Parallel Search of $k$-Nearest Neighbors with Synchronous Operations

Nikos Sismanis$^1$  Nikos Pitsianis$^{1,2}$  Xiaobai Sun$^2$

$^1$Dept. ECE  Aristotle University, Greece

$^2$Dept. CS  Duke University, USA

September 11, 2012
Outline

1 Motivational Applications
2 Problem Statement
3 State-of-the-Art Solutions
4 Qualitative Performance Analysis
5 Quantitative Performance Analysis: Placing Landmarks
6 Multistage Streaming: Planning & Tuning
**KNN search**: Primitive and Prevalent Operation

Identification of most matching points from a large and high dimensional data space/corpus, according to a well defined distance measure.

More applications with increased data acquisition for:
- machine learning and modeling
- pattern matching and (speech, image) recognition
- filtering or localization in data analysis & mining

Facilitating various research areas: computer/machine vision, computer-human interaction, computational imaging, geometry, computational statistics.
KNN Search for Image Queries

1 D. G. Lowe, Inter. J. Comp. Vis., 2004
2 http://www.rocq.inria.fr/imedia/belga-logo.html
KNN Search for Image Queries

KNN search in SIFT feature space for image corpus & queries

- Preprocessed feature vectors for corpus images
- Extraction of feature vectors for query images/subimages
- High dimensional feature space (long feature vectors)
- Similarity score, correlation or distance function over the space
- KNN search to locate close matches for further classification

1 D. G. Lowe, Inter. J. Comp. Vis., 2004
2 http://wwwrocq.inria.frimedia/belga-logo.html
The computation of the nearest neighbor for the purpose of feature matching is the most time-consuming part of the complete recognition and localization algorithm.

P. Azad, IROS, 2009

Fast KNN search will expedite

- GIS-moving objects in road networks  C. Shahabi et al., SIGSPATIAL GIS, 2002
- Network intrusion detection  L. Kuang and M. Zulkernine, ACM SAC, 2008
- Text categorization  S. Manne et al., Inter. J. Comp. Appl., 2011
Outline

1. Motivational Applications
2. Problem Statement
3. State-of-the-Art Solutions
4. Qualitative Performance Analysis
5. Quantitative Performance Analysis: Placing Landmarks
6. Multistage Streaming: Planning & Tuning
The KNN Search Problem

Problem Statement

To each and every query, locate $k$ nearest neighbors, according to a score function, among $n$ corpus data points in a $d$-dim space.

$d$: the dimensionality of the search space such as the length of the SIFT feature vectors

$n$: the number of corpus data points to query from

$q$: the number of query points

$k$: the number of nearest neighbors to locate for each query
Outline

1. Motivational Applications
2. Problem Statement
3. State-of-the-Art Solutions
4. Qualitative Performance Analysis
5. Quantitative Performance Analysis: Placing Landmarks
6. Multistage Streaming: Planning & Tuning
State-of-the-Art Solutions

Typical solution components

▷ Search hierarchy for rapid elimination of far neighbors
  - Kd-trees $^3$, Balltrees $^4$, Metric trees $^5$
  - Total # of comparisons:
    linear in $k$ and sub-linear in global corpus size $N$, e.g., $O(\log N)$

▷ Exact KNN search in a corpus of reduced size $n$
  - linear in $k$ and $n$

▷ Approximate KNN search
  - Locality-sensitive hashing $^6$

---

$^3$ J. L. Bentley, Comm. ACM, 1975
$^6$ P. Indyk, 30-th ACM STOC, 1999
Sort-Select-KNN Triangle

- sort-based KNN
- select-based KNN
- mutual connection between sort and select
- devil in algorithm and architecture detail
Inner and Outer KNNs

- **Outer KNN**
  - massive corpus data
  - quick reduction or decomposition
  - exploit data sparsity or clusters with heap data structure and operations

- **Inner KNN**
  - reduced corpus
  - utilize hardware architectures
  - exploit multiple queries
  - exploit relationship between query and corpus data
  - fast multi-dimension array operations
State-of-the-Art Solutions

More to be desired

- Synchronization on SIMD/SIMT processors such as GPUs
- Response latency for a single query
- Throughput rate for multiple queries
- Autotuning of performance
- Benchmarking at different integration scopes
## KNN Search on GPUs: some other works

