Discrete Cosine Transform

What is Transform Coding?

- Transform coding constitutes an integral component of contemporary image/video processing applications.
- Transform coding relies on the premise that pixels in an image exhibit a certain level of correlation with their neighboring pixels
- Similarly in a video transmission system, adjacent pixels in consecutive frames show very high correlation. Consequently, these correlations can be exploited to predict the value of a pixel from its respective neighbors.
- Transformation is a lossless operation, therefore, the inverse transformation renders a perfect reconstruction of the original image.

Why Transform Coding?

- Better image processing
  - Take into account **long-range correlations** in space
  - Conceptual insights in spatial-frequency information. → what it means to be "smooth, moderate change, fast change, …"
- Fast computation
- Alternative representation and sensing
  - **Efficient storage and transmission**
  - **Energy compaction**
  - Pick a **few "representatives" (basis)**
  - Just store/send the **"contribution" from each basis**

The Discrete Cosine Transform

- Discrete Cosine Transform (DCT) has emerged as the image transformation in most visual systems. DCT has been widely deployed by modern video coding standards, for example, **MPEG, JVT etc**.
- It is the same family as the Fourier Transform
  - **Converts data to frequency domain**
- Represents data **via <u>summation of variable frequency cosine waves.</u>**
- Captures only **real components** of the function.
  - Discrete Sine Transform (DST) captures odd (imaginary) components →not as useful.
  - Discrete Fourier Transform (DFT) captures both odd and even components →computationally intense.

Discrete Fourier Transform

- due to its computational efficiency the DFT is very popular
- however, it has strong disadvantages for some applications
  - it is complex
  - it has poor energy compaction

- What is energy compaction?
  - is the ability to pack the energy of the spatial sequence into as few frequency coefficients as possible
  - this is very important for image compression
  - if compaction is high, we only have to transmit a few coefficients instead of the whole set of pixels

Comparision to DFT:

- As compared to DFT, application of DCT results in **less blocking artifacts** due to the even symmetric extension properties of DCT.
- **DCT uses real computations**, unlike the complex computations used in DFT.
- DFT is a complex transform and therefore stipulates that both image magnitude and phase information be encoded.
- DCT **hardware simpler,** as compared to that of DFT.
- DCT provides **better energy compaction** than DFT for most natural images.
- The implicit periodicity of DFT gives rise to **boundary discontinuities** that result in significant high-frequency content. After quantization, *Gibbs Phenomenon* causes the boundary points to take on erroneous values .

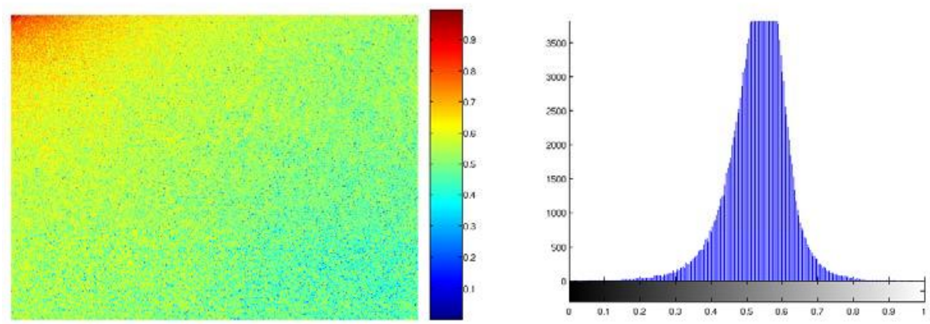
Discrete Cosine Transform

- in this example we see the amplitude spectra of the image above

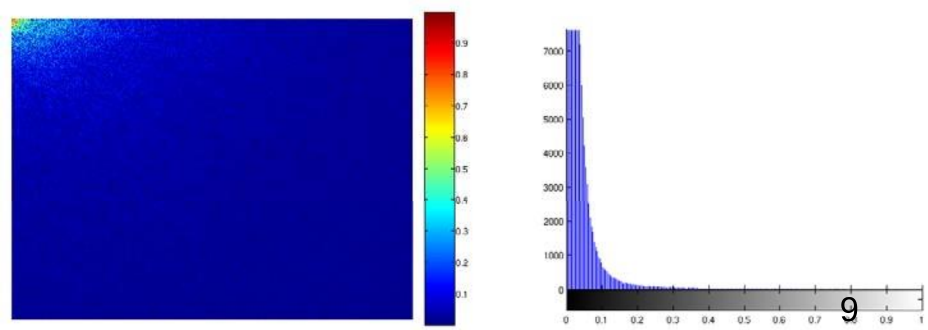
  under the DFT and DCT

- note the much more concentrated histogram obtained with the DCT

DFT



DCT



- why is energy compaction important?
  - the main reason is image compression
  - turns out to be beneficial in other applications

Mathematical Basis

- 1D DCT:

$$C(u) = \alpha(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{\pi(2x+1)u}{2N}\right]$$

Where:

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0. \end{cases}$$

1D DCT is ***O(n²)***

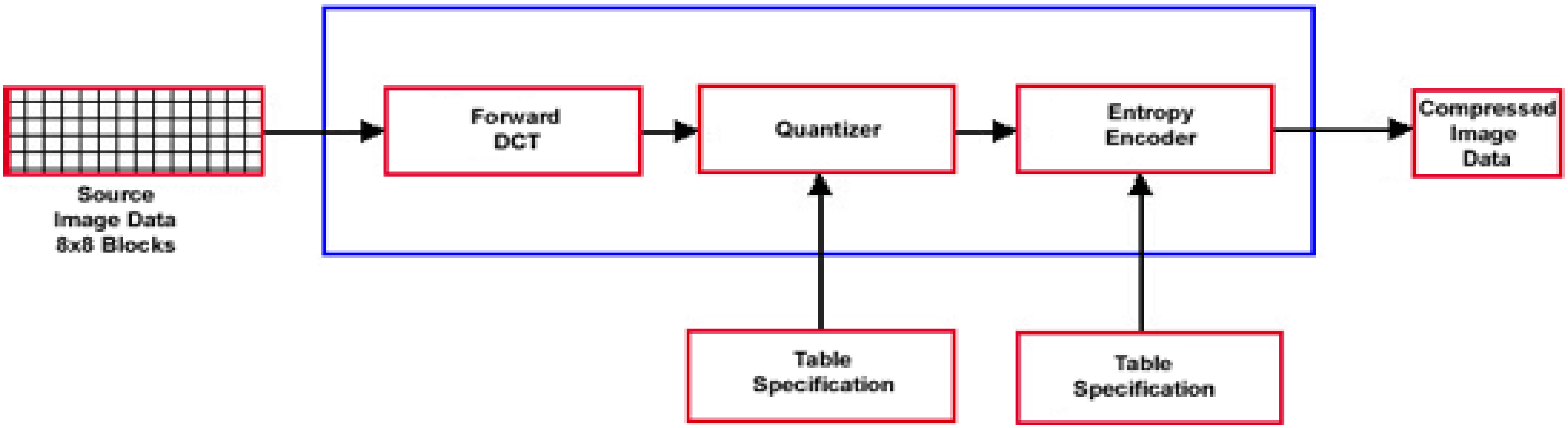
- 2D DCT:

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

- Where α(u) and α(v) are defined as shown in the 1D case.
- 2D DCT is *O(n³)*

Image compression

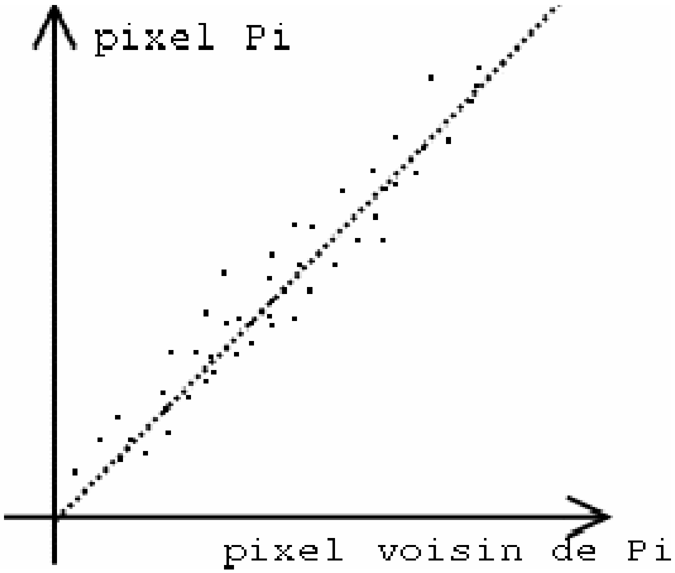
- an image compression system has three main blocks



  - a transform (usually DCT on 8x8 blocks)
  - a quantizer
  - a lossless (entropy) coder
- each tries to throw away information which is not essential to understand the image, but consume bits

Image compression

- the transform throws away correlations
  - if you make a plot of the value of a pixel as a function of one of its neighbors



  - you will see that the pixels are highly correlated (i.e. most of the time they are very similar)
  - this is just a consequence of the fact that surfaces are smooth

Image compression

- the transform eliminates these correlations
  - this is best seen by considering the 2-pt transform
  - note that the first coefficient is always the DC-value

$$\boxed{X[0] = x[0] + x[1]}$$

  - an orthogonal transform can be written in matrix form as

$$\boxed{X = Tx, \qquad T^T T = I}$$

  - i.e. T has orthogonal columns
  - this means that

$$\boxed{X[1] = x[0] - x[1]}$$

  - note that if x[0] similar to x[1], then

$$\boxed{\begin{cases} X[0] = x[0] + x[1] \approx 2x[0] \\ X[1] = x[0] - x[1] \approx 0 \end{cases}}$$

Image compression

- in the transform domain we only have to transmit one number without any significant cost in image quality
- by "decorrelating” the signal we reduced the bit rate to ½

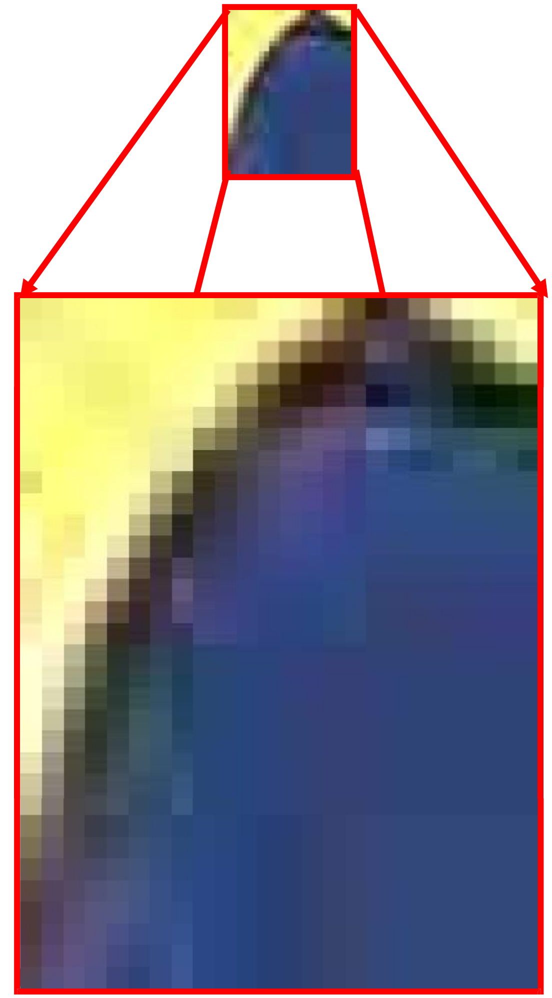
  - note that an orthogonal matrix

    $$T^T T = I$$

    applies a rotation to the pixel space
  - this aligns the data with the canonical axes

Image compression

- a second advantage of working in the frequency domain
  - is that our visual system is less sensitive to distortion around edges
  - the transition associated with the edge masks our ability to perceive the noise
  - e.g. if you blow up a compressed picture, it is likely to look like this
  - in general, the compression errors are more annoying in the smooth image regions



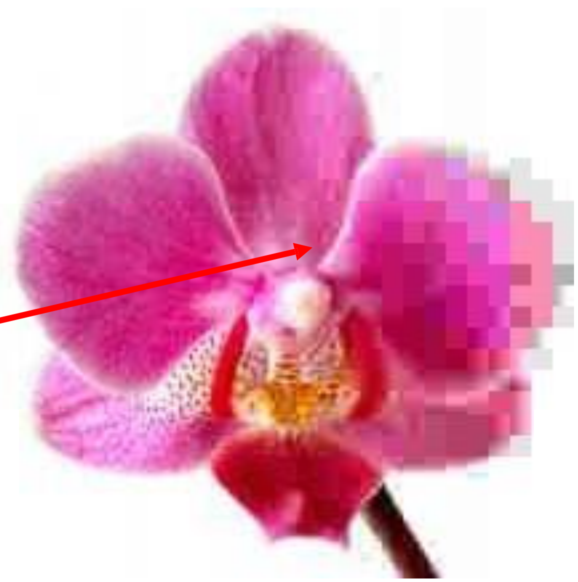


Image compression

- three JPEG examples

36KB 5.7KB 1.7KB

  - note that the blockiness is more visible more to compression

Image compression

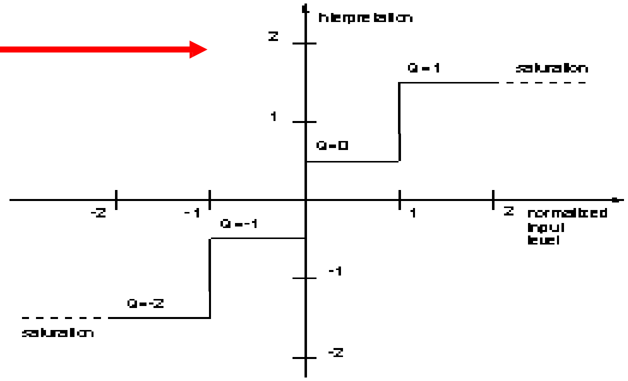
- the transform
  - does not save any bits
  - does not introduce any distortion
- But is possible when we throw away information
- this is called "lossy compression" and implemented by the quantizer

Image compression

- what is a quantizer?
  - think of the round() function, that rounds to the nearest integer
  - round(1) = 1; round(0.55543) = 1; round (0.0000005) = 0
  - instead of an infinite range between 0 and 1 (infinite number of bits to transmit)
  - the output is zero or one (1 bit)
  - we threw away all the stuff in between, but saved a lot of bits
  - a quantizer does this less drastically

Quantizer

- it is a function of this type
  - inputs in a given range are mapped to the same output



- to implement this, we
  - 1) define a quantizer step size Q
  - 2) apply a rounding function

$$x_q = \text{round}\,(x/Q)$$

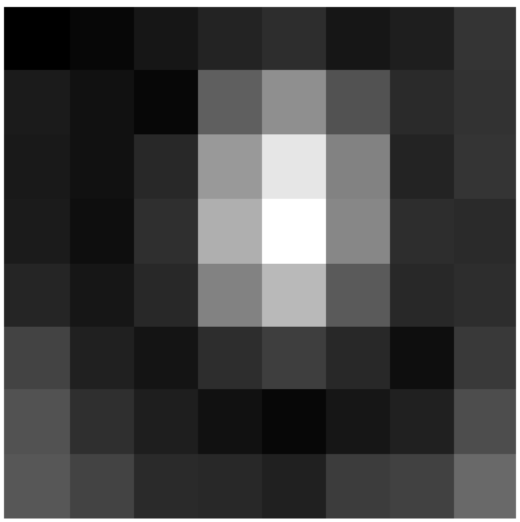
  - the larger the Q, the less reconstruction levels
  - more compression at the cost of larger distortion
  - e.g. for x in [0,255], we need 8 bits and have 256 color values
  - with Q = 64, we only have 4 levels and only need 2 bits
  - with Q = 32; how many bits required?

