Intro to image transforms

C306

Martin Jagersand

Image point operations

- So far we have considered operations on one image point, ie contrast stretching, histogram eq. Etc

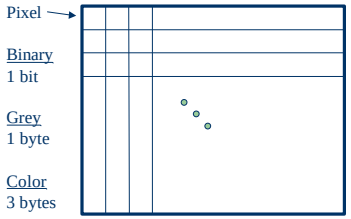
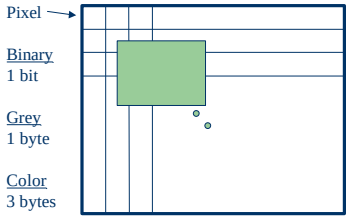


Image transforms

- By contrast, image transforms are defined over regions of an image or the whole image



Examples: Image transforms

- Fourier transform
  - For analysis of frequency spectrum in a image
- Cosine, sine transform
  - Compression, e.g. jpeg, mpeg
- Hotelling, KL, PCA
  - Statistically optimal basis. Used in compression, recognition and image variability representation

Image transforms

- Image transforms are some of the most important methods in image processing
- Will be used later in:
  1. Filtering
  2. Restoration
  3. Enhancement
  4. Compression
  5. Image analysis

Mathematically

- Learning the math can be daunting…

$$\begin{bmatrix} \mathrm{F}(0) \\ \mathrm{F}(2) \\ \vdots \\ \mathrm{F}(N-1) \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \mathrm{e}^{-\frac{\mathrm{j}2\pi}{N}} & & \mathrm{e}^{-\frac{\mathrm{j}2\pi(N-1)}{N}} \\ \vdots & & \ddots & \vdots \\ 1 & \mathrm{e}^{-\frac{\mathrm{j}2\pi(N-1)}{N}} & & \mathrm{e}^{-\frac{\mathrm{j}2\pi(N-1)^2}{N}} \end{bmatrix} \begin{bmatrix} \mathrm{f}(0) \\ \mathrm{f}(2) \\ \vdots \\ \mathrm{f}(N-1) \end{bmatrix}$$

$$\mathrm{F}(\mathbf{u}) = \frac{1}{\sqrt{N}} \sum_{\mathrm{x}=0}^{N-1} \mathrm{f}(\mathrm{x}) \mathrm{e}^{-\frac{\mathrm{j}2\pi \mathrm{ux}}{N}}$$

- Note: u=Ax form! $\mathrm{A}^{\mathrm{T}*}$

Vector spaces

- We will start with the concepts of
  1. vector spaces and
  2. coordinate changes,

just as used in the camera models:

Examples:

- Euclidean world space $= \Re^3, \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
- Image space, (q by q image): $= \Re^{q^2}, \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1q} \\ a_{21} & & & \\ \vdots & & \ddots & \\ a_{q1} & & & a_{qq} \end{bmatrix} \leftrightarrow \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a^{q^2} \end{bmatrix}$

Basis (of a vector space)

- A member can be described as a linear comb

$$\mathbf{x} = a\mathbf{e_1} + a\mathbf{e}_2 + \ldots + a\mathbf{e_m}$$

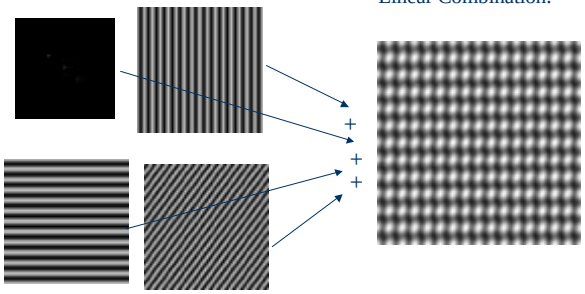
- Example: 3D vectors

- Example: Image  = a  +b  +c 

Constructing an image

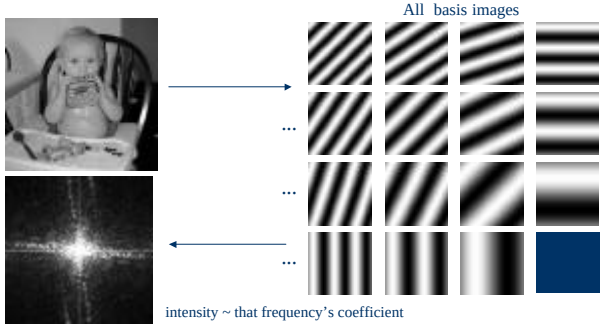
Basis vectors

Linear Combination:



Analyzing an image

All basis images



intensity ~ that frequency's coefficient

Vector spaces
Important properties from lin alg

1. Product
   1. With scalar
   2. "dot" product
   3. Vector product (for 3-vectors)
2. vector sum, difference
3. Norm
4. Orthogonality
5. Basis

See Shapiro-Stockman p. 178-181

Basic definitions for a vector space with a defined vector length.

Let U and V be any two vectors; u_i and v_i be real numbers denoting the coordinates of those vectors; and let a, b, c, etc. be any real numbers denoting scalars.

48 DEFINITION *For vectors* $U = [u_1, u_2, ..., u_n]$ *and* $V = [v_1, v_2, ..., v_n]$ *their* **vector sum** *is the vector* $U \oplus V = [u_1 + v_1, u_2 + v_2, ..., u_n + v_n]$.

49 DEFINITION *For vector* $V = [v_1, v_2, ..., v_n]$ *and real number (scalar)* a *the* **product of the vector and scalar** *is the vector* $aV = [av_1, av_2, ..., av_n]$.

