Progressive Image Compression Using Binary Trees

Computer Graphics Group,
Mathematics Dept.
Moscow State University

Authors: *Denis V. Ivanov*
Eugene P. Kuzmin
Sergey V. Burtsev

Compression Types

- Lossless (PCX,GIF,PING)
  - » No lose of any data
  - » Relatively low compression ratios
- Lossy (JPEG,Wavelets)
  - » May lose significant data
  - » High compression ratios
- Progressive = Lossy + Lossless
  - » Construct image sketches from incomplete data
  - » Enhance quality while receiving next portions of data
  - » Reconstruct the original having received the whole data

Progressive Transmission

- GIF, PING
  - » Image reconstruction on uniform grid basis
  - » Each cell is represented by top-left pixel (not average)
- Non-linear transforms
  - – S+P – transforms (similar to Haar wavelets)
    - » Reconstruction on block basis
    - » Each cell is represented by average value
  - – S+P + set partitioning
    - » Reconstruction ordering judging from MSE
    - » Computationally complex

Advantages of Our Technique

- Lossless compression ratios are among the best
  - » Better then PCX, GIF, PING, …
- Progressive decompression
  - » Decompression may be implemented in parallel to data extraction
- Fast and simple compression/decompression
  - » Parallel operations, MMX
- MSE estimation
  - » MSE of intermediate sketches may be obtained without any additional computations

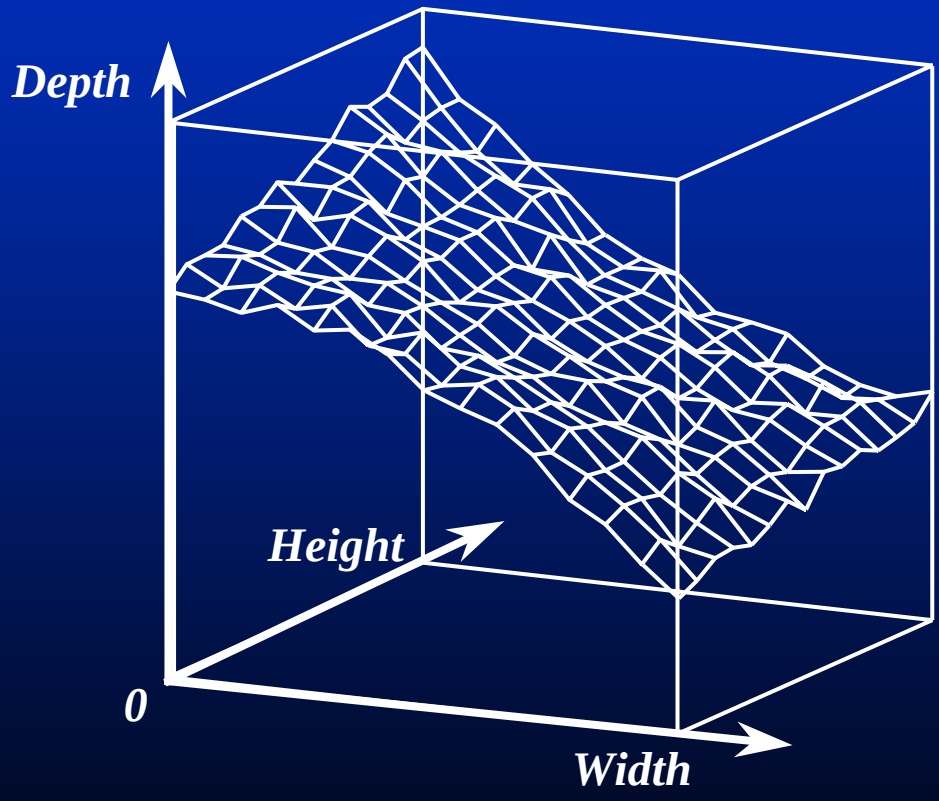
Compression Stages

- Binary tree representation
  - » Bitmap is represented by special binary tree structure
- Nodes sorting
  - » Nodes are sorted in the order of their appearance in the compressed data stream
- Entropy coding
  - » Nodes are encoded judging from their frequencies, which are generally the same for all images

