VDBSCAN+: Performance Optimization Based on GPU Parallelism

Carlos Roberto Valêncio
São Paulo State University
Departamento de Ciências de Computação e Estatística
São José do Rio Preto, São Paulo, Brazil
valencio@ibilce.unesp.br

Guilherme Priólli Daniel
São Paulo State University
Departamento de Ciências de Computação e Estatística
São José do Rio Preto, São Paulo, Brazil
guipriollidaniel@gmail.com

Camila Alves de Medeiros
São Paulo State University
Departamento de Ciências de Computação e Estatística
São José do Rio Preto, São Paulo, Brazil
camila.alves.medeiros@gmail.com

Adriano Mauro Cansian
São Paulo State University
Departamento de Ciências de Computação e Estatística
São José do Rio Preto, São Paulo, Brazil
adriano@ibilce.unesp.br

***Abstract*— Spatial data mining techniques enable the knowledge extraction from spatial databases. However, the high computational cost and the complexity of algorithms are some of the main problems in this area. This work proposes a new algorithm referred to as VDBSCAN+, which derived from the algorithm VDBSCAN (Varied Density Based Spatial Clustering of Applications with Noise) and focuses on the use of parallelism techniques in GPU (Graphics Processing Unit), obtaining a significant performance improvement, by increasing the runtime by 95% in comparison with VDBSCAN.**

Keywords*— *spatial data mining; spatial clustering; GPU (Graphics Processing Unit); VDBSCAN (Varied Density Based Spatial Clustering of Applications with Noise).

I. INTRODUCTION

The progresses of technologies for collection of geographical information, such as GPS (Global Positioning System), remote sensing, spatial location systems, geographic information systems, enabled the storage of a large amount of spatial data [1]. These data are essential for the decision-making process; however, due to the high complexity of this type of data and their relationships, the use of conventional data mining techniques is improper [2].

Accordingly, the concept of spatial data mining emerged, enabling the extraction and discovery of implied knowledge, spatial relationships, as well as other information that are not explicitly included in the spatial databases [1].

The spatial data mining techniques have been used in different fields of application. Thus, over the last few years, new algorithms have been proposed in order to make the extraction of useful knowledge from the spatial databases more efficient and effective [3].

One of the main difficulties in the application of spatial data mining is the processing time required by the algorithms, which increases proportionally to the number of tuples from the databases analyzed, as the performed operations are computationally expensive and complex.

On the other hand, the modern GPUs (Graphics Processing Unit) are robust architectures that perform intensive calculations and are increasingly used with respect to scientific issues with significant computational costs. Although the architecture was initially designed to be used in computer graphics, it also provides support for general-purpose computing [4]. Thus, GPU computing is able to implement many parallel algorithms with direct use of graphics hardware [5], making it an attractive alternative to overcome performance problems of the algorithms [6].

In this scenario, the use of parallelism techniques for GPU is proposed to optimize the algorithm VDBSCAN (Varied Density Based Spatial Clustering of Applications with Noise) [7], in order to make the process of extracting knowledge from spatial databases faster. This work presents an original contribution to the area, while the strategy adopted enabled to significantly reduce the processing time of the original algorithm.

This work is organized as follows: in section 2, the bibliographic review is presented; in section 3, related works are discussed; in section 4, is presented the developed algorithm; in section 5, the tests and results are shown and in section 6, the conclusions are presented.

II. BIBLIOGRAPHICAL SURVEY

In this section, we present the basic concepts for understanding the work developed as well as related works found in the literature.

A. Spatial data mining

Spatial data have a geographic location and they emerged in order to represent the real-world spatial entities. These types of data allow the geographic phenomena to be represented by

points, lines, polygons and other geometric shapes [8], and that is why they are considered complex data, whereas conventional data may be classified as alphanumeric data [3].

Spatial data mining is the technique that aims to extract hidden, unknown, reliable, useful and understandable knowledge from the large amount of data within spatial databases. In this process, knowledge from several areas is used: statistical methods, pattern recognition, artificial intelligence, neural networks, machine learning, expert systems, etc. The spatial data mining process aims to reveal internal relationships and data trends, providing the knowledge that will support for decision making related to technical and management areas [1], [8], [9].

