Plan

Context

State of the art

Denoising chain

Optimizations

Denoising efficiency

Time and energy consumption

Conclusion
Context

- Noise Appearance in bad conditions: Low light, Low contrast
- Physic limitation: Modern sensors quasi-perfect with very low read noise levels $\Rightarrow$ Noise $=$ Photon noise.
- Bigger optics:
  - Bigger systems.
  - Heavier
  - More expensive
- Need to propose a new solution: Overcome physics limitation with software.
- Proposed solution: Embedded real-time Spatial-temporal filter (25 fps).
State of the art

- Only a few articles on video denoising.
- A lot more on image denoising.
- Good displacements estimation needed for robust video denoising [1].
  ⇒ Kalman + bilateral method proposed.
  ⇒ Movement estimation using block matching.
  ⇒ No timing indications for this solution.

- [Zuo 2016] Existing algorithms very heavy [2].

Recent reconsideration for this problem

- Visually more efficient new methods:
  - VNLnet 2018 [5].
  - TOF denoising 2017 [6].
  - UNet 2018 [7].

  ⇒ Issue: Compute time too much important (Patchs + CNN).

- Real-time methods
  - STMKF 2017 [8].
  - Google 2018 [9].

  ⇒ Issue: Only for low noise level situations (like video compression).
Figure: Main denoising steps.

- Stabilization: Lucas Kanade global approach.
- Dense optical flow: TV-L1.
- Spatial filtering: Separated bilateral filter.
- Temporal filtering: Lateral filter.
Optimizations

Algorithmic Optimizations

- Various transformations applied for all steps:
  - SIMDization (Inefficient vectorization → Code hand-written in SIMD Neon).
  - Multi-task parallelism with OpenMP.
  - Operators fusion → Reduce the number of memory access.
  - Operators pipeline → Enhanced memory locality.
  - Cache blocking → Reduce memory footprint and enhanced memory locality.

- Other transformations more specific to each algorithm.
  - Lucas-Kanade stabilization.
    - Convolution computation using Integral images (summed area table).
    - Parallel computation of the convolution using partial integral images.
  - TV-L1 dense optical flow estimation.
    - Iterations pipeline (Dasip 2018) [10].
    - Fixed number of iterations chosen by studying the convergence speed.
    - Unbalanced distribution of the number of iterations between scales (3-20-80).
  - Spatial-temporal trilateral filter.
Optimizations
Impact of the optimizations (1/2)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Slow 1C</th>
<th>Fast 1C</th>
<th>Fast 8C</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stab (LK)</td>
<td>6.66</td>
<td>1.59</td>
<td>0.37</td>
<td>×18</td>
</tr>
<tr>
<td>Flow (TVL1)</td>
<td>260.73</td>
<td>107.93</td>
<td>27.59</td>
<td>×10</td>
</tr>
<tr>
<td>Filter</td>
<td>840.08</td>
<td>1.39</td>
<td>0.25</td>
<td>×3 360</td>
</tr>
<tr>
<td>Total</td>
<td>1107.47</td>
<td>110.90</td>
<td>28.21</td>
<td>×39</td>
</tr>
</tbody>
</table>

Table: Execution time (ms) and speedup for RTE-VD on AGX CPU.

- **Slow 1C**: Naive mono core implementation (vectorization on).
- **Fast 1C**: Fast mono core implementation.
- **Fast 8C**: Fast 8 cores parallel implementation.
- Mono core gain: \( \times 10 \).
- Total gain: \( \approx \times 40 \).
- Major gain on the filtering due to its approximation with separated filter.
- **Optical flow** = critical step: 98% total computation time.
Denoising efficiency (1/2)

- Evaluation on a well known database: *Derf’s Test Media Collection*.
- Comparison with other state of the art algorithms:
  - STMKF: State of the art for real-time methods.
  - VBM3D & VBM4D: State of the art for denoising efficiency (slow).
  - RTE-VD: This work.

<table>
<thead>
<tr>
<th>Noise</th>
<th>Method</th>
<th>crowd</th>
<th>park joy</th>
<th>pedestrians</th>
<th>station</th>
<th>sunflower</th>
<th>touchdown</th>
<th>tractor</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RTE-VD</td>
<td><strong>26.38</strong></td>
<td><strong>25.65</strong></td>
<td><strong>30.58</strong></td>
<td><strong>30.98</strong></td>
<td><strong>32.51</strong></td>
<td><strong>30.17</strong></td>
<td><strong>29.38</strong></td>
<td><strong>28.73</strong></td>
</tr>
<tr>
<td></td>
<td>VBM3D</td>
<td>28.75</td>
<td>27.89</td>
<td>35.49</td>
<td>34.19</td>
<td>35.48</td>
<td>32.85</td>
<td>31.44</td>
<td>31.34</td>
</tr>
<tr>
<td></td>
<td>VBM4D</td>
<td>28.43</td>
<td>27.11</td>
<td>35.91</td>
<td>35.00</td>
<td>35.97</td>
<td>32.73</td>
<td>31.65</td>
<td>31.11</td>
</tr>
<tr>
<td></td>
<td><strong>STMKF</strong></td>
<td>20.80</td>
<td>20.75</td>
<td>20.70</td>
<td>20.41</td>
<td>20.70</td>
<td>20.86</td>
<td>19.80</td>
<td>20.56</td>
</tr>
<tr>
<td></td>
<td>RTE-VD</td>
<td><strong>22.55</strong></td>
<td><strong>21.64</strong></td>
<td><strong>25.72</strong></td>
<td><strong>27.76</strong></td>
<td><strong>27.87</strong></td>
<td><strong>27.05</strong></td>
<td><strong>25.99</strong></td>
<td><strong>24.85</strong></td>
</tr>
<tr>
<td></td>
<td>VBM3D</td>
<td>24.81</td>
<td>23.78</td>
<td>30.65</td>
<td>30.62</td>
<td>30.88</td>
<td>30.21</td>
<td>27.82</td>
<td>27.43</td>
</tr>
<tr>
<td></td>
<td>VBM4D</td>
<td>24.65</td>
<td>23.22</td>
<td>31.32</td>
<td>31.53</td>
<td>31.39</td>
<td>30.09</td>
<td>28.09</td>
<td>27.35</td>
</tr>
</tbody>
</table>

Table: PSNR comparison on 7 *Derf’s Test Media Collection*’s sequences with other state of the art algorithms.

- Until +7dB over STMKF. Average: +4dB ($\sigma = 40$).
- Maximum -6dB under VBM3D/4D. Average: -2.5dB ($\sigma = 40$).
Denoising efficiency (2/2)

- Stronger denoising than STMKF.
- Less efficient denoising than VBM3D/4D.
- Weakness on details rendering for background.

Figure: Visual comparison on the pedestrian scene (Gaussian noise: $\sigma = 40$).
Time and energy consumption (1/3)

Execution time comparison with other state of the art algorithms

**Implementation on various platforms:**

<table>
<thead>
<tr>
<th>Board</th>
<th>Process</th>
<th>CPU</th>
<th>Fmax (GHz)</th>
<th>Idle Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX2</td>
<td>16 nm</td>
<td>4×A57 + 2×Denver2</td>
<td>2.00</td>
<td>2.0</td>
</tr>
<tr>
<td>AGX</td>
<td>12 nm</td>
<td>8×Carmel</td>
<td>2.27</td>
<td>6.3</td>
</tr>
<tr>
<td>NANO</td>
<td>12 nm</td>
<td>4×A57</td>
<td>1.43</td>
<td>1.2</td>
</tr>
<tr>
<td>XEON</td>
<td>14 nm</td>
<td>2×10C/20T Skylake</td>
<td>2.20</td>
<td>–</td>
</tr>
</tbody>
</table>

