Parametric configuration approach of a multidimensional content-based image retrieval model

Kyrylo Smelyakov[1,†], Stanislav Danylenko[1,*,†], Anastasiya Chupryna[1,†] and Victoria Vysotska[2,†]

[1] Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, 61166, Ukraine

[2] Department of Information Systems and Networks,Lviv Polytechnic National University, Stepan Bandera 12, Lviv, 79013, Ukraine

Abstract

This study addresses the importance of proper parameter configuration in search models and its impact on processes and retrieval results, using the content-based image retrieval model called the Multidimensional Cube as a case study. This model proposes a partitioning of the feature space to reduce the search space, using number of dimensions and number of segments on dimension as main parameters. The work analyzes these parameters and presents the experimental approach for configuring them. It also includes a series of experiments with the evaluation of model construction time and retrieval results under different configurations, comparing them with the outcomes obtained through brute-force search across the entire storage. The main conclusion is that incorrect parametric configuration, even of a highly efficient search model, can negate its advantages. The examined configuration enables a deeper analysis of the model, allowing for the identification of relationships and dependencies when applied to a specific task. It also helps to uncover subtle aspects of parameter usage that may not be explicitly documented by the model's creator or provider.

Keywords

software engineering, parametric configuration, content-based image retrieval, multidimensional model, image processing, image storage, search efficiency

1. Introduction

A data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints [1]. In a broader sense, a data model is a conceptual abstraction over data structures and the methods that operate on the data within those structures. When it comes to content-based image retrieval (CBIR), the data model refers to an abstract component of the search system that defines how it: interprets a search query; manages data: contains methods and algorithms for processing data, organizing and searching it in the data structures of feature database; and ranks results [2]. And a search system is one that provides a user interface for inputting queries and obtaining results, along with additional functionality such as filtering, pagination, and other interactive features.

That is, the key component in the image search process is precisely the search model, although it remains invisible to the end user. This model can be general-purpose, such as one that used by the Google search system, or specialized for use in advertising, scientific research, military affairs, medicine, and other fields. This means that the efficiency of the search and the satisfaction of the end user depend directly on the effectiveness of the model. Moreover, the effectiveness of the model depends on the correct configuration of its parameters [3].

[*] *MoDaST 2025: Modern Data Science Technologies Doctoral Consortium, June, 15, 2025, Lviv, Ukraine*

[1*] Corresponding author.

[†] These authors contributed equally.

kyrylo.smelyakov@nure.ua (K. Smelyakov); stanislav.danylenko@nure.ua (S. Danylenko); anastasiya.chupryna@nure.ua (A. Chupryna); victoria.a.vysotska@lpnu.ua (V. Vysotska).

0000-0001-9938-5489 (K. Smelyakov); 0000-0002-8142-3018 (S. Danylenko); 0000-0003-0394-9900 (A. Chupryna); 0000-0001-6417-3689 (V. Vysotska).

Parametric configuration of models is frequently encountered in real-world applications. For instance, in video surveillance and emergency systems, it is essential to correctly set parameters such as motion detector sensitivity, detection zones, minimum event duration, and system response time. Incorrect parameter values may result in missed critical incidents or false alarms that trigger emergency services unnecessarily. This type of configuration is also common in a wide range of services, including recommendation systems, antivirus software, voice recognition systems, and credit scoring in the banking sector [4, 5].

Similarly, in the field of CBIR, model parameters can significantly affect either search speed or result quality. This may have negative consequences in critical domains such as medical diagnostics or emergency recognition systems. Therefore, an important research problem arises – studying the parameters of CBIR systems, their configurability, and their impact on search outcomes [6].

The objective of this study is to explore this problem using the Multidimensional Cube model as a case study and experimental approach of the parametric configuration. To address this problem, the following tasks must be accomplished: define a parameter selecting approach, experimentally evaluate its effectiveness, and draw conclusions specific to the given model.