There are several techniques used to extract knowledge from spatial data. One of them is the spatial data clustering, which aims to organize a set of items into clusters so that the objects of a same cluster are similar to each other and different from the other clusters, taking into account their geographical location. One of the advantages of using this technique is that the clusters are obtained without the need for prior knowledge, in addition to be easily applied in several areas, such as: geographic information systems, analysis of satellite images, analysis of medical images, marketing, etc. [3],[10], [11].

B. Base Algorithms for VDBSCAN+

DBSCAN (Density Based Spatial Clustering of Applications with Noise) [12] is one of the main density-based spatial data clustering algorithms. For this reason, DBSCAN was always subject of research aiming at improving the results and the performance of the algorithm.

In 2007, a algorithm was proposed, referred to as VDBSCAN, which uses an improved and more recent approach to DBSCAN, enabling clusters with varied densities [7].

In this regard, VDBSCAN is organized in two stages: finding Eps values and creating clusters. During the first stage, the algorithm creates the k-dist set chart, which contains a set of points representing the distances measured between each object and their k-nearest neighbor. The k-value is the minimum number of points included in the clusters so that they can be formed and it is derived from the MinPts for DBSCAN. From this chart, the density levels searched in the database are defined [7].

During the creation of clusters, VDBSCAN executes the steps of DBSCAN for each Eps value, in order to be able to find clusters with varied densities [7].

In Fig. 1, the behavior of the algorithm DBSCAN during the classification of points may be observed. The point colored purple is the core in a circumference with an Eps radius, since around itself the number of objects contemplated is greater or equal to the minimum number required in a cluster. Blue points are reachable are reachable, since the distance between them and the core is less than the Eps radius. The point colored green also is the border, which is reachable by the circumference border relative to purple point, but the circumference around the green point do not include the minimum number of points. Finally, the red point is noise, not reachable from any point, which can be verified by the circumference made around it.

Generally, the density-based spatial data clustering algorithms are important for the identification of knowledge and they have some advantages, such as lack of dependence on the number of clusters and capacity of finding clusters with arbitrary shapes [13]. Nonetheless, most of them have a high computational cost, which could make the application in large spatial database unfeasible [14].

C. GPU (Graphics Processing Unit)

GPU is high-performance architecture specialized in processing images. Over the last few years, the GPUs, in addition to perform its traditional function of 3D graphics pipeline, has also become an integral part of the current high-performance computing systems. Currently, the GPUs may implement many parallel algorithms by directly using graphics hardware. Thus, the algorithms that require a high computational cost when using GPUs reached an extraordinary speedup in many cases [4-6].

The strategy of using the video card processor to execute general-purpose tasks traditionally carried out by the CPU is referred to as GPGPU (General-Purpose Computation on Graphics Processing Unit), also known as GPU Computing [15], provided that the best known solutions for GPU parallelism are OpenCL (Open Computing Language) and CUDA (Compute Unified Device Architecture) [16].

The algorithm proposed was developed in CUDA since it is the most widely used. Such architecture is detailed below.

CUDA is an architecture of data structure and a parallel processing in GPU introduced by NVIDIA [17]. Such architecture involves an API (Application Programming Interface) that includes support for mathematical functions, functions of computer graphics, libraries, runtime and driver support [18]. In Fig. 2, the CUDA architecture is presented.

CUDA enables to develop high-level parallel programs. The CPU executes the sequential part of the applications and the GPU executes their parallel functions. The host (CPU) and the device (GPU) should keep their own memory space separately [19].

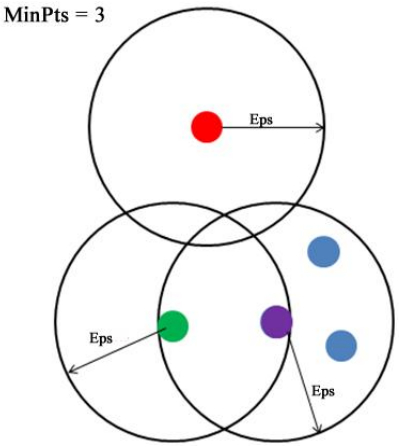

Fig. 1. DBSCAN behavior during classification of points

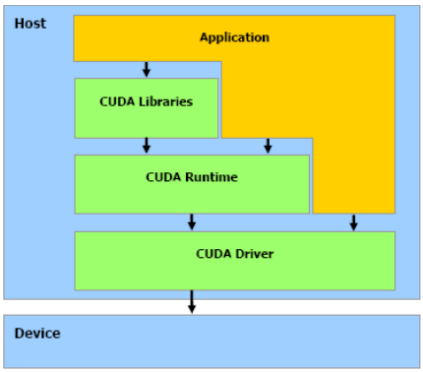

Fig. 2. CUDA Architecture [16], [20]

