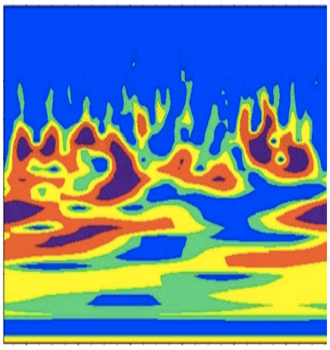
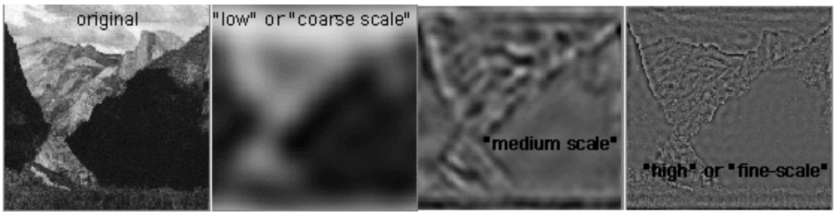
DIGITAL IMAGE PROCESSING

Lecture 5
DCT & Wavelets
Tammy Riklin Raviv
Electrical and Computer Engineering
Ben-Gurion University of the Negev



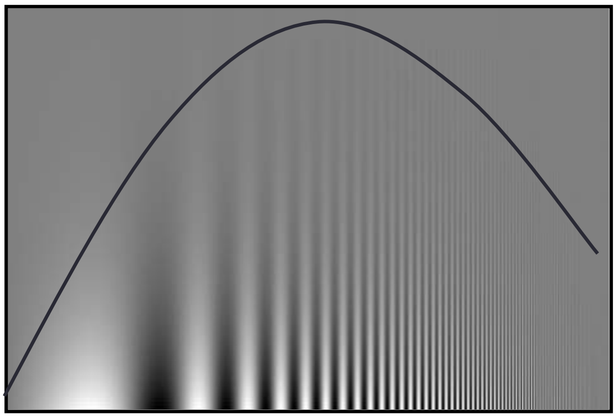
Spatial Frequency Analysis



images of naturally occurring scenes or objects (trees, rocks, bushes, etc.) tend to contain information at many different spatial scales, from very fine to very coarse.

Can't see the forest for the trees

Spatial Frequency Analysis



Campbell-Robson contrast sensitivity curve

http://www.psy.vanderbilt.edu/courses/hon185/SpatialFrequency/SpatialFrequency.html

This class

Why transform?

Discrete Cosine Transform

Jpeg Compression

Wavelets

Why transform?

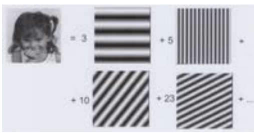
- Better image processing
  - Take into account long-range correlations in space
  - Conceptual insights in spatial-frequency information.
    what it means to be "smooth, moderate change, fast change, …"
  - Denoising

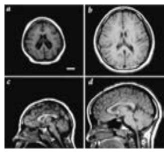
Why transform?

- 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, …"
  - Denoising
- Fast computation: convolution vs. multiplication

Why transform?

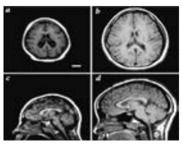
- Better image processing
  - Take into account long-range correlations in space
  - Conceptual insights in spatial-frequency information.
    what it means to be "smooth, moderate change, fast change, …"
  - Denoising
  - Fast computation: convolution vs. multiplication
- Alternative representation and sensing
  - Obtain transformed data as measurement in radiology images (medical and astrophysics), inverse transform to recover image



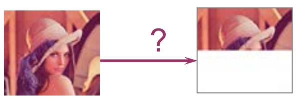


Why transform?

- 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, …"
  - Denoising
- Fast computation: convolution vs. multiplication



- Alternative representation and sensing
  - Obtain transformed data as measurement in radiology images (medical and astrophysics), inverse transform to recover image
- Efficient storage and transmission
  - Energy compaction
  - Pick a few "representatives" (basis)
  - Just store/send the "contribution" from each basis



Is DFT a Good (enough) Transform?

- Theory
- Implementation
- Application

The Desirables for Image Transforms

	DFT	???
• Theory		
• Inverse transform available	X	
• Energy conservation (Parsevell)	X	
• Good for compacting energy	?	
• Orthonormal, complete basis	X	
• (sort of) shift- and rotation invariant	X	
• Implementation		
• Real-valued	x	
• Separable	X	
• Fast to compute w. butterfly-like structure	X	
• Same implementation for forward and inverse transform	X	
• Application		
• Useful for image enhancement	X	
• Capture perceptually meaningful structures in images	?X	

Discrete Cosine Transform - overview

One-dimensional DCT

Orthogonality

Two-dimensional DCT

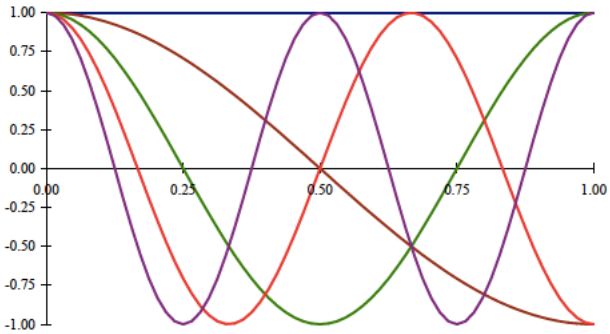
Image Compression (grayscale, color)

Discrete Cosine Transform (DCT)

- A variant of the Discrete Fourier Transform – using only real numbers
- Periodic and symmetric
- The energy of a DCT transformed data (if the original data is correlated) is concentrated in a few coefficients – well suited for compression.
- A good approximation to the optimal Karhunen-Loeve (KL) decomposition of natural image statistics over a small patches

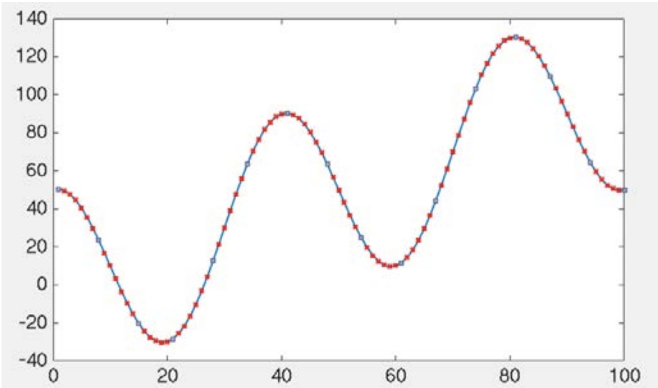
1-D DCT

$$F(k) = \sum_{i=0}^{N-1} \cos\left(\frac{\pi}{N}(i+\frac{1}{2})k\right) f(i)$$



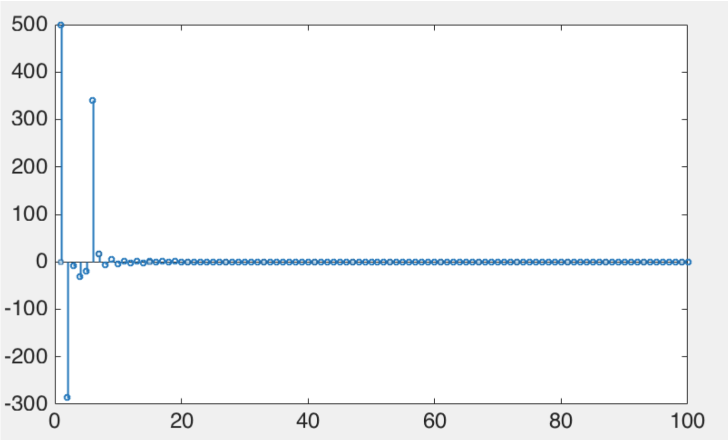
1-D DCT with Matlab

```
>>
>> x = (1:100) + 50*cos((1:100)*2*pi/40);  % Input Signal
>> figure;plot(x)
```



1-D DCT with Matlab

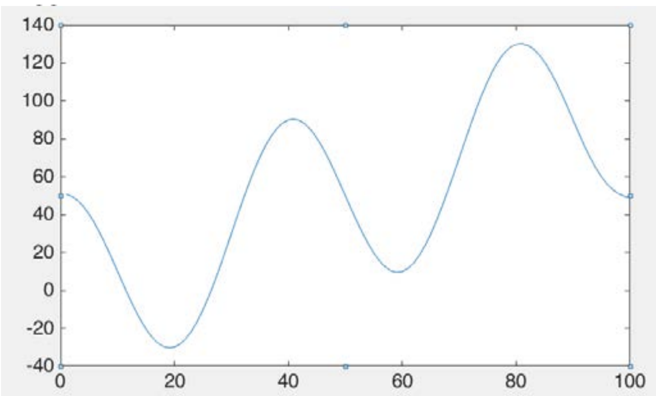
```
>>
>> X = dct(x);                          % Discrete cosine transform
>> figure;stem(X)
```



