High Performance Analytical Pathology Imaging Database for Algorithm Evaluation

Fusheng Wang, Jun Kong, Jingjing Gao, Cristobal Vergara-Niedermayr, David Adler*, Lee Cooper, Weian Chen, Tahsin Kurc, Joel Saltz

Center for Comprehensive Informatics, Emory University, Atlanta, GA, USA
* IBM Spatial Database Technology, Poughkeepsie, NY, USA

Abstract. Algorithm evaluation provides a means to characterize variability across image analysis algorithms, validate algorithms by comparison with human annotations, synergize results from multiple algorithms for performance improvement, and facilitate algorithm sensitivity studies. High-resolution pathology images contain rich micro-anatomic information. An enormous amount of data can be derived from these images – e.g., 10^6-10^7 objects and features from a whole slide image with 10^{10} pixels resolution. The sizes of images and analysis results pose significant challenges in validating and comparing pathology imaging algorithms. Addressing these challenges requires a high performance computing approach for managing and querying spatial objects and features. Our work proposes and develops a high performance, database-supported framework to support algorithm validation and comparison.

1 Introduction

Evaluation and validation of image analysis algorithms is an important component in biomedical imaging studies, because the efficacy of an analysis pipeline generally depends on the characteristics of specimens and images used in the study, types of image processing operations employed, and the study objectives. A systematic approach for algorithm evaluation can facilitate the development of refined algorithms that can better support biomedical research and computer aided diagnosis. It can also substantially help the development of common training and test datasets from various sources to establish public shared data archives with well-understood results and algorithm performance and to support further algorithm evaluation in a community of researchers. However, algorithm evaluation and comparison frameworks have to address data processing, management, and query requirements arising from large volumes of image data and analysis results. *We propose, develop, and evaluate a high performance, database-supported system to support validation and comparison of image analysis algorithms targeting high-resolution microscopy images.*

Our work is motivated by whole slide image analysis studies carried out as part of integrative in silico experiments at the In Silico Brain Tumor Research Center (ISBTRC), one of the NCI caBIG® In Silico Research Centers of Excellence (https://wiki.nci.nih.gov/display/ISCRE). ISBTRC is a collaborative effort of four

institutions: Emory University, Thomas Jefferson University, Henry Ford Hospital, and Stanford University, and focuses on integrative translational research on brain tumors by analyzing complementary data types (genomics, imaging, clinical outcome). Microscopy image analysis algorithms are central to in silico experiments carried out at ISBTRC. These algorithms are used to extract, quantify, and classify the spatial features and characteristics of high-resolution whole slide tissue images. The results from image analyses are correlated with genomic and clinical outcome data to develop better biomarkers and study the mechanisms of disease progression. In this paper, we focus on segmentation algorithms as a case study. The segmentation step in pathology image analysis is critical to the success of downstream analysis steps such as feature computation and classification.

The need for algorithm evaluation and comparison arises in many ISBTRC image analysis cases. **i) Algorithm Validation.** Algorithms are tested, evaluated and improved in an iterative manner. This involves a formal testing phase where segmentations done by pathologists are captured and compared to an algorithm's output. The results are evaluated to assess inter-observer variability between pathologists, between algorithms and humans, and between different algorithms. **ii) Algorithm Consolidation.** Multiple algorithms can be developed in a study to solve the same problem. Different algorithms may have their own unique strengths on analyzing images of certain characteristics. In those cases, an array of algorithms can be aggregated in a complementary way. Consolidating results from multiple algorithms (i.e., an ensemble approach) can lead to better analysis results. **iii) Algorithm Sensitivity Studies.** An algorithm often includes a set of parameters that can be adjusted to adapt to different types, resolutions, and qualities of images. Exploring the sensitivity of analysis output with respect to parameter adjustments can provide a guideline for the best deployment of algorithms in different scenarios and for rapid development of robust algorithms.

