The National University of Singapore
Inter-Faculty Industrial Seminar
2 September 1996

Highly Scalable Wavelet-Based Video Compression for Very Low Bitrate Environment

by

Tham Jo Yew

Wavelets Strategic Research Programme
Department of Mathematics
National University of Singapore

E-mail: thamjy@wavelets.math.nus.edu.sg
Webpage: http://wavelets.math.nus.edu.sg/thamjy

♠ Video Compression

♢ *Why must compress everything ??*

↪ Consider the NTSC color video:
- ∗ 720 pixels x 480 lines
- ∗ 8 bits/pixel (bpp) per color
- ∗ 30 frames per second (fps)

↪ This requires a whopping **237** Mbits per second!!

♢ *Wow, then it must be hungry for disk space ??*

↪ A typical CD has about 650 Mbytes

↪ This means only ≈ *20 secs.* of NTSC video!!

♢ *What about transmission bottlenecks & time ??*

↪ Transfer rates of CD-ROM devices (300 kbps - 1.5 Mbps)

↪ This is far *too low* for full-motion display!!

↪ Typical telephone lines (p x 64 kbps, where p is small)

↪ Even high-end modems (28.8 kbps or higher)

↪ Can say "bye-bye" to video conferencing, video phone, tele-shopping, video-on-demand etc.

♢ *So how much compression is needed ??*

↪ *E.g.* A CD can now store ≈ *one hour* of 200:1 compressed NTSC video

↪ Video phone applications - only ≈ 5-20 kbps! (Voice ?)

♠ Scalable Videos

♢ *Huh? What? Scalable ??*

↪ Consider large image database browsing, and video playback over heterogenous networks
↪ Users have different requirements and constraints
↪ Avoid storing *multiple copies* at the database server

♢ *So how does scalability address these problems ??*

↪ Store only *one* copy of a full-resolution and high bit-rate *scalable video*
↪ Different subsets can then be extracted from the same compressed bit stream **after** it has been generated

♢ *I see! What are some useful scalability features?*

- **Bit-rate/distortion -** exchanging video quality for different video bandwidths
- **Spatial resolution -** choosing different sizes (heights and widths) of the video
- **Temporal resolution -** selecting different frame rates
- **Hardware complexity -** varying the CPU and memory requirements for both the transmitter and receiver
- **End-to-end delay -** controlling the coding delays (useful for real-time interactive applications)

Outlines of Talk

- Overview of Encoder and Decoder
- Motion Estimation and Compensation
  - Three-Parameter Motion Model
  - Fast Center-Biased Diamond Search
- Three-Dimensional Wavelet Decomposition Framework
  - Temporal and Spatial Decomposition
  - Subband (Parent-Child) Relationships
  - Formation of TRI-ZTRs
- Progressive Video Coding via TRI-ZTRs
  - Primary Pass
  - Secondary Pass
  - Embedded and Scalable Compressed Bit Stream
- Video Scalability and Re-Scalability
  - Multirate Scaling
  - Multiresolution Scaling
- Performance Analysis
- Comparisons with MPEG-2, H.263 and JPEG
- Pointers for Further References

♠ Overview of Encoder and Decoder

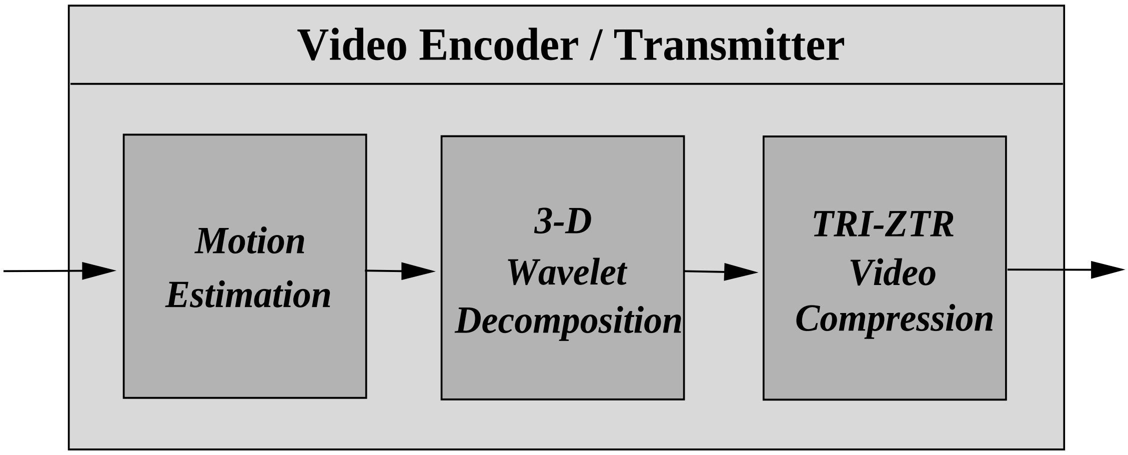


Figure 1: Overview of the encoder/transmitter.

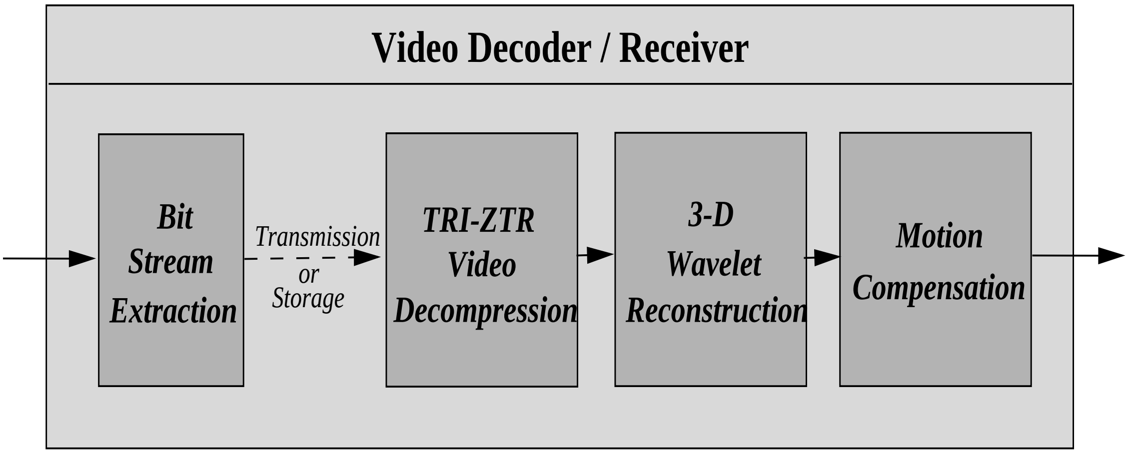


Figure 2: Overview of the decoder/receiver.

