A Survey of Wavelet Algorithms and Applications, Part 2

M. Victor Wickerhauser

Department of Mathematics

Washington University

St. Louis, Missouri 63130 USA

victor@math.wustl.edu

http://www.math.wustl.edu/~victor

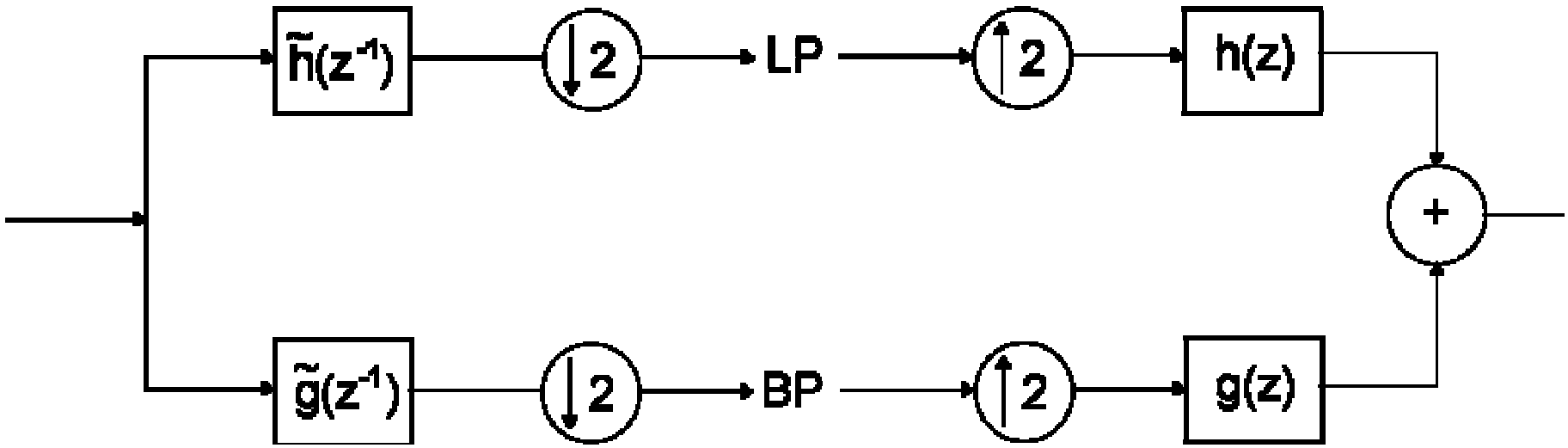
SPIE Orlando, April 4, 2002

Special thanks to ***Mathieu Picard***

Discrete Wavelet Transform

- Purpose: compute compact representations of functions or data sets
- Principle: a more efficient representation exists when there is underlying smoothness

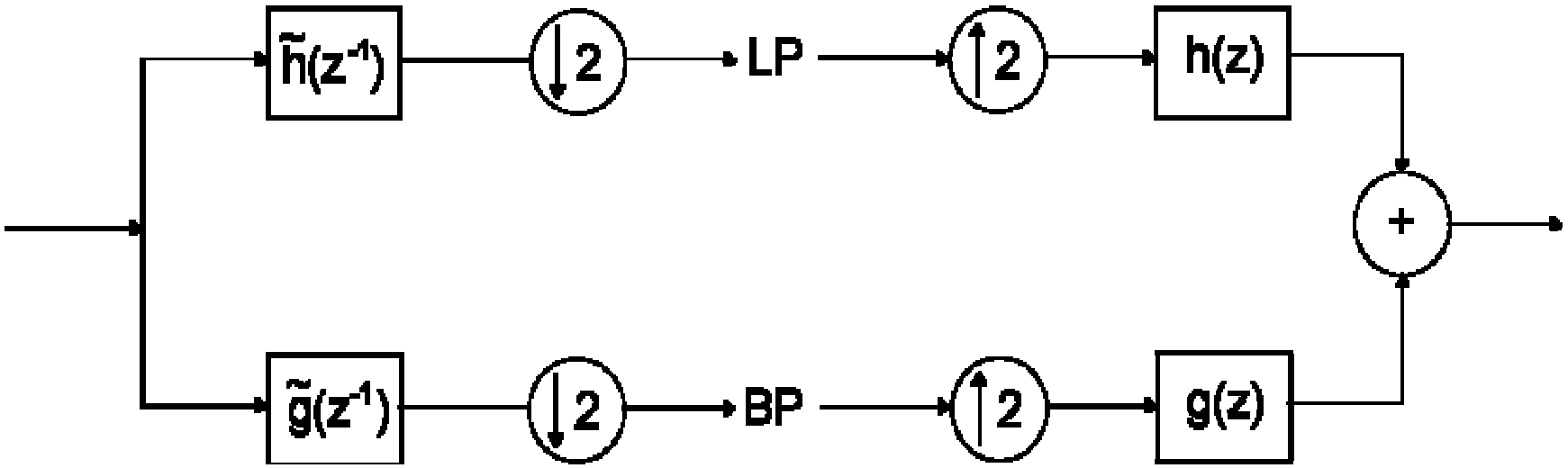
Subband Filtering



Low pass filter convolution: $$(hu)_k = \sum_n h_n u_{2k-n}$$

$$h(z) = \sum_n h_n z^n$$ is the equivalent Z −transform

Subband Filtering



Leads to a **perfect reconstruction** if :

$$h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2$$

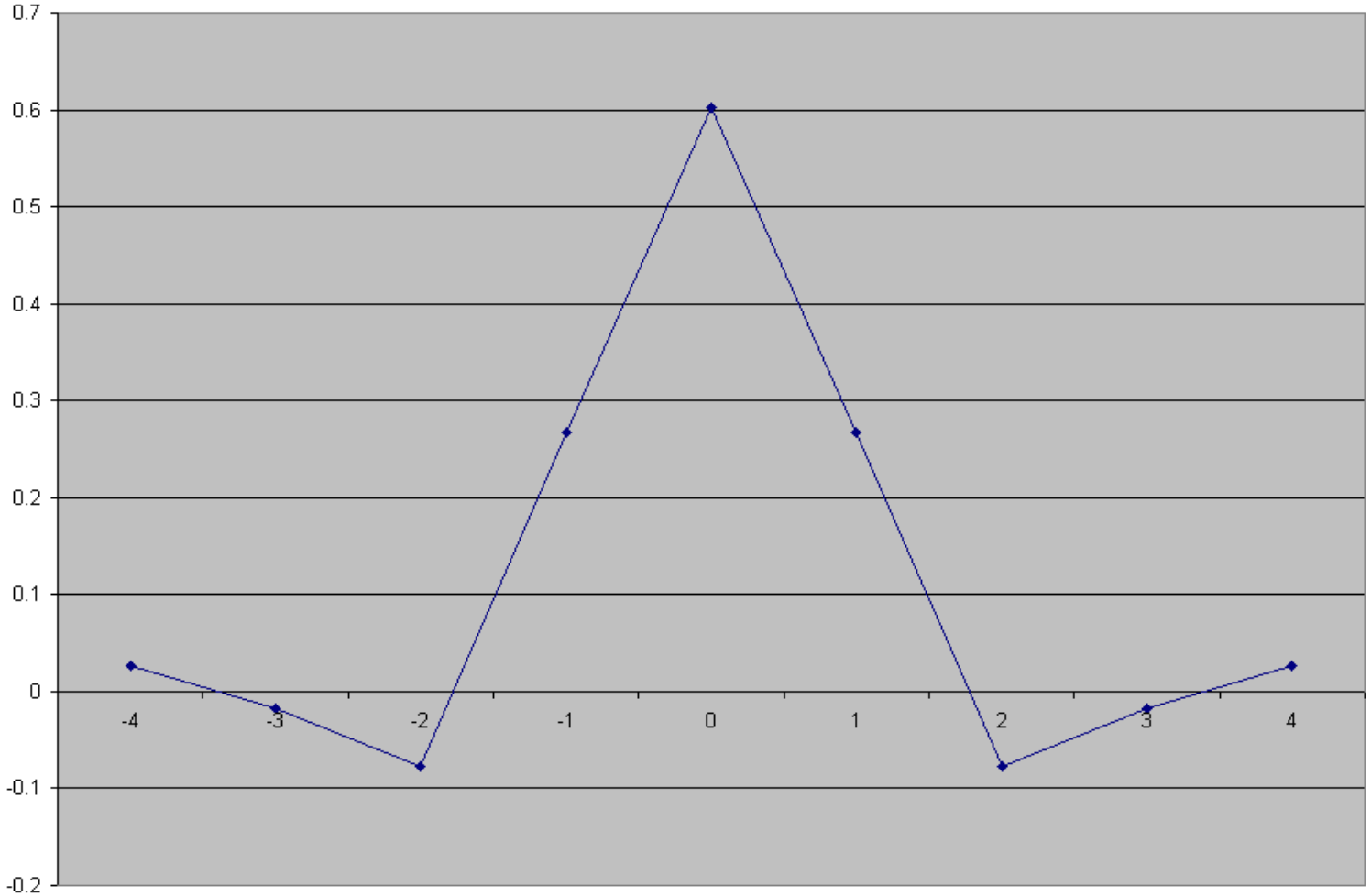
$$h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0.$$

(9−7) filter pair

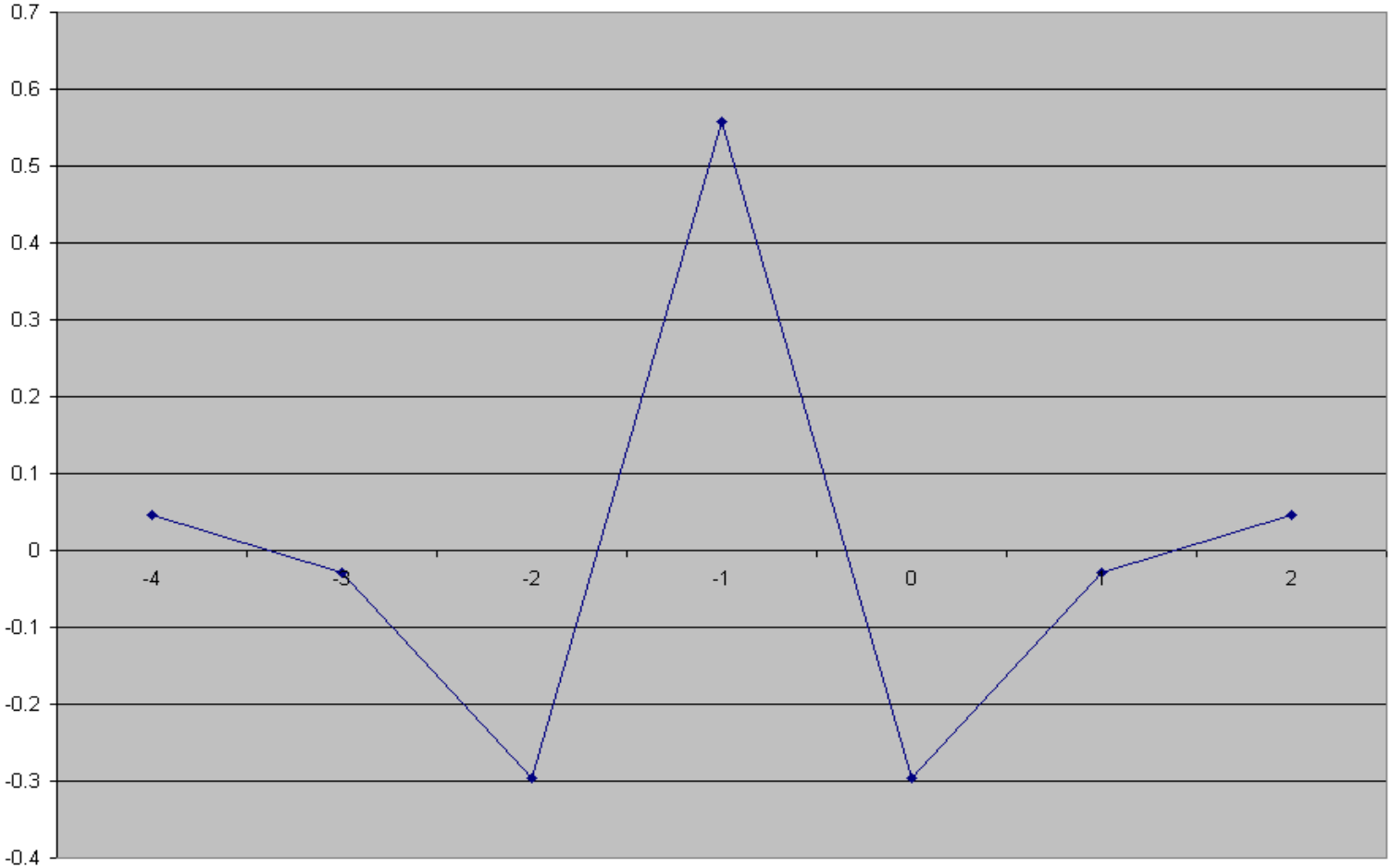
- Very popular and efficient for natural images (portraits, landscapes,…)
- Analysis filters
  - Low−pass : 9 coeff, High−pass : 7 coeff.
- Synthesis filters
  - Low−pass : 7 coeff, High−pass : 9 coeff.

n	0	± 1	± 2	± 3	± 4
$2^{-1/2}\tilde{h}_n$	0.602949	0.266864	-0.078223	-0.016864	0.026749
$2^{-1/2}\tilde{g}_{n-1}$	0.557543	-0.295636	-0.028772	0.045636	0

