Parallel Implementation of Nonparametric Clustering Algorithm HCA-MS on GPU Using CUDA

S. A. Rylov(✉)

Institute of Computational Technologies SB RAS, Novosibirsk, Russia
RylovS@mail.ru

Abstract. The present work explores nonparametric clustering algorithm HCA-MS. The combination of grid-based approach and Mean shift procedure allows the algorithm to discover arbitrary shaped clusters and to process large datasets, such as images. Parallel implementation of the HCA-MS algorithm on NVIDIA GPU using CUDA platform is presented. Provided experimental results on model data and multispectral images confirm the efficiency of the considered algorithm and its parallel implementation. The computation speedup on images was shown to be about 20x compared to 4 core CPU.

Keywords: Clustering · Nonparametric · Grid-based · Mean shift · Image segmentation · GPGPU · CUDA · Parallel computing

1 Introduction

Clustering is the task of grouping a set of objects in such way that objects in the same group are more similar to each other than to those in other groups (clusters). Clustering of large datasets is urgent in many applied problems of data analysis. For example, it is one of the most common approaches to satellite image segmentation [1]. Generally, a priori information about the probabilistic characteristics of classes, as well as training samples is often absent. Widely used clustering algorithms (k-means, ISODATA, EM) are based on the assumption of Gaussian distribution models and do not always provide required segmentation quality (especially for high spatial resolution satellite images) [2,3].

In such circumstances, nonparametric clustering methods are more attractive because of their ability to discover arbitrary shaped clusters [3]. However, high computational complexity strongly limits its application to large datasets, such as images. One of the best-known nonparametric mode-seeking algorithms that are capable of producing accurate results is Mean shift [4] and it has quadratic time complexity [5].

On the other hand, grid-based methods that divide feature space into a finite number of cells are also capable of discovering arbitrary shaped clusters and at

Y. Shokin and Z. Shaimardanov (Eds.): CITech 2018, CCIS 998, pp. 189–196, 2019.
https://doi.org/10.1007/978-3-030-12203-4_19

the same time they have high computational efficiency [6]. But their accuracy is limited by a grid structure [7].

Recently HCA-MS algorithm was proposed by the author to combine the best qualities of these two approaches [8]. It is based on the grid-based algorithm HCA with the following Mean shift procedure for clusters' borders refinement. The combination of these approaches allows obtaining higher clustering accuracy in comparison with grid-based approach and higher performance in comparison with Mean shift.

Today, most personal computers are equipped with graphics cards (GPU), the performance of which has grown dramatically in the last years. Consequently, general-purpose computing on graphics processing units (GPGPU) technologies are actively developing for solving time-consuming non graphics tasks [9].

The present work introduces a parallel implementation of HCA-MS clustering algorithm on GPU using CUDA platform. Experimental results on model data and multispectral images confirm the efficiency of the considered algorithm and its parallel implementation. The computation speedup on images was shown to be about 20x compared to 4 core CPU, where the parallel version of the algorithm for CPU was implemented with OpenMP.

2 Nonparametric Clustering Algorithm HCA-MS

This section briefly describes the nonparametric clustering algorithm HCA-MS [8]. This novel algorithm utilizes grid-based HCA clustering algorithm with Mean shift procedure, which is applied to the elements of border cells resulting in the refinement of the clusters' borders.

The first stage of the HCA-MS is to execute grid-based HCA algorithm, which was presented in [10] and briefly described below.

Let the set of objects X consist of d-dimensional vectors lying in the feature space R^d: $X = \{x_i = (x_i^1, \ldots, x_i^d) \in R^d, i = 1, \ldots, N\}$, and bounded by a hyper-rectangle $\Omega = [l^1, r^1] \times \cdots \times [l^d, r^d]$: $l^j = \min_{x_i \in X} x_i^j$, $r^j = \max_{x_i \in X} x_i^j$. Grid structure is formed by dividing Ω with hyperplanes $x^j = (r^j - l^j) \cdot i/m + l^j$, $i = 0, \ldots, m$ where m is the number of partitions in each dimension. The set of cells adjacent to B will be denoted by A_B. The *density* D_B of the cell B is defined as the number of elements from the set X belonging to the cell B.

