HW Implementation of MRF MAP Inference on an FPGA Platform

Jungwook Choi and Rob A. Rutenbar

FPL 2012
Aug. 30 2012
Overview

• Goal: Accurate & fast HW MRF MAP solver
  – Why MRF MAP inference and HW impl.? 
  – Loopy belief propagation
  – Tree-reweighted message passing (TRW-S)
  – Our TRW-S hardware architecture
  – FPGA experimental results (x30 faster than SW)
  – Conclusion & future work
MRF MAP Inference

Maximum a posteriori (MAP)

\[
\arg\max_x P(x|y) = \arg\max_x P(y|x)P(x)
\]

Energy minimization on Markov random fields (MRF)

\[
\arg\min_x \left( \text{Energy} (x) \right)
\]

Label assignments

Observations

3D depth map by matching pixels along the line

Likelihood \rightarrow Data cost

Prior \rightarrow Smoothness cost

3D depth map by MRF MAP inference
MRF MAP Inference

Maximum a posteriori (MAP)

\[
\arg\max_x P(x|y) = \arg\max_x P(y|x)P(x)
\]

Energy minimization on Markov random fields (MRF)

\[
\arg\min_x \text{Energy}(x)
\]

Label assignments

Observations

Posterior

Likelihood

Prior

Ground truth

Greedy method (Iterated conditional modes)

Belief propagation (BP)

Tree-reweighted (TRW)

Energy minimization on MRF

Images from http://vision.middlebury.edu/MRF/results/tsukuba/
Why Custom Hardware Impl.?

- Many apps map to a *common* MRF framework
- Computation is *local*, well matched for custom HW
Loopy Belief Propagation

- In **BP**, a node propagates *belief* to neighbors by passing *messages*
  - Message: “based on what I know now, what do I tell to my neighbor?”
  - Belief: “what label should I choose based on my neighbors?”
  - Energy computed by the best labels based on beliefs

- **BP** on a tree
  - Optimum energy can be found after all *inward/outward message passing* is done

- **BP** on a loopy graph
  - No guarantee of optimality due to *loops*
Tree-Rewighted Message Passing

- Idea: decompose a loopy graph to a set of trees

Energy is the weighted sum of tree energy

\[ \text{Energy}(\mathbf{x}) = \sum_{\text{trees } T} \omega_T \cdot \text{Energy}_T(\mathbf{X}_T) \]

- Equality \(\rightarrow\) optimum energy!
- New goal: maximize this lower bound

Goal: minimize overall energy

\[ \min_{\mathbf{x}} \text{Energy}(\mathbf{x}) \leq \sum_{\text{trees } T} \omega_T \cdot \left( \min_{\mathbf{x}_T} \text{Energy}_T(\mathbf{X}_T) \right) \]

BP on a tree \(\Rightarrow\) No loop!!
Sequential TRW (TRW-S)

- New goal: maximize lower bound by data cost update & message passing on trees
- Sequential message passing \(\Rightarrow\) convergence property
  - Lower bound is guaranteed not to decrease
  - \(\Rightarrow\) More chance to find the optimum energy!!

- Data cost update
  \[ \hat{\theta}_p(x_p) = d_p(x_p) + \sum_{s \in \text{Nb}(p)} M_{sp}(x_p) \]

- Message passing
  \[ M_{pq}(x_q) = \min_{x_p} \{ (\gamma_{pq} \cdot \hat{\theta}_p(x_p)) - M_{qp}(x_p) \} + V_{pq}(x_p, x_q) \]

- Challenge: parallelize “sequential message passing”
Comparison: BP-M and TRW-S

- Benchmark: Flower stereo images* (360x262x16 label)
  - BP-M: min-sum belief propagation, run 80 iterations.
  - TRW-S: sequential tree reweighted message passing, run 80 iterations.