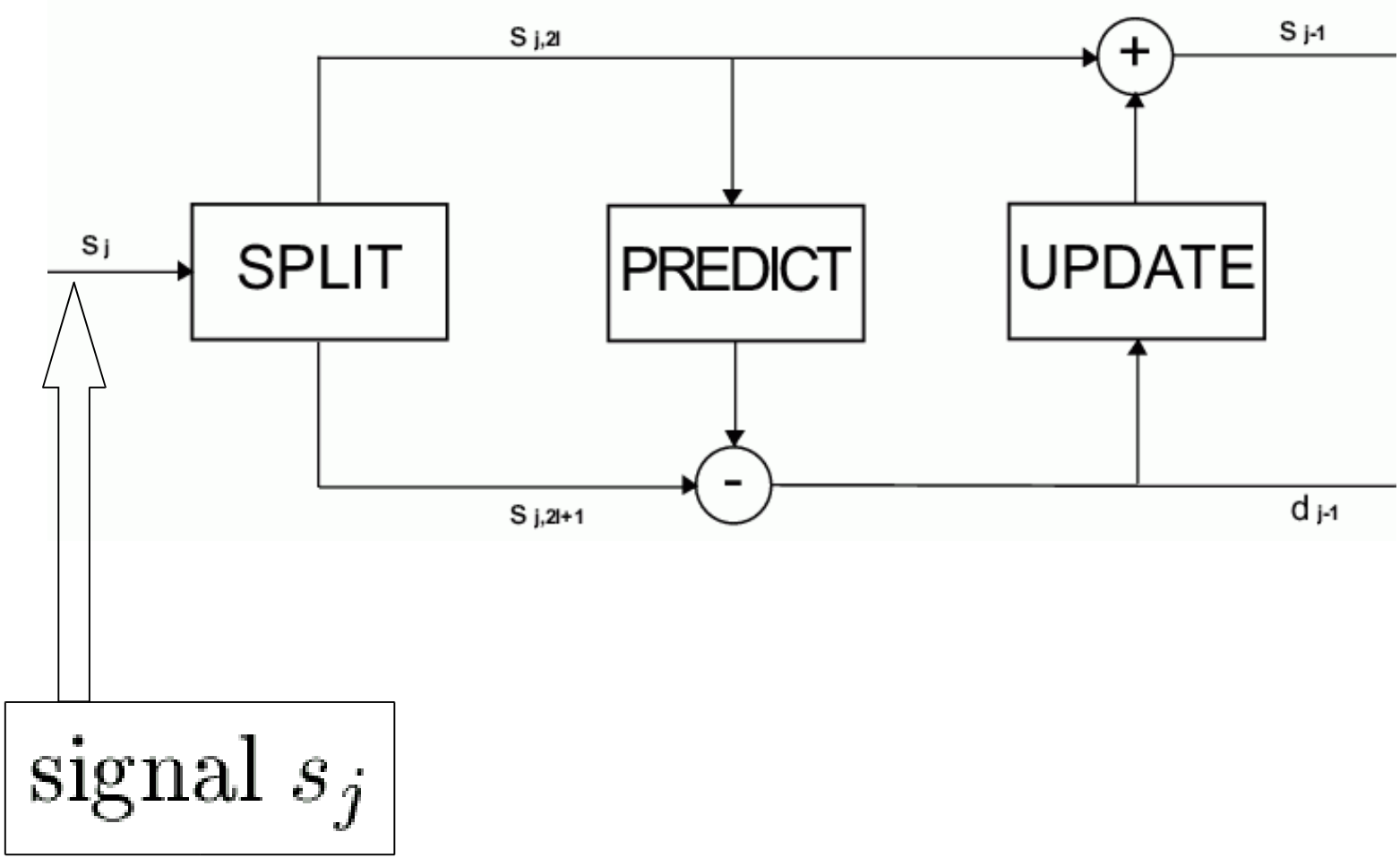
LOW−PASS filter



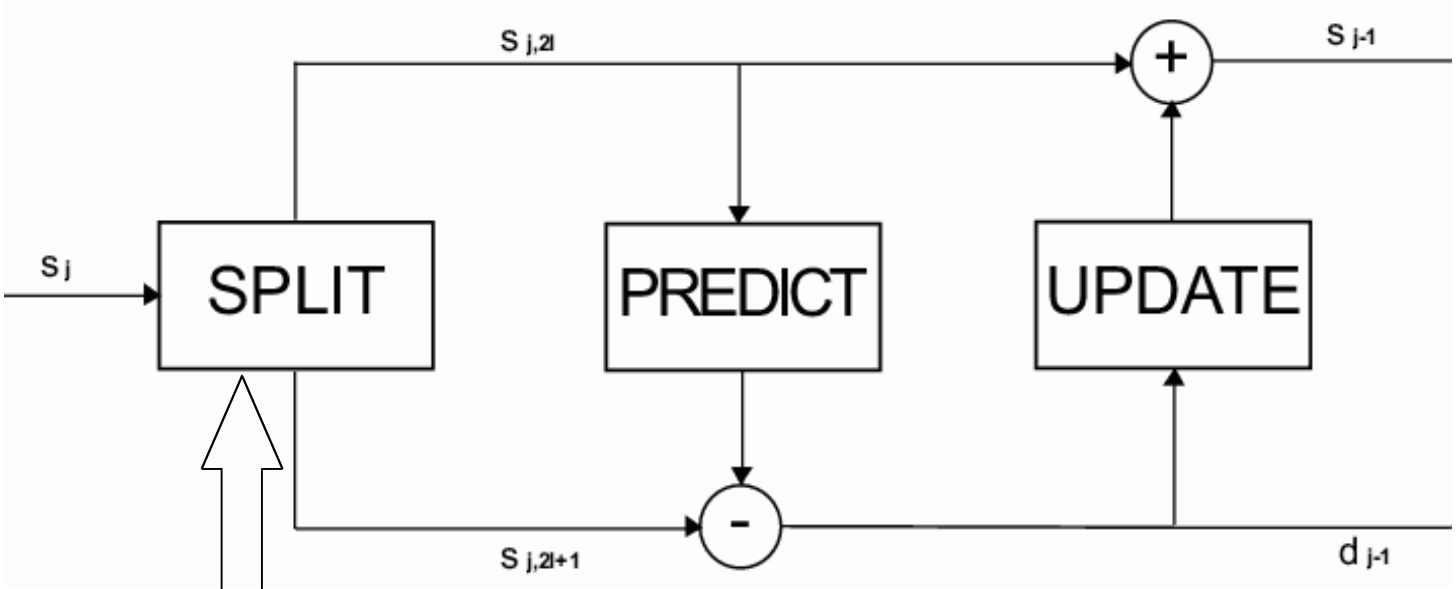
HIGH−PASS filter



Construction using *Lifting*

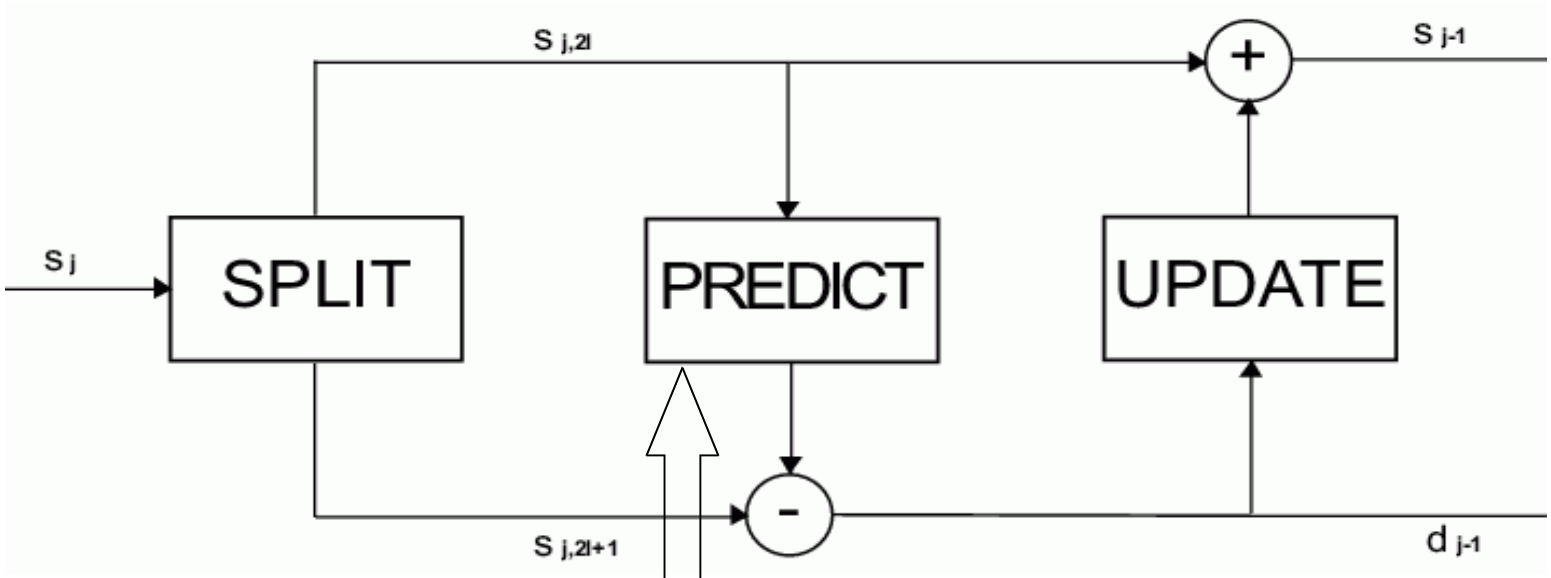


Construction using *Lifting*



$$\begin{cases} s_{j-1} = \text{even samples} \\ d_{j-1} = \text{odd samples} \end{cases}$$