♠ Motion Estimation and Compensation

♢ *Is it MEMC? For what ??*

↪ Video sequence (motion picture) vs. Still image
↪ *Main difference:* Interframe motions
↪ Exploit such temporal correlations for interframe coding via MEMC
↪ Many different techniques are available:
E.g. block-based, pel-recursive, motion models, Bayesian, optical flow, etc.

♢ *Well, which one you use in your video codec ??*

↪ The *three-parameter* motion model
↪ Good motion estimation with lower motion overheads
↪ Compensate both camera zoom and translations

♢ *What is this model actually? What do we want ??*

↪ The transformation/mapping function:

$$\begin{aligned} u(x, y, a) &= a_1 x + a_2 \\ v(x, y, a) &= a_1 y + a_3, \end{aligned} \tag{1}$$

where pixels (x, y) are mapped to (u, v)

$\hookrightarrow$ Estimate vector $a = [a_1, a_2, a_3]^T$ s.t. distortion function

$$E_N^{(i,j,n)}(a) = \sum_{x,y \in B_N^{(i,j,n)}} \sum \mathrm{TPD}^2[x, y, a], \tag{2}$$

where

$$\mathrm{TPD}[x, y, a] = I_n[u(x, y, a), v(x, y, a)] - I_0[x, y], \tag{3}$$

is minimized.

$\diamondsuit$ ***So how to estimate? Exhaustive search, huh ??***

$\hookrightarrow$ No, we use the Gauss-Newton iteration method

$$a^{(k+1)} = a^{(k)} - 2[B(a^{(k)})]^{-1}[J(a^{(k)})]^T[f(a^{(k)})], \tag{4}$$

where

$$[f(a^{(k)})] = \begin{bmatrix} \mathrm{TPD}\,(0, 0, a^{(k)}) \\ \mathrm{TPD}\,(0, 1, a^{(k)}) \\ \mathrm{TPD}\,(0, 2, a^{(k)}) \\ \vdots \\ \mathrm{TPD}\,(N-1, N-1, a^{(k)}) \end{bmatrix}, \tag{5}$$

and k denotes the iteration number.

$\hookrightarrow$ J := **Jacobian** matrix

$\hookrightarrow$ First order derivatives of the TPD w.r.t. a_1, a_2, and a_3

$$[J(a^{(k)})]^T = \begin{bmatrix} \frac{\partial TPD\,(0,0,a^{(k)})}{\partial a_1} & \cdots & \frac{\partial TPD\,(N-1,N-1,a^{(k)})}{\partial a_1} \\ \frac{\partial TPD\,(0,0,a^{(k)})}{\partial a_2} & \cdots & \frac{\partial TPD\,(N-1,N-1,a^{(k)})}{\partial a_2} \\ \frac{\partial TPD\,(0,0,a^{(k)})}{\partial a_3} & \cdots & \frac{\partial TPD\,(N-1,N-1,a^{(k)})}{\partial a_3} \end{bmatrix}. \tag{6}$$

$\hookrightarrow$ Using numerical differentiation and chain rule,

$$\begin{aligned} J_{(x,y),i}(a^{(k)}) &= \frac{\partial \mathrm{TPD}(x,y,a^{(k)})}{\partial a_i} \\ &= G_u.\frac{\partial u}{\partial a_i} + G_v.\frac{\partial v}{\partial a_i}, \end{aligned} \tag{7}$$

where the spatial gradients, G_u and G_v, are given by

$$\begin{aligned} G_u &= \frac{\partial \mathrm{TPD}(x,y,a^{(k)})}{\partial u} \\ &= \frac{\partial}{\partial u}\left[I_n[u(x,y,a^{(k)}),v(x,y,a^{(k)})] - I_0[x,y]\right] \\ &\approx \frac{1}{2}I_n[u(x,y,a^{(k)})+1,v(x,y,a^{(k)})] \\ &\quad -\frac{1}{2}I_n[u(x,y,a^{(k)})-1,v(x,y,a^{(k)})], \end{aligned} \tag{8}$$

and, similarly,

$$\begin{aligned} G_v &\approx \frac{1}{2}I_n[u(x,y,a^{(k)}),v(x,y,a^{(k)})+1] \\ &\quad -\frac{1}{2}I_n[u(x,y,a^{(k)}),v(x,y,a^{(k)})-1]. \end{aligned} \tag{9}$$

$\hookrightarrow$ Matrix B := **Hessian** matrix
$\hookrightarrow$ Second derivatives of $E(a^{(k)})$ w.r.t. a_1, a_2, and a_3

$$[B(a^{(k)})] = \begin{bmatrix} b_{11}(a^{(k)}) & b_{12}(a^{(k)}) & b_{13}(a^{(k)}) \\ b_{21}(a^{(k)}) & b_{22}(a^{(k)}) & b_{23}(a^{(k)}) \\ b_{31}(a^{(k)}) & b_{32}(a^{(k)}) & b_{33}(a^{(k)}) \end{bmatrix}, \qquad (10)$$

in which its elements are given by

$$\begin{aligned} b_{ij}(a^{(k)}) &= \frac{\partial^2}{\partial a_i \partial a_j} \sum_{x,y \in B_N}\sum [\,\mathrm{TPD}(x, y, a^{(k)})]^2 \\ &= \sum_{x,y \in B_N}\sum \frac{\partial}{\partial a_i} \left\{ \frac{\partial}{\partial a_j} [\,\mathrm{TPD}(x, y, a^{(k)})]^2 \right\} \\ &= \sum_{x,y \in B_N}\sum \frac{\partial}{\partial a_i} \left\{ 2 \,.\, \mathrm{TPD}(x, y, a^{(k)}) \,.\, J_{(x,y),j}(a^{(k)}) \right\} \\ &= 2 \sum_{x,y \in B_N}\sum J_{(x,y),i}(a^{(k)}) \,.\, J_{(x,y),j}(a^{(k)}) \end{aligned}$$

$\Diamond$ ***What strategy to use next ??***

$\hookrightarrow$ *Global motion* - the entire frame as a block
$\hookrightarrow$ *Local motion* - divide each frame into smaller blocks
$\hookrightarrow$ Simultaneous global-local motion estimation

$\Diamond$ ***Can show something more intuitive or not ??***

$\hookrightarrow$ Sure, consider some standard sequences
$\hookrightarrow$ See Figs. 3 and 4

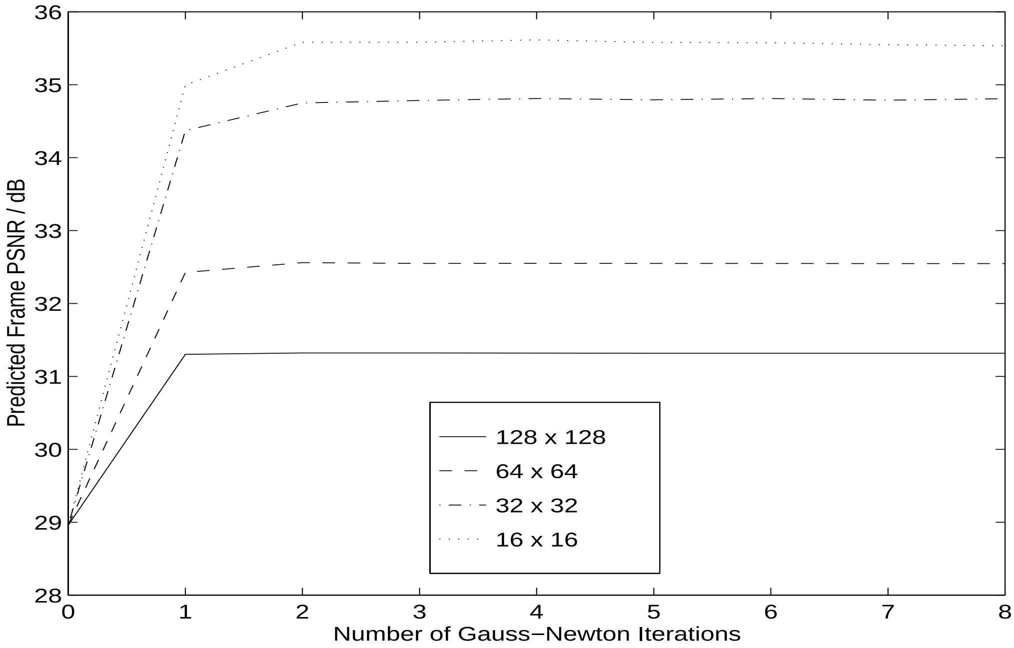


Figure 3: Motion Prediction PSNR for “Trevor”.

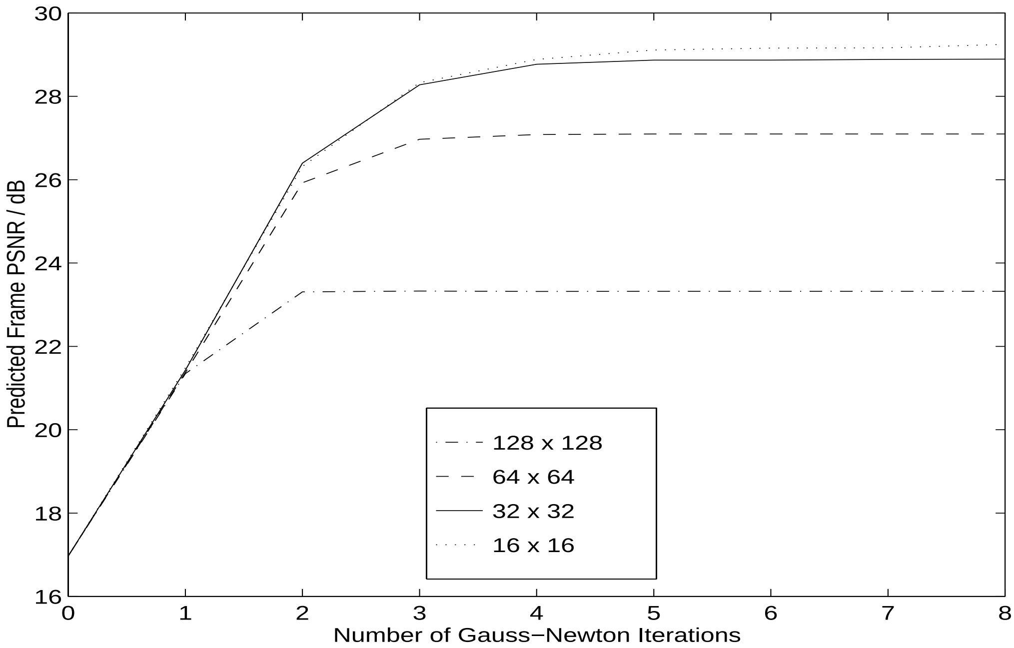
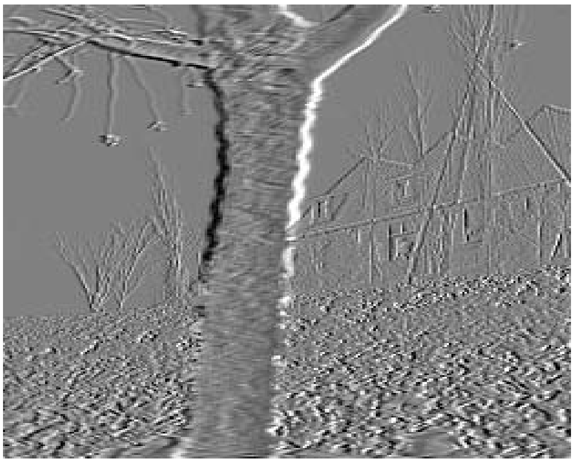


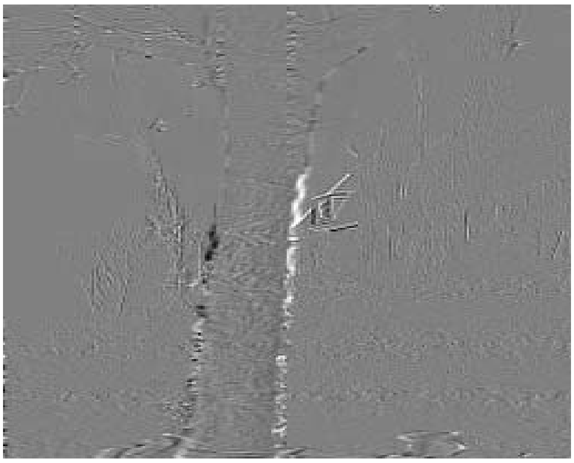
Figure 4: Motion Prediction PSNR for “Football”.

Frame 20 of *"Flower Garden"* sequence

Frame 21 of *"Flower Garden"* sequence



Frame difference *before* applying MEMC.



Frame difference *after* applying MEMC.

◇ ***Any new methods or recent breakthrough ??***

↪ Yes & No! We developed a new block-based method
↪ It's called the *Center-Biased Diamond Search* **(CBDS)**
↪ *Faster* and *better* than TSS, NTSS, 4SS, etc.
↪ *More robust* to different search ranges

♠ 3-D Wavelet Decomposition Framework

♢ *What do you mean by "decomposing" a video ??*

↪ Transform into another domain for easier analysis and more efficient coding
↪ Signal decorrelation and energy compaction
↪ Hierarchical 3-D subband structure
↪ Good time-frequency (time-scale) properties of wavelets

♢ *Why not use other types of transforms ??*

↪ Karhunen-Loève (K-L) transform – not very practical
↪ Discrete Cosine Transform (DCT) – "blocking" artifacts
↪ Others like DST, DFT, Hadamard, etc.

♢ *O.K.! How to decompose now ??*

1. Temporal Decomposition
↪ Partition video into disjointed *groups of frames* (GOFs)
↪ 1-D decomposition along temporal dimension
↪ Use Daubechies orthogonal 4-tap, Haar, etc.

2. Spatial Decomposition
↪ 2-D separable decomposition along horizontal and vertical dimensions
↪ Use biorthogonal splines, semiorthogonal splines, multiwavelets, etc.
↪ See Figs. 5 and 6

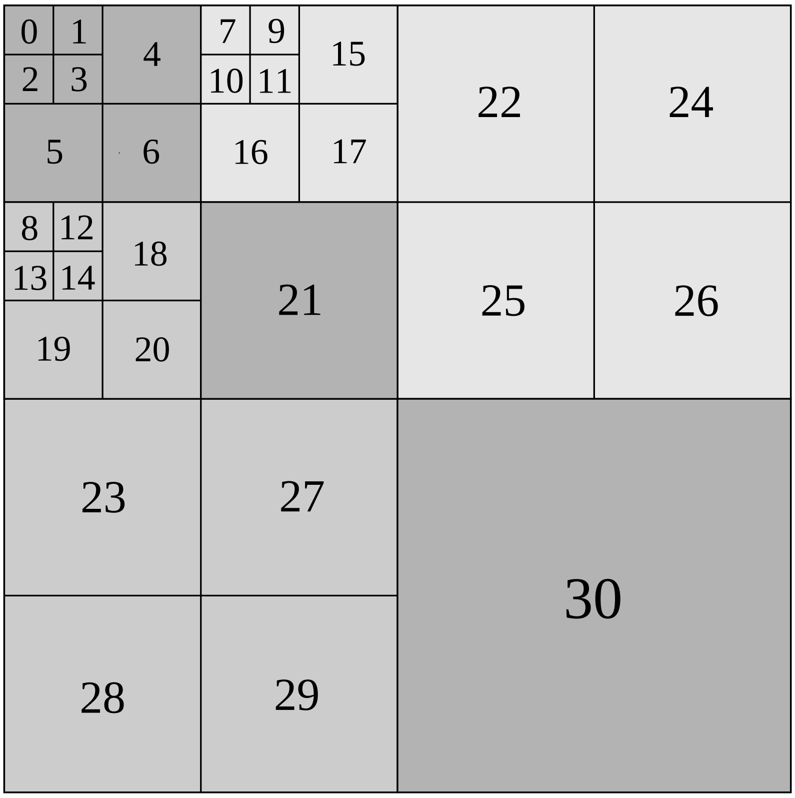


Figure 5: Spatial 2-D wavelet (packet) decomposition.

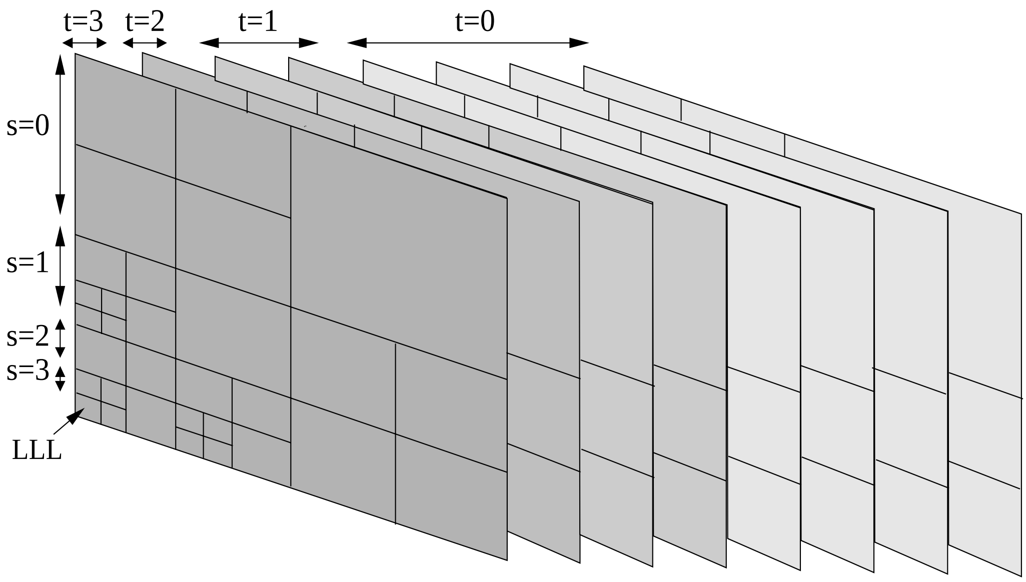


Figure 6: 3-D wavelet decomposition framework.

