Lecture 11 : Discrete Cosine Transform

Moving into the Frequency Domain

Frequency domains can be obtained through the transformation from one (**time** or **spatial**) domain to the other (**frequency**) via

- **Fourier Transform (FT)** (**see Lecture 3**) — **MPEG Audio**.
- **Discrete Cosine Transform (DCT)** (**new**) — Heart of **JPEG** and **MPEG Video**, **MPEG Audio**.

Note: We mention some image (and video) examples in this section with DCT (in particular) but also the FT is commonly applied to filter multimedia data.

External Link: MIT OCW 8.03 Lecture 11 Fourier Analysis Video

Recap: Fourier Transform

The tool which converts a spatial (real space) description of audio/image data into one in terms of its frequency components is called the **Fourier transform**.

The new version is usually referred to as the **Fourier space description** of the data.

We then essentially process the data:

- *E.g* . for **filtering** basically this means attenuating or setting certain frequencies to zero

We then need to convert data back to real audio/imagery to use in our applications.

The corresponding **inverse** transformation which turns a Fourier space description back into a real space one is called the **inverse Fourier transform**.

What do Frequencies Mean in an Image?

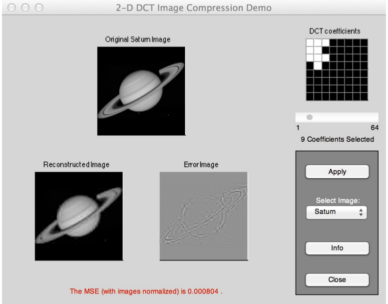
- Large values at **high** frequency components mean the data is changing rapidly on a short distance scale.
  E.g .: a page of small font text, brick wall, vegetation.
- Large **low** frequency components then the large scale features of the picture are more important.
  E.g . a single fairly simple object which occupies most of the image.

The Road to Compression

How do we achieve compression?

- Low pass filter — ignore high frequency noise components
  - Only store lower frequency components
- High pass filter — spot gradual changes
  - If changes are too low/slow — eye does not respond so ignore?

Low Pass Image Compression Example



MATLAB demo, dctdemo.m, (uses DCT) to

- Load an image
- **Low pass filter** in frequency (DCT) space
- **Tune** compression via a single slider value *n* to select coefficients
- **Inverse DCT**, **subtract input and filtered image** to see **compression artefacts**.

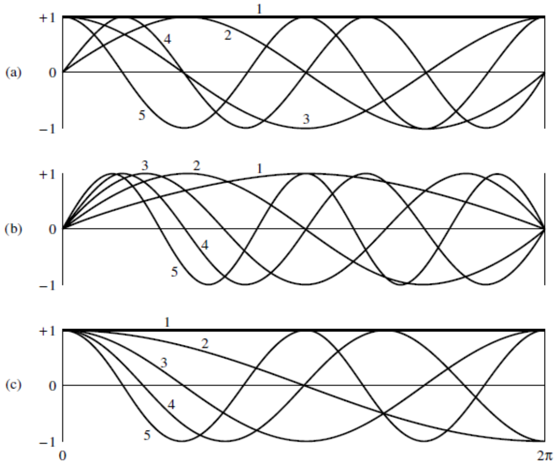
The Discrete Cosine Transform (DCT)

Relationship between DCT and FFT

DCT (Discrete Cosine Transform) is similar to the DFT since it decomposes a signal into a series of harmonic cosine functions. DCT is actually a **cut-down** version of the Fourier Transform or the Fast Fourier Transform (FFT):

- Only the **real** part of FFT (less data overheads).
- Computationally simpler than FFT.
- DCT— effective for multimedia compression (energy compaction).
- DCT **much** more commonly used (than FFT) in multimedia image/vi deo compression — **more later**.
- Cheap MPEG Audio variant — **more later**.
- FT captures more frequency “fidelity” (*e.g* . phase).

DCT vs FT



(a) Fourier transform, (b) Sine transform, (c) Cosine transform.

DCT Example

Let's consider a DC signal that is a constant 100,
i.e f(i) = 100 for i = 0 ...7 (DCT1Deg.m):

- So the domain is [0,7] for both i and u
  - We therefore have N=8 samples and will need to work 8 values for u=0. .. 7.

We can now see how we work out: F(u)

- As u varies we can work for each u, a component or *a basis* F(u).
- Within each F(u), we can work out the value for each $F_i(u)$ to define a *basis function*
- Basis function can be pre-computed and simply looked up in DCT computation.

