15-826: Multimedia Databases and Data Mining

Lecture #24: Compression - JPEG, MPEG, fractal

C. Faloutsos

Must-read Material

• JPEG: Gregory K. Wallace, The JPEG Still Picture Compression Standard, CACM, 34, 4, April 1991, pp. 31-44

Outline

Goal: ‘Find similar / interesting things’
• Intro to DB
• Indexing - similarity search
• Data Mining
Indexing - Detailed outline

- primary key indexing
- ..
- multimedia
- Digital Signal Processing (DSP) tools
  - Image + video compression
    - JPEG
    - MPEG
    - Fractal compression

Motivation

Q: Why study (image/video) compression?

A1: feature extraction, for multimedia data mining
A2: (lossy) compression = data mining!
JPEG - specs

- (Wallace, CACM April ’91)
- Goal: universal method, to compress
  - losslessly / lossily
  - grayscale / color (= multi-channel)
- What would you suggest?

JPEG - grayscale - outline

- step 1) 8x8 blocks (why?)
- step 2) (Fast) DCT (why DCT?)
- step 3) Quantize (fewer bits, lower accuracy)
- step 4) encoding
  - DC: delta from neighbors
  - AC: in a zig-zag fashion, + Huffman encoding

Result: 0.75-1.5 bits per pixel (8:1 compression) - sufficient quality for most apps

JPEG - grayscale - lossless

- Predictive coding:
  \[
  \begin{array}{ccc}
  C & B \\
  A & X \\
  \end{array}
  \]
  \[X = f(A, B, C)\]
  eg. \[X = (A+B)/2\], or?

- Then, encode prediction errors

Result: typically, 2:1 compression
JPEG - color/multi-channel

- apps?
- image components = color bands = spectral bands = channels
- components are interleaved (why?)

- to pipeline decompression with display

8x8 ‘red’ block  8x8 ‘green’ block  8x8 ‘blue’ block

- tricky issues, if the sampling rates differ
- Also, hierarchical mode of operation: pyramidal structure
  - sub-sample by 2
  - interpolate
  - compress the diff. from the predictions
JPEG - conclusions

• grayscale, lossy: 8x8 blocks; DCT; quantization and encoding
• grayscale, lossless: predictions
• color (lossy/lossless): interleave bands

Indexing - Detailed outline

• primary key indexing
• ..
• multimedia
• Digital Signal Processing (DSP) tools
• Image + video compression
  – JPEG
  – MPEG
  – Fractal compression

MPEG

• (LeGall, CACM April ’91)
• Video: many, still images
• Q: why not JPEG on each of them?
MPEG

• (LeGall, CACM April ‘91)
• Video: many, still images
• Q: why not JPEG on each of them?
• A: too similar - we can do better! (~3-fold)

MPEG - specs

• ??

MPEG - specs

• acceptable quality
• asymmetric/symmetric apps (#compressions vs #decompressions)
• Random access (FF, reverse)
• audio + visual sync
• error tolerance
• variable delay / quality
• editability
MPEG - approach

- main idea: balance between inter-frame compression and random access
- thus: compress *some* frames with JPEG (I-frames)
  - rest: prediction from motion, and interpolation
  - P-frames (predicted pictures, from I- or P-frames)
  - B-frames (interpolated pictures - never used as reference)

MPEG - approach

- useful concept: ‘motion field’

MPEG - conclusions

- with the I-frames, we have a balance between
  - compression and
  - random access
Indexing - Detailed outline

- primary key indexing
- multimedia
- Digital Signal Processing (DSP) tools
- Image + video compression
  - JPEG
  - MPEG
  - Fractal compression

Fractal compression

- ‘Iterated Function systems’ (IFS)
  - (Barnsley and Sloane, BYTE Jan. 88)
  - Idea: real objects may be self-similar, eg., fern leaf

Fractal compression

- simpler example: Sierpinski triangle.
  - has details at every scale -> DFT/DCT: not good
  - but is easy to describe (in English)
- There should be a way to compress it very well!
- Q: How??
Fractal compression

• simpler example: Sierpinski triangle.
  – has details at every scale -> DFT/DCT: not good
  – but is easy to describe (in English)
• There should be a way to compress it very well!
• Q: How??
• A: several, affine transformations
• Q: how many coeff. we need for a (2-d) affine transformation?

