível em: <`http://www.pee.ufrj.br/teses/textocompleto/2012020701.pdf`>.

[23] LIN, Z., HOFFMAN, M., SCHEMM, N., et al. "A CMOS Image Sensor for Multi-Level Focal Plane Image Decomposition", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, v. 55, n. 9, pp. 2561–2572, Oct 2008. ISSN: 1549-8328. doi: 10.1109/TCSI.2008.920094.