Quantizer

- e.g. for x in [0,255], we need 8 bits and have 256 color values
- with Q = 64, we only have 4 levels and only need 2 bits
- with Q = 32, we only have 8 levels and only need 3 bits hence compression ratio is 8/3= (2.67):1
-

Quantizer

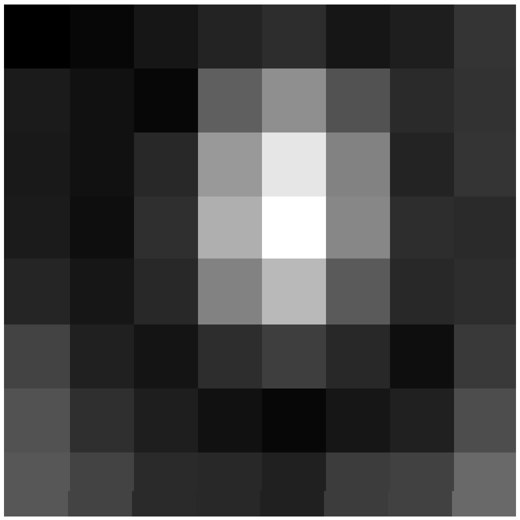
- note that we can quantize some frequency coefficients more heavily than others by simply increasing Q
- this leads to the idea of a quantization matrix
- we start with an image block (e.g. 8x8 pixels)



$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

Quantizer

- next we apply a transform (e.g. 8x8 DCT)



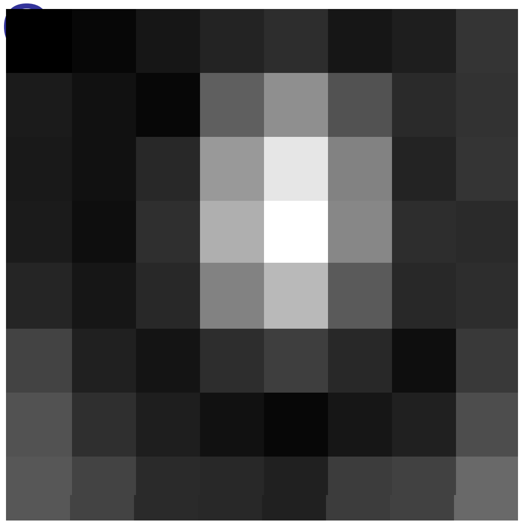
$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

DCT

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

Quantizer

- and quantize with a varying



$$
\begin{bmatrix}
52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\
63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\
62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\
63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\
67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\
79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\
85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\
87 & 79 & 69 & 68 & 65 & 76 & 78 & 94
\end{bmatrix}
$$

DCT

$$
\begin{bmatrix}
-415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\
4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\
-47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\
-49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\
12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\
-8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\
-1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\
0 & 0 & -1 & -4 & -1 & 0 & 1 & 2
\end{bmatrix}
$$

Q mtx

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

Quantizer

- note that higher frequencies are quantized more heavily

Q mtx

increasing frequency

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 1\text{[illegible]} & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

- in result, many high frequency coefficients are simply wiped out

DCT

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \longrightarrow$$

quantized DCT

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Quantizer

- this saves a lot of bits, but we no longer have an exact replica of original image block

DCT

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

quantized DCT

$$\begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

inverse DCT

$$\begin{bmatrix} 60 & 63 & 55 & 58 & 70 & 61 & 58 & 80 \\ 58 & 56 & 56 & 83 & 108 & 88 & 63 & 71 \\ 60 & 52 & 62 & 113 & 150 & 116 & 70 & 67 \\ 66 & 56 & 68 & 122 & 156 & 116 & 69 & 72 \\ 69 & 62 & 65 & 100 & 120 & 86 & 59 & 76 \\ 68 & 68 & 61 & 68 & 78 & 60 & 53 & 78 \\ 74 & 82 & 67 & 54 & 63 & 64 & 65 & 83 \\ 83 & 96 & 77 & 56 & 70 & 83 & 83 & 89 \end{bmatrix}$$

