Discrete cosine transform

David Bařina

May 4, 2011

Motivation

- Fourier transform
- sparse image representation, compression, JPEG
- integral transform

$$c(k) = \int f(t)\, g_k(t)\, \mathrm{d}t$$

$\{g_k\}$ . . . kernel, basis

$$f(t) = \int c(k)\, g_k^{-1}(t)\, \mathrm{d}k$$

$\{g_k^{-1}\}$ . . . inverse kernel

1D DCT

- basis of the complex space $\mathbb{C}^N$ (signal length of N)

$$\left\{ g_k[n] = \lambda_k \sqrt{\frac{2}{N}} \cos\left[\frac{k\pi}{N} \left(n + \frac{1}{2} \right) \right] \right\}_{0 \leq k < N}$$

$$\lambda_k = \begin{cases} 1/\sqrt{2} & : k = 0 \\ 1 & : k \neq 0 \end{cases}$$

- forward transform $f \in \mathbb{C}^N$

$$c[k] = \sum_{n=0}^{N-1} f[n]\, g_k[n]$$

- inverse transform

$$f[n] = \sum_{k=0}^{N-1} c[k]\, g_k[n]$$

2D DCT

- basis of $\mathbb{C}^{N\times N}$ space (image block of $N \times N$ pixels)

$$\left\{ g_{k,j}[n,m] = \lambda_k \lambda_j \frac{2}{N} \cos\left[\frac{k\pi}{N}\left(n+\frac{1}{2}\right)\right] \cos\left[\frac{j\pi}{N}\left(m+\frac{1}{2}\right)\right] \right\}_{0\le k,j<N}$$

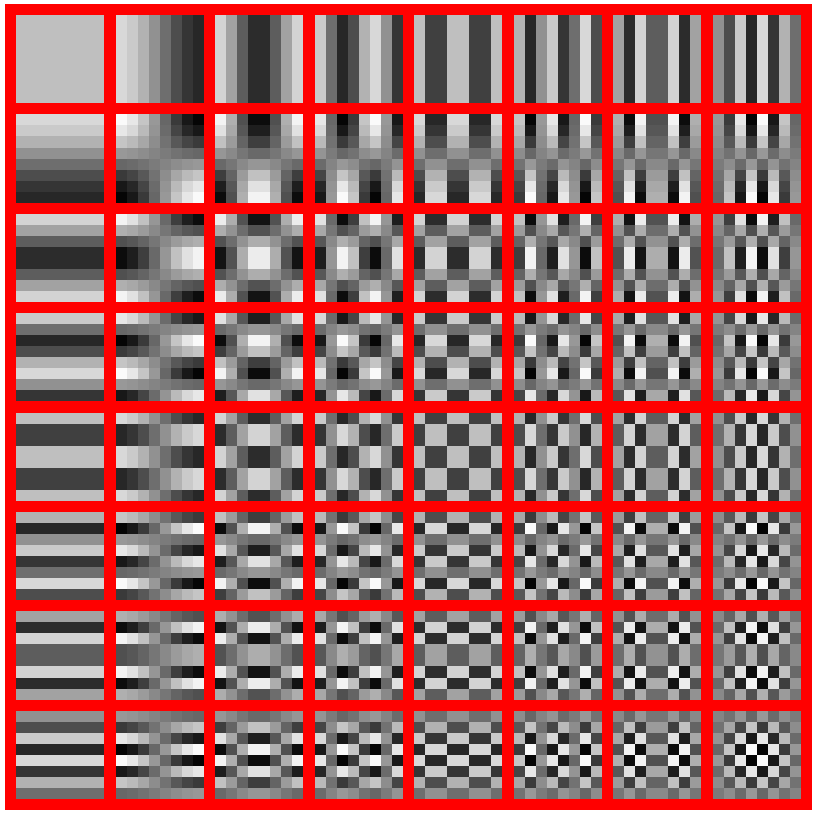
- forward transform

$$c[k,j] = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} f[n,m]\, g_{k,j}[n,m]$$

- inverse transform

$$f[n,m] = \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} c[k,j]\, g_{k,j}[n,m]$$

2D DCT basis



Separable 2D transform

- transform on $\mathbb{C}^{N\times N}$ space has $O(N^4)$ complexity
- identical separable transform has $O(N^3)$ complexity
- kernel of the transform can be separated

$$g_{k,j}[n,m] = g_k[n]\, g_j[m]$$

(simplified notation)

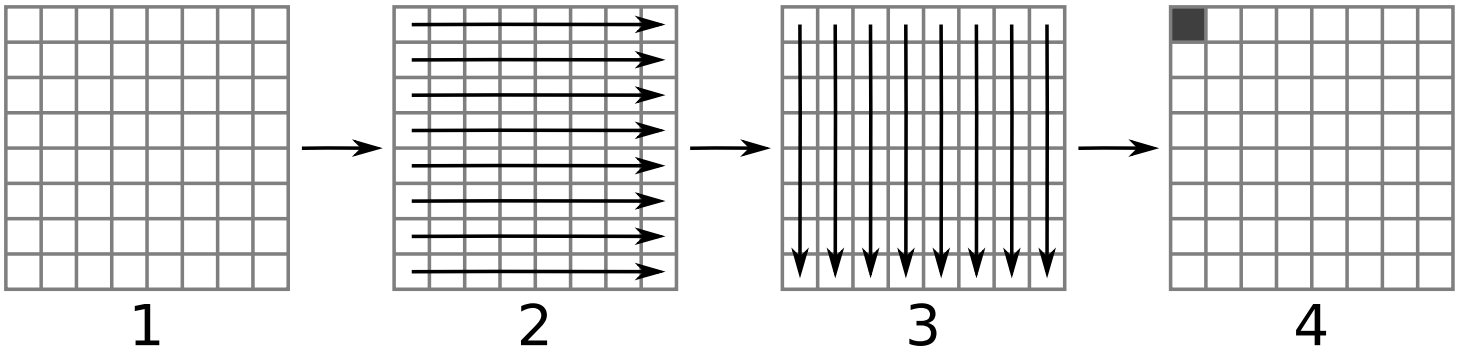
$$\{g_j[m]\} = \{g_k[n]\}^T$$

Separable 2D transform

- computation in two steps – the rows, columns

$$
\begin{aligned}
c[k,j] &= \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f[n,m]\, g_{k,j}[n,m] \\
&= \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f[n,m]\, g_k[n]\, g_j[m] \\
&= \sum_{n=0}^{N-1} g_k[n] \sum_{m=0}^{N-1} f[n,m]\, g_j[m] = \sum_{n=0}^{N-1} g_k[n]\, c_j[n] \\
&= \sum_{m=0}^{N-1} g_j[m] \sum_{n=0}^{N-1} f[n,m]\, g_k[n] = \sum_{m=0}^{N-1} g_j[m]\, c_k[m]
\end{aligned}
$$

Separable 2D transform



Coefficient quantization

coarser quantization $\Rightarrow$ larger loss of information $\Rightarrow$ better compression

$$Q[k,j] = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$b[k,j] = \text{round}\left(\frac{c[k,j]}{Q[k,j]}\right) \qquad \tilde{c}[k,j] = b[k,j]\,Q[k,j]$$

OpenCV

- documentation – http://opencv.willowgarage.com/

```
Mat img = Mat_<double>(
imread("Lenna.png", CV_LOAD_IMAGE_GRAYSCALE) );

img.at<double>(y,x)

Mat dst = src.t();

Rect rect (x, y, N, N);
Mat block(img, rect);

imshow("image", Mat_<uchar>(img) );
```

Tasks

1. play with two DCT implementation
2. play with coefficient quantization