Universidade Federal de Santa Maria
Departamento de Eletrônica e Computação
Prof. Cesar Tadeu Pozzer
Disciplina: Computação Gráfica
08/03/2018
pozzer@inf.ufsm.br

Computação Gráfica - Imagem

Imagem

- É a materialização de grande parte dos processos da computação gráfica (síntese de imagens)
- Serve como elo de ligação entre usuário e os procedimentos (resultados)
- Está presente em todas as áreas da CG, seja como produto final (visualização), ou como parte do processo de interação (modelagem) [6]

Modelo matemático

- Uma fotografia ou uma cena real é composta por um conjunto de pontos. Cada ponto é visualizado por meio de um impulso luminoso que determina a cor do ponto.
- Uma imagem pode ser definida como uma função definida em uma superfície bidimensional, cujos valores estão dentro de um espaço de cores.
- Uma imagem pode ser definida por

$$f : U \subset R^2 \rightarrow C$$

  onde
  - U é um conjunto chamado suporte da imagem. O conjunto de valores de f, que é um subconjunto de C, é chamado de conjunto de cores da imagem.
  - $C=R^n$ é um espaço de cor (conjunto de cores). Se n for 3, temos um espaço de representação de cores tricromático, em geral um espaço com base de primárias R, G e B.
- Uma imagem monocromática pode ser representada geometricamente como um gráfico G(f) (semelhante à representação de um terreno)

$$G(f) = \{(x, y, z); (x, y) \in U \ e \ z = f(x, y)\}$$

Representação espacial

- Representação matricial: utiliza-se a amostragem matricial para fazer a discretização espacial da imagem.

$$U = [a,b] \times [c,d] = \{(x, y) \in R^2; a \le x \le b \ e \ c \le y \le d\}$$

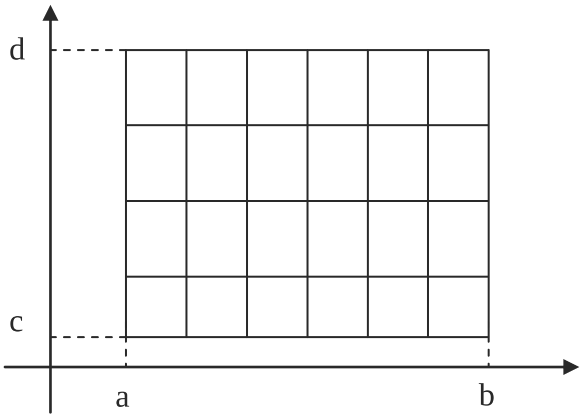
- Este retângulo é discretizado usando os pontos do reticulado bidimensional. Se a=c=*0*, o reticulado de discretização é o conjunto

$$P_\Delta = \{(x_j, y_k) \in R^2\}$$

onde

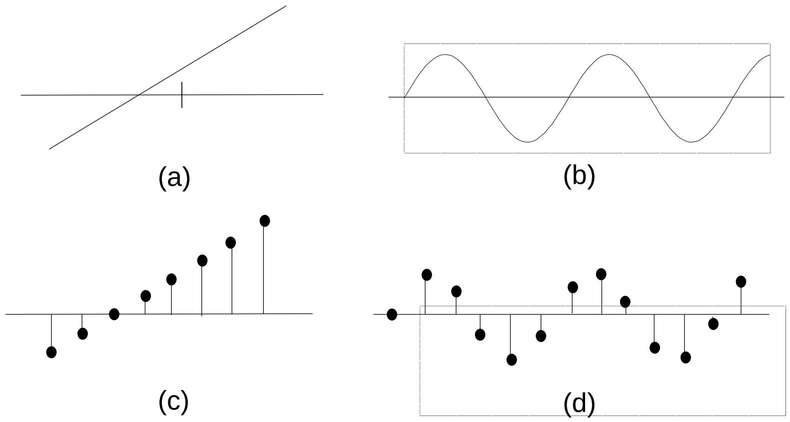
$$x_j = j.\Delta x,\ j = 0,1,\ldots,m-1,\ \Delta x = b/m$$

$$y_k = k.\Delta y,\ k = 0,1,\ldots,n-1,\ \Delta y = d/n$$



Amostragem

- Ao contrário dos sinais contínuos, sinais discretos (amostrados) são representados matematicamente como uma seqüência finita de números, denotada da forma x[n], onde n é definido somente para valores inteiros e representa o n-ésino elemento da seqüência.
- Esta representação possui uma fácil implementação em computadores digitais com uso de matrizes multidimensionais e por isso, é a forma como os sinais são tratados por processos computacionais.

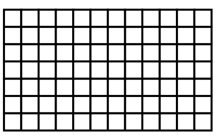


(a) (b)

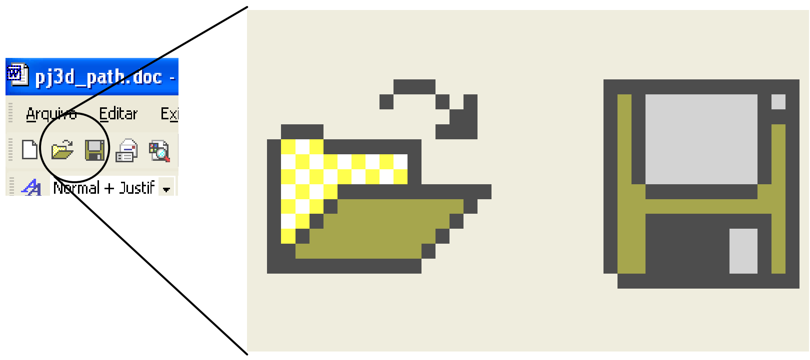
(c) (d)

Implementação

- Matriz retangular de pontos.
- Cada ponto é chamado de pixel.
- Cada pixel tem uma cor associada.