50 DEFINITION *For vectors* $U = [u_1, u_2, ..., u_n]$ *and* $V = [v_1, v_2, ..., v_n]$ *their* **dot product, or scalar product** *is the real number* $U \circ V = u_1v_1 + u_2v_2 + ... + u_nv_n$.

51 DEFINITION *For vector* $V = [v_1, v_2, ..., v_n]$ *its* **length, or norm,** *is the non-negative real number* $\| V \| = V \circ V = (v_1v_1 + v_2v_2 + ... + v_nv_n)^{1/2}$.

52 DEFINITION *Vectors* U *and* V *are* **orthogonal** *if and only if* $U \circ V = 0$.

53 DEFINITION *The* **distance between vectors** $U = [u_1, u_2, ..., u_n]$ *and* $V = [v_1, v_2, ...v_n]$ *is the length of their difference* $d(U, V) = \| U - V \|$.

54 DEFINITION *A* **basis** *for a vector space of dimension* n *is a set of* n *vectors* $\{ w_1, w_2, ..., w_n \}$ *that are independent and that span the vector space. The spanning property means that any vector* V *can be expressed as a linear combination of basis vectors:* $V = a_1w_1 \oplus a_2w_2 \oplus ... \oplus a_nw_n$. *The independence property means that none of the basis vectors* w_i *can be represented as a linear combination of the others.*

Properties of vector spaces that follow from the above definitions.

1. $U \oplus V = V \oplus U$
2. $U \oplus (V \oplus W) = (U \oplus V) \oplus W$
3. There is a vector O such that for all vectors V, $O \oplus V = V$
4. For every vector V, there is a vector $(-1)V$ such that $V \oplus (-1)V = O$
5. For any scalars a, b and any vector V, $a(bV) = (ab)V$
6. For any scalars a, b and any vector V, $(a+b)V = aV \oplus bV$
7. For any scalar a and any vectors U, V, $a(U \oplus V) = aU \oplus aV$
8. For any vector V, $1V = V$
9. For any vector V, $(-1V) \circ V = -\| V \|^2$

Important properites

Cachy-Schwarz inequality:

For any two nonzero vectors U and V, $-1 \leq \frac{U \circ V}{\| U \| \ \| V \|} \leq +1$ (5.24)

55 DEFINITION *Let U and V be any two nonzero vectors, then the* **normalized dot product** *of U and V is defined as* $(\frac{U \circ V}{\| U \| \ \| V \|})$

56 DEFINITION *Let U and V be any two nonzero vectors, then the* **angle between** *U and V is defined as* $\cos^{-1}(\frac{U \circ V}{\| U \| \ \| V \|})$

Finding local image properties:

Roberts basis: $W_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ $W_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ $W_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$ $W_4 = \frac{1}{2}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$

Constant region: $\begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} = \frac{20}{2}(\frac{1}{5}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}) = 10W_1 \oplus 0W_2 \oplus 0W_3 \oplus 0W_4$

Step Edge: $\begin{bmatrix} -1 & +1 \\ -1 & +1 \end{bmatrix} = 0W_1 \oplus \frac{2}{\sqrt{2}}W_2 \oplus \frac{2}{\sqrt{2}}W_3 \oplus 0W_4$

Step Edge: $\begin{bmatrix} +1 & +1 \\ -3 & +1 \end{bmatrix} = 0W_1 \oplus \frac{4}{\sqrt{2}}W_2 \oplus 0W_3 \oplus \frac{-8}{2}W_4$

Lines: $\begin{bmatrix} 0 & 8 \\ 8 & 0 \end{bmatrix} = 8W_1 \oplus 0W_2 \oplus 0W_3 \oplus 8W_4$

Roberts basis in matlab

- Basis:

```
W1 = 1/2*[
1 1
  1 1];

W2 = 1/sqrt(2)*[
0 1
-1 0];

W3 = 1/sqrt(2)*[
1 0
  0 -1];

W4 = 1/2*[
-1 1
   1 -1];
```

- Test region:

```
CR = [
5 5
  5 5];

% Coefficients:
c1 = sum(sum(W1.*CR))
% c1 = 10

c2 = sum(sum(W2.*CR))
% c2 = 0

% Do same for W3 and
W4
```

Better way to implement: Use matrix algebra!

- Construct basis matrix
```
B = [
W1(:)'
W2(:)'
W3(:)'
W4(:)']
```
- Flatten the test area:
```
CRf = CR(:)
```
- Compute coordinate change:
```
NewCoord = B*CRf
% = (10,0,0,0)'
```
- Test2: Step edge:
```
SE = [
-1 1
-1 1]
SEf = SE(:)

NewCoordSE = B*SEf
% =(0,1.41,-1.41,0)'
```
- Verify that we can transform back:
```
SEtestf =B'*NewCoordSE

reshape(SEtestf, 2, 2)
% SEtest =
%   -1.0000    1.0000
%   -1.0000    1.0000
```

Standard basis

- e1 e2 e3 ... • e8 e9

$$\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix}\ \begin{bmatrix}0&1&0\\0&0&0\\0&0&0\end{bmatrix}\ \begin{bmatrix}0&0&1\\0&0&0\\0&0&0\end{bmatrix}\ \cdots\ \begin{bmatrix}0&0&0\\0&0&0\\0&1&0\end{bmatrix}\ \begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}$$

Nine "standard" basis vectors for the space of all 3x3 matrices.