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

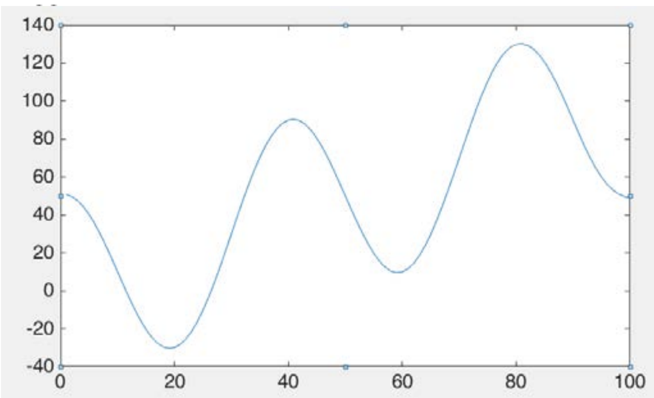
$m = 20$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

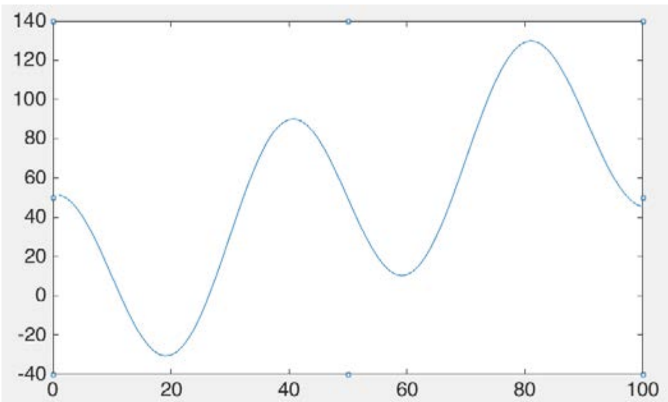
$m = 10$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

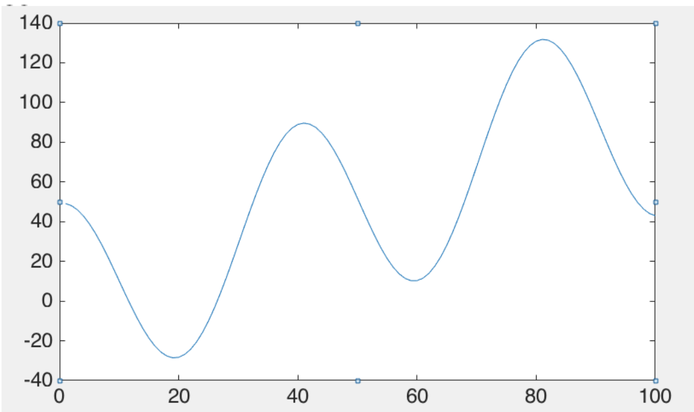
$m = 8$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

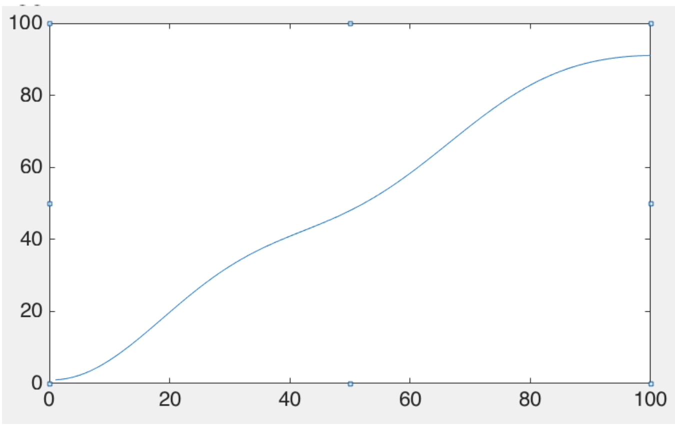
$m = 7$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

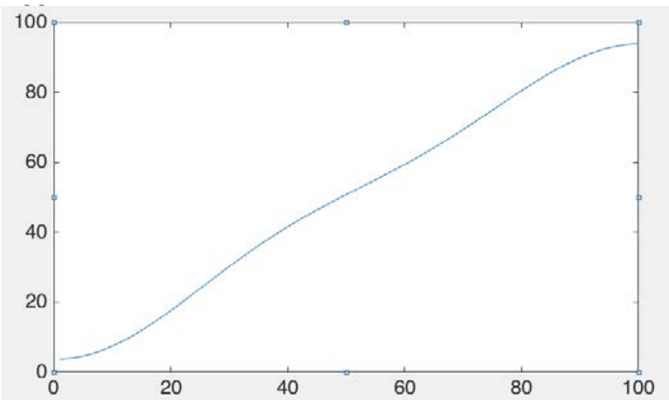
$m = 6$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

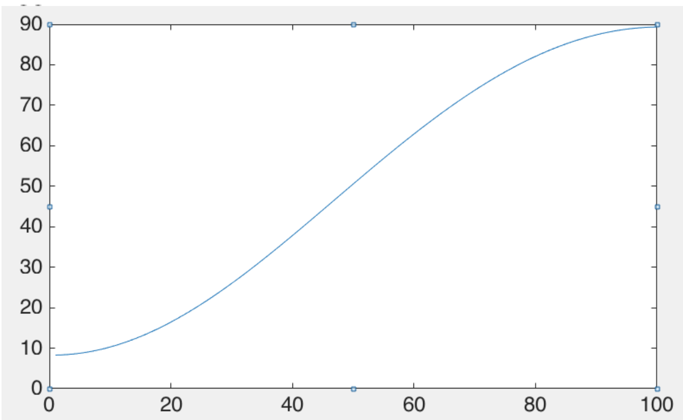
$m = 5$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);
```

$m = 4$



1-D Inverse DCT in Matlab

```
>> Y = X;
>> Y(m:end)=0;
>> z = idct(Y);
>> figure;plot(z);

>>
>> e = zeros(1,10);
>> for m =10:-1:1,
e(m)= norm([X(1:m) zeros(1,100-m)])/norm(X);
end

e =

    0.7452    0.8589    0.8590    0.8603    0.8608    0.9996    0.9999    0.9999    1.0000    1.0000
```

Orthogonality and Orthonormality

Two vectors, a and b, with respective lengths $\|a\|, \|b\|$

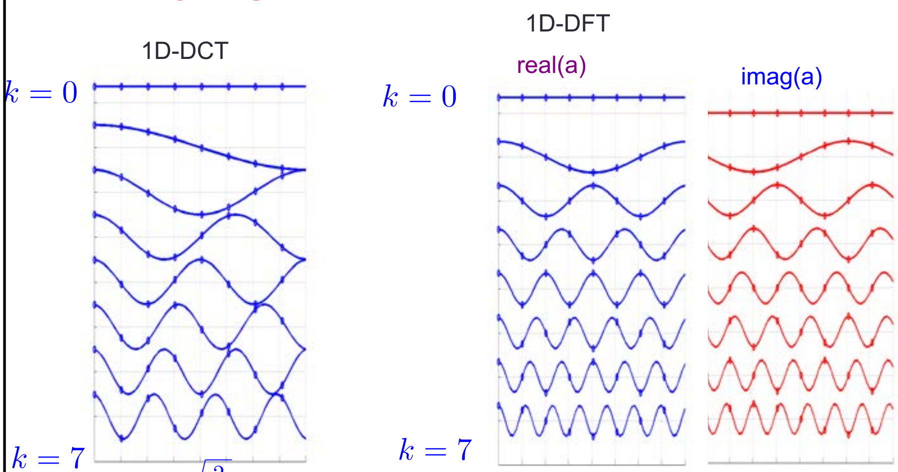
are orthogonal if and only if their dot product

$a \cdot b = \|a\|\|b\| \cos\theta$

is zero.

In addition, two vectors in an inner product space are orthonormal if they are orthogonal and both of unit length.

DFT vs. DCT



$$C(0,i)=\sqrt{\frac{2}{N}}$$

$$C(k,i)=\sqrt{\frac{2}{N}}\cos\left(\frac{\pi}{N}(i+\frac{1}{2})k\right)$$

$$C(k,i)=e^{-j2\pi\frac{ik}{N}}$$

$$=\cos(2\pi\frac{ik}{N})+j\sin(2\pi\frac{ik}{N})$$

DCT Properties

C is real and orthogonal:

The rows of C form an orthogonal basis.

C is not symmetric!

DCT is not the real part of unitary DFT!

The Advantage of Orthogonality

C is orthogonal. $C^T C = I$

$\longrightarrow$ $C^{-1} = C^T$

Makes matrix equation solving easy:

Solve Y = CXC^T for X:

C^{T}Y =$C^T C X C^T$= XC^T

C^{T}YC = XC^{T}C = X

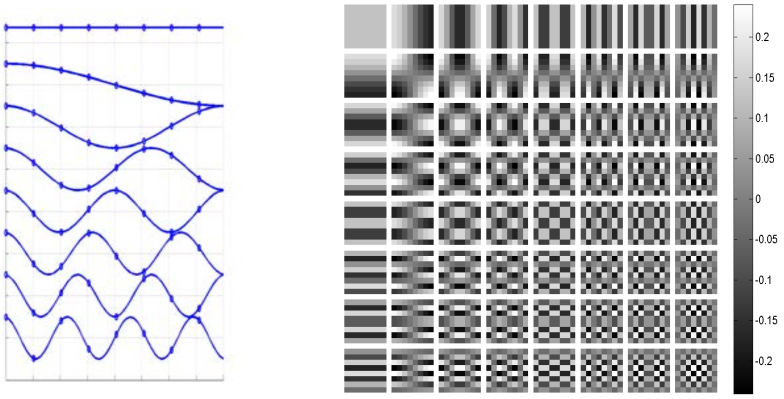
The Advantage of Orthogonality

The discrete cosine transform, C, has one basic characteristic: it is a real orthogonal matrix.

$$C = \sqrt{\frac{2}{n}}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{2n} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{(2n-1)\pi}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \cos\frac{(n-1)\pi}{2n} & \cos\frac{(n-1)3\pi}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

$$C^{-1} = C^T = \sqrt{\frac{2}{n}}\begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\frac{\pi}{2n} & \cdots & \cos\frac{(n-1)\pi}{2n} \\ \frac{1}{\sqrt{2}} & \cos\frac{3\pi}{2n} & \cdots & \cos\frac{(n-1)3\pi}{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{\sqrt{2}} & \cos\frac{(2n-1)\pi}{2n} & \cdots & \cos\frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

From 1D-DCT to 2D-DCT



2D DCT

$$F(k,l) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \cos\left(\frac{\pi}{N}(i+\frac{1}{2})k\right)\cos\left(\frac{\pi}{N}(j+\frac{1}{2})l\right) f(i,j)$$

Like the 2D Fast Fourier Transform, the 2D DCT can be implemented in two stages, i.e., first
computing the DCT of each line in the block and then computing the DCT of each resulting column.

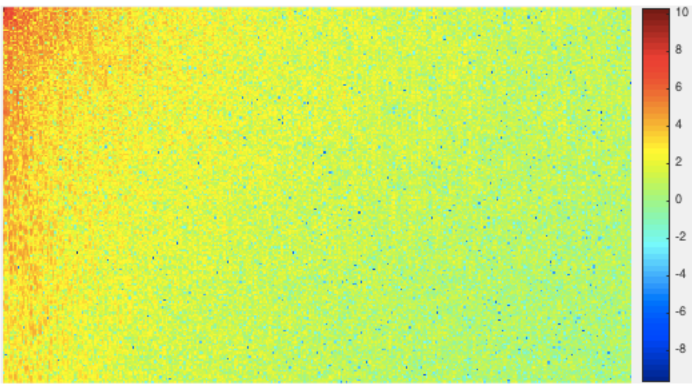
Like the FFT, each of the DCTs can also be computed in O(N logN) time.

DCT in Matlab

```
>> RGB = imread('autumn.tif');
>> figure;imshow(RGB);
```

DCT in Matlab

```
>> I = rgb2gray(RGB);
J = dct2(I);
imshow(log(abs(J)),[]), colormap(jet(64)), colorbar
```



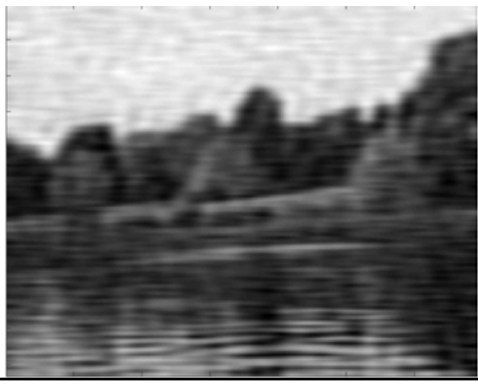
DCT in Matlab

```
>> J(abs(J) < 10) = 0;
K = idct2(J);
imshow(I)
figure, imshow(K,[0 255])
```

23.45%

DCT in Matlab

```
>> J(abs(J) < 100) = 0;
K = idct2(J);
imshow(I)
figure, imshow(K,[0 255])
```



1.33%

2D - DCT

- Idea 2D-DCT: Interpolate the data with a set of basis functions
- Organize information by order of importance to the human visual system
- Used to compress small blocks of an image
  (8 x 8 pixels in our case)

Why blocks?

2D DCT

Use One-Dimensional DCT in both horizontal and vertical directions.

First direction $F = C*X^T$
Second direction $G = C*F^T$

We can say 2D-DCT is the matrix:

$$Y = C(CX^T)^T$$

2D DCT (Matlab)

$$B_{k,l} = c_k c_l \cos\left(\frac{\pi}{N}(i+\frac{1}{2})k\right)\cos\left(\frac{\pi}{M}(j+\frac{1}{2})l\right)$$

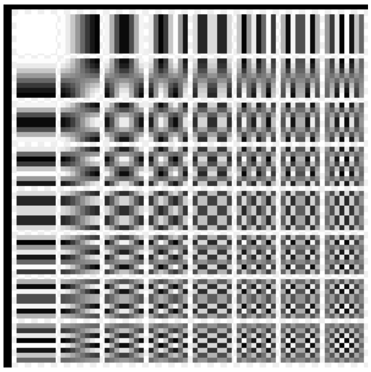
$$i = 0, \ldots N, \quad j = 0, \ldots M$$

$$c_{k=0} = \frac{1}{\sqrt{N}}$$

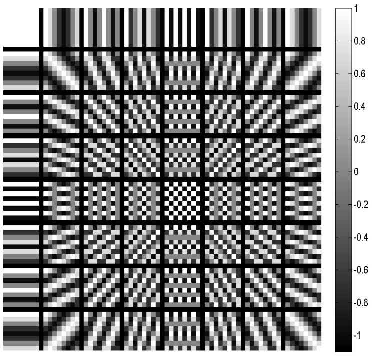
$$c_{k>0} = \frac{2}{\sqrt{N}}$$

$$c_{l=0} = \frac{1}{\sqrt{M}}$$

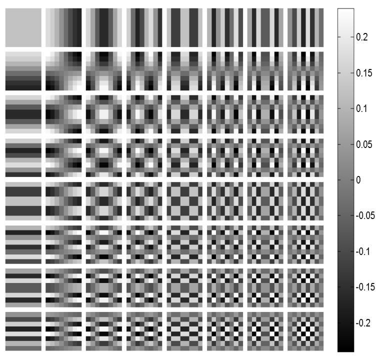
$$c_{l>0} = \frac{2}{\sqrt{M}}$$



basis images: DFT (real) vs DCT



DFT



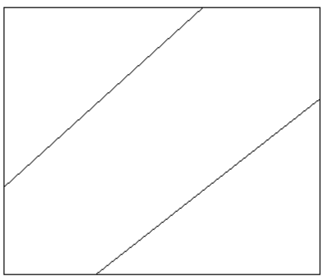
DCT

Image Compression

- Image compression is a method that reduces the amount of memory it takes to store in image.
- We will exploit the fact that the DCT matrix is based on our visual system for the purpose of image compression.
- This means we can delete the least significant values without our eyes noticing the difference.

Image Compression

- Now we have found the matrix $Y = C(CX^T)^T$
- Using the DCT, the entries in Y will be organized based on the human visual system.
- The most important values to our eyes will be placed in the upper left corner of the matrix.
- The least important values will be mostly in the lower right corner of the matrix.



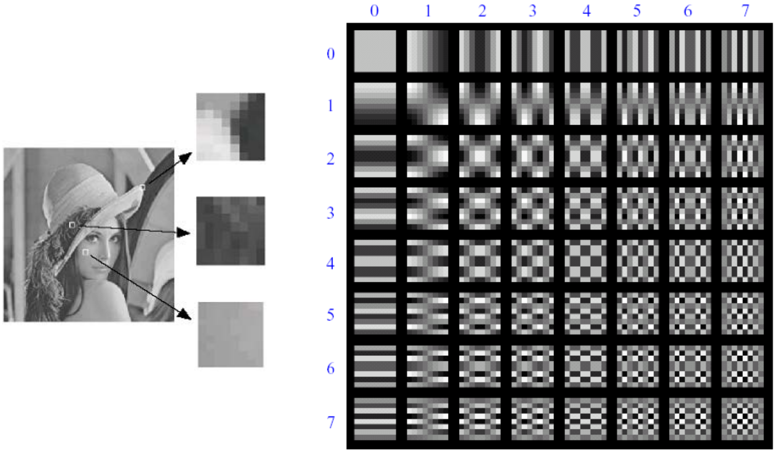
Common Applications

- JPEG Format
- MPEG-1 and MPEG-2
- MP3, Advanced Audio Coding, WMA

- What's in common?
  - All share, in some form or another, a DCT method for compression.

JPEG = Joint Photographic Experts Group

Lossy Image Compression (JPEG)



Block-based Discrete Cosine Transform (DCT)

Using DCT in JPEG

- The first coefficient $B(0,0)$ is the DC component, the average intensity
- The top-left coeffs represent low frequencies, the bottom right – high frequencies

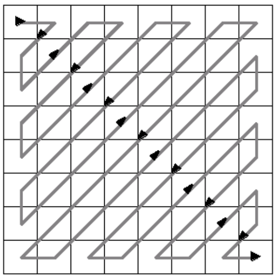


Image compression using DCT

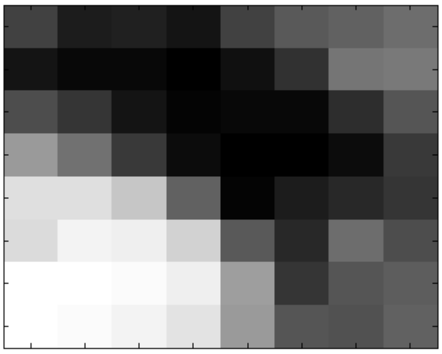
- DCT enables image compression by concentrating most image information in the low frequencies
- Loose unimportant image info (high frequencies)
- The decoder computes the inverse DCT – IDCT

Block size in JPEG

- What Should be the Optimal Block size?
  - small block
    - faster
    - correlation exists between neighboring pixels
  - large block
    - better compression in smooth regions
  - It's 8x8 in standard JPEG

Image Compression

8 x 8 Pixels



Image

Image Compression

- Gray-Scale Example:
- Value Range 0 (black) --- 255 (white)

63	33	36	28	63	81	86	98
27	18	17	11	22	48	104	108
72	52	28	15	17	16	47	77