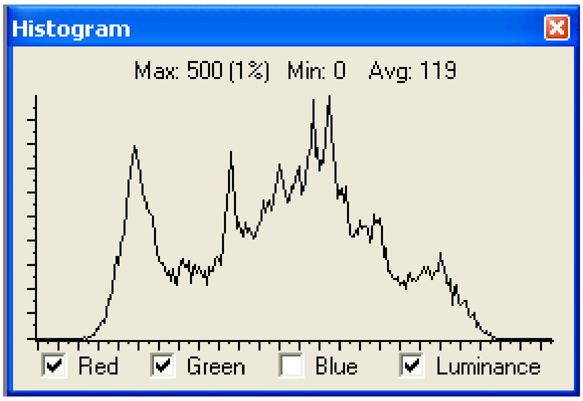
Matriz de pontos de uma imagem

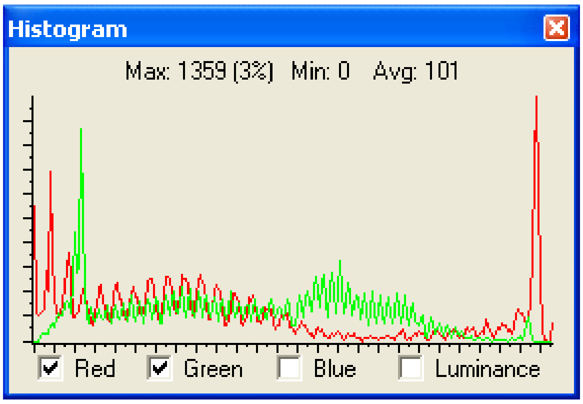


Região de uma imagem Ampliada, para ressaltar os pixels

Histograma

- Número de ocorrências de cada cor





Quantização

- É o processo de discretização de cor
- Permite a conversão de uma imagem com um conjunto contínuo de cores, em uma imagem com um conjunto discreto, com o objetivo de reduzir o espaço de armazenamento

256 tons | 16 tons | 2 tons (1 bit)

Imagem: Dimensão

- Dimensão: número de pixels nos eixos *x* e *y*
- O número de pixels de uma imagem é dado pela multiplicação da base pela altura. Logo, uma imagem com dimensão de 640 x 480 (padrão VGA) tem 307.200 pixels.
- Geralmente, a dimensão de imagens está associada com resoluções suportadas pelos monitores de vídeo. As resoluções padrão são (Todos estes formatos adotam a proporção 4x3):
  - VGA (640 x 480)
  - SVGA (800 x 600)
  - XGA (1024 x 768)
  - SXGA (1280 x 1024)

Imagem: Resolução

- Resolução ≠ Dimensão
- Resolução é medida em DPI (*Dots per inch* – Pontos por polegada). 1''= 2.54 cm
- Representa a razão entre o número de pixels por unidade de área.
- A resolução dos monitores de vídeo varia entre 72 e 100 DPI, enquanto as impressoras variam entre 300 e 2880 DPI.
- Como exemplo, para que uma foto em tamanho 10x15 seja impressa em resolução de 300 dpi, será necessário uma imagem com resolução de 1180 x 1770 pixels (10/2.54*300 x 15/2.54*300), ou seja, aproximadamente 2 Mega Pixels.
- A qualidade de impressão de uma imagem depende de dois fatores: da resolução da imagem e da resolução da impressora.

Imagem: Cores

- Número de cores: monocromática ou colorida
- Para representar uma imagem monocromática, necessita-se armazenar apenas 1 bit por pixel, ou seja, o pixel é preto (bit=0) ou branco (bit=1).
- Para imagens coloridas (definidas pelas componentes RGB – Vermelho, Verde e Azul), existem vários padrões:
    - 1 bit: 2 cores - preto/branco
    - 8 bits: 256 cores
    - 16 bits: padrão 565 → 65.536 cores
    - 24 bits: 3 bytes/pixel → 16.777.216 cores (256 x 256 x 256).
    - 32 bits: igual ao 24 + canal alpha
- Paleta de cores
    - GIF, BMP

Formato de dados gráficos

- Formato vetorial
    - A informação é representada por um conjunto de segmentos de retas, curvas descritos pelas coordenadas de seus pontos iniciais e finais
    - É utilizada em geral para descrever a estrutura geométrica de seus objetos gráficos
- Formato matricial - raster
    - A informação é representada por uma matriz onde cada elemento é uma estrutura de dados associados a cor e outras componentes da imagem
    - É frequentemente associada a imagem digital
- Conversão entre formatos
    - Vetorial → Matricial: rasterização
    - Matricial → Vetorial: área de reconhecimento de padrões

Imagem x Figura: Resolução

- Figura x Imagem
- Figura Vetorial x Figura Raster
- Geralmente figuras usam representação vetorial: pontos, retas, círculos, etc.
    - No caso de uma reta, por exemplo, armazenam-se apenas as coordenadas inicial e a final, ao invés de todos os "infinitos pontos" que fazem parte da reta, considerando-se que ela tenha uma largura e estilo conhecidos.
    - Isso traz economia de espaço em memória, além de oferecer uma melhor qualidade de impressão.
- Para figuras vetoriais, e a qualidade de impressão depende exclusivamente da resolução da impressora.