1D DCT

For N data items 1D DCT is defined by:

$$F(u) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \Lambda(u) \cos\left[\frac{\pi u}{2\mathrm{N}}(2\mathrm{i}+1)\right] f(i)$$

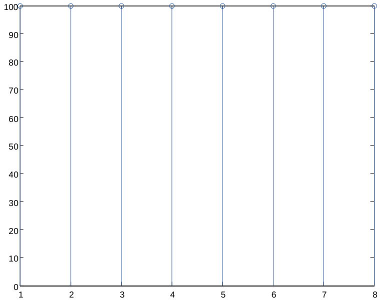
and the corresponding **inverse** 1D DCT transform is simply $F^{-1}(u)$, i.e.:

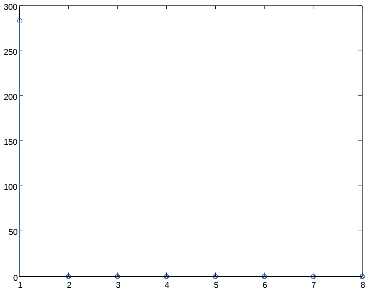
$$\begin{aligned} f(i) &= F^{-1}(u) \\ &= \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{u=0}^{N-1} \Lambda(u) \cos\left[\frac{\pi u}{2\mathrm{N}}(2\mathrm{i}+1)\right] f(u) \end{aligned}$$

where

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

Plots of f(I) and F(U)





f(i) = 100 for i=0. . .7

F(u): F(0)≈283, F(1...7)=0

DCT Example: F(0)

So for u = 0:

- Note: $\Lambda(0) = \frac{1}{2\sqrt{2}}$ and cos(0) = 1
- So F(0) is computed as:

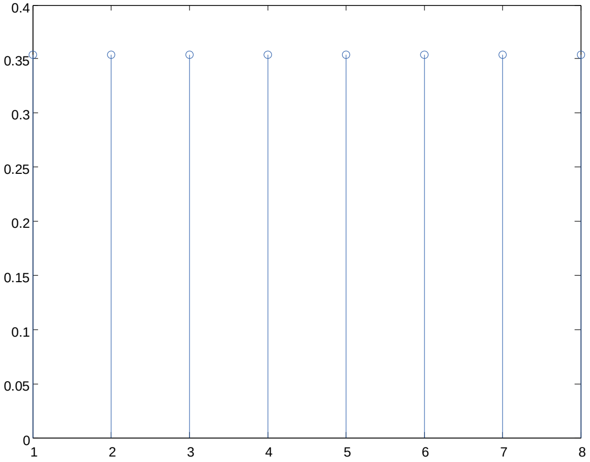
$$F(0) = \frac{1}{2\sqrt{2}}\,(1\cdot 100 + 1\cdot 100 + 1\times 100 + 1\cdot 100 + 1\cdot 100 + 1\cdot 100 + 1\cdot 100 + 1\cdot 100)$$

$$\approx 283$$

- Here the values $F_i(0) = \frac{1}{2\sqrt{2}}$ (i = 0. .. 7).

  These are bases of $F_i(0)$

F(0) Basis Function Plot



F(0) basis function

DCT Example: F(1 . . .7)

So for u = 1:
Note: $\Lambda(1) = 1$ and we have cos to work out: so F(1) is computed as:

$$\begin{aligned} F(1) &= \frac{1}{2}\left(\cos\frac{\pi}{16}\cdot 100+\cos\frac{3\pi}{16}\cdot 100+\cos\frac{5\pi}{16}\cdot 100+\cos\frac{7\pi}{16}\cdot 100\right. \\ &\quad \left.+\cos\frac{9\pi}{16}\cdot 100+\cos\frac{11\pi}{16}\cdot 100+\cos\frac{13\pi}{16}\cdot 100+\cos\frac{15\pi}{16}\cdot 100\right) \\ &= 0 \end{aligned}$$

(since $\cos\left(\frac{\pi}{16}\right)=-\cos\left(\frac{15\pi}{16}\right), \cos\left(\frac{3\pi}{16}\right)=-\cos\left(\frac{13\pi}{16}\right)$ *etc.*)

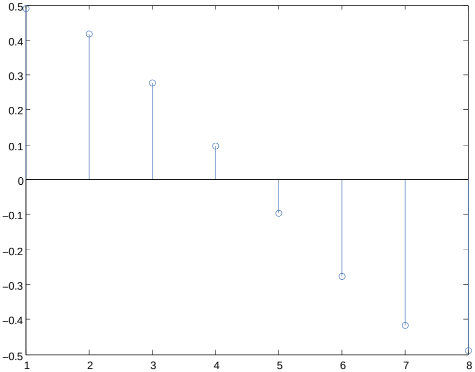
Here the values

$$F_i(1)=\left\{\frac{1}{2}\cos\left(\frac{\pi}{16}\right), \frac{1}{2}\cos\left(\frac{3\pi}{16}\right), \frac{1}{2}\cos\left(\frac{5\pi}{16}\right), \ldots, \frac{1}{2}\cos\left(\frac{11\pi}{16}\right), \frac{1}{2}\cos\left(\frac{13\pi}{16}\right), \frac{1}{2}\cos\left(\frac{15\pi}{16}\right)\right\}$$

form the basis function

F(2...7) similarly = 0

F(1) Basis Function Plot



F(1) basis function

Note:

- Bases are *reflected about centre* and *negated* since
  $\cos\left(\frac{\pi}{16}\right)=-\cos\left(\frac{15\pi}{16}\right), \cos\left(\frac{3\pi}{16}\right)=-\cos\left(\frac{13\pi}{16}\right)$ *etc.*
- **only** as our example function is a constant is F(1) zero.

DCT Matlab Example

DCT1Deg.m explained:

```
i = 1:8% dimension of vector
f(i) = 100 % set function
figure(1)    % plotf
stem(f);
%compute DCT
D =dct(f);
figure(2)    % plotD
stem(D);
```

- Create our function f, and plot it.
- Use MATLAB 1D dct function to compute DCT of f and plot it.

DCT Matlab Example

```
% Illustrate DCT bases compute DCT bases
% with dctmtx

bases =dctmtx(8);
% Plot bases:each row(j) of bases is the jth
%DCT Basis Function

for j= 1:8
figure %increment figure
stem(bases(j,:));     %plot rows
end
```

- MATLAB dctmtx function computes DCT basis functions.
- Each row j of bases is the basis function F(j).
- Plot each row.

DCT Matlab Example

```
% construct DCT from Basis Functions Simply
% multiply f' (column vector) by bases

D1 =bases*f';

figure(1)     % plot D1
stem(D1);
```

- Here we show how to compute the DCT from the basis functions.
- bases is an 8×8 matrix, f an 1×8 vector. Need column 8×1 form to do matrix multiplication so transpose f.

2D DCT

For a 2D N by M image 2D DCT is defined:

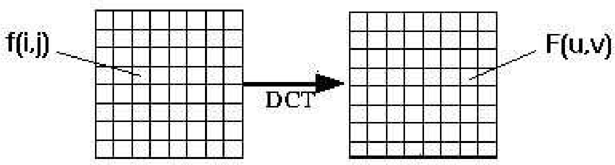
$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \cdot \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(u) \cdot \Lambda(v) \times$$
$$\cos\frac{\pi u}{2\mathrm{N}}(2\mathrm{i}+1)\cos\frac{\pi v}{2\mathrm{M}}(2\mathrm{j}+1) \cdot f(i,j)$$

and the corresponding **inverse** 2D DCT transform is simply $F^{-1}(u,v)$, i.e.:

$$f(i,j) = F^{-1}(u,v)$$
$$= \left(\frac{2}{N}\right)^{\frac{1}{2}} \cdot \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(u) \cdot \Lambda(v) \times$$
$$\cos\frac{\pi u}{2\mathrm{N}}(2\mathrm{i}+1)\cos\frac{\pi v}{2\mathrm{M}}(2\mathrm{j}+1) \cdot F(u,v)$$

Applying The DCT

- Similar to the discrete Fourier transform:
  - It transforms a signal or image from the spatial domain to the frequency domain.
  - DCT can approximate lines well with fewer coefficients.



- Helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality).

Performing DCT Computations

The basic operation of the DCT is as follows:

- The input image is N by M;
- f (i, j) is the intensity of the pixel in row i and column j.
- F(u,v) is the DCT coefficient in row u_i and column v_j of the DCT matrix.
- For JPEG image (and MPEG video), the DCT input is usually an 8 by 8 (or 16 by 16) array of integers. This array contains each image window's respective colour band pixel levels.

Compression with DCT

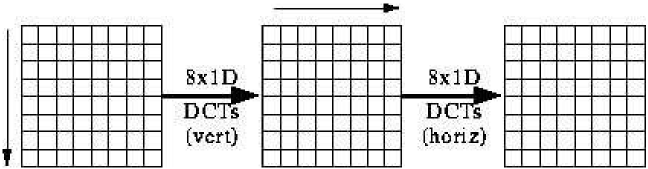
- For most images, much of the signal energy lies at low frequencies;
  - These appear in the upper left corner of the DCT.
- Compression is achieved since the lower right values represent higher frequencies, and are often small
  - Small enough to be neglected with little visible distortion.