A major challenge to efficient execution of these cases is the vast amount of image data and analysis results. ISBTRC has collected over 700 whole slide images with a goal of expanding to approximately 3500 slides in the next couple of years. State-of-the-art tissue slide scanners are capable of producing high-magnification, high-resolution images from whole slides and tissue microarrays within several minutes. One single whole slide image can contain 10^{10} pixels and 10^6-10^7 biological objects of interest such as nuclei. A brute-force pair-wise comparison between results from two analyses of a single whole slide image could incur a complexity of $O(n^2)$ -- 10^{12} operations, where n is the number of objects, with additional extensive I/O cost for reading the data. An efficient mechanism is needed to manage large volumes of results (along with provenance information). Moreover, the mechanism should support computational and data intensive queries – such as spatial join queries for comparison of image markups associated with different analysis results.

Algorithm validation has been studied for medical imaging [1]. Much work has been done for modeling image analysis results, including the work done by the caBIG Annotation and Image Markup project (AIM) [2] for radiology images, the Open Microscopy Environment project (OME) for cell imaging[3], and DICOM Structural Reporting for human annotations [4]. Biomedical imaging databases have been developed for managing results [3, 5, 6, 7]. However, they are not designed to provide support for data querying and retrieval for algorithm evaluation workflows. The use

of parallel and distributed computing for analysis enables researchers to process image data rapidly and produce large volumes of analysis results. For example, a distributed system has been developed by Yang et al. for computer-aided analysis of digitized breast tissue specimens [8]. Our work differs in that we propose a database-based approach to support algorithm evaluation by systematically managing large-scale results from algorithms and human annotations and supporting efficient queries using a parallel database architecture.

2 Algorithm Evaluation Workflows and Queries

We present an algorithm comparison workflow employed in ISBTRC studies as a motivating example (see Figure 1). The workflow consists of the following steps: 1) Multiple algorithms or algorithms with different parameters are applied to non-overlapping image tiles partitioned from whole slide images. Note that tiling is a common practice for pathology image analysis. Tiling of an image is done either by padding tile boundaries to include objects on boundaries (in most cases, objects, such as cells and nuclei, to be segmented are relatively small), or objects on tile boundaries are discarded when the final analysis result is a statistical aggregation; 2) results and provenance information are represented and managed in a database; 3) the results are queried for comparisons of objects and features extracted by the algorithms -- for example, to find the *overlap-to-intersection ratio* and *centroid distance* of intersected markups; and 4) for algorithm consolidation, analysis results are either filtered or aggregated based on certain criteria, and for algorithm sensitivity study, changes in results are recorded and stored as algorithm parameters are varied. Algorithms are applied to image tiles comprising whole slide images, thus the process of executing algorithms and carrying out result comparisons is highly data and computational intensive. In this paper, we focus on the data management and query aspect, and the workflow execution is left for future discussions.

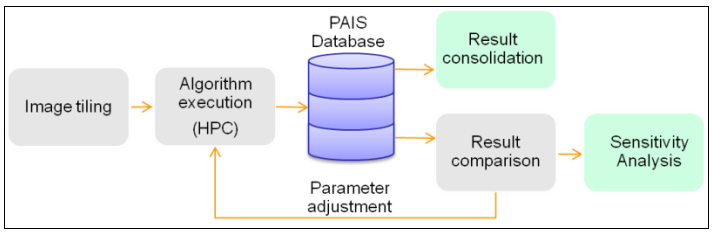

Fig. 1. Algorithm comparison workflow

We should note that this workflow can be modified to support algorithm validation, in which algorithm results are compared with human annotations, as seen in Figure 2. Even a single tile derived from a whole-slide image can contain tens of thousands of nuclei, making it infeasible for human annotators to mark boundaries for all objects in a tile. Thus, the algorithm validation workflow divides tiles into smaller regions (subregions) – e.g., a subregion could be an 8x8 division of a tile. In this way, an image dataset used in validation is organized by three hierarchical spatial concepts in increasing order of granularity: slide, tile, and subregion.

