Rate-distortion optimized tree structured compression algorithms for piecewise smooth images

1. Introduction
   Long-standing tradition of tree-structured approaches in
   - signal processing and communications: binary trees
   - image processing and computer vision: quadtrees
   - video and medical imaging: octrees

   **Reason:**
   - computational complexity
   - tractability
   - design
   - successive refinement (or divide and conquer)

   **Some examples**
   - split & merge image compression [Leonardi et al]
   - binary space partitioning trees [Radha et al]
   - Non-linear tilings [Cohen, Mattei]: adaptive segmentation
   - NP-completeness of optimal tiling

   **Note:** no bases or frames here!

2. Non-Separable Constructions Based on Quadtrees
   Going to two dimensions requires non-separable bases

   Objects in two dimensions we are interested in
   - textures: $D(R) = C_0 \cdot 2^{-2R}$ per pixel
   - smooth surfaces: $D(R) = C_1 \cdot 2^{-2R}$ per object!
Compression of non-separable objects

Objects we know how to compress....

Approximation

- Wavelets: $E_M \sim M^{-1}$
- Ridgelets: $E_M \sim 2^{-M}$

Rate/distortion

- Oracle: $D(R) = C \cdot 2^{-2R}$
- Wavelets: poor
- Ridgelets: suboptimal
- Adaptive schemes: close to oracle
- Fixed basis: under investigation

Basis element

Motivation: Natural images exhibit structure

Natural images represent a special class of 2-D functions.

- Dominant image structures:
  1. Smooth regions: Surface (2-D) regularity.
  2. Smooth edge contours: Geometrical (1-D) regularity (Perceptual).
- Image processing algorithms require efficient modeling/exploitation of both type of regularity. In particular, compression and denoising.

Idea

- Tree and quadtree algorithms popular, many pruning algorithms
- Optimality proofs for wedgelets [Donoho:99]

Here: new pruning and joining algorithm

Binary Tree Algorithms for 1-D piecewise polynomials

Prune Binary Tree (Parent Children Pruning) [~ Wavelet packets]

**Step 1:** Initialization:
(a) Segment the signal using a binary tree.
(b) Approximate each node by its best polynomial
(c) Generate R-D curve for each node of the tree.

**Step 2:** Prune the tree to minimize the Lagrangian cost $L(\lambda) = D + \lambda R$.

**Step 3:** Search $\lambda^*$ for a desired bit budget $R^*$.

Prune Binary Tree
(Parent Children Pruning)

For piecewise polynomials: $D(R) \sim \sqrt{R} \cdot 2^{-c_1 \cdot \sqrt{R}}$
**Neighbor Joint Coding Strategy**

Encode jointly if: \( L_1(\lambda) \leq L_1(\lambda) + L_2(\lambda) \), where \( L(\lambda) = D + \lambda R \).

**Binary Tree Segmentations (Recap)**

(a) Full tree

(b) Dyadic tree

(c) Prune-join tree

Results: Rate-distortion optimal for piecewise polynomials that is, like an oracle method (up to constants) but at much lower computational complexity.

**Extension to 2-D: Quadtree Algorithms**

Algorithms are similar to the binary tree schemes in 1-D.

**Prune Quadtree Algorithm:**

- Segment the image into dyadic squares.
- Code each block as an edge-tile with a linear discontinuity.
- Prune the tree to minimize the Lagrangian cost: \( L(\lambda) = D + \lambda R \).