- Se uma imagem tiver resolução de tela, certamente apresentará uma impressão de baixa qualidade, independente da impressora utilizada. Se as figuras tiverem uma representação baseada em pixels, como ocorre com as imagens, a qualidade de impressão geralmente é pior do que com imagens, considerando-se as mesmas resoluções.

Formato de Imagens

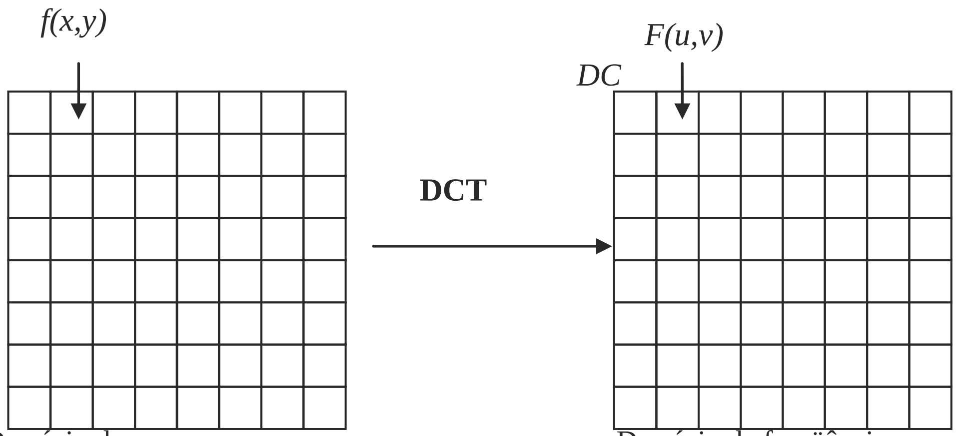
- Resultado de Necessidades:
  - Resolução de cores
  - Resolução geométrica
  - Precisão: cores e geometria
  - Processamento
  - Compactação: eliminação da redundância do sinal
  - Compressão: há perda de informação
    - Redução do domínio: resolução geométrica
    - Quantização: cores
    - Transformada: freqüências
- Formatos
  - BMP, GIF, JPEG, TGA, PNG, TIFF, etc.

Formato JPEG

- Formato de arquivo de imagem que tem como principais características apresentar [1]:
  - Compactação da imagem
  - Compressão da imagem variável
  - Modo progressivo
  - Modo Hierárquico: em diferentes resoluções (*Mipmap*)
  - 16 M cores
  - Faz uso da DCT (*Discrete Cossine Transform*) para comprimir a imagem

Transformadas

- Tem a função de transformar sinais do domínio tempo-espaço para o domínio da freqüência. Diversos processamentos em sinais podem ser melhor realizados no espectro da freqüência, como no caso da compressão de imagens;
  - Uma imagem é um sinal no domínio do espaço
  - Áudio é um sinal no domínio do tempo
- A DCT é similar a transformada discreta de Fourier: ela transforma um sinal ou imagem (1D ou 2D) do domínio espacial para o domínio da freqüência.
- Uma imagem no domínio da freqüência é caracterizada pela variação de tom entre pixels vizinhos. Quanto maior for a variação de intensidade, maior é a freqüência da imagem. Imagens que caracterizam um texto escaneado, por exemplo, são marcadas por altas freqüências, visto que existem transições freqüentes do branco (cor do papel) para o preto (cor da letra);
- A DCT tem a característica de concentrar a maior parte da energia em seus coeficientes iniciais, ou seja, de baixas freqüências;
- Imagens contêm uma grande parcela de redundância, ou seja, é grande a correlação entre os pixels formadores da imagem. Os processos de codificação por transformada reduzem a correlação. Técnicas que aplicam transformada geralmente dividem a imagem em blocos para diminuir a complexidade do algoritmo. A imagem é dividida em blocos de tamanho fixo, geralmente 8x8, 16x16, etc. Cada bloco é tratado como um vetor e codificado com a transformada, independentemente dos outros blocos. Posteriormente, para fazer a decodificação, este vetor é processado com a operação inversa da transformada, produzindo o vetor reconstruído. Os blocos reconstruídos são reunidos novamente para formar a imagem [9];
- A idéia é encontrar N valores (**coeficientes**) e N funções (**formas de onda ortogonais**) que, quando multiplicados e somados, podem reconstruir qualquer informação (representar quaisquer N amostras).