*Table: Technical specification of tested platforms.*

**Method comparison:**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (s)</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>STMKF</td>
<td>0.0045</td>
<td>Xeon</td>
</tr>
<tr>
<td><strong>RTE-VD (this work)</strong></td>
<td>0.0097</td>
<td>Xeon</td>
</tr>
<tr>
<td>VBM3D</td>
<td>2.0</td>
<td>Xeon</td>
</tr>
<tr>
<td>VBM4D</td>
<td>45</td>
<td>Xeon</td>
</tr>
<tr>
<td>STMKF</td>
<td>0.015</td>
<td>AGX</td>
</tr>
<tr>
<td><strong>RTE-VD (this work)</strong></td>
<td>0.037</td>
<td>AGX</td>
</tr>
</tbody>
</table>

*Table: Time per qHD images (960×540 pixels).*

- 200× faster than VBM3D.
- 4600× faster than VBM4D.
- 2.5× slower than STMKF.
- Embedded real-time in qHD.
Time and energy consumption (2/3)

Time vs energy efficiency: Dynamic consumption

Figure: Time/energy efficiency of RTE-VD on CPU for various frequencies ($E_{\text{dynamic}}$).
Time and energy consumption (3/3)

Time vs Energy efficiency: Dynamic + static consumption

Figure: Time/energy efficiency of RTE-VD on CPU for various frequencies ($E_{total}$).
VIRTANS : Denoiser Prototype

Figure: VIRTANS : Video Real-Time Algorithm : Noise Suppression.

- TX2i based Architecture.
- Real-time 480 × 270 pixels video denoising.
- SDI input / HDMI Output.
- Ethernet communication with LHERITIER Cameras.
Conclusion

▶ Introduction to a new Real-Time Embedded Video Denoising algorithm
  ▶ Slower than STMKF but denoising a lot more efficient
  ▶ Embedded real-time performances for qHD videos (960×540 pixels)
▶ Energy consumption study: positioning depending on the targeted system
▶ RTE-VD based real-time video denoiser : VIRTANS
⇒ RTE-VD is well positioned for speed/accuracy tradeoff

Future works

▶ GPU implementation and hybrid CPU/GPU computation
▶ 32 - 16 bits hybrid computation
▶ Reduce VIRTANS form factor even more.
Thank you!
References I


Optimizations

Impact of the optimizations (2/2)

Figure: Impact of TV-L1 optimizations on speed and energy on TK1’s CPU depending of the image size. Lower is better.

<table>
<thead>
<tr>
<th>Version</th>
<th>Base</th>
<th>II</th>
<th>Pipe</th>
<th>MA</th>
<th>SIMD</th>
<th>OMP8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (ms)</td>
<td>1742</td>
<td>56,0</td>
<td>33,5</td>
<td>29,0</td>
<td>9,2</td>
<td>1,6</td>
<td>–</td>
</tr>
<tr>
<td>Speedup</td>
<td>×1</td>
<td>×31</td>
<td>×1,75</td>
<td>×1,2</td>
<td>×3,1</td>
<td>×5,9</td>
<td>×1120</td>
</tr>
</tbody>
</table>

Table: Impact of LK Stabilization optimizations for fullHD images on AGX’s CPU.
Backup: Time and energy consumption

Time vs energy efficiency

Table: Best configurations for 25 fps

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Energy (nJ/pix)</th>
<th>Time (ns/pix)</th>
<th>Max size (#pix)</th>
<th>Freq (GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NANO min energy</td>
<td>616</td>
<td>311</td>
<td>358</td>
<td>1.4</td>
</tr>
<tr>
<td>NANO min time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX2 min energy</td>
<td>1046</td>
<td>242</td>
<td>406</td>
<td>1.2</td>
</tr>
<tr>
<td>TX2 min time</td>
<td>1209</td>
<td>165</td>
<td>492</td>
<td>2.0</td>
</tr>
<tr>
<td>AGX min energy</td>
<td>683</td>
<td>114</td>
<td>592</td>
<td>1.4</td>
</tr>
<tr>
<td>AGX min time</td>
<td>832</td>
<td>70</td>
<td>754</td>
<td>2.3</td>
</tr>
</tbody>
</table>

- NANO: the most energy efficient
- AGX: the fastest and the most energy efficient
- TX2: Penalized by its process but faster than NANO
- Greatest image at 25 fps: 754 × 754 pixels