GPU-BASED KERNELIZED LOCALITY-SENSITIVE HASHING FOR SATELLITE IMAGE RETRIEVAL

Niko Lukač, Borut Žalik

Faculty of Electrical Engineering and
Computer Science (FERI)
University of Maribor (UM)
Smetanova ulica 17, SI-2000, Maribor

Shiyong Cui, Mihai Datcu

German Aerospace Center (DLR)
Remote Sensing Technology Institute (IMF)
Münchener Straße 20, 82234, Wessling

Abstract

As the data acquisition capabilities of Earth observation (EO) satellites have been improved substantially in the past few years, large amount of high-resolution satellite images are downlinked continuously to ground stations. Such amount of data increases rapidly beyond the users' capability to access the images' content in reasonable time. Hence, automatic and fast interpretation of a large data volume is a computationally intensive task. Recently, approximate nearest neigbhour search has been used for content-based image retrieval in sublinear time. Kernelized locality sensitive hashing (KLSH) is a well-known approximate method, which has recently shown promising results for fast remote sensing image retrieval. This paper proposes a novel parallelization of KLSH using Graphical Processing Units (GPU), in order to perform fast parallel image retrieval. The proposed method was tested on high-dimensional feature vectors from two satellite-based image datasets, where an average speedup of 20 times was achieved.

1. INTRODUCTION

Given a large database of satellite images for real applications, developing methods and systems for exploring large-scale databases is of significant importance [1]. Thus, content-based image retrieval (CBIR) has been developed since years to solve this problem. In the past few years, CBIR systems that allow discovering hidden patterns and indexing of high-dimensional space, have been continuously developed, such as the Knowledge-driven Information Mining (KIM) system [2] and the Geospatial Information Retrieval and Indexing (GeoIRIS) system [3]. KIM is an interactive system based on human centered concepts, which allows the user to guide the interactive learning process. The system continuously provides the relevance feedback about the performed training actions and searches the archive for relevant images. On the contrary, GeoIRIS is a content-based multimodal geospatial information retrieval and indexing system, which allows automatic feature extraction, visual content mining from large-scale image databases, and high-dimensional database indexing for fast retrieval.

In theory, image retrieval is a trivial problem solved by the nearest neighbor search (NNS). Given a database and a query, the naive solution to finding similar images is to probe all images in the database and compute the similarity of the query to each image. Then all similar images (i.e. their feature vectors) can be found by sorting the similarity values of all images to the query. Approximate similarity search has received a lot of attention for solving this problem in sufficient time, where one prominent solution is the locality sensitivity hashing (LSH) [4]. The basic idea of LSH is that if two feature vectors are similar in the high-dimensional feature space, there is a high probability that they are also close in the low-dimensional hashing (e.g. Hamming) space. It consists of two steps. The first step is to build the data structure (hash tables), while in the second step is to find the candidate (i.e. approximately close) feature vectors that are saved in the same bucket as the query image. Following the same idea as LSH, kernelized LSH (KLSH) [5] performs indexing in a kernelized feature space, where the underlying data embedding is implicitly known, thus all the requirement is a kernel function.

Although hashing generally guarantees sublinear NNS query time, in practice it can be even further speedup when considering multiple queries over a large image database. Therefore, in the past few years many paralelizations of LSH have been proposed [6, 7]. Although domain specific methods [2, 3] have been developed for fast content retrieval over remote sensing images, until recently LSH was not considered for this purpose. Furthermore, supervised KLSH has recently shown promising results on the remote sensing image retrieval [8]. Therefore, in this paper a novel parallelizaton of KLSH is proposed using general-purpose computing on GPU (GPGPU), which can perform multiple queries at once over large-scale remote sensing image datasets.

The paper is structured into 4 sections, where the proposed method is described in detail within the next section. The results are presented in the third section, while the last section concludes the paper.

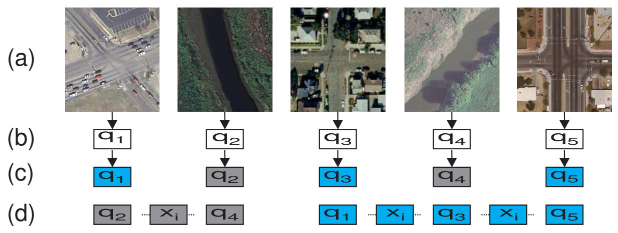

Fig. 1. (a) Example query images (subset of Q); (b) Transformation to feature space; (c) Hashing into different buckets (denoted with background color); (d) Sorting the buckets with already hashed X data.