The nonempty cell B_i is directly connected to the nonempty cell B_j ($B_i \to B_j$) if B_j is the cell with the maximum number that satisfies the conditions $B_j = \arg \max\limits_{B_k \in A_{B_i}} D_{B_k}$ and $D_{B_j} \geqslant D_{B_i}$. The nonempty adjacent cells B_i and B_j are directly connected ($B_i \leftrightarrow B_j$) if $B_i \to B_j$ or $B_j \to B_i$. The nonempty cells B_i and B_j are connected ($B_i \approx B_j$) if there exist $k_1, \ldots, k_l$ such that $k_1 = i$, $k_l = j$ and for all $p = 1, \ldots, l-1$ we have $B_{k_p} \leftrightarrow B_{k_{p+1}}$. The introduced connectedness relation leads to the natural partition of nonempty cells into the *connectedness components* $\{G_1, \ldots, G_S\}$. The connected component is defined as the maximum set of pairwise connected cells. *Representative cell* $Y(G)$ of the component G is defined as a cell with the maximum number that satisfies the condition $Y(G) = arg\, max_{B \in G} D_B$.

The determined connectedness components correspond to single-mode clusters, and their representative cells correspond to the density modes of these clusters.

Next, to construct an hierarchy between components we define the distance h_{ij} between adjacent components G_i and G_j as

$$h_{ij} = \min_{P_{ij} \in \Re_{ij}} \left[1 - \min_{B_{k_t} \in P_{ij}} D_{B_{k_t}} \Big/ \min \left(D_{Y(G_i)}, D_{Y(G_j)} \right) \right],$$

where $\Re_{ij} = \{P_{ij}\}$ is a set of all possible paths between representative cells $Y(G_i)$ and $Y(G_j)$, $P_{ij} = \langle Y(G_i) = B_{k_1}, \ldots, B_{k_t}, B_{k_{t+1}}, \ldots, B_{k_t} = Y(G_j) \rangle$ such that for all $t = 1, \ldots, l-1$: (1) $B_{k_t} \in G_i \cup G_j$; (2) B_{k_t}, $B_{k_{t+1}}$ are adjacent cells.

After forming a matrix of distances between adjacent components $\{h_{ij}\}$, the SLINK (nearest neighbor) algorithm for dendrogram construction is applied to it. The result of the algorithm is an hierarchical structure built on the set of connected components.

The HCA algorithm at low computational costs allows distinguishing clusters of complex shape and obtaining hierarchical clustering structure. Moreover, unlike the other well-known hierarchical algorithms [11], it allows separating clusters that are intersecting in the feature space. However, accuracy of cluster's separation highly depends on the grid structure, which can lead to mistakes, particularly if the grid parameter m is chosen unsuccessfully.

At the second stage of HCA-MS algorithm, data elements are grouped due to the cells they belong to, for the following quick access to the list of elements of an arbitrary cell.

At the third stage, non-empty cells located at the cluster's borders are considered. To each element of such cell Mean shift procedure is applied [4], which iteratively converges to the local density maximum: $x_{k+1} = m(x_k)$, where

$$m(x) = \frac{\sum\limits_{i=1}^{N} x_i \cdot K_{\mathrm{Ep}}(x - x_i)}{\sum\limits_{i=1}^{N} K_{\mathrm{Ep}}(x - x_i)}.$$

where the finite Epanechnikov kernel function is used:

$$K_{\mathrm{Ep}}\left(\frac{x - x_i}{h}\right) = \left(1 - \frac{||x - x_i||^2}{h^2}\right) \cdot I\left(||x - x_i|| \leqslant h^2\right),$$

where $I(x)$—an indicator function.

Smoothing parameter h is set equal to the width of the cell in the grid structure.

The shift process stops if the considered element moves to the other non-empty cell. In case the new cell belongs to another cluster the element is moved to this cluster. The maximum number of Mean shift iterations is limited by the parameter, which we will set to 3. Experimental studies on model data have shown that in most cases this parameter value is sufficient [8]. In general, it is possible to use more sophisticated stopping criteria, but since this is not the subject of this study, we will not consider them.

3 Parallel Implementation of the Clustering Algorithm HCA-MS with CUDA

CUDA is a powerful parallel computing platform and API model that allows using NVIDIA GPUs for general purpose processing. Modern GPUs contain thousands of cores, grouped into blocks under the control of multiprocessors. The cores of one block perform the same set of instructions, but on different data elements. Each core contains small number of registers and has quick access to a limited amount of shared memory within its block (managed cache). In addition, all the threads (executed on separate cores) can access large amount of global memory, but random access time to it is slow and takes hundreds of cycles. Synchronization of threads during execution is possible only within a block.