These two workflows involve several common query types. The first type of query is the spatial join query, i.e. spatial operations used to combine two or more datasets with respect to a spatial relationship. There are two steps involved in this query type: spatial filtering based on spatial relationships, such as intersection, and spatial measurements based on computational geometry algorithms, such as area, centroid, distance, and union of polygons. In our case, all segmentation result comparisons are performed through spatial joins by computing overlap-to-intersection ratio and centroid distance of markup pairs represented in polygons. The second type of query is the spatial containment query. In some cases, algorithm-generated results in certain regions are used for validation or comparison purposes. In this case, containing regions are pre-generated for subsequent analysis as a filtering condition, with which nuclei in tumor regions are retrieved by a spatial containment query that only returns markup objects of nuclei contained in said tumor regions. The third type of query involves finding objects contained in subregions and computing the density of those objects. This query is useful in spatial sampling approaches for algorithm validation. For example, a stratified sampling method can use this type of query to select subregions grouped by the densities of objects in those subregions.

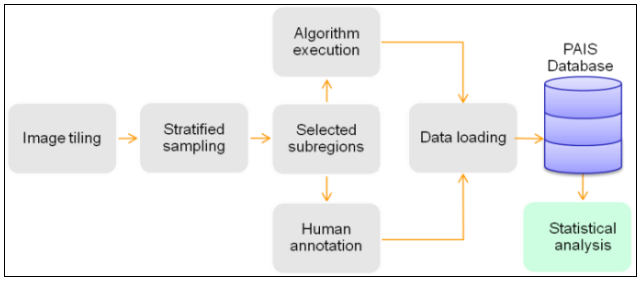

Fig. 2. Algorithm validation workflow

3 Pathology Analytical Imaging Database and Query Support

3.1 PAIS database

To manage algorithm results, human annotations and provenance, and to efficiently support complex queries, we employ a system called PAIS (Pathology Analytical Imaging Standards) [9, 10].

The PAIS data model consists of 62 UML classes, representing information on markups (spatial shapes representing regions, cellular or subcellular objects), annotations (including calculated features or observations such as classifications associated with markups), and provenance (image references, analysis, algorithm and parameters). PAIS provides highly generalized data objects and data types, flexible relationships across objects. Geometric shapes such as polygons are used to represent the boundaries of segmented objects, such as tumor regions, blood vessels, and nuclei. Algorithm results are converted into PAIS format by PAIS DocumentGenerator to be loaded into the database. The highly generalized PAIS model enables the support of different algorithms.

PAIS employs a spatial DBMS based implementation for managing data – we have used IBM DB2 with Spatial Extender[11] in our implementation. There are three major types of tables involved in the PAIS database: i) Spatial tables for representation of markup objects with geometric shapes. Spatial DBMS provides extensions to RDBMS to support spatial data types such as ST_POLYGON and ST_POLYLINE; ii) Feature and observation tables to capture calculated features, such as area, perimeter, and eccentricity, and descriptive observations, such as classifications of regions or nuclei; and iii) Provenance tables to represent image references, subject, specimen, the user performing the analysis, the analysis purpose, and the invoked algorithms. The database implementation provides dozens of functions to support comparison of relationships across spatial objects. Some most commonly used relationship functions include *intersects*, *overlaps*, *within*, *contains*, and *touches*, among others. It also provides numerous spatial measurement functions, including those to compute the area and centroid of a spatial object, to calculate the distance of two spatial objects, and to generate an intersected region (a spatial object generated from two spatial objects). Note that PAIS manages image analysis results – original images are managed in a separate database.

