# WAVELET COMPRESSION WITH FEATURE PRESERVATION AND DERIVATIVE DEFINITION

**Oleg Kiselyov and Paul Fisher**

Department of Computer Sciences
P.O.Box 13886
University of North Texas, Denton Texas 76203
Phone: 817-565-2767 Fax: 817-565-2799
Email: oleg@ponder.csci.unt.edu

# FEATURES

• Loose wavelet basis (frame) *

$\Rightarrow$ Lower enthropy of the wavelet decomposition

• Trimmimg of the Laplacian pyramid according to the "contrast-frequency ratio"*

• Run-length + Arithmetic/LZW compression of the Laplacian pyramid

$\rightarrow$ • **Introducing Criteria Sets**

$\rightarrow$ Criteria Set ***area***: control over the amount of loss at particular locations of the picture

$\rightarrow$ • **Discrete gradient** of the picture from the Wavelet decomposition

$\rightarrow$ • A number of C++ functions dealing with images

$\Rightarrow$ ***image manipulation language***

*As suggested by DeVore, Jawerth & Lucier, DCC 91

Unique features are marked with $\rightarrow$

# WAVELET DECOMPOSITION

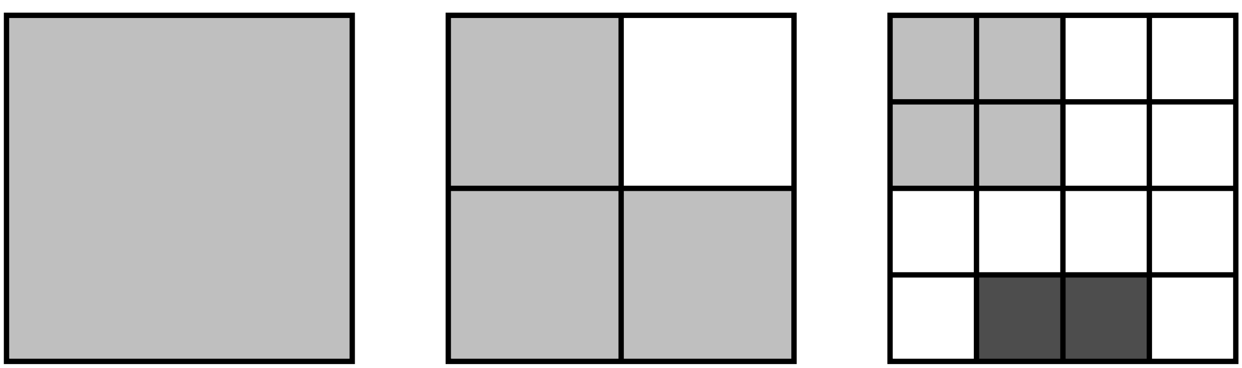
$$f(x,y) = \sum_{k=0}^{M} [\boldsymbol{P}^{k} f - \boldsymbol{P}^{k-1} f] = \sum_{k=0}^{M} \sum_{l,m} c^{k}_{lm} \; \Phi^{k}_{lm}(x,y) \qquad (1)$$

where

$f(x,y)$ - pixel value at $x^{th}$ row, $y^{th}$ column; $x,y = 0..N-1$

$\Phi^{k}_{lm}(x,y)$ - a frame (basis) function
= **1** over the square of size $N/2^{k}$
with upper left corner at $(l\, N/2^{k},\ m\, N/2^{k})$ (2)
= **0** everywhere else
$M = \log N$

$\boldsymbol{P}^{k} f$ - projector, computed as the mean intensity
over the squares of size $N/2^{k}$ ***rounded*** to the closest integer



$\boldsymbol{P}^{0} f$ $\boldsymbol{P}^{1} f$ $\boldsymbol{P}^{2} f$

# ALGORITHM

## a) Building Quadtree (Gaussian pyramid)

$$a^{k}_{lm} = f(l,m); \quad k = M, \; l,m=0..N-1$$

$$a^{k-1}_{lm} = \textbf{\textit{round}}\,\frac{1}{4}\left( a^{k}_{2l,2m} + a^{k}_{2l+1,2m} + a^{k}_{2l,2m+1} + a^{k}_{2l+1,2m+1} \right)$$

$$k=M,M-1,...1; \; l,m=0..2^{k-1}-1$$

## b) Building Laplacian Pyramid

$$c^{0}_{00} = a^{0}_{00}$$

$$c^{k+1}_{lm} = a^{k+1}_{lm} - a^{k}_{l/2,m/2}$$

$$k=0,1,..M-1; \; l,m=0..2^{k+1}-1$$

## c) Trimming/Quantization

==> SEE THE NEXT PAGE

## d) Run-Length Coding

of zero gaps left after trimmimg

## e) Arithmetic/LZW Compressing

the entire output file

NOTE,

Time complexity of the entire algorithm $\propto$ size of the image

# TRIMMING

## I. Uniform

- Sets $c^k_{lm} = 0$ if $\|c^k_{lm}\, \Phi^k_{lm}\,(x,y)\| < T$, a threshold
- Keeps only **significant features of the image**