Parallel implementation of the grid hierarchical HCA algorithm on GPU using CUDA is described in detail in [12]. By itself, this algorithm has very fast performance: four-bands images of the size up to 100 megapixels are processed on CPU within 1 s and on GPU within 0.1 s. Considering HCA-MS algorithm, its first stage (HCA) computing time is insignificant.

At the second stage, data elements are sorted by the cells and an array of the first elements indices for each cell is formed. This stage is also not computationally time consuming, and its execution is performed on CPU.

The use of a weighting table (the number of elements with the same feature values) can significantly reduce computational cost of processing color images with common 256 quantization levels. Yet, when processing multispectral satellite images containing more than three spectral bands with radiometric resolution of 10–14 bits, the efficiency of this approach is diminished. Therefore, in this particular study we will not use a weighting table.

Finally, the third stage is the one that is time consuming. Mean shift processing of the border cells can be done independently from each other, therefore, parallelization is applicable. On CPU different threads just process different cells. But, on GPU each cell is processed by its block, where block's threads process the elements of the cell in parallel. Let's consider the processing scheme of a non-empty cell by a block of threads.

First, the connectedness components of all neighboring cells are read into the shared memory by the threads, as well as the indices of the first and the last elements of these cells. If among the neighboring cells no cell belongs to the different component, which means it is not a border cell, then the block finishes its work.

Otherwise, the elements of the cell are processed in parallel by the threads. To execute Mean shift step each thread goes through all the elements from the adjacent cells to search those in the radius h. Due to the fact that it is impossible to guarantee that the number of considered elements will not exceed the limited size of shared memory, data access is made through the global memory.

Yet, to optimize data access, managed cache was successfully utilized. The elements of each cell are read into the allocated array in shared memory by fragments of fixed size that is equal to the block size (the number of threads in the block). After the threads finish processing this part of data, the next

data fragment is uploaded. All block threads upload new data fragment synchronously, even if some threads are not involved in the cell elements processing. This optimization additionally reduced computation time by 20%.

4 Experimental Results

This section presents the results of experimental studies of HCA-MS algorithm on model data and images. It was shown that HCA algorithm is capable of distinguishing clusters of complex shape and produces better results than well-known clustering algorithms like K-means, EM, DBSCAN, OPTICS, DeLiClu, SLINK and Mean shift on complex data [12]. The considered issue is the efficiency of the proposed implementation to correct border mistakes caused by the grid effect. The clustering accuracy of HCA-MS algorithm is compared with the initial grid-based algorithm HCA and density-based algorithm Mean shift. The computation time of the proposed CUDA implementation of HCA-MS algorithm on GPU (GeForce GXT 770, 1536 cores) compared with execution on one and four cores of the CPU (Intel Core i5, 3.5 GHz) is presented. The parallel version of the algorithm for CPU was implemented with OpenMP.

Experiment 1. Two-dimensional synthetic dataset containing 3 classes [13] was clustered. One cluster described by a normal distribution is surrounded by two clusters in the form of rings (Fig. 1c). Mean shift clustering algorithm cannot extract multimode clusters in the form of rings in principle. In its turn, HCA algorithm successfully separates all three clusters, but makes some mistakes when the grid parameter is selected unsuccessfully: at $m = 30$ clustering accuracy is 98.21% (Fig. 1a); at $m = 42$ mistakes are made in 3 points (Fig. 1b); 100% accuracy is obtained only at $m = 46$ (Fig. 1c). However, HCA-MS algorithm is able to correct mistakes caused by the grid effect. As a result, 100% accuracy is obtained in all three cases: at $m = 30$, $m = 42$ and $m = 46$ (Fig. 1c).

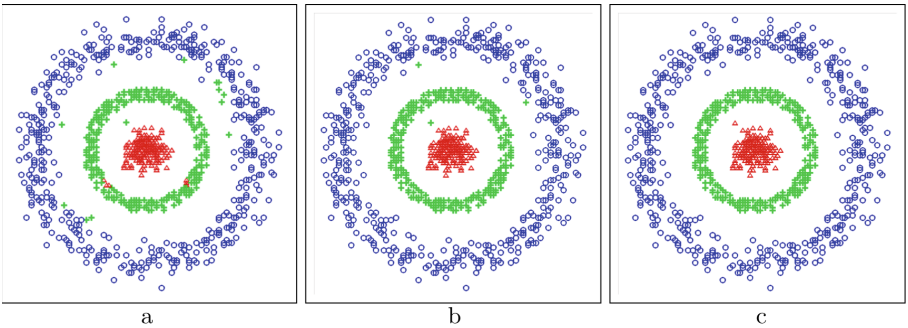


Fig. 1. The results of clustering the model dataset obtained by HCA algorithm at $m = 30$ (a), $m = 42$ (b), $m = 46$ (c); and the result of HCA-MS with the same parameter's values (c).