$$\begin{bmatrix}9&5&0\\5&0&0\\0&0&0\end{bmatrix} = 9\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} + 5\begin{bmatrix}0&1&0\\0&0&0\\0&0&0\end{bmatrix} + 5\begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix}$$

= 9*e1 + 5*e2 + 5*e4

Frei-Chen basis

gradient: $W_1 = 1/\sqrt{8}\begin{bmatrix}1&\sqrt{2}&1\\0&0&0\\-1&-\sqrt{2}&-1\end{bmatrix}$ $W_2 = 1/\sqrt{8}\begin{bmatrix}1&0&-1\\\sqrt{2}&0&-\sqrt{2}\\1&0&-1\end{bmatrix}$

ripple: $W_3 = 1/\sqrt{8}\begin{bmatrix}0&-1&\sqrt{2}\\1&0&-1\\-\sqrt{2}&1&0\end{bmatrix}$ $W_4 = 1/\sqrt{8}\begin{bmatrix}\sqrt{2}&-1&0\\-1&0&1\\0&1&-\sqrt{2}\end{bmatrix}$

line: $W_5 = 1/2\begin{bmatrix}0&1&0\\-1&0&-1\\0&1&0\end{bmatrix}$ $W_6 = 1/2\begin{bmatrix}-1&0&1\\0&0&0\\1&0&-1\end{bmatrix}$

Laplacian: $W_7 = 1/6\begin{bmatrix}1&-2&1\\-2&4&-2\\1&-2&1\end{bmatrix}$ $W_8 = 1/6\begin{bmatrix}-2&1&-2\\1&4&1\\-2&1&-2\end{bmatrix}$

constant: $W_9 = 1/3\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

Frei

Example of representing an intensity neighborhood using the Frei-Chen basis.

Consider the intensity neighborhood $N = \begin{bmatrix}10&10&10\\10&10&5\\10&5&5\end{bmatrix}$

We find the component of this vector along each basis vector using the dot product as before. Since the basis is orthonormal, the total image energy is just the sum of the component energies and the structure of N can be interpreted in terms of the components.

$N \circ W_1 = \frac{5+5\sqrt{2}}{\sqrt{8}} \approx 4.3$; $energy \approx 18$

$N \circ W_2 = \frac{5+5\sqrt{2}}{\sqrt{8}} \approx 4.3$; $energy \approx 18$

$N \circ W_3 = 0$; $energy = 0$

$N \circ W_4 = \frac{5\sqrt{2}-10}{\sqrt{8}} \approx -1$; $energy \approx 1$

$N \circ W_5 = 0$; $energy = 0$

$N \circ W_6 = 2.5$; $energy \approx 6$

$N \circ W_7 = 2.5$; $energy \approx 6$

$N \circ W_8 = 0$; $energy = 0$

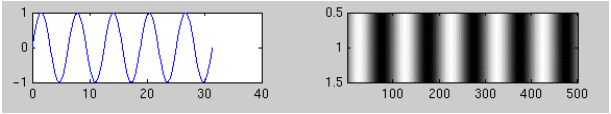
$N \circ W_9 = 25$; $energy = 625$

The total energy in N is $\mathbf{N} \circ \mathbf{N} = 675$, 625 of which is explained just by the

Sine function

```
t = 0:pi/50:10*pi
I = sin(t)
subplot(221)
plot(t,I)
```

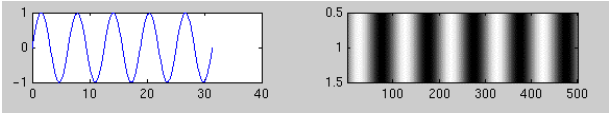
- `Image(I) ???`
- `subplot(222)`
  `image(64*(I+1)/2)`
  `colormap gray`



Sine function

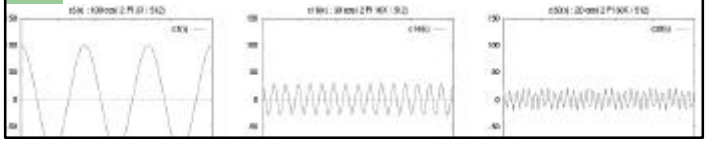
```
t = 0:pi/50:10*pi
I = sin(t)
subplot(221)
plot(t,I)
```

- `Image(I) ???`
- `subplot(222)`
  `image(64*(I+1)/2)`
  `colormap gray`

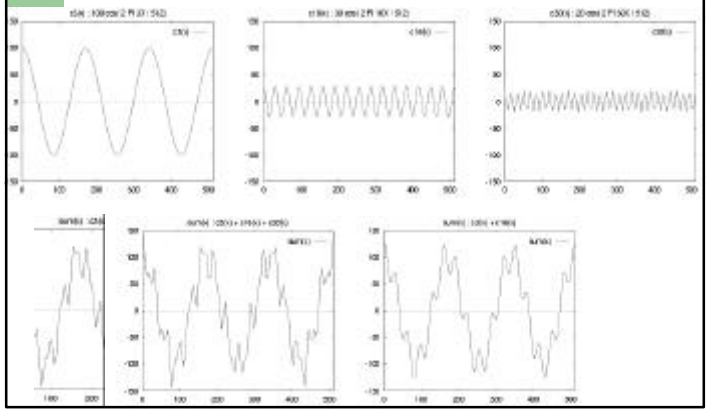


Adding sines

Adding sines



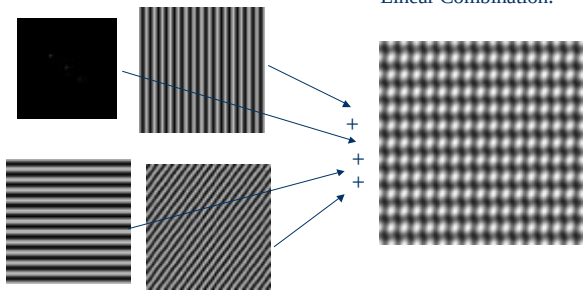
Adding sines



Adding sines in 2D images:

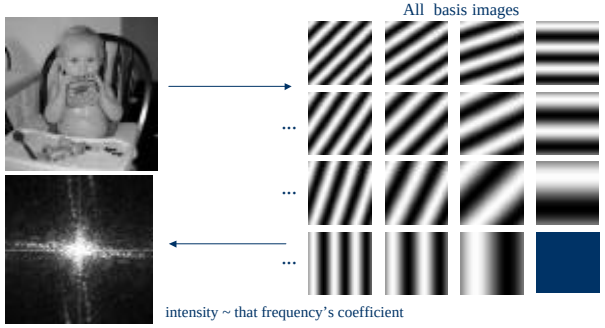
Basis vectors

Linear Combination:



Analyzing an image using sines and cosines

All basis images



intensity ~ that frequency's coefficient

Cosine Transforms:

- Used in jpeg, mpeg compression.
- These are the most common compression techniques.
- Example: Jpeg compression:

1. Image -> 8x8 blocks
2. Cos trans each block
3. Quantize the transform coefficients
4. Code non-zero coefficients

Definition, Gonzalez
Discrete Cosine Transform, DCT

- Forward Transform:

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

- Inverse:

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

- Where:

$$\alpha(u) = \begin{cases} 1/\sqrt{N}, & \text{for } u = 0 \\ 2/\sqrt{2N}, & \text{for } u \geqslant 1 \end{cases}$$

Definition, matrix form:

- Let C=Af
- f=A'C ('= transpose), where:

$$A = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \sqrt{2}\cos(3\pi/2N) & \cdots & \sqrt{2}\cos((2N-1)\pi/2N) \\ \vdots & & \ddots & \vdots \\ 1 & \sqrt{2}\cos(3(N-1)\pi/2N) & \cdots & \sqrt{2}\cos((2N^2-3N+1)\pi/2N) \end{pmatrix}$$

Example
4x4 Cosine transform

- Cosine basis, Test "image"

$$\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Transformed Im

- Why [2, 0, 0, 0]' ???

Why??
Transforms graphically

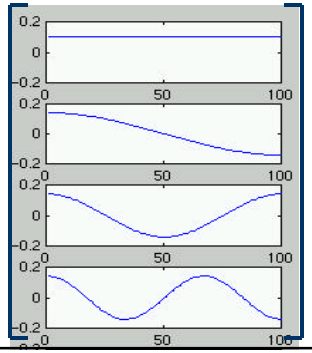
- Image – Transform pair 1

$$\begin{bmatrix} 1 \\ 0.5 \\ -0.5 \\ -1 \end{bmatrix} \sim \begin{bmatrix} 0 \\ 1.58 \\ 0 \\ -0.11 \end{bmatrix}$$

- Image – Transform pair 2

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \sim \begin{bmatrix} 5 \\ -2.23 \\ 0 \\ -0.16 \end{bmatrix}$$

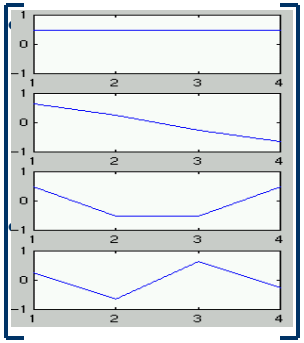
Why??
Transforms graphically



$$\begin{bmatrix} 1 \\ 0.5 \\ -0.5 \\ -1 \end{bmatrix} \sim \begin{bmatrix} 0 \\ 1.58 \\ 0 \\ -0.11 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \sim \begin{bmatrix} 5 \\ -2.23 \\ 0 \\ -0.16 \end{bmatrix}$$

Why??
Transforms graphically

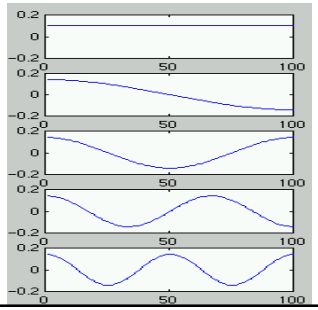


$$\begin{bmatrix} 1 \\ 0.5 \\ -0.5 \\ -1 \end{bmatrix} \sim \begin{bmatrix} 0 \\ 1.58 \\ 0 \\ -0.11 \end{bmatrix}$$

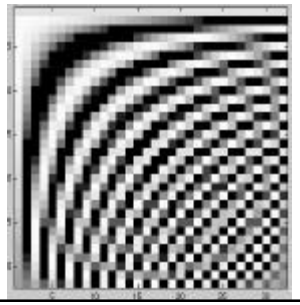
$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \sim \begin{bmatrix} 5 \\ -2.23 \\ 0 \\ -0.16 \end{bmatrix}$$

Cosine bases

- First 5 vectors:



- Image of full 32x32:



Properties of the cosine transform

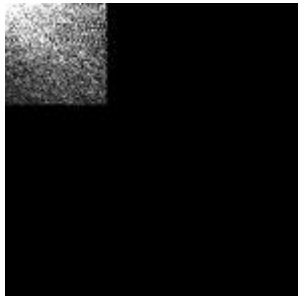
- Real and orthogonal, ie $A^{-1} = A^T$
- Not the real part of the DFT, but related (try type dct in matlab)
- Can be computed in O(NlogN) time
- "Compacts" the energy in images into a few coefficients. (Compression)

Examle:
Energy compaction

- Original Lena image
- 2D DCT


Example:
Cropping in image and DCT:

- Cropped Lena
- Cropped DCT

Comparison
Image recovery from cropped DCT

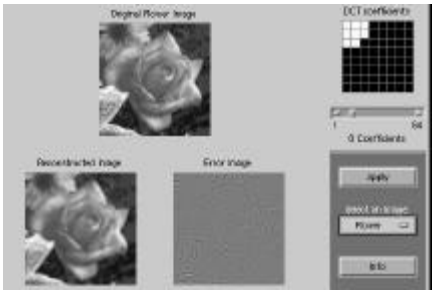
- Original image
- Image recovered from cropped DCT


Example
Image recovery from cropped DCT

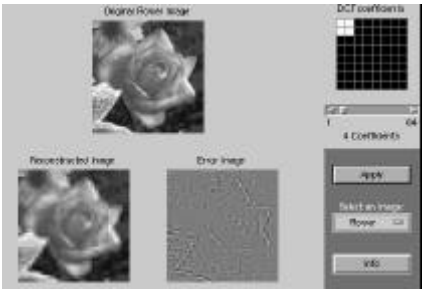
- Cropped image
- Recovered image


Block coding
Example, 8 coefficients

- Block coded image (8x8 pixel blocks)



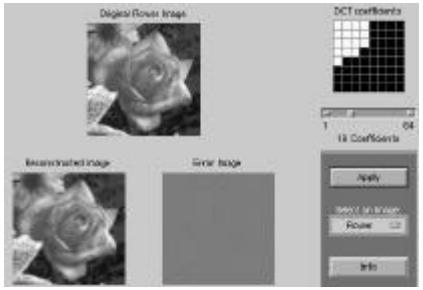
Example 2, using only 4 coeff



- Notice “grainy” result from using only low frequency coeff

Example 3

- With 16 of 64 coefficients almost perfect:



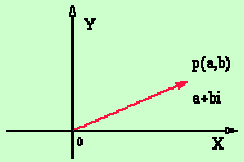
The continuous Fourier Transform

- Today:
- Introduction to complex numbers
- Continuous Fourier Transform

Complex Numbers- Definition

Imaginary unit: $i^2 = -1$

Rectangular form:



$x = \mathrm{Re}(z)$ – *real part*

$y = \mathrm{Im}(z)$ – *imaginary part*

Complex Numbers - Properties

Conjugate: $\bar{z} = x - iy$

Modulus: $|z| = \sqrt{x^2 + y^2}$

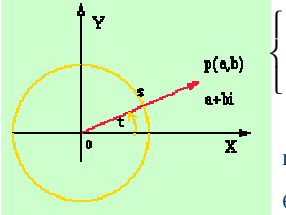
Addition $z_1 + z_2 = (x_1 + iy_1) + (x_2 + iy_2)$

$= (x_1 + x_2) + i(y_1 + y_2)$

Multiplication $z_1 z_2 = (x_1 + iy_1)(x_2 + iy_2)$

$= (x_1 x_2 + y_1 y_2) + i(x_1 y_2 + x_2 y_1)$

Polar Form



$\begin{cases} x = r\cos q \\ y = r\sin q \end{cases}$ $\begin{cases} r = \sqrt{x^2 + y^2} \\ \tan q = \dfrac{y}{x} \end{cases}$

r – modulus

θ - argument

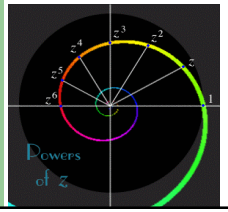
$z = x + iy = r(\cos q + i\sin q) = re^{iq}$

Polar form - Properties

Conjugate: $|z| = re^{-iq}$

Multiplication: $z_1 z_2 = r_1 e^{iq_1} r_2 e^{iq_2} = r_1 r_2 e^{i(q_1 + q_2)}$

Power: $z^n = \left(re^{iq}\right)^n = r^n e^{inq}$



Polar form - Formulas

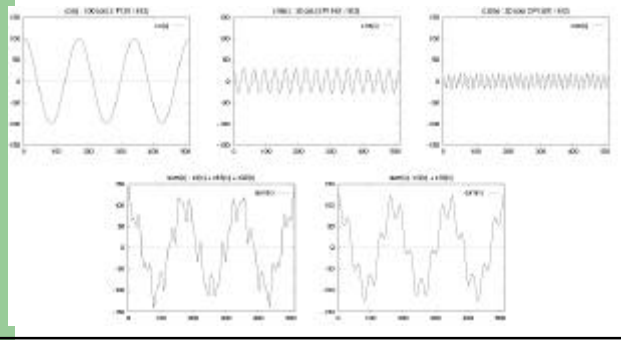
- De Moivre's Formula

$$(\cos q + i\sin q)^n = \cos nq + i\sin nq$$

- Euler Formula

$$\cos q = \frac{e^{iq} + e^{-iq}}{2}, \quad \sin q = \frac{e^{iq} - e^{-iq}}{2}$$

