Multimédia II

Universidade Fernando Pessoa

CALIC

Elaborado por grupo:
Luís Aguiar, nº 36452
Pedro Almeida, nº 3684

Resumo

O intuito da realização do nosso vídeo é expandir o reportório da educação nesta área, desta forma facilitando a aprendizagem a pessoas que procurem informações à cerca deste tópico futuramente. Com este projeto procuramos também aprender aprofundadamente o funcionamento do algoritmo e os seus detalhes de implementação.

Índice

1. Introdução do vídeo

O algoritmo que abordamos foi o CALIC, este foi criado em 1994. É do tipo “lossless” ou sem perdas e dedica-se à compressão de imagens. Para a obtenção do valor comprimido, este baseasse na análise do contexto e previsão do valor do pixel.

2. Corpo do vídeo

Este algoritmo é deduzido através da noção de que em toda e qualquer imagem, um dado pixel, tem obrigatoriamente um valor semelhante a pelo menos um dos seus pixéis vizinhos.
O exemplo concreto que vamos utilizar para a nossa explicação é o valor real de um pixel, uma vez que o nosso algoritmo faz parte do ramo da codificação de imagens. E durante a nossa análise iremos converter este valor real num valor comprimido, de forma a explicar todos os passos durante esta conversão.

2.1 Codificação

		NN	NNE
	NO	N	NE
OO	O	X	

A imagem anterior demonstra como são etiquetados os pixéis, demonstrando assim que estes são etiquetados conforme a sua orientação em relação ao pixel ao qual pretendemos atribuir o valor comprimido.

Tendo em conta que os pixéis são escritos de cima para baixo e da esquerda para a direita, podemos concluir que, os valores necessários para a descodificação estão também disponíveis no descodificador.

Numa primeira parte, temos de determinar:

A diferença horizontal:

$$dh = |O-OO| + |N-NO| + |NE-N|$$

A diferença vertical:

$$dv = |O-NO| + |N-NN| + |NE-NNE|$$

A partir destes valores é possível obter uma previsão inicial do valor do pixel X, que é calculada a partir da diferença entre **dy** e **dh**:

Se dh >> (80) dv – existe muita disparidade horizontal, logo X previsto fica com o valor de N

Se dv >> (80) dh – existe muita disparidade vertical, logo X previsto fica com o valor de O

Caso a diferença seja moderada temos de recorrer às regras, que são os "if's". do algoritmo.

```
if (dh – dv > 80)
        Xp ← N
else if (dv – dh > 80)
        Xp ← O
else {
        Xp ← (N + O)/2 + (NE - NO)/4
        if (dh – dv > 32)
                Xp ← (Xp + N)/2
        else if (dv – dh > 32)
                Xp ← (Xp + O)/2
        else if (dh – dv > 8)
                Xp ← (3Xp + N)/4
        else if (dv – dh > 8)
                Xp ← (3Xp + O)/4
}
```

Caso não entre nos dois primeiros casos, em que a diferença é elevada, entra nas regras secundárias "if's". Ao entrar nestas últimas é numa primeira parte calculado o X previsto através da fórmula **Xp ← (N + O)/2 + (NE – NO)/4.** Este Xp é depois utilizado nos cálculos das regras secundários, consoante a diferença entre dh e dv se encaixe num dos "if's"., é calculado um novo Xp utilizando o Xp antigo.

Após calcular o valor previsto do pixel passamos à computação do seu contexto.
Formamos primeiramente o vetor:

C = {N, W, NW, NE, NN, WW, 2N-NN, 2W-WW}

De seguida forma-se o vetor binário **(1)** usando o mapeamento **(2)**, sendo que cada *bk*

corresponde a um dos bits

(1)

$$\alpha = b7, b6, b5, b4, b3, b2, b1, b0$$

(2)

$$bk = 0 \text{ se } xk >= xp \text{ ou } bk = 1 \text{ se } xk < xp$$

Quantificamos uma quantidade que incorpora as variações verticais, horizontais e o erro previsto:

$$\delta = dh + dv + 2\,|N - \hat{N}|$$

Este range de valores é dividido em **4 intervalos**, cada um representado por **2 bits**.
Depois de obtida a previsão, a diferença entre o valor do pixel e a previsão tem de ser codificada. Embora o processo de previsão remova muito da estrutura contida na sequencia original, ainda existe alguma estrutura na sequencia residual.
Sendo assim tiramos partido desta estrutura e codificamos o seu contexto. Este contexto residual corresponde ao valor obtido na formula. Para reduzir a complexidade da codificação em vez de usar o valor real, o CALIC usa um intervalo de valores em que o resultado da formula e o contexto.