These 3 stages are executed simultaneously

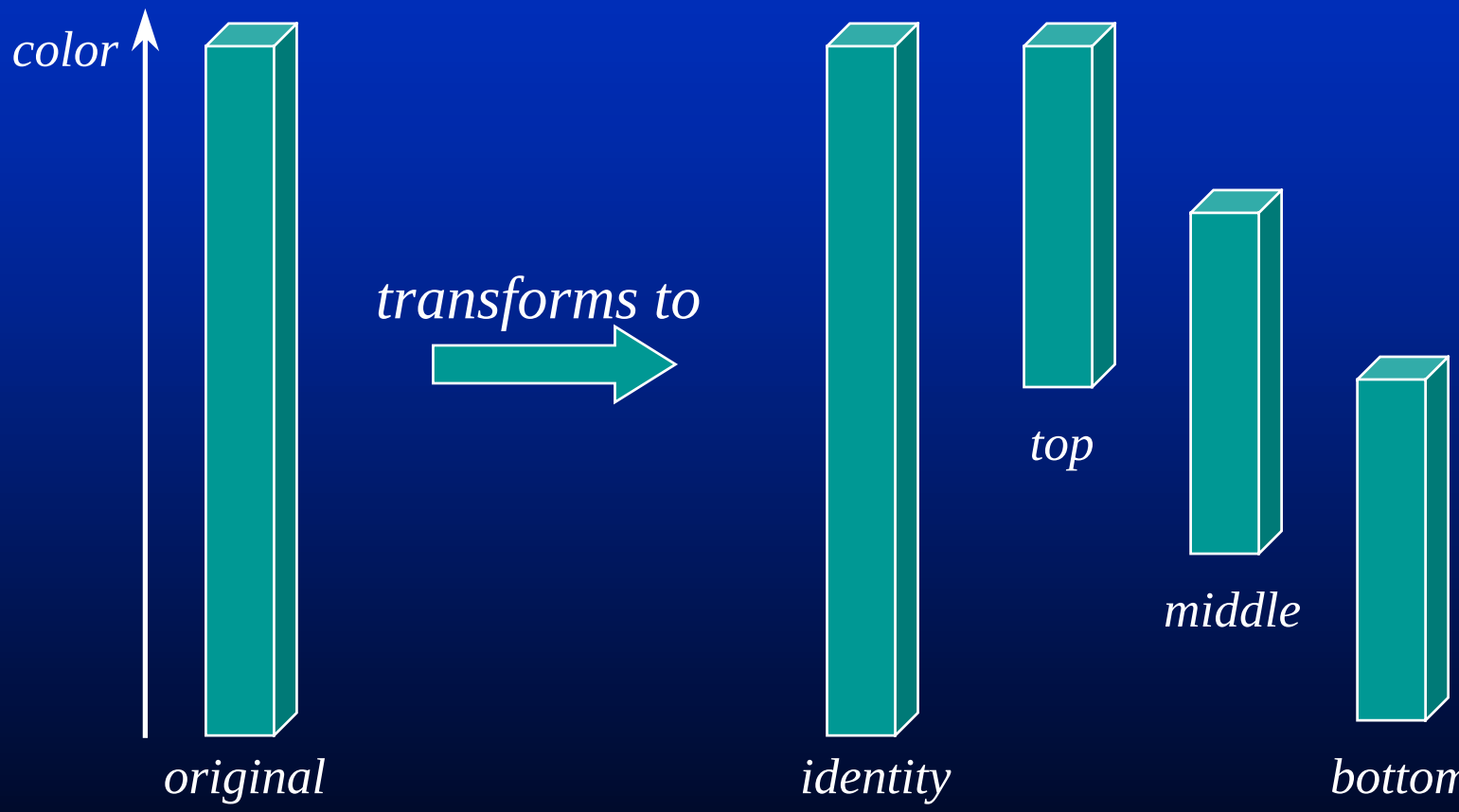
Binary Tree Representation (1)