Domínio do espaço
Matriz de pixels em *Y*, *R*, *G* ou *B*

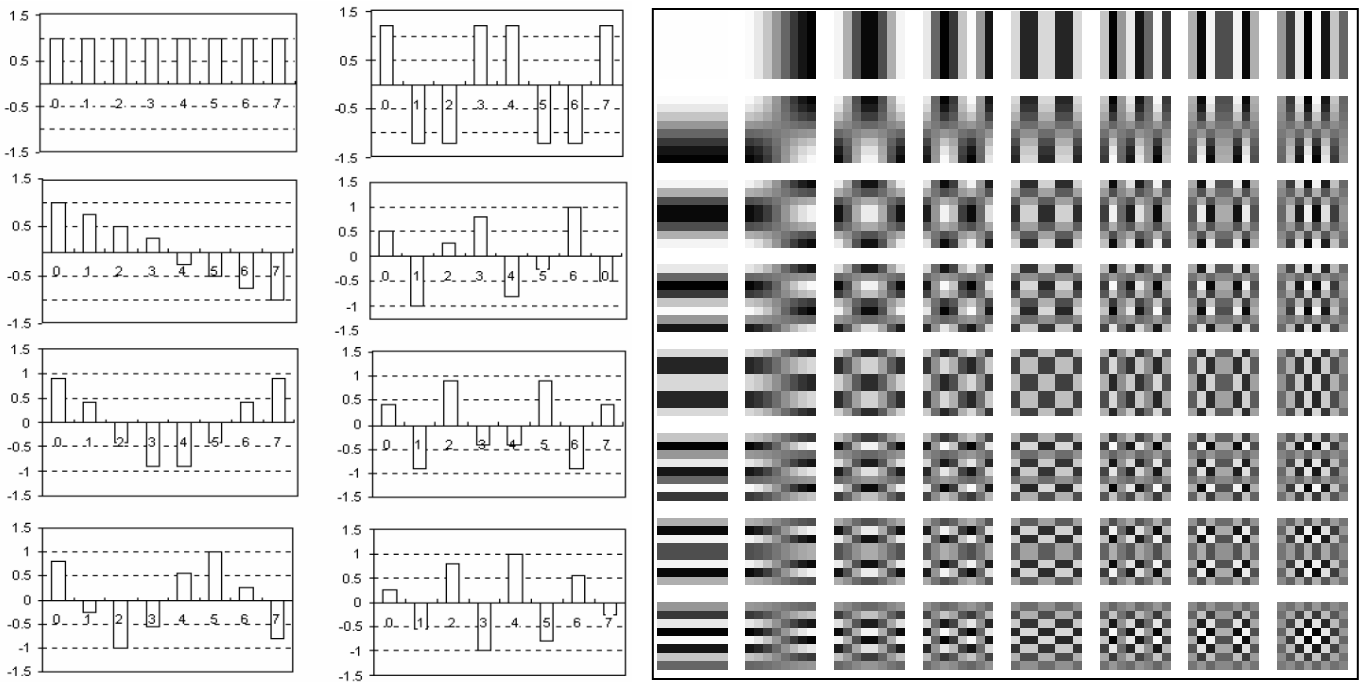
Domínio de freqüência
Matriz de coeficientes reais

- Se a imagem for colorida (R,G,B), deve-se aplicar a DCT para cada componente da imagem. Se for monocromática, deve-se aplica somente sobre a luminância *Y*.
- O conceito de formas de onda ortogonais é o mesmo conceito utilizado para definir uma base para um espaço R^n, como no caso do R^2 ou R^3. No espaço R^3, por exemplo, define-se 3 vetores (1,0,0), (0,1,0) e (0,0,1) que são a base do sistema. Esses vetores são *LI* (Linearmente independentes), ou seja, um não pode ser expresso como combinação linear dos demais. O mesmo vale para as formas e onda ortogonais.

Formas de onda ortogonais

1D – 8 ondas

2D – 64 ondas



Para visualizar as 64 Formas de onda ortogonais utilizadas na compressão de imagens JPEG, pode-se utilizar o seguinte algoritmo. Observe que forma retirados os coeficientes c(w). Como exercício, implemente este código na Canvas2D

```
for(u=0; u<8; u++)
   for(v=0; v<8; v++)
      for(x=0; x<8; x++)
         for(y=0; y<8; y++)
         {
            //pix vale entre -1 e 1.
            pix = cos(((2*x+1)*pi*u)/16.0) * cos(((2*y+1)*pi*v)/16.0);
            //normaliza entre 0 e 2
            cor = pix+1;
            //normaliza entre 0 e 255
            cor = cor*127;
            putpixel(u*10 + x, v*10+y, cor);
         }
```

Transformada Coseno Direta e Inversa

A formulação geral da transformada direta cosseno bidimensional direta (*Forward DCT*) e inversa (*Inverse DCT*), para o caso 2D (imagem) é dada por [8]

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) * \cos\frac{(2x+1)u\pi}{2N} * \cos\frac{(2y+1)v\pi}{2M} C(u)C(v)$$

$$f(x,y) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u,v) * \cos\frac{(2x+1)u\pi}{2N} * \cos\frac{(2y+1)v\pi}{2M} C(u)C(v)$$

onde

$$C(w) = \frac{1}{\sqrt{2}} \; para \; w = 0$$
$$C(w) = 1 \; para \; w = 1,\ldots,N-1$$

- M, N representam a dimensão da imagem (NxM)
- *f(x,y)* representa o valor dos pixels da imagem na posição *(x,y)* → *z = f(x,y)*.
- *F(u,v)* são os coeficientes da FDCT (valores reais associados à freqüência da imagem), ou seja, valores que quando multiplicados pelas funções ortogonais vão recriar os dados originais (pixels).
- Cada elemento transformado *F(u,v)* é o produto interno entre *f(x,y)* e NxM vetores básicos
- $-1 \le \cos(x) \le 1$
- Aplicando-se sobre uma imagem (ou outra função qualquer) a FDCT e em seguida a IDCT, obtém-se o sinal original. Deve-se observar que podem ser induzidos erros de arredondamento, visto que a matriz *F(u,v)* contém valores reais.