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Alg</th>
<th>Speedup</th>
<th>Parameter range</th>
<th>Parameter range</th>
<th>Parameter range</th>
<th>Parameter range</th>
<th>Parameter range</th>
<th>Parameter range</th>
</tr>
</thead>
<tbody>
<tr>
<td>(references)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kdd-cup</td>
<td>exact</td>
<td>50</td>
<td>CPU</td>
<td>262,144</td>
<td>65</td>
<td>7</td>
<td>12,000</td>
<td></td>
</tr>
<tr>
<td>uci adult</td>
<td>exact</td>
<td>15</td>
<td>ANN</td>
<td>30,956</td>
<td>123</td>
<td>16</td>
<td>1,605</td>
<td></td>
</tr>
<tr>
<td>inria holidays</td>
<td>exact</td>
<td>64</td>
<td>ANN</td>
<td>65,536</td>
<td>128</td>
<td>20</td>
<td>1,024</td>
<td></td>
</tr>
<tr>
<td>nasa images</td>
<td>exact</td>
<td>2</td>
<td>Sort</td>
<td>120,000</td>
<td>254</td>
<td>32</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>recom system</td>
<td>exact</td>
<td>160</td>
<td>CPU</td>
<td>80,000</td>
<td>256</td>
<td>100</td>
<td>any</td>
<td></td>
</tr>
<tr>
<td>labelme</td>
<td>aprox.</td>
<td>40</td>
<td>lshkit</td>
<td>100,000</td>
<td>512</td>
<td>500</td>
<td>any</td>
<td></td>
</tr>
</tbody>
</table>

---

7 S. Liang et al., IEEE Symp. Web. Soc., 2010  
8 Q. Kuang and L. Zhao, ISCSCT, 2009  
9 V. Garcia et al., ICIP, 2010  
10 R. J. Barientos et al., Euro-Par, 2011  
11 K. Kato and T. Hosino, CCGRID, 2010  
12 http://www.labelme.csail.mit.edu  
13 J. Pan and D. Manocha, GIS, 2011
Performance Analysis: Qualitative Factors

I. Architecture independent
   ▶ complexity in comparisons
   ▶ longest dependency path/depth
   ▶ variation in concurrency breadth

II. Architecture dependent
   ▶ effective concurrency breadth and dependency depth
   ▶ data locality: computation-communication ratio
   ▶ synchronization cost on GPUs

How well do we know the architectural impact quantitatively?
Outline

1. Motivational Applications
2. Problem Statement
3. State-of-the-Art Solutions
4. Qualitative Performance Analysis
5. Quantitative Performance Analysis: Placing Landmarks
6. Multistage Streaming: Planning & Tuning
Performance Assessment : Quantitative References

Explore the two-ways relationship between SORT and SELECT

○ SORT $\rightarrow$ SELECT
  - select or truncate *after* a complete ascending sort
  - **truncated sort**:
    truncate as early as possible *during* an ascending sort process

  *as reference landmarks for quantitative performance assessment, or even as competitive candidates*

○ SELECT $\leftarrow$ SORT

(omitted from this talk)
## Truncated Sort Algorithms: Brief Summary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Serial</th>
<th>Parallel (length)</th>
<th>Truncation Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>BubbleSort</td>
<td>$nk$</td>
<td>$k(\log n - \log k + 1)$</td>
<td>$k$ reversal passes</td>
</tr>
<tr>
<td>InsertionSort</td>
<td>$nk$</td>
<td>$k(\log n - \log k + 1)$</td>
<td>length-$k$ array</td>
</tr>
<tr>
<td>HeapSort</td>
<td>$n \log k$</td>
<td>$k(\log n - \log k + 1)$</td>
<td>max-heap of size $k$</td>
</tr>
<tr>
<td>MergeSort</td>
<td>$n \log k$</td>
<td>$k(\log n - \log k + 1)$</td>
<td>elimination by “half”</td>
</tr>
<tr>
<td>QuickSort</td>
<td>$nk$</td>
<td>$k(\log n - \log k + 1)$</td>
<td>elimination by “half”</td>
</tr>
<tr>
<td>RadixSort</td>
<td>$n \log r \ c$</td>
<td>$\log r \ c$</td>
<td>reverse radix (MSB)</td>
</tr>
<tr>
<td>BitonicSort</td>
<td>$n \log^2 k$</td>
<td>$\log k \log n$</td>
<td>length-$k$ bitonic</td>
</tr>
</tbody>
</table>

$1 \leq k \leq n$

---

15 D. E. Knuth, The Art of Comp. Prog. 3, Addison-Wesley, 1973
16 D. M. W. Powers, PACT, 1991
17 K. E. Batcher, AFIPS, 1968
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  - free of hashing or branching
- high data locality
  - within practical range of k
- regular structures
  - data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  - free of hashing or branching
- high data locality
  - within practical range of k
- regular structures
  - data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors

Sismanis, Pitsianis & Sun (AUTh & Duke)  Parallel Search of kNN with Synch Ops  September 11, 2012  19 / 30
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  
  free of hashing or branching
- high data locality
  
  within practical range of $k$
- regular structures
  
  data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  
  free of hashing or branching
- high data locality
  
  within practical range of \( k \)
- regular structures
  
  data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  - free of hashing or branching
- high data locality
  - within practical range of $k$
- regular structures
  - data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors

Sismanis, Pitsianis & Sun (AUTH & Duke) Parallel Search of $k$NN with Synch Ops September 11, 2012
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
- free of hashing or branching
- high data locality
- within practical range of $k$
- regular structures
- data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  \textit{free of hashing or branching}
- high data locality
  \textit{within practical range of } k
- regular structures
  \textit{data access, program}

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  - free of hashing or branching
- high data locality
  - within practical range of $k$
- regular structures
  - data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
- free of hashing or branching
- high data locality
- within practical range of \( k \)
- regular structures
- data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  free of hashing or branching
- high data locality
  within practical range of $k$
- regular structures
  data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  *free of hashing or branching*
- high data locality
  *within practical range of k*
- regular structures
  *data access, program*

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  - free of hashing or branching
- high data locality
  - within practical range of $k$
- regular structures
  - data access, program

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
Quantitative Landmark: Truncated Bitonic Sort

- higher # pairwise comparisons
- inherently synchronous
  *free of hashing or branching*
- high data locality
  *within practical range of k*
- regular structures
  *data access, program*

A remarkable quantitative reference for KNN search performance on SIMD/SIMT processors
THRUST::SORT vs Truncated Bitonic Sort

Inclusion of Score Evaluation

Exclusion of Score Evaluation
Truncated Sorting Interleaved with Scoring

Sismanis, Pitsianis & Sun (AUTH & Duke)  Parallel Search of $k$NN with Synch Ops  September 11, 2012  21 / 30
Comparison of Truncated Bitonic and Radix Select over thrust::sort

Here, thrust::sort used as a common base for comparison

```plaintext
Manifest of Synch. Cost
Truncated Bitonic Sort substantially outperforms MGPU Radix Select over the effective range
```
Outline

1 Motivational Applications
2 Problem Statement
3 State-of-the-Art Solutions
4 Qualitative Performance Analysis
5 Quantitative Performance Analysis: Placing Landmarks
6 Multistage Streaming: Planning & Tuning
KNN Search in Multistage Streaming on GPUs

- transporting and buffering large corpus data in batches (batch size $n$)
- merging KNNs between the previous and the current corpus batches
- inclusion of score evaluation and pre/post computation tasks (separated or interleaved)
- multiple queries (as desirable in certain applications)
Profile in total execution time

- Left bars: Truncate after sorting using `thrust::sort` in percentile.
  Data transfer dominant when the batch size $n$ is large.

- Right bars: Truncated Bitonic normalized against the left bars.
KNN Search Profile on GPUs: Multiple Queries

- Left bars: Truncate after sorting using `thrust::sort`
- Right bars: Truncated Bitonic normalized against the left bars
KNN Search in Multistage Streaming on GPUs

- 16,777,216 vectors of 128 dimensions
- Overlapping data transfer and computations
- Up to $\times 1.75$ speed-up from using only 1 GPU stream
SIFT Feature Matching:

- **VLFeat, a CV Library**
  - Sequential implementation of feature extraction (with SIFT) and KNN search
  - Approximate k-NN using tree space partition
- **Speed-up over VLFeat**
  - 60X with 128 queries
  - 180 ∼ 250X with 512 queries

---

- **Parallel Search of kNN with Synch Ops**

---

\( a \) http://www.vlfeat.org

\( b \) Parallel SIFT vector extraction available on GPUs: [http://www.cs.unc.edu/~ccwu/siftgpu/](http://www.cs.unc.edu/~ccwu/siftgpu/)
Summary

We have

▷ addressed response latency & throughput issues

▷ explored the SORT-SELECT relationship

▷ exposed the synchronization cost on GPUs & provided references for quantitative performance assessment (relevant for approximate KNN search as well)

▷ suggested options and opportunities to better exploit GPUs for rapid KNN search queries

▷ codes and test data available at http://autogpu.ee.auth.gr
Acknowledgments

NVIDIA academic research equipment support
Marie Curie International Reintegration Program, EU
National Science Foundation (CCF), USA