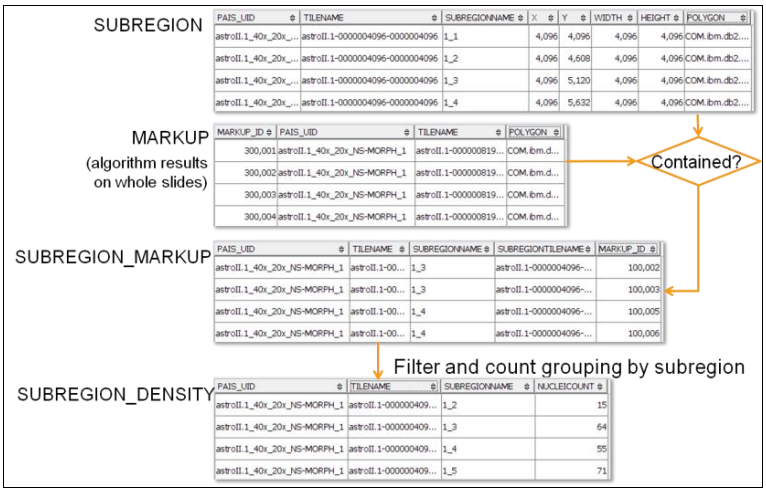

Fig. 3. Workflow for generating density data for sampling

Representing results in a structural format and managing them in a database provide significant advantage for expressing complex queries in SQL query language. Furthermore, extended built-in spatial functions make it easier to support workflows involving spatial operations. For example, as presented in Section 2, object densities in image subregions can be used as criteria for stratified sampling. Figure 3 illustrates database support for performing density computations and stratified sampling.

To compute the object density for subregions, we partition tiles to produce subregions. An initial segmentation of all markups is performed by a segmentation algorithm, and all segmented markups are stored in a MARKUP_POLYGON table [12, 13]. Then subregion size is defined in a way such that each subregion contains a reasonable set of objects that could be annotated by a human within a single short

session, e.g., within 10 minutes. With the size information, each tile of an image space is divided into subregions and captured in SUBREGION table in the database. SUBREGION contains PAIS_UID, TILENAME, SUBREGIONNAME, (X,Y) coordinates of the top-left corner of the subregion, WIDTH, HEIGHT, and a POLYGON to represent the boundary of each subregion. After the production of markups and subregions, markup and subregion containment relationship can be decided using a spatial containment query, as shown in Figure 4. In this query, the conditions specify that markups and regions are from the same tile and same image, and regions contain markups. The result is then inserted into the SUBREGION_MARKUP table. From this table, an aggregation query to compute and to filter the density for each region could be easily specified with a GROUP BY clause for each subregion.

```
INSERT INTO PAIS.SUBREGION_MARKUP
SELECT  r.pais_uid, r.tilename, r.subregionname, m.markup_id
FROM    PAIS.MARKUP_POLYGON m, VALIDATION.SUBREGION r
WHERE   m.pais_uid = r.pais_uid AND m.tilename = r.tilename AND
        r.pais_uid = 'gbm1.1_40x_20x_NS-MORPH_1' AND
        DB2GSE.ST_Contains(r.polygon, m.polygon) = 1;
```

Fig. 4. Query to generate subregion and markup containment relationships

3.2 Parallel Database Architecture

A major bottleneck for many database management systems is the I/O bottleneck. An evaluation of spatial queries in our studies has shown that most of the query execution time is spent in I/O and high I/O overhead is incurred, when executing aggregation or spatial join queries. In this section we describe the parallel database approach we have employed to address this issue.

I/O bandwidth can be increased through data partitioning and parallel data access. In our work, we use a shared-nothing parallel database architecture to manage and query PAIS data to provide scalable data management and to speed up complex queries. Figure 5 illustrates this architecture in which multiple database instances are run on separate physical nodes. We also refer to these nodes as database partitions (Partition 1, Partition 2, …, and Partition N). Each node has its own disks, CPU and memory, and the partitions are connected through a fast switched network. There is one master node and multiple slave nodes. The master node accepts queries from users, translates and parallelizes queries across all nodes, and aggregates the results – this enables the simplicity and expressiveness of SQL queries as such distributed parallel query execution is transparent to users. Our implementation employs IBM DB2's parallel query execution support [14]. This support provides a single logical view of partitioned data so that clients can compose SQL queries as if they interact with a serial database with no data partitioning.

