Distributed image retrieval with colour and keypoint features

 Michał Łągiewka, Marcin Korytkowski and Rafal Scherer
 Computer Vision and Data Mining Lab, Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland

ABSTRACT
Content-based image retrieval poses many problems to computer systems. The content of images has to be described by some feature extraction methods. As image databases are often very large, they are sometimes to complex to be processed by traditional computing methods. We have to use big data solutions to fast retrieve images. The paper presents a system for retrieving images in relational databases in a distributed environment. Content of the query image and images in the database is compared using global colour information and local image keypoints. Image keypoint descriptors are indexed by fuzzy sets directly in a relational database by our algorithm. The process is distributed to several machines thanks to the Apache Hadoop software framework with HDFS.

1. Introduction
Content-based image retrieval (CBIR) has become recently well established in the literature. Yet, nearly all of the solutions presented so far are not designed nor suited for relational databases. Relational databases reign supreme in the business world but storing a huge amount of undefined and unstructured binary data and its fast and efficient search and retrieval is a problem for them. Examples of such data are images or video files. One, old solution devoted to storage and retrieval of images in a database is the methodology proposed in Ogle and Stonebraker (1995), where PostgreSQL database server was used to store and compare images by colour-based features. There were also attempts to implement CBIR in commercial database systems. An example might be the Oracle database environment called ‘interMedia’, where image retrieval is based on the global colour representation, low-level patterns and textures within the image, such as graininess or smoothness, the shapes that appear in the image created by a region of uniform colour and their location. It was described in Oracle Database Online Documentation (10 g Release 2, Chapter 6, Content-Based Retrieval Concepts) but was abandoned in newer Oracle versions. Thus, the standard SQL language does not have commands for handling multimedia data and image files are stored often directly in database tables which causes low efficiency of the whole system and even time-consuming data backup. To address...
these problems, the authors proposed earlier (Korytkowski, 2017; Korytkowski, Rutkowski, & Scherer, 2016) a CBIR system that was able to store and index images in a relational database with local interest points. Local invariant features have gained a wide popularity (Scherer, 2019) with the most popular local keypoint detectors and descriptors SURF (Bay, Ess, Tuytelaars, & Van Gool, 2008), SIFT (Lowe, 2004) or ORB (Rublee, Rabaud, Konolige, & Bradski, 2011). In the previous work and this paper, we use 128-element SIFT descriptors. The system also allowed to query the database about the image content by SQL commands. Information about local visual features was indexed in the relational database by fuzzy sets (Cpalka, Lapa, Przybyl, & Zalasinski, 2014; Harmati, Bukovics, & Koczy, 2016; Łapa, Szczypta, & Venkatesan, 2012; Prasad et al., 2017; Scherer, Smolag, & Gaweda, 2016; Stanovov, Semenkin, & Semenkina, 2016). and the AdaBoost algorithm (Viola & Jones, 2001). The mechanism used for database image indexing is depicted in details in Korytkowski et al. (2016) and Korytkowski (2017). Usually, classifiers are used for the purposes they are intended (Hoang, 2017; Pham, Nguyen, Tran, Nguyen, & Ha, 2017), but in the paper, we use weak classifiers to obtain distinctive features for a given visual class. Currently, deep learning-based approaches (Bologna & Hayashi, 2017; Chang, Constante, Gordon, & Singana, 2017) are gaining popularity in image analysis, but they are slower than the proposed approach and not suitable for relational database purposes.

Nowadays we face a constant increase in the need for processing big data (Marszalek, 2016; Nguyen, Fujita, & Dieu, 2016; Serdah & Ashour, 2016). A concept of high-speed solutions in content-based image retrieval is a natural solution for improving efficiency. The algorithms become more and more accurate, but in most cases, this fact does not contribute to decreasing the time of retrieval. Possible solutions are parallel (Piech & Stys, 2007) or distributed computing. In the paper, we propose a distributed framework for image retrieval from a relational database based on the system described in the previous paragraph. We decrease the time of searching across the file system and delivering the final results using file management by the Hadoop Distributed File System (HDFS) and Apache Spark (http://spark.apache.org/streaming/). By using the Spark framework, we manage file streams to determine the fastest way to cooperate with the Hadoop file system. This brings us to maintain the improvement of the algorithm accuracy without occurring processing-time spikes, comparing to traditional mass-storage entities.

Every time a user queries the proposed system with an image, keypoints for this image are generated using a local interest point detection (SIFT in our case). Those results are compared with a set of pre-generated features stored in a relational database engine, e.g. MS SQL Server. As a result, we obtain a list of linked image addresses (given by filenames). We spread this list through the Apache Spark framework to find, map and reduce the final results to receive unique file names and percentage matching to objects found in the query image.

The proposed system is unique as it is a rare example of CBIR in a relational database with an image index created by fuzzy sets, boosting meta learning and image local keypoints. Creating the index is fast so is its expanding in the case of adding new image classes. Moreover, thanks to using the Hadoop Distributed File System, it is highly scalable, and the performance of the system depends only on the hardware resources. The paper is an extended version of the paper (Lagiewka, Korytkowski, & Scherer, 2017). We added the description of the indexing method used for fast image retrieval and extended the
experiments. The rest of the paper is organised as follows. Section 2 describes our method for image indexing. In Section 3 we present the proposed distributed CBIR system. We show the experiment results in Sections 4 and 5 concludes the paper.

2. Image indexing by fuzzy rules

The method for image indexing described in this section was first presented in detail in our previous work Korytkowski et al. (2016) and Korytkowski (2017). We use boosting meta-learning to obtain the intervals of visual feature values that will be utilised in the database index. Boosting is used to find the most representative features, similar to the idea presented by Viola and Jones (2001). We create weak classifiers in the form of fuzzy rules. Fuzzy rules have an antecedent and a consequent part. The antecedent part has fuzzy sets with Gaussian membership functions whose parameters are adapted during the boosting procedure. The main idea is to find the most representative fuzzy rules for a given image class and to classify query images fast afterwards. The algorithm uses boosting meta-learning to generate a suitable number of weak classifiers. In each step, we randomly choose one local feature from the set of positive images according to its boosting weight. Then we search for similar feature vectors from all positive images. Using these similar features, we construct one fuzzy rule. Undoubtedly, it is impossible to find exactly the same features in all images from the same class; thus we search for feature vectors which are similar to the feature picked randomly in the current step. This is one of the reasons for using fuzzy sets and fuzzy logic. The main idea of the method presented in this section is to find the most representative fuzzy rules for a given class \( \omega_c \), \( c = 1, \ldots, V \), of visual objects and to classify query images fast afterwards. This section describes the learning process, i.e. generating fuzzy rules from a set of examples. The algorithm uses boosting meta-learning to generate a suitable number of weak classifiers. The classifiers feature space \( \mathbb{R}^N \) consists of elements \( x_n, n = 1, \ldots, N \). For example, in the case of using the standard SIFT descriptors, \( N = 128 \). In each step, we randomly choose one local feature from the set of positive images according to its boosting weight. Then we search for similar feature vectors from all positive images. Using these similar features, we construct one fuzzy rule. Undoubtedly, it is impossible to find exactly the same features in all images from the same class; thus we search for feature vectors which are similar to the feature picked randomly in the current step. This is one of the reasons for using fuzzy sets and fuzzy logic. The rules have the following form:

\[
R^c_t: \text{IF } x_1 \text{ is } G^c_{1,t} \text{ AND } x_2 \text{ is } G^c_{2,t} \text{ AND } \ldots \text{ AND } x_{128} \text{ is } G^c_{128,t} \text{ THEN image } i \in \omega_c(\beta^c_t) \quad (1)
\]

where \( t = 1, \ldots, T^c \) is the rule number in the current run of boosting, \( T^c \) is the number of rules for the class \( \omega_c \) and \( \beta^c_t \) is the importance of the classifier, designed to classify objects from the class \( \omega_c \), created in the \( t \)th boosting run. The weak classifiers (1) consist of fuzzy sets with Gaussian membership functions

\[
G^c_{n,t}(x) = e^{-\frac{(x-m^c_{n,t})^2}{\sigma^c_{n,t}^2}}, \quad (2)
\]

where \( m^c_{n,t} \) is the centre of the Gaussian function (2) and \( \sigma^c_{n,t} \) is its width. For the clarity of presentation this section describes generating the ensemble of weak classifiers for a class \( \omega_c \); thus the class index \( c \) will be omitted.
Let $I$ be the number of all images in the learning set, divided into two sets: positive images and negative images, having, respectively, $I_{pos}$ and $I_{neg}$ elements. Obviously $I = I_{pos} + I_{neg}$. Positive images belong to a class $v_c$ that we train our classifier with. For every image from these two sets, we determine local features, for example, local interest points using, e.g. SIFT or SURF algorithms. The points are represented by descriptors, and we operate on two sets of vectors: positive descriptors $\{p_i: i = 1, \ldots, L_{pos}\}$ and negative ones $\{n_j: j = 1, \ldots, L_{neg}\}$. In the case of the standard SIFT algorithm, each vector $p_i$ and $n_j$ consists of 128 real values. Let $v_i$ be the number of keypoint vectors in the $i$th positive image, let $u_j$ be the number of keypoint vectors in the $j$th negative image. Then, the total number of learning vectors is given by

$$L = \sum_{i=1}^{I_{pos}} v_i + \sum_{j=1}^{I_{neg}} u_j,$$

where $L = L_{pos} + L_{neg}$. According to the AdaBoost algorithm, we have to assign a weight to each keypoint in the learning set. When creating new classifiers, the weights are used to indicate keypoints which were difficult to handle. At the start of the algorithm, all the weights have the same, normalised values

$$D'_1 = \frac{1}{L} \text{ for } I = 1, \ldots, L.$$

Let us define matrices $P_t$ and $N_t$ constituting the learning set

$$P_t = \begin{bmatrix} p_1^1 & D'_t^1 & \cdots & p_{N_{pos}}^1 & D'_t^{I_{pos}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ p_{L_{pos}}^1 & D'_t^{L_{pos}} & \cdots & p_{N_{pos}}^{L_{pos}} & D'_t^{L_{pos}} \end{bmatrix},$$

$$N_t = \begin{bmatrix} n_1^1 & D'_t^1 & \cdots & n_{N_{neg}}^1 & D'_t^{I_{neg}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{L_{neg}}^1 & D'_t^{L_{neg}} & \cdots & n_{N_{neg}}^{L_{neg}} & D'_t^{L_{neg}} \end{bmatrix}.$$

The learning process creates $T$ simple classifiers (weak learners in boosting terminology) in the form of fuzzy rules (1). Each run $t$, $t = 1, \ldots, T$, of the proposed algorithm yields a fuzzy rule $R_t$. The process of building a single fuzzy classifier is presented below.

(1) Randomly choose one vector $p_r^i$, $1 \leq r \leq L_{pos}$ from positive samples using normalized distribution of elements $D'_t^1, \ldots, D'_t^{I_{pos}}$ in matrix (5). This drawn vector becomes a basis for generating a new classifier, and the training set weights contribute to the probability of choosing a keypoint.

(2) For each image from the positive image set, find the feature vector which is nearest to $p_r^i$ (for example according to the Euclidean distance) and store this vector in matrix $M_t$ of the size $I_p \times N$. Every row represents one feature from a different image $v_i$. 
\( i = 1, \ldots, I_{\text{pos}}, \) and no image occurs more than once

\[
M_t = \begin{bmatrix}
\hat{p}_{t,1} & \cdots & \hat{p}_{t,N} \\
\vdots & \ddots & \vdots \\
\hat{p}_{t,1} & \cdots & \hat{p}_{t,N}
\end{bmatrix},
\quad (7)
\]

Each vector \([\hat{p}_{t,1} \cdots \hat{p}_{t,N}]\), \( j = 1, \ldots, I_{\text{pos}}, \) in matrix (7) contains one visual descriptor from the set \([p; i = 1, \ldots, I_{\text{pos}}]\). For example, in view of descriptions (5) and (3), the first row in matrix (7) is one of the rows of the following matrix

\[
\begin{bmatrix}
p_1^1, \ldots, p_1^N \\
\vdots \\
p_N^1, \ldots, p_N^N
\end{bmatrix},
\quad (8)
\]

where \(v_1\) is the number of feature vectors in the first positive image.

(3) In this step a weak classifier is built, i.e. we find centres and widths of Gaussian functions which are membership functions of fuzzy sets in a fuzzy rule (1).

(a) Compute the absolute value \(d_{t,n}\) as the difference between the smallest and the highest values in each column of the matrix (7)

\[
d_{t,n} = \left| \min_{i=1,\ldots,I_{\text{pos}}} p_{n}^i - \max_{i=1,\ldots,I_{\text{pos}}} p_{n}^i \right|,\quad (9)
\]

where \(n = 1, \ldots, N\). Compute the centre of fuzzy Gaussian membership function (2) \(m_{t,n}\) in the following way

\[
m_{t,n} = \max_{i=1,\ldots,I_{\text{pos}}} p_{n}^i - \frac{d_{t,n}}{2}.\quad (10)
\]

Now we have to find the widths of these fuzzy set membership functions. We assume that for all real arguments in the range of \([m_{t,n} - d_{t,n}/2; m_{t,n} + d_{t,n}/2]\) the Gaussian function (fuzzy set membership function) values should satisfy \(G_{n,t}(x) \geq 0.5\). Only in this situation do we activate the fuzzy rule. As we assume that \(G_{n,t}(x)\) is at least 0.5 to activate a fuzzy rule, using simple substitution \(x = m_{t,n} - d_{t,n}/2\), we obtain the relationship for \(\sigma_{t,n}\)

\[
\sigma_{t,n} = \frac{d_{t,n}}{2\sqrt{-\ln(0.5)}}\quad (11)
\]

Finally, we calculate the values \(m_{t,n}\) and \(\sigma_{n,t}\) for every element of \(n\)th column of matrix (7), by repeating the above steps for all \(N\) dimensions. In this way, we obtain \(N\) Gaussian membership functions of \(N\) fuzzy sets. Of course, we can label them using fuzzy linguistic expressions such as ‘small’ or ‘large’, but here we denote them only in a mathematical sense by \(G_{n,t}\), where \(n, n = 1, \ldots, N\), is the index associated with feature vector elements and \(t\) denotes the fuzzy rule number.
(b) Using values obtained in point a) we can construct a fuzzy rule which creates a fuzzy classifier (1).

(4) Now we evaluate the quality of the classifier from in step 3 by the use of the standard AdaBoost algorithm (Schapire, 1999). Let us determine the activation level of the rule $R_t$ which is computed by a t-norm of all fuzzy sets membership function values

$$f_t(\bar{x}) = \frac{N}{n=1} G_{n,t}(\bar{x}_n),$$

(12)

where $\bar{x} = [x_1, \ldots, x_N]$ is a vector of values of linguistic variables $x_1, \ldots, x_N$. In the case of the minimum t-norm formula (12) becomes

$$f_t(\bar{x}) = \min_{n=1}^{N} G_{n,t}(x_n).$$

(13)

As a current run of the AdaBoost is for a given class $\omega_c$, we can treat the problem as a binary classification (dichotomy), i.e. $y^l = 1$ for descriptors of the positive images and $y^l = 0$ for descriptors of the negative images. Then the fuzzy classifier decision is computed by

$$h_l(\bar{x}) = \begin{cases} 1 & \text{if } f_t(\bar{x}) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}.$$  

(14)

For all the keypoints stored in matrices $P_t$ and $N_t$ we calculate new weights $D_{l+1}^t$. To this end, we compute the error of classifier (14) for all $L = L_{pos} + L_{neg}$ descriptors of all the positive and the negative images

$$\varepsilon_t = \sum_{l=1}^{L} D_l^t(I(h_l(\bar{x}) = y^l),$$

(15)

where $l$ is the indicator function

$$l(a \neq b) = \begin{cases} 1 & \text{if } a \neq b \\ 0 & \text{if } a = b \end{cases}.$$  

(16)

If $\varepsilon_t = 0$ or $\varepsilon_t > 0.5$, we finish the training stage. If not, we compute new weights

$$\alpha_t = 0.5 \ln \frac{1 - \varepsilon_t}{\varepsilon_t},$$

(17)

$$D_{l+1}^t = D_l^t \exp \left\{ - \alpha_t l(h_l(\bar{x}) = y^l) \right\} / C,$$

(18)

where $C$ is a constant such that $\sum_{l=1}^{L} D_{l+1}^t = 1$. Finally, the classifier importance is determined by $\beta_t = \alpha_t / \sum_{t=1}^{T} \alpha_t$.

It should be noted that the classifier importance $\beta_t$ is needed to compute the overall response of the boosting ensemble for the query image. The above boosting procedure should be executed for every visual object class $\omega_c$, $c = 1, \ldots, V$, thus after the training procedure, we obtain a set of $V$ strong classifiers that describe the whole image dataset. Let us assume that we have a new query image and an associated set of $u$ visual features
represented by matrix $Q$

$$Q = \begin{bmatrix} q^1 \\ q^2 \\ \vdots \\ q^u \end{bmatrix} = \begin{bmatrix} q^1_1 \cdots q^1_N \\ q^2_1 \cdots q^2_N \\ \vdots \\ q^u_1 \cdots q^u_N \end{bmatrix}.$$  \hspace{1cm} (19)

Let us determine the value of

$$F_t(Q) = \sum_{j=1}^u \left( \sum_{n=1}^N G_{n,t}(q^j_n) \right),$$  \hspace{1cm} (20)

where $S$ and $T$ are $t$-norm and $t$-conorm, respectively (see Scherer, 2012). To compute the overall output of the ensemble of classifiers, for each class $\omega_c$ we sum weak classifiers outputs (20) taking into consideration their importance, i.e.

$$H^c(Q) = \sum_{t=1}^{T_c} \beta_t F_t(Q).$$  \hspace{1cm} (21)

And we can assign a class label to the query image in the following way

$$f(Q) = \arg \max_{c=1,\ldots,V} H^{c}\hspace{1cm} (22)$$

In formulas (21) and (22) we restored class label index $c$, which was removed at the beginning of this section. In formula (20) $t$-norm and $t$-conorm can be chosen as min and max operators, respectively.

### 3. Proposed system

Figure 1 presents a general concept of the designed solution. The system has to be initialised by preparing a dataset using our previous work (Lagiewka, Scherer, & Angryk, 2016) to index each image and generate local interest points with the procedure presented in Section 2. Then, the system is ready to analyse it, to find the most relevant images from the database and finally to add it to the collection for future queries. Generally, to improve the retrieval speed with local features, we face a choice between decreasing the precision, e.g. by lowering descriptors’ detail level and reducing the pool of the dataset, e.g. by choosing a group of images as a result of a database query for further comparing. We decided to use both of these methods to strengthen the effect of reducing the list of possible data to compare with higher precision and creating percentage coverage index in order to sort it out. At this point we decided to reduce hardware deficiencies by virtualization, however, we took a step ahead and moved the virtual machines dedicated to data storage to separate hard disk drives to protect searching across the distributed file system from the bottleneck effect. We kept data operating servers (Master and SQL Server) within one solid state drive to physically increase processing time and maintain better data transfer between software and hardware (for writing operations, e.g. collecting results from HDFS). For better performance of the image retrieval we require our storage to be accessible on all worker nodes. Distributed File System (DFS) is an approach which allows to store expansible varieties of data. The significant aspect of accessing files is a
single namespace for the entire system (Salehian & Yan, 2016). The most common distributed storage system implementation with high fault-tolerance are Hadoop DFS, Google Cloud, Amazon Simple Storage Service (S3) and Tachyon, Li, Ghodsi, Zaharia, Shenker, and Stoica (2014). HDFS is a file-oriented divided storage with permissions integrated with the operating system, whereas S3 is object-oriented, and files are stored as objects within containers (called buckets). Tachyon is a memory-centric distributed storage system in which Master operates on permissions and global metadata, data are stored as replicated blocks close to HDFS, because it is based on it. Google Cloud Platform mixes object-oriented buckets with HDFS as it is based on Spark and Hadoop. Each of the listed platforms is paid (either per user account or used hardware), except of the Apache Hadoop, which allows to build a custom setup for better adjustment to the needs of the proposed system. Of course, building such a system requires an investment in hardware. The fact of hardware limitations is the most common issue in the image retrieval and may cause unexpected delays while processing. We build the presented system with a specific hardware configuration in which (Table 1):

- Server (Master) is a virtual machine managing input, output and database requests, equipped with 2 virtual cores and 2 GB of RAM (SSD1),

Table 1. Virtual machines setup.

| Name          | vCores | RAM     | Storage |
|---------------|--------|---------|---------|
| Server        | 2      | 2 GB    | SSD     |
| SQL Server    | 2      | 3 GB    | SSD     |
| Worker 1      | 1      | 1.5 GB  | HDD     |
| Worker 2      | 1      | 1.5 GB  | HDD     |
| Worker 3      | 1      | 1.5 GB  | HDD     |

Figure 1. Diagram of the proposed system for image retrieval. Images are stored in the Hadoop Distributed File System distributed among slave machines and are indexed by a relational database management system basing on the colour information and local keypoints. The Master handles query images, queries the database for similar images and retrieves them from the slaves.
- SQL Server is a virtual machine handling database (MS SQL Server), equipped with 2 virtual cores and 3 GB of RAM (SSD1),
- Worker 1–3 (Slaves) are virtual machine set up for the Hadoop environment with 1 virtual core and 1.5 GB of RAM each (HDD1, HDD2, HDD3).

To reduce the bottleneck effect we placed every worker on a separate physical drive and assigned a single virtual core, each with a clock speed of 3.5 GHz to avoid hardware limitations. Our solution is based on the Java platform to provide cross-platform communication and input-output interface. Every query (input) image is served by the Master, which generates keypoints and supports transactions with MS SQL Server. After the return of the database query results, the Master searches across the distributed file system and collects (maps) possible results with matching percentage of each image coverage of image keypoints (in the case of multiple objects participating in the input image, percentage coverage is expressed by the average of each matching). Next, the Master reduces the results to create a list of unique file links ordered by similarity to the query image and sends it to the end-user. We can describe the time effort needed to complete the retrieval by $T_f = T_p + T_s + T_c + T_l$, where $T_f$ is the total time from submitting the input image to receiving the final output, $T_p$ is the time of the input image processing (generating keypoints and database querying), $T_c$ is the time of reducing and comparing for the best match, $T_l$ is the time of ordering the final list for output, $T_s$ is the time of searching across the distributed file system and merging the results, given by

$$T_s = T_{s1} \lor T_{s2} \lor \cdots \lor T_{sn} \text{ or } T_s = \text{MAX}(T_{s1}, \ldots, T_{sn}).$$

(23)

This means that $T_p$ depends on the algorithm speed, $T_p$ is always the longest time from each Worker (Slave), $T_c$ is strictly connected with the input image complexity (e.g. number of objects, generated keypoints to compare within) and $T_l$ depends on the sorting time (i.e. on the list size). Although Slaves work asynchronously, there have to be set a watch to control if every Slave responded (from Slaves which are currently available in the network). Time $T_c$ can be also represented as

$$T_c = \sum_{i=1}^{n} T_{oi} + T_n + \sum_{j=1}^{m} T_{vij},$$

(24)

where $n$ is a number of Slaves, $T_o$ is the time of the network connection and operations (since we use local network, we assume it is approximately constant for every Master-Slave communication), $T_r$ is the time of reducing list of further proceeded images (e.g. names, links, paths), $m$ is the number of objects detected in initial processing ($T_p$), $T_v$ is the time of calculating percentage participation of keypoints in a single object between input and compared image (Juan & Gwun, 2009). Every input image at the system entry is blurred with Gaussian blur to denoise it from small details and real noise. At the next step we determine if there is a need of a limitation for colour occupying less than certain percentage, e.g. 5%. The reason for strict limitation can be limited resources or limited retrieval (processing) time. The other method is to pick a number of dominant colours by the occupation ratio, e.g. three dominating colours. After reducing the number of colours we work with, we are able to select from the database objects (rows) describing images by keypoints, which were stored with relations to certain groups of
colours with their participation percentage. Having indexed images from the dataset stored in the database, we are provided with the list of images related to selected keypoints. At this point we test the input image with each row of the selected keypoints within matching colour group. If we were able to find any matching object within or near a colour group, we build a local histogram for these objects. These matched objects can be related to a certain position in the image to retrieve those structures at a specific position, unless the matched object covers most of the image or it is the image itself. The next step allows to verify if the list of local histograms with related keypoint structures is not empty. If the list does contain at least one object, we can assume there is at least one image in the dataset matched with the input image. This method allows us to check related list of images indexed in the database in relation with keypoint structures to determine if any of these images contain matching objects from the input image. Related images do not have to cover all matched objects, but the more is matched, the better the result is. These results containing any of the matched objects in the input image are added to the final list in preparation for returning to the user. An important issue in retrieval systems is setting if we want to return the final list as an ordered or unordered list. Returning unordered objects is faster, as we return matching images as soon as the system notices a match between images (Figure 2).

Figure 2. Workflow of the proposed distributed image retrieval based on global colour information (colour histogram) and local features (SIFT keypoints).
4. Experimental results

We performed several tests, passing multiple images from the PASCAL Visual Object Classes (VOC) dataset (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010) as input to check if our system is able to handle various object shapes, angles and image distortions. We also tested cases when some parts of our system are unreachable (e.g. by shutting down one of the Workers) and we were able to confirm that a distributed file system (HDFS with Spark in the presented case) can provide meaningful results due to its replication methods. Then, we migrated part of the Slaves to other physical hardware and by analysing communication times we were able to conclude that the time-results were only slightly slower. In order to consider all the possibilities, we moved further in optimizing metadata of the dataset in SQL Server. We designed a table structure to handle a simple attribute, colour in our case, for each indexed object. We decided to store it as quantized colour histogram at each image segment. During empirical tests, initial dataset reduction using pre-indexed metadata improved efficiency and effectiveness of the retrieval due to a smaller dataset and the additional parameter. As a result, the end-user receives better matched results ordered by similarity to the query image.

Content-based image retrieval is a complex process requiring hardware resources and a specific knowledge about the expected results. Our aim was to find a balance between the final image list accuracy and performance by reducing the number of colours and ignoring comparison of object sizes between matched objects in the input image and an image being compared to. We tested the system by querying it with various images and an example with returned similar images is shown in Figures 3 and 4. Generally, the accuracy of the retrieval is similar to the one presented in Korytkowski et al. (2016) and Korytkowski (2017) as the image indexing procedure is the same. The goal of the solution presented in the paper is to achieve speed and scalability. It is hard to compare the solution to the previous one by the authors as it is very hardware-depended.

We recommend to load objects to RAM before retrieving the queue to minimize the latency of the hard drive reading process (creating retrieving buffer). Even if the reduced dataset is small enough to fit it fully to RAM we strongly recommend to split it in buffered queue parts, unless the dataset is smaller than a single buffer queue part. A partial buffer should not be used in the case of parallel requests to the system. This method is inspired by the Tachyon system of storing metadata of last used objects in fast memory (RAM or SSD). If there exists a possibility to add another parameter (no hardware limitation) to store in a SQL database, e.g. colour, texture, structure spatial position, we strongly recommend to do so. This is an important step in reducing dataset within certain parameters which translates into faster processing of smaller datasets.

5. Conclusion

Existing CBIR systems are rather not designed to work with database environment. The presented system for content-based image retrieval can work in a relational database environment. The system has a good scalability, which means that the number of slave machines can be increased. To optimize transactions between Master and SQL Server we can merge them into one virtual machine to maximally reduce network operations (with migrating Hadoop to Windows environment due to MS SQL Servers used in the
The proposed approach demonstrates several advantages partly coming from the original method presented in Korytkowski et al. (2016) and Korytkowski (2017). The indexing method is relatively accurate in terms of visual object classification. The training phase is relatively fast and image classification stage is very fast. Expanding the system knowledge is efficient as adding new visual classes to the system requires generation of new fuzzy rules whereas in the case of, e.g. bag-of-features it requires new dictionary generation and re-training of classifiers. The system is highly scalable and the performance of the system depends only on the hardware resources. The accuracy of the image retrieval is
Figure 4. Example query image number two with the returned list of images. The first image is the query image.
similar to the one presented in Korytkowski et al. (2016) and Korytkowski (2017) as the image indexing procedure is the same. The goal of the solution presented in the paper is to achieve speed and scalability. It is hard to compare the solution speed to the previous one by the authors as it is very hardware-dependent. Generally, it is faster than a not-distributed version and adding more hardware and slave machines makes it faster. There can be done some further changes in the processing of the initial input image by incorporating features similar to Lagiewka et al. (2016). This means the proposed system can recognize objects with colour parameter given. Such a feature might be able to reduce a subset of compared images, which means faster processing on a smaller amount of objects matching colour requirements. Storing additional data such as texture, colour or approximate size of objects (proportionally to relevant objects in other stored images and compared to background objects at the same image) can lead to reduce processing time due to a smaller dataset containing only objects referenced by SQL query results. The proposed solution is semi-parallel because only the file system has been distributed. Our work is mostly aimed at eliminating a bottleneck of image retrieval systems designed as single-entity solutions. The list sorting process can be further parallelized but it was not within the scope of the presented work. Performance of the database part of the solution can be also increased through the use of a SQL server cluster, where the process of generating the index in the form of rules can be parallelized and spread across several servers. There is also a possibility to exchange the relational database engine into a distributed database, e.g. Apache HBase or MapR-DB.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Notes on contributors**

Michał Lagiewka is a PhD candidate in computer science at Czestochowa University of Technology. His current work is related to indexing of visual data in databases, distributed storage and cloud computing. Currently is pursuing his PhD in the field of computer vision and databases.

Marcin Korytkowski is an associate professor of computer science at Czestochowa University of Technology. His scientific interests are neural networks, network security and databases.

Rafal Scherer is an associate professor of computer science at Czestochowa University of Technology. His research focuses on developing new methods in computational intelligence and data mining, ensembling methods in machine learning, content-based image indexing and classification and computer security. He authored more than 100 research papers and two books.

**ORCID**

Marcin Korytkowski http://orcid.org/0000-0002-6002-2733

Rafal Scherer http://orcid.org/0000-0001-9592-262X

**References**

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding, 110*(3), 346–359.
Bologna, G., & Hayashi, Y. (2017). Characterization of symbolic rules embedded in deep DIMLP networks: A challenge to transparency of deep learning. *Journal of Artificial Intelligence and Soft Computing Research, 7*(4), 265–286.

Chang, O., Constante, P., Gordon, A., & Singana, M. (2017). A novel deep neural network that uses space-time features for tracking and recognizing a moving object. *Journal of Artificial Intelligence and Soft Computing Research, 7*(2), 125–136.

Cpalka, K., Lapa, K., Przybyl, A., & Zalasinski, M. (2014). A new method for designing neuro-fuzzy systems for nonlineair modelling with interpretability aspects. *Neurocomputing, 135*(C), 203–217.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision, 88*(2), 303–338.

Harmati, I. A., Bukovics, A., & Koczy, L. T. (2016). Minkowski’s inequality based sensitivity analysis of fuzzy signatures. *Journal of Artificial Intelligence and Soft Computing Research, 6*(4), 219–229.

Hoang, V. D. (2017). Multiple classifier-based spatiotemporal features for living activity prediction. *Journal of Information and Telecommunication, 1*(1), 100–112.

Juan, L., & Gwun, O. (2009). A comparison of sift, PCA-sift and surf. *International Journal of Image Processing, 3*(4), 143–152.

Korytkowski, M. (2017). Novel visual information indexing in relational databases. *Integrated Computer-Aided Engineering, 24*(2), 119–128.

Korytkowski, M., Rutkowski, L., & Scherer, R. (2016). Fast image classification by boosting fuzzy classifiers. *Information Sciences, 327*, 175–182.

Lagiewka, M., Korytkowski, M., & Scherer, R. (2017). Distributed image retrieval with colour and keypoint features. *2017 IEEE international conference on innovations in intelligent systems and applications (INISTA) (pp. 45–50). Gdynia, Poland.*

Lagiewka, M., Scherer, R., & Angryk, R. (2016). Color-based large-scale image retrieval with limited hardware resources. *Artificial intelligence and soft computing: 15th international conference, ICAISC 2016, Proceedings, Part II, Zakopane, Poland, June 12–16, 2016 (pp. 689–699). Springer International Publishing.*

Lapa, K., Szczypa, J., & Venkatesan, R. (2015). Aspects of structure and parameters selection of control systems using selected multi-population algorithms. *International conference on artificial intelligence and soft computing (pp. 247–260). Springer.*

Li, H., Ghodsi, A., Zaharia, M., Shenker, S., & Stoica, I. (2014). Tachyon: Reliable, memory speed storage for cluster computing frameworks. *Proceedings of the ACM symposium on cloud computing (pp. 1–15). ACM.*

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision, 60*(2), 91–110.

Marszalek, Z. (2016). *Novel recursive fast sort algorithm* (pp. 344–355). Cham: Springer International Publishing.

Nguyen, K. P., Fujita, G., & Dieu, V. N. (2016). Cuckoo search algorithm for optimal placement and sizing of static var compensator in large-scale power systems. *Journal of Artificial Intelligence and Soft Computing Research, 6*(2), 59–68.

Ogle, V. E., & Stonebraker, M. (1995). Chabot: Retrieval from a relational database of images. *Computer, 28*(9), 40–48.

Pham, T. N., Nguyen, V. Q., Tran, V. H., Nguyen, T. T., & Ha, Q. T. (2017). A semi-supervised multi-label classification framework with feature reduction and enrichment. *Journal of Information and Telecommunication, 1*(4), 305–318.

Piech, H., & Stys, M. (2007). Serialization of input and output data transfers in parallel structures. *Scientific Research of the Institute of Mathematics and Computer Science, 6*(1), 241–252.

Prasad, M., Liu, Y. T., Li, D. L., Lin, C. T., Shah, R. R., & Kaiwartya, O. P. (2017). A new mechanism for data visualization with TSK-type preprocessed collaborative fuzzy rule based system. *Journal of Artificial Intelligence and Soft Computing Research, 7*(1), 33–46.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). Orb: An efficient alternative to sift or surf. *2011 IEEE international conference on computer vision (ICCV) (pp. 2564–2571). Barcelona, Spain.*

Salehian, S., & Yan, Y. (2016). Comparison of spark resource managers and distributed file systems. *2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing
and networking (SocialCom), sustainable computing and communications (SustainCom) (BDCloud-
SocialCom-SustainCom) (pp. 567–572). Atlanta, GA.
Schapire, R. E. (1999). A brief introduction to boosting. Proceedings of the 16th international joint con-
ference on artificial intelligence - Volume 2, IJCAI’99 (pp. 1401–1406). San Francisco, CA: Morgan
Kaufmann Publishers Inc.
Scherer, R. (2012). Multiple fuzzy classification systems (Vol. 288). Berlin: Springer-Verlag.
Scherer, R. (2019). Computer vision methods for fast image classification and retrieval. Springer
International Publishing.
Scherer, M., Smolag, J., & Gaweda, A. (2016). Predicting success of bank direct marketing by neuro-
fuzzy systems. International conference on artificial intelligence and soft computing (pp. 570–576).
Springer International Publishing.
Serdah, A. M., & Ashour, W. M. (2016). Clustering large-scale data based on modified affinity propa-
gation algorithm. Journal of Artificial Intelligence and Soft Computing Research, 6(1), 23–33.
Stanovov, V., Semenkin, E., & Semenkina, O. (2016). Self-configuring hybrid evolutionary algorithm for
fuzzy imbalanced classification with adaptive instance selection. Journal of Artificial Intelligence
and Soft Computing Research, 6(3), 173–188.
Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceed-
ings of the 2001 IEEE computer society conference on computer vision and pattern recog-
nition. CVPR 2001 (Vol. 1, pp. I-511–I-518).