Parallel Object Recognition System

Bor-Yiing Su
Multicore Group
Computer Science Lab
SISA
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Object Recognition

Trained Categories
- Bottles
- Apple Logos
- Mugs
- Swans
- Giraffes

Image Queries

Outputs

Object Recognition System

SISA CSL “Innovate by doing”
Object Recognition System

- It is a computational intensive problem
- An algorithm developed by Prof. J. Malik and his students can achieve 87.1% accuracy on the ETHZ shape database.
- However, it takes 5.5 minutes to identify the objects in one 0.15M-pixel image
- How about a database with 100 images? We need parallelism!
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Computation Flow

1. Input Image
2. Contour Detection
3. Image Segmentation
4. Feature Extraction
5. Trained Data
6. Classification
7. Object Bounding Box
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Image Contour Detection

- State-of-the-art algorithm which achieves the highest accuracy: gPb
Parallel Contour Detector

- For more information, please see: Bryan Catanzaro, Bor-Yiing Su, Narayanan Sundaram, Yunsup Lee, Mark Murphy, Kurt Keutzer, “Efficient, High-Quality Image Contour Detection,” ICCV, 2009.
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Image Segmentation

- Contours are not closed. We need closed segmentations for defining objects.
- The UCM (ultrametric contour map) approach can find segmentations without sacrificing accuracy of the contours.
From Contours to Segmentations

Input Contours

Watershed Algorithm

Node and Edge Extraction

Edge Linearization

Edge Reweighting

UCM Clustering

Output Segmentations
Watershed Algorithm

1. Input Contours
2. Local Minimum Extraction
3. Label Assignment
4. Watershed Propagation
5. Watershed Boundaries
   - BFS Graph Traversal
   - BFS Graph Traversal
   - Priority Queue
Parallel Watershed Algorithm

• Apply parallel structured grids algorithm for BFS graph traversal
• **A new idea for parallelizing BFS graph traversal**
• Main idea: each grid updates its state according to the previous stage
• Advantage: Achieves the largest parallelism, highly scalable, no race condition
• Disadvantage: Some redundant work
• Optimally, if we can have the number of processors equal to the number of pixels in the image, computational complexity is $O(\text{dist}(\max(n1, n2)))$
Parallel Watershed Algorithm

- Apply parallel structured grids for local minimum extraction and label assignment
- For watershed propagation, iteratively apply the structured grids traversal from gray level 0 to gray level 255
Edge Extraction and Reweighting

- Watershed Algorithm
- Node and Edge Extraction
- Edge Linearization
- Edge Reweighting
- Convolution
- BFS Graph Traversal
- BFS Graph Traversal
- Assignment
Parallel Edge Extraction and Reweighting

- Use parallel structured grids for BFS graph traversal
- Node extraction: Convolve each pixel with a filter
- Edge extraction, edge linearization: Apply parallel structured grids
- Edge Reweighting: Reweigh each pixel according to its orientation
- Alternative: After edge extraction, first gather the extracted edges, then do the edge linearization and the edge reweighting
UCM Clustering

- **Main idea:**
  - Calculate the average weight on each edge
  - Extract the edge with the lowest weight
  - Cluster regions connected by the edge and reweight the boundary by the average weights of the boundary
  - Repeat the previous 2 steps until all segments are clustered
- It is a priority queue problem, very restricted parallelism
- Usually there are only several hundreds of segments, not worth parallelization
  - Apply serial implementation

![Reweigh all edges](image1)
![Reweigh the purple segment](image2)
![Reweigh the blue segment](image3)
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Feature Extraction

• Given the segmentation, we need to collect features that are representative of the objects

• Challenge: Because of the over segmentation, an object is divided into several segments. How can we define the proper segmentation that corresponds to an object?

• Solution: Segmentation thresholding and segmentation clustering
  – Thresholding: Only keep boundaries with edge weight $\geq$ threshold
  – Clustering: Recursively cluster regions until all the segmentations are clustered
  – Collect features of the segments after thresholding and the clustered segments

From Segmentations to Features

- Input Segmentations
- Thresholding
- Clustering
- Feature Extraction
- Features of Segments
- Assignment
- Priority Queue
- Histogram Accumulation
Thresholding and Clustering

- Thresholding is to turn off pixels that have weights smaller than a threshold: Apply data parallel to do the assignment.
- Clustering is very similar to UCM approach. It iteratively clusters regions with the smallest edge weight until all the segments are clustered: Similar to UCM, apply serial clustering.
- We will have a hierarchical of segments after thresholding and clustering.

![Diagram of Thresholding, Clustering, Assignment, and Data Parallel processes]
Feature Collection

• In general
  – More features: More accurate, but more computation is required
  – Less features: Less accurate, but can be calculated fast

• The current object recognition system only collects one feature
  – Contour features on the 16 grids and 8 orientations
    • Advantage: Invariant against shifting and scaling
    • Disadvantage: Not good at rotating and flipping
Parallel Feature Extraction

- Possible choices:
  - Atomic operation: For each pixel, add its value into the histogram bin using atomic operations
    - Too many conflicts in a grid, not very efficient
  - Geometric decomposition: We have 16 grids and 8 orientations for each segmentation, results in $16 \times 8 = 128$ bins of the histogram. Assign the accumulation task for one bin to one thread
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Classification

• Problem: Each exemplar is represented by a hierarchical tree of segments, the query image is also represented by a hierarchical tree of segments. How can we identify the objects in the query image?
• Solution: Apply the Hough voting algorithm
Hough Voting Algorithm (1/2)

- Step 1: Generate a distance table of each pair of segments between the query image and an exemplar

```
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Distance Table
Hough Voting Algorithm (2/2)

- Step 2: Given a paired matched region (I, J). Let I be a segment of exemplar, J be a segment of query. Calculate the bounding box of object in query by using the center of gravity of I, J, and bounding box of the object in exemplar.
- Step 3: The bounding box that most of the matched regions agree on was declared as the bounding box of the object in query image.
Computation of Hough Voting Algorithm

- Exemplar Region Features
- Pair-wise Chi-square Distance
- Hough Transformation
- Mean Shift Clustering
- Object Bounding Box

- MapReduce
- Simple Arithmetic
- Dense Linear Algebra
Parallel Hough Voting

- For the pair-wise distance calculation, we calculate each pair of features in parallel.
  - When calculating the chi-square distance between a pair of feature, we use data parallelism for implementing parallel map reduce computation.
- For bounding box transformation, we calculate the bounding box for each pair of features in parallel.
- The voting is done by mean shift clustering. It is parallelized by parallel dense linear algebra BLAS libraries.

Pair-wise Chi-square Distance

Bounding Box Transformation

Bounding Box Voting

MapReduce

Simple Arithmetic

Dense Linear Algebra

Data Parallel

Task Parallel

Data Parallel
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Speedup Against Serial Implementation

<table>
<thead>
<tr>
<th>Component</th>
<th>Original (Core i7)</th>
<th>This Work (GTX 280)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contour Detection</td>
<td>236.7</td>
<td>1.822</td>
<td>130x</td>
</tr>
<tr>
<td>Region Extraction</td>
<td>2.27</td>
<td>0.357</td>
<td>6.36x</td>
</tr>
<tr>
<td>Feature Extraction</td>
<td>7.97</td>
<td>0.279</td>
<td>28.57x</td>
</tr>
<tr>
<td>Hough Voting on 127 Exemplar Images</td>
<td>84.13</td>
<td>1.688</td>
<td>49.84x</td>
</tr>
<tr>
<td>Total</td>
<td>331.07</td>
<td>4.146</td>
<td>79.85x</td>
</tr>
</tbody>
</table>

- For region extraction, the parallel structured grids computation introduces some redundant work, and the UCM clustering procedure is still in serial, so the speedup is not very significant.
- For feature extraction, the region clustering procedure is in serial, so the speedup is still limited.
- We finally get 80x speedup on the overall classification procedure.
Run on 9400M G (2-Core GPU)

**Computation Time:** 66.8 seconds
Run on 9800 GTX (16-Core GPU)

Computation Time: 7.66 seconds
Run on GTX 280 (30-Core GPU)

Start Watershed image size: 159506
Max gph value 9996
$>$ findMax | 3.781000 | ms
Local Min Iter 87
Labeling Iter 80
Watershed Iter 891
$>$ watershed | 173.593994 | ms
CUDA Status: no error
Skeletonizing ...
Skeletonization Iteration = 4
Edge Extraction Iter 110
Edge subdivive local Iter 98
$>$ edge extraction | 120.198804 | ms
CUDA Status: no error
New Labeling Iter 207
$>$ ucm clustering | 94.075996 | ms
CUDA Status: no error
New New Labeling Iter 389
regCount 131 regionsize 143
RegCount 131
$>$ combine region | 115.027000 | ms
CUDA Status: no error
$>$ Feature Extraction | 71.136002 | ms
CUDA Status: no error
bblist size 79
newBB size 3
bblist size 33
newBB size 3
bblist size 83
newBB size 4
bblist size 127
newBB size 5
bblist size 63
newBB size 6

Found Bounding Box: CategoryId 1 Left-Upper Position (177, 114) Size (112, 130), Score(5.116283)
Found Bounding Box: CategoryId 4 Left-Upper Position (35, 111) Size (334, 329), Score(3.581726)
$>$ Computation time: | 3.322382 | seconds

Computation Time: 3.32 seconds
Intermediate Results

- Contours
- Watershed Boundaries
- Segmentations
- Bounding Box of recognized object
Outline

• Object Recognition
• Parallel Object Recognition System
  – Top Level Computation Flow
  – Contour Detection
  – Image Segmentation
  – Feature Extraction
  – Classification
• Experimental Results
• Conclusion
Conclusion

• We have developed a highly parallel object recognition system
• A new method for parallelizing graph traversal has been proposed
• A new method for parallelizing watershed algorithm has been proposed
• The parallel object recognition system reduces the runtime from 5.5 minutes to 4.2 Seconds
• The implementation is highly scalable, with core numbers from 2 → 16 → 30, we get runtime from 66.8s → 7.66s → 3.32s
Thank You!