2. KERNELIZED LOCALITY SENSITIVE HASHING

LSH is an efficient method for performing approximate NNS for a given data input (i.e. high-dimensional point). The idea is to project D-dimensional data $\mathbf{x_1}, ..., \mathbf{x_n} \in X$ into low-dimensional space U by using hashing function $h : \mathrm{R}^{\mathrm{D}} \rightarrow \mathrm{U}$, which satisfies the locality sensitive property:

$$Pr_h[h(x_i) = h(x_j)] = sim(x_i, x_j) \quad (1)$$

where $sim(x_i, x_j) \in [0, 1]$ is the considered similarity function. One of the more interesting approaches is the use of an inner product with a random vector $r \in \mathcal{N}(0, 1)$. It is important that r is created from the zero-mean multivariate Gaussian distribution, in order to establish locality sensitive property. The probability of having an equal sign when calculating the inner product between r and two different points is inversely proportional to cosine similarity between the two given points [4, 5]:

$$\begin{aligned} Pr_h[sign(x_i^T r) = sign(x_j^T r)] = \\ = 1 - \frac{1}{\pi} cos^{-1} \left(\frac{x_i^T x_j}{||x_i||_2 ||x_j||_2} \right). \end{aligned} \quad (2)$$

Hence, the hashing function is defined as [5]:

$$h(x_i) = sign(x_i^T r). \quad (3)$$

The next step is to construct a hash-table by hashing X, where the bucket index is calculated by concatenating the output of m hash functions $H(x_i) = \{h_0(x_i) h_1(x_i) ... h_m(x_i)\}$, where each has its own vector r. The most basic NNS query on x_i is to calculate exact distances to candidates located within the same bucket $H(x_i)$, while more sophisticated methodology was developed by using multi-probe LSH (MLSH) [9], where also the neighbouring buckets within the hash-tables are considered (i.e. probed). This is based on the probability that x_i could have neighbours within a δ-neighbouring bucket from current bucket, i.e. $H(x_i) + \delta \in \{-2, -1, +1, +2\}$ [9]:

$$Pr[H(x_i) = H(x_i) + \delta] \approx e^{-Cf(x_i, \delta)^2}, \quad (4)$$

based on the probability density function of Gaussian random variable (i.e. $e^{-x^2/2\sigma^2}$). $f(x_i, \delta)$ is the distance to the δ-neighbouring bucket, and C is a tuning constant. The main advantage of MLSH is the considerably reduced memory requirements due to lower m.

KLSH on the other hand considers kernel-based similarity function [5]:

$$sim(x_i, x_j) = \kappa(x_i, x_j) = \theta(x_i)^T \theta(x_j), \quad (5)$$

where $\theta(x_i)$ maps a given point into a low-dimensional feature space. KLSH proposes an efficient way to calculate $r \in \mathcal{N}(0, 1)$, where $r^T \theta(x_i)$ can be computed with a kernel function. They have shown that this can be achieved by defining r as [5]:

$$r = \sum_{j=1}^{|P|} w_i \theta(x_j); x_j \in P, \quad (6)$$

where w is a weighting vector and $P \in X$ is a relatively small subset that is used to form the zero-centered kernel matrix K. Hence, in order to satisfy the locality sensitive property, the hashing functions are created from the same kernelized feature space by considering random data samples. This in itself enables the application of LSH to linearly inseparable data. The weighting vector w is calculated as:

$$w = K^{(-1/2)} e_P = V V^T e_P, \quad (7)$$

where V is the eigendecomposition of K, and e_P is $|P|$-dimensional random binary vector, which corresponds to indices of P. $t << |e_P|$ random values in e_P are set to 1. Finally, the hashing function is then defined as:

$$h(\theta(x_i)) = sign \left(\sum_{j=1}^{|P|} w_j (\theta(x_i)^T \theta(x_j)) \right), \quad (8)$$

where $x_i \in X$ and $x_j \in P$. New hashing functions can be formed by zeroing e_P, and then choosing new t values to 1, in order to calculate new w. For each query the existing LSH methodology is used with the given hash function, in order to find its approximate nearest neighbours.

