Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 827–832

DOI: 10.15439/2015F272
ACSIS, Vol. 5

The Scalable Distributed Two-layer Content Based Image Retrieval Data Store

Stanisław Deniziak
Kielce University of Technology
al. Tysiaclecia Panstwa Polskiego 7,
25-314 Kielce, Poland
Email: s.deniziak@tu.kielce.pl

Tomasz Michno, Adam Krechowicz
Kielce University of Technology
al. Tysiaclecia Panstwa Polskiego 7,
25-314 Kielce, Poland
Email: {t.michno, a.krechowicz}@tu.kielce.pl

***Abstract*—The multimedia databases are becoming more and more popular nowadays. One of their main problem is a huge data amount storage. Another problem with multimedia databases is querying. Traditional approaches, based on textual keywords are not sufficient. More advanced techniques, incorporating image content features, should be used. In this paper we propose new multimedia database structure with ability of Content Based Image Retrieval which is based on our previous work: Query by Shape method (QS). Query by Shape is a method which is based on decomposing an object into features. Each feature may consists of shape primitive, a color or a texture. In this paper we only use shape primitives. In order to achieve high scalability and workload control, we propose a modified Scalable Distributed Two-layer Data Structure, as a storage. The modification incorporates adding tree structure, comparing algorithm and returning a set of results to the client.**

I. Introduction

THE MULTIMEDIA databases are becoming more and more popular nowadays. There are many applications where they are needed, like social media portals (e.g. Facebook, Instagram, Flickr and Google+) or monitoring systems. Because modern cameras produce high resolution images, the amount of data which has to be stored is very huge. Another problem with multimedia databases is their querying. Traditional approaches, based on textual keywords are not sufficient. More advanced techniques incorporating image content features should be used.

In this paper we propose the idea of a multimedia database structure with ability of Content Based Image Retrieval which is based on our previous work described in [1], Query by Shape (QS). Query by Shape is a method which is based on decomposing an object into features [1]. Each feature may consists of shape primitive, a color or a texture. In this paper we only use shape primitives. As a data structure for storage we use a modified Scalable Distributed Two-layer Data Structure which is highly scalable, distributed data store [2]. The modification incorporates adding tree structure, comparing algorithm and returning a set of results to the client.

This paper is organized as follows. The Section II presents the survey of image retrieval algorithms. The Section III contains a short review of NoSQL data stores. The idea of Scalable Distributed Two-layer Data Structures is described in the Section IV. The Section V shows the motivation of our research. The idea of our database structure is presented in the Section VI. The conclusion of the research is given in the Section VII.

II. Image Retrieval algorithms

In the area of multimedia databases, three types of retrieval algorithms can be distinguished: Keyword-Based Image Retrieval (KBIR), Content-Based Image Retrieval (CBIR) and Semantic-Based Image Retreival(SBIR).

The first group, KBIR, is based on the relational database approach, where images are stored in the database and they are described using keywords. During the query, the proper keywords should be given. The database structure is very simple but strongly relies on the textual annotations given by a human. This approach is prone to mistakes because of the subjective kind of descriptions [1]. Moreover it is very hard to cover the whole information, present in the image, using textual description [3], [4].

The CBIR algorithms are based on different approach than KBIR. Image features are used to index images and perform queries [1]. All algorithms in this group could be divided into two categories: low-level and high-level algorithms. The low-level algorithms process images globally, extracting features from the whole frame, using e.g. a normalized color histogram [5], a spatial domain [6], a difference moment and entropy [7] or an MPEG-7 image descriptors like shape and texture [8]. The low-level features used by CBIR algorithms are easy to compute but they are insufficient if the query is oriented on searching for similar objects rather than whole images, which incorporates separating the object from the background.