$\neq$

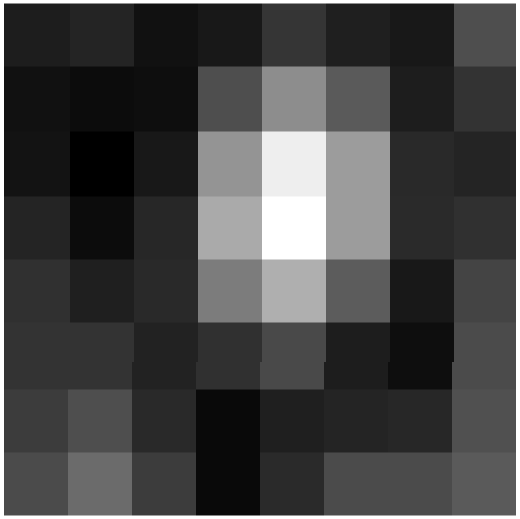
original pixels

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

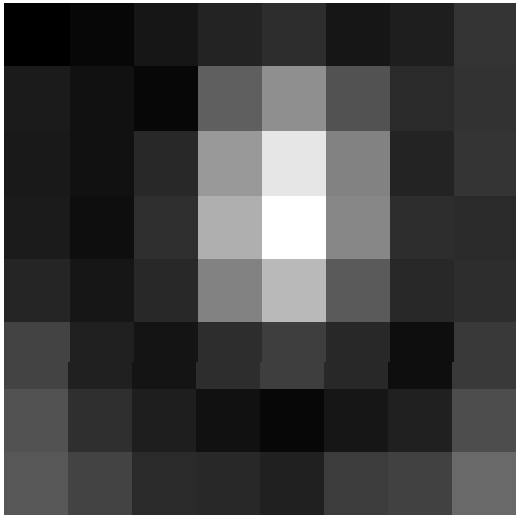
Quantizer

- note, however, that visually the blocks are not very different

original



decompressed



– we have saved lots of bits without much “perceptual” loss

– this is the reason why JPEG and MPEG work

Image compression

- three JPEG examples

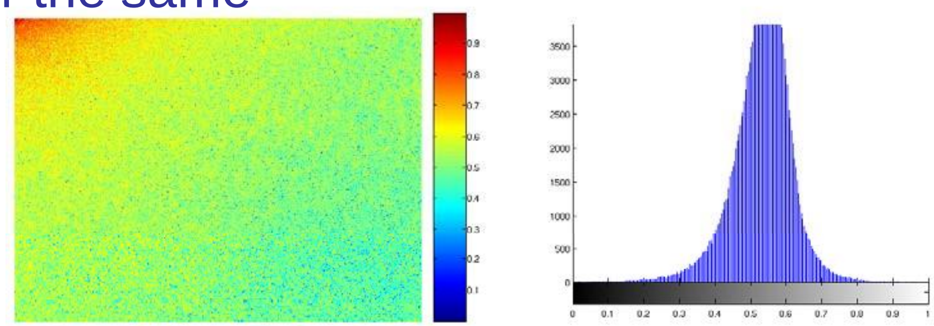
36KB 5.7KB 1.7KB

  - note that the two images on the left look identical
  - JPEG requires 6x less bits

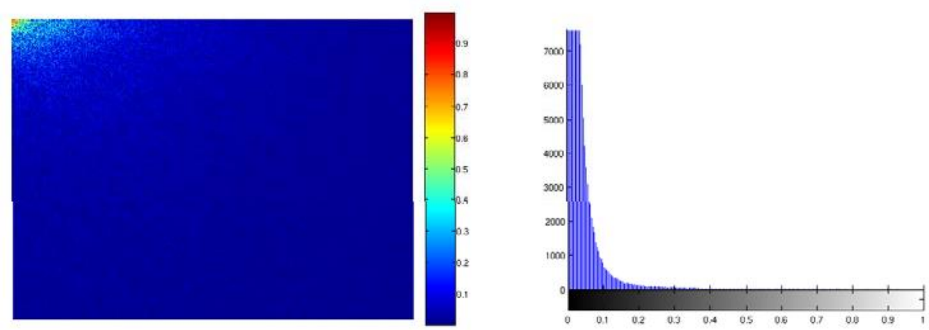
Discrete Cosine Transform

- note that
  - the better the energy compaction
  - the larger the number of coefficients that get wiped out
  - the greater the bit savings for the same loss
- this is why the DCT is important
- we will do mostly the 1D-DCT
  - the formulae are simpler the insights the same
  - as always, extension to 2D is trivial

DFT



DCT



Discrete Cosine Transform (Formal)

- the first thing to note is that there are various versions of the DCT
  - these are usually known as DCT-I to DCT-IV
  - they vary in minor details
  - the most popular is the DCT-II, also known as even symmetric DCT, or as "the DCT"

$$w[k] = \begin{cases} \frac{1}{2}, & k = 0 \\ 1, & 1 \le k < N \end{cases}$$

$$C_x[k] = \begin{cases} \sum_{n=0}^{N-1} 2x[n] \cos\left(\frac{\pi}{2N} k(2n+1) \right), & 0 \le k < N \\ 0 & otherwise \end{cases}$$

$$x[n] = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} w[k] C_x[k] \cos\left(\frac{\pi}{2N} k(2n+1) \right) & 0 \le n < N \\ 0 & otherwise \end{cases}$$

Energy compaction

$$\underbrace{x[n]}_{N-pt} \leftrightarrow \underbrace{y[n]}_{2N-pt} \overset{DFT}{\leftrightarrow} \underbrace{Y[k]}_{2N-pt} \leftrightarrow \underbrace{C_x[k]}_{N-pt}$$

- To understand the energy compaction property
  - we start by considering the sequence y[n] = x[n]+x[2N-1-n]
  - this just consists of adding a mirrored version of x[n] to itself

x[n]

y[n]

