Image Processing 1: Filters

The Discrete Cosine Transform DCT

The Discrete Cosine Transform DCT is derivated from the Discrete Fourier Transform DFT.
It maps a spatial or time function `xx[0...N-1]` into the frequency domain,
that is to say into amplitudes `XX[0...N-1]` of `N` cosine basis functions.
The transform is completely or partially invertible (= back transform).
Varied filtering is possible by restricting the back transform to a limited set of frequencies.

The 1D-DCT is defined by:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0,\dots,N-1.$$

and its back (=inverse) transform:

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\frac{\pi}{N}n\left(k+\frac{1}{2}\right)\right] \qquad k = 0,\dots,N-1.$$

See Wikipedia: **DCT** and **DFT** and Miszalok: **Lecture on the Fourier Transform**

```
//C#-Code of the Discrete Cosine Transform DCT and of its back transform:
//xx = spatial domain input/output array, XX = frequency domain array
private void DCT( float[] xx, float[] XX )
{ int k, n, N = xx.Length;
  float sum;
  double piN =   Math.PI / N;
  float scaleFactor = 2f / N; //adapts XX[k] to DCTBack
  for ( k=0; k < N; k++ )
  { sum = 0;
    for ( n=0; n < N; n++ ) sum += xx[n] * (float)Math.Cos( piN*(n+0.5)*k );
    XX[k] = sum*scaleFactor;
  }
}
private void DCTBack( float[] XX, float[] xx, int lowestFrequency. int highestFrequency )
{ int k, n;
  double piN = Math.PI / lastFrequency;
  for ( k = 0; k < N; k++ )
  { float sum = XX[0] / 2;
    for ( n = 1; n < N; n++ )
      sum += XX[n] * (float)Math.Cos( piN*n*(k+0.5) );
    xx[k] = sum;
  }
}
```

Sample of a 1D Discrete Cosine Transform and of its Back Transform

The 2D DCT consist of two consecutive 1D-DCTs and its 2D back transform consists of 2 consecutive 1D-inverseDCTs.
The algorithm needs 6 big arrays:

```
img0 = new WriteableBitmap(BitmapImage);  //original JPG or PNG image
Byte [,] R0 = new Byte [ySize,xSize]      //red channel  values from img0
float[,] R1 = new float[ySize,xSize]      //horizontally transformed R0
float[,] R2 = new float[ySize,xSize]      //vertically   transformed R1
float[,] R3 = new float[ySize,xSize]      //vertically   back transformed R2
img1 = new WriteableBitmap(xSize,ySize);  //horizontally back transformed R3
```

	There are 4 consecutive transformation steps:
1.	horizontal 1D-DCT of all rows of `R0` to `R1`.
2.	vertical 1D-DCT of all columns of `R1` to `R2`. `R2` is the 2D-DCT matrix.
3.	vertical 1D-inverseDCT of all columns of `R2` to `R3` restricted to frequencies between `VerticLowest` and `VerticHighest`.
4.	horizontal 1D-inverseDCT of all rows of `R3` to `img1` restricted to frequencies between `HorizLowest` and `HorizHighest`.

Drag the sliders to generate input !

****
Experiments:
1. Drag the sliders arbitrarily.
2. Drag all sliders to 0.
3. Drag all sliders to 100.
4. Drag 1., 3., 5., etc. sliders to 0 and 2., 4., 6., etc. sliders to 100. Result: frequencies 2, 4, 6, 8, 10 are almost 0.
5. Drag 1+2, 5+6, 9+10 to 100 and all others to 0. Result: frequencies 2, 4, 6, 8, 10 are almost 0.
****

Sample of a 1D Discrete Cosine Transform with a Limited Back Transform

```
//Just one line changed within the Discrete Cosine Transform back transform.
//xx = spatial domain input/output array, XX = frequency domain array
private void DCTBack( float[] XX, float[] xx, int lastFrequency )
{ int k, n;
  double piN = Math.PI / lastFrequency;
  for ( k = 0; k < N; k++ )
  { float sum = XX[0] / 2;
    for ( n = 1; n < lastFrequency; n++ )
      if ( checkBox[n].IsChecked == true ) sum += XX[n] * (float)Math.Cos( piN*n*(k+0.5) );
    xx[k] = sum;
  }
}
```

****
Experiments:
1. Drag the sliders arbitrarily.
2. Drag all sliders to 0.
3. Drag all sliders to 100.
4. Drag 1., 3., 5., etc. sliders to 0 and 2., 4., 6., etc. sliders to 100.
****

2D Discrete Cosine Transform of monochrome images

Experiments:
For small slider increments and decrements click first on the horizontal guide rail right or left and finally click the valve itself.
1. Lower HorizHighestFrequ down to 3.
2. Lower VerticHighestFrequ down to 3..
3. Increase HorizLowestFrequ to 4, 5, 6 .....
4. Increase VerticLowestFrequ to 4, 5, 6 ....
5. Shift all sliders to their minimum and increment both HighestFrequ-Sliders stepwise to 4, 5, 6 etc.
6. a) Shift VerticLowestFrequ to 1 and VerticHighestFrequ to maximum.
   b) Shift both horizontal frequencies to minimum.
   c) Increment HorizLowestFrequ to 2, 3, 4, etc.
   d) HorizHighestFrequ automatically increments to 5, 6, 7 etc.
   e) The narrow horizontal bandwidth results in strange artefacts.
7. a) Shift HorizLowestFrequ to 1 and HorizHighestFrequ to maximum.
   b) Try the same as 6. b), c), d) e) in vertical direction.

Questions & Answers

****
Q: What is the Fourier series ?

A: Functions *F(x)* are written as the sum of sine- and cosine-waves:

$$F(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

****
Q: What is the Cosine series ? Comparison to Fourier series ?
A: Functions are written as a sum of cosine functions only.
Advantage: Less complicated algorithm
Disadvantage: Cosine series have double the length of Fourier series.
Same: Enormous computational time is needed.
****
Q: Explanations of: spatial domain, frequency domain, DFT, DCT, inverseDFT, inverseDCT in image processing ?
A:
spatial domain: a set of pixels delivered by a camera.
frequency domain: a set of sine- and/or cosine-functions oscillating in different frequencies.
Both DFT = Discrete Fourier Transform and DCT = Discrete Cosine Transform express a sequence of pixels in terms of a set of oscillating sine- and/or cosine-functions.
Both inverseDFT = Discrete Fourier Back Transform and inverseDCT = Discrete Cosine Back Transform express a set of oscillating sine and/or cosine functions as a sequence of pixels.
****
Q: Benefit of filtering the frequency domain ?
A: 1. Noise reduction by throwing away high frequencies: lowpass.
   2. Shading correction by throwing away low frequencies: highpass.
   3. Amplification of a detail by throwing away foreign frequencies: bandpass
   4. Compression by throwing away irrelevant frequencies: JPG, PNG.
****
Q: Computing time of DFT and DCT of an NxN image ? What is FFT ?
Both DFT and DCT and their inverse transforms run 3 nested for-loops 0...N which results in N^3 operations.
The Fast Fourier Transform FFT reduces N^3 to $\log_2 N * N^2$ operations.
Drawback: The FFT assumes N being a power of 2 as 128, 256, 512 etc.
****
Q: JPG is a lossy image compression standard basing on DCT. Why is it so fast ?
A: The image is subdivided into small 8x8 pixel blocks which undergo DCT.
The resulting data for all 8x8 blocks is further compressed by a faster compression algorithm.
****