One major challenge for a shared-nothing architecture is load balancing, i.e., how to distribute data and computational load evenly across all partitions. A skewed data distribution can lead to bottlenecks and deteriorate overall performance. We have implemented two different approaches for partitioning data: partitioning based on images and partitioning based on tiles. The first approach distributes all of the

segmentation and feature results obtained from a whole image onto the same node, and the second approach distributes all results from a tile onto the same node. The second approach has the benefit of speeding up queries on a single image, as tiles could be queried simultaneously on multiple nodes.

Whether it is applied on tiles or images, the partitioning algorithm tries to minimize data communication overhead across partitions for efficient spatial joins between different result sets. Given multiple algorithm results from a list of tiles, the algorithm distributes the results across multiple nodes such that 1) the amount of results on each node is the same across all the nodes and 2) results from different algorithms on the same tile are assigned to the same node. The algorithm reads the list of tiles and the sizes of result sets associated with these tiles for each analysis run. It then executes a bin-packing heuristic, where each node represents a bin, to distribute the tiles across nodes as follows. The algorithm sorts the tiles in descending order of the total result size of each tile. Starting from the top-most tile, it assigns the tile to the node with the minimum results set size and increments the results set size of the node by the size of the tile's results set. The algorithm proceeds down the sorted list, assigning the current tile to the node with the minimum results set size and incrementing the node's results set size by the tile's results set size.

In our implementation of the parallel PAIS database the partitioning algorithm is invoked to generate a HashMap to store the (*title*, *partition_key*) pairs. When data is loaded to the parallel database, the partition_key is read from the HashMap for each data insert, and the database automatically distributes the data element to the right partition when the insert is executed. Each database partition is stored on the local disk attached to the corresponding node.

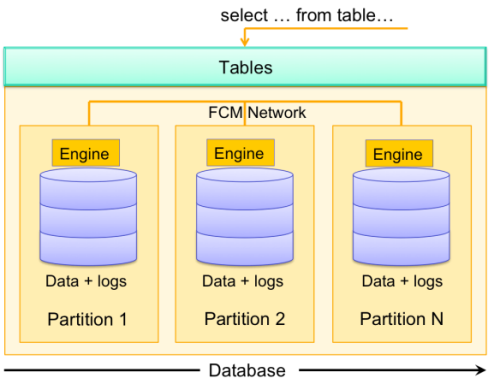

Fig. 5. Partitioning based parallel database architecture

In order to further reduce I/O overhead, each partition creates a spatial index on local data. There are a number of indexing and data access methods [15] to support efficient spatial queries. These methods can be categorized into two main classes: space based and data based. Grid indexing is a common space based approach, where space is partitioned into fixed grid cells. R-Tree indexing [16], on the other hand, is a common data based approach. In our work, grid based indexing is used, as shown in Figure 6. Each spatial region is divided into multi-level grids and indexed. These

grids can be used to efficiently identify spatial intersections of markups in spatial joins. For example, in order to find the intersected markups of the green markup in grid cell (5,2) based on the minimal bounding box of the markup, the grid cell number is generated. Then all the markups of the other dataset (in red color) that intersect this cell are retrieved based on the grid index. These markups are further filtered by geometric computation. Thus, the total number of markups to be compared is significantly reduced via the grid based indexing.