– next we remember that the DFT is identical to the DFS of the periodic extension of the sequence

– let's look at the periodic extensions for the two cases

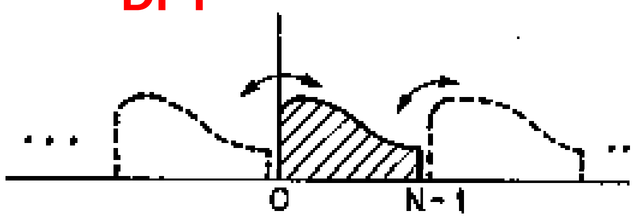
- when transform is DFT: we work with extension of x[n]
- when transform is DCT: we work with extension of y[n]

Energy compaction

$$\underbrace{x[n]}_{N-pt} \leftrightarrow \underbrace{y[n]}_{2N-pt} \overset{DFT}{\leftrightarrow} \underbrace{Y[k]}_{2N-pt} \leftrightarrow \underbrace{C_x[k]}_{N-pt}$$

- the two extensions are

DFT



DCT

- note that in the DFT case the extension introduces discontinuities
- this does not happen for the DCT, due to the symmetry of y[n]
- the elimination of this artificial discontinuity, which contains a lot of high frequencies,
- is the reason why the DCT is much more efficient

Fast algorithms

- The interpretation of the DCT as

$$\boxed{\underbrace{x[n]}_{N-pt} \leftrightarrow \underbrace{y[n]}_{2N-pt} \quad \overset{DFT}{\leftrightarrow} \quad \underbrace{Y[k]}_{2N-pt} \leftrightarrow \underbrace{C_x[k]}_{N-pt}}$$

  - also gives us a fast algorithm for its computation
  - it consists exactly of the three steps
  - 1) y[n] = x[n]+x[2N-1-n]
  - 2) Y[k] = DFT{y[n]}
    this can be computed with a 2N-pt FFT
  - 3)

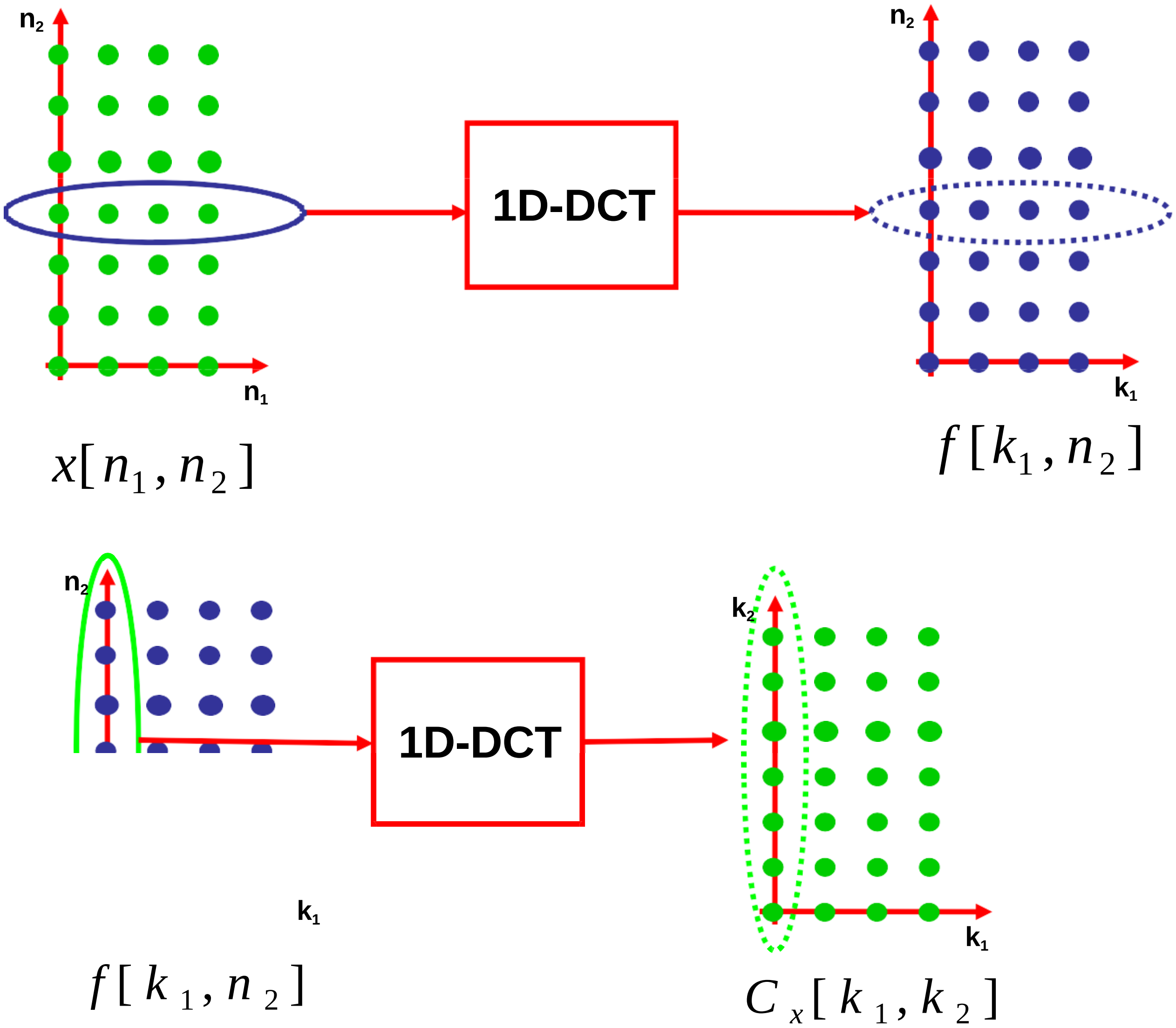
$$\boxed{C_x[k] = \begin{cases} e^{-j\frac{\pi}{2N}k} Y[k], & 0 \leq k < N \\ 0, & otherwise \end{cases}}$$