  ***(contrast)*** $\times$ ***(grain-level) > threshold***

  **uniformly** over the entire picture

## → II. Non-Uniform

- Preserves certain image features in lossy compression according to a **(predefined) Criteria Set**
- Criteria Set **Area**
  sets out regions to trim finer/harsher
  $\Rightarrow$ **user-specified weight function/image r(x,y)**
- Trimming criterion

  $|c^k_{lm}|\, \|\Phi^k_{lm}\,(x,y)\|_{r(x,y)} < T$

  ***(contrast)*** $\times$ ***(grain-level)*** $\times$ ***(weight$_{(x,y)}$) < threshold***
- **Benefits**
  - Areas of special interest are encoded almost lossless
  - Higher compression ratio
  - Very smooth transition between coarse/fine areas

# DISCRETE GRADIENT

## Definition

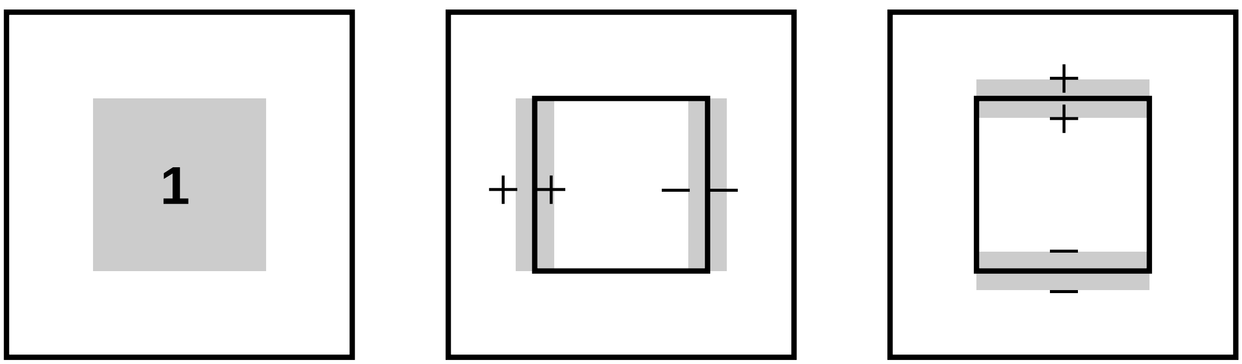
$$\boldsymbol{D}f(x,y) = |\boldsymbol{D}_x f(x,y)| + |\boldsymbol{D}_y f(x,y)| \tag{3}$$

$$\boldsymbol{D}_x f(x,y) = f(x+1,y) - f(x-1,y),$$

$$\boldsymbol{D}_y f(x,y) = f(x,y+1) - f(x,y-1), \tag{4}$$

## Computation

$$\boldsymbol{D}_x f(x,y) = \sum_{k=0}^{M} \sum_{l,m} c^k_{lm} \, \boldsymbol{D}_x \Phi^k_{lm}(x,y)$$




$\Phi^k_{lm}(x,y)$ $\quad$ $\boldsymbol{D}_x\Phi^k_{lm}(x,y)$ $\quad$ $\boldsymbol{D}_y\Phi^k_{lm}(x,y)$

## The point

- Highlighting lines, borders, etc.
- Control over the scale of non-regularities to emphasize
- Preventing the noise enhancement

# EXAMPLE of PROGRAM

```
// This may look like C code, but it is really -*- C++ -*-
//                      Main module

#include "laplpyr.h"
#include <builtin.h>
extern void system(const char * command);

main()
{
  IMAGE image("../old_images/lenna.xwd");

  image.display("Original image");
  LaplPyr lp(image,log2(image.q_nrows())+1);
#if 0
  IMAGE weight(image);          // Non-uniform Trimming
  weight = 1;
  weight.square_of(256,rowcol(127,127)) = 10;
  lp.parse_coeffs(400,weight);
#else
  lp.parse_coeffs(400);         // Uniform Trimming
#endif

  lp.write("/tmp/aa");
  system("comp_ratio ../old_images/lenna.xwd /tmp/aa");
  LaplPyr lp1("/tmp/aa");       // Read the pyramid back

  IMAGE & new_image = *(lp1.compose());
  new_image.display("Reconstructed image");
  compare(image,new_image,"Original and reconstructed images");

  IMAGE diff_image(image);
  diff_image = image; diff_image -= new_image;
  message(" -->Root mean square error is %g ",
          sqrt( (diff_image * diff_image) / diff_image.q_nrows() /
                                  diff_image.q_ncols() ));

  IMAGE & d_image = *(lp1.derivative_image());
  d_image *= 4;
  d_image.invert();
  d_image.display("Derivative image");
}
```

Original $512 \times 512 \times 8$ image "lenna"

Discrete Derivative

Compression 12:1, threshold 50

Discrete Derivative

Compression 58:1, threshold 400

Discrete Derivative

Discrete Derivative

Non-uniform trimming

threshold 400, weight function:

Compression 21:1