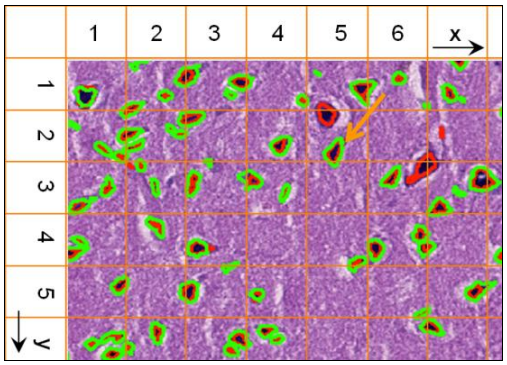

Fig. 6. Fixed grid indexing to support spatial join queries

4 Experimental Performance Evaluation

We have performed experiments with different partitions using IBM DB2 as the underlying relational database management system. We used a cluster machine with five nodes. Each node has two quad-core CPUs and 16GB memory (only a single core is used for each database instance) and runs 64-bit CentOS 5.6. We use IBM Infosphere Data Warehouse edition Vesion 9.7.3 with DB2 partitioning feature as our database engine. We have DB2 setup with five partitions on five cluster nodes. The dataset includes 18 whole-slide microscopy images analyzed by two methods. The total number of spatial objects is about 18 million and the average number of spatial objects per image is 0.5 million. Analysis results from different runs are stored in files. We use file size as results set size for the partitioning algorithm in order to avoid the expensive process of parsing the file for spatial objects twice – once for counting the objects and once for loading the objects to the database.

We report the results of two experiments in Figures 7 and 8: one experiment looks at the distribution of data across backend nodes, the other measures the reduction in execution time of spatial joins. The spatial joins in our experiments involve comparison of overlap-to-intersection ratios and centroid distances between markups from two result sets generated for a single whole slide image. Figure 7 shows the object distribution across five nodes. As is seen in the figure, while partitioning is estimated on file sizes, the objects are almost evenly distributed. The results indicate that the bin-packing scheme works well and the file size provides a good approximation of number of spatial objects. Figure 8 illustrates spatial join query speed up from a single node setup to a five-node setup for an example image with 2 million markup objects. The results show that linear speedup is achieved with our

partitioning approach. The partitioning algorithm allows the DB2's parallel query support to take advantage of partitioned data for higher I/O bandwidth. In addition, indices set up on each node reduce the local I/O overhead on each node.

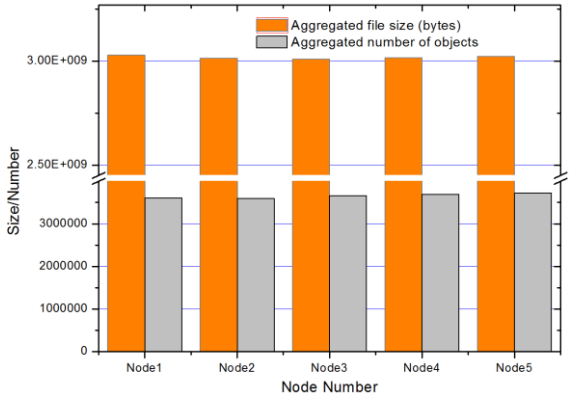

Fig. **7**. Performance of data partitioning with five computation nodes

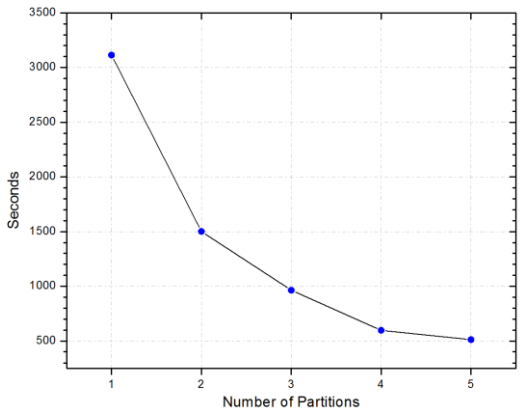

Fig. 8. Performance of spatial join queries for comparing two analysis results for an example image

5 Conclusions

Pathology image algorithm validation and comparison is essential to iterative algorithm development and refinement. The critical component for this is to support efficient spatial join queries between results. In our work, we develop an efficient data model and parallel database approach to model, manage and query large volumes of analytical image result data. Our experiments demonstrate that a bin-packing based data partitioning strategy combined with grid-based indices result in good data distribution across database nodes and reduce I/O overhead in spatial join queries through parallel retrieval of relevant data and quick subsetting of datasets.