- **the complexity of the N-pt DCT is that of the 2N-pt DFT**

2D-DCT

- 1) create intermediate sequence by computing 1D-DCT of rows
- 2) compute

  1D-DCT of columns



Properties of DCT:

- De correlation
- Energy Compaction
- Separability
- Symmetry
- Orthogonality

De correlation:

- The principle advantage of image transformation is the removal of redundancy between neighboring pixels. **This leads to uncorrelated transform coefficients which can be encoded independently.**
- **The amplitude of the autocorrelation after the DCT operation is very small at all lags.** Hence, it can be inferred that **DCT exhibits excellent de correlation properties.**

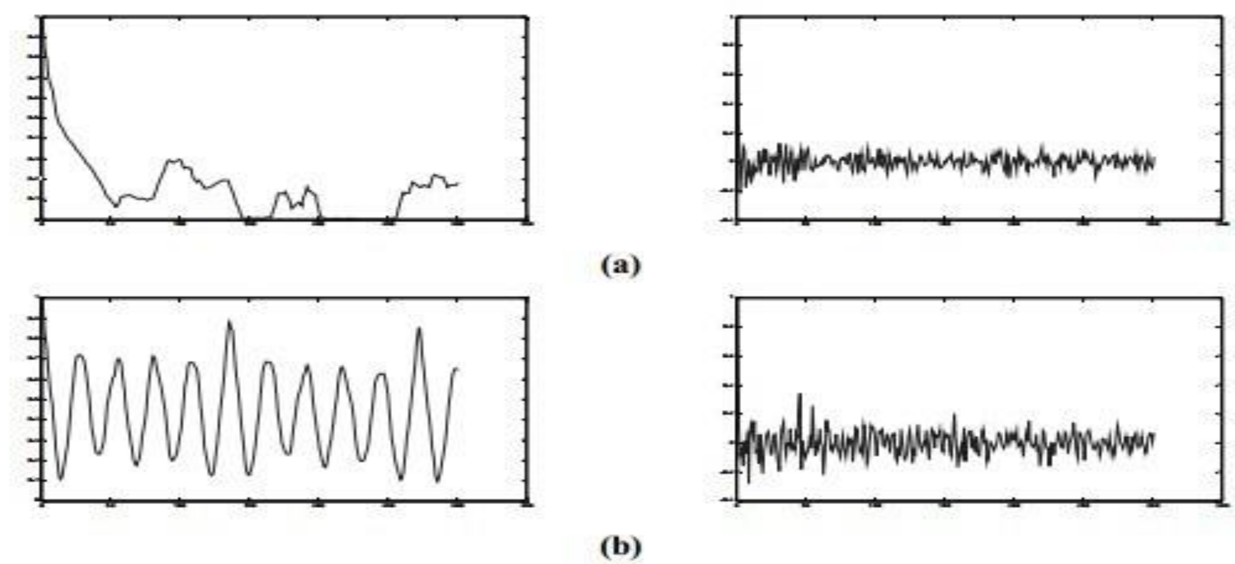
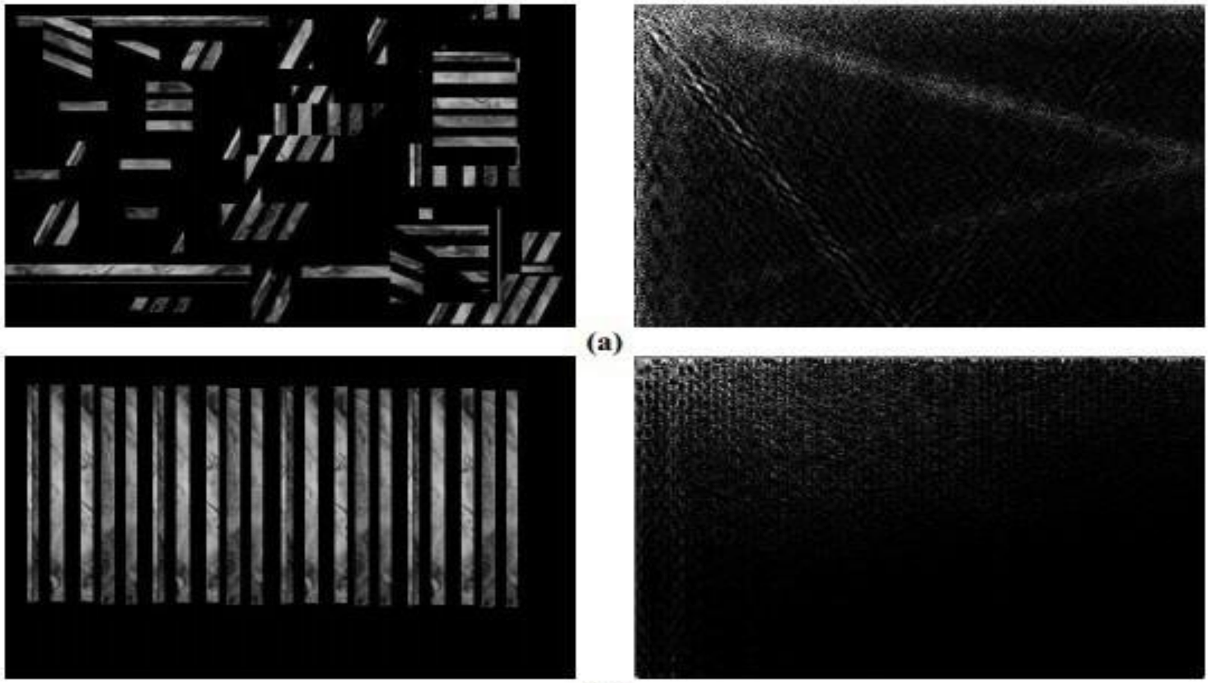