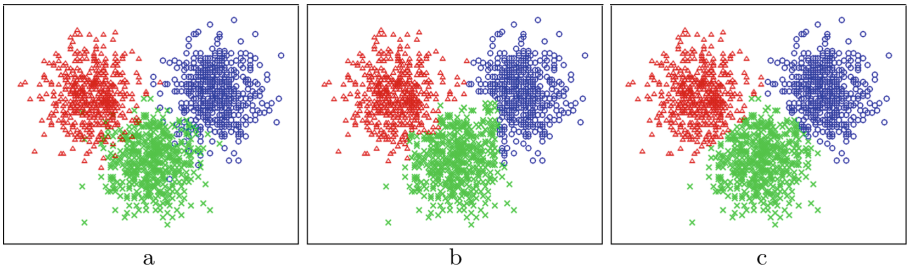


Fig. 2. Clustering results of the model dataset (a) by HCA algorithm (b) and by HCA-MS algorithm (c) at $m = 20$.

Experiment 2. The synthetic dataset containing three strongly intersecting classes with normal distribution [13] is shown in Fig. 2a. Clustering accuracy of this model by HCA algorithm at $m = 20$ is 94.93% (Fig. 2b). At this case, the accuracy can be increased by using finer grid: for example, at $m = 40$, the accuracy is 96%. However, very fine grid can be unacceptable when extracting clusters with complex structure. HCA-MS algorithm demonstrates 96.4% accuracy at $m = 20$ (Fig. 2c). While Mean-shift algorithm at most achieves 96.33% accuracy at $h = 27$. At the same time, other density-based algorithms (DBSCAN, OPTICS, DeLiClu) fail to adequately separate strongly intersecting classes.

Experiment 3. To assess the speedup effect of the GPU implementation, test data-sets of different size were generated. The data-sets follow uniform distribution in the limited three-dimensional feature space. HCA-MS algorithm processing time on the test data depending on the number of elements (from 500'000 to 10'000'000) performed on the GPU and one and four cores of the CPU is presented in Fig. 3a. The obtained speedup of the GPU execution in comparison with four cores of the CPU is shown in Fig. 3b. The results are presented for the grid parameter values $m = 20$ and $m = 32$, because these values are usually used for image clustering. Average speedup of the OpenMP parallel implementation performed on four cores of the CPU is 3.8x compared to the non-parallel version. The speedup of the proposed CUDA implementation on the GPU in comparison with OpenMP performance reaches 28x.

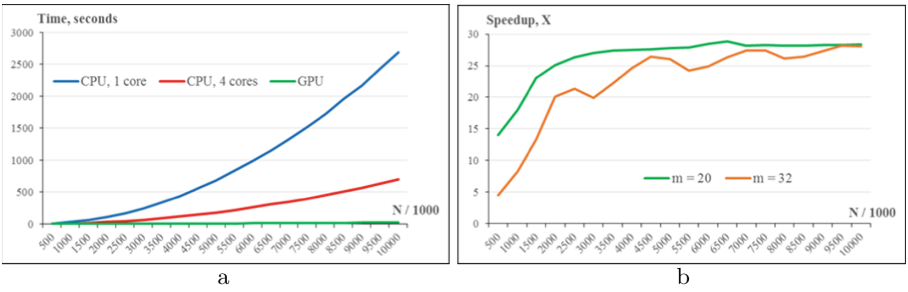
Experimental studies have shown that the speedup on GPU depends on cell density: denser cells are processed more efficiently. This is caused by the fact that each cell is processed by a block of threads. Therefore, if there is insufficient number of elements in a cell, then some part of the reserved threads may be unused. Thus, the speedup on GPU can decrease if the grid parameter m is increased. On the other hand, the cells are processed independently by the blocks. Therefore, the speedup directly depends on the number of streaming multiprocessors (SM). And while the number of cores per SM is almost constant (192 in Kepler and 128 in Maxwell and Pascal architectures), the number of SMs is determined by the total number of GPU cores, which provides a direct dependence of the algorithm CUDA performance on the number of GPU cores.