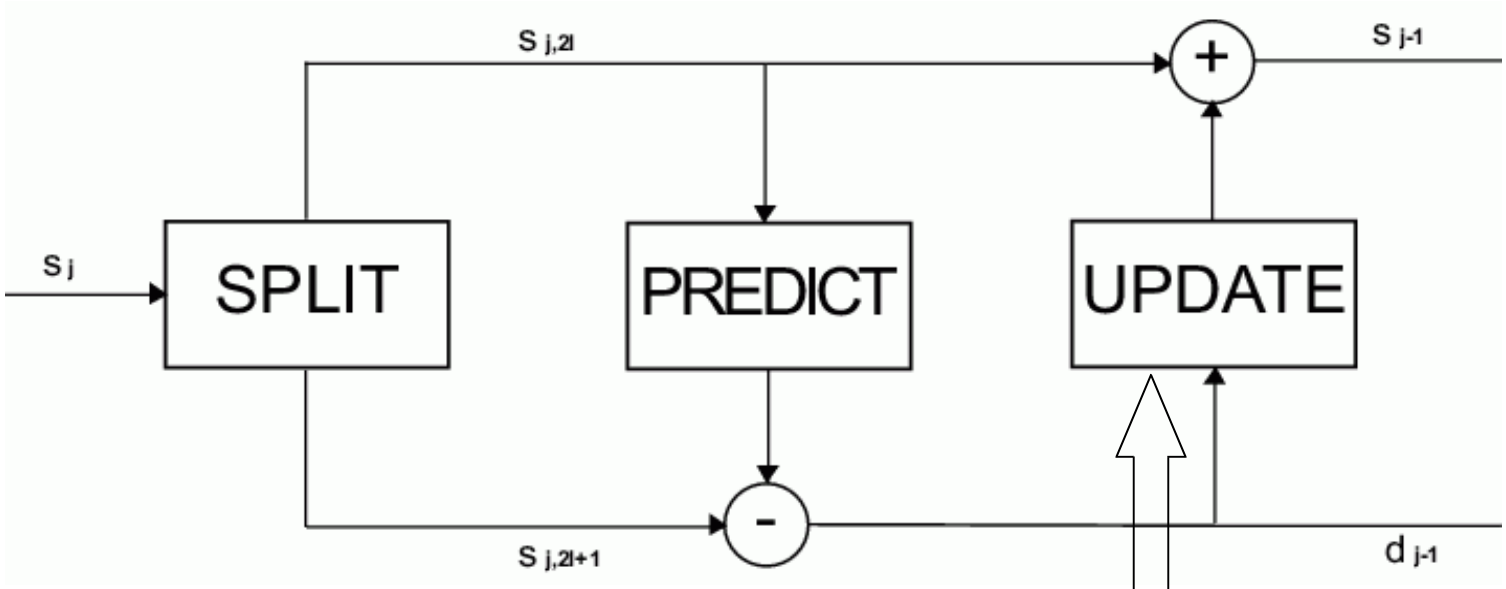
Construction using *Lifting*



$$d_{j-1} = d_{j-1} - \mathbf{P}(s_{j-1})$$

$$\mathtt{d}_{j-1}\mathtt{-=}\mathbf{P}(\mathtt{s}_{j-1})$$

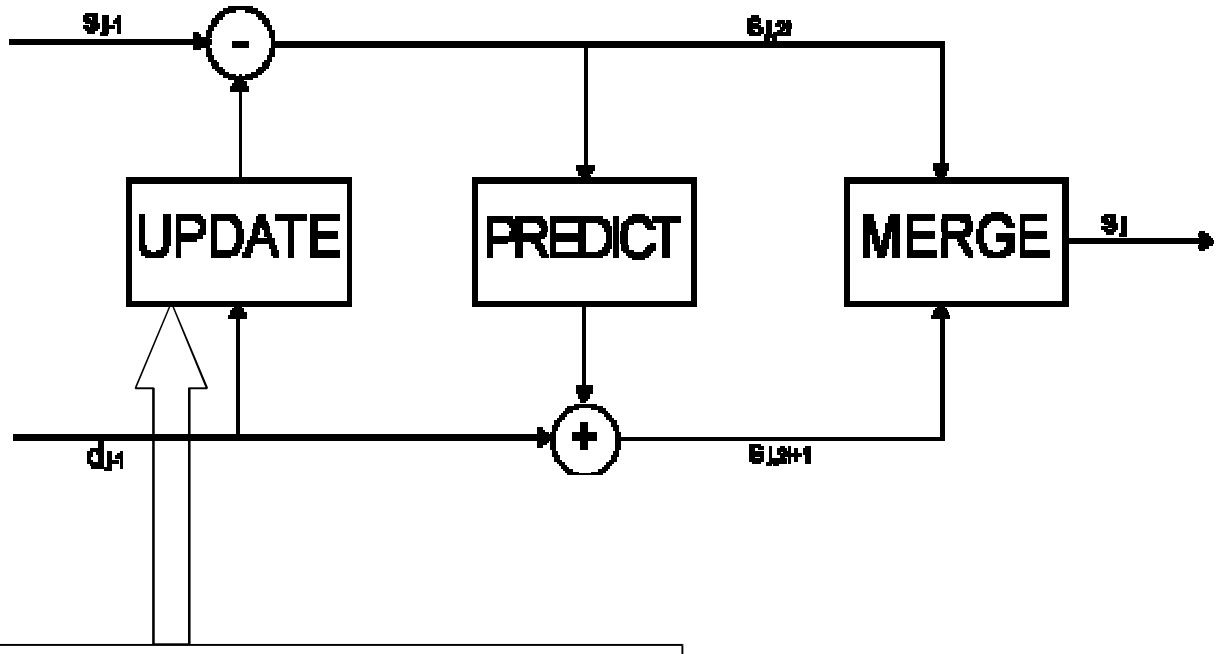
Construction using *Lifting*



$$s_{j-1} = s_{j-1} + \mathbf{U}(d_{j-1})$$

$$\mathbf{s}_{j-1}\mathbf{+=}\mathbf{U}(\mathbf{d}_{j-1})$$

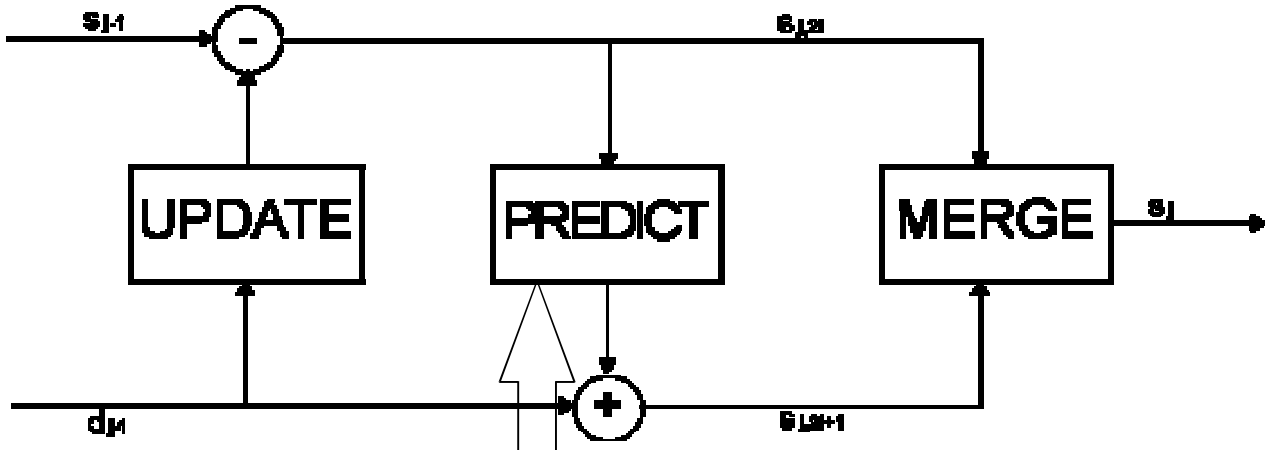
Inverse Transform



$$s_{j-1} = s_{j-1} - \mathbf{U}(d_{j-1})$$

$$\mathtt{s}_{j-1} \mathtt{-=} \mathbf{U}(\mathtt{d}_{j-1})$$

Inverse Transform



$$d_{j-1} = d_{j-1} + \mathbf{P}(s_{j-1})$$

$$\mathtt{d}_{j-1}\mathtt{+=}\mathbf{P}(\mathtt{s}_{j-1})$$

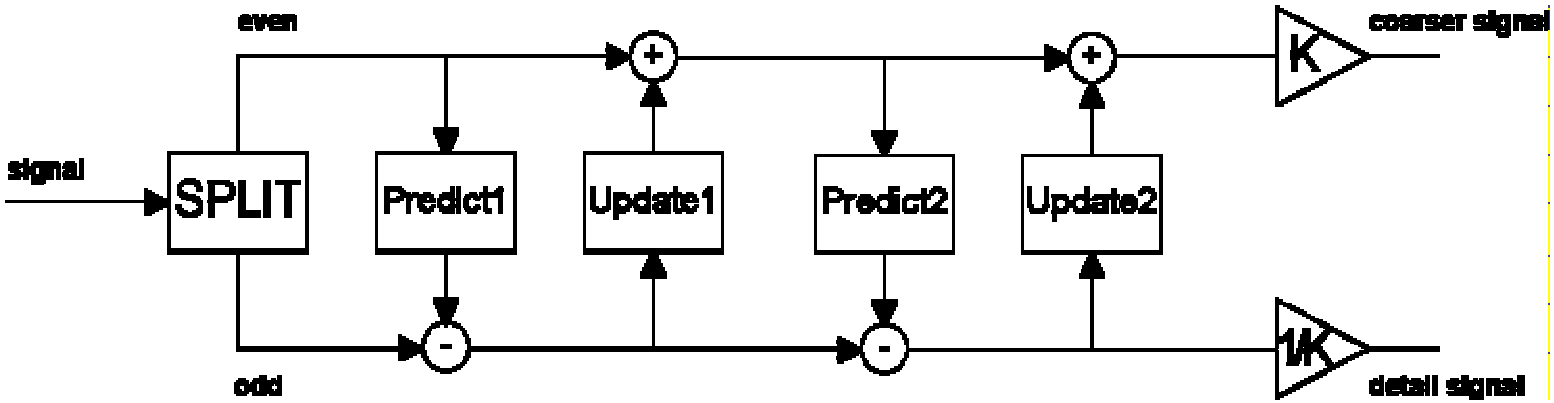
Advantages of *Lifting*

- **In−place** computation
- **Parallelism**
- **Efficiency**: about half the operations of the convolution algorithm
- **Inverse Transform** : follows immediately by reversing the coding steps

Factoring a subband transform into *Lifting* steps (Daubechies, Sweldens)

Theorem: Every subband transform with FIR filters can be obtained as a splitting step followed by a finite number of predict and update steps, and finally a scaling step.

Application: (9−7) filter pair



Application: (9,7) filters

$$s_l^{(0)} = x_{2l}$$

$$d_l^{(0)} = x_{2l+1}$$

Predict 1 :

$$d_l^{(1)} = d_l^{(0)} + \alpha(s_l^{(0)} + s_{l+1}^{(0)})$$

Update 1 :

$$s_l^{(1)} = s_l^{(0)} + \beta(d_l^{(1)} + d_{l-1}^{(1)})$$

Predict 2 :

$$d_l^{(2)} = d_l^{(1)} + \gamma(s_l^{(1)} + s_{l+1}^{(1)})$$

Update 2 :

$$s_l^{(2)} = s_l^{(1)} + \delta(d_l^{(2)} + d_{l-1}^{(2)})$$

$$s_l = K\, s_l^{(2)}$$

$$d_l = d_l^{(2)}/K.$$

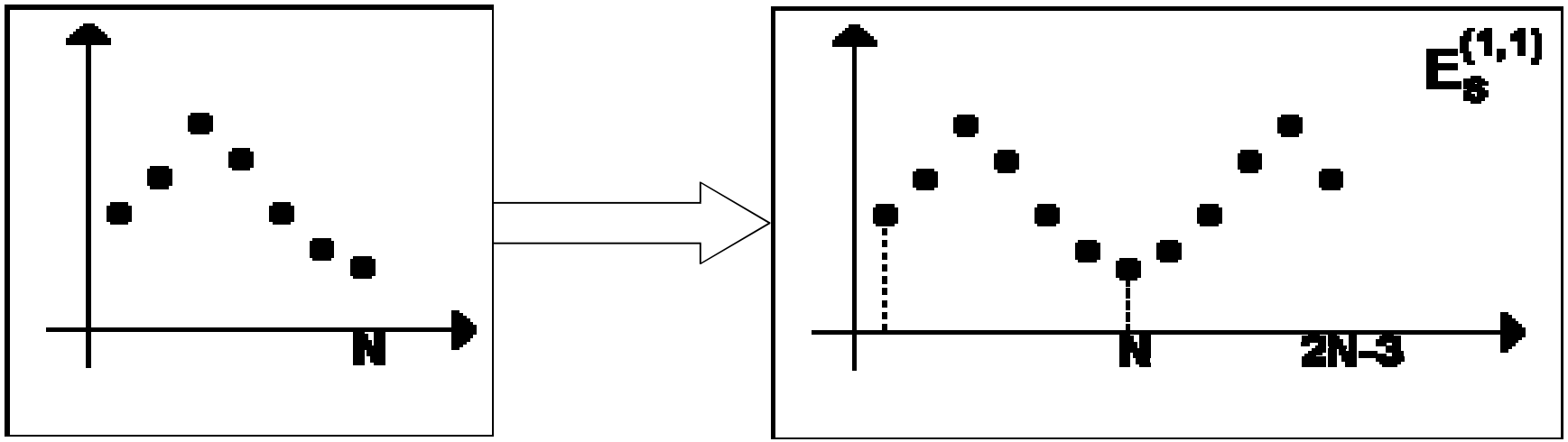
with

$\alpha \approx -1.586134342$
$\beta \approx -0.05298011854$
$\gamma \approx 0.8829110762$
$\delta \approx 0.4435068522$
$K \approx 1.149604398$

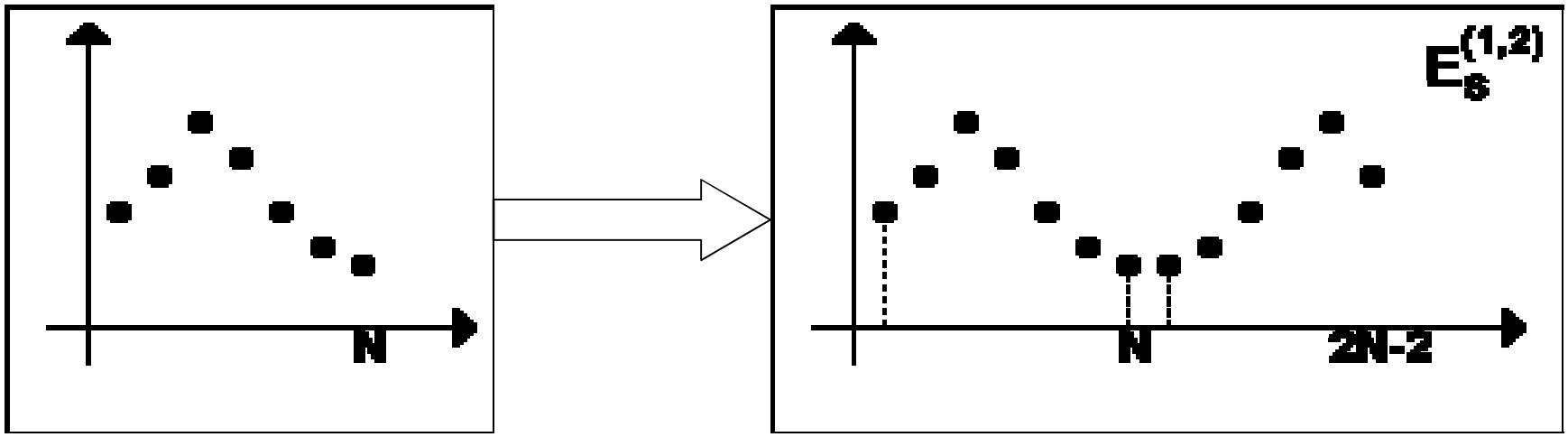
Boundary problems with finite length signals

- Applying the (9,7) filters to a finite length signal *x(n)* requires samples outside of the original support of *x*
- Taking the infinite periodic extension of *x* may introduce a jump discontinuity
- With symmetric biorthogonal filters, we can use nonexpansive symmetric extensions

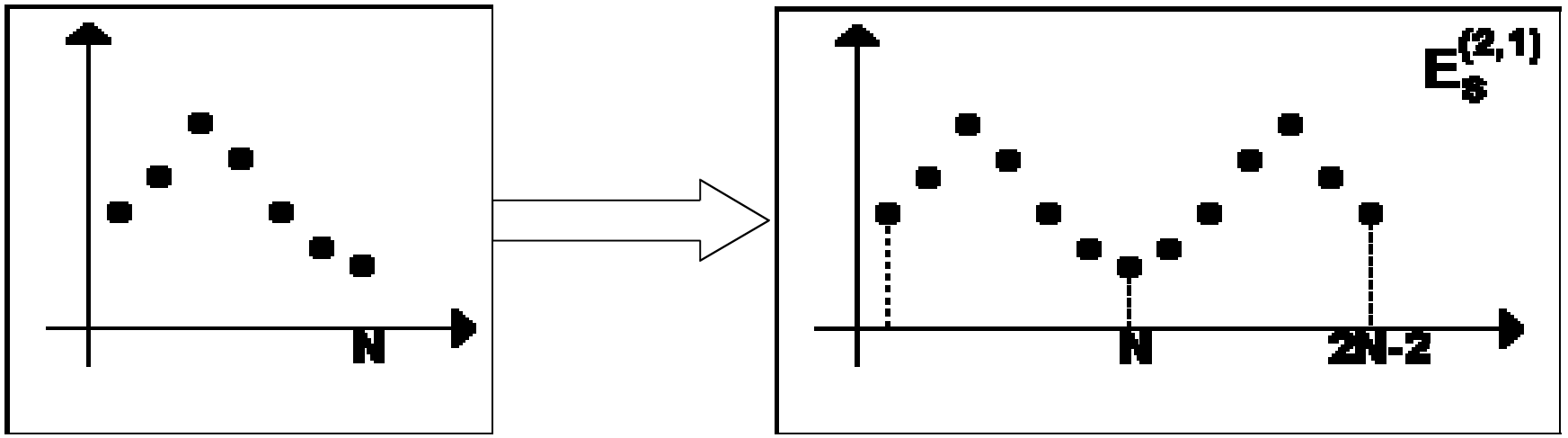
symmetric extension operators



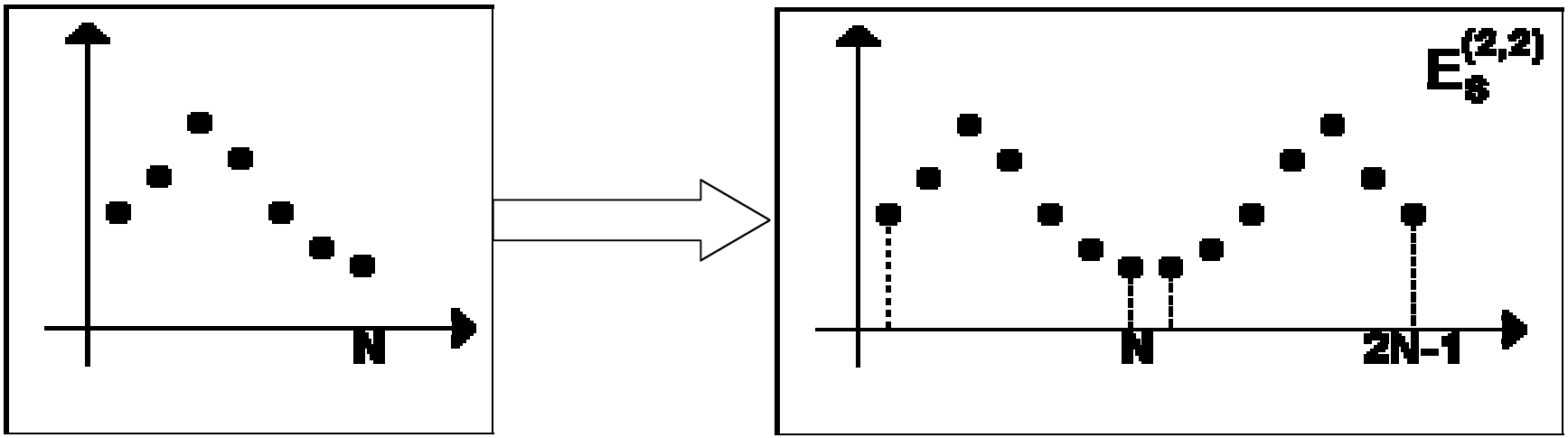
symmetric extension operators



symmetric extension operators



symmetric extension operators



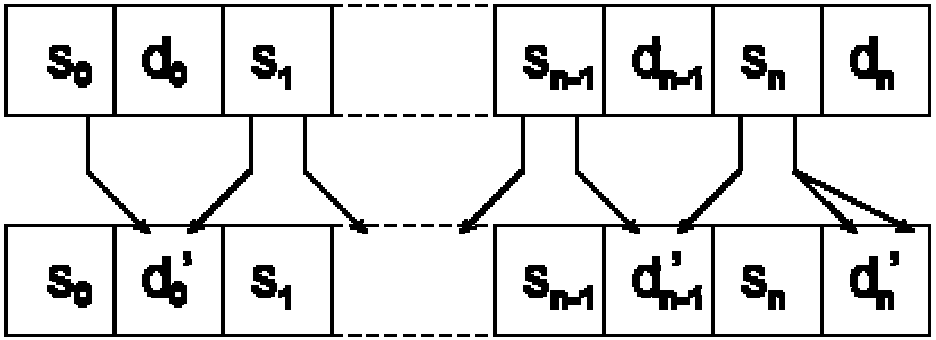
For 2 −subband filters symmetric about one of their taps, use the Es*(1,1)* extension for both forward and inverse transforms

Signal x length N_0	Filter h Center	Subband Center	Subband Dimension	Shift	Synthesis extension
Even	2ν	ν	$N_0/2$	ν	$E_S^{(1,2)}$
	$2\nu+1$	$\nu+1/2$	$N_0/2$	$\nu+1$	$E_S^{(2,1)}$
Odd	2ν	ν	$(N_0+1)/2$	ν	$E_S^{(1,1)}$
	$2\nu+1$	$\nu+1/2$	$(N_0-1)/2$	$\nu+1$	$E_S^{(2,2)}$

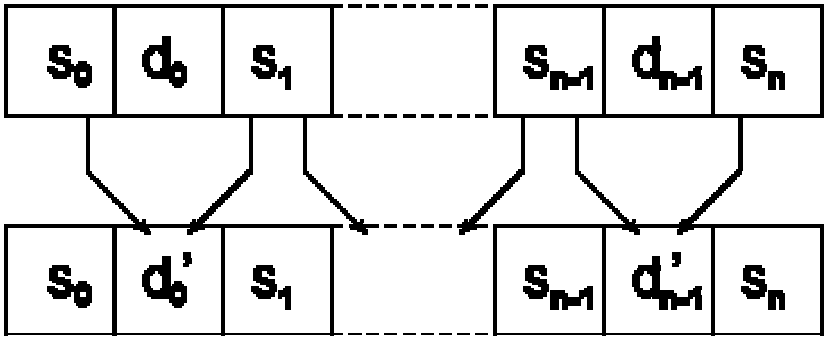
Symmetric extension and *Lifting*

PREDICT

n even



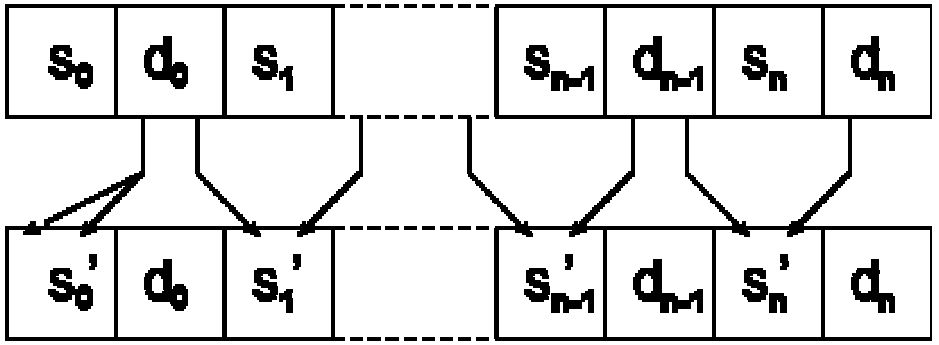
n odd



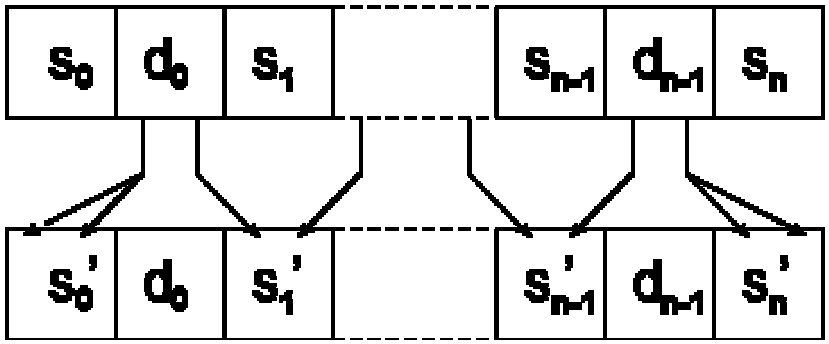
Symmetric extension and *Lifting*

UPDATE

n even



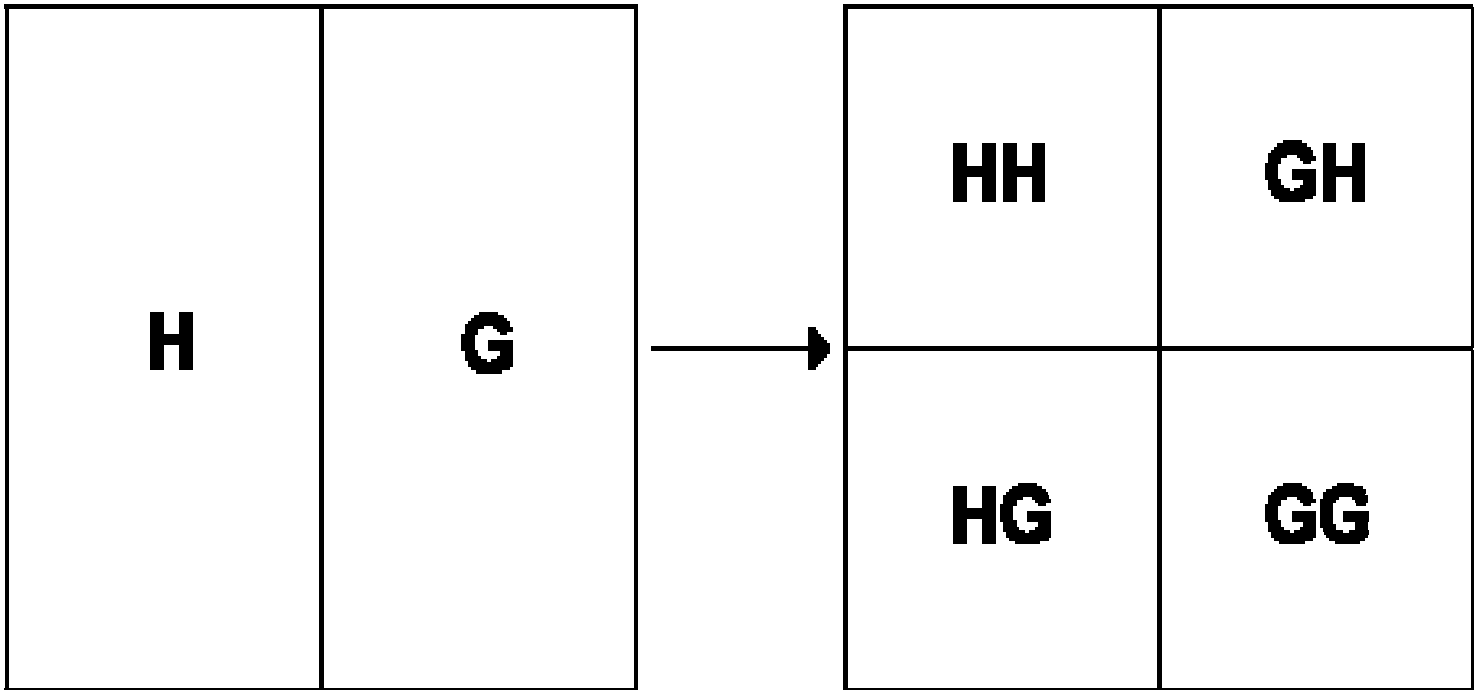
n odd



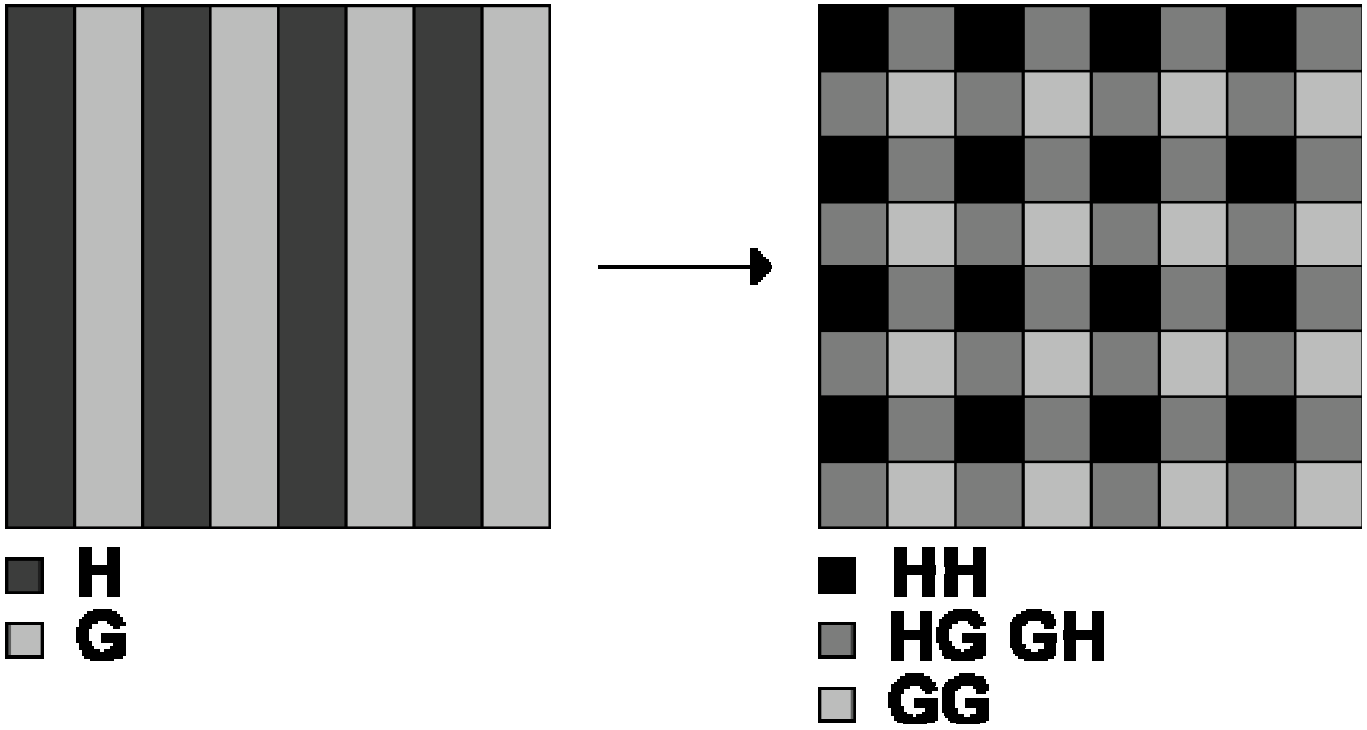
Extension to the 2D case

- Horizontal and vertical directions are treated separately
- Apply the 1D wavelet transform to rows, and then to columns, in either order
  => **4 subbands: *HH, HG, GH, GG***
- Reapply the filtering transformation to the ***HH*** subband, which corresponds to the coarser representation of the original image

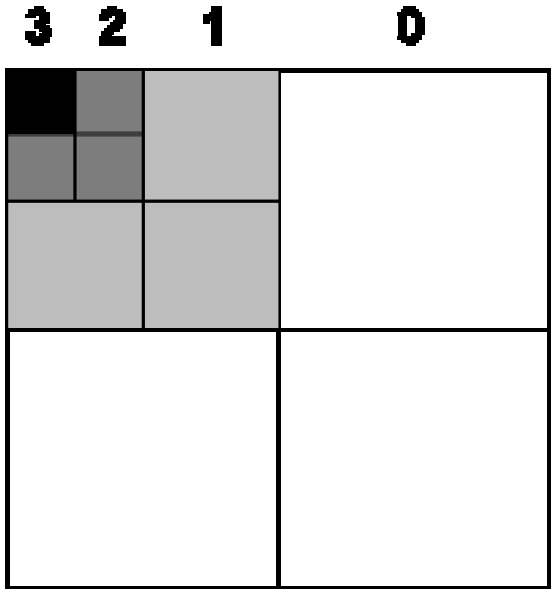
Extension to the 2D case

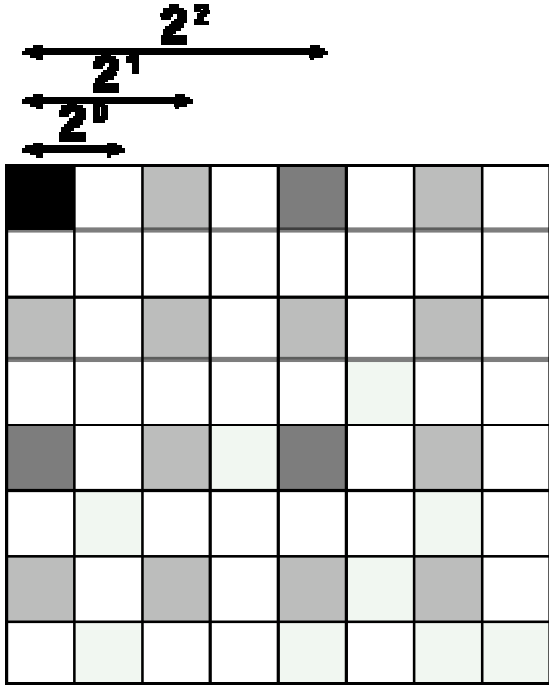


In−place computation



Pyramidal structure





IN PLACE

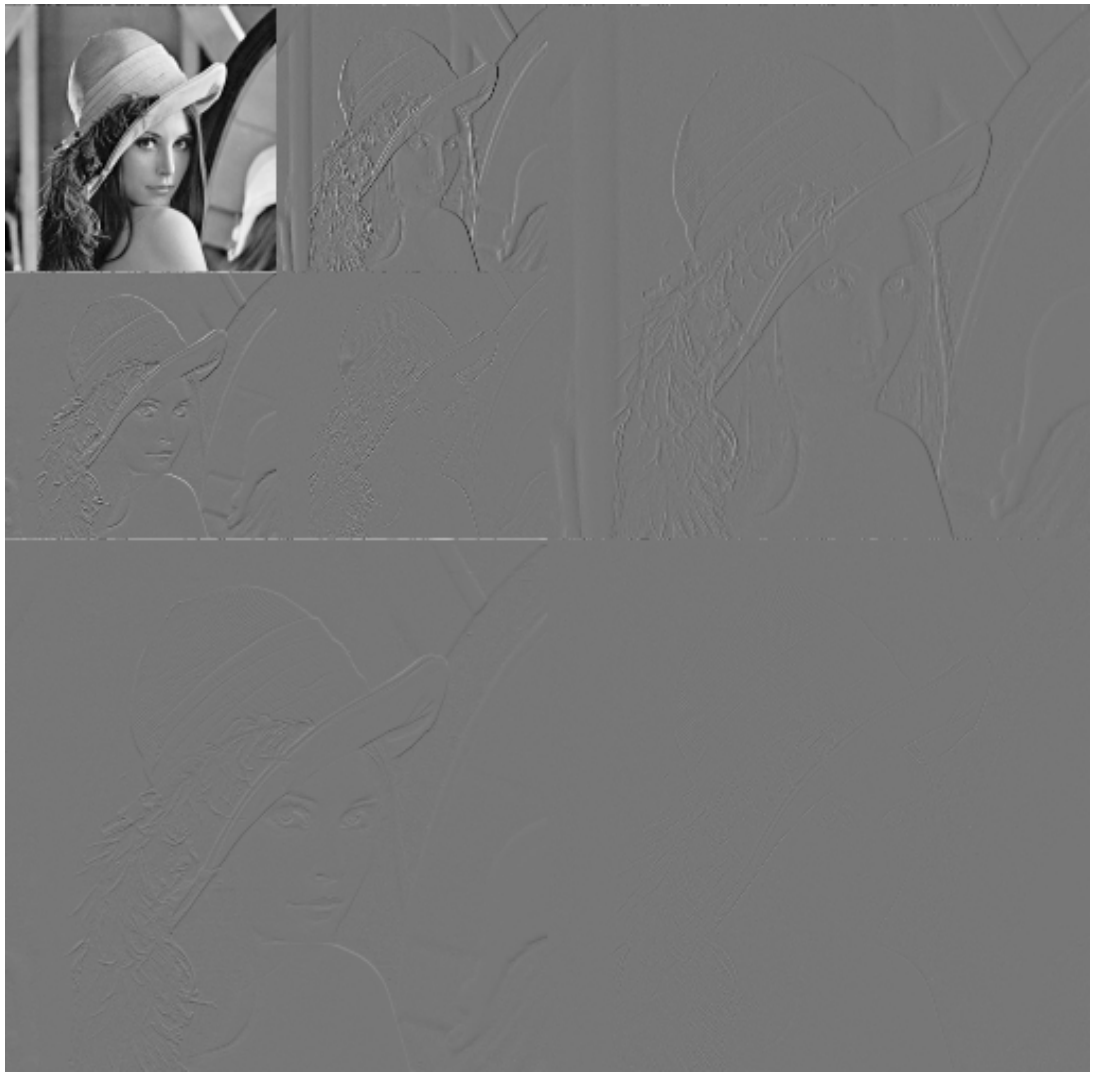
Multiscale representation

- For coefficients organized by subbands: if *(i,j)* belongs to scale **k**, then *(2i,2j)*, *(2i+1,2j)*, *(2i,2j+1)*, *(2i+1,2j+1)* belong to scale **k−1**
- For coefficients are computed in place: *(i,j)* belongs to scale **min(k,l)** where **k** (respectively **l**) is the number of 2s in the prime factorization of *i* (respectively *j*)