Figure 5. (a) Normalized autocorrelation of uncorrelated image before and after DCT; (b) Normalized autocorrelation of correlated image before and after DCT.

Energy Compaction:

- Efficiency of a transformation scheme can be directly gauged by its **ability to pack input data into as few coefficients** as possible. This **allows the quantizer to discard coefficients with relatively small amplitudes without introducing visual distortion** in the reconstructed image. **DCT exhibits excellent energy compaction for highly correlated images**.



(a)

(b)

Figure 6. (a) Uncorrelated image and its DCT; (b) Correlated image and its DCT.

Separability:

- The principle advantage that $C(u, v)$ can be computed in **two steps by successive 1-D operations** on rows and columns of an image.

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

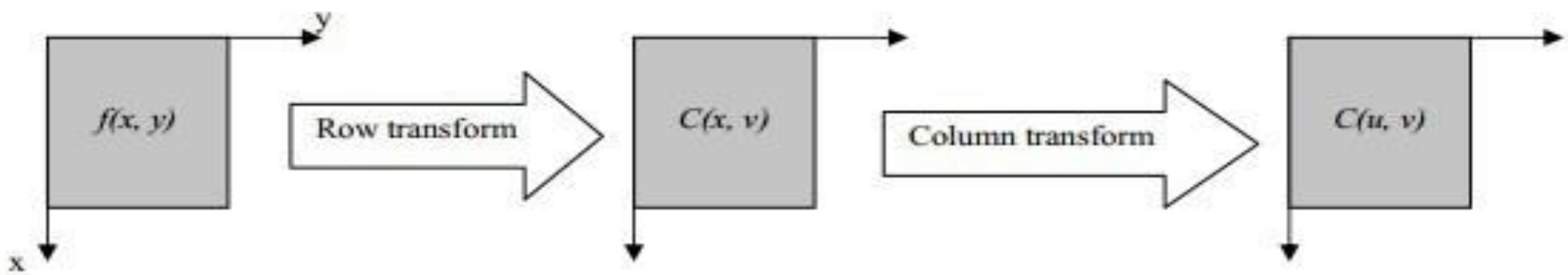


Figure 8. Computation of 2-D DCT using separability property.

12

Symmetry:

- A separable and symmetric transform can be expressed in the form

$$T = AfA \,,$$

where A is an $N \times N$ symmetric transformation matrix with entries $a(i\,,\,j)$ given by,

$$a(i,j) = \alpha(j) \sum_{j=0}^{N-1} \cos\left[\frac{\pi(2j+1)i}{2N}\right],$$

and f is the N x N image matrix.

- This is an extremely useful property since it implies that **the transformation matrix can be pre computed offline** and then applied to the image thereby providing orders of magnitude improvement in computation efficiency.

Orthogonality:

- The inverse transformation as

$$f = A^{-1}\ T\ A^{-1}\ .$$

DCT basis functions are **orthogonal**. The **inverse transformation matrix of *A* is equal to its transpose i.e.**

$$A^{-1} = A^{T}.$$

Therefore, and in addition to it de correlation characteristics, this property **renders some reduction in the pre-computation complexity**

Significance/Where is this DCT used?

- Image Processing
  - Compression - Ex.) JPEG
  - Scientific Analysis - Ex.) Radio Telescope Data
- Audio Processing
  - Compression - Ex.) MPEG – Layer 3, aka. MP3
- Scientific Computing / High Performance Computing (HPC)
  - Partial Differential Equation Solvers

Limitation of DCT:

- **Truncation of higher spectral coefficients results in blurring** of the images, especially wherever the details are high.

- **Coarse quantization of some of the low spectral coefficients introduces graininess** in the smooth portions of the images.

- **Serious blocking artifacts are introduced at the boundaries, since each block is independently encoded**, often with a different encoding strategy and the extent of quantization.