# Advanced Transform Methods

## Discrete Cosine Transform (DCT)

Andy Watson

# The Discrete Cosine Transform (DCT)

- Reversible (lossless) transform that represents a discrete signal as a set of cosine coefficients.
- Similar to DFT, but uses only cosines and therefore avoids any complex numbers.
- Real Input, real output
- Consider only discrete with number of input= number of output (n=k).

# 1-dimensional DCT

$$DCT[k] = c(k)\sum_{n=0}^{N-1} s[n]\cos\frac{\pi(2n+1)k}{2N} \qquad c(k) = \begin{cases} \sqrt{1/N} & k=0 \\ \sqrt{2/N} & k \neq 0 \end{cases}$$

$$k = 0,1,2\ldots N-1$$

- Orthonormal $DCT[k] = \langle s, \psi_k \rangle$ $\quad \langle \psi_m, \psi_n \rangle = \begin{cases} 1 & \text{if } m=n \\ 0 & \text{if } m \neq n \end{cases}$
- Example: $N$=4

$$\psi_0 = (1,1,1,1)/2$$

$$\psi_1 = \sqrt{1/2}(\cos(\pi/8), \cos(3\pi/8), \cos(5\pi/8), \cos(7\pi/8))$$

$$\psi_2 = \sqrt{1/2}(\cos(\pi/4), \cos(3\pi/4), \cos(5\pi/4), \cos(7\pi/4))$$

$$\psi_3 = \sqrt{1/2}(\cos(3\pi/8), \cos(9\pi/8), \cos(15\pi/8), \cos(5\pi/8))$$

$$DCT[0] = \frac{1}{\sqrt{N}}\sum_{n=0}^{3} s[n] \qquad DCT[1] = \sqrt{\frac{2}{N}}\sum_{n=0}^{3} s[n]\cos\frac{\pi(2n+1)}{8}$$

$$DCT[2] = \sqrt{\frac{2}{N}}\sum_{n=0}^{3} s[n]\cos\frac{\pi(2n+1)}{4} \qquad DCT[3] = \sqrt{\frac{2}{N}}\sum_{n=0}^{3} s[n]\cos\frac{\pi(2n+1)3}{8}$$

# DCT Basis Functions

$s(0)$
$s(1)$
$s(2)$
$s(3)$
$s(4)$
$s(5)$
$s(6)$
$s(7)$

$$\frac{1}{2\sqrt{2}}\sum_{x=0}^{7} s(x)\cos(\frac{2x+1}{16}*0\pi)$$

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

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

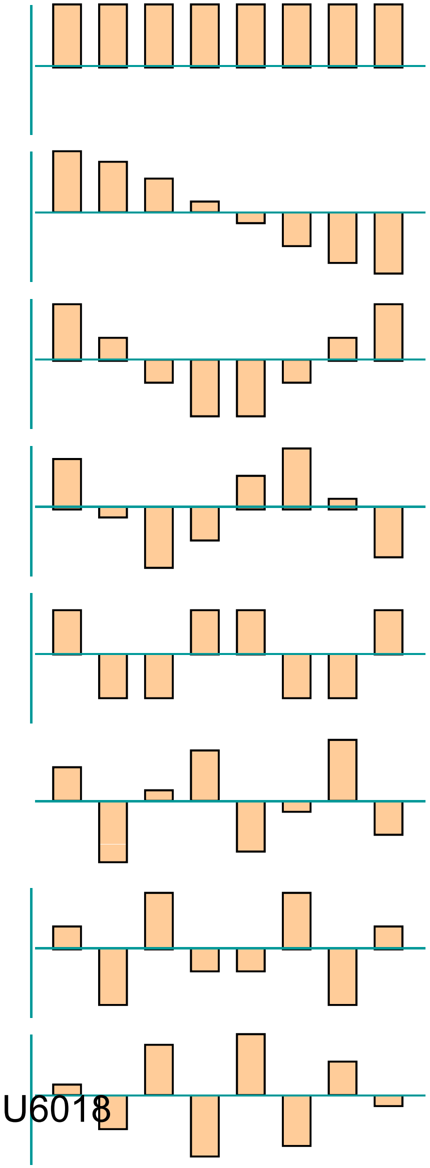
$$\frac{1}{2}\sum_{x=0}^{7} s(x)\cos(\frac{2x+1}{16}*3\pi)$$

$$\frac{1}{2}\sum_{x=0}^{7} s(x)\cos(\frac{2x+1}{16}*4\pi)$$

$$\frac{1}{2}\sum_{x=0}^{7} s(x)\cos(\frac{2x+1}{16}*5\pi)$$

$$\frac{1}{2}\sum_{x=0}^{7} s(x)\cos(\frac{2x+1}{16}*6\pi)$$

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



$S(0)$ DC

$$\left.\begin{matrix} S(1) \\ S(2) \\ S(3) \\ S(4) \\ S(5) \\ S(6) \\ S(7) \end{matrix}\right\} \text{AC}$$

# Inverse DCT

- DCT $$DCT[k] = c(k)\sum_{n=0}^{N-1} s[n]\cos\frac{\pi(2n+1)k}{2N}$$

- Inverse DCT $$s[n] = \sum_{k=0}^{N-1} c(k)DCT[k]\cos\frac{\pi(2n+1)k}{2N}$$

*Compare with*

- DFT $$\widetilde{S}[k] = \sum_{n=0}^{N-1} \widetilde{s}[n]e^{-j2\pi nk/N} \qquad k = 0,1,2,\ldots, N-1$$