2.1. Parallelization using GPGPU

The proposed GPU implementation works as following. At first, a preprocessing step is performed on the host (CPU) side, where kernel matrix K_X is formed between $P \in X$ and $X - P$ by using radial basic function (RBF) $\kappa(x_i, x_j) = exp\left(\frac{-||\theta(x_i)-\theta(x_j)||^2}{2\sigma^2}\right)$, where $\{x_i, x_j\} \in P$ and σ is a free parameter. Then m hash functions are computed using eigen-decomposition of K, which represents the training dataset. Similarly, the query kernel matrix K_Q is build. The idea of the proposed method is to perform the approximate nearest neighbour query in parallel for multiple query points. The proposed implementation considers NVIDIA's Compute Unified Device Architecture (CUDA), which groups threads into blocks and these into grids. In the proposed method the data is stored in 1D grid. The computed vectors w_1,...,$w_{|P|}$ are stored in GPU constant (i.e. read-only) fast cached memory, while K_X and K_Q are transferred to the GPU global memory. In the first parallel stage each thread is responsible for a single point $x_i \in K$, where the E$_2$LSH [4] methodology is considered. Thus, both X and Q are hashed. This works by bucketing the hashed result [4, 7]:

$$h(x_i) = \left(\sum_{j}^{|P|} w_j K(x_i, x_j) \, mod \, M \right) mod \, c \qquad (9)$$

where $x_j \in P$, and K is either K_X or K_Q. $x_i \in P$, M is a large prime number to avoid integer overflow, and c denotes the number of buckets. In the second parallel stage, k approximate nearest neighbours from hashed X are computed for K_Q by calculating and comparing the actual distances to the points that are hashed into the same buckets as given query points. The GPU-based MLSH approximation proposed in [7] is considered strategy in this paper, in order to efficiently probe approximate neighbours of q_i. In this stage, each CUDA thread is responsible for one query point. Fig. 1 shows the process of applying parallel hashing for a single hash function h. Moreover, B_Q and I_Q integer arrays of size $|Q|$ are created in global memory. B_Q denotes the target bucket for each q_i, while I_Q denotes the index of q_i, which is initially set as $I_Q[q_i] = i$. After hashing is complete and B_Q is populated, it is sorted in ascending order together with I_Q by using parallel radix sort. Furthermore, two arrays of size c are required in GPU global memory, in order to store the starting position and size of each bucket within B_Q. The query is then performed with $|Q|$ threads in parallel, where the result of k nearest neighbours are calculated and R_Q of size $k \times |Q|$ is updated. Finally, I_Q is sorted back to the original ascending order of the query points, together with corresponding R_Q.

3. RESULTS

Two different satellite remote sensing datasets were considered in the experiments. Namely, a SAR dataset consisting of 3434 images with resolution of 160×160 pixels, and the UC Merced Land Use dataset [10] [1] consisting of 2100 images with the resolution of 256×256 pixels. Both datasets were already classified into urban and natural objects, the SAR dataset into 15 classes, while the optical dataset into 21 classes. At first, the images were transformed into a high-dimensional space based on the Bag-of-Words feature extraction method. The used local features are the raw pixel values and the codebook was learned by k-means clustering. The SAR images were represented by 250-dimensional feature vectors, while the optical imagery with 500-dimensional vectors (i.e. points). During the experiments, all the high-dimensional feature vectors were considered as the query dataset. The used GPU hardware was GeForce GTX Titan Black with 2880 CUDA cores.

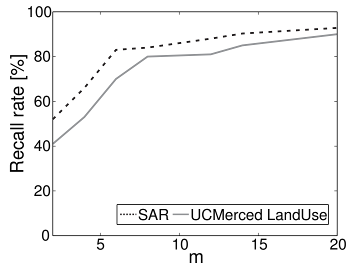

Fig. 2. Recall rate for $k = 5$ as the number of hash functions (m) increases.

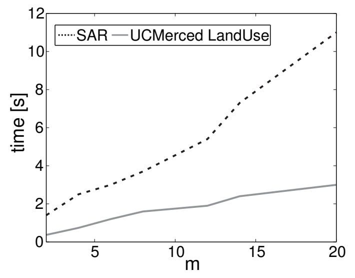

Fig. 3. Runtime for $k = 5$ as the number of hash functions (m) increases.