A thread d is the smallest execution unit in CUDA, and a group of threads constitutes a block. All threads of a block share the same processor cache memory. A group of blocks constitutes a grid, and that a GPU contains several grids [18]. Fig. 3 shows a grid with dimensions 2x3 with blocks sized 3x4, in which the using of the hierarchy of threads is proposed for CUDA programming model.

The parallel program executed in the device is referred to as kernel. In the prototype, a kernel represents the parallel procedure in a grid and it is used to specify the number and the dimensions of the threads and blocks [18].

D. Related Works

Some works have been found in the literature aiming at solving performance problems of spatial data mining algorithms through parallelism.

CLARANS (Clustering Large Applications based on Randomized Search) is a clustering algorithm based on the partitioning method [21]. In 2002, the algorithm ICLARANS was proposed that improves of CLARANS through the Parallel Virtual Machine (PVM), which is a model that enables a heterogeneous collection of distributed Workstations and personal computers functioning as a single high-performance parallel machine [22].

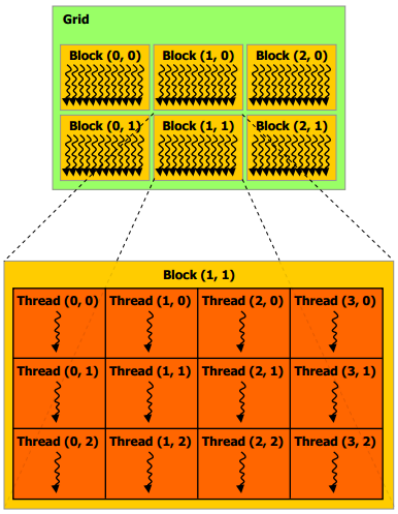

Fig. 3. Hierarchy of Threads for CUDA [19]

In a more recent approach, CLARANS was also used as a basis for a parallel algorithm for spatial data mining, known as PCLARANS, which uses the distributed processing as parallelism, however only the verification of neighbors is performed in parallel [23].

The algorithms ICLARANS and PCLARANS proved to be superior to the traditional CLARANS both in quality and performance. Nevertheless, for being derived from CLARANS, the algorithms ICLARANS and PCLARANS need two input parameters to be processed: maximum and minimum number of neighbors [23], [24].

The DBSCAN is another algorithm that has always been subject of researches in order to improve the results and the performance, as it is one of the mostly used algorithms in the area. Additionally, DBSCAN requires only one input parameter and it is superior to CLARANS in efficiency and effectiveness [12]. Hence, it was proposed the parallel DBSCAN, which is an algorithm that uses the R*-tree as data structure with spatial index to improve the computational performance and, consequently, to reduce the runtime [24].

Another algorithm derived from DBSCAN was CUDA-DClust, which combines several concepts to explore the characteristics of the GPU for data clustering, such as the parallel expansion of the cluster supported by concepts of search strings, the parallel nearest neighbor that may be accelerated by the use of a hierarchical index structure and the effective load balancing between microprocessors. The experiments demonstrated that CUDA-DClust outperforms DBSCAN executed in CPU by order of magnitude and produces equivalent clusters. The results may be improved due to the use of an appropriate index structure, which originated another algorithm referred to as CUDA-DClust* [25].

Recently, another approach was proposed to improve DBSCAN performance through GPU. However it was based on the parallel DBSCAN algorithm [25] and used the R*-tree as data structure to improve memory management. The algorithm is based on GPU to scan large databases and it maintains linear memory scalability. A performance analysis showed that the new implementation is faster than the sequential algorithm in terms of total runtime [14].

Despite the fact that the optimized algorithms based on DBSCAN are very effective, they have the same difficulty as DBSCAN in finding datasets with varied densities. As the algorithm VDBSCAN does not present such difficulty, it was used as basis for the development of the proposed work, which involves the use of the GPU to improve such algorithm performance without impairing the results.