$0 \le \delta < q1$	**contexto 1**
$q1 \le \delta < q2$	**contexto 2**
$q2 \le \delta < q3$	**contexto 3**
$q3 \le \delta < q4$	**contexto 4**
$q4 \le \delta < q5$	**contexto 5**
$q5 \le \delta < q6$	**contexto 6**
$q6 \le \delta < q7$	**contexto 7**
$q7 \le \delta < q8$	**contexto 8**

2.2. Exemplo de aplicação

Utilizando esta tabela que indica os valores de vizinhança do pixel:

		200 (NN)	100 (NNE)
	200 (NO)	100 (N)	100 (NE)
200 (OO)	100 (O)	X	

(Tabela 1)

dh = |100-200| + |100-200|+|100-100| = 200

dv = |100-200| + |100-200| + |100-100| = 200

d = dv- dh = 200 - 200 = **0**

Como **d = 0**, não entra em nenhuma das regras iniciais, nem em nenhuma das regras secundárias, ou seja, o único calculo que temos de fazer é o de Xp:

Xp ← (N + O)/2 + (NE – NO)/4

Xp = (100 + 100)/2 + (100-200)/4 = **75**

Supondo que o X real inicial era 100, o nosso prediction error seria:

Prediction error = X – Xp = 100 - 75 = 25

Para o exemplo do quadro de valores acima (Tabela 1), usando a as indicações de formação do vetor será originado o vetor:

C = {100, 100, 200, 100, 200, 200, 0, 0}

Apos formar o vetor C já podemos formar o vetor binário. Para calcular cada valor bk (bit do vetor binário) temos de recorrer ao uso do mapeamento (2).

Ex.: Calculo do primeiro bit do vetor binário usando o primeiro valor do vetor C.

Para x0 = 100 e Xp = 75, b0 = 0 pois 100 >= 75

Calculo dos restantes bits do vetor binário resultam em:

b1 = 0; b2 = 0; b3 = 0; b4 = 0; b5 = 0; b6 = 1; b7 = 1;

resultado no vetor binário: α = **11000000** (8 bits)

Apos calcularmos o vetor binário do pixel calculamos o contexto em que este se insere.

Para tal calculo usamos a formula: $\delta = dh + dv + |X - Xp|$ e de seguida inserimos o mesmo dentro de um dos oito contextos estipulados.

2.3. Comparação com outros métodos semelhantes

Num estudo realizado por outrem, em que comparam o CALIC com o JPEG-LS (JPEG mais recente) e com o JPEG (JPEG mais antigo), foi demonstrada a eficácia do CALIC, uma vez que, em 6 de 7 imagens de teste o algoritmo foi o que teve melhores resultados.

Esta tabela demonstra alguns desses resultados expressados através do tamanho dos ficheiros comprimidos obtidos por cada algoritmo:

Image	Old JPEG	New JPEG	CALIC
Sena	31,055	27,339	26,433
Sensin	32,429	30,344	29,213
Earth	32,137	26,088	25,280
Omaha	48,818	50,765	48,249

3. Conclusão do vídeo

Este “codec” tem como principal e única funcionalidade a compressão de imagens, funcionalidade esta que desempenha com extrema eficácia. Demonstra um rácio de compressão muito elevado, numa grande maioria dos casos melhor até que a norma JPEG (como podemos observar no ponto anterior), norma esta que é a que tem uma maior taxa de utilização na atualidade.

Ao realizar o projeto encontramos muitas dificuldades em entender algumas partes do algoritmo, partes estas que levaram a uma pesquisa exaustiva para a sua compreensão, uma vez que, a informação disponível é muito reduzida e de difícil compreensão. Não encontramos também nenhuma informação referente à descodificação, tanto nos documentos recomendados pelo professor, como na internet.

4. Desenvolvimento do vídeo/tutorial sobre o "*codec*"

Para gravação do vídeo utilizamos a ferramenta Debut Video Capture and Screen Recorder Software da NCH Software. Para unir as partes dos dois integrantes do grupo utilizamos uma ferramenta chamada MP4 Joiner. Por fim, para edição de vídeo, por exemplo adição de setas de forma a tornar o vídeo mais interativo, utilizamos a ferramenta Fotografias do Windows.

Com a realização deste vídeo aprendemos como funciona o "codec" e chegamos à conclusão que este é extremamente eficaz na compressão de imagens.

As maiores limitações durante a realização do projeto foram a escassez de informação, principalmente referente à descodificação, tópico este que não conseguimos abordar exatamente por este mesmo motivo.

Bibliografia

David Motta, Handbook of Data Compression, 5th Edition, Springer-Verlag London, 2010.

Khalid Sayood, CALIC, 3rd Edition, Introduction tression-Morgan Kaufmann, 2017.

Outras referências bibliográficas e *online* utilizadas para estudar e compreender o algoritmo.