ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
english only

Biomedical Imaging Group

IP-LAB: Image Processing Laboratories

Wavelet Transformation and Applications

1. Code a simple wavelet transform

We propose to study and implement the Haar wavelet transformation:

$$\begin{bmatrix} y_1(k) \\ y2(k) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x(2k) \\ x(2k+1) \end{bmatrix}$$

ImageAccess Notes

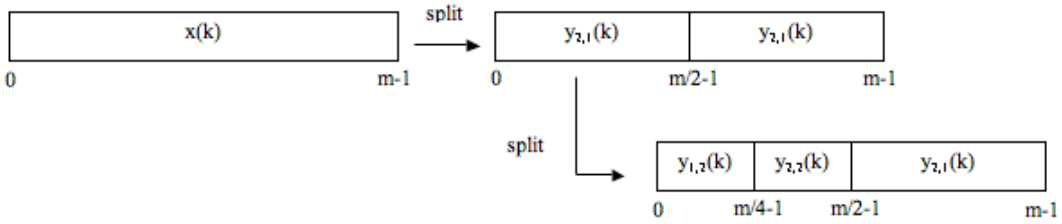
Display an ImageAccess object:	image.show("title");
Fill all pixels with a constant:	image.setConstant(3.0);
Build a image of size [nx, ny]:	ImageAccess sub = new ImageAccess(nx, ny);
Extract a sub-image from the position [off_x, off_y]:	image.getSubImage(off_x, off_y, sub);
Put a sub-image into a large image starting at the position [off_x, off_y]:	image.putSubImage(off_x, off_y, sub);

This transformation splits a discrete signal x(k) of size m into two parts $y_1(k)$ and $y_2(k)$ of size m/2. For splitting an image, the same decomposition is applied first on columns, then on rows. In this session, we assume that the image is square and that its size is a power of 2.

1.1 Analysis

The wavelet transform is implemented using 3 methods:

- `analysis()` performs n iterations of the wavelet transform of an image. It calls n times the method `split()`; n is the number of scales.
- `split()` performs a single iteration of the wavelet transform of an image. It takes advantage of the separability of the transform to call `split_1D()` for the rows first, and then for the columns.
- `split_1D()` decomposes a 1D signal x into the parts y_1 and y_2. The result is combined into a single output array y that contains $y_1[k]$ for k in the range 0 to m/2-1 and $y_2[k]$ for k in the range m/2 to m-1.



The methods `analysis()` and `split()` are already written. Your assignment is to code the method `split_1D()` in the file Code.java.

Test your work on mire.tif and aletsch.tif using the plugin **Image Transform**: check "Transform", uncheck "Processing" and "Inverse" and select n=3 in the dialog box. Select the output in "True Values" or in "Scaled Values" in which case the coefficients of the wavelet transform are rescaled between 0 to 255.

1.2 Synthesis filter and implementation

Write the synthesis filter in matrix form to reconstruct back the original signal x(k) form its sub-parts $y_1(k)$ and $y_2(k)$; fill in the report.

Your assignment is to code the three methods that perform the inverse wavelet transform.

- `synthesis()` should perform n iterations of the inverse wavelet transform of an image.
- `merge()` should perform a single iteration of the inverse wavelet transform of an image.
- `merge_1D()` should perform the reconstruction of a 1D signal from the 2 parts using the synthesis filter.

Test your work on mire and aletsch.tif, check "Transform" and "Inverse", uncheck "Processing".

1.3 Check the transformation and reconstruction process

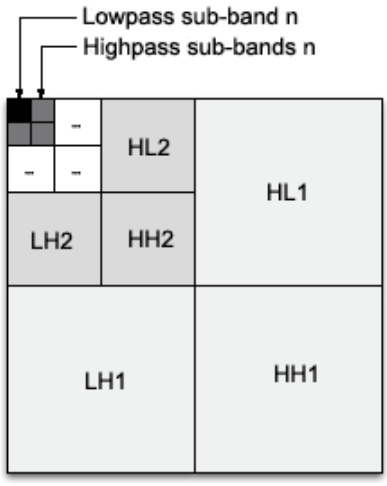
We propose to compare the reconstruction of 4 transformations on the image aletsch.tif

- Fourier Transform: using the ImageJ's commands (Process⇒FFT⇒FFT and Process⇒FFT⇒Inverse FFT);
- Haar Wavelet Transform (your code or the teacher solution) for n=3 using the plugin **Image Transform**;
- Discrete Cosine Transform (the DCT 8x8 is provided) using the plugin **Image Transform**;
- Spline Wavelet Transform (provided) for n=3 and n=8 using the plugin **Image Transform**.