*From stereo movie sample, http://www.stereomaker.net/sample/index.html
Streaming TRW-S HW Architecture

- Key: diagonal ordering of message passing for parallelism
- Decoupled, streaming arch.
- Launch/retire 1 pixel/clock
  - Complete label-set likelihood updates for all labels
- Deep pixel-proc pipeline
  - 14 stages deep
  - So: 14 pixels “in flight” / clock
Streaming TRW-S HW Architecture

- Pipelined message passing

\[ H_p(x_p) = d_p(x_p) + \sum_{s \in \text{Nb}(p)} M_{sp}(x_p) \]

\[ M(j) = \min \left\{ H'_p(j), H'_p(j \pm 1) + \lambda, H'_p(j \pm 2) + 2\lambda, \min\{H'_p(j)\} + 2\lambda \right\} \]
Experimental Platform: FPGA

- Our platform: Convey HC-1
  - Host-FPGA cache-coherent virtual memory system
  - Max memory BW: 1Kbit/cycle (~20GB/sec)/FPGA (runs @150MHz)
Experimental Results

• Stereo matching of Middlebury benchmark*
  – Speed (per iteration)
    • FPGA impl. of streaming TRW-S (F-sTRW-S) runs in Convey HC-1 (@ 150MHz)
    • SW impl. [Szeliski 2008] runs in Intel Core i7 (@ 1.87GHz)

<table>
<thead>
<tr>
<th>Task</th>
<th>Task Size</th>
<th>Cost Fn.</th>
<th>Our HW: F-sTRW-S</th>
<th>SW Impl.*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tsukuba</td>
<td>384x288x16L</td>
<td>Truncated linear</td>
<td>478,134 cy</td>
<td>0.0032 sec</td>
</tr>
<tr>
<td>Venus</td>
<td>434x383x20L</td>
<td>Truncated quadratic</td>
<td>1,436,257 cy</td>
<td>0.0096 sec</td>
</tr>
<tr>
<td>Teddy</td>
<td>450x375x60L</td>
<td>Potts model</td>
<td>2,914,599 cy</td>
<td>0.0194 sec</td>
</tr>
</tbody>
</table>

– F-sTRW-S is 34.5~49.0 times faster than SW impl.

Experimental Results

- Stereo matching of Middlebury benchmark (cont’d)
  - Comparison of 3D depth maps after 500 iterations

- F-sTRW-S speeds-up SW impl without loss of quality of results
Experimental Results

- Rough comparison with other VLSI impl. [Liang 2011]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Tile-based BP*</th>
<th>F-sTRW-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spec.</td>
<td>320x240x64L</td>
<td>384x288x16L (max: 512x512x64L)</td>
</tr>
<tr>
<td>Num. of Iteration</td>
<td>(B, T₁, T₀) = (16, 20, 5)</td>
<td>T₀ = 5</td>
</tr>
<tr>
<td>Minimum Energy</td>
<td>396,953</td>
<td>393,434</td>
</tr>
<tr>
<td>Speed</td>
<td>7.28 frames/sec</td>
<td><strong>38.32 frames/sec</strong></td>
</tr>
</tbody>
</table>

- F-sTRW-S shows compelling speed and inference capability

## Experimental Results

- **Comparison of speed with other GPU impl.**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>NVIDIA GeForce 7900 GTX</td>
<td>NVIDIA GeForce 8800 GTS</td>
<td>NVIDIA GeForce GTX 260</td>
<td>N/A</td>
</tr>
<tr>
<td># Iteration</td>
<td>(4 coarse to fine scales) = (5,5,10,20)</td>
<td>(B, T₁, T₀) = (16, 20, 5)</td>
<td>(3 coarse to fine scale) = (9,6,2)</td>
<td>T₀ = 5</td>
</tr>
<tr>
<td>Time (ms)</td>
<td>79.71</td>
<td>124.38</td>
<td>61.41</td>
<td>26.10</td>
</tr>
</tbody>
</table>

- F-sTRW-S outperforms other GPU impl. in speed.

---

Conclusion & Future work

• Conclusion
  – The FIRST custom hardware implementation of Sequential tree-rewighted message passing (TRW-S) algorithm is introduced.
  – Our streaming TRW-S implementation shows not only compelling speed but also superior quality of results compared to other belief propagation implementation on VLSI and GPU.

• Future work
  – Streaming video-rate TRW-S stereo matching engine
  – Expand Streaming TRW-S for more apps
Key References


Thank You