A Heuristic-Based Decision Tree for Connected Components Labeling of 3D Volumes

Maximilian Söchting¹, Stefano Allegretti², Federico Bolelli² and Costantino Grana²

¹University of Potsdam, Germany
²Università degli Studi di Modena e Reggio Emilia, Italy
Connected Components Labeling (CCL)

- Find all connected, foreground pixel regions within a binary image
- Each pixel region, or **connected component**, receives a unique label
- Fundamental for image segmentation and object recognition
- CCL should be as **fast** as possible
History of CCL Research (1/2)

- Rosenfeld and Pfaltz\textsuperscript{1} invented \textbf{two scans} algorithms
  - First scan: gives each pixel a \textit{provisional label} based on the neighborhood mask, and \textbf{solves label equivalences}
  - Second scan: assigns \textbf{definitive labels}
- Wu et al.\textsuperscript{2} proposed \textbf{Optimal Decision Trees (ODTs)}
  - Label for a pixel can be decided \textbf{without reading} every neighbor pixel
  - Optimal binary decision tree minimizes pixel reads
  - Each tree node represents a pixel read

History of CCL Research (2/2)

• Grana et al.\textsuperscript{1} proposed block-based scan mask
  • In a 2x2 block, all foreground pixels share the same label
  • Generated the ODT automatically since it is unfeasible by hand: 2\textsuperscript{16} cases, 136 nodes

• What about 3D CCL?
  • Multiple possible block-based masks: 2x1x1, 2x2x1 and 2x2x2
  • Explosion in computational complexity makes the ODT generation infeasible
  • No existing 3D CCL algorithm employs a block-based mask
  • Goal: generate a near-optimal tree with a heuristic strategy

Heuristics – Concept

• **Shannon Entropy** (information theory)
  • Given a set of events $E$, with $P_i$ being the probability of an event $i \in E$, the entropy $H_E$ is:
    $$H_E = \sum_i -P_i \log P_i$$
  • Entropy describes the uncertainty of outcomes

• **Decision Tree Learning**
  • Generate decision trees for complex datasets quickly
  • **Recursively partition** the dataset through entropy calculation
    1. Try *splitting* on every attribute
    2. Calculate **Information Gain (IG)** on subsets – IG measures average entropy reduction
    3. Apply *split* with the highest information gain
  • For CCL, the dataset is the decision table

OR-decision table for the Rosenfeld mask
Applying Decision Tree Learning to CCL

- **Entropy Partitioning Decision Tree (EPDT) generated for the Rosenfeld mask**
- For each node, the pixel with maximum IG is chosen

<table>
<thead>
<tr>
<th>Node</th>
<th>Depth</th>
<th>$H(S)$</th>
<th>$p$</th>
<th>$q$</th>
<th>$r$</th>
<th>$s$</th>
<th>$x$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$H_0$</td>
<td>$H_1$</td>
<td>$IG$</td>
<td>$H_0$</td>
<td>$H_1$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2.2</td>
<td>2.0</td>
<td>1.4</td>
<td>0.5</td>
<td>2.3</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2.4</td>
<td>2.0</td>
<td>0.8</td>
<td>1.0</td>
<td>2.5</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2.0</td>
<td>2.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.8</td>
<td>1.0</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

- The result is **near-optimal**
  - Only one more node than the optimal decision tree

- Next step: apply EPDT to 3D block-based masks
3D EPDT algorithms

- New 3D EPDT CCL algorithms
- Varying block size and number of pixels
- **EPDT_19c**
  - Block size 2x1x1
  - Smallest 3D block-based mask
- **EPDT_22c**
  - Block size 2x1x1
  - Add borders pixels, for more efficient actions
- **EPDT_26c**
  - Block size 2x2x1
  - Largest tree that compilers can handle
Experimental Results

- EPDT algorithms improve the performance of the first scan by saving many memory accesses
- EPDT_26c has a very large decision tree → bad impact on instruction cache
- EPDT_22c improves current state-of-the-art\(^1\)

---