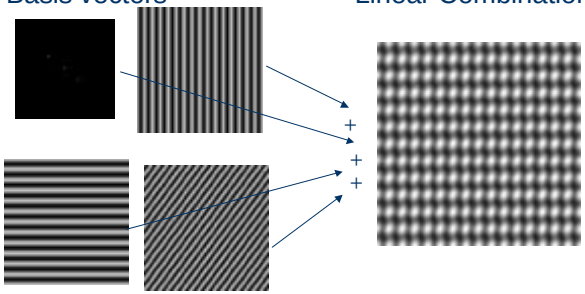
Combination of sines



Constructing an image

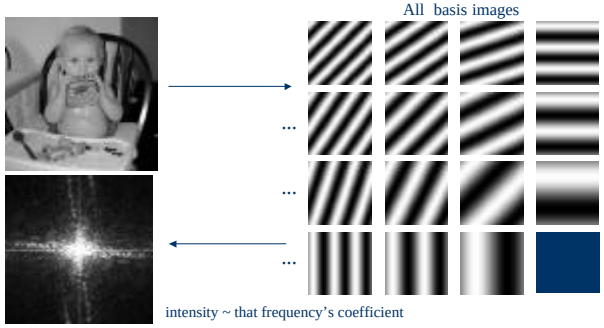
Basis vectors

Linear Combination



Analyzing an image

All basis images



intensity ~ that frequency's coefficient

Till now …

Im = a_1 I_1 + a_2 I_2 + … + a_n I_n

Real Basis

Im can be recovered from **a** if **I** invertible

Fourier transform = real and imaginary part

Fourier Transform - definition

Direct:

$$\Im\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi ux}dx =$$

$$= \int_{-\infty}^{\infty} f(x)(\cos 2\pi ux - i\sin 2\pi ux)dx =$$

$$= \int_{-\infty}^{\infty} f(x)\cos 2\pi uxdx - i\int_{-\infty}^{\infty} f(x)\sin 2\pi uxdx$$

even odd

Oddness and Evenness

Any function … $f(x) = O(x) + E(x)$

Where $O(x)$ is an odd function and $E(x)$ is an even function

$$E(x) = \frac{1}{2}(f(x) + f(-x))$$

$$O(x) = \frac{1}{2}(f(x) - f(-x))$$

The Fourier transform …

$$\Im\{f(x)\} = 2\int_{\infty} E(x)\cos(2\pi ux)dx - 2i\int_{\infty} O(x)\sin(2\pi ux)dx$$

FT of a **real even** function is **real** (and even)

real odd function is **imaginary** (and odd)

FT- spectrum and phase

F(u) = R(u) + i I(u)

- Spectrum $|F(u)| = \sqrt{R^2(u) + I^2(u)}$
- Phase $\Phi(u) = \text{atan}\frac{I(u)}{R(u)}$

$$F(u) = R(u)e^{i\Phi(u)}$$

Inverse FT

$$\Im^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u)e^{i2pux}du$$

Convolution

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du$$

Convolution theorem

$$\Im\{f_1(x) * f_2(x)\} = \Im\{f_1(x)\}\Im\{f_2(x)\} = F_1(u)F_2(u)$$

Product theorem

$$\Im\{f_1(x)\, f_2(x)\} = \Im\{f_1(x)\} * \Im\{f_2(x)\} = F_1(u) * F_2(u)$$

Obs: Discrete case – FFT + convolution theorem = fast convolution

Properties

	f(x)	F(u)
• Linearity	$af_1(x) + bf_2(x)$	$aF_1(u) + bF_2(u)$
• Time shifting	$f(x - x_0)$	$F(u)e^{-2piux_0}$
• Frequency shifting	$f(x)e^{-2piux}$	$F(u - u_0)$
• Scaling	$f(ax)$	$\lvert a\rvert^{-1}F(u/a)$

The Discrete Fourier transform

Continuous Ft: Integral over continuous function

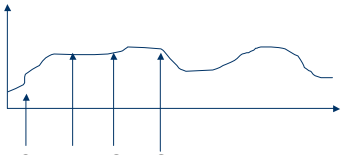
$$\mathrm{F}(u) = \int_{-\infty}^{\infty} \mathrm{f}(x)\mathrm{e}^{-\mathrm{j}2\pi ux}\, dx \qquad \mathrm{f}(x) = \int_{-\infty}^{\infty} \mathrm{F}(u)\mathrm{e}^{\mathrm{j}2\pi ux}\, du$$

But images and other digital data is represented as discrete vectors or matrices

Today: Develop a discrete version of the F-t

Discrete functions

- Continuous function: f(x)



- Discretized at t = 0, 1, 2, 3,…
- (f_0, f_1, f_2, f_3, …)

Discrete Fourier Transform (DFT)

Forward transform:

$$F(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{-j\frac{2\pi ux}{N}}$$

Reverse transform:

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} F(u) e^{j\frac{2\pi ux}{N}}$$

DFT in matrix form

$$\begin{bmatrix} F(0) \\ F(2) \\ \vdots \\ F(N-1) \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{j2\pi}{N}} & & e^{-\frac{j2\pi(N-1)}{N}} \\ \vdots & & \ddots & \vdots \\ 1 & e^{-\frac{j2\pi(N-1)}{N}} & & e^{-\frac{j2\pi(N-1)^2}{N}} \end{bmatrix} \begin{bmatrix} f(0) \\ f(2) \\ \vdots \\ f(N-1) \end{bmatrix}$$

$$F(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{-\frac{j2\pi ux}{N}}$$

- Note: u=Ax form! Inverse: A^{T*}

Compactly written

Let: $\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ N-1 \end{bmatrix}$

A can be written: $A = \frac{1}{\sqrt{N}} \exp(-j\frac{2\pi \mathbf{b}^T \mathbf{b}}{N})$