The high-level CBIR algorithms provide more reliable and precise results in this situation. The major part of the algorithms from this group are based on the regions extraction and graphs matching. A region is a group of similar pixels, most often grouped by colors. There are also methods which uses during region extraction: a set of primitives [9] or fuzzy pattern [10] detection, moment-based local operators [11] or parallelograms, ellipses, corners and arcs detection [12]. After region extraction, a graph is being constructed in order to store the relations between them [1]. During the multimedia database query, the graph-subgraph matching is performed

978-83-60810-66-8/$25.00

e.g. using the classic Ullman algorithm [13] or more advanced ones. There are algorithms that are automatic or semi-automatic with ability to present preliminary results to the user who may choose important regions and repeat the query [14]. Another group of CBIR algorithms allows queries without full knowledge about searched images or objects. One of the first and most successive approaches in this area uses a human drawn sketches which are compared with the corresponding sketches in the database, globally for the whole image [15]. Another example is Query by Shape method [1], which is our previous research and which is used as a base of this paper. The idea of the algorithm is to decompose objects into features like shape primitives or color features. The features are not used for region extraction but for constructing an object's sketch or skeleton which is strictly a graph. Also the matching algorithm is proposed which uses the $similarity$ coefficient. The $similarity$ informs how similar two graphs are. If they are the same, the $similarity$ is equal to 1, if they are completely different, it takes 0 value. All intermediate values indicates that graphs are partially similar.

The SBIR algorithms are algorithms that try to overcome the "semantic gap", which is the difference between what is present on the image and what a human could interpret [4], [16]. Most SBIR algorithms are based on textual description which, in contrast to the KBIR algorithms, is a much longer phrase. The phrase is easy to create, use and understand by the human, the example could be "the sunrise in the mountains". The textual descriptions are not used directly, but they are mapped onto semantic features and then a query is performed [17].

All groups of algorithms need efficient data storage methods. One of the most often used structure is a cell or a tree, because it can store the relations between similar images [8]. There are also approaches that joins both data structures. One of the example is [18] which is based on gathering similar records in the same cells. Additionally, some biological processes are added, like mitosis, when the similarity between items in the same cells is below the specified level.

III. The NoSQL data stores

The multimedia databases very often stores millions of photos or images. Because there has to be stored and processed very huge amount of data, with high availability and workload management, the traditional SQL-based databases may not be sufficient. Much better results are obtained using NoSQL databases. The NoSQL databases may be classified into the following groups: Graph databases, Key-Value data stores, Document data stores and Column-based data stores [19].

The Graph databases provides similar features as relational databases but they are not well-prepared to store huge amount of data because of the problems with scalability [19]. Key-Value data stores use an unique single key (e.g. a number) for storing data (value). Most often they have only two operations: inserting and retrieving data [20]. The examples of such data stores are Dynamo data store by Amazon which uses single integer key [20] or Apache Hbase which identifies the data by a timestamp, column name and row name [21]. The Document data stores use textual data interchange formats, like JSON or xml to store data. The most popular document data store is MongoDB [19].

The NoSQL data stores were successfully adapted to store images. As an example the Apache Hbase may be given, which was used to store Google Earth images [21].

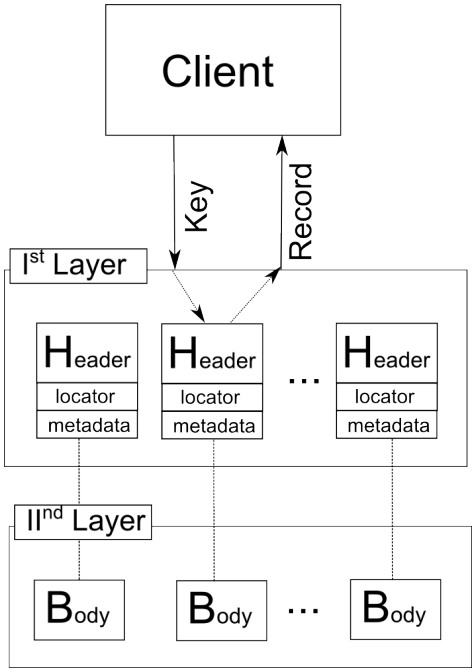

Fig. 1. The SD2DS structure overview.