Para o caso unidimensional temos o seguinte

$$f(x) = \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{u=0}^{M-1} F(u) * \cos\frac{(2x+1)u\pi}{2M} C(u)$$

Essa transformada pode ser expandida para qualquer dimensão. Para o caso de uma matriz 8x8, a transformação direta e inversa é dada por [1]

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

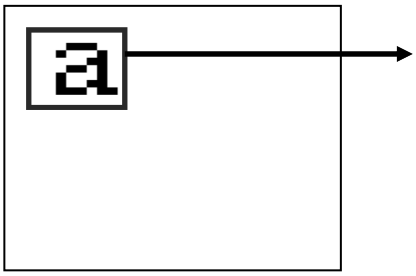
A transformada direta pode ser otimizada evidenciando-se os fatores *c(w)*

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

Transformada Co-seno – Matriz de coeficientes

- Imagem com alta variação de freqüência (cor) – **Texto.** O valor dos coeficientes é distribuído de forma bastante homogênea na matriz.

F(0,0)

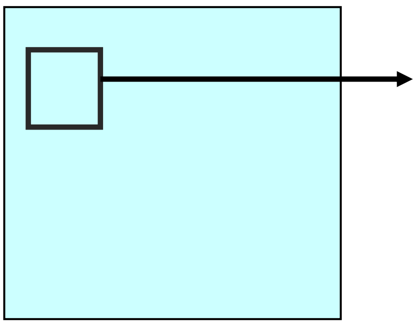


1402	-37	318	-106	255	158	-62	187
229	-1	6	62	-216	-41	-41	-140
-34	-46	-45	120	14	-73	146	-69
162	-17	-51	15	-69	5	-46	3
0	69	34	67	0	-216	-83	-23
162	86	-134	-275	-24	265	69	-17
82	-12	-236	-34	201	6	45	0
0	-39	-33	41	67	-27	0	0

- Imagem com baixa variação de freqüência (Cor) – **Pôr do sol.** Os coeficientes com valores significativos se concentram em baixas freqüências.

1442	33	41	45	44	38	25	12
29	-4	-7	-1	-1	-2	0	1
49	-3	-1	-5	-2	-2	-2	0
43	-2	0	-2	-2	-2	-2	-2
45	-3	-3	-3	-3	-1	0	-2
37	-4	-1	-2	-2	-1	-2	0
26	0	0	-1	-2	0	0	0
13	0	-2	0	-3	-2	0	0

- Imagem sem variação de freqüência (Cor). Apenas o coeficiente DC tem valor não zero, visto que a imagem somente possui freqüência zero.



2040	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
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

FDCT em Java

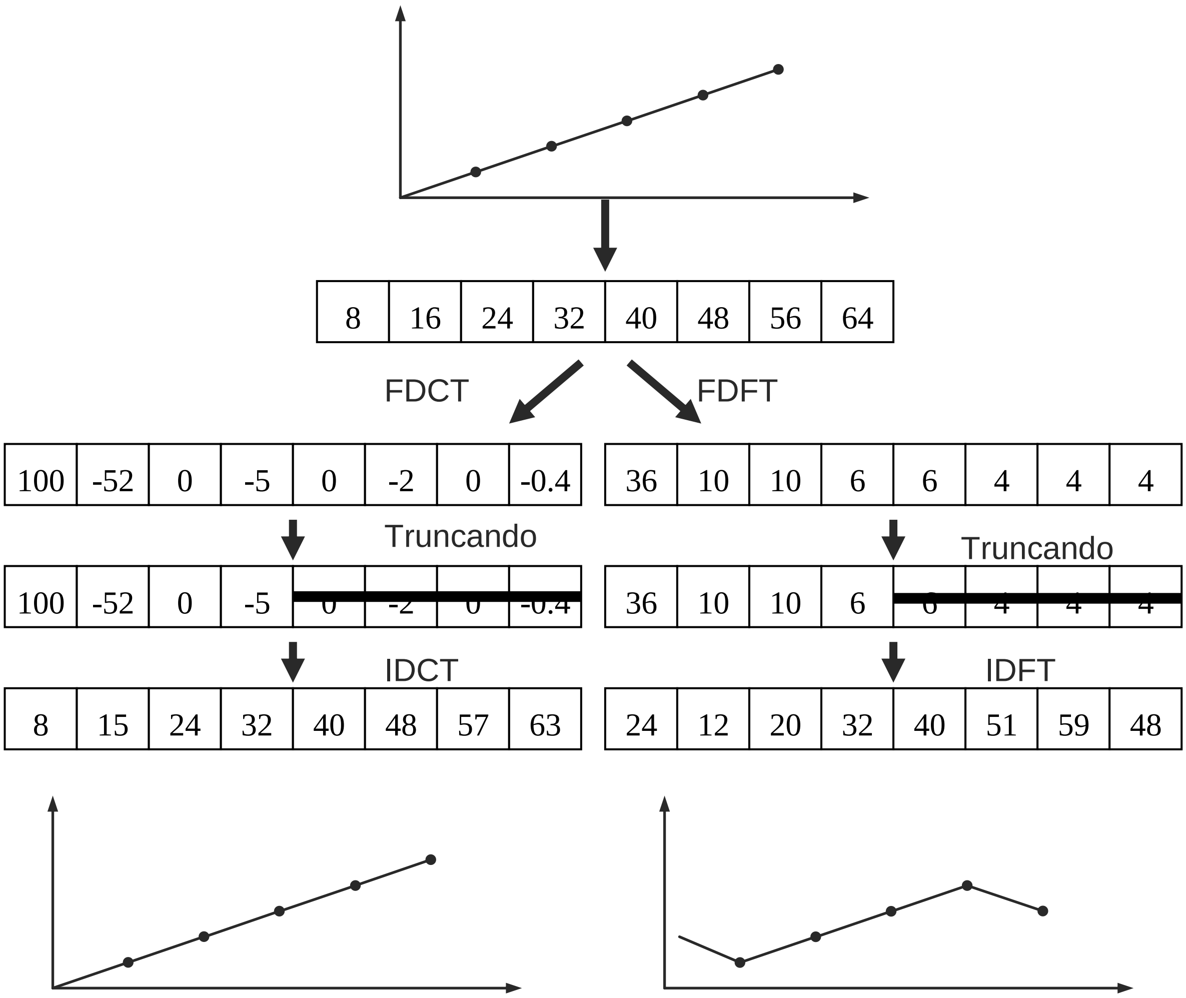
```
Matriz fdct(Imagem i) //forward discrete cosine transform
{
   int u, v, x, y;
   double pix, pi = Math.PI;
   for(u=0; u<8; u++)
   {
      for(v=0; v<8; v++)
      {
         pix = 0.0;
         for(x=0; x<8; x++)
         {
            for(y=0; y<8; y++)
            {
               pix += i.mat[x][y] *
                      cos(((2.0*x+1.0)*pi*u)/16.0) *
                      cos(((2.0*y+1.0)*pi*v)/16.0);
            }
         }
         m.mat[u][v] = (pix/4.0)*C(u)*C(v);
      }
   }
   return m;
}
```

IDCT em Java

```
Imagem idct(Matriz m) //inverse discrete cosine transform
{
   int u, v, x, y;
   double pix, pi = Math.PI;
   for(x=0; x<8; x++)
   {
      for(y=0; y<8; y++)
      {
         pix = 0.0;
         for(u=0; u<8; u++)
         {
            for(v=0; v<8; v++)
            {
               pix += C(u)*C(v)*m.mat[u][v] *
                      cos(((2.0*x+1)*pi*u)/16.0) *
                      cos(((2.0*y+1)*pi*v)/16.0);
            }
         }
         i.mat[x][y] = (int)Math.round(pix/4.0);
     }
   }
   return i;
}
```

DFT x DCT [10]

Neste exemplo faz-se um comparativo da aplicação da FDCT e da FDFT. Para obter compressão da imagem, deve-se remover freqüências altas. Pode-se observar que, para o mesmo conjunto de entrada, a FDCT concentra a informação em coeficientes maiores (baixas freqüências). Desta forma, coeficientes de alta frequência possuem baixa relevância na imagem e ao serem eliminados, não causam muito erro na reconstrução da imagem. A transformada de Fourier, para esta aplicação não se mostra tão eficiente, pois distribui de forma muito homogênea os coeficientes em todo espectro de freqüência.



Este exemplo pode ser utilizado para testar se a implementação da FDCT e a IDCT para o caso 1D com 8 amostras está correta.

Compressão JPEG - Quantização

- Obtém compressão pela representação dos coeficientes *F(u,v)* com menor precisão (N^o de bits).
- Consiste em dividir cada componente gerado pela DCT por um valor inteiro, definida em uma matriz de quantização (com valores entre 1 e 255). A escolha destes valores é difícil [1, 13].
- Objetivo de descartar informação com pouco valor. É a principal fonte de perda de informação.

$$F^{Q}(u,v) = ArredondamentoInteiro\left(\frac{F(u,v)}{Q(u,v)}\right)$$

$$F^{Q'} = F^{Q}(u,v) * Q(u,v) \text{ (processo inverso usado na descompressão da imagem)}$$

- Q[i, j] = 1 + (1 + i + j) * FatorQuantização
- FatorQuantização é considerado entre 2 e 25 e os índices iniciam no zero.
- Quanto maior for, maiores serão as perdas. Dessa maneira é feita uma quantização por zona, sendo os coeficientes associados às mais altas freqüências quantizados com mais severidade.

Matriz de quantização para fator 2

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

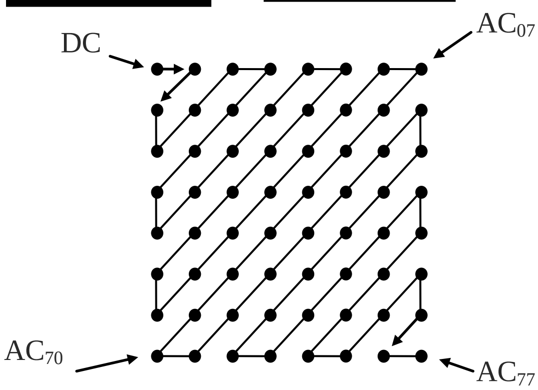
Wallance [1]

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Recomenda-se [13] como leitura adicional.

Compressão JPEG - Entropia

- Após a quantização, tem-se uma matriz de coeficientes de tamanho 8x8, onde o coeficiente (0,0) é chamado de DC, e os demais AC. O coeficiente DC mede a média de todos os 64 coeficientes, e por isso é o valor mais significativo da matriz [1].
- O zig-zag é usado para concentrar coeficientes de baixa freqüência antes dos de alta, facilitando assim a codificação de entropia.
- Métodos de entropia: *Huffman coding* e *Arithmetic coding*.

