Fast and Adaptive BP-based Multi-core Implementation for Stereo Matching

The all-around winner

Armin Ahmadzadeh, Hatef Madani, Kianoush Jafari,
Farzad Salimi-jazi, Shervin Daneshpajouh, Saeid Gorgin

Present By:
Farzad Salimi-Jazi
Outline

- Introduction
- Optimization techniques
- Implementation method
- Experimental results
- Conclusion
Introduction

Loopy Belief Propagation  Message passing [1]

The closer The whiter

Disparity Map
Belief Propagation Message passing

- Each vertex has 4 messages from its right/left/up/down edges and a data cost.
- Each message should update L labels (16)
- We do this in some iterations (t)
  - In each iteration we update all of vertexes (n)
    - In each message we update L labels using sender’s values
    - Finding better probability in $O(L^2)$

```
struct Vertex {
    TYPE msg[5][LABELS];
};
struct MRF2D {std::vector<Vertex> grid;};
```

$O(t \cdot n \cdot L^2)$
Our Optimization techniques

- Min-convolution
- Hierarchy of vertices
- Bipartite method
- Bi-direct message passing
Min-convolution

- Using Min-convolution instead of nested loops to find better probabilities.

\[
\begin{align*}
\text{for } f_q \text{ from } 1 \text{ to } k - 1 : \\
m(f_q) &\leftarrow \min(m(f_q), m(f_q - 1) + s).
\end{align*}
\]

\[
\begin{align*}
\text{for } f_q \text{ from } k - 2 \text{ to } 0 : \\
m(f_q) &\leftarrow \min(m(f_q), m(f_q + 1) + s).
\end{align*}
\]

Linear algorithm for finding the min value for suitable label [2]

\[O \left( t \cdot n \cdot L^2 \right) \quad \rightarrow \quad O \left( t \cdot n \cdot L \right)\]
Hierarchy of vertices

- Original idea, several layers
  - like pyramid

- BP is used in coarse layers and data is passed to finer layers

- Data combination and migration between layers
Reduce Layers in 2 layers

- **Motivation:**
  - Using modified hierarchical BP
  - Reduce number of iterations

- **Idea:**
  - Concentrate coarser Level (Layer 1)
  - Reduce computation time rather than finer levels
  - Create two layers: Coarse and Fine.

Also, we have a **variable** scale factor
Bipartite method

- Bipartite method’s two subsets
- Performance improvement
- Reducing memory requirement by half

MRF graph split into two subsets. Subsets are specified through different colors. In each iteration one of the subsets pass message.
Bidirect message passing

- Bi-direct message passing scheme
- Messages can pass in parallel
- Cache hit improvement
Implementation Method

- Computational Optimization
- Memory Optimization
Computational Optimization

- Initialize Optimization
  - Initiate data in two layers (Coarse and Fine)
  - Compact initial data with Scale Factor
  - Parallel initiate data

- Message Passing Parallelizing
  - Parallel message passing in two direction (row and column)
  - Utilizing CPU cores (Bidirect message passing)
    - Compute each thread one row (Left & right) or column (up & down)
Memory Optimization

- System Level Optimization
  - Linearizing initial data at memory
  - Vector processing

- Bidirect Memory Optimization
  - Improve cache hit rate (data already exist in cache)
  - Message passing across a direction prefetches data for the other
Experimental Results
Experimental Results

Accuracy

Contest reference image
The reference code’s output (17743 mismatches)
Our method’s output (11367 mismatches)

Contest Barrels image
The reference code’s output (11870 mismatches)
Our method’s output (11833 mismatches)
Experimental Results

Time Results

Our accuracy-satisfying configuration

<table>
<thead>
<tr>
<th>Platforms</th>
<th>BP our opt.</th>
<th>naive BP–M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core i5–460M</td>
<td>48X</td>
<td>122X</td>
</tr>
<tr>
<td>Core i7–960</td>
<td>53X</td>
<td>230X</td>
</tr>
<tr>
<td>Xeon X5650</td>
<td>82X</td>
<td>384X</td>
</tr>
</tbody>
</table>
Experimental Results

Time Results I: Xeon X5650

Our implementation results in detail:

<table>
<thead>
<tr>
<th>Implementation Methods</th>
<th>Iter.</th>
<th>Scale Factor</th>
<th>tsukuba Image</th>
<th>Barrels Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mismatch</td>
<td>Time Req.</td>
</tr>
<tr>
<td>Reference Code</td>
<td>40</td>
<td>-</td>
<td>17743</td>
<td>16.7 s</td>
</tr>
<tr>
<td>Our - Config. 4-8</td>
<td>4</td>
<td>8</td>
<td>13695</td>
<td>0.6 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mismatch</td>
<td>Time Req.</td>
</tr>
<tr>
<td>Our - Config. 4-4</td>
<td>4</td>
<td>4</td>
<td>15047</td>
<td>1.1 ms</td>
</tr>
<tr>
<td>Our - Config. 40-4</td>
<td>40</td>
<td>4</td>
<td>11367</td>
<td>6.3 ms</td>
</tr>
<tr>
<td>Our - Config. 40-1</td>
<td>40</td>
<td>1</td>
<td>12727</td>
<td>204 ms</td>
</tr>
<tr>
<td>Our - Config. 14-1</td>
<td>14</td>
<td>1</td>
<td>17487</td>
<td>73 ms</td>
</tr>
<tr>
<td>Our - Config. 20-1</td>
<td>20</td>
<td>1</td>
<td>16333</td>
<td>103 ms</td>
</tr>
<tr>
<td>Our - Config. 30-1</td>
<td>30</td>
<td>1</td>
<td>14073</td>
<td>155 ms</td>
</tr>
</tbody>
</table>

The effect of itter and scale factor are obvious

For generating appropriate accuracy and runtime
### Experimental Results

#### Time Results II: Core i7

Our implementation results in detail:

<table>
<thead>
<tr>
<th>Implementation Methods</th>
<th>Iter.</th>
<th>Scale Factor</th>
<th>tsukuba Image</th>
<th>Barrels Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mismatch</td>
<td>Time Req.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mismatch</td>
<td>Time Req.</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core i7 960 @ 2.6 GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Code</td>
<td>40</td>
<td>-</td>
<td>17743</td>
<td>14.7 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11870</td>
<td>14.7 s</td>
</tr>
<tr>
<td><strong>Our – Config. 4-8</strong></td>
<td>4</td>
<td>8</td>
<td>13695</td>
<td>0.8 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17084</td>
<td>0.8 ms</td>
</tr>
<tr>
<td>Our – Config. 4-4</td>
<td>4</td>
<td>4</td>
<td>15047</td>
<td>1.5 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15348</td>
<td>1.5 ms</td>
</tr>
<tr>
<td>Our – Config. 40-4</td>
<td>40</td>
<td>4</td>
<td>11367</td>
<td>8.1 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15328</td>
<td>8.1 ms</td>
</tr>
<tr>
<td><strong>Our – Config. 40-1</strong></td>
<td>40</td>
<td>1</td>
<td>12727</td>
<td>277 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11833</td>
<td>277 ms</td>
</tr>
<tr>
<td>Our – Config. 14-1</td>
<td>14</td>
<td>1</td>
<td>17487</td>
<td>98 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13318</td>
<td>98 ms</td>
</tr>
<tr>
<td>Our – Config. 20-1</td>
<td>20</td>
<td>1</td>
<td>16333</td>
<td>139 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12570</td>
<td>139 ms</td>
</tr>
<tr>
<td>Our – Config. 30-1</td>
<td>30</td>
<td>1</td>
<td>14073</td>
<td>206 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12048</td>
<td>206 ms</td>
</tr>
</tbody>
</table>
Experimental Results

Time Results III : Core i5

Our implementation results in detail:

<table>
<thead>
<tr>
<th>Implementation Methods</th>
<th>Iter.</th>
<th>Scale Factor</th>
<th>tsukuba Image</th>
<th>Barrels Image</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mismatch</td>
<td>Time Req.</td>
</tr>
<tr>
<td>Intel(R) Xeon(R) CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core i5 @ 2.57GHz [5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Code</td>
<td>40</td>
<td>-</td>
<td>17743</td>
<td>40.5 s</td>
</tr>
<tr>
<td><strong>Our – Config. 4-8</strong></td>
<td>4</td>
<td>8</td>
<td>13695</td>
<td>2.9 ms</td>
</tr>
<tr>
<td>Our – Config. 4-4</td>
<td>4</td>
<td>4</td>
<td>15047</td>
<td>5.9 ms</td>
</tr>
<tr>
<td>Our – Config. 40-4</td>
<td>40</td>
<td>4</td>
<td>11367</td>
<td>40 ms</td>
</tr>
<tr>
<td><strong>Our – Config. 40-1</strong></td>
<td>40</td>
<td>1</td>
<td>12727</td>
<td>850 ms</td>
</tr>
<tr>
<td>Our – Config. 14-1</td>
<td>14</td>
<td>1</td>
<td>17487</td>
<td>300 ms</td>
</tr>
<tr>
<td>Our – Config. 20-1</td>
<td>20</td>
<td>1</td>
<td>16333</td>
<td>426 ms</td>
</tr>
<tr>
<td>Our – Config. 30-1</td>
<td>30</td>
<td>1</td>
<td>14073</td>
<td>635.3 ms</td>
</tr>
</tbody>
</table>
Acknowledgments

- Prof. Hamid Sarbazi Azad
- Ahmad Lashgar
- Mohsen Mahmoudi-Aznaveh
- Alireza Majidi
- Soroush Bateni
- Kamyar Mirzazad
- Mehrshad Vosoughi

http://cs.ipm.ac.ir
Any Question
References