IV. The SD2DS data stores

Scalable Distributed Two-layer Data Structures (SD2DS) are one of variants of Scalable Distributed Data Structures (SDDS) [2], which may be classified as an distributed NoSQL data store. They stores data using a key and a value in so-called "buckets". Each bucket has storage limit and when it is reached a split is performed. Buckets may be distributed through many machines, e.g. nodes on the multicomputer. The SD2DS stores data in two separate layers. The first layer contains only data headers which consists of the data locator and metadata. The data locator is used to locate the stored data in the second layer. The metadata is an additional information connected with the actual data and may contain optional information e.g. the data length, a checksum, insertion date or data description. The second layer contains only the actual data, called body [2]. That structure maintains data more efficiently, because both layers are independent of each others and may be stored on different machines. Moreover the data store is highly scalable, without theoretical limitations and may contain as many data as is needed. The SD2DS allows also to use as a second layer other solutions, even without buckets. The SD2DS structure is presented in the Fig. 1.

The SD2DS may be easily extended e.g. by adding throughput adjustment [22] which highly improves the data access time.

V. Motivation

The problem of multimedia database querying is very complicated. As a continuation of the previous research [1], we would like to propose the Content Based Image Retrieval Database incorporating Query by Shape Method. The database has to store a very huge amount of data. Simple records table, a tree or a cell would be not sufficient. Because of that, more advanced data stores, especially NoSQL-based should be used. In order to obtain very high scalability and availability, as a base for our system the Scalable Distributed Two-Layer Data Structures should be used. As a result of our research we would like to create the system which will fulfil the following requirements:

- queries using a graph of features e.g. primitives,
- similarity control for graphs comparison, the similarity should be set by the client,
- client feature: graphical queries easily drawn by a human (without drawing skill), e.g. using predefined shapes,
- client feature: automatic image transformation into a graph, used for a query,
- very fast data access,
- scalability without turning off the system,
- easy addition of the new hardware used for storage,
- good performance for the very huge data workload stored in the database.

VI. The system overview

The proposed system extends the SD2DS data store structure. The classical approach consists of two layers with the first layer containing record's headers, the second with their bodies and the client which uses single key to retrieve the single record. In order to provide features mentioned in the previous section, some modifications have to be made. The system overview is shown in the Figure 2.

A. The Client

The client should query the data store by a graph, instead of a standard key. Moreover it has to be able to receive the result of a query which may consists of many records. Therefore, the following messages are defined:
Messages send by the client:

- record (graph and image) insertion (GI) - a message used to insert a new record containing a graph and an image into the database,
- query (SQ) - a query after which all result records are sent by the database in a one message,
- distributed query (DQ) -a query after which result records are sent gradually with one message per one record, allowing e.g. to show progressively results for a user,
- get record (GR) - a message used to retrieve the record using its key.

Messages received by the client:

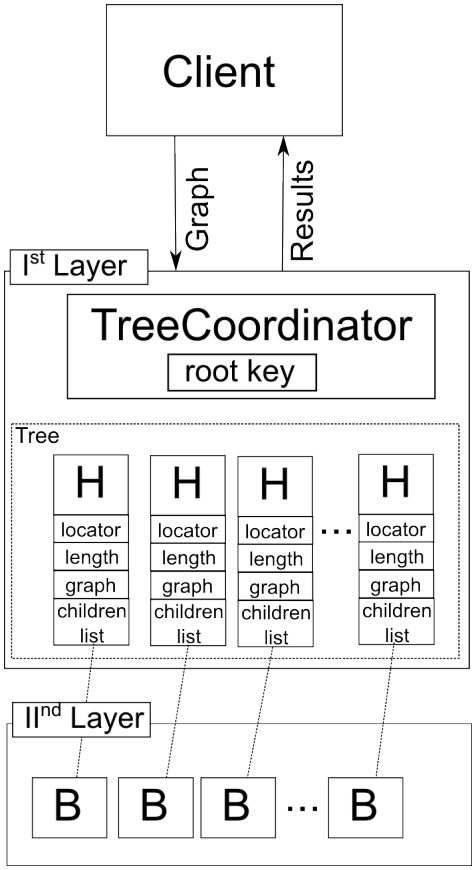

Fig. 2. The proposed system overview.

- result of the insertion (RI) - sent by the database as a response to GI message, returning the failure code or new node's record key,
- not found (NF) - sent by the data store when there are no similar graphs/images in the data store, returned for both SQ and DQ messages,
- results (RR) - contains list of similar records, returned only as a response to SQ message,
- single result (SR) - contains only one similar record, returned only as a response to DQ or GR messages.

All sent messages contain the query graph and the minimal similarity which is used during comparisons. Moreover the messages may define if as a result full record or only a header should be returned.

B. The First Layer

The first layer of the data store was redesigned into two sub-layers: the Tree Coordinator and the Tree. The Tree sub-layer consists of the standard SD2DS record headers. Each record header is treated as a one tree node, containing as a metadata a graph and a list of children nodes (a list of records keys). The Tree Coordinator is responsible for communicating with the clients and the logic of the tree: storing the tree root record key, adding new nodes, traversing tree and making queries. Moreover it is able to execute queries in two ways,

analogously to the SQ and DQ messages. In order to improve the performance, some of the Tree Coordinator features, like comparing graphs and tree traversing may be implemented as a part of buckets. Also the DQ message could be directly sent by the bucket.

1) The tree structure: The tree nodes used in the database are stored in the first layer using standard SD2DS headers records. The graphs are inserted into the tree hierarchically, storing its common parts in parent nodes. Because of that, the root node may contain a graph with only one shape or an empty graph. The example tree structure with scooter, bicycle and car objects is shown in the Fig. 3. Moreover, because there are indirect nodes which stores only graphs without connection to any images, the SD2DS modification allows records headers in the first layer with empty bodies.

2) Record Insertion: after receiving the GI message the Tree Coordinator compares the graph with graphs stored in the tree, starting from the root node using the Query by Shape comparison algorithm (QS) [1] and the minimal similarity parameter extracted from the GI message. The algorithm is presented in the Alg. 1. The Tree Coordinator retrieves the record using similar procedures as the client in the standard SD2DS. Then children nodes are extracted from the record and used for the next records retrieval.

3) Querying: The querying algorithm is very similar to the insertion of a node. The algorithm is shown in the Alg. 2. During the query, the results could be sent immediately to the client or stored in the Tree Coordinator temporary buffer before completion and then sent.

C. The Second Layer

The second layer has not been modified. Because of the tree structure, there may be less bodies than records in the first layer. The images are retrieved during querying, but the client in the SQ or DQ could force the database to only send headers from the first layer or even only the keys.

VII. Experimental results

The presented tree-based structure has been evaluated using experimental implementation written in C++. The results were compared with the linear SD2DS QS system and are shown in the Tables I and II. The linear SD2DS QS system uses also first SD2DS layer to store graphs, but without tree structure. During the query, all elements has to be compared with the query graph. In order to evaluate algorithms performance, the precision and recall coefficients were used [1]:

$$precision = \frac{number\ of\ relevant\ results\ images}{total\ number\ of\ results\ images} \quad (1)$$

$$recall = \frac{number\ of\ relevant\ results\ images}{total\ number\ of\ relevant\ images\ in\ the\ database} \quad (2)$$

As a number of relevant results images we assume the number of images which are from the same class as the query image. For the tests, about 117 real life images of different objects classes were used.