2. Related Works

CBIR models use calculated image descriptors in form of the vector of fractional numbers for the images from the main storage. The simplest CBIR search model is the brute-force model. It does not require additional configuration parameters, as it does not partition the feature space but instead exhaustively compares all descriptors upon receiving a query. While this approach ensures high retrieval accuracy, it is inefficient in terms of search speed. However, even this model, like any other, relies on a similarity metric for descriptor comparison, such as Euclidean distance, Squared Euclidean distance, Manhattan distance, Jaccard index, or Hamming distance [7].

Multidimensional tree-based models, such as KD-Trees, have several tunable parameters, including the number of dimensions, splitting criterion, tree depth, the number of points per leaf node, stopping criteria, and whether or not backtracking to alternative branches is enabled. Such trees may also be self-balancing – dynamically rebuilding as new data is added, or grow incrementally without rebalancing [8].

Graph-based models, such as Hierarchical Navigable Small World, are defined by parameters such as the maximum number of connections per node, search breadth (during both indexing and querying), and dimensionality. Wider search leads to more accurate results, but also increases search time [9].

In hash-based models, such as Locality-Sensitive Hashing, key parameters include the number of hash buckets and the number of hash functions per bucket. A higher number of buckets and functions typically improves accuracy, but also increases insertion and query time [10].

Clustering-based models, such as those using k-means, are primarily influenced by the number of clusters, the number of training rounds, and the stopping condition for element assignment. More clusters and iterations generally lead to higher clustering accuracy and lower density per cluster, but increase training time and the number of clusters to examine during retrieval [11].

Models that employ Product Quantization, such as the Inverted Multi-Index, require specification of the number of subspaces and the number of clusters within each subspace. These parameters affect memory consumption, clustering precision, search speed, and overall retrieval quality [12-14].

This analysis demonstrates that each CBIR model has its own parameter set, and the impact of these parameters on model behavior and final search performance is often underdocumented. As a result, suboptimal configuration may lead to inefficient use of otherwise effective retrieval models.

3. Case study

The Multidimensional Cube (MDC) is a high-performance and efficient CBIR model. It employs a reduction and partitioning approach to the multidimensional feature space by dividing it into clusters, which serve as cells within a bounded multidimensional space. Like other models discussed in this work, MDC has its own set of parameters, the influence of which on search performance is the subject of investigation in this study [2].

The process of constructing the MDC is fairly standard for CBIR models and is schematically illustrated in Figure 1. Features are extracted from images, based on which a descriptor is formed. Important: MDC uses already prepared and provided descriptions. It can use a variety of descriptors, but they must be homogeneous, contain the vector of fractional numbers and can be aggregated. Based on the total number of images in the source storage and the requirements, the number of MDC cells is determined, based on which the parameters for the number of dimensions and the number of segments are selected. Some of the processed descriptors are inserted into the MDC, its content is analyzed and, based on this, the cell boundaries are optimized. The MDC is cleared, the cell boundaries are rebuilt, and the MDC is filled with all descriptors. The structure is ready to accept search requests [2].

The search process, however, is unique and leverages the advantages of the MDC structure. It is also illustrated schematically in Figure 1. The same preprocessing steps are performed for the uploaded image as for images from the storage. Based on the processed descriptor, the cell in which similar (or the same) images are located is determined. For descriptors, their difference from the searched one is calculated. If necessary, the same search is performed for descriptors from cells adjacent to the current one. Search results are ordered based on similarity measures. Images are retrieved from the main storage using the found IDs and returned to the user [2].