♣ Progressive Video Coding via TRI-ZTRs

♢ *What so special ??*

↪ Layered/progressive coding – *multirate* scalability
↪ Coding of resolution and frame blocks – *multiresolution*
↪ Good rate-distortion performance at very low bit rates

♢ *I heard something about Shapiro's EZW Coding ??*

↪ Good, it's a very elegant still image coding technique
↪ Bit rate scaling is possible

♢ *So what about TRI-ZTR coding ??*

↪ It stands for "Tri-Zerotrees"!
↪ A truly embedded video sequence coding technique
↪ Both multirate and multiresolution scalings are possible

♢ *Subband/parent-child relationships ??*

↪ Hierarchical tree relationship across scales (see Fig. 7)

$$\mathrm{CHILD}\{c(x_s, t_n; \mathrm{TREE})\} = \bigcup_{t \in [t_n, t_F)} c(x_{s-1}, t; \mathrm{TREE}), \tag{11}$$

where F is the size of GOF, and TREE $\in$ {DIAGONAL, VERTICAL, HORIZONTAL}.
↪ Inherent scanning sequence (coarse-to-fine)

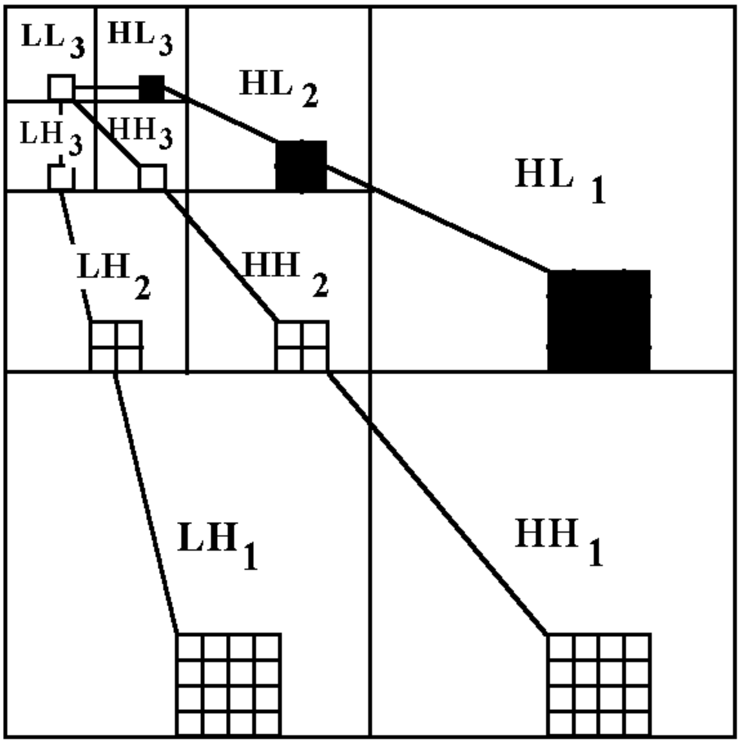


Figure 7: Octave subband (parent-child) relationships.

◇ *Can give an overview of TRI-ZTR Video Codec ??*

↪ No problem, please refer to Fig. 8

♣ 1. Primary Pass

◇ *What are the motivations ??*

↪ Coefficients with higher energy – more important
↪ Coded and transmitted earlier in the bit stream
↪ Attempt to have the "optimum" rate-distortion performance (*e.g.* for progressive transmission)

$\diamondsuit$ ***Sounds great! How to achieve these features ??***

$\hookrightarrow$ Compare the magnitude of each coefficient with a series of decreasing thresholds, $T_n, (n \in \mathcal{Z})$

$\hookrightarrow$ Code the significant ones ($\geq T_n$) first

$\hookrightarrow$ See Fig. 9

Step 1: Dominant Pass

$\hookrightarrow$ A discrimination process: (See Fig. 10)

$\hookrightarrow$ Dominant list! Significant list! Compressed bit stream!

Step 2: Insertion of Resolution Flags (RFG)

$\diamondsuit$ ***Why need RFG symbols ??***

$\hookrightarrow$ Critical for multiresolution scalability (choosing frame size and frame rate)

$\hookrightarrow$ Segment compressed bit stream into unique *resolution blocks* and *frame blocks*

$\diamondsuit$ ***How can this be done ??***

$\hookrightarrow$ Insert a RFG symbol at each required spatial and temporal scale – say, R_s of them (see Fig. 11)

$\hookrightarrow$ Resulting bit stream – see Fig. 12

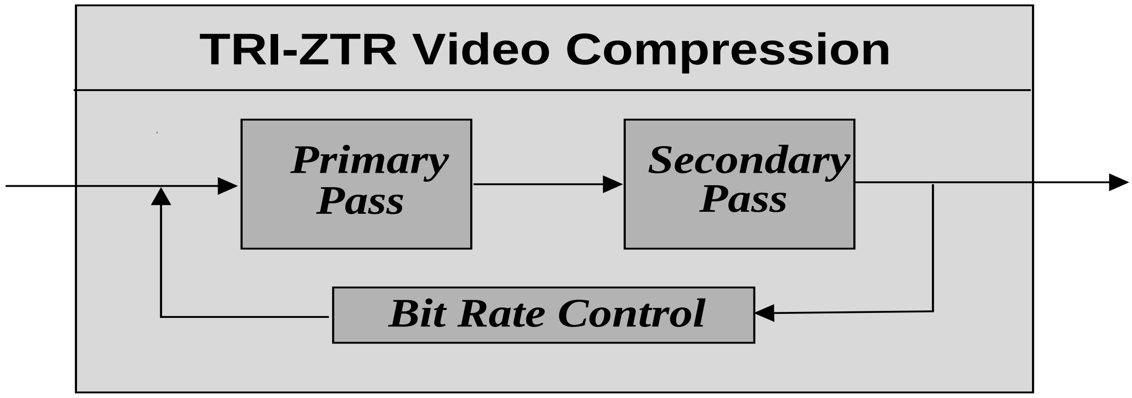


Figure 8: The main TRI-ZTR compression algorithm. It consists of many rounds for layered coding, and precise bit rate can be achieved.

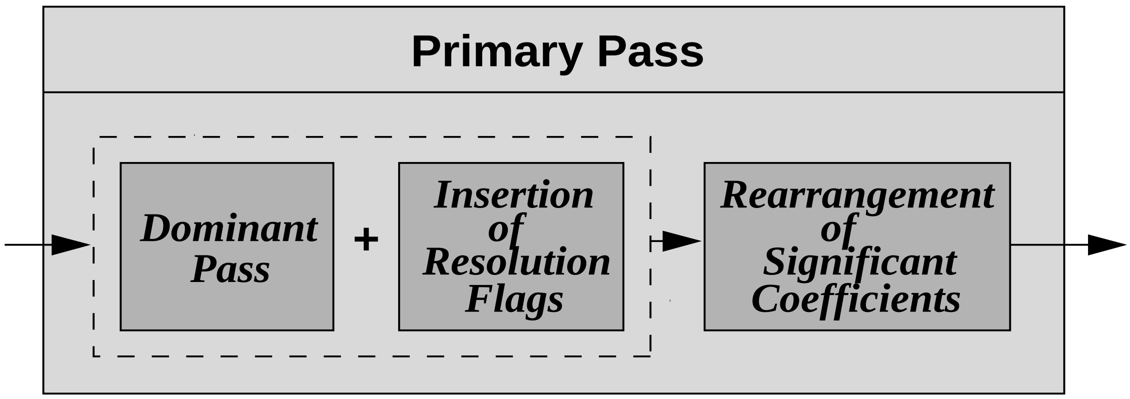


Figure 9: A primary pass is made up of three key steps.

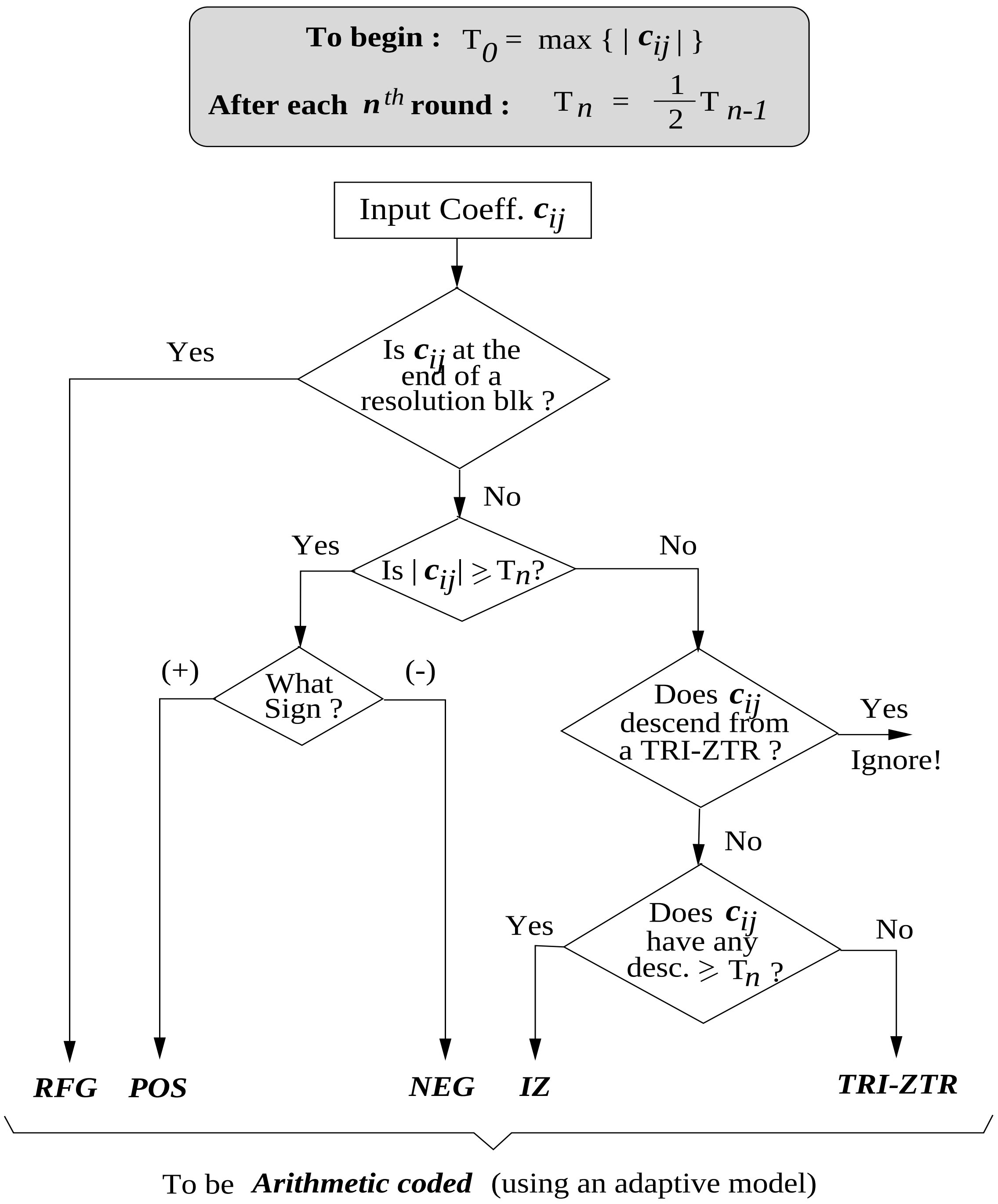


Figure 10: The dominant pass as a discrimination process.

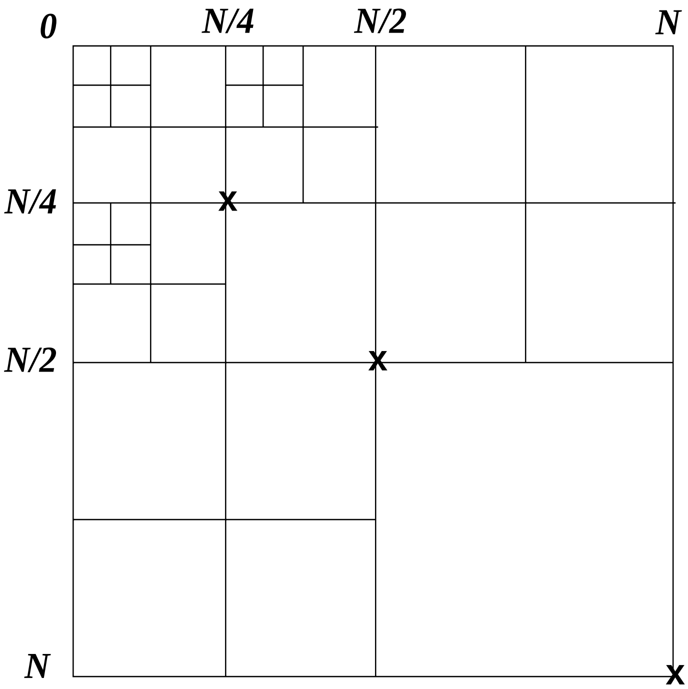


Figure 11: The positions (crosses) where $R_s = 3$ RFG are inserted.

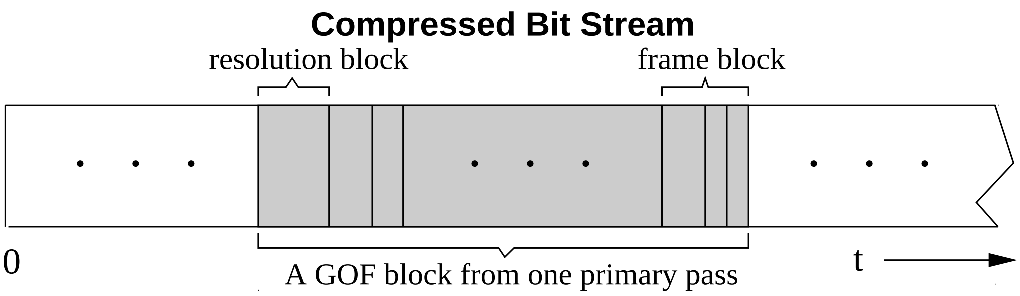


Figure 12: A portion of the bit stream showing unique resolution and frame blocks.

Step 3: Reordering of Significant Coefficients

♢ *Why? Won't this mess up everything ??*

↪ To *reduce* the cost of coding the RFG symbols
↪ *Without destroying* the integrity of each resolution block
↪ Decoder must replicate the same reordering process *without* any additional explicit overheads
↪ See Fig. 13 – "appending" at the end

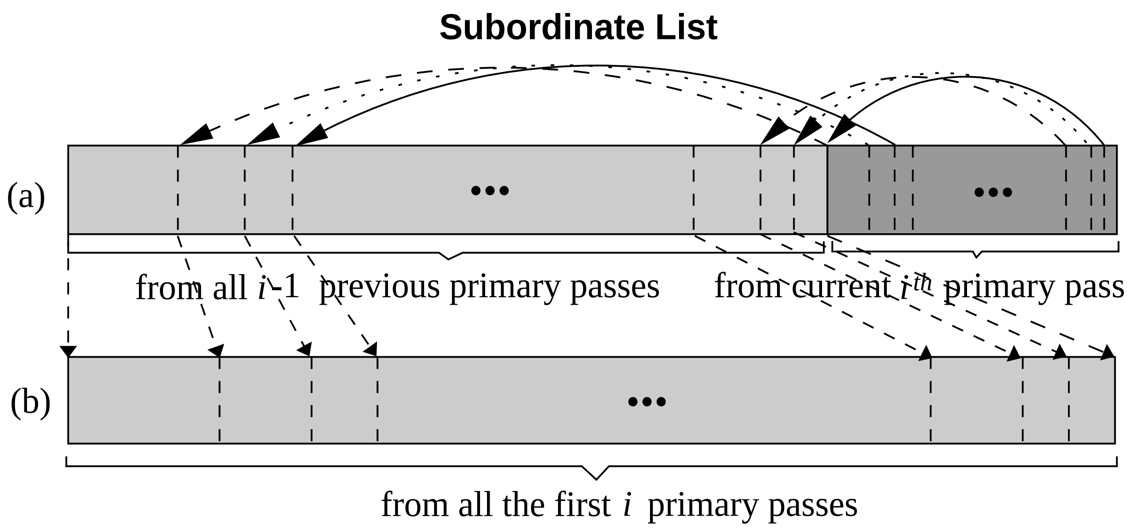


Figure 13: A snapshot (a) just before, and (b) just after, the rearrangement step.

♢ *But..But, I still can't see this ??*

↪ Well, this will become clearer when we perform the secondary pass later

♣ 2. Secondary Pass

↪ After each i^{th} primary pass
↪ See Fig. 14

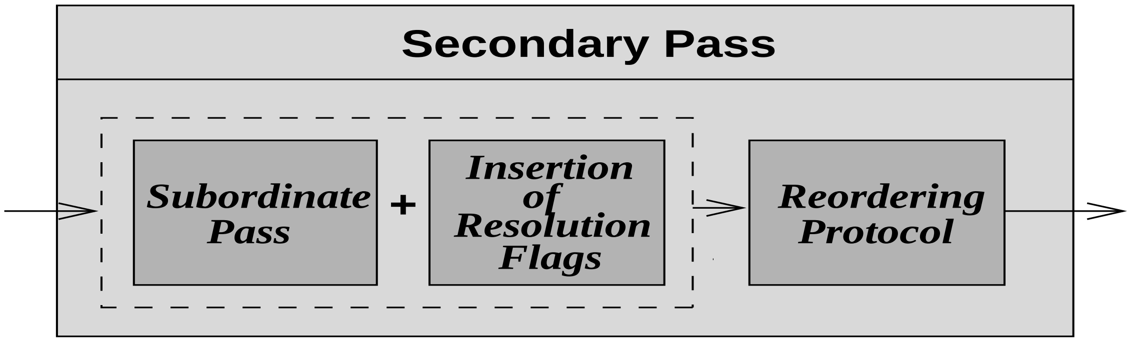


Figure 14: A secondary pass is also made up of three key steps.

Step 1: Subordinate Pass

◇ ***Why have "subordinate" some more ??***

↪ Because we are performing *layered coding*
– multiple rounds of primary and secondary passes!
↪ *Successive refinement* of quantization levels (bit plane!)
↪ Now we have both signs and positions information only
↪ What about their *magnitudes*?
↪ Answer:- See Fig. 15
↪ *Trade-off:* Direct quantization vs. finding of new smaller significant coefficients

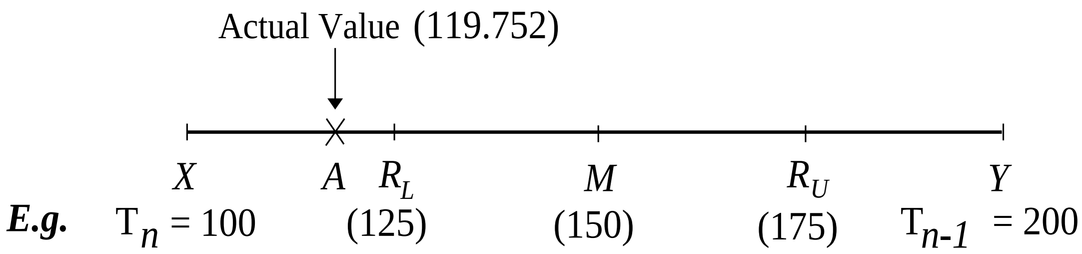


Figure 15: Uncertainty interval XY for a coefficient with $A = |c_{ij}|$ and $|XY| \equiv |T_n|$.