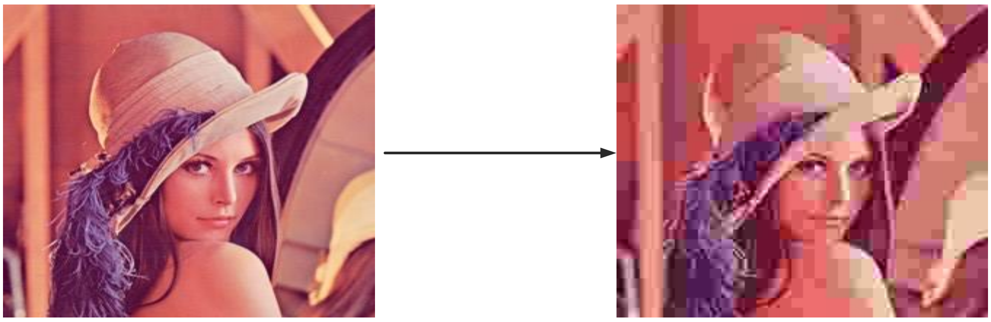


Compressão JPEG – Modo Progressivo

- Cada componente da imagem é codificado em múltiplas passadas.
- Permite em um ambiente de rede, por exemplo, que a imagem seja codificada de modo gradativo (múltiplos scans). No primeiro scan tem-se uma versão grosseira da imagem. Nos subseqüentes, versões mais detalhadas.
- Faz uso de duas abordagens:
  - *Spectral selection*: Codifica os coeficientes segundo a ordem do zig-zag (alta ou baixa freqüências)
  - *Sucessive aproximation*: codifica a cada passada, inicialmente os bits mais significativos até os menos significativos.

Deve-se observar que os dados salvos no arquivo JPEG representam frequências e não cores.
Quando o arquivo é carregado, todo o processo é executado na ordem inversa, e após a realização da IDCT, têm-se novamente cores.

Lena”. Exemplo de uma imagem normal e a mesma após processo de compressão muito elevado. [4]



Programa DCT [8]

- Ilustra o processo da transformada co-seno direta e inversa
- Mostra as funções básicas
- Mostra o valor dos pixels (componente R) do primeiro bloco 8x8 da imagem e seus respectivos coeficientes da DCT
- Mostra as componentes R, G, B e Y da imagem

Imagem Original

Diferença entre original e reconstruída

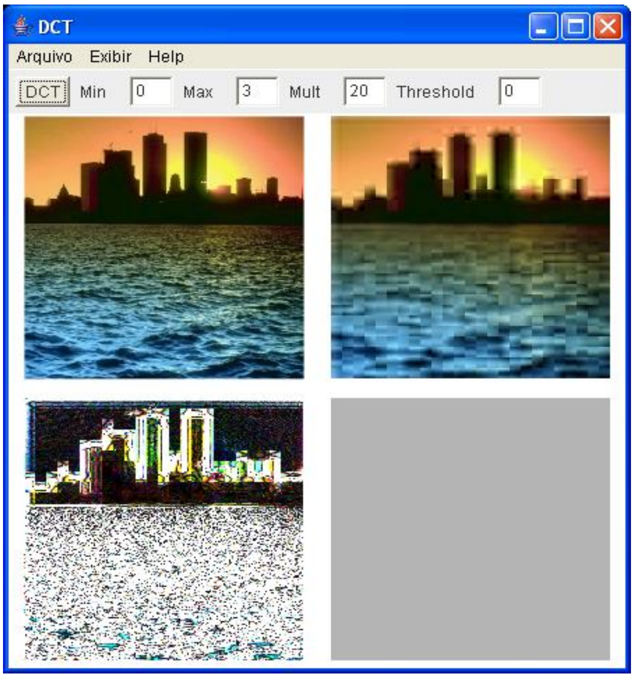
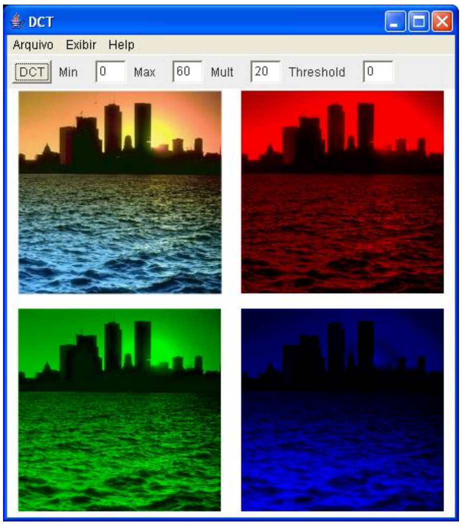
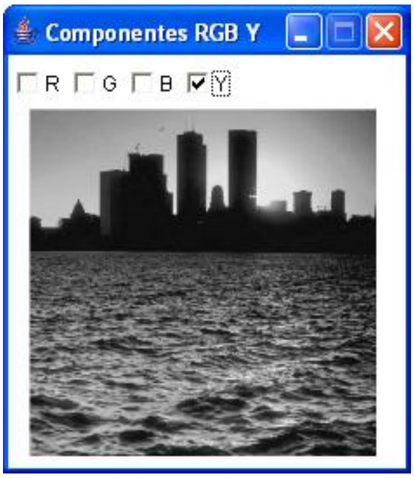