- Inverse DFT $$\widetilde{s}[n] = \frac{1}{N}\sum_{k=0}^{N-1} \widetilde{S}[k]e^{j2\pi nk/N} \qquad n = 0,1,2,\ldots, N-1$$

# 2D & nD Transforms

- Mostly seen transforms applied in 1 dimension.
- Many can also be applied in 2D or *n*D. E.g. Fourier:

1-D: $S(\omega) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt$

2-D: $S(w, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(x, y) e^{-j(wx+vy)} dxdy$

$$= \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} s(x, y) e^{-jwx} dx \right) e^{-jvy} dy$$

$n$-D: $S(\mathbf{w}) = \int_{-\infty}^{\infty} \quad \int_{-\infty}^{\infty} s(\mathbf{x}) e^{-j(\mathbf{w} \cdot \mathbf{x})} d\mathbf{x}$

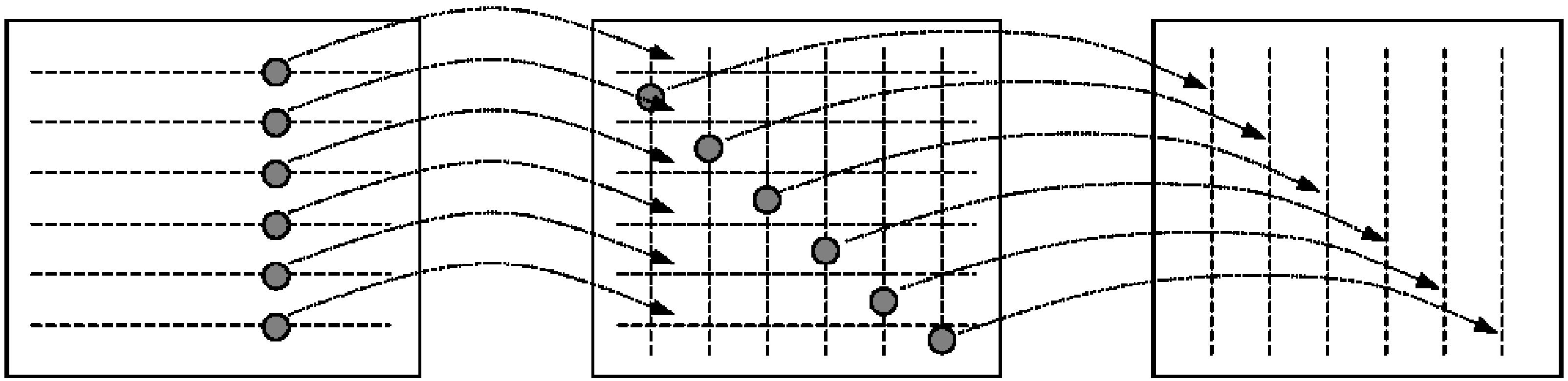
Dot Product

where $\mathbf{w} = (w_1, w_2, \quad , w_n)$ and $\mathbf{x} = (x_1, x_2, \quad , x_n)$.

An $n$-D transform is *separable* if we can apply sequence of 1-D transforms (see 2-D case above).

# Separable Transforms

May be implemented by applying the one dimensional transform first to the rows of the image and then to its columns (note that changing the application order does not change the result).



$x_{p_1,p_2}$ $y'_{p_1,q_2}$ $y_{q_1,q_2}$

# 2-dimensional DCT

- Defined as:

$$DCT_{2d}[i, j] = c(i, j) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} s[m, n] \cos \frac{\pi(2m+1)i}{2N} \cos \frac{\pi(2n+1)j}{2N}$$

$$c(i, j) = \begin{cases} 1/N & i = 0, j = 0 \\ \sqrt{2}/N & i \neq 0, j = 0 \\ \sqrt{2}/N & i = 0, j \neq 0 \\ 2/N & i \neq 0, j \neq 0 \end{cases}$$

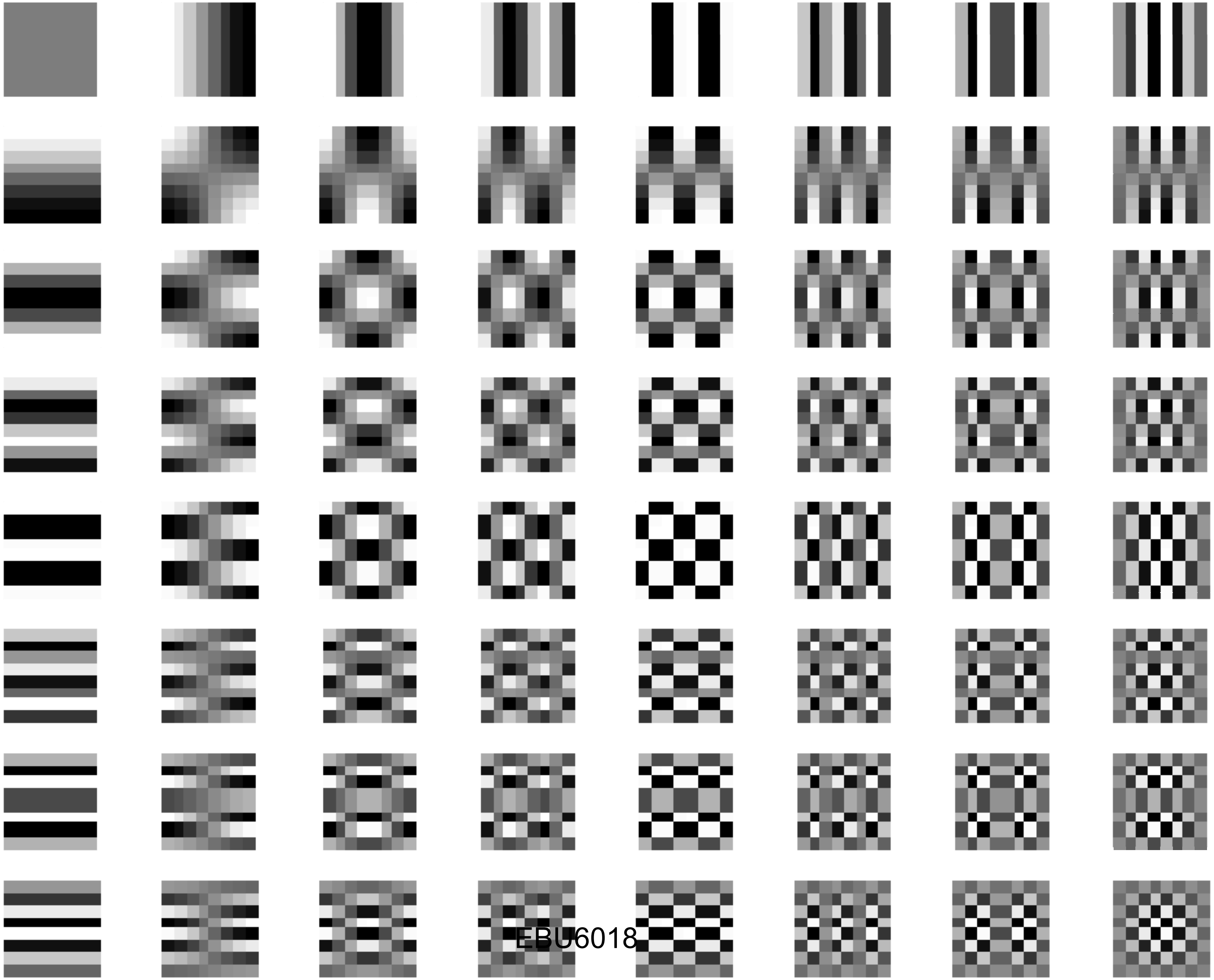
- Compare with:

$$DCT[k] = c(k) \sum_{n=0}^{N-1} s[n] \cos \frac{\pi(2n+1)k}{2N}$$

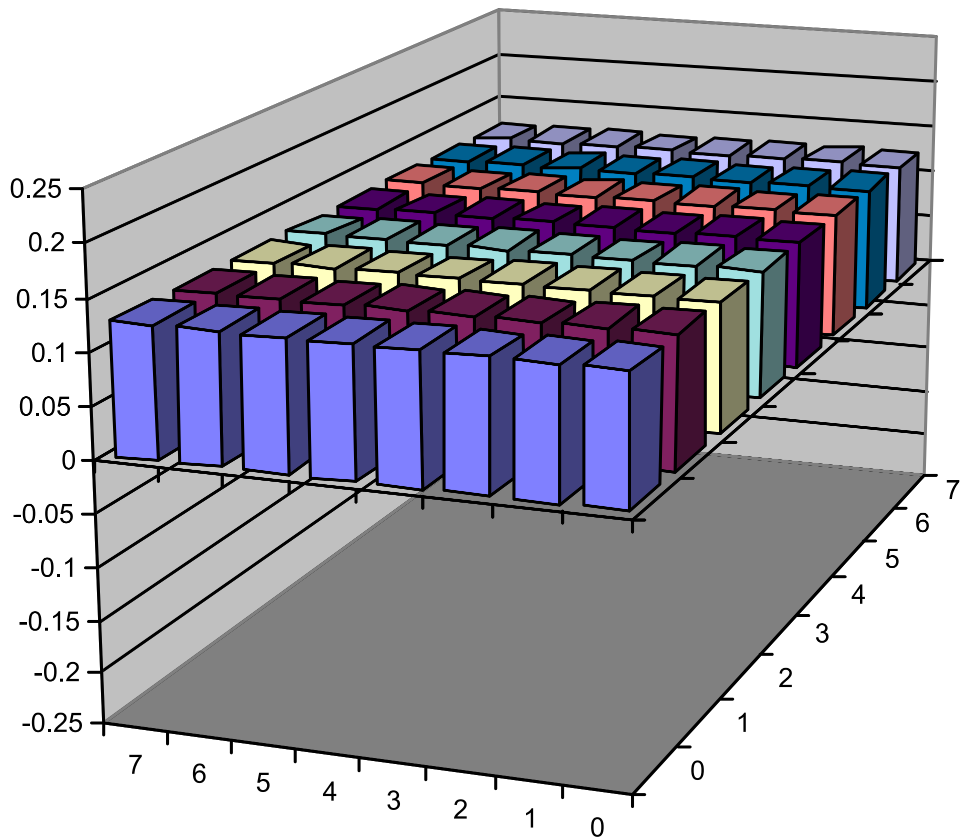
- Separable:

$$DCT_{2d}[i, j] = c(i) \sum_{m=0}^{N-1} \cos \frac{\pi(2m+1)i}{2N} \left[ c(j) \sum_{n=0}^{N-1} s[m, n] \cos \frac{\pi(2n+1)j}{2N} \right]$$

# The 2d-DCT Basis



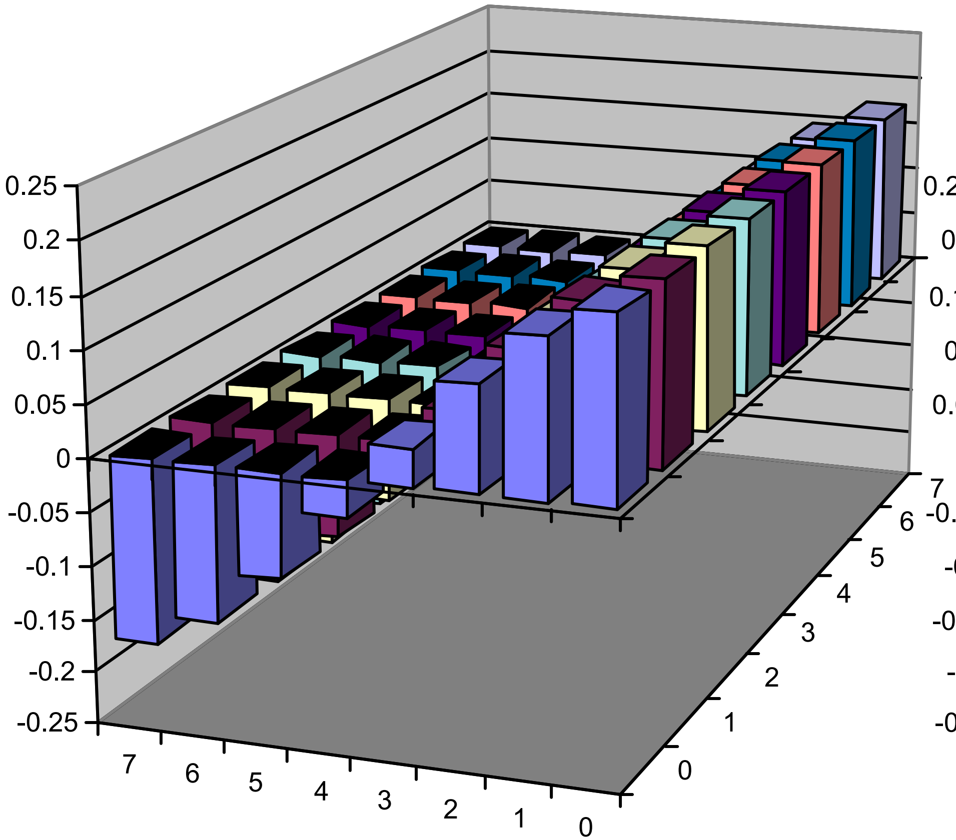
# Inverse DCT of Selected Coefficients

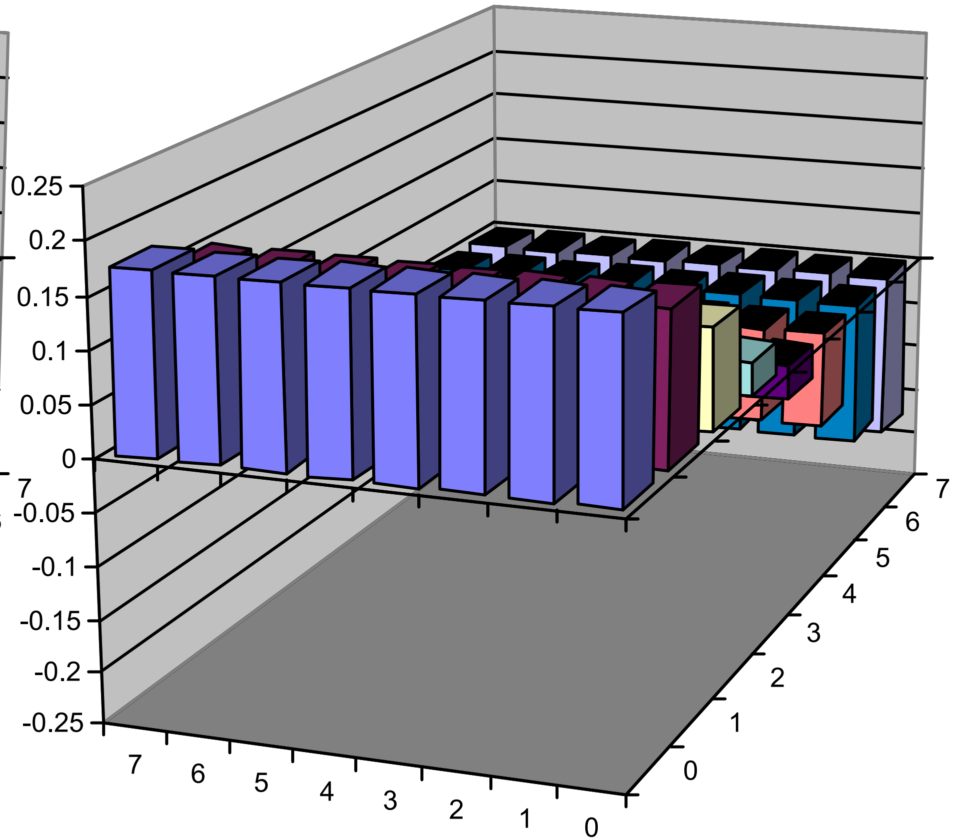



– Coefficient (0, 0)
  - i.e., F(0, 0) = 1, all others = 0

– Characterizes overall average

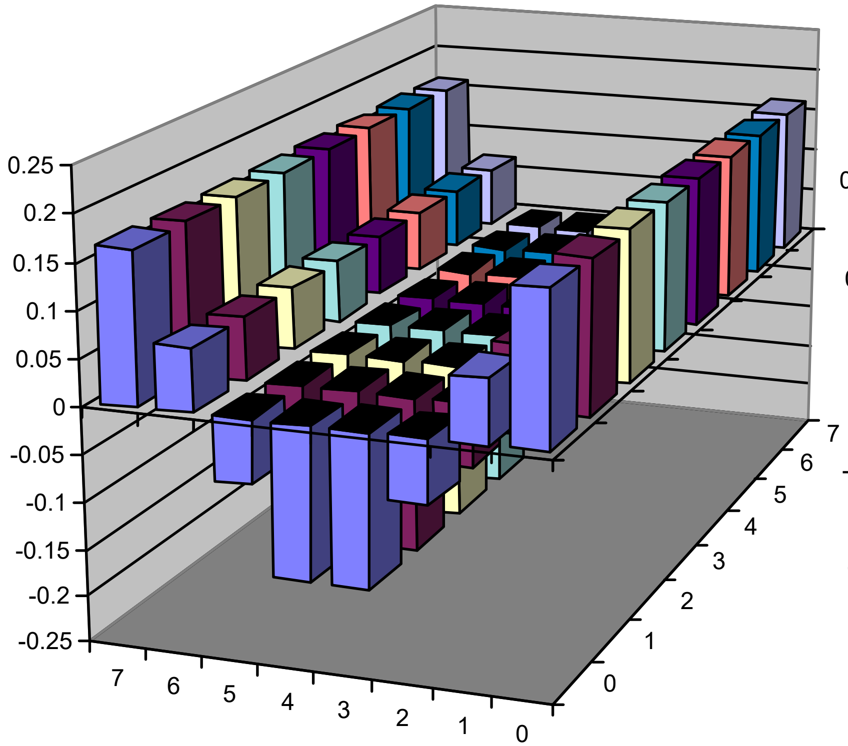
# Inverse DCT of Selected Coefficients



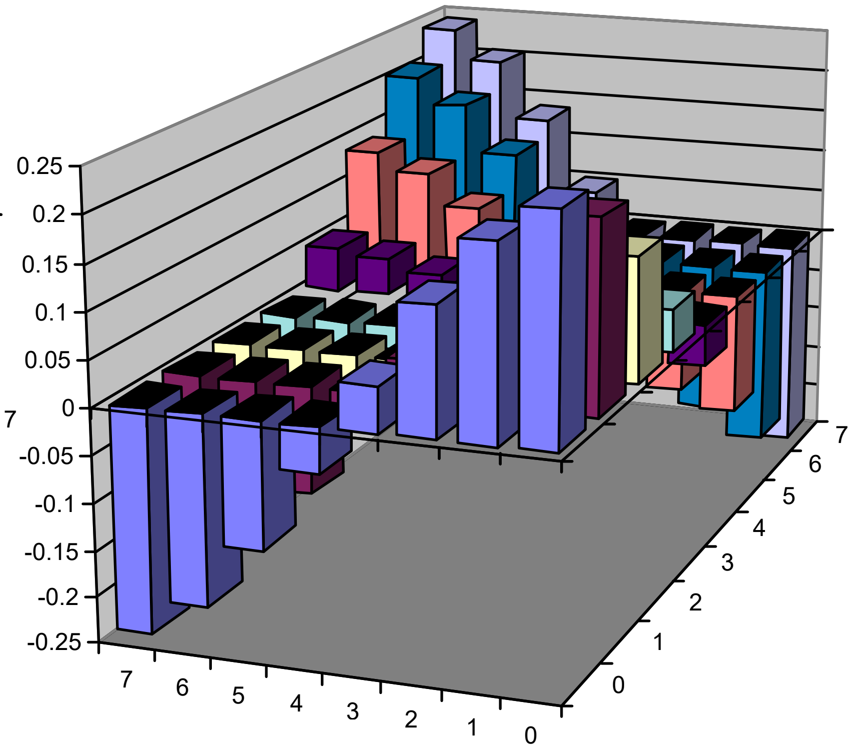




- Coefficients (1,0) and (0,1)
- Capture horizontal or vertical gradient

# Inverse DCT of Selected Coefficients

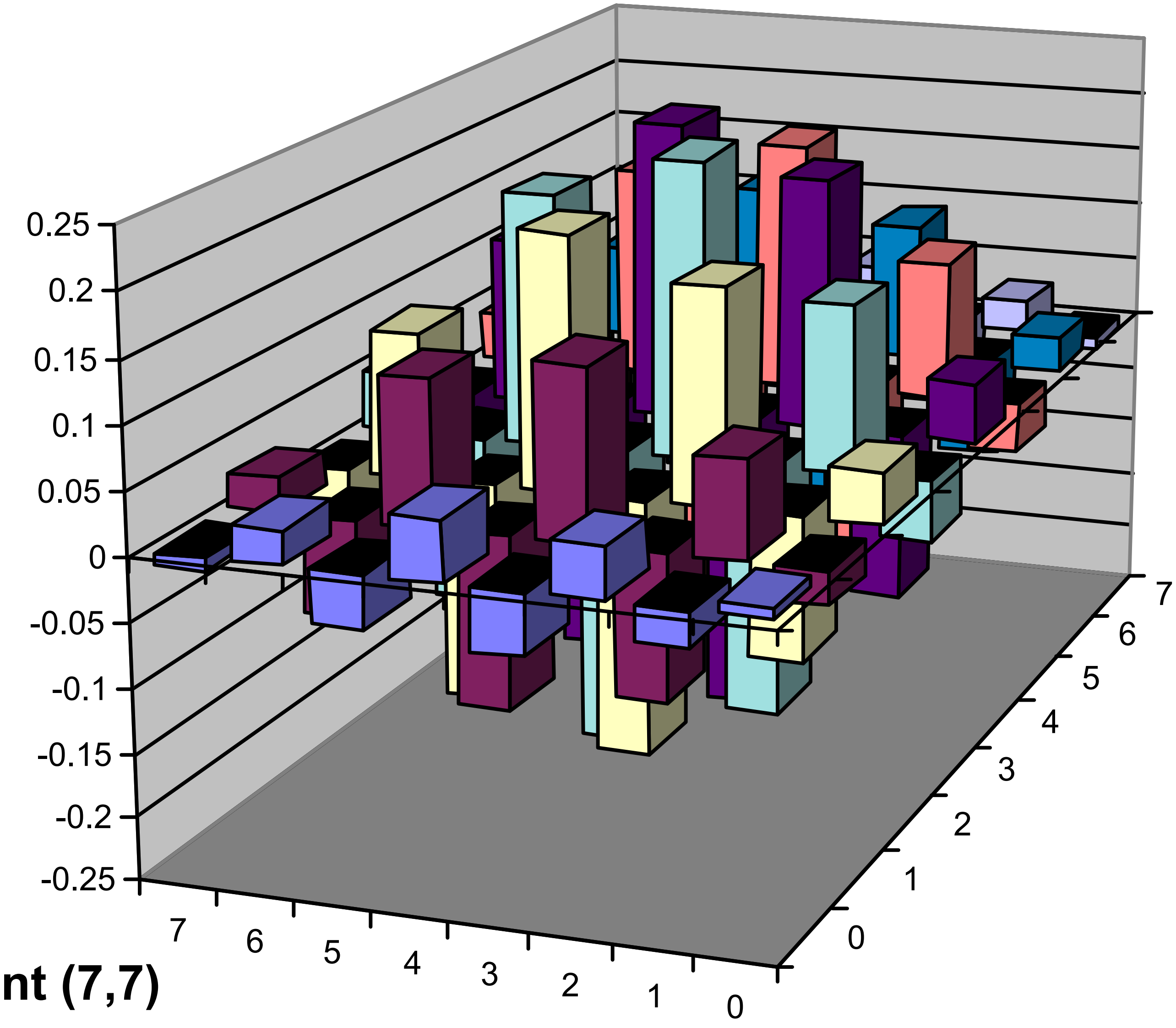



- Coefficient (2,0)
- Captures vertical banding




- **Coefficient (1,1)**
- **Captures diagonal variation**

# Inverse DCT of Selected Coefficients




- Coefficient (7,7)
- Captures high spatial variations

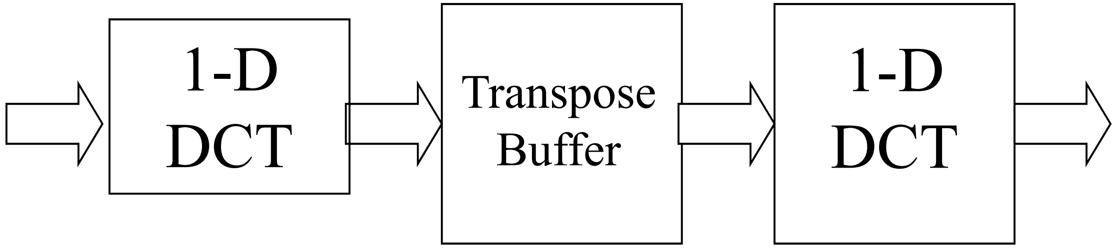
# DCT and Image Compression

- DCT is relatively easy to implement.
- DCT energy compaction allows lossy image compression
  - Just a few of the transform coefficients represent the majority of the **energy** in the sequence
- DCT used in
  - JPEG image compression format
  - MPEG video compression formats.
- Fast algorithms exist for computation.
  - Fixed point integer arithmetic
- Good perceptual properties.
  - Losing higher frequency results in a bit of blurring.

# 2D DCT

- Break the image up into 8 x 8 (64-pixel) blocks and transform them independently.
- Simplifies computation and memory requirements
- DCT is separable, so 2D DCT can be computed by applying 1D transforms separately to the rows and columns
- Rows and columns of blocks are 8 pixels each
- So: need only design a DCT for 8 x 8 transforms
- Result – transform:
  block of 64 intensity values into 64 coefficients

# 2d DCT Implementation



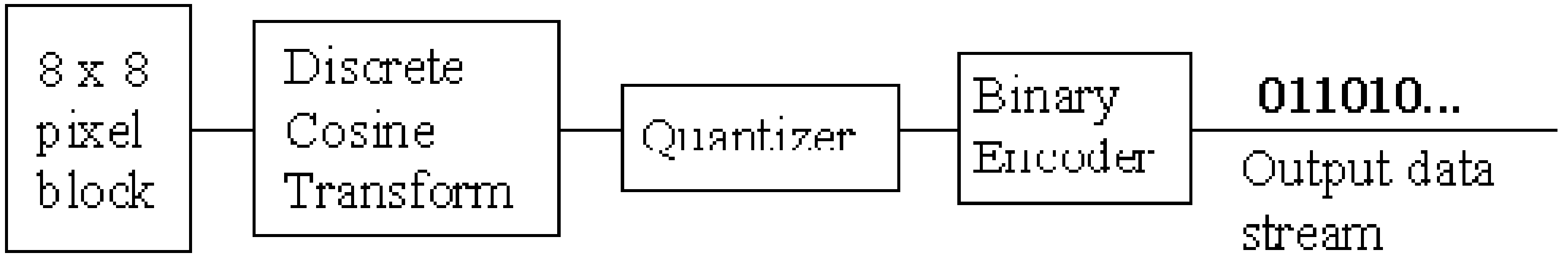

The 8-point DCT can be written as a matrix transform
*Y=AX*

Simplification due to symmetry of A.

Rearranging is possible yielding fast algorithms to compute the DCT (c.f. Fast Fourier algorithms).

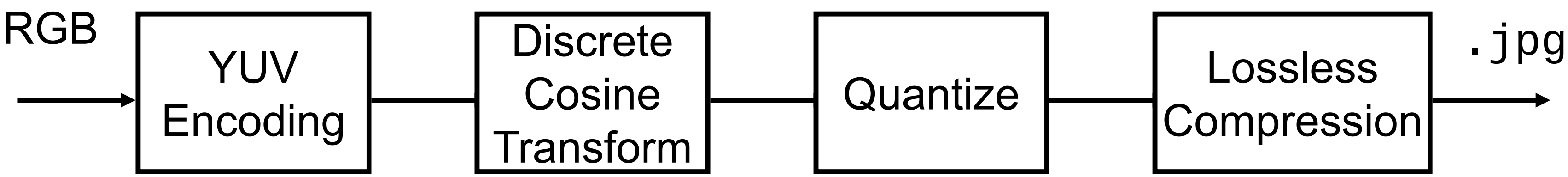
For instance, the 8 point DCT may be computed in just 11 multiplications.

# JPEG Compression - Macroblocks




- YUV encoded image (like RGB)
- Y is cut into 8x8 tiled pixel regions.
- U and V cut into 8x8 tiled pixel regions.
- Macroblock defined as 4 Y tiles that form a 16x16 pixel region and associated U and V tiles.
- Macroblocks organized in row order fashion from top to bottom.

# JPEG Encoding Steps




## Encoding

- **Convert to different color representation**
- **Typically get 2:1 compression**

## Discrete Cosine Transform (DCT)

- **Transform 8 X 8 pixel blocks**

## Quantize

- **Reduce precision of DCT coefficients**
- **Lossy step**

## Lossless Compression

- **Express image in information in highly compressed form**

# YUV Encoding

## Computation

- **RGB numbers between 0 and 255**
- ***Luminance* Y encodes grayscale intensity between 0 and 255**
- ***Chrominance* U, V encode color information between –128 and +127**
  - Similar to Color (Hue) and Tint (Saturation) controls on color TV

## Conversion

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- **Values saturate at ends of ranges**

## Color Subsampling

- **Average U,V values over 2 X 2 blocks of pixels**
- **Human eye less sensitive to variations in color than in brightness**

# DCT Compression

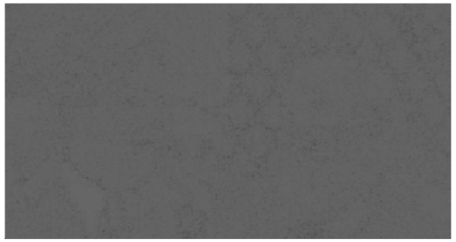
- Each tile (aka block) in a macroblock is transformed with a 2D DCT.
  - 1d: 8 pixel values are transformed into 8 DCT coefficients.
  - 2d: apply 1d transform to all of the rows and then apply 1d transform to all of the columns.
- Each block is now 64 coefficients instead of 64 pixel values.
- Each coefficient quantized independently.
  - Allows larger quantization factors to be used with higher frequency coefficients.
- Quantization is controlled by Quantization table