Fractal compression

• A: 6 (4 for the rotation/scaling matrix, 2 for the translation)
• (x,y) -> w((x,y)) = (x’, y’)
  – x’ = a x + b y + e
  – y’ = c x + d y + f
• for the Sierpinski triangle: 3 such transformations - which ones?

Fractal compression

• A:

<table>
<thead>
<tr>
<th>A</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>w1</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
</tr>
<tr>
<td>w2</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>w3</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>1/3</td>
</tr>
</tbody>
</table>
Fractal compression

• The above transformations 'describe' the Sierpinski triangle - is it the only one?
• i.e., how to de-compress?

A: YES!!
• i.e., how to de-compress?
• A1: Iterated functions (expensive)
• A2: Randomized (surprisingly, it works!)

Fractal compression

• Sierpinski triangle: is the ONLY fixed point of the above 3 transformations:

w1  w2  w3
Fractal compression

- We'll get the Sierpinski triangle, NO MATTER what image we start from! (as long as it has at least one black pixel!)
- thus, (one, slow) decompression algorithm:
  - start from a random image
  - apply the given transformations
  - union them and
  - repeat recursively
- drawback?

Fractal compression

- A: Exponential explosion: with 3 transformations, we need $3^k$ sub-images, after $k$ steps
- Q: what to do?

Fractal compression

- A: PROBABILISTIC algorithm:
  - pick a random point $(x_0, y_0)$
  - choose one of the 3 transformations with prob. $p_1/p_2/p_3$
  - generate point $(x_1, y_1)$
  - repeat
  - [ignore the first 30-50 points - why?]
- Q: why on earth does this work?
- A: the point $(x_n, y_n)$ gets closer and closer to Sierpinski points $(n=1, 2, ...)$, ie:
Fractal compression

... points outside the Sierpinski triangle have no chance of attracting our ‘random’ point \((x_n, y_n)\)

Q: how to compress a real (b/w) image?
A: ‘Collage’ theorem (informally: find portions of the image that are miniature versions, and that cover it completely)

Drills:

Drill#1: compress the unit square - which transformations?

Fractal compression

Drill#1: compress the unit square - which transformations?

Fractal compression

Drill#1: compress the unit square - which transformations?
Fractal compression

Drill#2: compress the diagonal line:

Fractal compression

Drill#3: compress the ‘Koch snowflake’:

Fractal compression

Drill#3: compress the ‘Koch snowflake’: (we can rotate, too!)
Fractal compression

Drill#4: compress the fern leaf:

Fractal compression

Drill#4: compress the fern leaf: (rotation + diff. p_i)

PS: actually, we need one more transf., for the stem

Fractal compression

- How to find self-similar pieces automatically?
- A: [Peitgen+]: eg., quad-tree-like decomposition
Fractal compression

- Observations
  - may be lossy (although we can store deltas)
  - can be used for color images, too
  - can ‘focus’ or ‘enlarge’ a given region, without JPEG’s ‘blockiness’

Conclusions

- JPEG: DCT for images
- MPEG: I-frames; interpolation, for video
- IFS: surprising compression method

Resources/References

- IFS code: www.cs.cmu.edu/~christos/SRC/ifs.tar
- Gregory K. Wallace, The JPEG Still Picture Compression Standard, CACM, 34, 4, April 1991, pp. 31-44
References

• D. Le Gall, MPEG: a Video Compression Standard for Multimedia Applications CACM, 34, 4, April 1991, pp. 46-58
• Heinz-Otto Peitgen, Hartmut Juergens, Dietmar Saupe: Chaos and Fractals: New Frontiers of Science, Springer-Verlag, 1992