CM3106 Multimedia

Discrete Cosine Transform

Dr Kirill Sidorov
SidorovK@cardiff.ac.uk
www.facebook.com/kirill.sidorov

Prof David Marshall
MarshallAD@cardiff.ac.uk

School of Computer Science and Informatics
Cardiff University, UK

Recap: frequency domain

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

- (Discrete) Fourier Transform (DFT) (see Chapter 2 and recall from CM2202) — used in MPEG Audio.
- (Discrete) Cosine Transform (DCT) (new) — heart of **JPEG** and MPEG Video, MPEG Audio.

Strongly recommended MIT video lecture by Prof Walter Lewin:
External Link: MIT OCW 8.03 Lecture 11 Fourier Analysis Video

Recap: Fourier transform

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

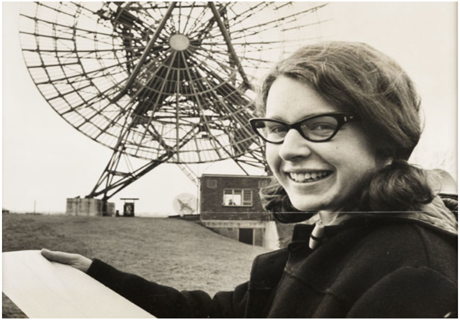
The result of the transform version is usually referred to as the Fourier- (or frequency-) space representation of the signal.
We can then manipulate the signal:

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

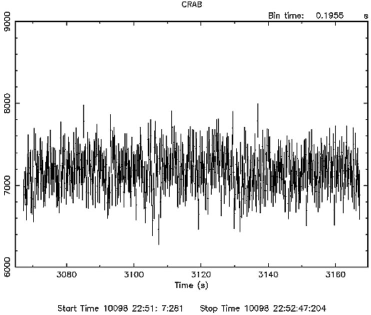
We then need to convert data back to real audio/imagery to use in our applications.

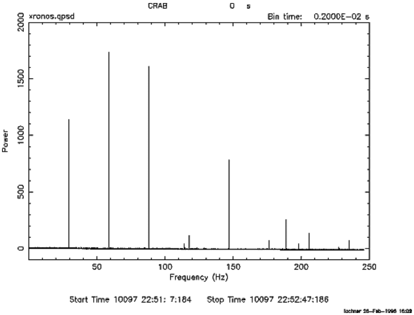
The corresponding inverse transformation which turns a Fourier space representation back into a real space one is called the inverse Fourier transform.

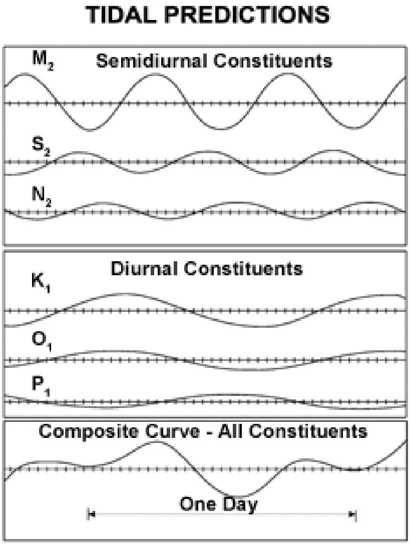
Little Green Men or pulsars?



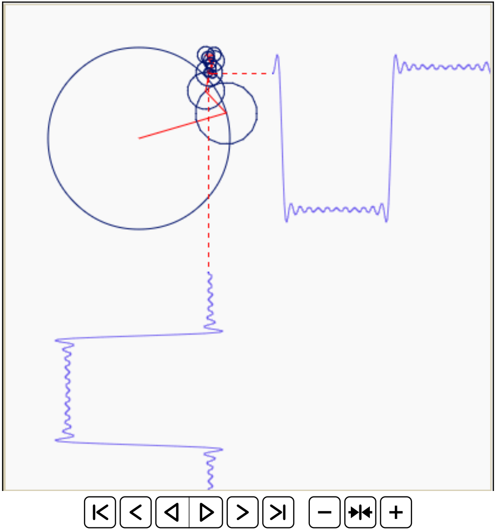
FT is absolutely essential in *e.g.* astronomy to study periodic processes: pulsars, exoplanets.






TIDAL PREDICTIONS
M2
Semidiurnal Constituents
S2
N2
Diurnal Constituents
K1
O1
P1
Composite Curve - All Constituents
One Day

Animation



Recap: What do frequencies mean in an image?