Original

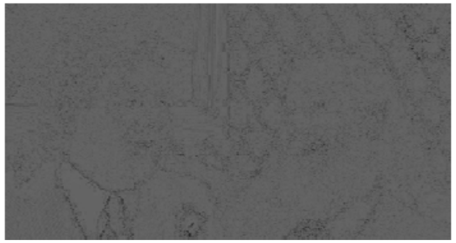
Compressed

Error

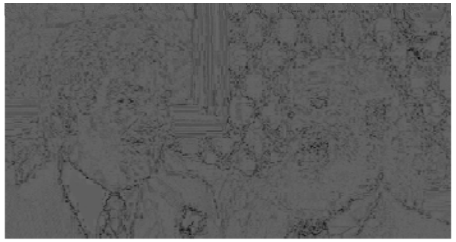
75%



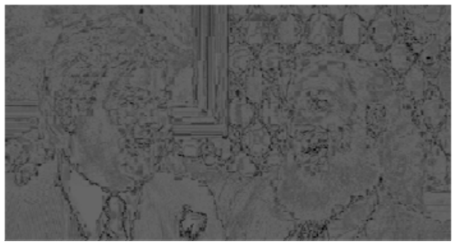
20%

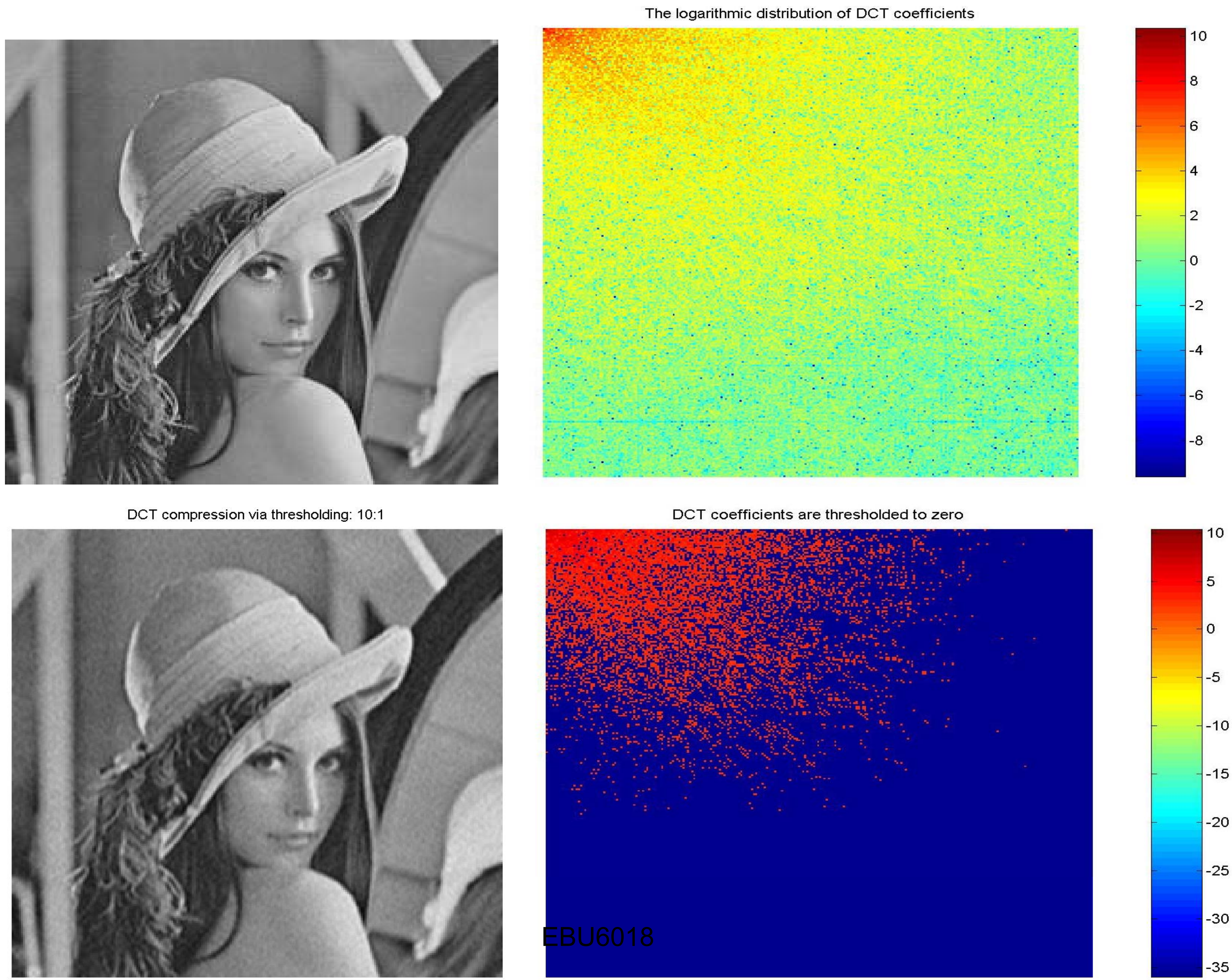


5%



3%




The logarithmic distribution of DCT coefficients
10
8
6
4
2
0
-2
-4
-6
-8
DCT compression via thresholding: 10:1
DCT coefficients are thresholded to zero
10
5
0
-5
-10
-15
-20
-25
-30
-35

EBU6018