A Hierarchical Image Segmentation Algorithm

Wei Yu, Fangting Sun,
and Jason Fritts
Drawbacks of Existing Segmentation Methods

• May not preserve spatial relationships

• Potentially high computational complexity

• Segmentation primarily uses color intensity

• Single condition for when to stop segmentation
Key Aspects of New Hierarchical Segmentation Method

- Preserve spatial relationships
  - use hierarchical segmentation

- Low computational complexity
  - force a specified amount of segmentation at each level of hierarchy

- Segmentation primarily uses color intensity
  - perform wavelet analysis on each $K \times K$ block of pixels
    ($K = 4$ or $8$ typically)
  - provides info on orientation, texture, and energy

- Single condition for when to stop segmentation
  - provide three user-controllable stopping conditions
Target Markets

- Content Based Image Retrieval (CBIR)
- Object-based video compression
  - e.g. MPEG-4
- Pattern recognition
- Moving object tracking
Implementation

- Separate M x N image into a set of non-overlapping blobs
  - each blob contains $K \times K$ pixels
  - blobs are usually small, e.g. 4 x 4 or 8 x 8 pixels
- Perform wavelet on each blob
  - provides orientation, texture, and energy info in addition to color intensity
- Initialize hierarchical tree representation
  - define tree using basic UNION-FIND data structures:
    - each level of tree has a weighted graph, $G_{\text{level}} = (V_{\text{level}}, E_{\text{level}})$
    - each level of tree has a disjoint set of nodes, $S_{\text{level}}$
  - each node, $v_i$ in $V_{\text{level}}$ at lowest level of tree hierarchy represents a blob
  - each edge, $e_i$ in $E_{\text{level}}$ connects a pair of spatially adjacent nodes
  - weight of each edge indicates feature distance between adjacent nodes
  - each set, $s_j$ in $S_{\text{level}}$ points to one or more nodes in that level
    - at lowest level of hierarchy, each set initially points to a single node/blob
Implementation, cont.

• Merge sets at each level of hierarchy
  – first, merge all pairs of sets whose feature distance is less than specified threshold
  – then, merge each remaining set of unit size (i.e. only contains one node) with at least one other set
    • forced set merging ensures low computation complexity

• Create next higher level of tree hierarchy after completing current level
  – create a node, \( v_i \) in \( V_{level} \) for each set \( s_i \) in \( S_{level-1} \)
  – each edge, \( e_i \) in \( E_{level} \) connects a pair of spatially adjacent nodes
  – initialize each set, \( s_i \) in \( S_{level} \) to point to one node, \( v_i \) in \( V_{level} \)

• Repeat above two steps until one of three stopping conditions met:
  – current level contains a single node (i.e. root node of tree)
  – user specified number of segments as \( N \), and current level contains \( N \) nodes
  – user specified feature distance of \( X \), and feature distance between all connected nodes at current level is greater than \( X \)
Hierarchical Tree Representation
Feature Distance

- Each node has 12 feature dimensions
  - Six values for the mean and deviation values of RGB color intensity
  - Six values for the mean and deviation values of texture from the wavelet transform in the horizontal, vertical, and oblique directions
    - for the lowest hierarchy level, only the three mean values from wavelet analysis are used, since no deviation values can be obtained from nodes containing only one blob

- Computation of the feature value for the six mean values \((1 \leq i \leq 6)\) and deviation values \((7 \leq j \leq 12)\), respectively:
  \[
  v.f_i = \frac{1}{\sum_{u \in S} u.size} \sum_{u \in S} u.size * u.f_i \\
  v.f_j = \sqrt{\frac{\sum_{u \in S} u.size * (u.f_{j-6} - v.f_{j-6})^2}{\sum_{u \in S} u.size}}
  \]

- Computation of the feature distance between two adjacent sets:
  \[
  e.dist = \sqrt{\frac{\sum_{i=1}^{12} (v_1.f_i - v_2.f_i)^2}{12}}
  \]
**Computation Complexity**

- Overall computation complexity

\[ O\left( MN + \frac{MN}{K^2} \log\left( \frac{MN}{K^2} \right) \right) \]

- Forcing each set to merge with at least one other set at each level ensures that:
  - the number of nodes in each parent level is no more than half the number of nodes in its child level
  - the number of hierarchy levels is no more than:

\[ \log\left( \frac{MN}{K^2} \right) \]

- Please see paper for full details of computation complexity analysis
Segmentation Examples
Segmentation Examples

[Images of segmented objects]
Conclusions

• Examples confirm that hierarchical segmentation effectively preserves spatial relationships  
  – i.e. there are no discontinuous segments

• Achieved low computational complexity through forced merging  
  – will partially relax the degree of forced merging in future models

• Using wavelet transform on $K \times K$-sized blobs of pixels provides texture and orientation information as well as color intensity

• Developed a flexible stopping mechanism  
  – segmentation stop point may be specified via three methods:  
    • desired number of segments  
    • maximum feature distance for merging segments  
    • unspecified – produces full tree hierarchy representation, which may be then be traversed as necessary to find desired segment separation points
Future Work

• Allow pixel-level segmentation instead of just blob-level segmentation
  – requires special method for computing feature values for pixels in the same blob

• Partially relax requirement that each unit-sized set be merged with at least one other set in each level
  – require at least $N$ percent of unit-sized sets to be merged, so that resulting number of nodes at parent levels is markedly reduced
  – computation complexity will become a function of $N$

• Experiment with different weights for each feature dimension
  – currently weighting is equal across all dimensions

• Use machine learning for feedback-based segmentation
  – machine learning will adjust feature weights as appropriate