III. VDBSCAN+ ALGORITHM

The proposed algorithm, VDBSCAN+, is classified into the category of density-based spatial clustering, which obtains clusters with varied densities and uses GPU resources as to significantly improve the efficiency.

VDBSCAN+ was implemented in CUDA aiming at using all the potential of the GPU architecture and so achieving a

high degree of parallelism especially in the calculations of distances between the points, and becoming more efficient than the algorithms executed in CPU.

By proposing VDBSCAN, the authors suggest that the k-value is 3. However, no formula was found to define the k-value. Thus, after carrying out several experiments, it was verified that it would be necessary to adjust the k-value to 35 for the database used in this work, so that VDBSCAN+ achieves better clustering results. The algorithm VDBSCAN+ is organized into 5 steps executed in CPU and uses 4 kernels executed in GPU, as shown in Fig. 4.

Step 1 is the initial step of the algorithm, in which the main point, that shall be taken as a basis for comparison of distances between the other points, is selected and transcribed together with the others to the GPU memory.

In CUDA k-dist Kernel, the distances between the main point and the other points are calculated and stored in the distance vector, in parallel within the graphic card. Such task may be carried out as the GPU architecture enables to use several kernel methods simultaneously, making the process more agile.

After calculating the distances for all points, the CUDA k-dist Kernel returns the distance vector to the CPU, where it will sort this vector and add the value of the Kth position to the k-dist vector. Steps 1, 2 and the CUDA k-dist Kernel are executed as long as there are points to be calculated.

Step 3 is responsible for sorting the k-dist vector in order to identify, in step 4, the Eps that will be used to create the clusters. After step 4, DBSCAN is executed for each Eps value found so as to find clusters with varied densities. Considering the implementation in CUDA using all potential of the GPU processing, DBSCAN is programmed differently from the traditional way.

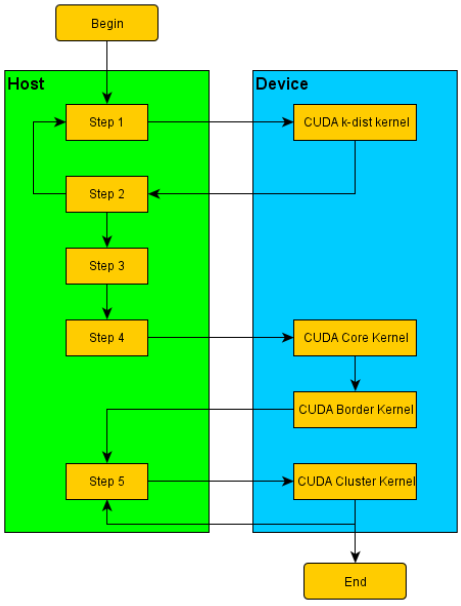

Fig. 4. Flowchart of VDBSCAN+

Firstly, the CUDA Core Kernel is executed in parallel for several objects. This method receives the main point and the Eps value and calculates the distance for each neighbor. If the distance is lower or equal to the Eps, the number of neighbors will be increased. If the number of neighbors is higher than the minimum number of points required to create a cluster, the main point is defined as center.

Once all centers are signaled, the algorithm executes the CUDA Border Kernel, and the borders are identified. This kernel is also executed at once for several points in order to harness the parallel power of the video card.

In CUDA Border Kernel, the main point and the Eps value are received. If the point is signaled as center, then the distance is calculated for each neighbor. If the distance is lower or equal to the Eps, the neighbor is signaled as border.

Finally, after defining all cluster centers and borders, the algorithm executes step 5 by selecting a main point characterized as a center, but that has no cluster, and it verifies if the current core region will have at least K members to create the cluster.

Thereafter, the CUDA Cluster Kernel is initiated in the GPU in order to create the clusters. The method receives the main point, the Eps value and the clustering index that the algorithm wishes to create and it occurs as follows. If the point is defined as center and if it is part of the current cluster, it calculates the distance for each neighbor and verifies if the distance from this object is lower or equal to the Eps, if so, the point is added to the cluster.

During this final stage, step 5 and the CUDA Cluster Kernel method will be repeated until there are no more points to be tested.

IV. Experiments and Results

In this section, the experiments and results obtained by the proposed algorithm are presented in order to demonstrate the efficiency of the algorithm VDBSCAN+ with respect to the original VDBSCAN.