Imagem Reconstruída

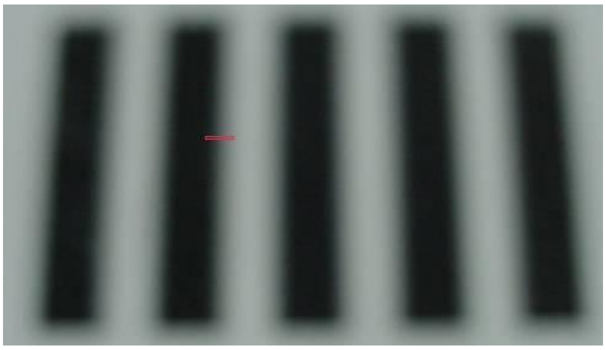
Taxa de compressão

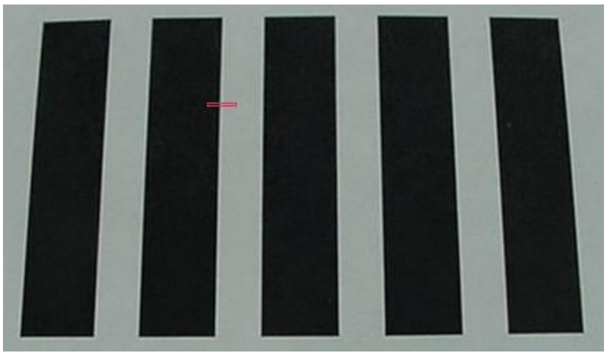




Aplicação de Análise de Freqüências [12]

> ***Passive** autofocus, commonly found on single-lens reflex (SLR) autofocus cameras, determines the distance to the subject by **computer analysis** of the image itself. The camera actually looks at the scene and drives the lens back and forth searching for the best focus. A typical autofocus sensor is a **charge-coupled device** (CCD) that provides input to algorithms that compute the contrast of the actual picture elements. The CCD is typically a single strip of 100 or 200 pixels. Light from the scene hits this strip and the microprocessor looks at the values from each pixel. The following images help you understand what the camera sees*. **Resumindo: a imagem está em foco quando existirem altas freqüências, como mostrado na segunda imagem.**





Tópicos adicionais

Funções ortogonais

Duas funções são **ortogonais** quando o seu produto interno (produto escalar) for zero, ou seja, formam um ângulo de 90 graus entre si [14]. Em termos matemáticos, dadas duas funções f(x) e g(x), temos que o produto entre elas tem que ser zero.

$$\int_{-\infty}^{\infty} f(x) \cdot g(x) = 0$$

Para funções trigonométricas, pode-se definir o intervalo de integração como múltiplo de π.

$$\int_0^{\pi} \sin(x) \cdot \cos(x) = 0$$

Isso pode ser verificado utilizando-se o seguinte algoritmo

```c
#include<stdio.h>
#include<math.h>

void main()
{
   double soma = 0;
   double ang = 0;
   for(ang=0; ang < M_PI; ang+=0.00001)
   {
       soma = soma + sin(ang)*cos(ang);
   }
   printf("%lf", soma);
}
```

Considerando-se a transformada DCT 1D com 8 amostras, temos a seguinte formulação:

$$F(u) = \frac{1}{2} C(u) \sum_{x=0}^{7} f(x) * \cos \frac{(2x+1)u\pi}{16}$$

Para mostrar que essas 8 funções são ortogonais, utiliza-se o seguinte algoritmo. Considerando-se quaisquer dois $u_1 \neq u_2$, temos que o produto interno das funções sempre dá como resultado valor zero.

```c
void ortogonal()
{
    double u, x, f1, f2;
    double soma = 0;
    for(x=0; x<8; x++)
    {
        u1 = 3;
        u2 = 2;
        f1 = cos(((2*x+1)*PI*u1)/16.0);
        f2 = cos(((2*x+1)*PI*u2)/16.0);
        soma += f1*f2;
    }
    printf("%lf", soma);
}
```

Exercícios

1. Faça um programa para carregar uma imagem colorida em formato BMP. Após, implemente funções para:
    a. Gerar o histograma de cada canal
    b. Gerar imagem com cada canal de cor separado
    c. Fazer rotação na imagem em 90 graus
    d. Fazer reescala da imagem
    e. Aumentar e diminuir o brilho da imagem.
    f. Aplicar a DCT e fazer a redução de altas frequências
    g. Salvar a nova imagem em disco.

Referências:

[1] Wallace, G. K., *The JPEG Still Picture Compression Standard*, IEEE Transactions on Consumer Electronics, 1991

[2] Explicando o significado de DPI. Disponível em: http://www.epinions.com/content_1883086980
[3] Resolução de Monitor. Disponível em: http://www.proaxis.com/~ferris/docs/dpi-monitor-bw.html
[4] Resolução de Monitor, impressora e scanner. Disponível em: http://www.bancodaimagem.com.br/curso/html/cap03-4.html
[5] The Lenna Story. Disponível em: http://www.cs.cmu.edu/~chuck/lennapg/
[6] Gomes, J., Velho, L. *Computação Gráfica*, *Volume 1*. IMPA, 1998.
[7] Soares, L. F. G. *Notas de aula*, *Curso de Fundamentos de Multimídia*, PUC-Rio.
[8] Pozzer, C. T. *Programa JPEG Compressor*. Disponível em: http://www-usr.inf.ufsm.br/~pozzer/, na seção Trabalhos.
[9] Rigotti, H. G. *Codificação de Imagem Usando Transformada Cosseno Discreta*. Dissertação de Mestrado, UFMS, 2004.
[10] Marshall, D. *DCT*. Disponível em: http://www.cs.cf.ac.uk/Dave/Multimedia/node231.html
[11] DCT – Disponível em http://en.wikipedia.org/wiki/Discrete_cosine_transform
[12] Auto foco. Disponível em: http://electronics.howstuffworks.com/autofocus3.htm
[13] Guetzli: Perceptually Guided JPEG Encoder
[14] Funções ortogonais. https://en.wikipedia.org/wiki/Orthogonal_functions