Step 2: Insertion of Resolution Flags (RFG)

$\hookrightarrow$ Same reason - to enable multiresolution scalability
$\hookrightarrow$ Only $(R_s$ x $F)$ RFG symbols, *not* $(n+1)(R_s$ x $F)$!!
$\hookrightarrow$ That's why the reordering just now is important

Step 3: Reordering Protocol

$\Diamond$ ***Is that something to do with "prioritization" ??***

$\hookrightarrow$ Precisely, we attempt to place a more important piece of information *earlier* in the compressed bit stream
$\hookrightarrow$ *E.g.* receiving $|c_{ij}| \equiv$ MSE decreasing by $\frac{|c_{ij}|^2}{M}$
$\hookrightarrow$ Hence, choose those larger $|c_{ij}|$ to be quantized and refined earlier than the smaller ones

♢ ***O.K., but how to sort them properly ??***

↪ Based on a pre-determined *prioritization protocol* below:

1. **Temporal Scale** - assumed good MEMC, temporal energy compaction
2. **Spatial Scale** - coarse-to-fine manner, raster-scanned
3. **Reconstruction Magnitude** - decreasing order of magnitude within the same resolution block
4. **Spatial Position** - according to original scanning sequence, to syncronize with the decoder

↪ *Gist:* We reorder the significant coefficients by their reconstruction magnitudes (as can be seen by the decoder) according to the inherent scanning sequence *without* any explicit overhead

↪ We also must ensure that the integrity of each resolution and frame block to be properly preserved even *after* the reordering process in order to support multiresolution scalability

♠ Video Scalability & Re-scalability

$\boxed{\diamondsuit\ \textit{So, what video scalabilities are possible ??}}$

↪ Recall the followings:

- **Bit-rate (Bandwidth)**
- **Distortion (Quality)**
- **Spatial resolution (Frame size)**
- **Temporal resolution (Frame rate)**
- **Hardware complexity (CPU & Memory)**
- **Interactivity (Coding delays)**

$\boxed{\diamondsuit\ \textit{Yeah I know, but how to know what is what ??}}$

↪ Well, this question has been answered!!
↪ We already have a one-to-one correspondence for each resolution and frame block
↪ See Fig. 16
↪ Direct bit stream extraction is now possible

$\boxed{\diamondsuit\ \textit{Still not very clear leh! Any examples ??}}$

↪ Fine, see Table 1 below:

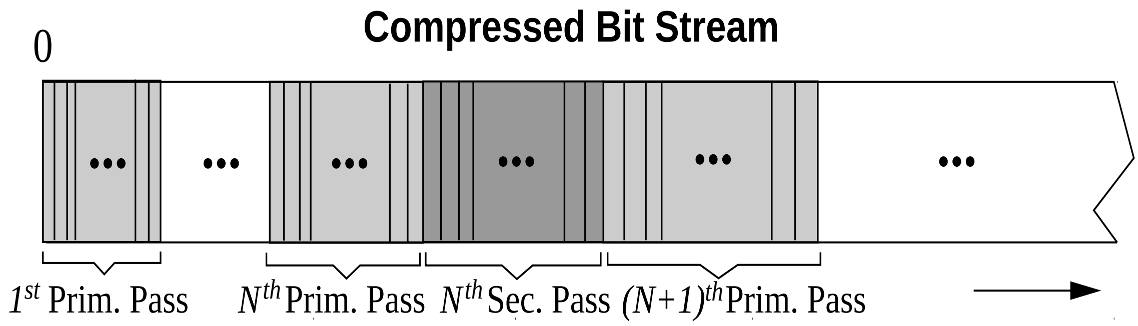


Figure 16: Compressed bit stream consisting of unique resolution blocks.

Resolution Scaling		*Bit Rate Scaling*
Spatial Resolution (pixels x lines)	Frame Rate (frames/s)	Bit Rate (kbits/s)
352 x 288	30	100
352 x 288	15	64
176 x 144	7.5	35
176 x 144	15	30
88 x 72	30	20
88 x 72	15	20
88 x 72	3.75	12

Table 1: Examples of combinations of display parameters.

$\diamondsuit$ ***You mean only these are possible ??***

$\hookrightarrow$ Not at all, we can any combinations of the followings:

- *Spatial resolution* = {352 x 288, 176 x 144, 88 x 72}
- *Frame rate* = {30, 15, 7.5, 3.75}
- *Bit rate* = {Any *precise* bit rate subject to the maximum available in the original bit stream}

$\hookrightarrow$ More could also be obtained – choose larger R_s and F!

$\diamondsuit$ ***What about "re-scalability" ??***

$\hookrightarrow$ It is definitely possible - for example?

$\diamondsuit$ ***Well, talk so much! Any good results to show ??***

♠ Simulation Results & Comparisons

$\hookrightarrow$ Take a good look at these images, friends!
$\hookrightarrow$ Only TRI-ZTR video codec is fully multirate and multiresolution scalable
$\hookrightarrow$ Almost free from the annoying "blocking" aftifacts
$\hookrightarrow$ However, blurring and ringing do occur at such very low bit rates
$\hookrightarrow$ Subjectively, the videos are very acceptable for bit rates of 15 kpbs, or lower (suitable for video phones)

Original frame 126 of "Miss America"

TRI-ZTR Codec: CR = 250:1 - *without* wavelet-packet decomposition

TRI-ZTR codec: CR = 250:1 - *with* wavelet-packet decomposition.

TRI-ZTR codec: CR = 500:1

MPEG-2 codec: CR = 80:1

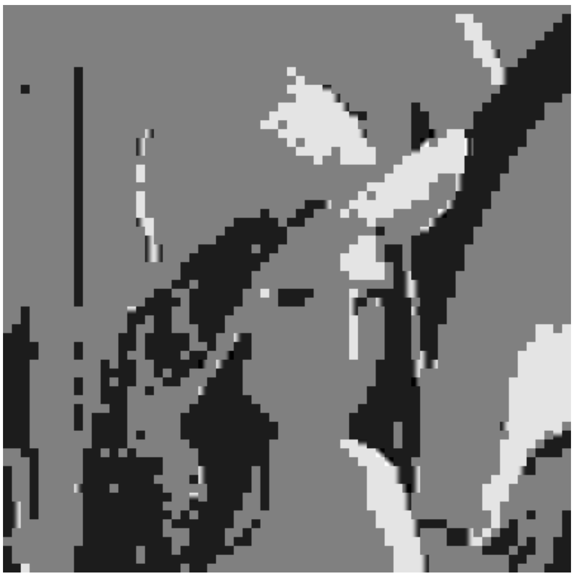
H.263 codec: CR = 250:1

H.263 codec: CR = 500:1

TRI-ZTR codec: CR = 250:1 - temporally scaled by half

TRI-ZTR codec: CR = 300:1 - spatially scaled by half

TRI-ZTR codec: CR = 500:1 - spatially scaled by half



JPEG (xview 3.10a): Still image (CR = 70:1)

TRI-ZTR codec: Still image (CR = 100:1) - GOF = 1