132	100	56	19	10	9	21	55
187	186	166	88	13	34	43	51
184	203	199	177	82	44	97	73
211	214	208	198	134	52	78	83
211	210	203	191	133	79	74	86

I

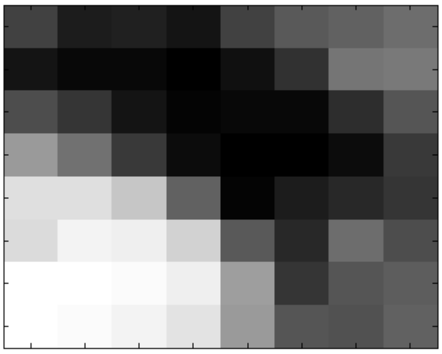


Image Compression

- 2D-DCT of matrix

-304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1

J

Abs(J)

Image Compression

-304	210	104	-69	0	20	0	0
-327	-260	67	70	0	0	21	0
93	-84	-66	0	24	0	0	0
89	33	0	-20	-26	21	0	0
0	42	0	27	0	0	29	0
0	0	0	0	32	0	0	0
0	0	0	0	0	0	0	0
0	30	0	0	0	0	0	0

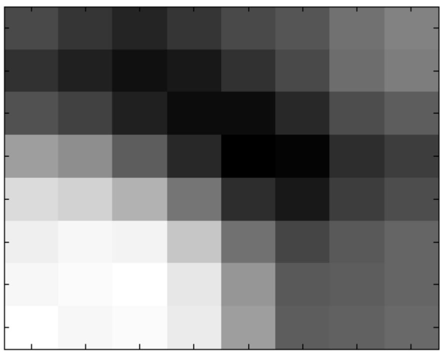
J(abs(J)<20) =0

abs(J)

Reconstructing the Image

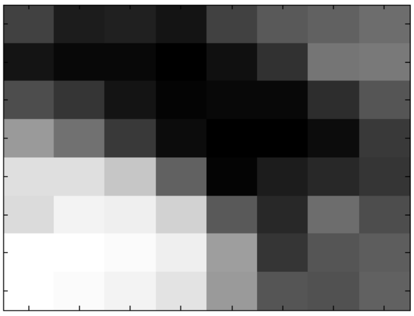
- New Matrix and Compressed Image

55	41	27	39	56	69	92	106
35	22	7	16	35	59	88	101
65	49	21	5	6	28	62	73
130	114	75	28	-7	-1	33	46
180	175	148	95	33	16	45	59
200	206	203	165	92	55	71	82
205	207	214	193	121	70	75	83
214	205	209	196	129	75	78	85



Can You Tell the Difference?

Original



Compressed

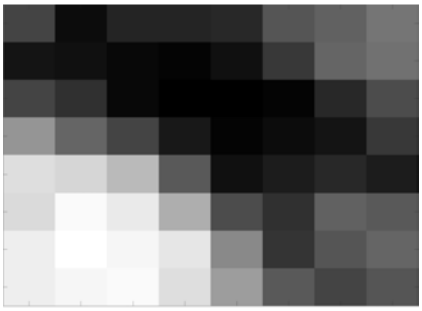


Image Compression

Original

Compressed

PSNR – Class work

- Define what is PSNR
- Read and image and present it.
- Deform the image as you wish. e.g., apply low-pass filter, add noise, compress and expand again …
- Present the deformed image
- Calculate the PSNR between the images

Linear Quantization

- We will not zero the bottom half of the matrix.

- The idea is to assign fewer bits of memory to store information in the lower right corner of the DCT matrix.

Linear Quantization

Use Quantization Matrix (Q)

$q_{kl} = 8p(k + l + 1)$ for $0 \leq k, l \leq 7$

Q = p *

8	16	24	32	40	48	56	64
16	24	32	40	48	56	64	72
24	32	40	48	56	64	72	80
32	40	48	56	64	72	80	88
40	48	56	64	72	80	88	96
48	56	64	72	80	88	96	104
56	64	72	80	88	95	104	112
64	72	80	88	96	104	112	120

Linear Quantization

- p is called the loss parameter
- It acts like a “knob” to control compression
- The greater p is the more you compress the image

Linear Quantization

We divide the each entry in the DCT matrix by the Quantization Matrix

-304	210	104	-69	10	20	-12	7
-327	-260	67	70	-10	-15	21	8
93	-84	-66	16	24	-2	-5	9
89	33	-19	-20	-26	21	-3	0
-9	42	18	27	-7	-17	29	-7
-5	15	-10	17	32	-15	-4	7
10	3	-12	-1	2	3	-2	-3
12	30	0	-3	-3	-6	12	-1

8	16	24	32	40	48	56	64
16	24	32	40	48	56	64	72
24	32	40	48	56	64	72	80
32	40	48	56	64	72	80	88
40	48	56	64	72	80	88	96
48	56	64	72	80	88	96	104
56	64	72	80	88	95	104	112
64	72	80	88	96	104	112	120

Linear Quantization

p = 1

-38	13	4	-2	0	0	0	0
-20	-11	2	2	0	0	0	0
4	-3	-2	0	0	0	0	0
3	1	0	0	0	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

New Y: 14 terms

p = 4

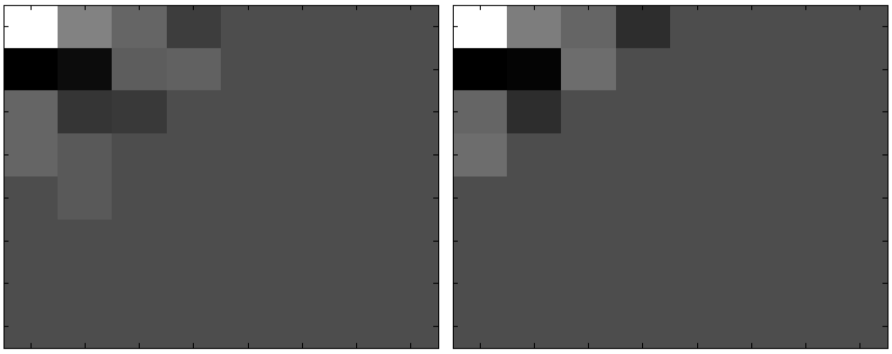
-9	3	1	-1	0	0	0	0
-5	-3	1	0	0	0	0	0
1	-1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

New Y: 10 terms

Linear Quantization

p = 1

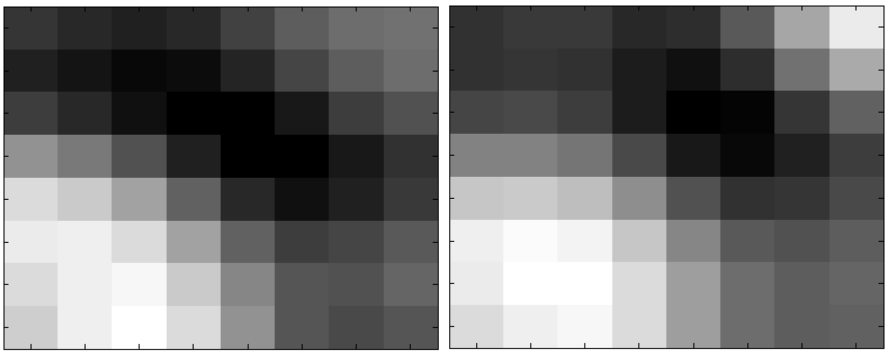
p = 4



Linear Quantization

Compressed with P=1

Compressed with P=4



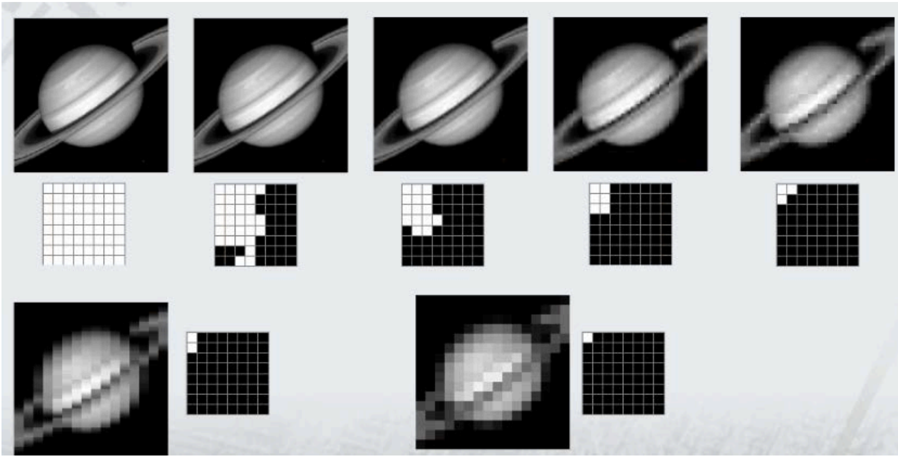
Linear Quantization

p = 1

Linear Quantization

P= 4

Quantization Example



Further Compression

- Run-Length Encoding (RLE)
- Huffman Coding

Memory Storage

- The original image uses one byte (8 bits) for each pixel. Therefore, the amount of memory needed for each 8 x 8 block is:
  - $8 \times (8^2) = 512$ bits

Is This Worth the Work?

- The question that arises is "How much memory does this save?"