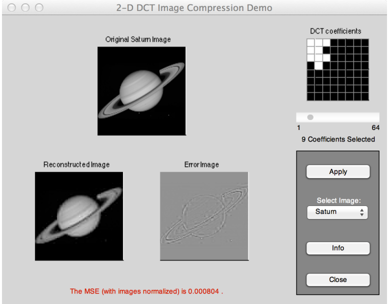
- Large values at high frequency components mean the data is changing rapidly on a short distance scale.
  E.g.: a page of small font text, brick wall, vegetation.
- Large low frequency components then the large scale features of the picture are more important.
  E.g. a single fairly simple object which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

The road to compression

How do we achieve compression?

- Low pass filter — ignore (or better: store with lower fidelity) high frequency (noise) components
  - Only store lower frequency components
- High pass filter — gradual changes in an image
  - If changes are too low/slow — eye does not respond so ignore?

Low pass image compression example



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

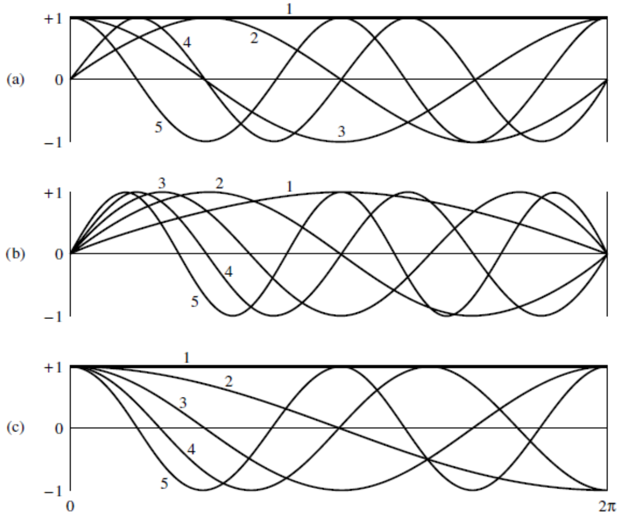
- Load an image.
- Low pass filter in frequency (DCT) space.
- Tune compression via a single slider value to select n coefficients.
- Inverse DCT, subtract input and filtered image to see compression artefacts.

Relationship between DCT and DFT

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 (Discrete) Fourier Transform:

- Only the real part of DFT.
- Computationally simpler than DFT.
- DCT — effective for multimedia compression (energy compaction).
- DCT is much more commonly used (than DFT) in multimedia image/video compression — more later.
- Cheap MPEG Audio variant — more later.
- DFT captures phase, though.

Cosine-, Sine-, and Fourier Transform



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

For N-dimensional vectors, the 1D DCT is defined by:

$$F(k) = \lambda(k) \sum_{n=1}^{N} f(n) \cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right), \quad k = 1, 2, \ldots, N.$$

and the corresponding inverse 1D DCT transform is:

$$f(n) = \sum_{k=1}^{N} \lambda(k) F(k) \cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right), \quad n = 1, 2, \ldots, N.$$

All indices k, n start with one, following MATLAB convention.
The normalising weight $\lambda(k)$ is:

$$\lambda(k) = \begin{cases} 1/\sqrt{N} & \text{when } k = 1, \\ \sqrt{2/N} & \text{when } 2 \leq k \leq N. \end{cases}$$

Compare with DFT

Recap: Discrete Fourier Transform

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

$$k = 1, 2, \ldots, N.$$

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

$$n = 1, 2, \ldots, N.$$

Very similar idea, but different basis functions.

Algebraically:

- $F = Df$, where D is the matrix of DCT coefficients.
- Inverse transform: $f = D^{-1}F$.
- D is orthogonal, therefore $D^{-1} = D^T$.
- Easy to see that DCT is linear: $\texttt{dct}(\alpha x + \beta y) = \alpha\,\texttt{dct}(x) + \beta\,\texttt{dct}(y)$.

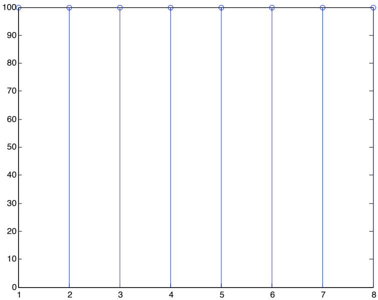
Let's consider a DC signal that is a constant 100,
that is $f(n) = 100$ for $n = 1 \ldots 8$
(see DCT1Deg.m):

- So the domain is $[1, 8]$ for both n and k
  - We therefore have $N = 8$ samples and will need to compute the 8 values (DCT coefficients) for $k = 1 \ldots 8$.

