MSX Article

Error Diffusion

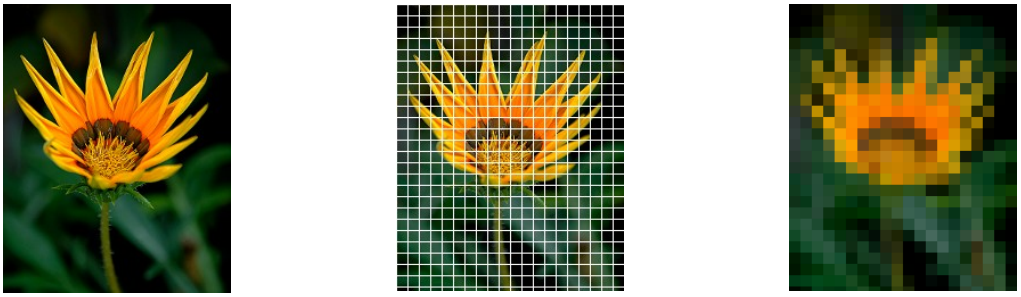
Resumo

O objetivo deste artigo é apresentar o método de redução de cores Error Diffusion, utilizado no projeto MSX Viewer.

1- Introdução

Quando reduzimos a quantidade de cores de uma imagem, a perda na qualidade é inevitável. Porém, esta perda de qualidade pode ser minimizada através de alguns métodos de redução de cor.

Uma imagem digital é uma representação discreta do mundo real, tanto pelo aspecto espacial, como espectral e radiométrico. A figura 1 ilustra o processo de aquisição de uma imagem digital.



a) Mundo real b) Discretização c) Imagem digital

Figura 1. Aquisição de imagem digital.

A grade branca da figura 1b representa o tamanho do pixel que o sensor do exemplo irá amostrar a imagem. Quanto menor for tamanho do pixel, mais próximo da realidade estará a imagem digital resultante. Esta é a resolução espacial da imagem.

A resolução espectral diz respeito à capacidade do sensor em perceber determinadas faixas de freqüência e também separá-las em bandas. Quanto maior for a resolução espectral, maior a capacidade de distinguir faixas de freqüências em uma determinada banda. Uma imagem em tons de cinza (monocromático) possui apenas uma banda espectral, enquanto que uma imagem colorida possui três bandas: vermelho, verde e azul.

A resolução radiométrica é a capacidade de um sensor em discriminar as diferentes intensidades de energia eletromagnética em uma banda, e é expressa em bits ou níveis. A figura 2 apresenta um espectro monocromático e sua discretização em diferentes níveis.

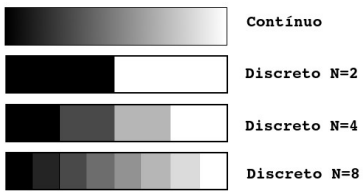


Figura 2. Resolução radiométrica.

As imagens digitais coloridas comuns, captadas a partir de câmeras ou celulares, possuem resolução radiométrica de 8 bits ou 256 níveis. O MSX 2 possui resolução de 3 bits ou 7 níveis em cada canal de cor, totalizando 512 cores. Já o MSX 2+ e o processador gráfico v9990 possuem resolução de 5 bits ou 32 níveis por canal.

As imagens digitais deverão ter seu espaço de cor reduzido para se tornarem compatíveis com o MSX. Entretanto, no caso do MSX 2, cuidados especiais deverão ser tomados para se obter uma imagem de melhor qualidade. O processo de discretização do espaço de cor de uma imagem é chamado de quantização [1].

2- Quantização

Chamamos de quantização a transformação de um espaço de cor C, no qual as cores são representadas por M bits, para o espaço de cor C', representado por N bits, onde M > N.

Se os espaços de cor C e C' são unidimensionais (1 banda), a quantização é chamada de quantização escalar ou unidimensional. Caso contrário, ela é chamada de quantização vetorial ou multidimensional (figura 3). Dado que uma imagem colorida possui o espaço de cores multidimensional, devemos aplicar a quantização multidimensional, através da quantização escalar em cada banda ou componente de cor [1].

a) imagem em tons de cinza (escalar) b) imagem colorida (vetorial)

Figura 3. Espaço de cor escalar e vetorial.

No exemplo a seguir, desejamos converter uma imagem em tons de cinza com resolução de 8 bits (256 níveis) para uma imagem com 1 bit (2 níveis). Uma possível solução é quantizar os valores com intensidade abaixo do valor médio do canal para o valor 0, e o restante para o valor 1. Este processo de quantização particiona o espaço de cor original em dois conjuntos [1].

Sendo *I(x,y)* a intensidade de um pixel da imagem de 8 bits e *T(x,y)* o pixel correspondente na imagem de 1 bit, a quantização é realizada da seguinte maneira:

```
if I(x,y) < 128
  T(x,y) = 0;
else
  T(x,y) = 1;
```

A figura 4 apresenta o resultado da quantização de 8 bits para 1 bita aplicada em uma imagem.