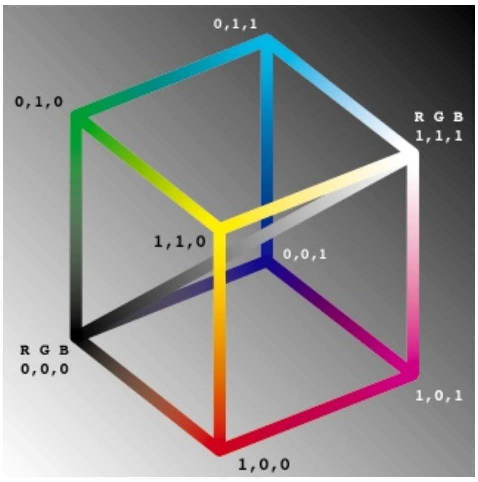
Linear Quantization

p	Total bits	Bits/pixel
X	512	8
1	249	3.89
2	191	2.98
3	147	2.30

JPEG Imaging

- It is fairly easy to extend this application to color images.
  - These are expressed in the RGB color system.
  - Each pixel is assigned three integers for each color intensity.

RGB Coordinates

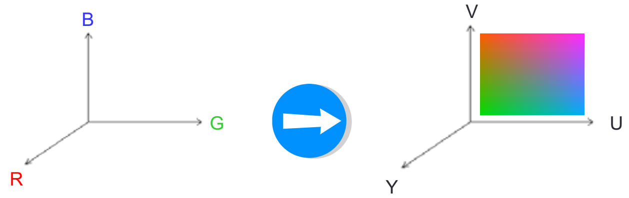


The Approach

- There are a few ways to approach the image compression.
  - Repeat the discussed process independently for each of the three colors and then reconstruct the image.
  - Baseline JPEG uses a more delicate approach.
    - Define the luminance coordinate to be:
      - $Y = 0.299R + 0.587G + 0.114B$
    - Define the color differences coordinates to be:
      - $U = B - Y$
      - $V = R - Y$

More on Baseline

- This transforms the RGB color data to the YUV system which is easily reversible.



- It applies the DCT filtering independently to Y, U, and V using the quantization matrix Q_Y.

JPEG Quantization

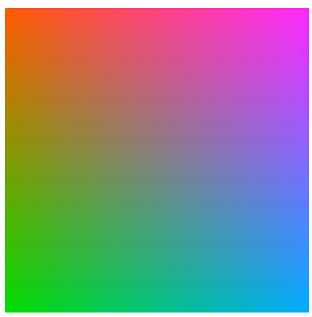
Luminance:

Q_Y =

p { 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}

JPEG Quantization

Chrominance:



Q_C =

{ 17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99}

Luminance and Chrominance

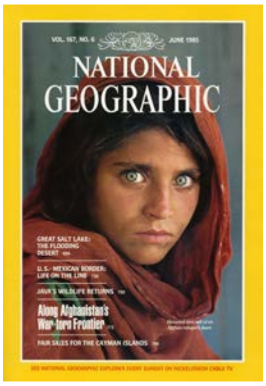
- Human eye is more sensible to luminance (Y coordinate).
- It is less sensible to color changes (UV coordinates).
- Then: compress more on UV !
- Consequence: color images are more compressible than grayscale ones

Reconstitution

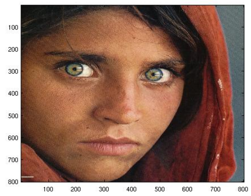
- After compression, Y, U, and V, are recombined and converted back to RGB to form the compressed color image:

B= U+Y
R= V+Y
G= (Y- 0.299R - 0.114B) / 0.587

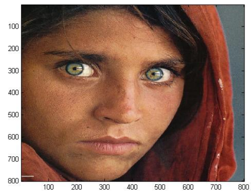
Comparing Compression



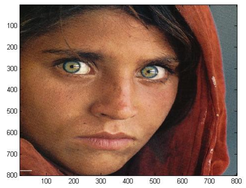
Original



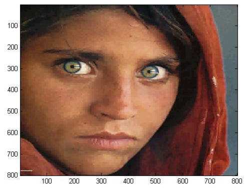
p = 1



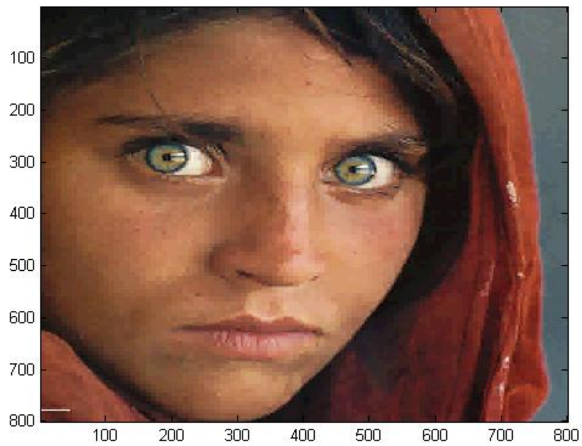
p = 4



p = 8



Up Close



JPEG compression comparison


89k


12k

See also

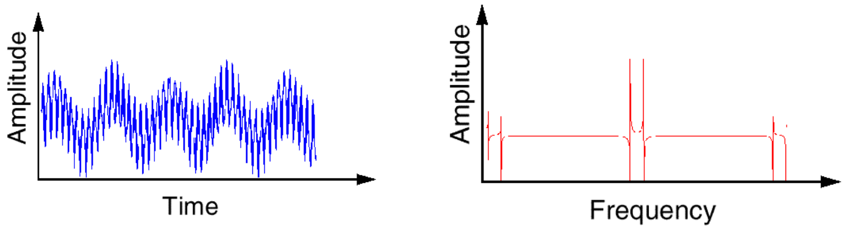
https://en.wikipedia.org/wiki/JPEG

Wavelets - overview

- Why wavelets?
- Wavelets like basis components.
- Wavelets examples.
- Fast wavelet transform .
- Wavelets like filter.
- Wavelets advantages.

Fourier Analysis

- Breaks down a signal into constituent sinusoids of different frequencies



In other words: Transform the view of the signal from time-base to frequency-base.

What's wrong with Fourier?

- By using Fourier Transform , we loose the time information : WHEN did a particular event take place ?
- FT can not locate drift, trends, abrupt changes, beginning and ends of events, etc.
- Calculating use complex numbers.

Time and Space definition

- Time – for one dimension waves we start point shifting from source to end in time scale .
- Space – for image point shifting is two dimensional .
- Here they are synonyms .

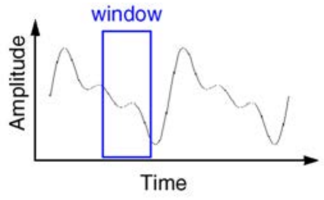
Kronneker function

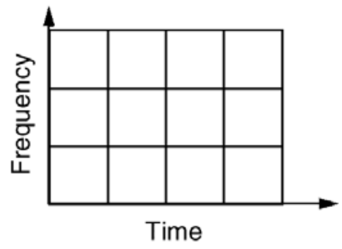
$$\psi_k(t) = \delta_k(t) = \left\{ \frac{1, k=t}{0, k \neq t} \right.$$

Can exactly show the time of appearance but have not information about frequency and shape of signal.

Short Time Fourier Analysis