Algorithm 1 Inserting new node to the tree

Require: $graph$ and $image$ extracted from the GI message
  if tree is empty **then**
    add record as a root node, store its key as a Root Key;
    send RI message with the key to the client;
  else
    add Root Key to the FIFO queue;
    while FIFO queue is not empty **do**
      $key \leftarrow$ pop first element from FIFO;
      $treeNode \leftarrow$ get record with key == key;
      compare $graph$ with $treeNode$ using QS;
      if $treeNode$ is not a root node **then**
        if graphs similarity $\leq$ similarity of $graph$ and $treeNode$'s parent **then**
          add $graph$ as a child of $treeNode$'s parent;
          send RI message with the $graphs$'s key to the client;
        end if
      else
        add $graph$ as a new root node;
        add $treeNode$ to the $graph$'s children list

      end if
      if graphs similarity $\geq$ highSimilarity **then**
        add record as a child to $treeNode$;
        send RI message with the record's key to the client;
      else
        if $treeNode$ does not have any children **then**
          add $graph$ to the tree;
          add the common part of $graph$ and $treeNode$ as their parents;
          if $treeNode$ is a root node **then**
            the Root Key $\leftarrow$ common part's key
          end if
          send RI message with the $graph$'s key to the client;
        else
          add each $treeNode$ children to the FIFO queue;
        end if
      end if
    end while
  end if

The test results shows that for most queries the tree structure increased the precision comparing to the previous, linear structure. This is due to the additional steps during the query algorithm (Alg. 2) which omits the whole sub-tree with too low precision and check if the precision is increasing in children nodes. Moreover, the lower precision in the automated queries was caused by some failures of shape detection algorithms.

The recall measurements shows that for the manual queries more relevant results were returned by the tree algorithm version. The automatic query was problematic for same cases for tree version because of occurrence of many unconnected

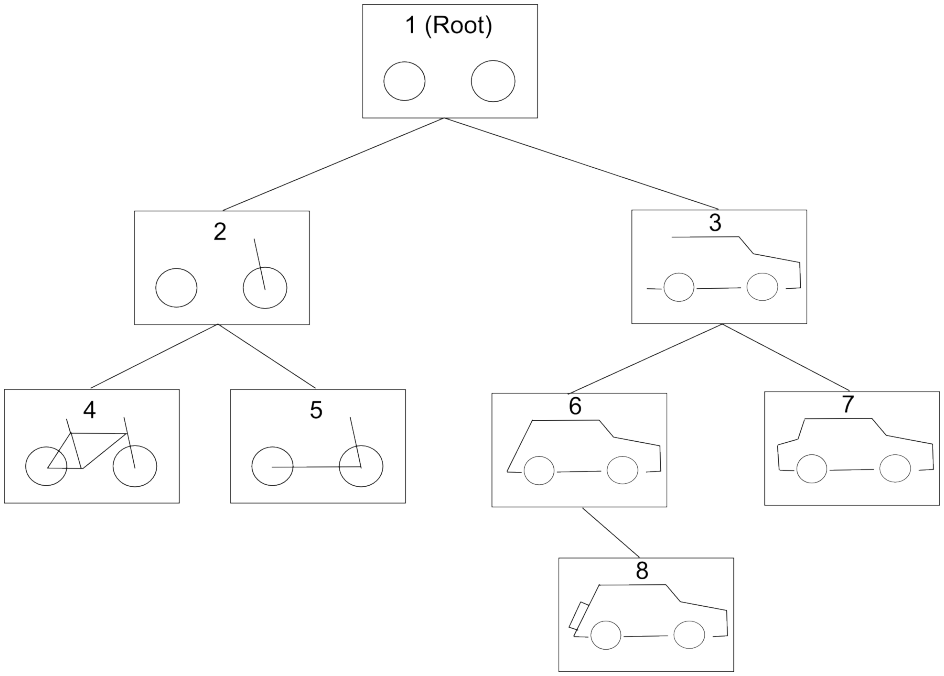

Fig. 3. The example of the tree with bicycle (node 4), scooter (node 5) and 3 cars graphs (nodes 6-8). The shape sizes were omitted during tree construction in order to show the main idea and simplify the structure.

TABLE I
THE PRECISION RESULTS FOR THE EXAMPLE BICYCLE AND CAR QUERIES

	Manual queries		**Automated queries**	
	Tree-based	**Linear**	**Tree-based**	**Linear**
bicycle no. 1	0.7708	0.6065	1	0.9
bicycle no. 2	0.8529	0.5714	1	0.8
car no. 1	1	0.7059	0.4182	0.5658
car no. 2	1	0.8	0.5902	0.5428

TABLE II
THE RECALL RESULTS FOR THE EXAMPLE BICYCLE AND CAR QUERIES

	Manual queries		**Automated queries**	
	Tree-based	**Linear**	**Tree-based**	**Linear**
bicycle no. 1	0.9737	0.9737	0.3023	0.2093
bicycle no. 2	0.7636	0.8421	0.2558	0.0930
car no. 1	0.5814	0.5581	0.5	0.86
car no. 2	0.7209	0.4651	0.72	0.76

nodes in skeletons. This caused problems with inserting them in the proper sub-tree.

The performance of the structure is dependent on the number of elements. During our experiments using SD2DS, to up to 1000 elements there were very small difference between tree and linear version. For higher number of elements, the tree structure was becoming more and more faster. We also implemented version which uses as a storage vector array. During experiments the tree version occurred to be up to 2x faster than linear version.

VIII. CONCLUSION

In this paper a new Content Based Image Retrieval database structure was presented. The main idea of our research is to apply our previous research results, Query by Shape method [1], into Scalable Distributed Two-layer Data Structure. The modification of data store included redesigning it to store records in a tree. Also a Tree Coordinator was added to the first layer in order to communicate with the client and perform tree queries.

The future research includes implementing the version of the algorithm with comparison algorithm moved to buckets. We expect that it should improve the querying time as well as allows to use buckets and nodes more efficiently. Moreover the throughput adjustment may be added in order to improve the image data access time.

Another direction of research will concern the Query by Shape method in order to improve the results precision. This may include e.g. adding other primitive types and color features. Moreover more advanced graph matching algorithms should be applied, as well as some optimization methods should be evaluated, e.g. [23].

ACKNOWLEDGMENT

The research used equipment funded by the European Union in the Innovative Economy Programme, MOLAB - Kielce University of Technology.

REFERENCES

[1] S. Deniziak and T. Michno, "Query by shape for image retrieval from multimedia databases," in *Beyond Databases, Architectures and Structures*, ser. Communications in Computer and Information Science, S. Kozielski, D. Mrozek, P. Kasprowski, B. Malysiak-Mrozek, and D. Kostrzewa, Eds. Springer International Publishing, 2015, vol. 521, pp. 377–386. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-18422-7_33

Algorithm 2 Querying the database with a graph

Require: $graph$ extracted from the SQ or DQ query message
if tree is empty **then**
  send NF message to the client;
else
  add Root Key to the FIFO queue;
  while FIFO queue is not empty **do**
    $key \leftarrow$ pop first element from FIFO;
    $treeNode \leftarrow$ get record with key == key;
    compare $graph$ with $treeNode$ using QS;
    if graphs similarity $\geq$ highSimilarity **then**
      if the client sent DQ message **then**
        send RI message with the record's key to the client;
        send RI message with all keys of the sub-tree consisting of record's children to the client;
      else
        add record to the results;
        add all keys of the sub-tree consisting of record's children to the results;
      end if
    end if
    if $treeNode$'s parent similarity $- minSimilarity >$ graphs similarity **then**
      continue;
    end if
    if $treeNode$ has children **then**
      add each $treeNode$ children to the FIFO queue;
    end if
  end while
end if

[2] K. Sapiecha and G. Lukawski, "Scalable distributed two-layer data structures (sd2ds)," *IJDST*, vol. 4, no. 2, pp. 15–30, 2013. [Online]. Available: http://dx.doi.org/10.4018/jdst.2013040102

[3] C.-Y. Li and C.-T. Hsu, "Image retrieval with relevance feedback based on graph-theoretic region correspondence estimation," *IEEE Transactions on Multimedia*, vol. 10, no. 3, pp. 447–456, April 2008.

[4] H. H. Wang, D. Mohamad, and N. A. Ismail, "Approaches, challenges and future direction of image retrieval," *CoRR*, vol. abs/1006.4568, 2010.

[5] M. Mocofan, I. Ermalai, M. Bucos, M. Onita, and B. Dragulescu, "Supervised tree content based search algorithm for multimedia image databases," in *6th IEEE International Symposium on Applied Computational Intelligence and Informatics*, May 2011, pp. 469–472.

[6] T. K. Shih, "Distributed multimedia databases," T. K. Shih, Ed. Hershey, PA, USA: IGI Global, 2002, ch. Distributed Multimedia Databases, pp. 2–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=510695.510697

[7] H.-P. Kriegel, P. Kroger, P. Kunath, and A. Pryakhin, "Effective similarity search in multimedia databases using multiple representations," in *12th International Multi-Media Modelling Conference Proceedings*, 2006, pp. 4 pp.–.

[8] C. Lalos, A. Doulamis, K. Konstanteli, P. Dellias, and T. Varvarigou, "An innovative content-based indexing technique with linear response suitable for pervasive environments," in *International Workshop on Content-Based Multimedia Indexing*, June 2008, pp. 462–469.

[9] R. Jakubowski, "Extraction of shape features for syntactic recognition of mechanical parts," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-15, no. 5, pp. 642–651, Sept 1985.

[10] M. Bielecka and M. Skomorowski, "Fuzzy-aided parsing for pattern recognition," in *Computer Recognition Systems 2*, ser. Advances in Soft Computing, M. Kurzynski, E. Puchala, M. Wozniak, and A. Zolnierek, Eds. Springer Berlin Heidelberg, 2007, vol. 45, pp. 313–318.

[11] A. Sluzek, "On moment-based local operators for detecting image patterns," *Image and Vision Computing*, vol. 23, no. 3, pp. 287 – 298, 2005. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2004.03.003

[12] H.-C. Lee and K.-S. Fu, "Generating object descriptions for model retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, no. 5, pp. 462–471, Sept 1983.

[13] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976. [Online]. Available: http://doi.acm.org/10.1145/321921.321925

[14] G. Aggarwal, T. Ashwin, and S. Ghosal, "An image retrieval system with automatic query modification," *IEEE Transactions on Multimedia*, vol. 4, no. 2, pp. 201–214, Jun 2002.

[15] T. Kato, T. Kurita, N. Otsu, and K. Hirata, "A sketch retrieval method for full color image database-query by visual example," in *11th IAPR International Conference on Pattern Recognition, Vol.I. Conference A: Computer Vision and Applications*, Aug 1992, pp. 530–533.

[16] A. Singh, S. Shekhar, and A. Jalal, "Semantic based image retrieval using multi-agent model by searching and filtering replicated web images," in *Information and Communication Technologies (WICT), 2012 World Congress on*, Oct 2012, pp. 817–821.

[17] C.-Y. Li and C.-T. Hsu, "Image retrieval with relevance feedback based on graph-theoretic region correspondence estimation," *Multimedia, IEEE Transactions on*, vol. 10, no. 3, pp. 447–456, April 2008.

[18] S. Kiranyaz and M. Gabbouj, "Hierarchical cellular tree: An efficient indexing scheme for content-based retrieval on multimedia databases," *Multimedia, IEEE Transactions on*, vol. 9, no. 1, pp. 102–119, Jan 2007.

[19] P. J. Sadalage and M. Fowler, *NoSQL distilled : a brief guide to the emerging world of polyglot persistence*. Upper Saddle River, NJ: Addison-Wesley, 2013. [Online]. Available: http://opac.inria.fr/record=b1135051

[20] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, Oct. 2007. [Online]. Available: http://doi.acm.org/10.1145/1323293.1294281

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 4:1–4:26, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1365815.1365816

[22] K. Sapiecha, G. Lukawski, and A. Krechowicz, "Enhancing throughput of scalable distributed two – layer data structures," in *Parallel and Distributed Computing (ISPDC), 2014 IEEE 13th International Symposium on*, June 2014, pp. 103–110.

[23] P. Sitek and J. Wikarek, "A hybrid framework for the modelling and optimisation of decision problems in sustainable supply chain management," *International Journal of Production Research*, 2015.