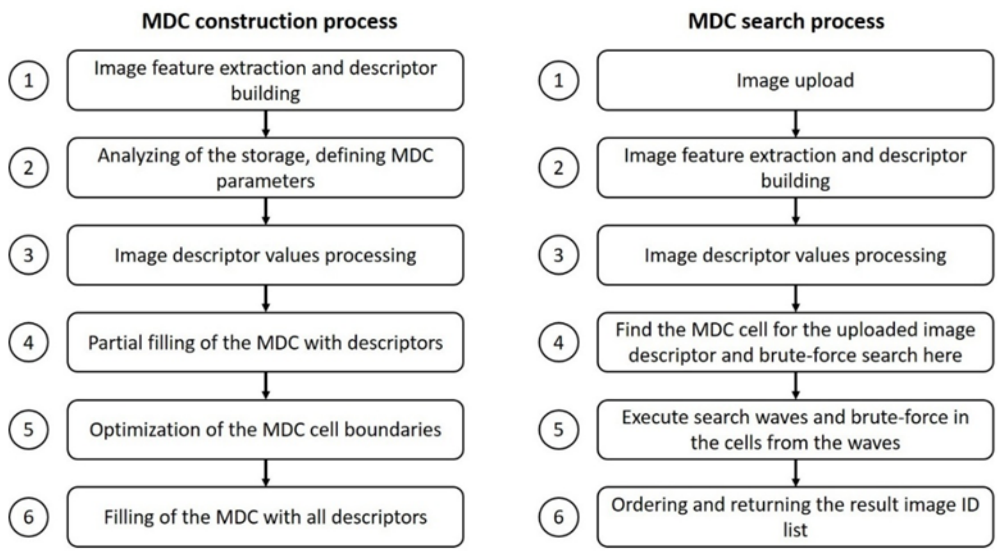
MDC construction process
1 Image feature extraction and descriptor building
2 Analyzing of the storage, defining MDC parameters
3 Image descriptor values processing
4 Partial filling of the MDC with descriptors
5 Optimization of the MDC cell boundaries
6 Filling of the MDC with all descriptors
MDC search process
1 Image upload
2 Image feature extraction and descriptor building
3 Image descriptor values processing
4 Find the MDC cell for the uploaded image descriptor and brute-force search here
5 Execute search waves and brute-force in the cells from the waves
6 Ordering and returning the result image ID list

Figure 1: The main flows of the MDC model: construction process (left) and search process (right)

As can be seen from the diagrams of the main processes, the parameters defined during the model preparation stage directly influence both the construction of the MDC and the execution of the search within it.

3.1. Main Parameters

The main parameters of the MDC model are the number of MDC cells (*c*), the number of dimensions (*d*), and the number of segments (*s*). The number of cells directly depends on the expected number (*en*) of images to be retrieved in a single search iteration (search page) from the total number of images in the repository (*D*), and also on the specific task for which the MDC is applied. These parameters are related as follows:

$$c = s^d, sp = \frac{D}{c}, en \approx sp, \tag{1}$$

where c – is the number of cells, s – is the number of segments, d – is the number of dimensions, sp – is the real search page size, D – is the total number of images in the storage, en – is the expected search page size.

Thus, the parameters s and d are selected in such a way that the number of image descriptors within a single MDC cell sp is close to the number en that needs to be displayed on one search results page.

Another important parameter is applied to the model's storage system. Currently, the MDC can be stored in a relational database (RDB). In this configuration, the segment boundaries for each dimension are stored in separate tables, the descriptors are stored in another table, and the index vectors – corresponding to the segment numbers for each dimension – are stored in yet another table. Therefore, selecting the most efficient database management system (DBMS) is a critical component of the configuration [2].

3.2. Applying parameters

The main parameters are applied to the model construction process. Descriptors are often of high dimensionality and commonly have a length that is a power of two. During parameter selection, dimensionality reduction is performed – typically through pairwise aggregation of adjacent elements – until the target dimensionality d is achieved. Descriptor values are normalized within a specific range, and this range may change after aggregation. Next, the resulting range is divided into s segments, each defined by clear boundaries. Each descriptor value is then assigned to a segment with a specific index, thereby forming an index vector that determines the descriptor's position in the multidimensional space.

An example of descriptor processing for parameters d = 4 and s = 4 is shown in Figure 2.