- In order to analyze small section of a signal, Denis Gabor (1946), developed a technique, based on the FT and using *windowing* : STFT





STFT (or: Gabor Transform)

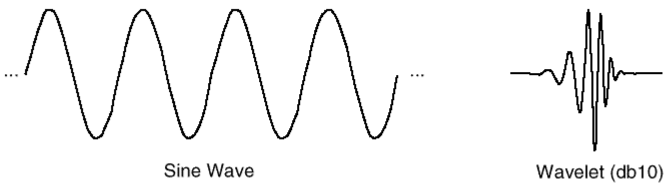
- A compromise between time-based and frequency-based views of a signal.
- both time and frequency are represented in limited precision.
- The precision is determined by the size of the window.
- Once you choose a particular size for the time window - it will be the same for all frequencies.

What's wrong with Gabor?

- Many signals require a more flexible approach - so we can vary the window size to determine more accurately either time or frequency.

What is Wavelet Analysis ?

- And…what is a wavelet…?



- A wavelet is a waveform of effectively limited duration that has an average value of zero.

Wavelet's properties

- Short time localized waves with zero integral value.
- Possibility of time shifting.
- Flexibility.

The Continuous Wavelet Transform (CWT)

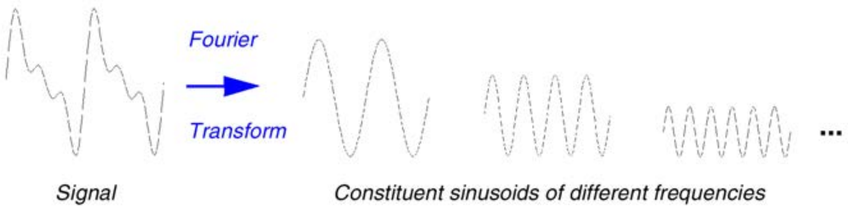
- A mathematical representation of the Fourier transform:

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-iwt}dt$$

- Meaning: the sum over all time of the signal *f(t)* multiplied by a complex exponential, and the result is the Fourier coefficients F(ω) .

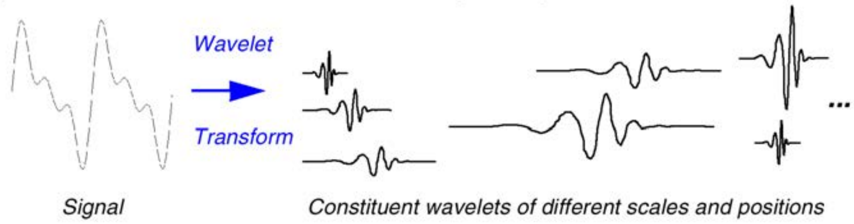
Wavelet Transform (Cont'd)

- Those coefficients, when multiplied by a sinusoid of appropriate frequency , , yield the constituent sinusoidal component of the original signal:



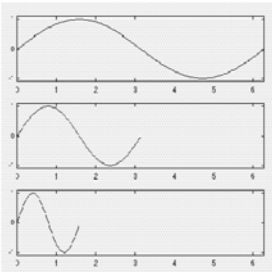
Wavelet Transform

- And the result of the CWT are Wavelet coefficients .
- Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelet of the original signal:



Scaling

- Wavelet analysis produces a time-scale view of the signal.
- *Scaling* means stretching or compressing of the signal.
- scale factor (a) for sine waves:



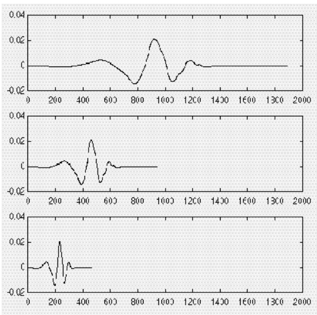
$$f(t) = \sin(t) \;\; ; \;\; a = 1$$

$$f(t) = \sin(2t) \;\; ; \;\; a = \tfrac{1}{2}$$

$$f(t) = \sin(4t) \;\; ; \;\; a = \tfrac{1}{4}$$

Scaling (Cont'd)

- Scale factor works exactly the same with wavelets:



$$f(t) = \Psi(t) \;\; ; \;\; a = 1$$

$$f(t) = \Psi(2t) \;\; ; \;\; a = \tfrac{1}{2}$$

$$f(t) = \Psi(4t) \;\; ; \;\; a = \tfrac{1}{4}$$

Wavelet function 1D -> 2D

$$\Psi_{a,b}(x) = \frac{1}{\sqrt{a}}\Psi\left(\frac{x-b}{a}\right)$$

- b – shift coefficient
- a – scale coefficient

$$\Psi_{a,b_x,b_y}(x,y) = \frac{1}{|a|}\Psi\left(\frac{x-b_x}{a}, \frac{y-b_y}{a}\right)$$

- 2D function

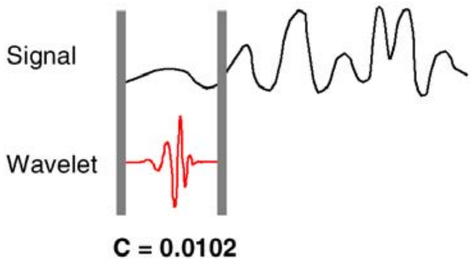
CWT

- Reminder: The *CWT* Is the sum over all time of the signal, multiplied by scaled and shifted versions of the wavelet function

Step 1:
Take a Wavelet and compare
it to a section at the start
of the original signal

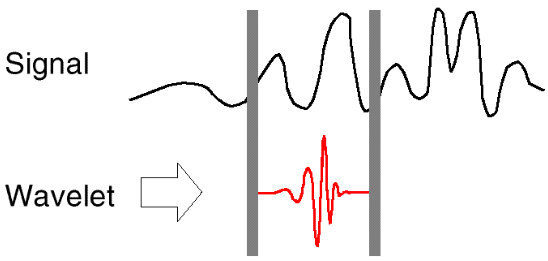
CWT

Step 2:
Calculate a number, C, that represents how closely correlated the wavelet is with this section of the signal. The higher C indicates higher similarity.



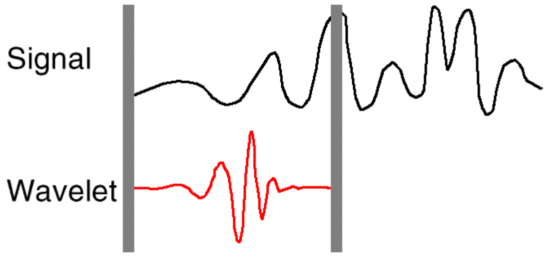
CWT

- Step 3: Shift the wavelet to the right and repeat steps 1-2 until you've covered the whole signal



CWT

- Step 4: Scale (stretch) the wavelet and repeat steps 1-3



C = 0.2247

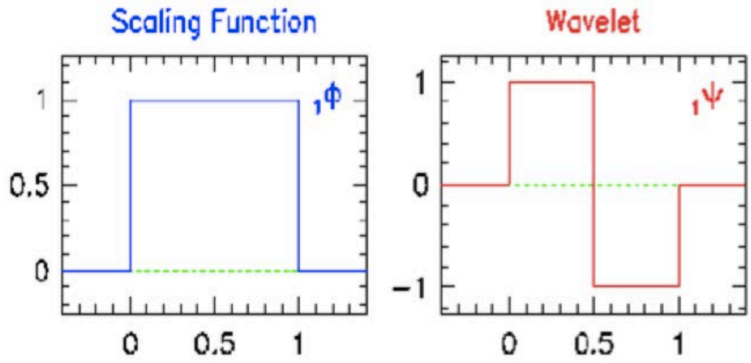
Wavelets examples

Dyadic transform

- For easier calculation we can discretize a continuous signal.
- We have a grid of discrete values that called dyadic grid .
- Important: wavelet functions are compact (e.g. no overcalculatings) .

$$a = 2^j$$
$$b = k2^j$$

Haar Wavelets



$$\phi(t) = \begin{cases} 1 & 0 \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

$$\psi(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2}, \\ -1 & \frac{1}{2} \le t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

Haar Wavelets Properties I

Any continuous real function with compact support can be approximated uniformly by linear combination of:

$$\phi(t), \phi(2t), \phi(4t), \ldots, \phi(2^n t), \ldots$$

and their shifted functions

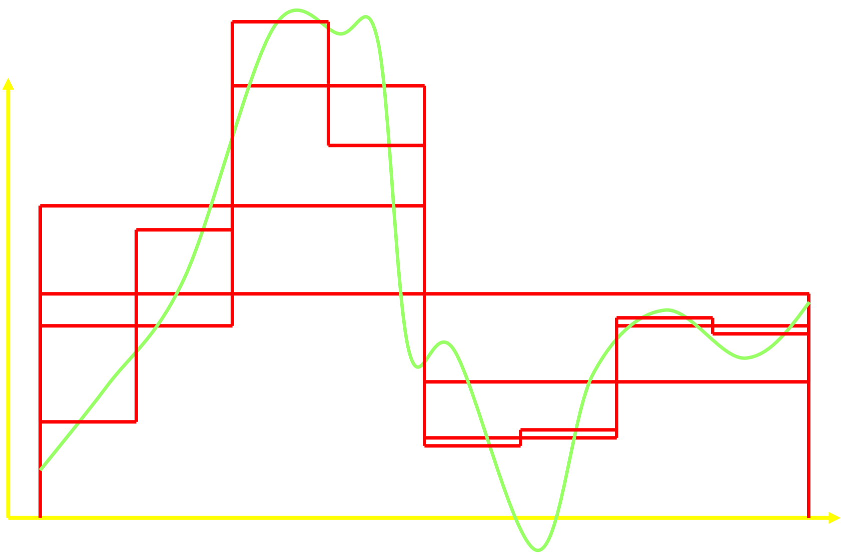
Haar Wavelets Properties II

Any continuous real function on [0, 1] can be approximated uniformly on [0, 1] by linear combinations of the constant function

$$\psi(t), \psi(2t), \psi(4t), \ldots, \psi(2^n t), \ldots$$

and their shifted functions

Haar transform



Haar Wavelets Properties III

Orthogonality

$$\int_{-\infty}^{\infty} 2^{(n+n_1)/2}\psi(2^n t - k)\psi(2^{n_1} t - k_1)\, dt = \delta_{n,n_1}\delta_{k,k_1}$$

Haar Wavelets Properties

Functional relationship:

$$\phi(t) = \phi(2t) + \phi(2t-1)$$
$$\psi(t) = \phi(2t) - \phi(2t-1),$$

It follows that coefficients of scale *n* can be calculated by coefficients of scale *n+1*:

If $\chi_w(k,n) = 2^{n/2}\int_{-\infty} x(t)\phi(2^n t - k)\, dt$

and $\mathrm{X}_w(k,n) = 2^{n/2}\int_{-\infty}^{\infty} x(t)\psi(2^n t - k)\, dt$

then

$$\chi_w(k,n) = 2^{-1/2}\left(\chi_w(2k,n+1) + \chi_w(2k+1,n+1)\right)$$
$$\mathrm{X}_w(k,n) = 2^{-1/2}\left(\chi_w(2k,n+1) - \chi_w(2k+1,n+1)\right)$$

Haar Matrix

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

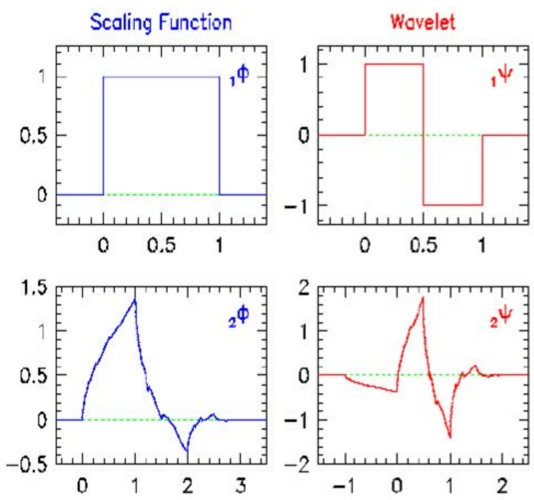
$$(a_0, a_1, \dots, a_{2n}, a_{2n+1}) \Longrightarrow ((a_0, a_1), \dots, (a_{2n}, a_{2n+1}))$$

$$\Longrightarrow ((s_0, d_0), \dots, (s_n, d_n))$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \qquad H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Wavelet functions examples

- Haar function
- Daubechies function



Properties of Daubechies wavelets

I. Daubechies, *Comm. Pure Appl. Math.* 41 (1988) 909.

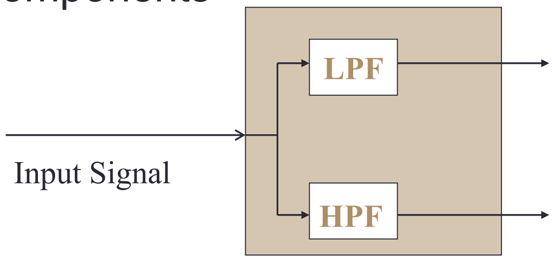
- Compact support
  - finite number of filter parameters / fast implementations
  - high compressibility
  - fine scale amplitudes are very small in regions where the function is smooth / sensitive recognition of structures
- Identical forward / backward filter parameters
  - fast, exact reconstruction
  - very asymmetric

Mallat* Filter Scheme

- Mallat was the first to implement this scheme, using a well known filter design called "two channel sub band coder", yielding a *'Fast Wavelet Transform'*

Approximations and Details:

- Approximations: High-scale, low-frequency components of the signal
- Details: low-scale, high-frequency components

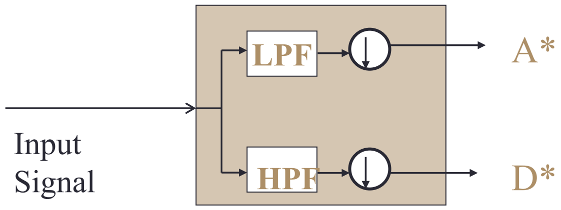


Decimation

- The former process produces twice the data it began with: N input samples produce N approximations coefficients and N detail coefficients.
- To correct this, we *Down sample* (or: *Decimate)* the filter output by two, by simply throwing away every second coefficient.

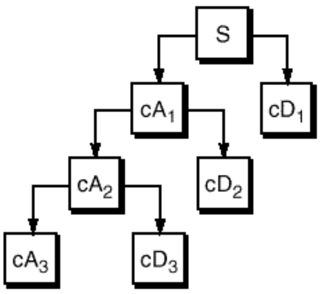
Decimation (cont'd)

So, a complete one stage block looks like:

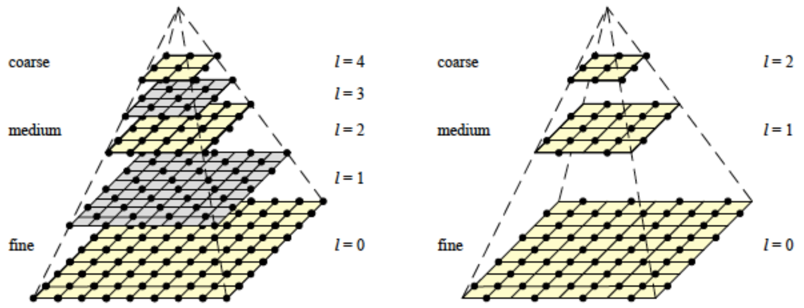


Multi-level Decomposition

- Iterating the decomposition process, breaks the input signal into many lower-resolution components: *Wavelet decomposition tree*:

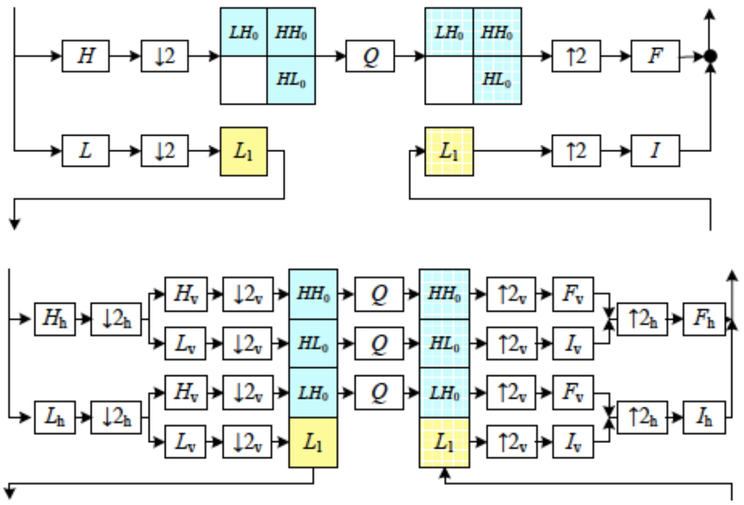


Wavelets and Pyramid

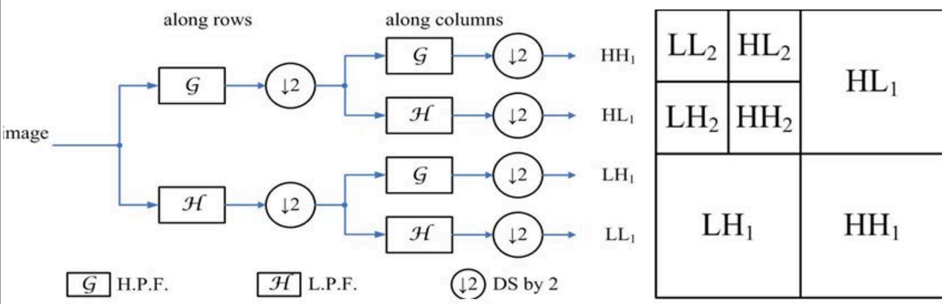


pyramid—each wavelet level stores 3/4
of the original pixels (usually the horizontal, vertical, and mixed
number of wavelet coefficients and original pixels is the same.

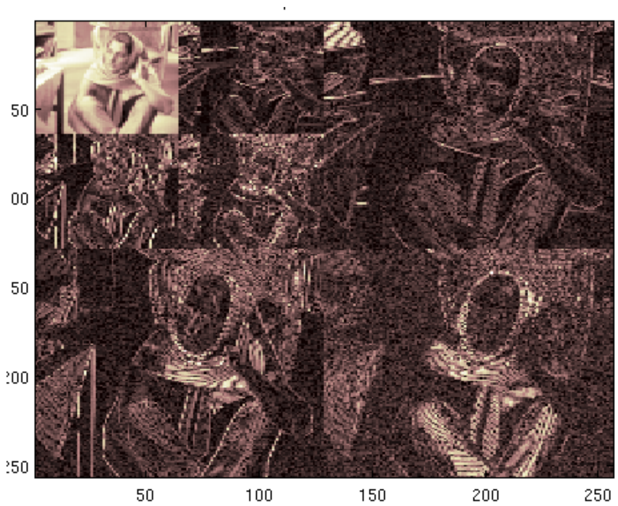
2D Wavelet Decomposition

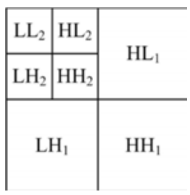


2D Wavelet Decomposition

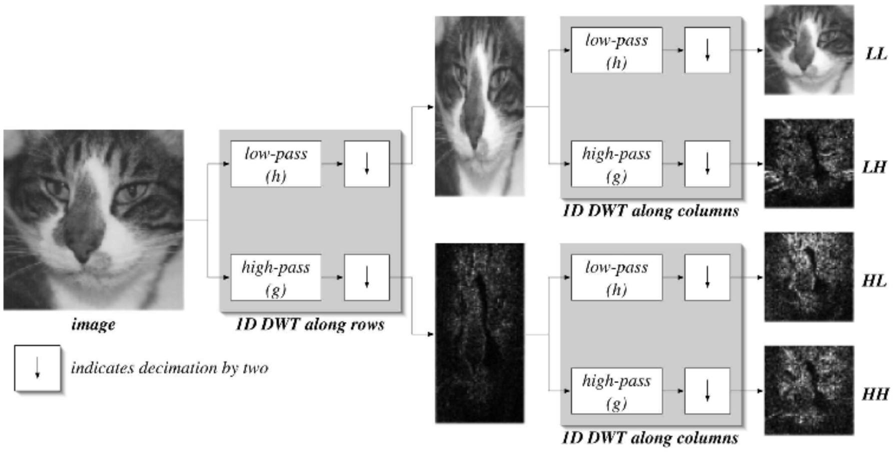


2D Wavelet transform

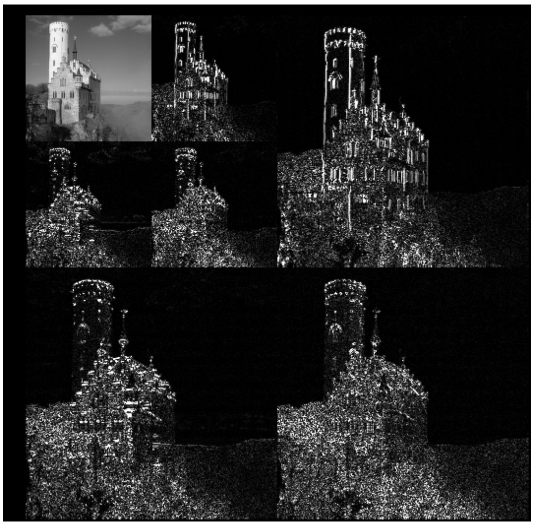




2D Wavelet transform



2D Wavelet transform – Jpeg2000



Orthogonality

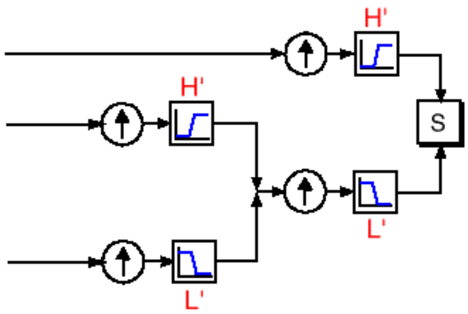
- For 2 vectors $$\langle v, w \rangle = \sum_n v_n w_n * = 0$$
- For 2 functions $$\langle f(t), g(t) \rangle = \int_a^b f(t) g*(t) dt = 0$$

Why wavelets have orthogonal base ?

- It easier calculation.
- When we decompose some image and calculating zero level decomposition we have accurate values .
- Scalar multiplication with other base function equals zero.

Wavelet reconstruction

- Reconstruction (or synthesis) is the process in which we assemble all components back



Up sampling (or interpolation) is done by zero inserting between every two coefficients

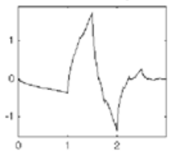
Wavelets like filters

Relationship of Filters to Wavelet Shape

- Choosing the correct filter is most important.
- The choice of the filter determines the shape of the wavelet we use to perform the analysis.

Example

- A low-pass reconstruction filter (L') for the db2 wavelet:



The filter coefficients (obtained by Matlab *dbaux* command:

0.3415 0.5915 0.1585 -0.0915

reversing the order of this vector and multiply every second coefficient by -1 we get the high-pass filter H':

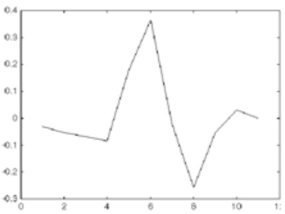
-0.0915 -0.1585 0.5915 -0.3415

Example (Cont'd)

- Now we up-sample the H' coefficient vector:

-0.0915 0 -0.1585 0 0.5915 0 -0.3415 0

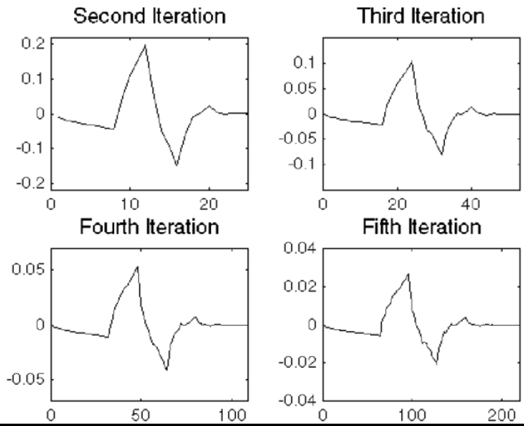
- and Convolving the up-sampled vector with the original low-pass filter we get:



Example (Cont'd)

- Now iterate this process several more times, repeatedly up-sampling and convolving the resultant vector

with the original low-pass filter, a *pattern* begins to emerge:



Example: Conclusion

https://www.mathworks.com/examples/wavelet

- The curve begins to look more like the *db2* wavelet: the wavelet shape is determined entirely by the coefficient Of the reconstruction filter
- You can't choose an arbitrary wavelet waveform if you want to be able to reconstruct the original signal accurately!

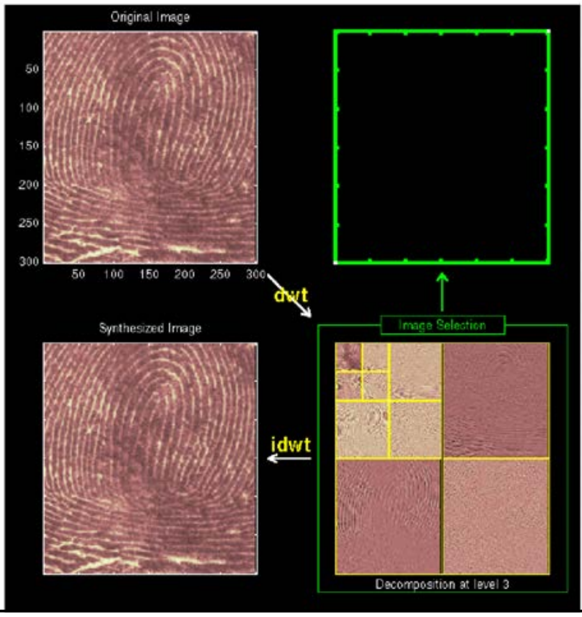
Wavelets in Matlab

https://www.mathworks.com/examples/wavelet

Compression Example

- A two dimensional (image) compression, using 2D wavelets analysis.
- The image is a Fingerprint.
- FBI uses a wavelet technique to compress its fingerprints database.

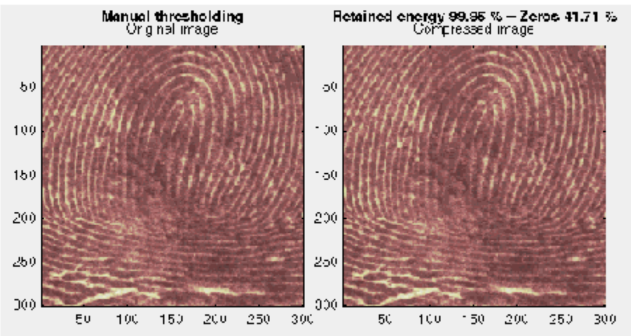
Fingerprint compression



Wavelet: Haar
Level:3

Results

Original Image Compressed Image



Threshold: 3.5
Zeros: 42%
Retained energy: 99.95%

Next Class

More Transforms; More on Transforms