Separability

- One of the properties of the 2-D DCT is that it is separable meaning that it can be separated into a pair of 1-D DCTs.
- To obtain the 2-D DCT of a block a 1-D DCT is first performed on the rows of the block then a 1-D DCT is performed on the columns of the resulting block.
- The same applies to the IDCT.

Separability

- Factoring reduces problem to a series of 1D DCTs (No need to apply 2D form directly):
  - As with 2D Fourier Transform.
  - Apply 1D DCT (vertically) to columns.
  - Apply 1D DCT (horizontally) to resultant vertical DCT.
  - Or alternatively horizontal to vertical.



Computational Issues

- The equations are given by:

$$G(i,v)=\frac{1}{2}\sum_{j}\Lambda(v)\cdot\cos\frac{\pi v}{16}(2j+1)\cdot f(i,j)$$
$$F(u,v)=\frac{1}{2}\sum_{i}\Lambda(u)\cdot\cos\frac{\pi v}{16}(2i+1)\cdot G(i,v)$$

- Most software implementations use fixed point arithmetic. Some fast implementations approximate coefficients so all multiplies are shifts and adds.

2D DCT on an Image Block

- Image is partitioned into 8 x8 regions (See JPEG)—The DCT input is an 8 x 8 array of integers.
- So in N = M = 8, substitute these in DCT formula.
- An 8 point DCT is then:

$$F(u,v) = \frac{1}{4}\Lambda(u)\Lambda(v)\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{\pi u}{16}(2i+1)\times \cos\frac{\pi v}{16}(2j+1)\cdot f(i,j)$$

where

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

- The output array of DCT coefficients contains integers; these can range from −1024 to 1023.

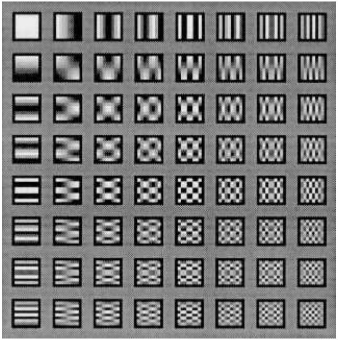
2D DCT Basis Functions

From the above formula, extending what we have seen with the 1D DCT we can derive basis functions for the 2D DCT:

- We have a basis for a 1D DCT (see bases = dctmtx(8) example above).
- We discussed above that we can compute a DCT by first doing a 1D DCT in one direction (e.g. horizontally) followed by a 1DCT on the intermediate DCT result.
- This is equivalent to performing matrix pre-multiplication by bases and matrix post-multiplication the transpose of bases.
  - take each row i in bases and you get 8 basis matrices for each j.
  - there are 8 rows so we get 64 basis matrices.

Visualisation of DCT 2D Basis Functions

- Computationally easier to implement and more efficient to regard the DCT as a set of **basis functions**.
  - Given a known input array size (8 x 8) they can be precomputed and stored.
  - The values as simply calculated from DCT formula.



See MATLAB demo, dctbasis.m, to see how to produce these bases.

DCT Basis Functions

```
A =dctmtx(8);
A =A';

Offset =5;

basisim =ones(N*(N+offset))*0.5;
```

- Basically just set up a few things A := 1D DCT basis functions
- basisim will be used to create the plot of all 64 basis functions.

DCT Basis Functions

```
B=zeros(N,N,N,N);
for i=1:N
 for j=1:N
    B(:,:,i,j)=A(:,i)*A(:,j)';
    %max(max(B(:,:,i,j)))-min(min(B(:,:,i,j)))
 end;
end;
```

- B = computation of 64 2D bases.
- Create a 4D array: first two dimensions store a 2D image for each i,j.
- 3rd and 4th dimension i and j store the 64 basis functions.

DCT Basis Functions

```
for i=1:N
for j=1:N
minb = min(min(B(:,:,i,j)));
maxb = max(max(B(:,:,i,j)));
rangeb =maxb -minb;

if rangeb ==0
      minb =0;
      rangeb =maxb;
end;

imb = B(:,:,i,j) - minb;
imb =imb/rangeb;
```

DCT Basis Functions

```
iindex1 =(i-1)*N + i*offset-1;
iindex2 =iindex1 + N -1;

jindex1 =(j-1)*N + j*offset -1;
jindex2 =jindex1 + N -1;

basisim(iindex1: iindex2, jindex1:jindex2) = imb;
end;
end;
```

- Basically just put up 64 2D bases in basisim as image data.

DCT Basis Functions

```
figure(1)
imshow(basisim)

figure(2)
dispbasisim = imresize(basisim,4,'bilinear');
imshow(dispbasisim);
```

- Plot normal size image and one 4 times up sampled.