The purpose of the experiments within this paper was to show fast image retrieval. The recall rate was calculated as the percentage of images that had the nearest neighbour with the same class. During the experiments, size of P was chosen as $\sqrt{|X|}$ of the total given dataset, in order to guarantee sub-linear search times. Moreover, $t = |P|/2$ when generating e_P. The number of threads per CUDA block was 64, while the number of buckets for SAR dataset was set at 300, and for optical imagery at 200. Fig. 2 shows the recall rate and Fig. 3 the runtime as the number of used hash functions increase at $k = 5$. Transfer times between the host and GPU memory are already included within the measured runtime. Moreover, Fig

[1]UC Merced Land Use dataset public release: `http://vision.ucmerced.edu/datasets/landuse.html`

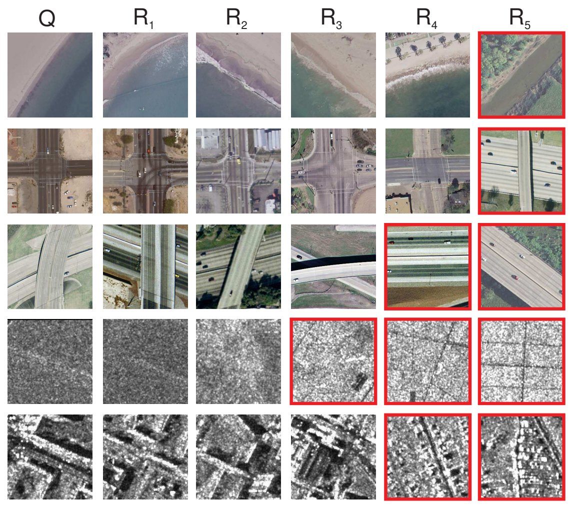

Fig. 4. Results for 5 query images (3 optical and 2 SAR) when $k = 5$ and $m = 5$. False positives are denoted with red border.

4. shows an example of false positives that can occur when using $m = 10$ and $k = 5$.

4. CONCLUSION

As the preliminary results have shown the proposed method can achieve fast image retrieval over remote sensing imagery, where the quality-speed trade-off can be adequately tuned. For future-work, the demanding tasks at the host side shall be computed on the GPU.

Acknowledgment

The authors would like to thank the TerraSAR-X science service system for the SAR data. This work was in part supported by the Slovenian Research Agency under grants 1000-13-0552 and J2-6764.

5. REFERENCES

[1] P. Blanchart, M. Ferecatu, and M. Datcu, “Cascaded active learning for object retrieval using multiscale coarse to fine analysis,” in *Proc. 18th IEEE Int Image Processing (ICIP) Conf*, 2011, pp. 2793–2796.

[2] M. Datcu and K. Seidel, “Human-centered concepts for exploration and understanding of Earth observation images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 3, pp. 601–609, March 2005.

[3] Chi-Ren Shyu, M. Klaric, G. J. Scott, A. S. Barb, C. H. Davis, and K. Palaniappan, “GeoIRIS: Geospatial information retrieval and indexing system — content mining, semantics modeling, and complex queries,” *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 4, pp. 839–852, 2007.

[4] M. Slaney and M. Casey, “Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes],” *Signal Processing Magazine, IEEE*, vol. 25, pp. 128–131, 2008.

[5] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing for scalable image search,” in *Computer Vision, 2009 IEEE 12th International Conference on*, 2009, pp. 2130–2137.

[6] J. Pan and D. Manocha, “Fast GPU-based locality sensitive hashing for k-nearest neighbor computation,” in *In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2011, pp. 211–220.

[7] N. Lukač and B. Žalik, “Fast approximate k-nearest neighbours search using GPGPU,” in *GPU Computing and Applications*, pp. 221–234. Springer Singapore, 2015.

[8] B. Demir and L. Bruzzo, “Kernel-based hashing for content-based image retrval in large remote sensing data archive,” in *Geoscience and Remote Sensing Symposium (IGARSS), 2014 IEEE International*, 2014.

[9] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe LSH: efficient indexing for high-dimensional similarity search,” in *Proceedings of the 33rd international conference on Very large data bases*, 2007, pp. 950–961.

[10] Y. Yang and S. Newsam, “Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification,” in *Proc. 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS ’10*, New York, NY, 2010, pp. 270–279.