- The root corresponds to 3D cube holding the whole image surface



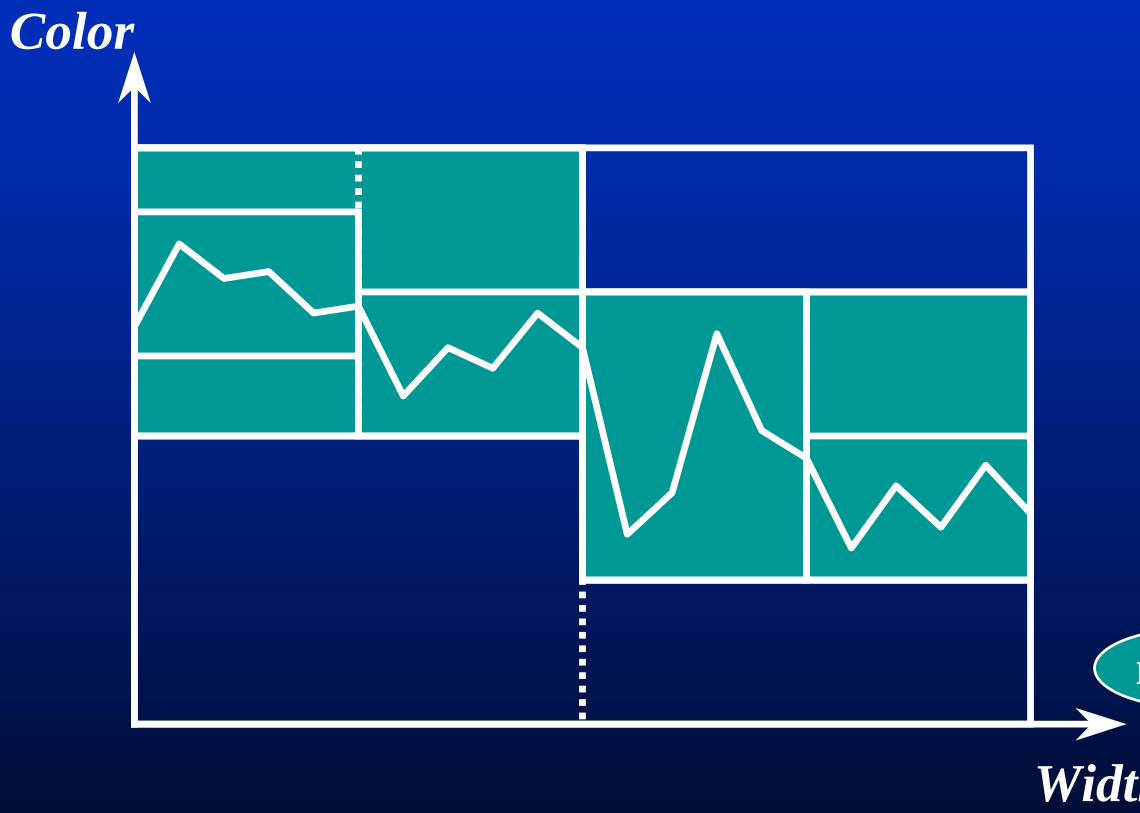
Binary Tree Representation (2)

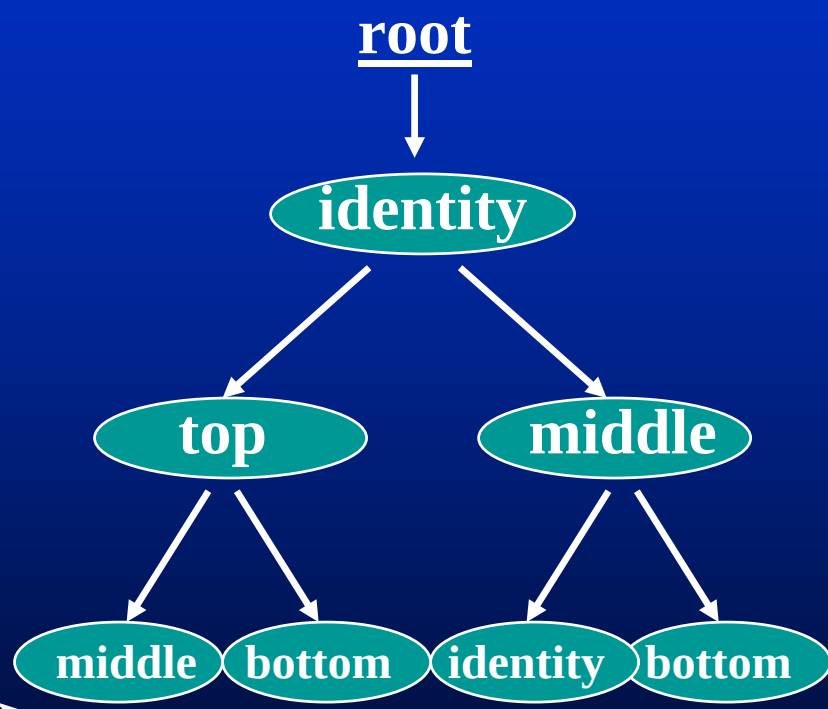
- Each tree node represents one of the contractions in color dimension