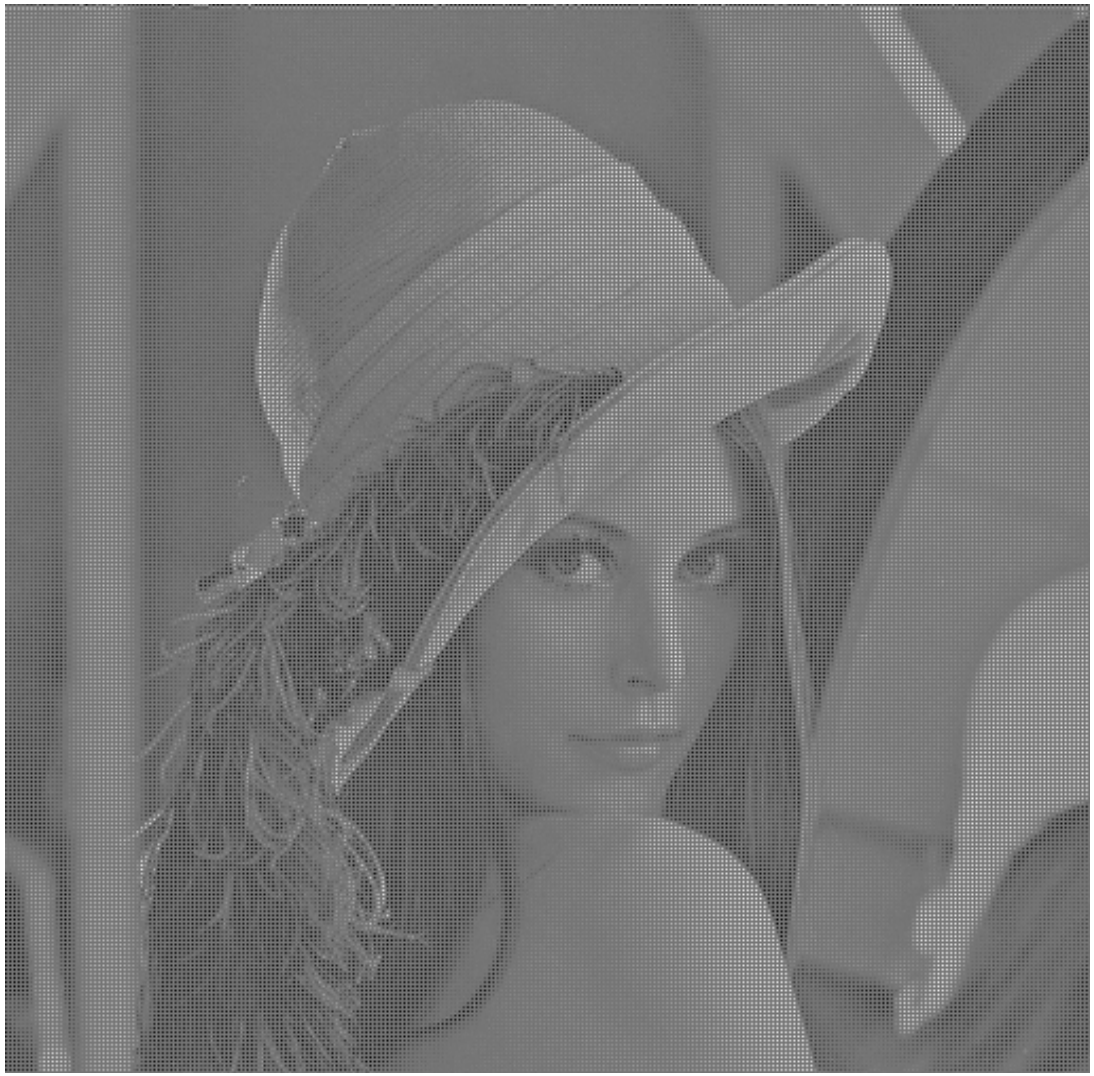
Example



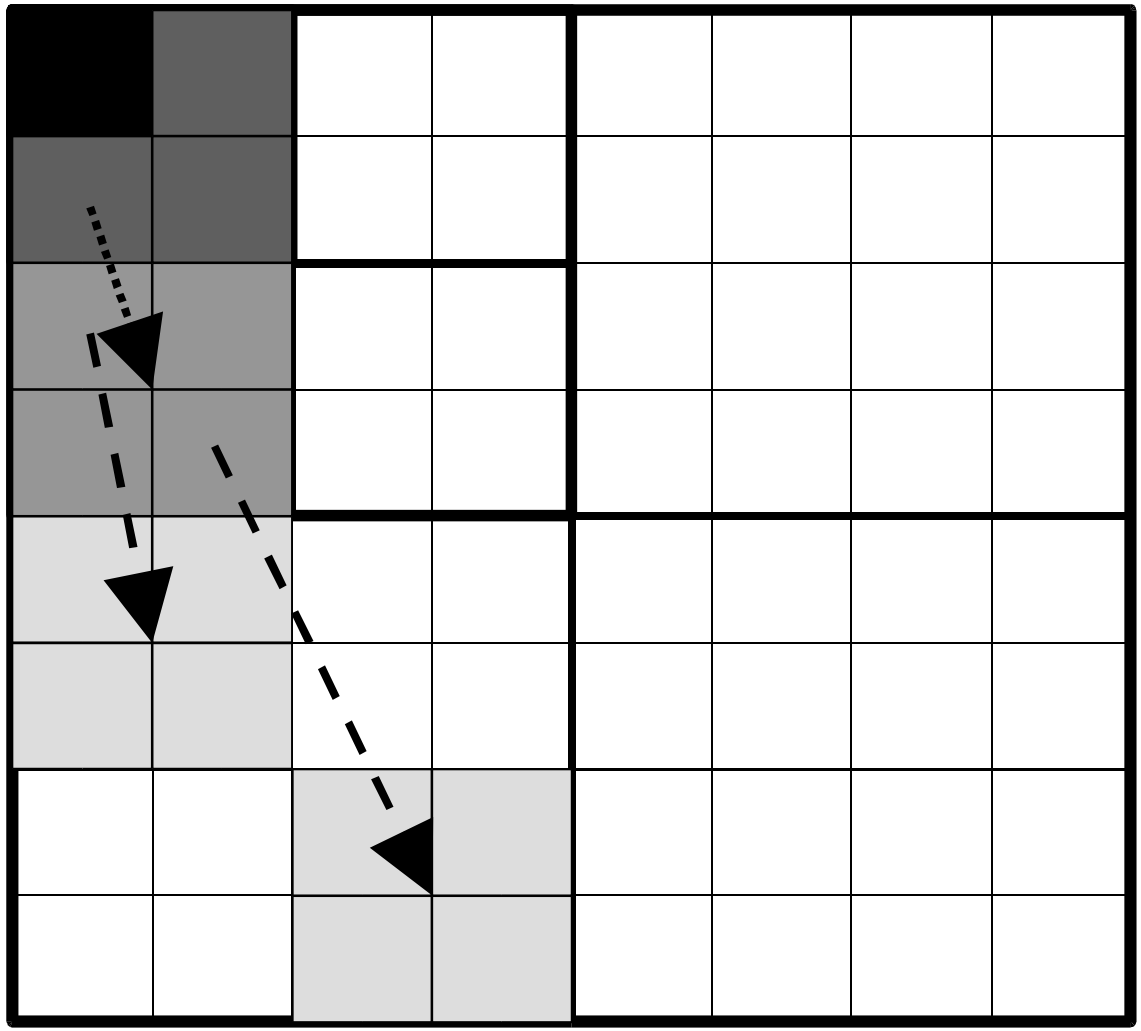
Example



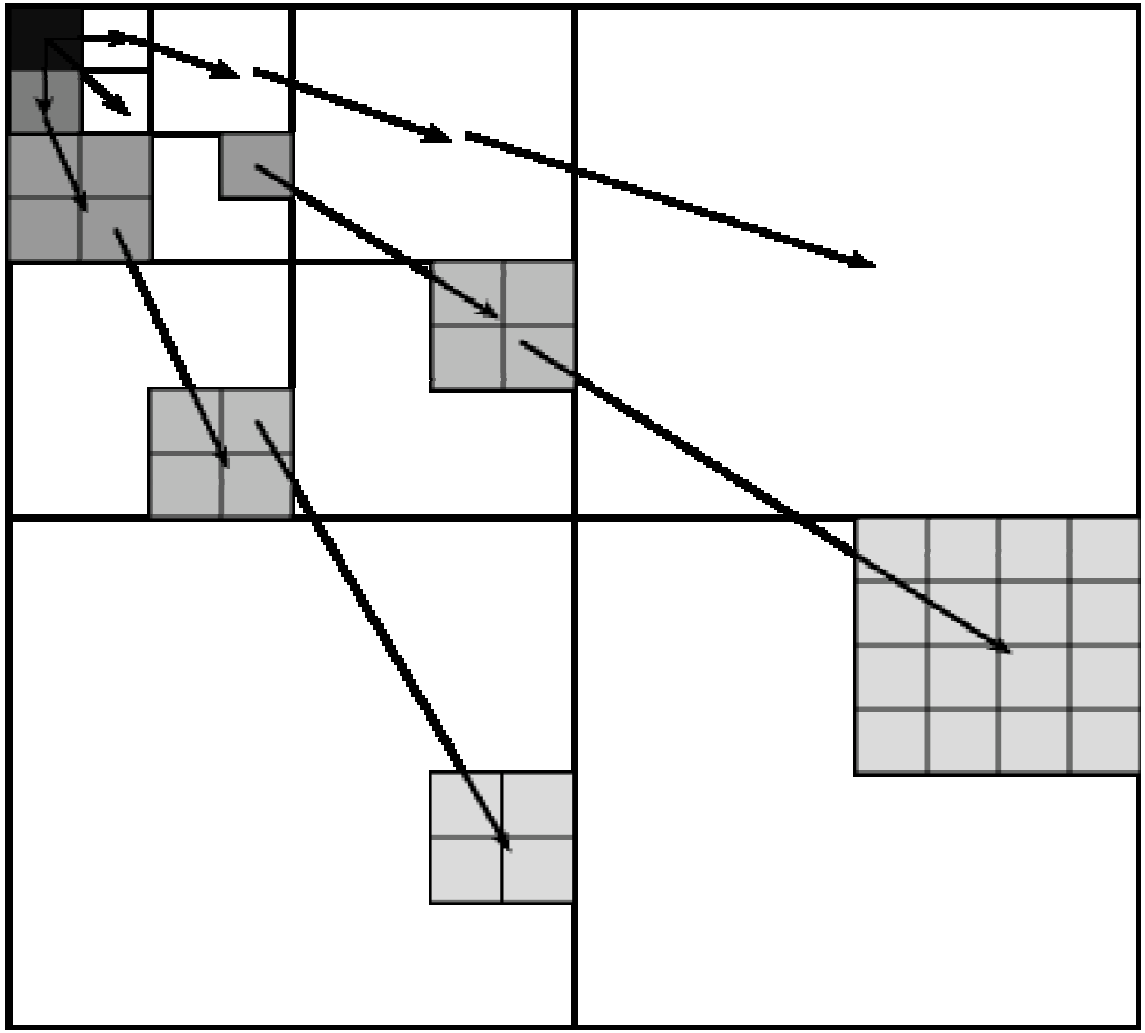
Example: In−Place



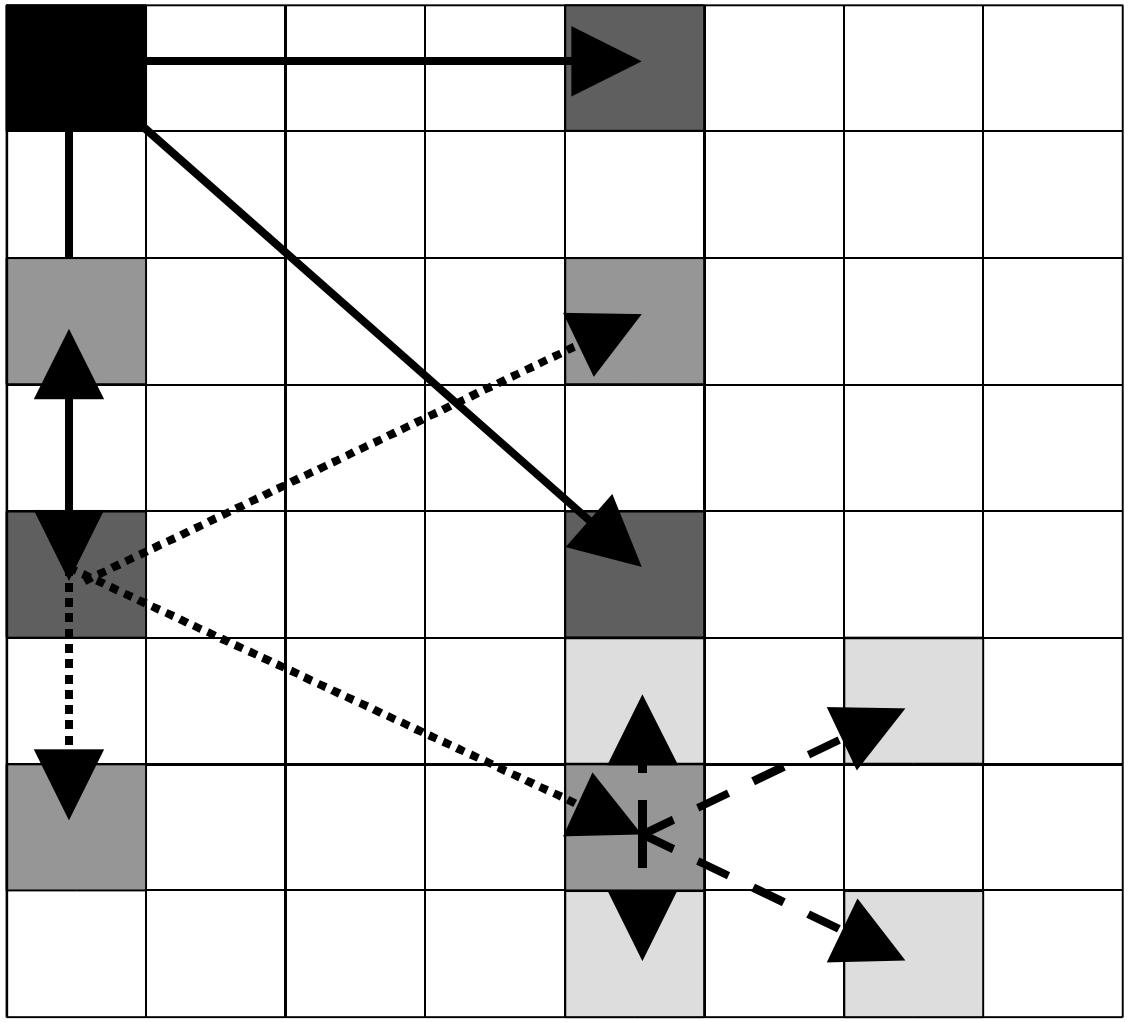
Spatial Orientation Trees



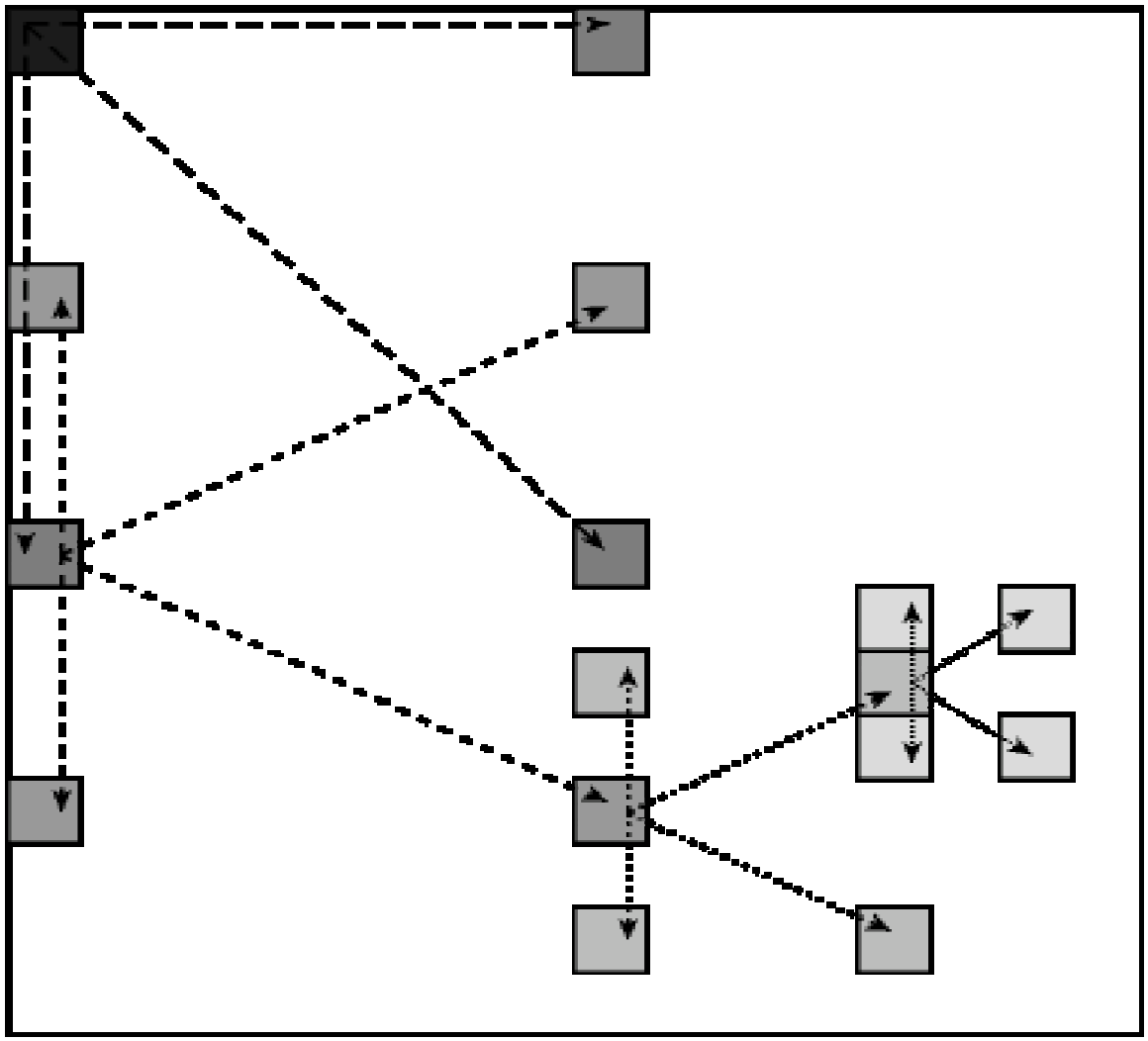
Spatial Orientation Trees



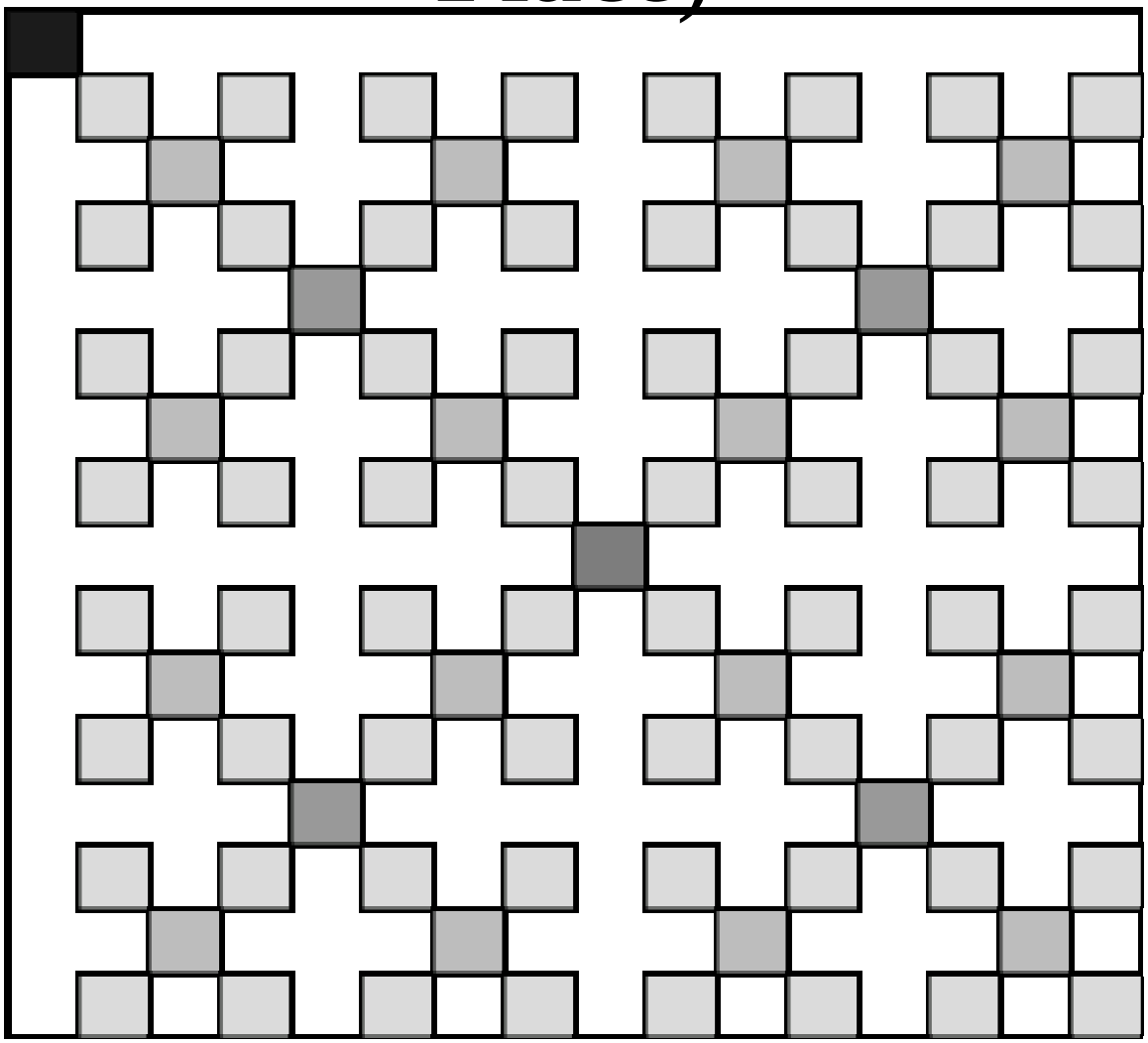
Spatial Orientation Trees (**In Place**)



Spatial Orientation Trees (**In Place**)



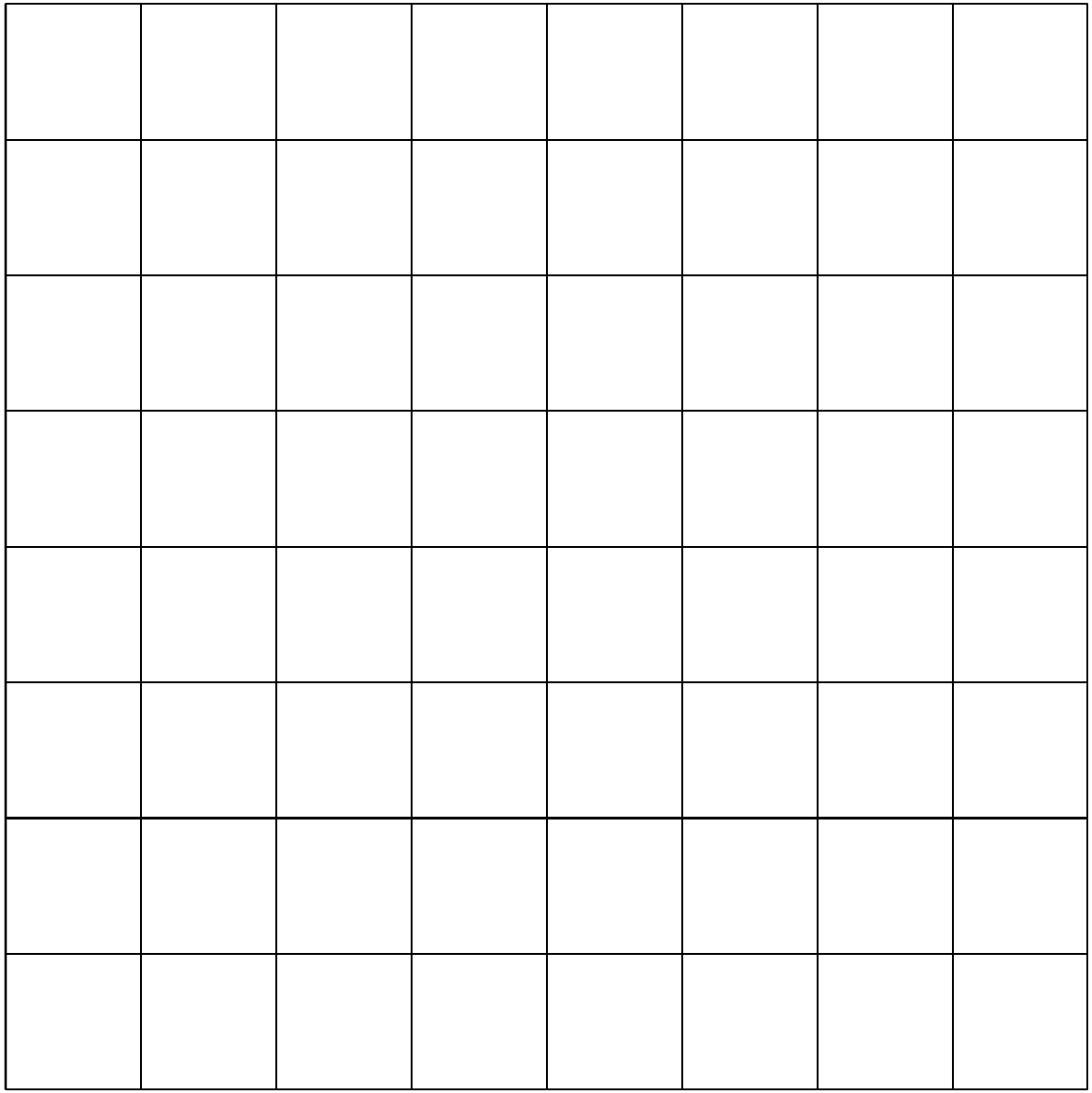
Spatial Orientation Trees (**In Place**)



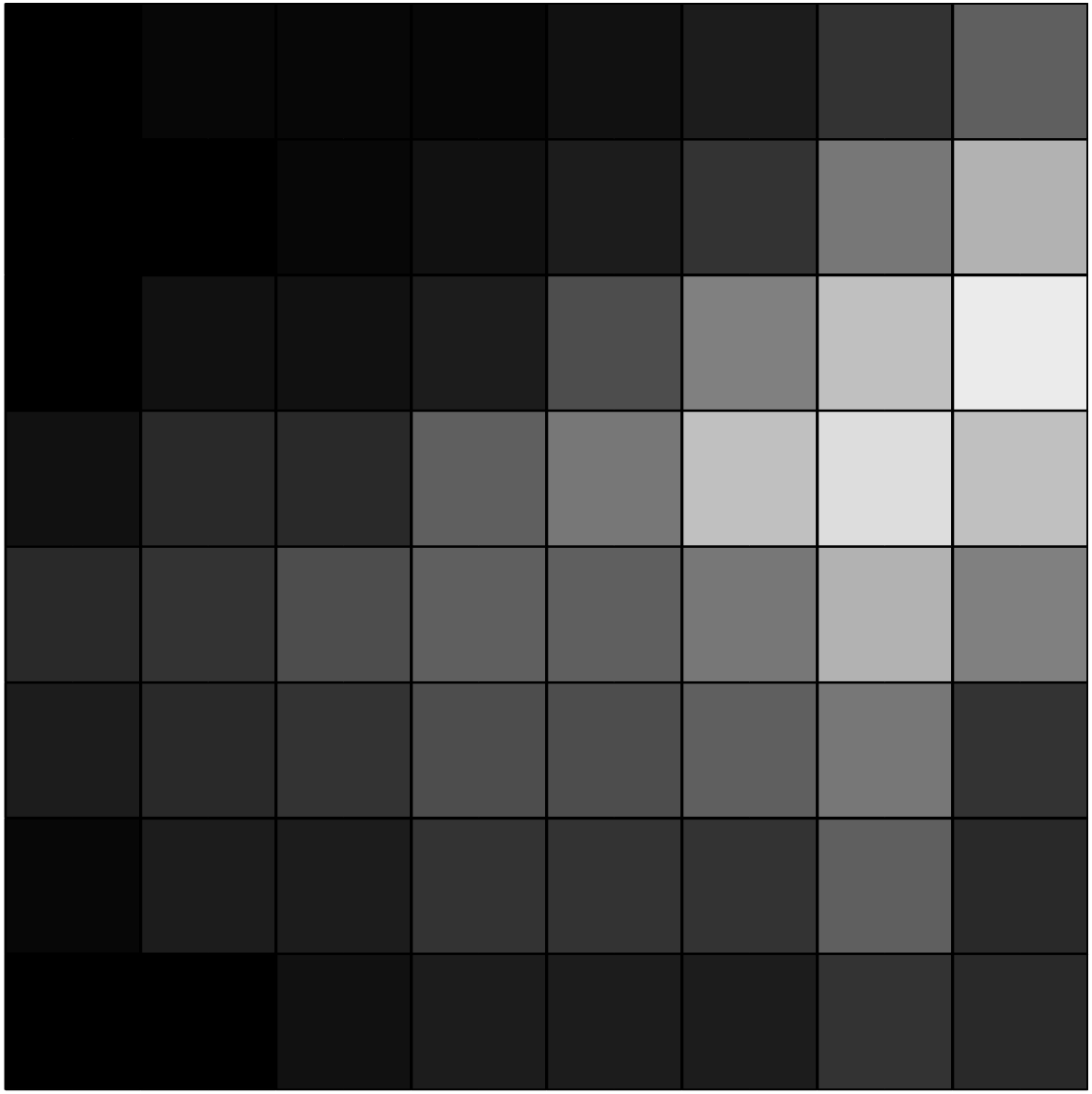
Experimental Facts

- Most of an image's energy is concentrated in the low frequency components, thus the variance is expected to decrease as we move down the tree
- If a wavelet coefficient is insignificant, then all its descendants in the tree are expected to be insignificant

A small example: 8x8 sample



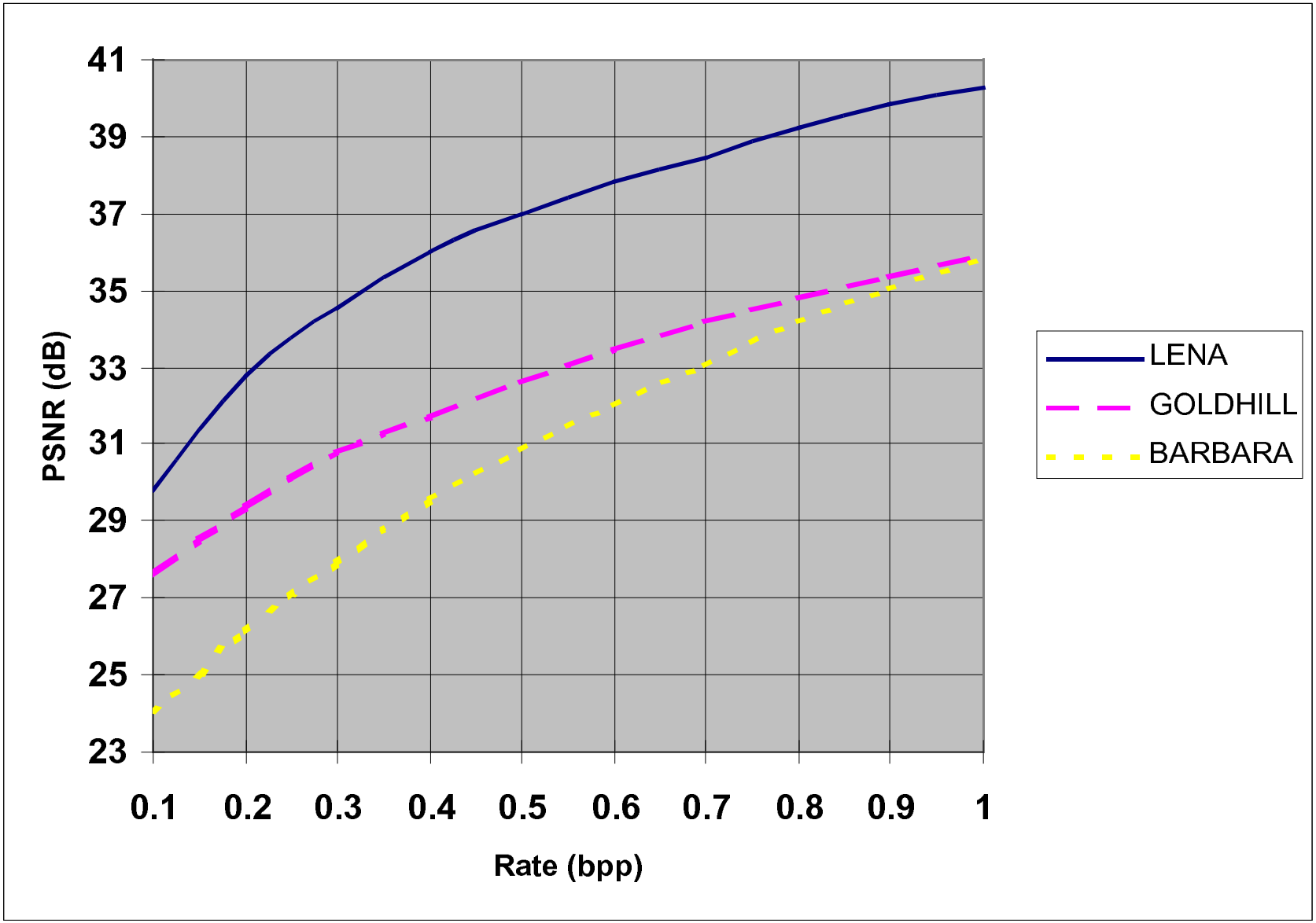
Grayscale picture, 4 bits/pixel



Average : 4.9

0	1	1	1	2	3	5	7
0	0	1	2	3	5	8	11
0	2	2	3	6	9	12	14
2	4	4	7	8	12	13	12
4	5	6	7	7	8	11	9
3	4	5	6	6	7	8	5
1	3	3	5	5	5	7	4
0	0	2	3	3	3	5	4

Results : PSNR(rate)



Original : lena.pgm, 8bpp, 512x512



Compression rate: 160, 0.05bpp; PSNR = 27.09dB



Compression rate: 80, 0.1bpp; PSNR = 29.80dB



Compression rate: 64, 0.125bpp; PSNR = 30.64dB



Compression rate: 32, 0.25bpp; PSNR = 33.74dB



Compression rate: 16, 0.5bpp; PSNR = 36.99dB



Compression rate: 8, 1.0bpp; PSNR = 40.28dB



Compression rate: 4, 2.0bpp; PSNR = 44.61dB



Original : barbara.pgm, 8bpp, 512x512



Compression rate: 32, 0.25bpp; PSNR = 27.09dB



Compression rate: 16, 0.5bpp; PSNR = 30.85dB



Compression rate: 8, 1.0bpp; PSNR = 35.82dB



Compression rate: 4, 2.0bpp; PSNR = 41.94dB



Original : goldhill.pgm, 8bpp, 512x512



Compression rate: 32, 0.25bpp; PSNR = 30.17dB



Compression rate: 16, 0.5bpp; PSNR = 32.58dB



Compression rate: 8, 1.0bpp; PSNR = 35.87dB



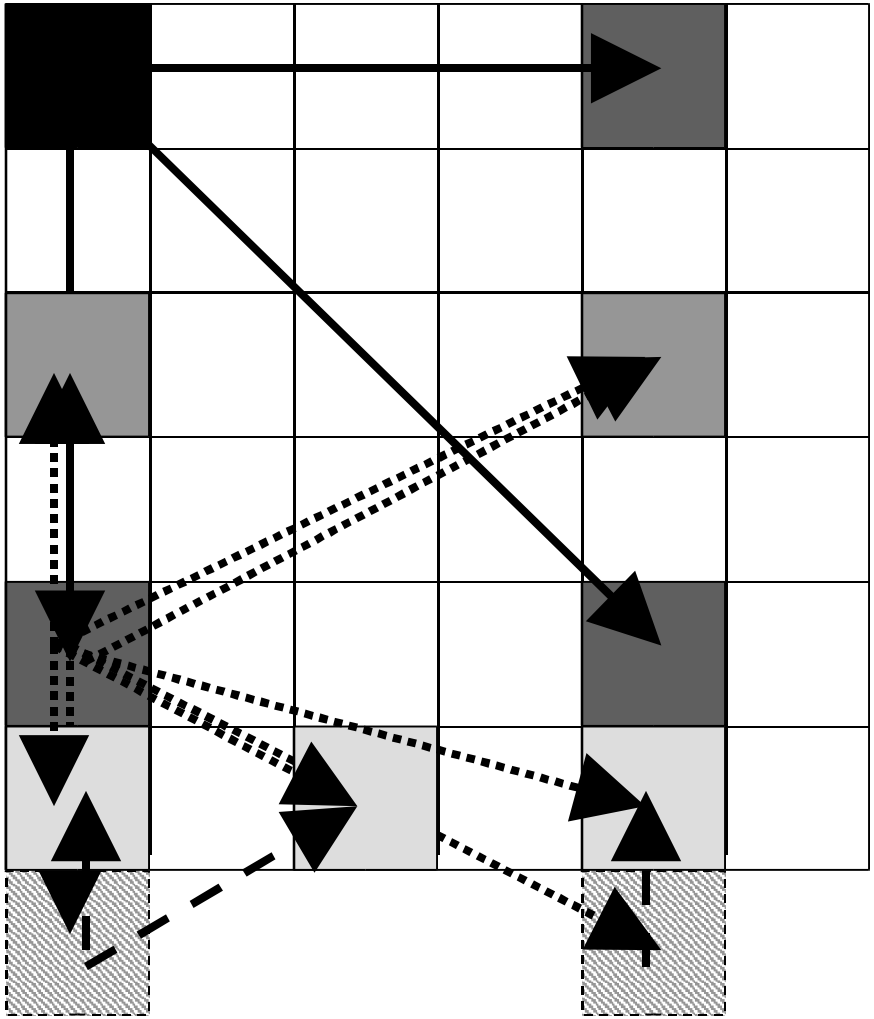
Compression rate: 4, 2.0bpp; PSNR = 40.95dB



Image height or width is not a power of 2?