Compute the SNR between the original image and the reconstructed image (transform+inverse). Using the plugin **SNR**. Conclude and fill in the report.

2. Modifying sub-band wavelet coefficients

The wavelet coefficients are stored according to the arrangement shown on the right figure where n is the total number of scales.



Write the routines `keepHighpassSubBand(coef, n)` that only preserves coefficients of 3 highpass sub-bands (HL_n, HH_n, LH_n) at level n and sets to 0 the other coefficients, and `keepLowpassSubBand(coef, n)` that only preserves the coefficients of the lowpass (LLn) sub-band at level n. Choose the "Keep 3 Highpass sub-band" or "Keep 1 Lowpass sub-band" as processing in the plugin.

Make the following experience on the aletsch.tif image:

- Haar Wavelet, n=3, lowpass (i.e. LL_3)
- Spline Wavelet, n=3, lowpass, save the reconstructed image
- Spline Wavelet, n=2, highpass
- Spline Wavelet, n=4, highpass, save the reconstructed image

Comment on the wavelet family and on the effect of the parameter *n* and fill in report.doc.

3. Comparison of the transform in term of compression

Simple data compression is achieved by applying a hard threshold to the coefficients of the wavelet transform (used in JPEG2000) or the DCT (used in JPEG). Note that this is only a rudimentary form of compression. A true coder would further quantize the wavelet coefficients which induces additional errors. The resulting coefficient map would also need to be encoded efficiently using, for example, the EZW algorithm (Embedded Zero-tree Wavelet coding).

To apply the compression function, check "Transform", "Processing" and "Inverse" , select "Hard threshold" and the rate (percentage of non-zero coefficients). If the rate is 10%, the image will be reconstructed using the 10% largest transform coefficients.

Compute the SNR between the original image (aletsch.tif) and the compressed image to assess the quality of the compression for the 3 transformations "Haar Wavelet, 6 scales", "Spline Wavelet, 6 scales" and "DCT 8x8" and for the rate = 2% and rate = 10%. Insert in the report the best obtained image for the rate = 2% and rate = 10% and fill in report.doc.

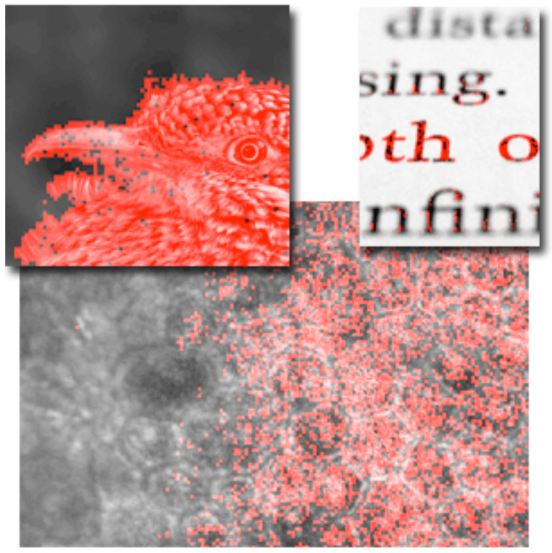
4. Detection of in-focus areas

We propose to design a simple detector of sharp (i.e in-focus) areas based on the wavelet transform. The sharp areas are characterized by larger coefficients (high-energy) in highpass sub-bands. The algorithm is the following:

- Evaluate a wavelet transform with *n* scales;
- Compute a normalized sharpness factor f(x,y) for each coefficients of the sub-band *n*;

$$f_{sharp}(x,y) = \frac{\sqrt{g_{HL}^2(x,y) + g_{HH}^2(x,y) + g_{LH}^2(x,y)}}{2^n}$$

- Build a mask image of the detected areas: 255 if f(x,y) is greater than the threshold *T* and 0 otherwise.
- Enlarge the mask to obtain the original size of the image.



4.1. Coding

Write the routine `detectSharpArea()` which implements the described algorithm. The parameters *n* and *T* are given through the dialog box.

- The Spline Wavelet Transform can be called using `ImageAccess coef = WaveSpline.analysis(input, n);`
- The red display of the mask over the image can be done using `Detect_SharpArea.showMaskOverlay(input, mask);` where input and mask are two ImageAccess objects of the same size.

4.2 Experimentation

Report the result of your experiences in the report.doc for the 3 images: roadrunner.tif, paper-text.tif and retina.tif.

webmaster.big@epfl.ch • 10.05.2007