Binary Tree Representation (3)

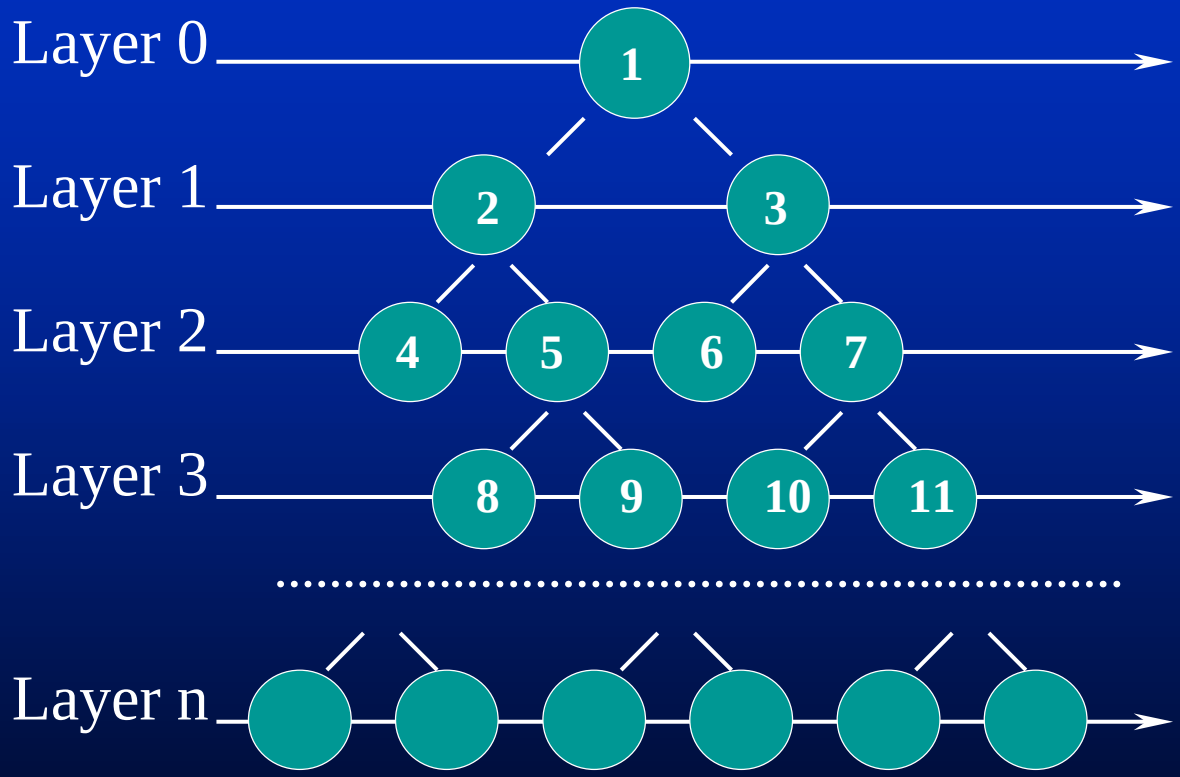
- The tree building is implemented as follows





