Particle Growing Method in Medical Image Segmentation

Youngmin Kim
Hyunyoung Song
University of Maryland
CMSC828V Spring 2005

Medical Image Processing

- Data
  - Sampled representation (Image)
  - Acquired from medical instrumentation as CT or MRI scanners
- Registration
  - Aligns or develops correspondences between data
  - CT scan may be aligned with MRI scan to combine
- Segmentation
  - Identifies and classifies the sampled data

Segmentation Issue

- Several Free Parameters
  - Control the quality of the results
- Needs for the tools that
  - Evaluates the algorithm very fast
  - Visualizes the results very fast when we change the parameters

Previous Works

- ITK (Insight Toolkit)
  - NLM, NIH open source project
  - Leading-edge segmentation and registration algorithm
- GLUI
  - GLUT-based C++ user interface library which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications
- VTK- 3D visualization toolkit (Kitware)
- Utah
  - SCI-RUN – Problem Solving Environment for simulation, modeling, and visualization of scientific problems
  - Bio-PSE – superset of SCIRun, adding capabilities for investigating bioelectric field problems

Latest Papers in Segmentation

- Siggraph papers
  - Ex: Thermo-key-Image Segmentation using Thermal Methods, 2003
- IEEE Visualization 2004
  - Medical Visualization
  - Implicit Surfaces, Level Sets
    - Ex: Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware
- ITK Applications for Segmentation
  - Region Growing
    - Connected Threshold
    - Otsu Segmentation
    - Neighborhood Connected
    - Confidence Connected
    - Isolated Connected
    - Confidence Connected in Vector Images
  - Level Sets
    - Fast Marching Segmentation
    - Shape Detection Segmentation
    - Geodesic Active Contours Segmentation
    - Threshold Level Set Segmentation
    - Canny-Edge Level Set Segmentation
    - Laplacian Level Set Segmentation
  - Hybrid
    - Fuzzy Connectedness and Confidence Connectedness
    - Fuzzy Connectedness and Voronoi Classification
### 1. Region Growing
- Basic segmentation filter in ITK
- Starting from boundary preserving smoothing
  - Removes noise
- Seed Selection
- Region Growing based on two thresholds
  - Upper threshold
  - Lower threshold
- Fast, but we should provide two thresholds

### 2. Level Set Methods
- Level Set Function: \( \psi(X, t) \)
- Generic level-set equation
  \[
  \frac{d}{dt} \psi = -\alpha \nabla \psi - \beta |\nabla \psi| \nabla \psi + \gamma |\nabla \psi| \nabla \psi
  \]
- Track the evolution of contours and surfaces
  - By computing the update to the solution \( \psi \) of the PDE
  - Can omit one or more terms depending on the algorithm
- Typical Way in Practice
  - Contour is initialized by a user
  - Evolve until it fits the form of the segment in the image

### 2-(1) Fast Marching Method
- The simplest level set approach
- Usually used as the initial step for other level set methods
- Propagates a contour from a set of user-selected seed
- Maintains the internal pixel(or voxel) lists
- Contour advances with a speed image
  - Computed from the intensity of input image gradient magnitude
- Very fast in all the level methods, but sometimes it cannot detect the complete segmentation

---

### 2-(1) Fast Marching Preprocess
- Noise Minimization
  - itk::CurvatureAnisotropicDiffusionImageFilter
- Gradient of the Image
  - itk::GradientMagnitudeRecursiveGaussianImageFilter

---

### 2-(1) Fast Marching Preprocess
- Speed Image
  - itk::SigmoidImageFilter
  \[
  I = (I_{\text{Max}} - I_{\text{Min}}) \left( 1 - \frac{1}{1 + e^{-\alpha X - \beta Y}} \right) + I_{\text{Min}}
  \]

---

### Fig 1: Calculate Gradient Magnitude
### Fig 2: Original to gradient magnitude image
2-(2) Shape Detection Level Set Method

- Consider curvature term in general level-set PDE
- Use “Fast Marching Method” as a helper in the determination of an initial level set
- Propagates a contour with a speed
- Segmented shapes are rounder than FastMarching due to the curvature term
- Segment better, but much more slower (about 500~600 times slower) than the fast marching method

3. Hybrid Methods

- Integrates boundary-based and region-based segmentation
  - For example:
    - Fuzzy connectedness: region-based
    - Voronoi diagram classification: boundary-based
- ITK provides two methods
  - (1) Fuzzy Connectedness and Confidence Connectedness
  - (2) Fuzzy Connectedness and Voronoi Classification
- Usually needs two steps
  - Use the first segmentation method as a prior to roughly segment or to estimate
    - Confidence Connectedness in (1)
    - Fuzzy connectedness in (2)
  - Then, use the second segmentation methods
    - Fuzzy Connectedness in (1)
    - Voronoi Diagram in (2)

4. Particle Growing - Algorithm

- Initialization
  - Speed image from ITK filters
  - Generate 4 control points from seed point
- Progress
  - Pick a control pt from TrialQueue
    - If(speedVal > minSpeedValue) march toward the average normal direction ProgressAmt*ProgressVector
    - else add into BoundaryNodeQueue
- Control Point Insertion
  - If(distance btw control pts > DistanceThr) add one control pts in the middle

Problems in previous methods

- Region Growing
  - Needs user intervention to select thresholds
- Level Set
  - Usually slow for correct segmentation
- Hybrid
  - Needs preprocessor that produces a rough segmentation
  - Therefore, slow
- Visualization in all three methods
  - How we can extract polygons from each method for visualization is another matter
4. Particle Growing - Problem

- Intersection
  - Self Intersection

Heuristics to detect and remove intersection
- If (A progress to inside of segmented area)
  - first intersection nodes to second intersection nodes; delete A;
- If (Intersection Nodes are Odd or Even) arrange links accordingly
  - Delete dangling nodes, insert intersection nodes

- Cross Intersection

Resolve Intersection

- Distance Field to Prevent intersection – trial
  - Sample distance value along normal direction of progress
  - If distance value doesn’t increase or decrease monotonously then intersection possibility after the progress

Good! Prevents A and B from potential intersection
Bad! Prevents C from marching toward although intersection doesn’t occur along the path

Performance

- Originally $O(N^2)$ complexity for segmentation
  - Iterate the entire loop to detect intersection (per progress)
- QuadTree
  - Simple spatial partitioning
  - Intersection detected locally
  - $O(N \log N)$ complexity when there is no intersection
  - 3~5 times faster than original version
- Dealing with intersection
  - It takes $O(N)$ time to deal with intersection in the current implementation
  - It makes the entire complexity $O(N^2)$
- We have found a way to process it $O(1)$, but it needs some more data structure (Future Work)
**Result Comparisons**

<table>
<thead>
<tr>
<th>Execution Time</th>
<th>Left Ventricle</th>
<th>Right Ventricle</th>
<th>White Matter</th>
<th>Gray Matter</th>
<th>Quality Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region Growing</td>
<td>0.031</td>
<td>0.031</td>
<td>0.064</td>
<td>0.058</td>
<td>-</td>
</tr>
<tr>
<td>Fast Marching</td>
<td>0.024</td>
<td>0.023</td>
<td>0.021</td>
<td>-</td>
<td>White matter Malfunction</td>
</tr>
<tr>
<td>Shape Detection</td>
<td>1.611</td>
<td>1.574</td>
<td>13.541</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Particle Growing</td>
<td>0.005</td>
<td>0.004</td>
<td>0.088</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

System: Intel Pentium 4 CPU 1.5 GHz, 2GB RAM

**Evaluation of Particle Growing**

- Quality of Segmentation
  - Hard to evaluate the quality
  - Dependent on Speed Input Image
  - Dependent on the other parameters
    - Min-Speed, Distance Threshold, Progress Amount
- Computation Time
  - Very fast in segmenting convex region
  - Slow in very complex region
    - Currently, 2 times slower than Fast Marching
    - Self-intersections happen a lot
    - Not inherent in the algorithm, but due to the implementation

**Conclusion**

- Present a new segmentation method using particle growing
- Use speed input image for propagating particles
- Very fast for segmenting convex regions
- Very easy to visualize the segmented parts
  - Draw the lines in 2D
  - *Draw triangles in 3D (Future work)*

**Future Works**

- Fix some bugs in processing intersections
  - The program breaks sometimes
- Apply efficient data structure to help process self-intersection
  - Makes the entire algorithm work in $O(M\log N)$ time
- 2D to 3D extension
- Parameter adjustment and test variety of 2D and 3D images