Note: pointwise

Gonzalez 3.2-2: F=Af

3.2-3: $f=A^{T*}F$

For images: 2-Dim DFT

- Gonzales 3.2-9:

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-j\frac{2\pi}{N}(ux+vy)}$$

- Written with the A-matrix:

Forward $F = AfA^T$

Inverse $f = A^{T*}fA^*$

2-Dim DFT separates

- Notice: We can separate sums over x and y:

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-j\frac{2\pi}{N}(ux+vy)}$$

$$= \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1} e^{-j\frac{2\pi ux}{N}} \left(\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} e^{-j\frac{2\pi vy}{N}} f(x)\right)$$

- With respect to: x y

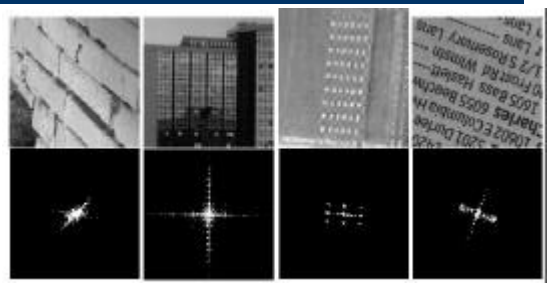
Same as $F = AfA^T$ if f 2D image

Displaying the Fourier Transform

- Practically the Fourier transform coefficients have a large range
- Usually the magnitude |F(u,v)| is what we want to see
- Compress range variation to see as an image:

D(i,j) = c*log(1+|F(u,v)|)

Manmade object images:

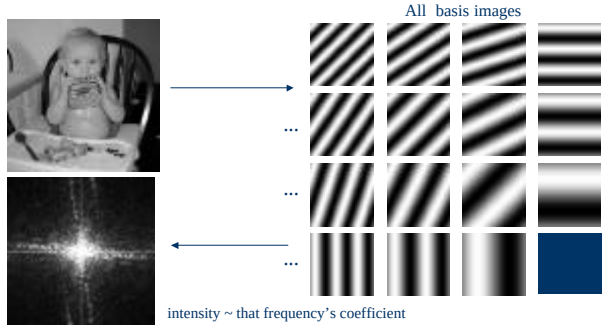


- What does the F-t express?

Periodicity

- The fourier transform highlights/identifies periodic structure in images.
- This is seen as peaks (usually several) along the same axis as the repetition in the original image

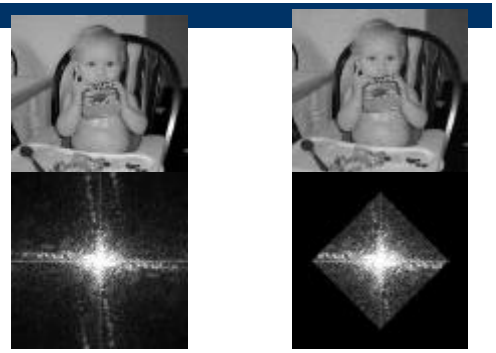
What does the F-t express?



Frequency spectrum

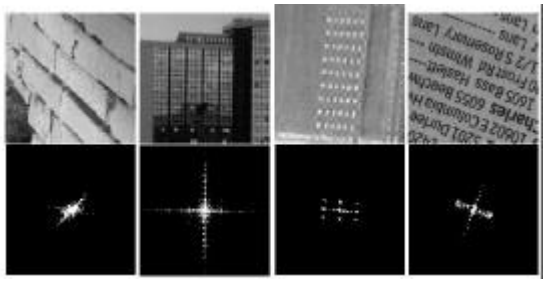
- By convention the lowest frequency components are shifted to the center of the image. (use fftshift in matlab)
- Fourier transform coefficients tells us the relative composition of low and high spatial frequencies, I.e. the frequency spectrum
- Usually more lows than highs

Cropping in Fourier space



Data Reduction: only use *some* of the existing frequencies

Properties: Rotation



- Notice direction of F-t:

Properties: Rotation

- Rotating f(x,y) by an angle θ_0 rotates F(u,v) by the same angle
- expressing the variables in polar coordinates

$$x = r\cos\theta,\ y = r\sin\theta,\ u = w\cos\phi,\ v = w\sin\phi$$

- substitution leads to

$$f(r,\theta+\theta_0) \leftrightarrow F(w,\phi+\theta_0)$$

Properties: Translation

$$f(x,y)e^{j2\pi(u_0x+v_0y)/N} \leftrightarrow F(u-u_0,v-v_0)$$

$$f(x-x_0,y-y_0) \leftrightarrow F(u,v)e^{-j2\pi(ux_0+vy_0)/N}$$

- the magnitude is unaffected because

$$\left|F(u,v)e^{-j2\pi(ux_0+vy_0)/N}\right| = \left|F(u,v)\right|$$

Properties: Distributivity

- the Fourier transform and its inverse are distributive over addition but not over multiplication:

$$\mathrm{F}\{f_1(x,y)+f_2(x,y)\} = \mathrm{F}\{f_1(x,y)\}+\mathrm{F}\{f_2(x,y)\}$$