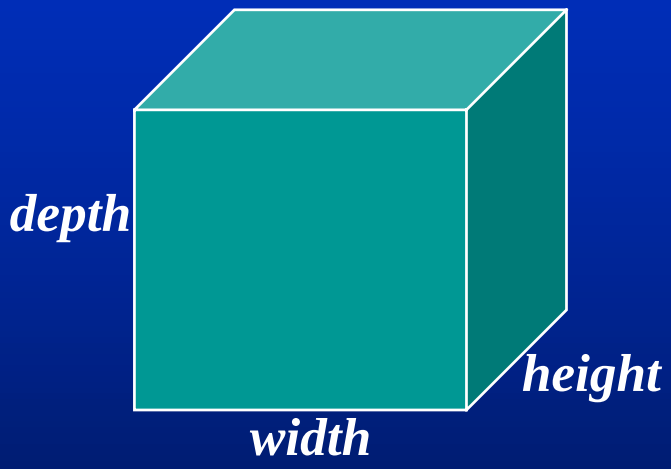
Nodes Sorting (1)

- Layer-by-Layer



Nodes Sorting (2)

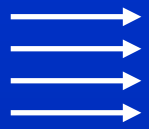
- Volume based



- ✓ Select the node with the largest volume
- ✓ Generate contraction and bisect it in X or Y direction

Volume = Width * Height * Depth

Nodes Sorting (3)



Layer-by-Layer

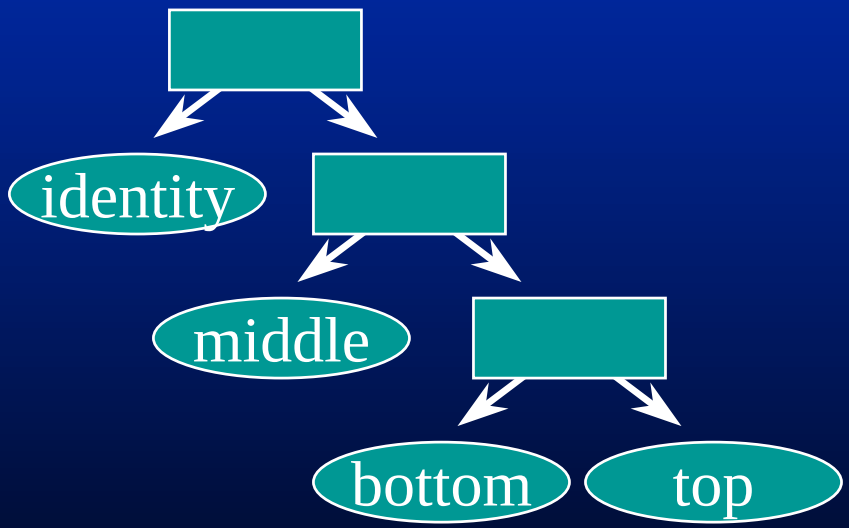
- *Construction*
  - Nodes ordering is simple and natural
- *Reconstruction*
  - MSE is not taken into consideration
  - Reconstruction is made on the uniform grid basis (like in S+P)

Volume based

- *Construction*
  - Nodes ordering procedure is a little bit more complex
- *Reconstruction*
  - MSE is considered locally to perform reconstruction
  - Areas with higher variance are updated at first order

Entropy coding

- Tree nodes (contraction operators) are coded judging from their frequencies
- Frequency histograms are much alike for all image types



Huffman tree for node values

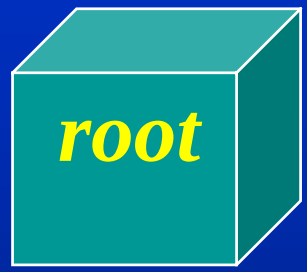
Contraction operator	Code
identity	0
middle	10
bottom	110
top	111