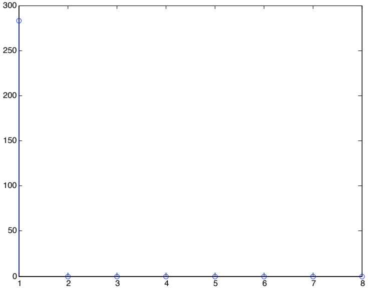
We can now see how we work out $F(k)$:

- As k varies we work can work for each k the k-th DCT coefficient.
- For each $F(k)$, we can compute the value for each $F_n(k)$ to define a basis function.
- Basis functions can be pre-computed and simply looked up in DCT computation.

Plots of $f(n)$ and $F(k)$



$f(n) = 100$ for $n = 1 \ldots 8$



$F(k)$: $F(1) \approx 283$, $F(2 \ldots 8) = 0$

DCT example: $F(1)$

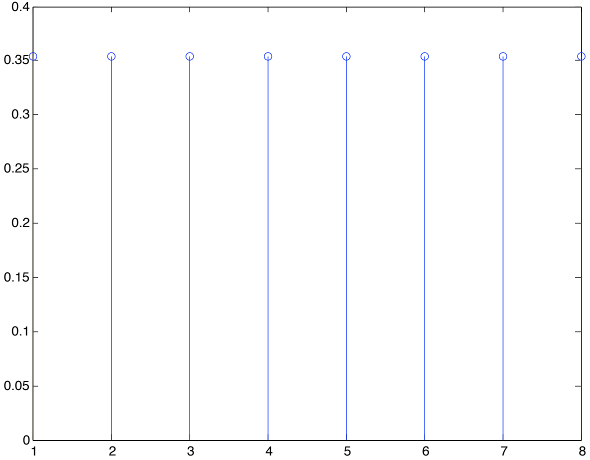
So for $1 = 0$:

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

$$\begin{aligned} F(1) &= \frac{1}{2\sqrt{2}}(1 \cdot 100 + 1 \cdot 100 + 1 \times 100 + 1 \cdot 100 + 1 \cdot 100 \\ &\quad +1 \cdot 100 + 1 \cdot 100 + 1 \cdot 100) \\ &\approx 283 \end{aligned}$$

- Here the values $F_n(1) = \frac{1}{2\sqrt{2}}$ ($n = 1 \ldots 8$).
  These are the bases of $F_n(1)$

$F(1)$ basis function plot



$F(1)$ basis function

So for $k=2$:
Note: $\lambda(1)=1$ and we have $\cos$ to work out: so $F(2)$ is computed as:

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

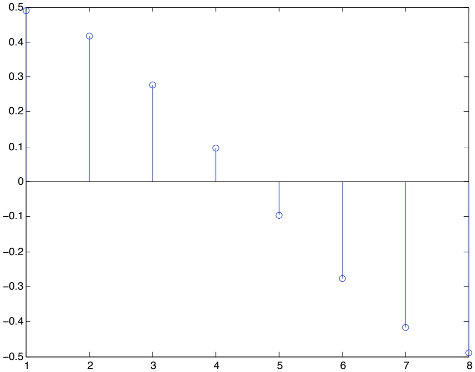
(since $\cos\frac{\pi}{16}=-\cos\frac{15\pi}{16}$, $\cos\frac{3\pi}{16}=-\cos\frac{13\pi}{16}$ *etc.*)
Here the values

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

form the basis function

$F(3\ldots8)$ similarly $=0$

$F(1)$ basis function plot



$F(1)$ basis function

Note:

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

DCT1Deg.m explained:

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

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

```
% 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.

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

D1 = bases*f';

figure  % 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.

For a 2D N by M matrix (*e.g.* image) the 2D DCT is:

$$F(p,q) = \lambda(p)\lambda(q) \sum_{m=1}^{M} \sum_{n=1}^{N} \Big(f(m,n) \times$$

$$\cos\left(\frac{\pi}{2M}(2m-1)(p-1)\right) \cos\left(\frac{\pi}{2N}(2n-1)(q-1)\right) \Big),$$

for $p \in 1 \ldots M, q \in 1 \ldots N$

and the corresponding inverse 2D DCT transform is:

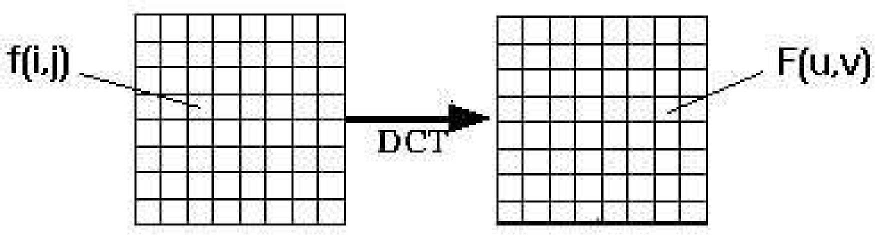
$$f(m,n) = \sum_{p=1}^{M} \sum_{q=1}^{N} \lambda(p)\lambda(q) \Big(F(p,q) \times$$

$$\cos\left(\frac{\pi}{2M}(2m-1)(p-1)\right) \cos\left(\frac{\pi}{2N}(2n-1)(q-1)\right) \Big),$$

for $m \in 1 \ldots M, n \in 1 \ldots N$

Applying the 2D DCT

- Similar to the 2D discrete Fourier transform:
  - It also transforms an image from the spatial domain to the frequency domain.
  - DCT can approximate lines well with fewer coefficients.

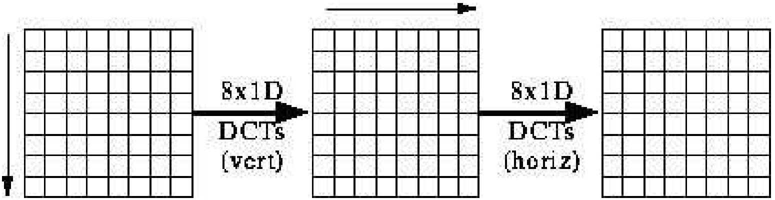


- Helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality).

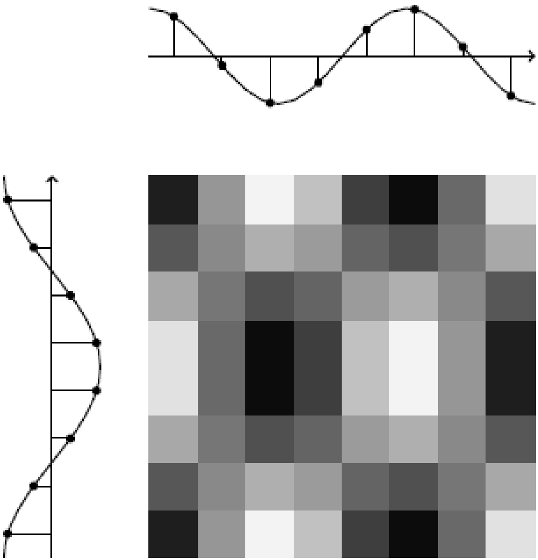
Separability

- 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.

- 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.



Separability



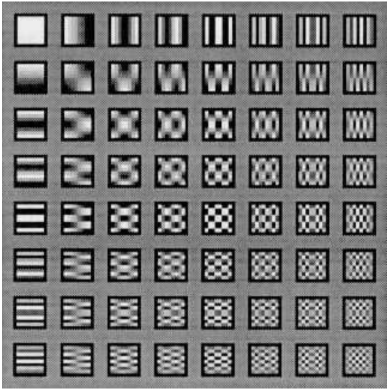
2D DCT basis functions

From the above DCT formulæ, 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 1 DCT 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

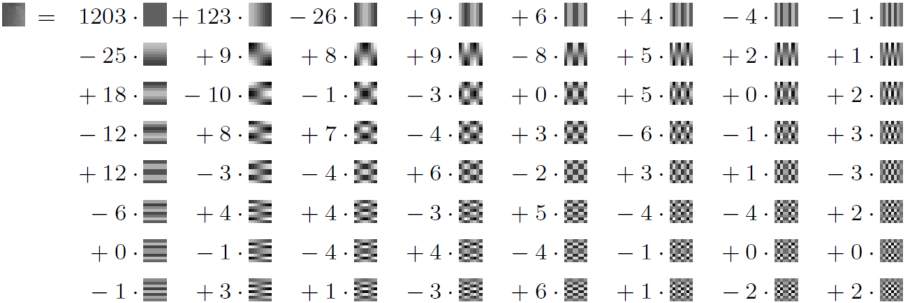
- Computationally easier to implement and more efficient to regard the DCT as a set of basis functions.
  - Given a known input array size (8 x 8) they can be precomputed and stored.
  - The values are simply calculated from DCT formulæ.



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

http://weitz.de/dct/
nice DCT 2D demo.

Example: DCT of 8×8 image block



```
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.

```
B=zeros(N,N,N,N);
for i=1:N
 for j=1:N
   B(:,:,i,j)=A(:,i)*A(:,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.

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.