- If a row or a column has an odd number ***N*** of samples, the transform will lead to ***(N+1)/2*** coefficients for the ***H*** subband or ***(N−1)/2*** for the ***G*** subband.
- Let $l = \min(width, height)$; if $2^{n-1} < l \leq 2^{n}$, then the subband pyramid will have ***n*** different detail levels, and the spatial orientation tree will have depth ***n***.
- If the width or the height is not an integer power of 2, some detail subbands at certain scales will have fewer coefficients than if width and height were padded up to the next integer power of 2.

Example



Image's height or width is not a power of 2?

<u>Idea :</u> If a node ***(i,j)*** has a son outside of the picture, look for further descendants of this one that come back into the picture, and also considers them as sons of ***(i,j)***

Colored Pictures

- A colored picture can be represented as a triplet of 2D arrays corresponding to the colors **(Red,Green,Blue)**
- The coder performs the same linear transform as JPEG does, changing ***(R,G,B)*** into ***(Y,Cr,Cb)***, to get 1 luminance and 2 chrominance channels
- The human eye is much more sensitive to variations in luminance than to variations in either of the chrominance channels
- In the following examples, 90% of the output data is dedicated to the luminance channel

Original : lena.ppm, 24bpp, 512x512



Compression rate: 128, 0.1875bpp;



Compression rate: 64, 0.375bpp;



Compression rate: 32, 0.75bpp;



Compression rate: 16, 1.5bpp;



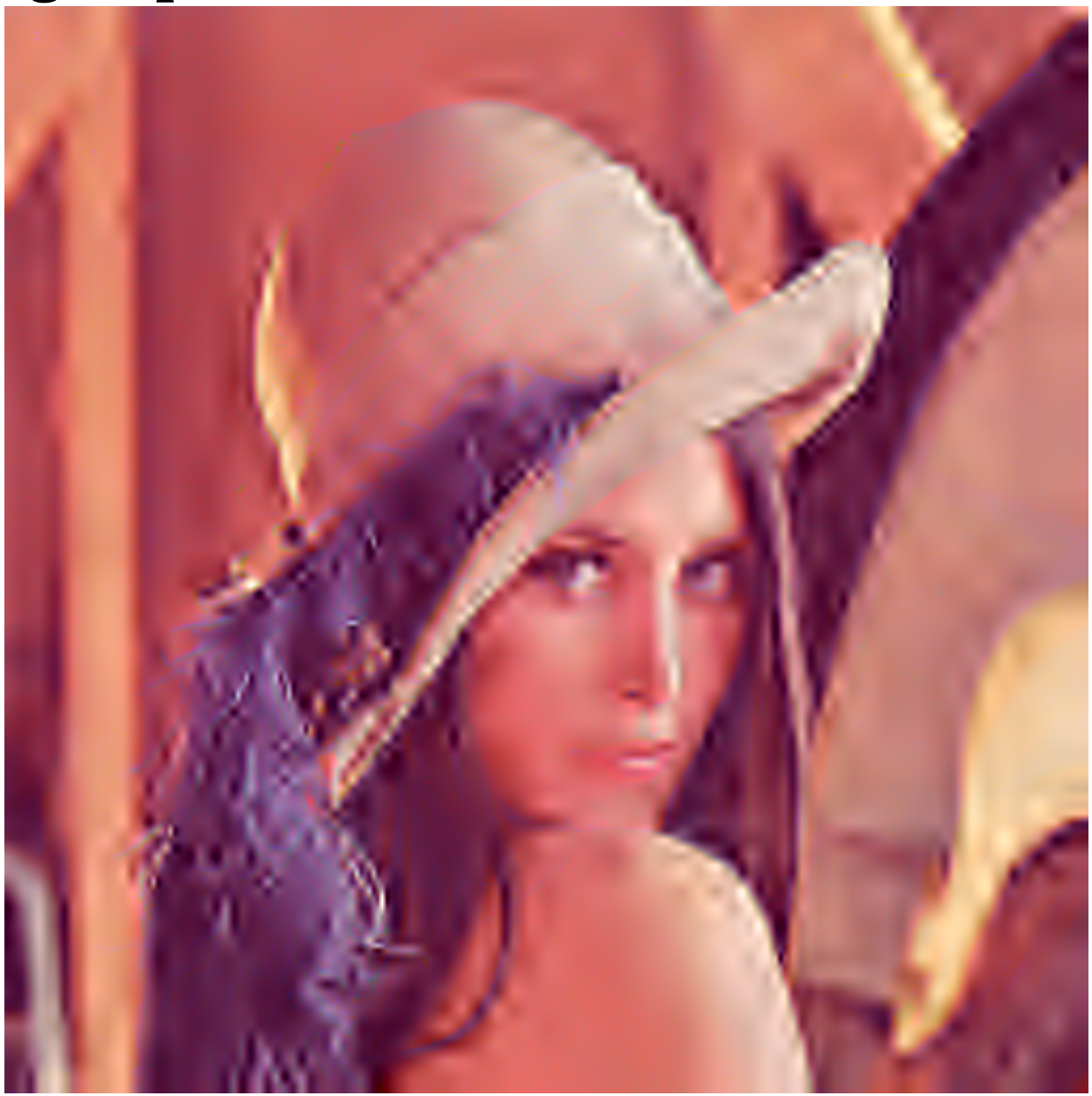
Compression rate: 8, 3.0bpp;



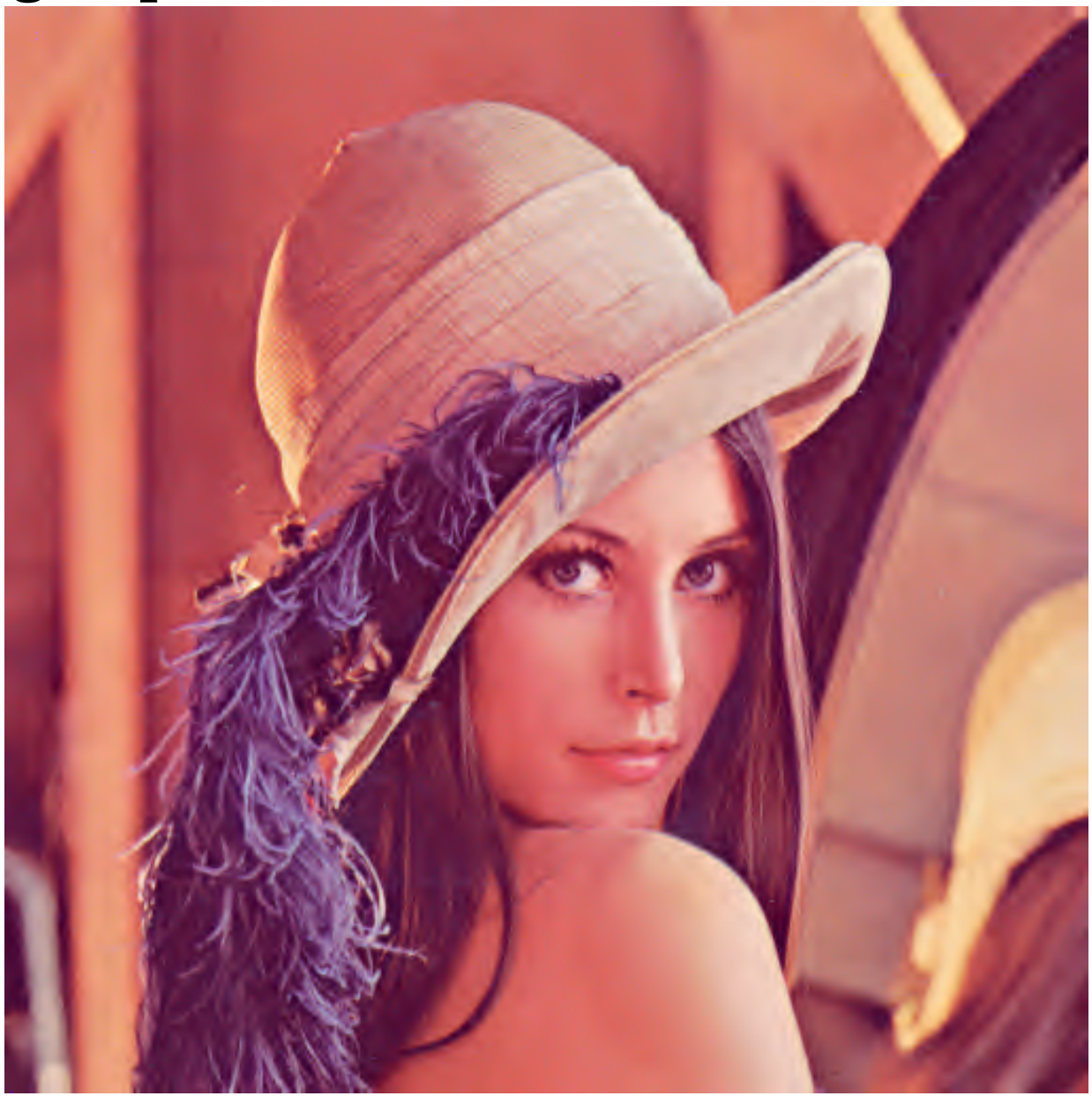
Compression rate: 4, 6.0bpp;



Compression rate: 8, 3.0bpp;percentage of bits budget spent of the luminance channel = 1%



Compression rate: 8, 3.0bpp;percentage of bits budget spent of the luminance channel = 10%



Compression rate: 8, 3.0bpp;percentage of bits budget spent of the luminance channel = 50%



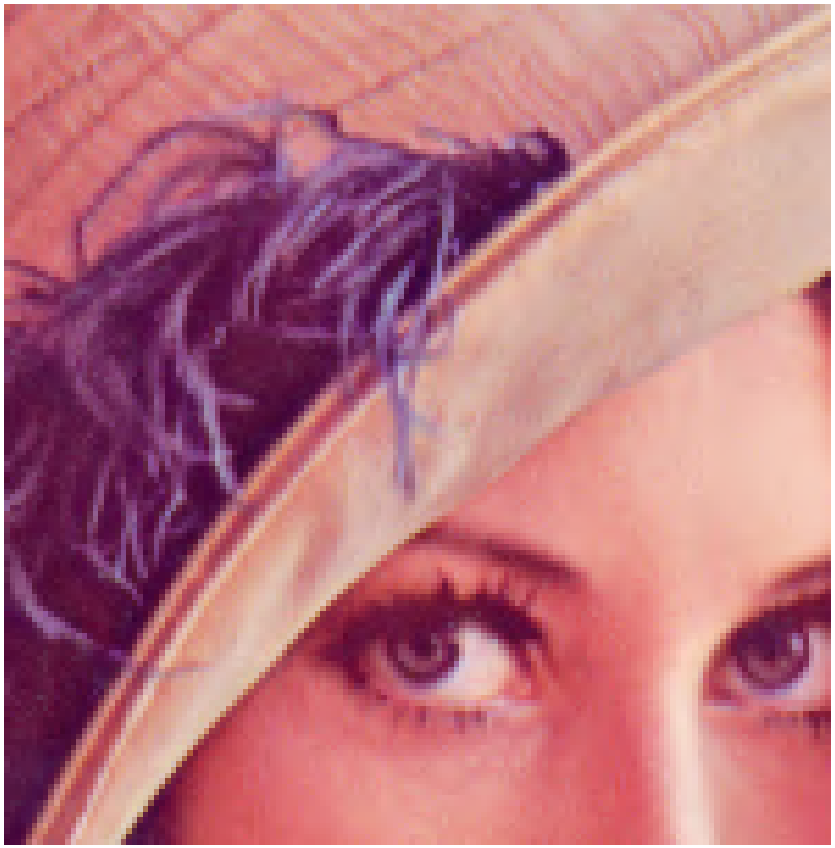
Compression rate: 8, 3.0bpp;percentage of bits budget spent of the luminance channel = 90%



Compression rate: 8, 3.0bpp;percentage of bits budget spent of the luminance channel = 99%



ZOOM



50%



99%

Sharpening Filters

- Idea: a better PSNR does not always mean a better looking picture. Even for grayscale pictures, the human eye does not exactly see the images of difference
- Problem: especially at low bit rates, reconstructed pictures look too smooth, with subjective loss of contrast
- Fix: letting $\boldsymbol{c}' = (2\boldsymbol{I} - \boldsymbol{H})\,\boldsymbol{c}$ is one way to reverse the effects of applying a smoothing filter $\boldsymbol{H}$ to $\boldsymbol{c}$

Compression rate: 32, sharpened
loss of PSNR = 1.4dB



Compression rate: 16, sharpened
loss of PSNR = 2.75dB



Compression rate: 8, sharpened
loss of PSNR = 5.11dB



Compression rate: 16
COMPARISON



sharpened



unsharpened