Image Reconstruction

Input

Bit stream: 1010010010110101O…

Decoder



= Width x Height x Depth

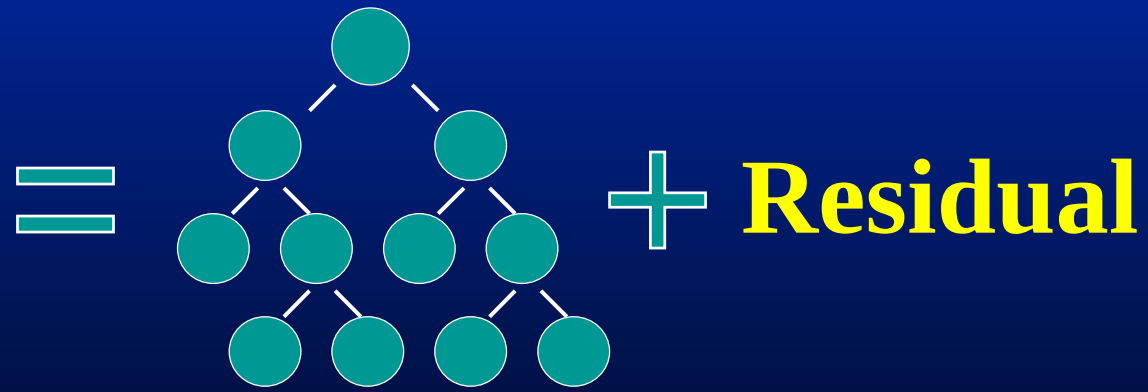
1. Get next code (Huffman)
2. Select the corresponding node
3. Apply the corresponding contraction in color dimension
4. Form children = Bisect parallelepiped in X or Y dimension
5. Goto 1

Tree Truncation (1)

- Our research showed that it is better to construct the tree up to the certain level, such as

Layer = 7 or Volume = 32 or Area = 4

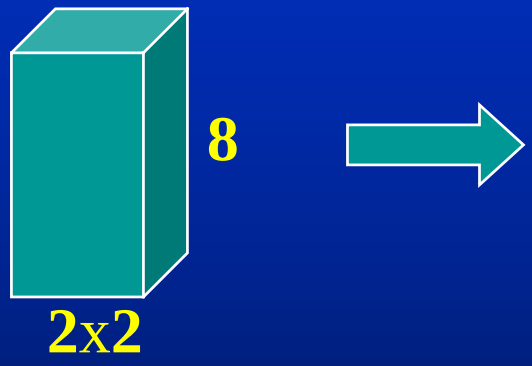
Original image



Tree Truncation (2)

- Residual storage

Volume (*Node*) = 32



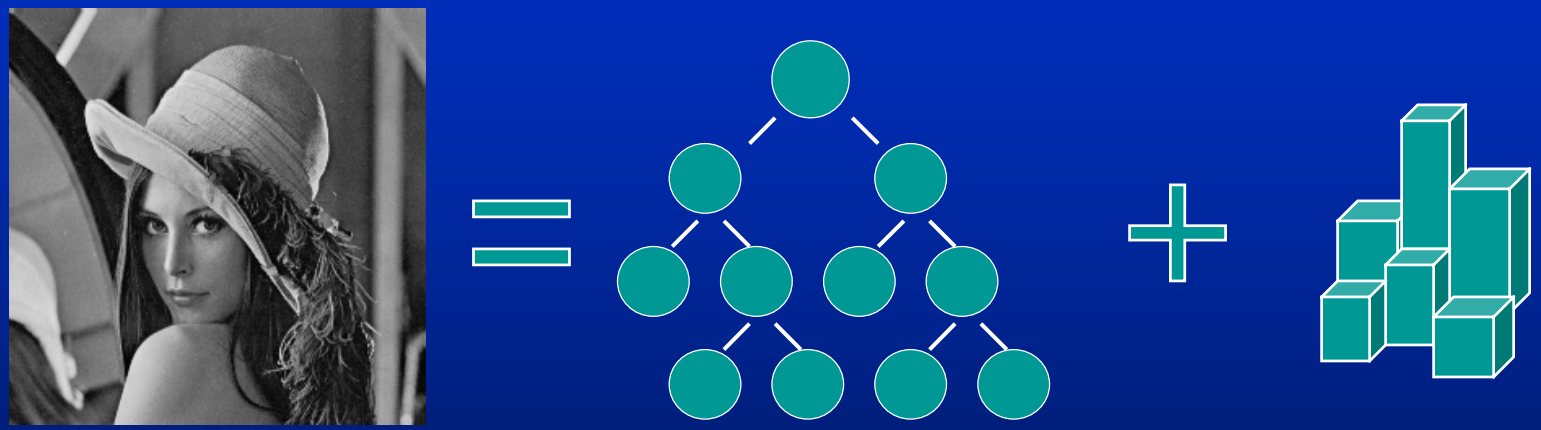
4 *pixels* * **3** *bits/pixel* = **12** *bits*

- We state: *For some tree limit (depending on image) the residual can not be efficiently coded with entropy coding methods.*

Residual = Incompressible Noise

Practical Results (1)

- Compression of *Lena* image



Original image
65536 (100%)

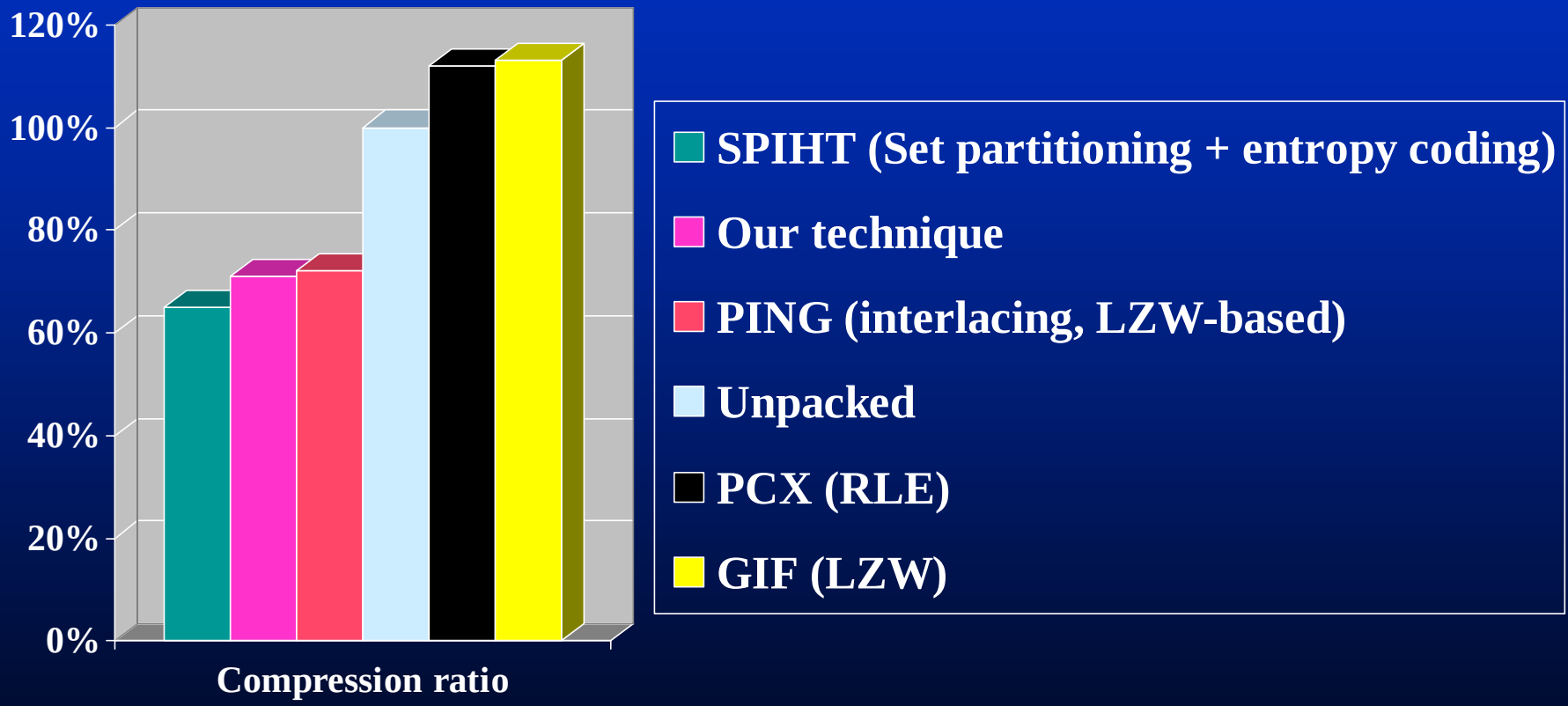
depth = 16 or *area* = 4
7332 (11%)

Residual
39515 (71%)

Total: 46847 (71%)

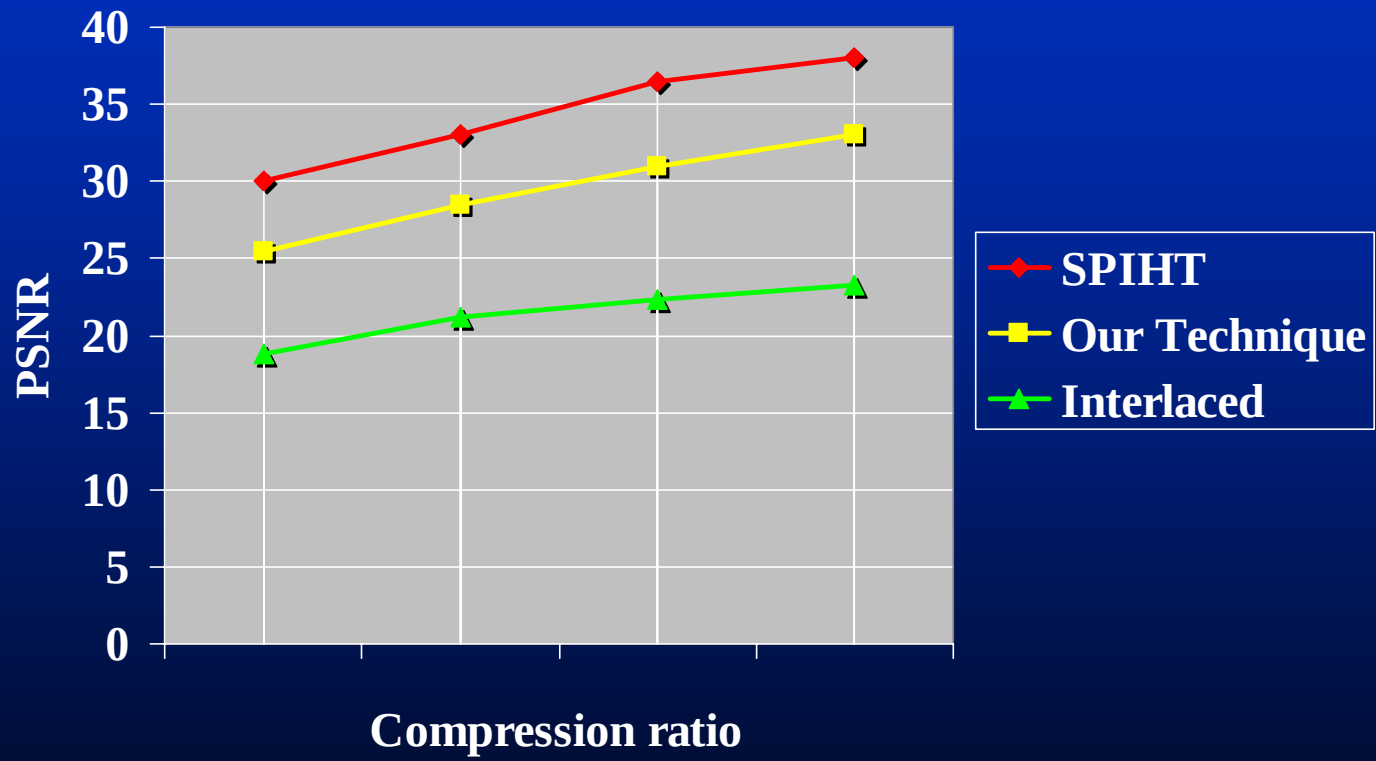
Practical Results (2)

- Compression ratio (*Lena* image)



Practical Results (3)

- Visual quality (*Lena* image)



Practical Results (4)

Lena (*256* x *256* x *256*) - scaled

8 bpp (100%)

Range of Use

- Efficient lossless compression of important data
  - » High compression ratios
- Fast inspection of large image sets via network
  - » Only a short messages should be extracted to reconstruct image sketches
- Hardware implementation
  - » Very simple algorithm
  - » Parallelism

Acknowledgements

Presenting technique has been developed within the research project which is held by Math. Dept of Moscow State University under the agreement with Intel Technologies, Inc.

Intel Technologies, Inc.

Moscow State University
Mathematics Dept.