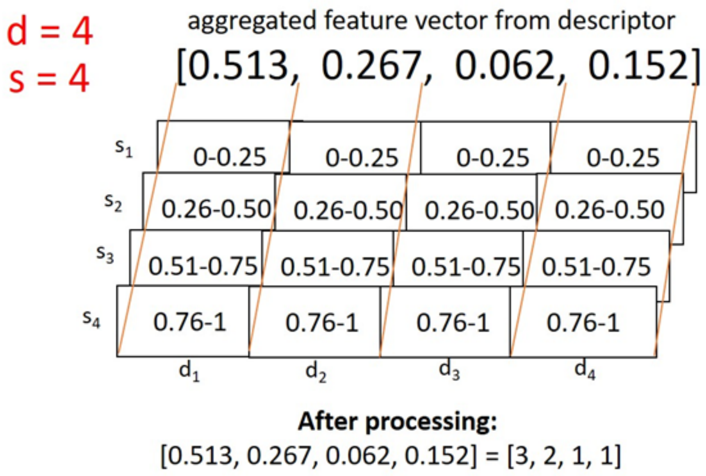

Figure 2: Processing of the descriptor vector to achieve specific count of the dimensions and segments and produce an index vector

An example of placing a processed descriptor into the multidimensional feature space and into the MDC is presented in Figure 3, for the configuration with parameters: d = 3, s = 3, and index vector [1, 3, 3].

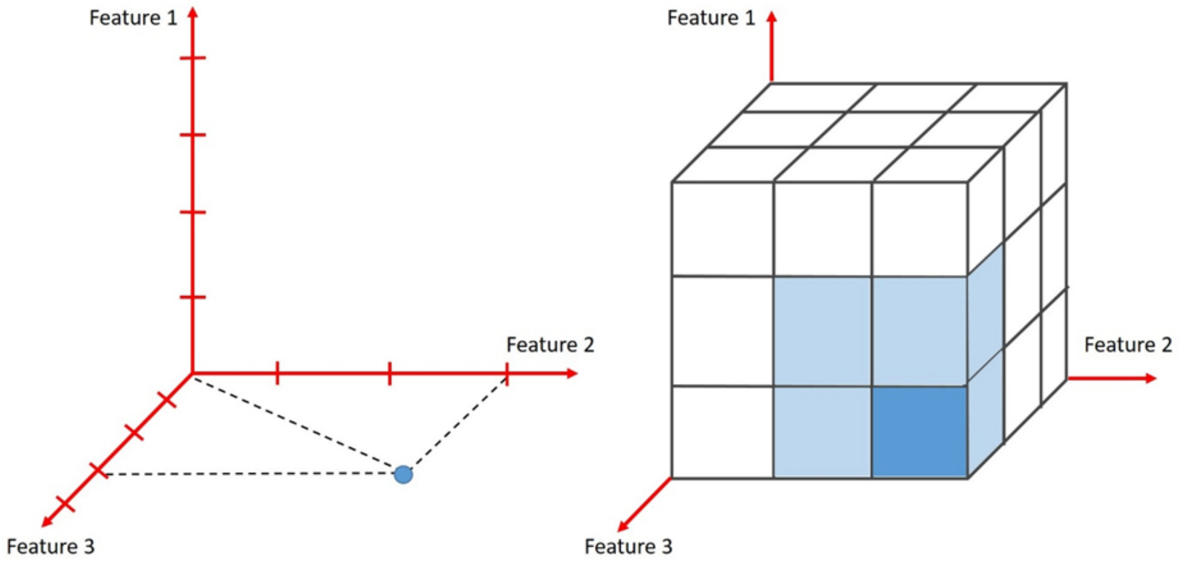

Figure 3: Image descriptor in the bounded multidimensional space (left) and in the MDC cell with index [1, 3, 3]. Example of the search with wave around target cell.

The initial segment boundaries are defined uniformly within the valid value range after dimensionality reduction. However, an optimization step follows, aiming to ensure that each segment in every dimension contains an equal number of values [2].

Approximately 20% of image descriptors from the storage are inserted into the MDC at this stage. The number of values per segment under uniform partitioning is calculated, and the segment boundaries are then shifted slightly, either increased or decreased, to achieve the desired number of values in each segment. This process is illustrated in more detail in Figure 4.

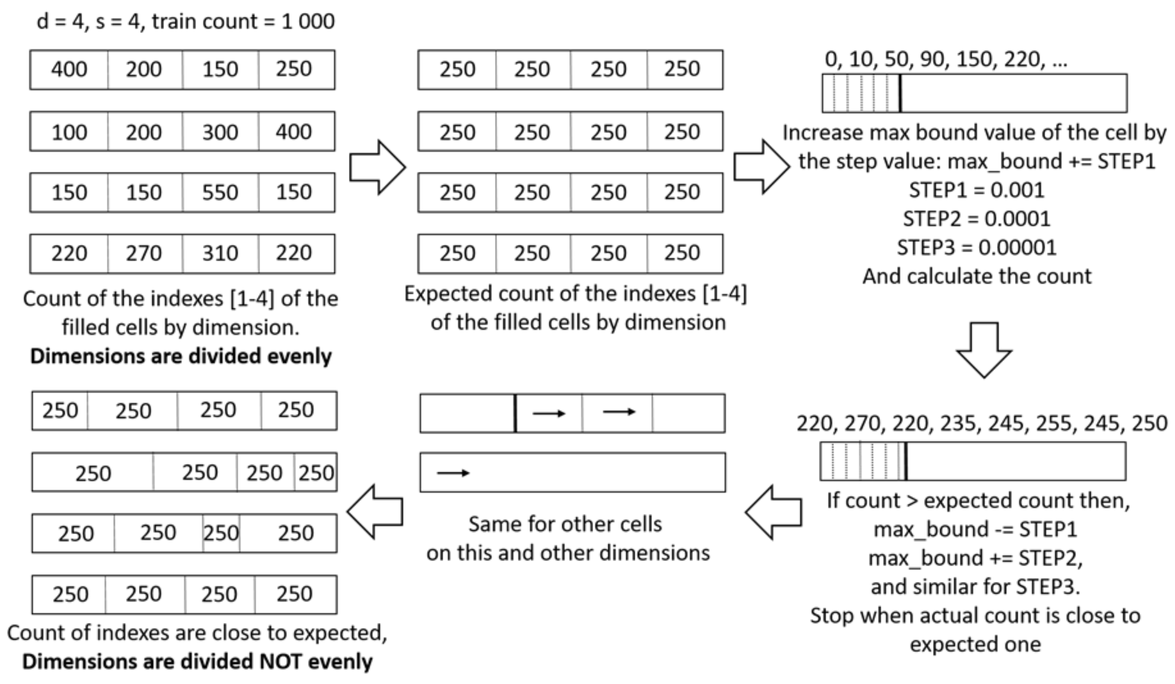
d = 4, s = 4, train count = 1 000
400 200 150 250
100 200 300 400
150 150 550 150
220 270 310 220
Count of the indexes [1-4] of the filled cells by dimension.
Dimensions are divided evenly
250 250 250 250
250 250 250 250
250 250 250 250
250 250 250 250
Expected count of the indexes [1-4] of the filled cells by dimension
0, 10, 50, 90, 150, 220, ...
Increase max bound value of the cell by the step value: max_bound += STEP1
STEP1 = 0.001
STEP2 = 0.0001
STEP3 = 0.00001
And calculate the count
220, 270, 220, 235, 245, 255, 245, 250
If count > expected count then,
max_bound -= STEP1
max_bound += STEP2,
and similar for STEP3.
Stop when actual count is close to expected one
Same for other cells on this and other dimensions
250 250 250 250
250 250 250 250
250 250 250 250
250 250 250 250
Count of indexes are close to expected,
Dimensions are divided NOT evenly

Figure 4: Cell boundaries optimization process

As a result, descriptors are efficiently distributed across the MDC cells, and the search process can begin. The descriptor vector of the input image is processed in the same way as during construction to determine the cell to which it would belong if it were added to the model. This query vector is then compared against all vectors located in the identified cell, using one of the available similarity metrics: Euclidean, Manhattan, or cosine distance. [2]

If fewer descriptors are found in the cell than requested by the user or the specific needed images were not found, a search wave is triggered. This involves generating indices of neighboring cells by incrementing or decrementing the vector values by an amount corresponding to the wave number: 1, 2, 3, etc. The search is then repeated in these neighboring cells. Once the required number of descriptors is found, they are sorted by similarity and returned to the user. An example of the search wave mechanism is illustrated in Figure 3 (right).

3.3. Parametric Configuration

The essence of the experimental parametric configuration approach lies in selecting model parameters for given input conditions and evaluating the model's effectiveness under those parameters to identify patterns or correlations. In the case of the MDC model, the key parameters and their interrelationships are clearly defined, as are the optimization and search processes. However, the influence of these parameters on the efficiency of these processes remains non-obvious and requires experimental investigation.

The parametric configuration process can be visualized in the form of a table: the first columns contain the parameter values, the third shows the total number of cells, and the fourth displays the number of descriptors per cell. As mentioned earlier, d can only take values that are powers of two. An example of parameter selection for an MDC model intended to store 100 000 image descriptors, with a defined search page size of 10, is presented in Table 1. Only a fragment of the parameter selection table is presented.

Table 1
Fragment of the MDC parameter selection, c = 100 000, en = 10

Number of the dimensions d	Number of the segments s	Number of the cells c	Estimated number of descriptors per cell sp
2	100	10 000	10
4	10	10 000	10
4	3	6 561	15

The table summarizes the best-performing configurations across varying dimensionalities. Further experiments may help establish whether increasing the number of dimensions or the number of segments per dimension leads to greater efficiency. This conclusion is based on the metric values obtained during the experiments. The specific evaluation metrics will be discussed in detail in the following sections.

The identification of the most suitable DBMS is also carried out experimentally, as numerous factors may influence the results – including the operating system, software versions, and the performance and architecture of the underlying hardware. In this context, candidate approaches are compared based on the time required for model construction and search execution. These aspects will also be addressed in more detail later.

4. Experiments and Discussions

To validate the impact of the discussed parametric configuration for the search model (MDC), this section presents the results of experiments on content-based image retrieval using descriptors. It includes relevant performance metrics, comparative analysis with existing search model, as well as conclusions and recommendations for the configuration of the model.

4.1. Software implementation and workstation

To conduct the experiments, the specialized application was developed. It was implemented in Java 17 using the Spring Framework 6. It allows to upload descriptors to the application from a csv file. The experiment results can be downloaded as an Excel file. The JDBC interface is used as the communication interface between the server and the database.

The workstation for the experiments is based on CPU Intel i5 8250U, RAM 16Gb DDR4 2133MHz, SSD 512 Gb, GPU NVidia MX-150.

The Manhattan distance is used as a similarity measure for descriptors.

4.2. Initial Data

The COCO2017 dataset [15], which consists of 123 403 images, was used for the experiment. Although this dataset is relatively small, it is quite sufficient to show all the implementation features and key estimates of the effectiveness of the proposed image search model. The images of the COCO2017 dataset are very diverse, which is one of the important conditions for the independence of the experiment. The dataset was pre-processed by an independent service: image descriptors of the COCO2017 dataset were built and provided in the form of a csv file. They were imported into the storage of the developed search application.

All descriptors had a length of 32 values. Each value contained the frequency in 4 byte floating point format. Thus, the size of each descriptor was 128 bytes. For some images in the dataset, descriptors of shorter length were obtained. They were excluded from the experiments. There were 122 628 descriptors left in the dataset (with a length of 32).

4.3. Metrics

Since the main goal of the work is to evaluate the impact of parameter configuration on the performance of the search model, so the main metrics are labor intensity, search quality and search time. All other metrics are optional.

Labor Intensity (*li*) – is the number of descriptors that were examined during the search.

Search Quality (*q*) – is mainly determined by the order in which the images appear in the final sorted list *L*. Let us say in storage *p1* images of the same type (we consider the original, its copies and transformations). *max* is the number of the last such image in a list *L*. Let us find the quantity *p2* images in the storage that do not belong to the group of images of the same type under consideration, but whose serial numbers are in the final selection *L* are less than *max*. The presence of images with such numbers means that the images we are looking for are not a continuous list. Between them, there are atypical images, which is a sign of a decrease in search quality. Although, first of all, this shows the shortcomings of the selected descriptor type. This is analogous to the presence of impurities in the material. In such a situation, let's evaluate the search quality as follows:

$$q=1-\frac{(p2)}{(p1+p2)}, \qquad (2)$$

where *q* – the quality of the search, *p1* – the number of the images of the same type, *p2* – the number of the images that do not belong to the group of images of the same type.

Search Time (*t*) – this indicator is more interesting for the final estimate of the effectiveness of the software and hardware implementation in the complex. When it is difficult to evaluate individual solutions and hardware implementations.

Count of Waves (*cw*) to find the last image from the group images of the same type in MDC.

Percent (*p*) of the viewed descriptors from all descriptors in the storage.

Additional metrics are important for estimation intermediate results, the effectiveness of internal processes, and also for testing.

4.4. Experiments plan

The experiment consists of four stages:

1. Selection of parameters and configuration of the model with varying settings.
2. Optimization of the model under different parameter configurations and comparison of the results.
3. Evaluation of the efficiency of various DBMSs for storing the MDC model, focusing on both model construction and search execution.
4. Using the best-performing DBMS from stage 3, executing searches within MDC models configured with different parameters and comparing the results. A baseline model that performs a brute-force search over the entire dataset is also included in the comparison. To determine whether the results produced by a poorly configured model can be inferior to those of the most primitive model. This baseline was implemented using the same technologies as the MDC model.

For the stage 4: 100 images are randomly selected from the storage for search experiments. For each of them, 2 transformations are created (rotated by 180 degrees, scaled down by 2 times), then their descriptors are built and loaded into the storage. Two modes of the search are performed: search until find only original file (case for examining only target cell) and search until find original and modified images (case for applying wave-search). It is not classical top-N search because we want to evaluate metrics for the case when the user want to find all necessary images instead of first N images. All metrics are estimated and the results are presented in average (arithmetic mean), minimum and maximum values.

4.5. Experiments plan

A total of 122 628 descriptors need to be placed in the model. The search page size is set to 10. Parameter selection for *d* and *s* is performed according to the rules described earlier, following the example shown in Table 1. A fragment of the parameter selection results is presented in Table 2.

Table 2
Fragment of the MDC parameter selection, *c* = 122 628, *en* = 10

Number of the dimensions *d*	Number of the segments *s*	Number of the cells *c*	Estimated number of descriptors per cell *sp*
2	100	10 000	12.26
2	95	9 025	13.58
2	90	8 100	15.13
4	10	10 000	12.2
4	11	14 641	8.37
4	12	20 736	5.91
8	3	6 561	18.69

The full set of configurations is not shown, as many combinations yield only minor differences in results. Configurations with $d = 2$ are not considered suitable, as the impact of dimensionality reduction may be too significant in this case. Moreover, the concept of the cube's multidimensionality is effectively lost under such a low-dimensional setting. Configurations with the same value of d are also not selected for the experiment, in order to make the comparison more illustrative and the experiment more demonstrative.