To carry out the experiments, the algorithms were applied to a database of occupational accidents occurred in the region of São José do Rio Preto, São Paulo. The data are supplied by SIVAT - Sistema de Informação e Vigilância de Acidentes do Trabalho (Work Accident Vigilance System), a computer system developed by the GBD - *Grupo de Banco de Dados* (Database Group) together with the CEREST - *Centro de Referência de Saúde do Trabalhador* (Worker's Health Reference Center) of São José do Rio Preto and Ilha Solteira, in which may be found more than 70 thousand notifications reported and more than 17 thousand are georeferenced data, enabling the application of the algorithms VDBSCAN and VDBSCAN+. For the execution of the algorithms we used the spatial attribute that corresponds to the geographical location of the accident.

The algorithms were executed in a machine with CPU Intel i7 multi-core and a GPU NVIDIA GeForce 525M, which has 96 processing cores. The experiments and results obtained are described below.

A. *Experiment I*

In the first experiment, it was decided to compare VDBSCAN+ executed in GPU with the parallel execution of VDBSCAN in CPU using different number of threads.

The experiments were performed considering 1, 2, 4 and 8 threads executed in CPU, totalizing 17,131 points. The results obtained are presented in Table I. In the chart of Fig. 5, it is presented a comparison between the differences of VDBSCAN+ and VDBSCAN runtimes obtained for different numbers of threads.

After performing the experiment, it is evident that GPU performance is greater than the CPU performance, regardless of the number of threads the CPU uses to execute the algorithm VDBSCAN. And, the lower the number of threads that the algorithm executes in CPU, the greater is the runtime difference with respect to the algorithm executed in GPU. In this case, the algorithm in GPU reduced by 95% of the runtime consumed by the CPU and when it comes to large-scale database, it is regarded as a significant improvement.

B. *Experiment II*

The second experiment aimed at verifying the efficiency of the algorithm proposed for different volumes of data in the database, provided that the runtimes were obtained for each data set and are contemplated in Table II. In the chart of Fig. 6, it is observed that VDBSCAN+ obtained a significant improvement in the runtimes compared to VDBSCAN.

By analyzing the curves of each algorithm in the chart presented in Fig. 6, it is possible to note that the curve of VDBSCAN has an exponential growth in the runtime in comparison with the number of points processed, but, on the other hand, the curve of VDBSCAN+ has a linear growth. This emphasizes the efficiency of the algorithm proposed even for different datasets.

TABLE I. NUMBER OF THREADS FOR VDBSCAN AND THEIR RELEVANT RUNTIMES

Algorithm	VDBSCAN+	VDBSCAN			
Processing	GPU	CPU			
Number of Threads	-	8	4	2	1
Runtime (s)	11	56	72	125	238

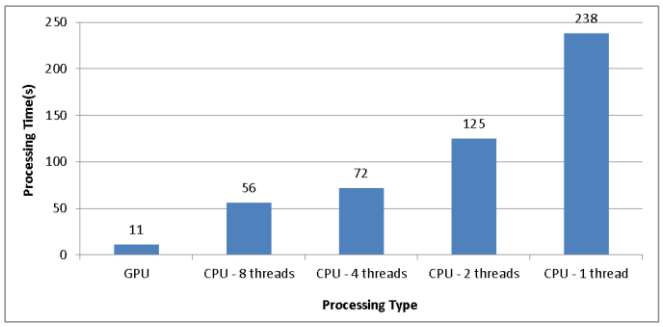

Fig. 5. Comparison between VDBSCAN+ and VDBSCAN runtimes (in seconds) using different numbers of threads

C. *Experiment III*

In the third experiment performed, we analyzed a spatial dataset of the clusters created by VDBSCAN+, which were identical to those created by the original VDBSCAN, besides having their runtime reduced. First, both algorithms processed 2,534 georeferenced accident sites and it was verified that they obtained 32 identical clusters. The clusters generated by both algorithms may be observed in Fig. 7.

As the algorithm proposed did not change the clusters found, the runtimes were compared and the result was quite satisfactory once the algorithm proposed outperformed the VDBSCAN. In Fig. 8, a comparison chart between both algorithms is presented, demonstrating the performance improvement obtained with the algorithm proposed.

VDBSCAN was executed in CPU using 8 threads and its runtime was of 43.5 seconds, on the other hand, the runtime of the algorithm executed in GPU was of 2.5 seconds. Therefore, Experiment 3 showed that proposed algorithm is capable of reducing the runtime by 94% and at the same time it maintains the clusters generated by the original algorithm, which proves VDBSCAN+ efficiency.

TABLE II. SETS OF SPATIAL POINTS AND ITS RELEVANT PROCESSING TIMES (IN SECONDS) IN THE CPU AND GPU

No. of Points	**15,000**	**30,000**	**45,000**	**60,000**
VDBSCAN	60 s	206 s	462 s	783 s
VDBSCAN+	11 s	26 s	54 s	76 s

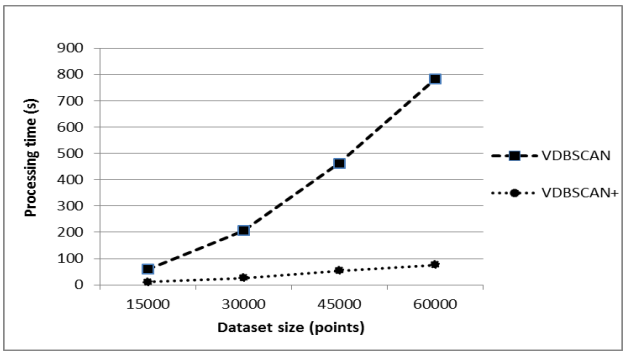

Fig. 6. Runtime(s) of the algorithms for different sets of points

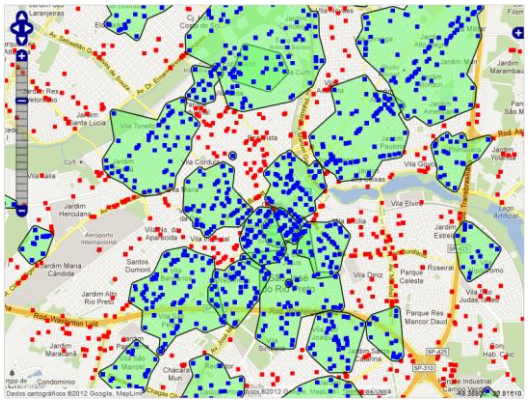

Fig. 7. Clusters generated by the algorithm VDBSCAN and VDBSCAN+

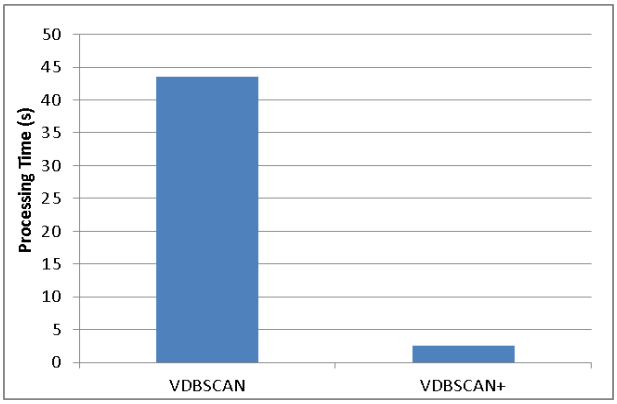

Fig. 8. Runtimes of VDBSCAN+ and VDBSCAN

V. CONCLUSIONS

The use of techniques and algorithms applied to spatial data is very important to knowledge extraction. However, the runtimes of these algorithms may be unfeasible to large volumes of data. In this scenario, the algorithm VDBSCAN+ is suggested as it uses GPU resources to achieve a better performance during the execution of one of these algorithms.

Through the results obtained, it is verified that spatial data mining techniques may be optimized by using GPU resources, as in the experiments performed in an actual database with more than 17 thousand georeferenced records, VDBSCAN+ was capable of reducing by 95% of the processing time. Experiments were also carried out with different volumes of data and with different numbers of threads for CPU processing, provided that in all cases the algorithm was efficient.

For future works, it is intended to use multiple GPUs and data structure in the algorithm, to add semantic identification in the database and also techniques for multi-relational search.

REFERENCES

[1] H. Yueshun and X. Wei, "A study of spatial data mining architecture and technology," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 163–166.

[2] S. Shekhar, P. Zhang, and Y. Huang, "Trends in spatial data mining," in *Next Generation Challenges and Future*, Minneapolis: MN, 2003.

[3] C. R. Valêncio, C. A. de Medeiros, F. T. Ichiba and R. C. G. de Souza, "Spatial clustering applied to health area," in *2011 12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2011, pp. 427–432.

[4] D. M. Coleman and D. R. Feldman, "Porting existing radiation code for GPU acceleration," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 1–6, 2013.

[5] Luebke and G. Humphreys, "How GPUs work," *Computer*, vol. 40, no. 2, pp. 96–100, Feb. 2007.

[6] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, May 2008.

[7] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise," in *2007 International Conference on Service Systems and Service Management*, 2007, pp. 1–4.

[8] J. Wang, X. Chen, K. Zhou, H. Zhang, and W. Wang, "Research of GIS-based spatial data mining model," in *2009 Second International Workshop on Knowledge Discovery and Data Mining*, 2009, pp. 159–162.

[9] K. Ravikumar and A. Gnanabaskaran, "ACO based spatial data mining for traffic risk analysis," in *2010 International Conference on Innovative Computing Technologies (ICICT)*, 2010, pp. 1–6.

[10] J. Mennis and D. Guo, "Spatial data mining and geographic knowledge discovery—An introduction," *Computers, Environment and Urban Systems*, vol. 33, no. 6, pp. 403–408, Nov. 2009.

[11] C. R. Valêncio, T. Kawabata, C. A. Medeiros, R. C. G. Souza, and J. M. Machado, "3D geovisualisation techniques applied in spatial data mining," in *Machine Learning and Data Mining in Pattern Recognition, LNCS*, P. Perner, Ed. Springer Berlin Heidelberg, 2013, pp. 57–68.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996, pp. 226–231.

[13] Z. Hua, W. Zhenxing, Z. Liancheng, and W. Qian, "Clustering algorithm based on characteristics of density distribution," in *2010 2nd International Conference on Advanced Computer Control*, 2010, pp. 431–435.

[14] R. J. Thapa, C. Trefftz, and G. Wolffe, "Memory-efficient implementation of a graphics processor-based cluster detection algorithm for large spatial databases," in *2010 IEEE International Conference on Electro/Information Technology*, 2010, pp. 1–5.

[15] L. G. A. Yano, "Avaliação e comparação de desempenho utilizando tecnologia," Monograph (Undergraduate Course), São Paulo State University, São José do Rio Preto, 2010 [in Portuguese].

[16] R. Cook, E. Dube, I. Lee, L. Nau, C. Shereda, and F. Wang, "Survey of novel programming models for parallelizing applications at exascale," 2011.

[17] Y. Zhao, Z. Huang, B. Chen, Y. Fang, M. Yan, and Z. Yang, "Local acceleration in distributed geographic information processing with CUDA," in *2010 18th International Conference on Geoinformatics*, 2010, pp. 1–6.

[18] R. C. Detomini,. "*Exploração de paralelismo em criptografia utilizando GPUs"*. Monograph (Undergraduate Course), São Paulo State University, São José do Rio Preto, 2010 [in Portuguese].

[19] NVIDIA, "NVIDIA CUDA C programming guide," 2012. [Online]. Available:http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf. [Accessed: 12-Aug-2013].

[20] NVIDIA, "CUDA programming guide," 2009. [Online]. Available: http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf. [Accessed: 04-Feb-2010].

[21] R. T. Ng, "CLARANS: a method for clustering objects for spatial data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1003–1016, Sep. 2002.

[22] X. Z. Ya-Ping Zhang, Ji-Zhao Sun, Yi Zhang, "Parallel implementation of CLARANS using PVM," in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, vol. 4, pp. 1646–1649.

[23] J.-S. Xue, "Parallel CLARANS - improvement and application of CLARANS algorithm," in *2010 International Conference on Computer and Communication Technologies in Agriculture Engineering*, 2010, pp. 248–251.

[24] D. Arlia and M. Coppola, "Experiments in parallel clustering with DBSCAN," in *7th International Euro-Par Conference Manchester on Parallel Processing*, 2001, pp. 326–331.

[25] C. Böhm, R. Noll, C. Plant, and B. Wackersreuther, "Density-based clustering using graphics processors," in *Proceeding of the 18th ACM Conference on Information and Knowledge Management* - CIKM'09, 2009, p. 661.