Experiment 4. The table below shows HCA-MS algorithm time performance on the GPU and four cores of the CPU on the images of different size at $m = 20$. In total 40 color photos and multispectral satellite images (WorldView-2 and Landsat-8) of different size [14] were processed. The full table is avable here [15]. The results showed that the average speedup of the OpenMP parallel implementation performed on four cores of the CPU is 3.7x compared to the non-parallel version and it does not fluctuate considerably. The average speedup of the GPU execution in comparison with four cores of the CPU is 22x at $m = 20$ and 19x at $m = 32$ (for the images containing more than 1 million pixels). Time performance of the algorithm at $m = 32$ is about 4x faster than at $m = 20$. However, it should be noticed that processing time strongly depends on the data itself, and for different images of the same size it can vary greatly.

Thus, the proposed parallel implementation on the GPU can process large multispectral images just in few minutes (Table 1).

Table 1. HCA-MS algorithm time performance on the images (in seconds).

Number of bands	3	3	3	3	4	3	3	4	3	3	4
Image size (megapixels)	0.3	1.2	4.2	4.2	4.2	5	9	12.5	13.8	25	25
CPU, 4 cores	6.6	93	302	6872	701	1465	1948	2055	18186	16623	11013
GPU	0.5	5.2	10.6	357	33	49	96	115	679	587	510
Speedup	14.4	18.0	28.4	19.2	21.3	29.7	20.3	17.8	26.8	28.3	21.6



a b

Fig. 3. HCA-MS algorithm processing time on the test data depending on the number of elements N performed on the GPU and one and four cores of the CPU (a); the speedup of the GPU performance in comparison with 4 cores of the CPU (b).

5 Conclusion

Parallel implementation of the nonparametric HCA-MS clustering algorithm on GPU using CUDA platform is presented. Experimental results on model datasets showed the ability of the algorithm to correct mistakes caused by the grid effect, reaching clustering accuracy level of the well-known Mean shift algorithm. The experiments showed, that the proposed parallel implementation on GPU allows processing multispectral images 20 times faster than on CPU (4 cores). Thereby, large multispectral images can be clustered just in few minutes.

References

1. Xie, Y., Sha, Z., Yu, M.: Remote sensing imagery in vegetation mapping: a review. J. Plant Ecol. **1**(1), 9–23 (2008)
2. Zadkarami, M.R., Rowhani, M.: Application of skew-normal in classification of satellite image. J. Data Sci. **8**, 597–606 (2010)
3. Sarmah, S., Bhattacharyya, D.K.: A grid-density based technique for finding clusters in satellite image. Pattern Recogn. Lett. **33**(5), 589–604 (2012)
4. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 790–799 (1995)
5. Freedman, D., Kisilev, P.: Fast mean shift by compact density representation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1818–1825. IEEE (2009)
6. Ilango, M.R., Mohan, V.: A survey of grid based clustering algorithms. Int. J. Eng. Sci. Technol. **2**(8), 3441–3446 (2010)
7. Krstinic, D., Skelin, A.K., Slapnicar, I.: Fast two-step histogram-based image segmentation. IET Image Process. **5**(1), 63–72 (2011)
8. Rylov, S.A.: Nonparametric clustering algorithm for image segmentation combining grid-based approach and mean-shift procedure. In: CEUR Workshop Proceedings, vol. 2033, pp. 150–155 (2017)
9. Choquette, J., Giroux, O., Foley, D.: Volta: performance and programmability. IEEE Micro **38**(2), 42–52 (2018)
10. Pestunov, I.A., Rylov, S.A., Berikov, V.B.: Hierarchical clustering algorithms for segmentation of multispectral images. Optoelectron. Instrument. Data Process. **51**(4), 329–338 (2015)
11. Lu, Y., Wan, Y.: PHA: a fast potential-based hierarchical agglomerative clustering method. Pattern Recogn. **46**(5), 1227–1239 (2013)
12. Rylov, S.A., Pestunov, I.A.: Fast hierarchical clustering of multispectral images and its implementation on NVIDIA GPU. JPCS **1096**, 012039 (2018)
13. Rylov, S.A.: Model datasets. [Electronic resource]. https://drive.google.com/open?id=0ByK9GtU5ExExRnZwdFNmRHRWdFk. Accessed 20 Apr 2018
14. Image datasets for clustering. [Electronic resource]. https://drive.google.com/open?id=0ByK9GtU5ExExWXpGRjU5WVFHcDg. Accessed 20 Apr 2018
15. Table: HCA-MS algorithm time performance on the images. [Electronic resource]. https://drive.google.com/file/d/1xA89kC3tixwEUMLaX_pfEOTJ-s-Uh0-8/view?usp=sharing Accessed 14 Oct 2018