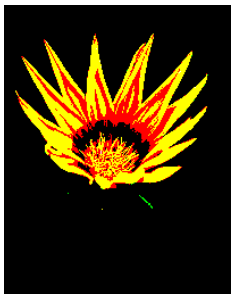
a) imagem em tons de cinza (N=8, 256 níveis)

b) imagem quantizada (N=1, 2 níveis)

Figura 4. Quantização escalar.

Conforme mencionado anteriormente, a quantização pode ser aplicada a imagens coloridas. Nesse caso, aplica-se a quantização em cada canal de cor separadamente. O resultado da quantização vetorial pode ser visto na figura 5.

a) imagem colorida (N=8, 256 níveis por canal)



b) imagem quantizada (N=1, 2 níveis por canal)

Figura 4. Quantização vetorial.

Para imagens quantizadas com $N > 1$, o processo é o mesmo, ou seja, dividir o espaço de cores original em 2^N conjuntos e verificar a qual conjunto pertence o pixel $I(x,y)$ em questão, através da seguinte fórmula:

$$T(x,y) = I(x,y) \times \frac{2^N}{256}$$

Importante:
Arredondar para baixo o valor obtido.

Nesse caso, cada conjunto é um nível de intensidade do sistema de cor destino.

A figura 5 ilustra como é feita a quantização de um pixel de um sistema com N=8 (256 níveis) para outro com N=3 (8 níveis).

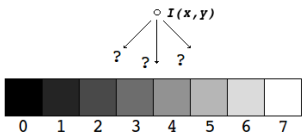


Figura 5. Quantização.

Caso se deseje visualizar a imagem de um sistema com N bits em um sistema com 8 bits por canal, deve-se aplicar a seguinte fórmula para se obter os níveis de intensidade:

$$I'(x,y) = T(x,y) \times \frac{255}{2^N - 1}$$

Outro método de quantização utilizado para imagens coloridas é aplicado a imagens resultantes do tipo indexada (paleta de cores) [3]. Nesse caso, o espaço de cor resultante não é mais uma sub-divisão regular do espaço de cor, como nos exemplos anteriores, e sim um sub-conjunto de *N* cores quaisquer do espaço de cores.

Em um sistema de cores indexado, valor do pixel *T(x,y)* não possui diretamente os valores das intensidades de cor, mas sim um índice para uma tabela (paleta de cores), que contém as intensidades de cor daquele pixel.

A solução para a quantização de imagens desse tipo é verificar para cada pixel *I(x,y)* da imagem original, qual a cor da paleta de cores é a mais parecida com a cor dele. A Distância Euclidiana é utilizada para calcular a distância entre a cor do pixel e a cor de um índice da paleta. O índice *i* da paleta *P*, cuja distância calculada em relação ao pixel for o menor valor, será considerada a cor mais parecida. Assim, temos:

$d_{min} = \{ [I(x,y,red) - P(1,red), 2)]^2 + [I(x,y,green) - P(1,green), 2)]^2 + [I(x,y,blue) - P(1,blue), 2)]^2 \}^{1/2}$
$T_{best} = 1$
for i = 2 to N
  $d = \{ [I(x,y,red) - P(i,red), 2)]^2 + [I(x,y,green) - P(i,green), 2)]^2 + [I(x,y,blue) - P(i,blue), 2)]^2 \}^{1/2}$
  if $(d < d_{min})$
    $d_{min} = d$
    $T_{best} = i$
next N
$T(x,y) = T_{best}$

3- O Problema da Quantização: Bandas de Mach

Quanto menor o nível *N* da quantização da imagem, o olho humano começa a perceber os contornos de quantização. Esse fenômeno é chamado de Bandas de Mach [1].

Observe o exemplo de quantização da figura 6. Quanto menor o espaço de cor resultante, mais visível se tornam os contornos de quantização.

a) 8 bits / 256 níveis b) 6 bits / 64 níveis c) 3 bits / 8 níveis d) 2 bits / 4 níveis

Figura 6. O problema da quantização – bandas de mach.

Observe também a figura 2, comparando a primeira linha (contínua) com a última (N=8). Na última linha, as transições de cores são bem mais visíveis que na primeira linha.

4- Error Diffusion

Foi visto na seção anterior, que a quantização para um espaço de cor pequeno gera um problema na imagem resultante. No caso das screens 2-8 do MSX, os efeitos da bandas de mach são intensos, uma vez que esses modos de tela possuem o espaço de cor pequeno.

A técnica de Error Diffusion (Floyd-Steinberg) tem como objetivo minimizar os efeitos das Banda de Mach, tentando fazer com que os contornos de quantização não sejam percebidos.

O processo geral do Error Diffusion é o seguinte:

- Quantizar o pixel atual $I(x,y)$, resultando em $T(x,y)$ e depois $I'(x,y)$.
- Calcular o erro E, onde: $E = I(x,y) – I'(x,y)$.
- Aplicar o erro aos pixels vizinhos, utilizando-se uma máscara.

A máscara utilizada é:

1	0	*	7
16	3	5	1

Assim como na quantização, o algoritmo do Error Diffusion deve ser aplicado a cada componente de cor separadamente.

O Apêndice em inglês do MSX Viewer 5 [2] apresenta dois algoritmos para aplicar o Error Diffusion em imagens: um para imagens RGB (sem paleta), outro para imagens indexadas (paleta).

Programas em C para o Error Diffusion

```
for (int y=0; y<img.height(); y++)
{
  for (int x=0; x<img.width(); x++)
  {
    // Pixel quantization
    old_pixel = img(x,y);
    new_pixel = quantize(old_pixel);
    new_img(x,y) = new_pixel;

    for (int c=0; c<3; c++)
    {
      // Calculate quantization error
      quant_error = old_pixel(c) – new_pixel(c);

      // Spread error
      img(x+1,y,c) += quant_error * 7/16;
      img(x-1,y+1,c) += quant_error * 3/16;
      img(x,y+1,c) += quant_error * 5/16;
      img(x+1,y+1,c) += quant_error * 1/16;
    }
  }
}
```

Imagem RGB (8 bits, 3 canais)

```
for (int y=0; y<img.height(); y++)
{
  for (int x=0; x<img.width(); x++)
  {
    // Pixel quantization
    old_pixel = img(x,y);
    index = quantize(old_pixel);
    new_pixel = palette(index);
    new_img(x,y) = index;

    for (int c=0; c<3; c++)
    {
      // Calculate quantization error
      quant_error = old_pixel(c) – new_pixel(c);

      // Spread error
      img(x+1,y,c) += quant_error * 7/16;
      img(x-1,y+1,c) += quant_error * 3/16;
      img(x,y+1,c) += quant_error * 5/16;
      img(x+1,y+1,c) += quant_error * 1/16;
    }
}
```

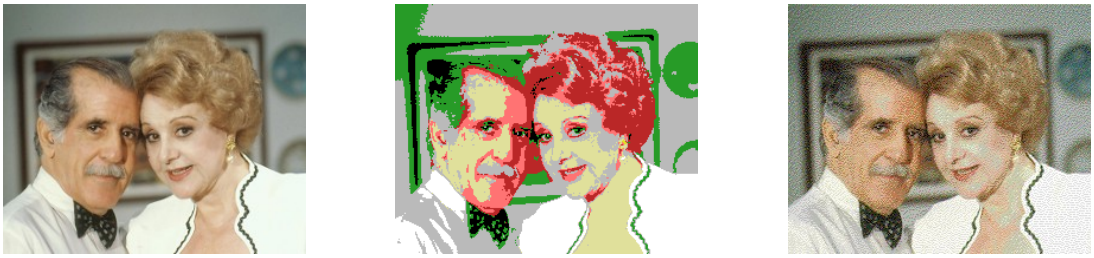
Imagem indexada

Na figura 7, observamos o resultado da aplicação da técnica de Error Diffusion a uma imagem. Compare os resultados com a figura 6.

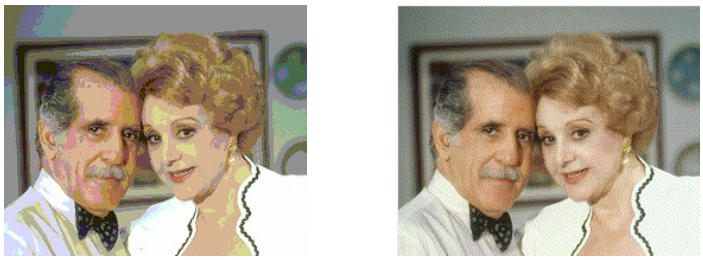
a) 8 bits / 256 níveis b) 6 bits / 64 níveis c) 3 bits / 8 níveis d) 2 bits / 4 níveis

Figura 7. Aplicação do Error Diffusion.

A figura 8 apresenta a conversão de uma imagem para as screens 7 e 8, utilizando as técnicas de quantização e Error Diffusion.



a) Original b) Screen 7, Quantização c) Screen 7, Error Diffusion



d) Screen 8, Quantização e) Screen 8, Error Diffusion

Figura 8. Comparação entre as técnicas de quantização e Error Diffusion no MSX.

5- Créditos e Bibliografia

O artigo foi escrito por Marcelo Silveira, Engenheiro de Sistemas e Computação, formado pela UERJ.

Data: Maio de 2005, revisado em Julho de 2017 e Maio de 2018.
E-mail: flamar98@hotmail.com
Homepage: marmsx.msxall.com

Referências:

[1]- Computação Gráfica: Imagem, Luiz Velho e Jonas Gomes, IMPA.
[2]- MarMSX, MSX Viewer 5, Appendix, em http://marmsx.msxall.com.
[3]- Error Diffusion, Wikipedia, em http://en.wikipedia.org/wiki/Error_diffusion