FastANN for High Quality Collaborative Filtering

Yun-Ta Tsai\(^1\), Markus Steinberger\(^2\), Dawid Pająk\(^3\), Kari Pulli\(^4\)

\(^1\)Google, Inc. \quad \(^2\)TU Graz \quad \(^3\)NVIDIA Research \quad \(^4\)Light
Collaborative Filtering - Aggregation
Related Work
Distance Table (sorted)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.13</td>
<td>0.33</td>
<td>0.28</td>
<td>1.33</td>
<td>1.21</td>
<td>1.43</td>
<td>2.33</td>
</tr>
<tr>
<td>0.21</td>
<td>0.45</td>
<td>0.48</td>
<td>2.02</td>
<td>1.75</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>0.21</td>
<td>0.32</td>
<td>0.92</td>
<td>3.18</td>
<td>0.33</td>
<td>0.16</td>
<td></td>
</tr>
</tbody>
</table>

Garcia et al. ICIP 2010
Limitation
Our solution
Our solution

- Efficient implementation on GPU
- General solution for different filters
- High image quality
- Applicable to different applications
Our solution
Our solution

- Clustering
- kNN Query
- Filtering
Design challenges

- Register pressure
- Memory access pattern
- Thread divergence
- Kernel launch overhead
- Memory footprint
Our solution

- Clustering
- kNN Query
- Filtering
Clustering
Warp-wide operation

3 8 2 6 3 9 1 4
Warp-wide operation
Warp-wide operation

3  11  10  8  9  12  10  5
Warp-wide operation

3 11 10 8 9 12 10 5

3 11 10 8 9 12
Warp-wide operation

3 11 13 19 19 20 19 17
Warp-wide operation
Warp-wide operation

- Reduce register usage
- Better parallelism
- Minimize thread and warp divergence
Clustering
Our solution

Clustering

kNN Query

Filtering
Our solution

Clustering

kNN Query

Filtering
kNN Query
kNN Query

\[ \{ p^0, p^1, p^2, p^3, p^4 \} \]
kNN Query
kNN Query

kNN

\{ \)

\{ \)

p^0 \hspace{0.1cm} p^1 \hspace{0.1cm} p^2 \hspace{0.1cm} p^3 \hspace{0.1cm} p^4

0.0 \hspace{0.1cm} 0.4 \hspace{0.1cm} 1.3 \hspace{0.1cm} 0.9 \hspace{0.1cm} 0.7

p^0 \hspace{0.1cm} p^1 \hspace{0.1cm} p^2 \hspace{0.1cm} p^3 \hspace{0.1cm} p^4
kNN Query

0.0  0.4  1.3  0.9  0.7

\[ p^0 \leq 0.8 \]
kNN Query

\[ p^0 \leq 0.0 \]

\[ \begin{array}{c|cccc}
   & p^0 & p^1 & p^2 & p^3 & p^4 \\
   \hline
   p^0 & 0.0 & 0.4 & 1.3 & 0.9 & 0.7 \\
   1 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \leq 0.0 \]
kNN Query

\[
\begin{array}{ccccc}
\text{p}^0 & \text{p}^1 & \text{p}^2 & \text{p}^3 & \text{p}^4 \\
0.0 & 0.4 & 1.3 & 0.9 & 0.7 \\
1 & 0 & 0 & 0 & 0 \\
\leq 0.0 \\
1 & 1 & 0 & 0 & 0 \\
\leq 0.4 \\
\end{array}
\]
kNN Query

<table>
<thead>
<tr>
<th></th>
<th>p^0</th>
<th>p^1</th>
<th>p^2</th>
<th>p^3</th>
<th>p^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p^0</td>
<td>0.0</td>
<td>0.4</td>
<td>1.3</td>
<td>0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
kNN Query

<table>
<thead>
<tr>
<th>p0</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>1.3</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>

- 1 0 0 0 0 0 ≤ 0.0
- 1 1 0 0 0 0 ≤ 0.4
- 1 1 1 1 1 1 ≤ 1.3
- 1 1 0 1 1 1 ≤ 0.9
kNN Query

<table>
<thead>
<tr>
<th>p^0</th>
<th>p^1</th>
<th>p^2</th>
<th>p^3</th>
<th>p^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>1.3</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>

- \( p^0 \leq 0.0 \)
- \( 1 \mid 0 \mid 0 \mid 0 \mid 0 \leq 0.4 \)
- \( 1 \mid 1 \mid 0 \mid 0 \mid 0 \leq 0.4 \)
- \( 1 \mid 1 \mid 1 \mid 1 \mid 1 \leq 1.3 \)
- \( 1 \mid 1 \mid 0 \mid 1 \mid 1 \leq 0.9 \)
- \( 1 \mid 1 \mid 0 \mid 0 \mid 1 \leq 0.7 \)
## kNN Query

<table>
<thead>
<tr>
<th>p^0</th>
<th>p^1</th>
<th>p^2</th>
<th>p^3</th>
<th>p^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.4</td>
<td>1.3</td>
<td>0.9</td>
<td>0.7</td>
</tr>
</tbody>
</table>

