Hardware/Software Partitioned Implementation of Real-time Object Oriented Camera for Arbitrary-shaped MPEG-4 Contents

Jisu Kim

Computer Architecture and Parallel Processing Lab.
School of EECS
Seoul National University
Seoul, Korea
Contents

- Introduction
- Overview of the Proposed Algorithm
- Low Complexity Segmentation Algorithm
- Foreground Object Extraction
- HW/SW Partitioned Implementation
- Evaluation
- Conclusion
Background

- Why Video Object Segmentation is required?
  - To support object-oriented video compression technology, like MPEG-4 standard (Core, Main Profile)
  - To support content-based video processing application

- What’s the problem with Video Object Segmentation technology?
  - Video Object Segmentation technology doesn’t catch up with most recent video compression technology until now
  - Many approaches have been proposed, but they are far from practical scenarios, such as mobile video conferencing
Previous Works & Objectives

- Previous Works Examples
  - Advanced techniques using spatio-temporal information
    - Accuracy is good enough, but computational complexity is high
  - Background registration
    - Computational complexity is relatively low
    - But some constraints are existed: No camera motion, Pre-registered background image

- My Objectives
  - Low complexity
    - Enough for real-time processing
    - With embedded system, targeting mobile devices
  - Quality of results
    - Almost accurate object masks compared with other approaches
    - For real scenarios, like hand-held mobile devices
General View

- General view of the proposed object segmentation system

Input Images → Image Partitioning → Object Region Selection → Extracted Object

- To get exact boundaries of objects and to take advantage of the region-based processing
- To extract objects with the characteristics of interests from input images
The Flow of the Region Segmentation

1. **Image Smoothing**
   - Morphological Open-Closing

2. **Gradient Calculation**
   - Morphological Gradient Calculation
   \[ G(f) = (f \oplus B) - (f \ominus B) \]

3. **Scene Cut Detection**
   - Current frame is I-type, if the total scene is changed
   - Total scene is changed, if the count of background pixels that changed is larger than predetermined threshold
Marker Extraction

Initial Marker Extraction
- Connected set of pixels of which gradient levels are less than threshold
- For effective region segmentation

Predictive Marker Extraction
- For temporal coherency of results
- Segmentation results are only updated for pixels which are changed
- Can reduce computational complexity
Watershed Transform & Region Merging

Watershed Transform
- Marker-based watershed transform is performed

Region Merging
- Results of watershed transform are tend to be over-segmented
- The difference of the average color and the ratio of weak edges are tested for merging two adjacent regions
The Flow of the Foreground Object Extraction

- Spatial Assumption
  - Foreground Object Mask Definition
  - Applying Mask to RAG
  - Classifying Undefined Regions into FG or BG
  - Final Object Classification Result

- Region Segmentation Result
  - Region Adjacency Graph Generation

- Foreground Object Mask Definition
  - Based on assumption that object is usually positioned at the center
  - Three groups: {Foreground, Background, Undefined}
Foreground Object Extraction (Continued)

- **Region Adjacency Graph Generation**
  - Based on region segmentation result
  - Graph vertices: Regions
  - Graph edges: Adjacency weights

- **Applying Mask to RAG**
  - Classify vertices into three groups
    - {Foreground, Background, Undefined}

- **Classification of Undefined Areas**
  - Classify undefined areas as foreground or background
  - Undefined vertex with minimal weight is first processed
Profiling Result

- **Profiling Environment**
  - ARMulator ADS 1.2
- **Target**
  - ARM7TDMI, 100MHz

- Preprocessing step occupies about 70% of total computation
  - Preprocessing step includes “Image Smoothing” and “Gradient Calculation”, which are both of morphological operation
Grayscale Morphological Operation Analysis

- **Image Smoothing**: Opening followed by Closing
  - Opening (Dilation after Erosion): \((f \ominus b) \oplus b\)
  - Closing (Erosion after Dilation): \((f \oplus b) \ominus b\)

- **Gradient Calculation**: 
  \[ g = (f \oplus b) - (f \ominus b) \]

\[
(f \oplus b)(s, t) = \max \{ f(s - x, t - y) + b(x, y) \mid (s - x), (t - y) \in D_f; (x, y) \in D_b \}
\]

\[
(f \ominus b)(s, t) = \max \{ f(s + x, t + y) - b(x, y) \mid (s + x), (t + y) \in D_f; (x, y) \in D_b \}
\]

- **Analysis on Morphological Operation**
  - 3x3 window operation
    - Access 9 pixels for 1 pixel processing
    - Can be easily implemented using hardware
  - Data chaining
    - Results of one operation are inputs of next operation
    - Hardware can reduce burden of data read/write

**About 90% of Processing Time can be saved with hardware accelerator**
The Proposed System Implementation

- Gate count of the Morphological Operation Unit : 16,612
- Program size : 18.9Kbytes
- Required memory : under 100Kbytes
Simulation Environments & Processing Time

- RTL Simulation Environments
  - ARM7TDMI processor with hardware accelerator @ 100MHz
  - QCIF (176x144) size test sequences

- Processing Time
  - For first 6 frames of “Akiyo” sequence

Table 1. Processing time

<table>
<thead>
<tr>
<th>frame #</th>
<th>frame type</th>
<th>Processing time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I</td>
<td>88.390</td>
</tr>
<tr>
<td>2</td>
<td>P</td>
<td>60.239</td>
</tr>
<tr>
<td>3</td>
<td>P</td>
<td>58.010</td>
</tr>
<tr>
<td>4</td>
<td>P</td>
<td>58.059</td>
</tr>
<tr>
<td>5</td>
<td>P</td>
<td>58.901</td>
</tr>
<tr>
<td>6</td>
<td>P</td>
<td>58.778</td>
</tr>
</tbody>
</table>
Simulation Results

Fig. 9. I-type foreground extraction results. (a) Original image; (b) gradient calculation; (c) initial marker; (d) WT; (e) region merging; (f) final foreground extraction result.

Fig. 10. P-type Foreground extraction results. (a) Original image; (b) change detection mask; (c) predictive marker; (d) WT followed by region merging; (e) final foreground extraction result.

Fig. 11. Fixed and moving camera comparison. (a) “Claire” image (fixed camera); (b) foreground object extraction; (c) “Foreman” image (moving camera); (d) foreground object extraction.
**Comparisons with Previous Works**

- **Comparison of Subjective Quality**

  (a) Lee’s method  
  (b) Chien’s method  
  (c) Proposed method

- **Comparison of Processing Time and Test Environment**


<table>
<thead>
<tr>
<th>Test environment</th>
<th>Picture Size</th>
<th>Processing time per frame (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) AMD 1.5GHz processor</td>
<td>368×240</td>
<td>9,000 ~ 65,000</td>
</tr>
<tr>
<td>(b) 800MHz Pentium-III</td>
<td>176×144</td>
<td>90 ~ 100</td>
</tr>
<tr>
<td>Proposed work</td>
<td>176×144</td>
<td>55 ~ 90</td>
</tr>
</tbody>
</table>
Conclusion & Future Works

■ Conclusion
  - A real-time foreground object segmentation algorithm that can process QCIF size video
  - HW/SW partitioned implementation in a SoC consisting of ARM7TDMI processor with hardware accelerator
  - Simulation shows that 15 fps speed is achieved at the frequency of 100MHz, and almost accurate results can be obtained

■ Future Works
  - Advanced algorithm and architecture that are closely combined with practical video encoder system
  - Several object selection algorithm levels that are optionally chosen according to available computing resources