Acknowledgement. This research is supported in part by PHS Grant UL1RR025008 from the CTSA program, by R24HL085343 from the NHLBI, by Grant Numbers 1R01LM011119-01 and R01LM009239 from the NLM, by NCI Contract No. N01-CO-12400 and 94995NBS23 and HHSN261200800001E, by NSF CNS 0615155, 79077CBS10, and CNS-0403342, and P20 EB000591 by the BISTI program. IBM provides academic license for DB2. Susan Malaika from IBM provided many insightful suggestions.

References

1. Clunie, D.: Algorithm Validation Toolkit (AVT), Presentation at the 2008 Radiological Society of North America Annual Meeting. http://www.dclunie.com/papers/AVT-RSNA-2008-RC730-DAC2.pdf (2008)
2. Channin., D., Mongkolwat, P., Kleper, V., Sepukar, K., and Rubin, D.: The caBIG Annotation and Image Markup Project. J. Digit. Imaging 23(2):217-25 (2009)
3. Goldberg, I., Allan, C., Burel, J.-M., Creager, D., Falconi, A., Hochheiser, H., et al.: The Open Microscopy Environment (OME) Data Model and XML File: Open Tools for Informatics and Quantitative Analysis in Biological Imaging. Genome Biol., 6(R47), 2005.
4. Clunie., D.: DICOM Structured Reporting and Cancer Clinical Trials Results. Cancer Inform, 4, 33-56 (2007)
5. Kvilekval, K., D. Fedorov, B. Obara, A. Singh, and B.S. Manjunath, Bisque: A Platform for Bioimage Analysis and Management. Bioinformatics 26(4), 544--552 (2010)
6. Martone, M.E., S. Zhang, A. Gupta, X. Qian, H. He, D.L. Price, et al., The Cell-centered Database: A Database for Multiscale Structural and Protein Localization Data from Light and Electron Microscopy. Neuroinformatics, 1(4), 379--395 (2003)
7. Yang, L., Chen, W., Meer, P., Feldman, M., Goodell, L., Bagg, A., He, W., Foran, D.: PathMiner: A Web-Based Tool for Computer Assisted Diagnostics in Pathology. IEEE Trans. on Inform. Tech. in Biomedicine, 13(3), pp. 291--299 (2009).
8. Yang, L., W. Chen, P. Meer, G. Salaru, M.D. Feldman, and D. Foran: High-throughput Analysis of Breast Cancer Specimens on the Grid. In: 10th International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 617-25, Springer-Verlag (2007)
9. Pathology Analytical Imaging Informatics Standards Wiki. https://web.cci.emory.edu/confluence/display/PAIS
10. Wang, F., Kong, J., Lee, Cooper, et al: Data Model and Database for High-resolution Pathology Analytical Image Informatics. J. Pathol. Inform. 2:32 (2011)
11. DB2 Spatial. http://www.ibm.com/software/data/spatial/db2spatial/
12. Cooper, L., Kong, J., Gutman, D. A, Wang, F, et al.: An Integrative Approach for In Silico Glioma Research. IEEE Trans. on Biomed. Eng., 57(10), 2617--2621 (2010)
13. Kong, J. , Cooper, L., Wang, F., Chisolm, C., Moreno, C., Kurc, T., Widener, P., Brat, D., Saltz, J.: A Comprehensive Framework for Classification of Nuclei in Digital Microscopy Imaging: An Application to Diffuse Gliomas. In: IEEE International Symposium on Biomedical Imaging, 2128—2131. IEEE Press, New York (2011)
14. Baru., C. and Fecteau, G.: An Overview of DB2 Parallel Edition. In: ACM SIGMOD, pp. 460--462. ACM, New York (1995)
15. V. Gaede and O. Gunther: Multidimensional Access Methods. ACM Computing Surveys, 30(2) (1998)
16. Guttman., A: R-trees: A Dynamic Index Structure for Spatial Searching. In: SIGMOD Conference, pp 47--57. ACM, New York (1984)