Prune-Join Quadtree Algorithm with Joint Coding

- Find the pruned tree using the Prune Quadtree Algorithm.
- Code neighbor segments with "similar" parameters jointly.
**Polygonal Image Model**

- **(a) Prune tree**
  - \( N_j \sim J \)
  - \( D(R) \sim \sqrt[4]{R} \cdot 2^{-c_1 \cdot \sqrt{R}} \)

- **(b) Prune-join tree**
  - \( N_j \sim J^2 \)
  - \( D(R) \sim 2^{-c_2 \cdot R} \)

Prune-join quadtree => Object based paradigm

**The prune-join quadtree algorithm on an image**

- polynomial fit to surface and to boundary on a quadtree
- rate-distortion optimal tree pruning and joining

**Upperbounds on R-D Behaviors**

**A. Boundary is piecewise polynomial:**
- **Oracle:**
  - \( D(R) \sim 2^{-c \cdot R} \)
- The prune quadtree algorithm (independent coding):
  - \( D(R) \sim \sqrt[4]{R} \cdot 2^{-c_1 \cdot \sqrt{R}} \)
- The prune-join quadtree algorithm with joint coding:
  - \( D(R) \sim 2^{-c_2 \cdot R} \)

**B. Boundary is piecewise smooth:**
- The prune quadtree algorithm (code blocks independently) achieves the oracle performance (up to log factor):
  - \( D(R) \sim \left( \frac{\log R}{R} \right)^p \)

**C. Computational complexity of quadtree algorithms:**
- \( O(N^2 \cdot \log N) \)

**Simulation Results: Piecewise Polynomial Images**

- **(a) Prune tree**
- **(b) Prune-join tree**
### Simulation Results: Piecewise Smooth Images

(a) Prune Tree                  (b) Prune-join Tree           (c) JPEG2000
Rate=0.03 bpp Rate=0.02 bpp   Rate=0.065 bpp   PSNR=44.43 dB PSNR=44.24 dB           PSNR=43.81 dB

### Cameraman Image at 0.15 bpp

(a) Prune-join tree           (b) JPEG2000   (c) Quadtree vs JPEG2000
(a) PSNR=30.68dB       (b) PSNR=29.21 dB   (c) R-D Performance

### Residual cameraman image at 0.15 bpp

(a) Prune-join tree    (b) JPEG2000

Note:
- Standard residual coding cannot improve the overall R-D performance because residual image is neither smooth nor geometrically simple.

### Lena image at 0.15 bpp

(a) Prune-join tree (b) JPEG2000

(a) PSNR=30.86 dB (b) PSNR=30.34 dB

Note
- Due to more texture and the small number of large smooth regions, the performance improvement is relatively small.
**Behavior of tree algorithms on piecewise smooth fcts**

**ppf:** piecewise polynomial functions

**psf:** piecewise smooth functions

<table>
<thead>
<tr>
<th>Signal</th>
<th>Oracle Coder</th>
<th>Wavelet Coder</th>
<th>Prune tree Coder</th>
<th>Prune-join tree Coder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-D PPF</td>
<td>$2^{-c_1 \sqrt{R}}$</td>
<td>$2^{-c_2 \sqrt{R}}$</td>
<td>$2^{-c_3 \sqrt{R}}$</td>
<td>$2^{-c_4 \sqrt{R}}$</td>
</tr>
<tr>
<td>2-D PPF</td>
<td>$2^{-d_R}$</td>
<td>$\frac{\log R}{R}$</td>
<td>$2^{-d_2 \sqrt{R}}$</td>
<td>$2^{-d_3 \sqrt{R}}$</td>
</tr>
<tr>
<td>1-D PSF</td>
<td>$R^{-2p}$</td>
<td>$R^{-2p}$</td>
<td>$\frac{\log R}{R}$</td>
<td>$\frac{\log R}{R}$</td>
</tr>
<tr>
<td>2-D PSF</td>
<td>$R^{-p}$</td>
<td>$\frac{\log R}{R}$</td>
<td>$\frac{\log R}{R}$</td>
<td>$\frac{\log R}{R}$</td>
</tr>
</tbody>
</table>

at most log penalty with polynomial complexity
(and a bit more work gets rid of logs...)

**Conclusions**

**Performance**
- oracle like behavior for piecewise polynomials with polynomial boundaries
- similar behavior for piecewise smooth
- initial "practical" coder beats state of the art coder

**Complexity**
- $O(N^2)$

**Other applications**
- model based denoising

**Publications**

**Thesis**

**Papers:**