- \[ 1 \ 0 \ 0 \ 0 \ 0 \ \leq 0.0 \]
- \[ 1 \ 1 \ 0 \ 0 \ 0 \ \leq 0.4 \]
- \[ 1 \ 1 \ 1 \ 1 \ 1 \ \leq 1.3 \]
- \[ 1 \ 1 \ 0 \ 1 \ 1 \ \leq 0.9 \]
- \[ 1 \ 1 \ 0 \ 0 \ 1 \ \leq 0.7 \]
kNN Query

\[
\{ kNN \}
\]

\[
p^0 \quad p^1 \quad p^2 \quad p^3 \quad p^4
\]

\[
\rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow
\]

\[
\circ \quad \circ \quad \circ \quad \circ \quad \circ
\]
Our solution

- Hierarchical 2-mean clustering
- Warp-wide operators
- kNN search
- Distance table
- Voting
- Binary coding
- Filtering and aggregation

Diagram: Clustering → kNN Query → Filtering
Results
NN quality

% of patches matching the kNN result, k=16
(the higher the better)

Randomized KD-trees: 24.87
K-means: 34.86
Composite: 35.21
Hierarchical Clustering: 7.18
Generalized PatchMatch: 0.22
Random Cover: 97.88
Ours: 39.01
NN quality

$D_{ann}/D_{knn}, \ k=16$

(the lower the better)

- Randomized KD-trees: 3.01
- K-means: 2.00
- Composite: 1.99
- Hierarchical Clustering: 1.99
- Generalized PatchMatch: 23.91
- Random Cover: 1.01
- Ours: 1.32
Single Frame Noise Reduction

Nonlocal Means - PSNR [dB], k=16
(the higher the better)

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomized KD-trees</td>
<td>26.88</td>
</tr>
<tr>
<td>K-means</td>
<td>27.13</td>
</tr>
<tr>
<td>Composite</td>
<td>27.02</td>
</tr>
<tr>
<td>Hierarchical Clustering</td>
<td>25.65</td>
</tr>
<tr>
<td>Generalized PatchMatch</td>
<td>21.24</td>
</tr>
<tr>
<td>Random Ball Cover</td>
<td>27.83</td>
</tr>
<tr>
<td>Ours</td>
<td>27.79</td>
</tr>
<tr>
<td>Exhaustive Search</td>
<td>28.55</td>
</tr>
</tbody>
</table>
Single Frame Noise Reduction

BM3D - PSNR [dB], k=16
(the higher the better)

- Randomized KD-trees: 30.72
- K-means: 30.68
- Composite: 30.57
- Hierarchical Clustering: 29.87
- Generalized PatchMatch: 28.92
- Random Ball Cover: 30.71
- Ours: 31.05
- Exhaustive Search: 31.10
Single Frame Noise Reduction

Run-time [ms], k=16, 0.25MPix
(the lower the better)

<table>
<thead>
<tr>
<th>Method</th>
<th>Query Time (ms)</th>
<th>Clustering Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomized KD-trees</td>
<td>788</td>
<td>0.25MPix</td>
</tr>
<tr>
<td>K-means</td>
<td>982</td>
<td></td>
</tr>
<tr>
<td>Composite</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td>Hierarchical Clustering</td>
<td>1017</td>
<td></td>
</tr>
<tr>
<td>Generalized PatchMatch</td>
<td>8930</td>
<td></td>
</tr>
<tr>
<td>Ours (GPU)</td>
<td>8.19</td>
<td></td>
</tr>
</tbody>
</table>
Single Frame Noise Reduction

Run-time [ms], k=16, 0.25MPix (the lower the better)

- kNN-Garcia (GPU): 26359 ms
- Random Ball Cover (GPU): 11328 ms
- Window Search (GPU): 594.99 ms
- Window Search Opt (GPU): 48.3 ms
- Ours (GPU): 8.19 ms

NVIDIA
Architectures

MPix/s/Watt, k=16, 0.25MPix
(the higher the better)
Burst Denoising - Single Frame

First frame of stack
26.45dB
Burst Denoising - Single Frame

First frame of stack
26.45dB
Burst Denoising - Single Frame

First frame of stack
26.45dB

First frame of stack
26.45dB

GKD-Trees / \textit{first frame}
31.01dB / 11.3s

Ours NLM / \textit{first frame}
31.90dB / 0.02s
Burst Denoising - All Frames

- First frame of stack: 26.45dB
- GKD-Trees / stack: 31.53dB / 1080s
- Ours NLM / stack: 34.10dB / 0.52s
Burst Denoising - All Frames

Ground truth

GKD-Trees / stack
31.53dB / 1080s

Ours NLM / stack
34.10dB / 0.52s
Global Illumination

4spp

18.99dB / 2.17s
Global Illumination

4spp

18.99dB / 2.17s

Ours

35.11dB / 2.20s
Global Illumination

Ours: 35.11dB / 2.20s
512spp: 35.63dB / 243s
Geometry Noise Reduction

Noisy Input
Geometry Noise Reduction

Ours
Geometry Noise Reduction

Ours

Exhaustive Search
Conclusions

- Efficient implementation on GPU
- High image quality
- Applicable to different applications
Thank you

Paper and Binary: