Accelerating Computer Vision Algorithms for Medical Applications on STHORM Platform

T. L. B. Cheikh $^1$  J. Trajkovic $^2$  G. Nicolescu $^1$
Y. Bouchebaba $^3$  P. Paulin $^3$

$^1$ Ecole Polytechnique de Montreal
$^2$ Concordia University
$^3$ ST Microelectronics
Agenda

1. Context

2. Augmented Reality System
   1. Canny Edge Detection
   2. Shape Refining
   3. Euclidean Distance Mapping

3. Experiments & Results

4. Conclusions & Future Work
Minimally Invasive Surgery: Today’s Surgery Trend

Benefits

- Less scarring & Injury to Tissue
- More Effective Surgery
- Fast Recovery and Shorter Hospital Stay
Minimally Invasive Surgery: Today’s Surgery Trend

Benefits

- Less scarring & Injury to Tissue
- More Effective Surgery
- Fast Recovery and Shorter Hospital Stay

Shortcomings

- Narrow Filed of View
- Loss of Depth (No 3D View)
- Need of Long Training Hours
Augmented Reality System for MIS

**Context and Application**

**Basic Steps**

- **Principle:** Build 3D View of Surgery Scene
  1. Determine 2D Position of the Instrument in Captured Frame
  2. Correlate Instrument Position with Camera Motion
Augmented Reality System for MIS

Benefits

- Less Efforts Needed from Surgeons
- High Accuracy and More Effectiveness
- Image Guided Surgical Robots Facility
Augmented Reality System for MIS

Context and Application

Challenges

- Computationally Demanding Algorithms
- Continuous Asking for Improving Accuracy
- Need of Real-time or Near Real-time Execution
Augmented Reality System for MIS

Our Role

- Accelerate Involved Algorithms
- Provide Optimal Implementations on Several Hardware Platforms
- Build a Set of Programming Strategies
Augmented Reality System for MIS

Our Role

- Accelerate Involved Algorithms
- Provide Optimal Implementations on Several Hardware Platforms
- Build a Set of Programming Strategies

Approach and Target Hardware Platforms

- Go with Parallelization Approach
- Mainstream Multiprocessors Architectures: Multi-core CPUs and Many-core GPGPUs
- Embedded Multiprocessors Architecture: STHORM
Augmented Reality System

Involved Algorithms

- **Basic Image Processing Operations** (Filtering, Morphological Operations, Distance Calculation)

- **Commonly Used Algorithms** (Image Segmentation, Camera Calibration, Edge Detection, Euclidean Distance Mapping, Shape Refining)
2. Augmented Reality System
Image Segmentation

1. Input image
2. Detect objects in endoscope image
3. Identify instrument
4. Track instrument
5. Next frame
6. Intermediate image
7. Output image

Medical Imaging Application
Image Segmentation

- Edge Detection
- Refining Edges
- Skeletons Extraction
- Refining Skeletons
- Skeletons Grouping

Input Image 1
Input Image 2
Output Image 1
Output Image 2
### Application: Canny Edge Detector

<table>
<thead>
<tr>
<th>Involved Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Canny Edge Detection</strong></td>
</tr>
<tr>
<td>Filtering Operations</td>
</tr>
<tr>
<td>Data-parallel + Sequential</td>
</tr>
<tr>
<td>Structured Grids</td>
</tr>
</tbody>
</table>
1. Canny Edge Detection
### Generalities

- Multi-stage Edge Detector Algorithm
- Considered as One of the Most Accurate Edge Detectors
- Preprocessing Step in Several Computer Vision Applications

### Known Implementations

- Intel Computer Vision Library OpenCV (C Language)
- MATLAB Image Toolbox (MATLAB Language)
- GPU Implementations
Application: Canny Edge Detector

Previous Implementations & Accuracy

Input Image

OpenCV Implementation

MATLAB Implementation

GPU Implementation (2008)

GPU Implementation (2010)

Our Implementation
Application: Canny Edge Detector

Application Diagram

- Gaussian Blur
- Gradient at X Direction
- Gradient at Y Direction
- Magnitude Calculation
- Non Maxima Suppression at Direction 1
- Non Maxima Suppression at Direction 2
- Non Maxima Suppression at Direction 3
- Non Maxima Suppression at Direction 4
- Edge Points Connection
Parallelization Strategy

Parallelization Opportunities

Section 1

Gaussian Blur
  Gradient at X Direction
  Gradient at Y Direction

Magnitude Calculation

Non Maxima Suppression at Direction 1
  Non Maxima Suppression at Direction 2
  Non Maxima Suppression at Direction 3
  Non Maxima Suppression at Direction 4

Edge Points Connection

Sequential

Data-Parallelism

Parallelization Opportunities

Application: Canny Edge Detector
Canny Edge Detection

Computations and Dependences

Section 1

Section 2
2. Shape Refining
Thinning

Input Image → Thinning Edges → Output Image
Shape Refining

Computations and Dependences

More refining is needed?

Yes

Refining Shapes
Tiling Pattern

while(condition)
{
    Function_1;
    Function_2;
}

Input Data

inImg
Lookup Table

while(condition)
{
    Function_1;
    Function_2;
}

Thinning

First Pass
Second Pass
Output Image: Unchanged?
Yes
No

Thinning

Slice 0
Slice 1
Slice N

Thinning

Final Image
Canny Edge Detection + Thinning
Implementation on STHORM

Canny Edge Detector Execution Flow

Input Image

STHORM Architecture
3. Euclidean Distance Mapping
Euclidean Distance Mapping

Application Diagram

- Input Image
- Distance Calculation
- Distance Update
- Output Image
Euclidean Distance Mapping

Computations and Dependences

Pre-computed Distances Table

Image with one Edge → ROI around Edge → Updated Distances

Distance Map around two Edges → Skeleton Extraction
Euclidean Distance

Implementation on STHORM

Global Memory

List of Edges #1

List of Edges #2

List of Edges #3

List of Edges #4

Edge #1

Edge #2

Edge #3

Edge #4

STHORM Architecture

Application: Canny Edge Detector
3. Experiments & Results
## Experiments & Results

<table>
<thead>
<tr>
<th>Multiprocessor Platform</th>
<th>Canny Edge Detection (512x512)</th>
<th>Euclidean Distance Mapping (177 Edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-core CPU (8 cores, 2 GHz) (Sequential)</td>
<td>152 ms</td>
<td>121 ms</td>
</tr>
<tr>
<td></td>
<td>1x</td>
<td>1x</td>
</tr>
<tr>
<td>Multicore-CPU (8 cores, 2 GHz) (Parallel)</td>
<td>21 ms</td>
<td>21 ms</td>
</tr>
<tr>
<td></td>
<td>7.23x</td>
<td>5.76x</td>
</tr>
<tr>
<td>Many-core GPU NVIDIA GTX 590 (512 cores, 1250 MHz) (Parallel)</td>
<td>4 ms</td>
<td>12 ms</td>
</tr>
<tr>
<td></td>
<td>38x~(60.8)</td>
<td>10.08x~(16.21)</td>
</tr>
<tr>
<td>Embedded Platform STHORM (64 cores, 450 MHz) (Sequential)</td>
<td>2234 ms</td>
<td>2273 ms</td>
</tr>
<tr>
<td></td>
<td>1x</td>
<td>1x</td>
</tr>
<tr>
<td>Embedded Platform STHORM (64 cores, 450 MHz) (Sequential)</td>
<td>141 ms</td>
<td>81 ms</td>
</tr>
<tr>
<td></td>
<td>15.84x~(23.48x)</td>
<td>28.06x~(41.59x)</td>
</tr>
</tbody>
</table>
4. Conclusions & Future Work
Conclusions & Future Work

Work Summary

- Functional Validation of Computer Vision Applications on STHORM Board
- Acceleration of Computation Intensive Algorithms using Parallelization Strategies Suitable for STHORM

Future Work

- Test of New Optimization Techniques
- Explore New Parallelization Strategies
- Implement More Computer Vision Algorithms on STHORM
Thank you