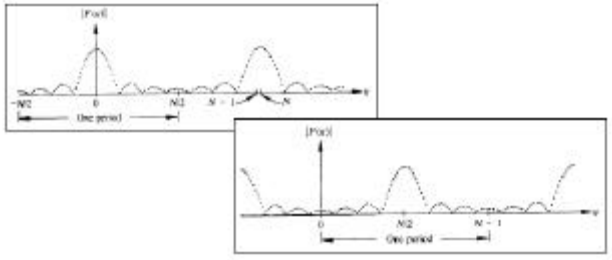
- and

$$\mathrm{F}\{f_1(x,y)*f_2(x,y)\} \neq \mathrm{F}\{f_1(x,y)\}*\mathrm{F}\{f_2(x,y)\}$$

Properties: Periodicity

- discrete Fourier transform and inverse are periodic with period N

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N)$$



Slow Fourier Transform: DFT in matrix form

$$\begin{bmatrix} \mathrm{F}(0) \\ \mathrm{F}(2) \\ \vdots \\ \mathrm{F}(N-1) \end{bmatrix} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \mathrm{e}^{-\frac{\mathrm{j}2\pi}{N}} & & \mathrm{e}^{-\frac{\mathrm{j}2\pi(N-1)}{N}} \\ \vdots & & \ddots & \vdots \\ 1 & \mathrm{e}^{-\frac{\mathrm{j}2\pi(N-1)}{N}} & & \mathrm{e}^{-\frac{\mathrm{j}2\pi(N-1)^2}{N}} \end{bmatrix} \begin{bmatrix} \mathrm{f}(0) \\ \mathrm{f}(2) \\ \vdots \\ \mathrm{f}(N-1) \end{bmatrix}$$

$$\mathrm{F}(\mathbf{u}) = \frac{1}{\sqrt{N}} \sum_{\mathrm{x}=0}^{N-1} \mathrm{f}(\mathrm{x}) \mathrm{e}^{-\frac{\mathrm{j}2\pi \mathrm{ux}}{N}}$$

- In general O(n^2)

Fast Fourier Transform, FFT

- Recall DFT:

$$\begin{aligned} F(u) &= \sum_{x=0}^{N-1} f(x)\, e^{-j\frac{2}{N}\pi ux} \\ &= \sum_{x=0}^{N/2-1} f(2x)\, e^{-j\frac{2}{N}\pi u2x} + \sum_{x=0}^{N/2-1} f(2x+1)\, e^{-j\frac{2}{N}\pi u(2x+1)} \\ &= \sum_{x=0}^{N/2-1} f(2x)\, e^{-j\frac{2}{N/2}\pi ux} + e^{-j\frac{2}{N}\pi u} \sum_{x=0}^{N/2-1} f(2x+1)\, e^{-j\frac{2}{N/2}\pi u2x} \\ &= F^{\mathrm{Even}} + e^{-j\frac{2}{N}\pi u} F^{\mathrm{Odd}} \end{aligned}$$

DFT vs FFT time comparison

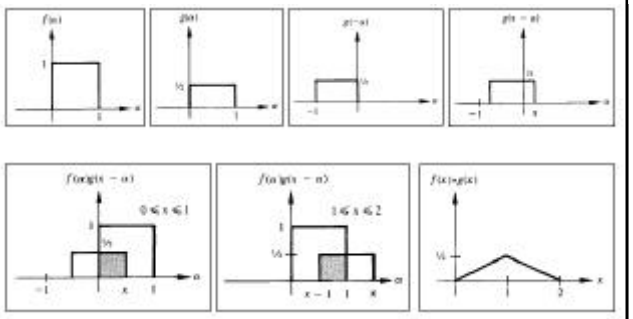
- DFT: F=Af
- O(n^2) time complexity

- FFT: F(0..255) = F(0,2,…,254)+eF(1,3,…,255)
- So F(256)->2xF(128)->4xF(64)->…->256xF(1)
- In general: O(nlogn) time complexity

Convolution

- Like an inner product, but with g reversed.
- Def: $f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x-\alpha) d\alpha$
  (Gonzalez 3.3.23)
- Theorem: $\Im(f * g) = FG$
- Convolution integral -> Point wise multiplication!!
- Discrete: FFT+Convolution theorem = fast convolutions.

Convolution example



Discrete convolution

- Gonzalez 3.3.29

$$f(x) * g(x) = \sum_{\alpha=0}^{M-1} f(\alpha) g(x-\alpha)$$

- Can also be written with a circulant matrix, e.g.:

$$\begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}1\\1\\1\end{bmatrix} = \begin{bmatrix}1&1&1&0&0\\0&1&1&1&0\\0&0&1&1&1\\1&0&0&1&1\\1&1&0&0&1\end{bmatrix} \begin{bmatrix}0\\0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\2\\3\\2\\1\end{bmatrix}$$

Convolution: Applications

- Models many physical processes like lens and diffraction distortion/degradation.
- Can also be used to “undo” degratation in image enhancement.
- Basic operation in shift invariant linear systems.
- (Used to describe generalized functions)

Practical discrete FFT-convolution

1. Let x=[1,2,3,2,1], y=[1,1,1]
2. Pad to: x=[0,1,2,3,2,1,0,0], y=[0,0,1,1,1,0,0,0]
3. Compute FFT-> X, Y
4. Compute pointwise product-> X.*Y=Z
5. Compute IFFT(Z)

(Result hopefully =[1,3,6,7,6,3,1,0])

- Note need n=2^k for FFT
- Pad with zeros to avoid overlap and get 2^k