The combination of the parameters $d = 4$, $s = 10$ and $d = 8$, $s = 3$ were selected. Hereafter, the configuration with parameters $d = 4$, $s = 10$ will be referred to as Configuration 1 and the configuration with parameters $d = 8$, $s = 3$ will be referred to as Configuration 2. Parameters $d = 4$, $s = 10$ are better align with the expected result; however, in addition to the initial evaluation, there are also secondary factors that will be examined during the model optimization and search stages (original with modifications search case). It is possible that a less favorable parameter combination at this stage may demonstrate more promising results in later evaluations.

For these selected parameters the effectiveness of the DBMSs was evaluated: for the Configuration 1 it is presented in Table 3, for the Configuration 2 in Table 4. For comparative analysis of the DBMSs, the latest versions of the each of at the time of the experiments were used: PostgreSQL 15.2, MySQL 8.0.31, MS SQL Server 2022 16.0.4065, Oracle DB XE 21. Build time means spent time for constructing and optimizing of the MDC. Search time is time spent for one stage of the experiments, that will be described later.

Table 3
Execution time for operations, Configuration 1

DBMS name	Build time (min)	Search time (min)
MySQL	84.3	1.0
PostgreSQL	23.3	0.5
SQL Server	29.5	1.89
Oracle DB	30.3	0.6

Table 4
Execution time for operations, Configuration 2

DBMS name	Build time (min)	Search time (min)
MySQL	88.0	10.9
PostgreSQL	9.7	3.7
SQL Server	20.8	9.2
Oracle DB	31.4	20.1

The results were measured after a cold start to exclude the influence of caches formed inside the DBMS, and without any additional settings of the MDC. As a result, it was found that the optimal DBMS for further experiments was PostgreSQL 15.2 since it demonstrated the best results for filling and search for both MDC configurations.

4.6. Optimizing MDC filling

Initially, the feature dimensions are divided into *s* segments evenly. With a uniform distribution, the count of the vector values for each segment should be approximately as follows: for the Configuration 1 is 122 628 /10 = 12 262, for the Configuration 2 is 122 628/3 = 40 876.

Prior to optimization, the obtained values deviate considerably from the expected targets. Figures 5 and 6 illustrate the actual distribution statistics for the Configuration 1 and Configuration 2, respectively.

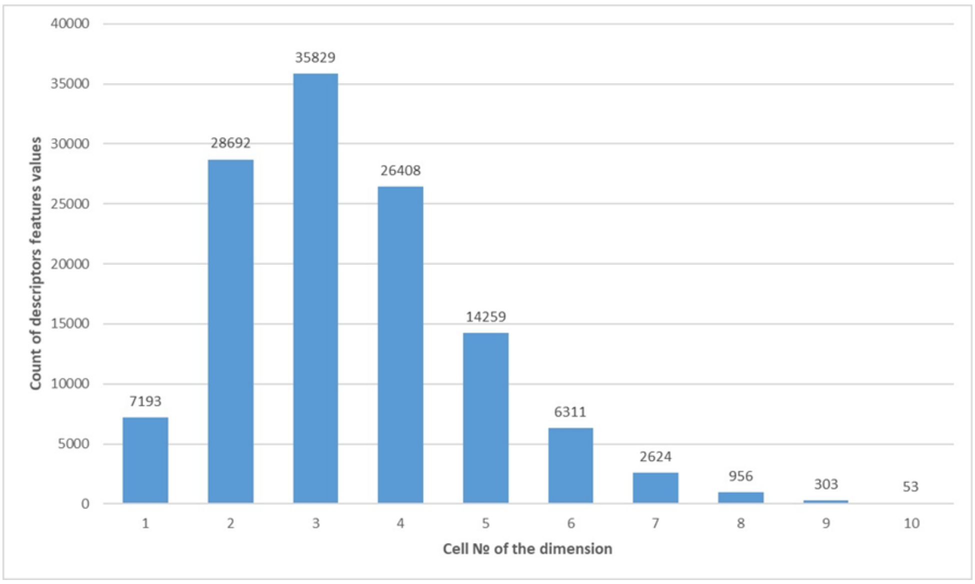

Figure 5: Statistics of dimension values distribution by segments for the Configuration 1

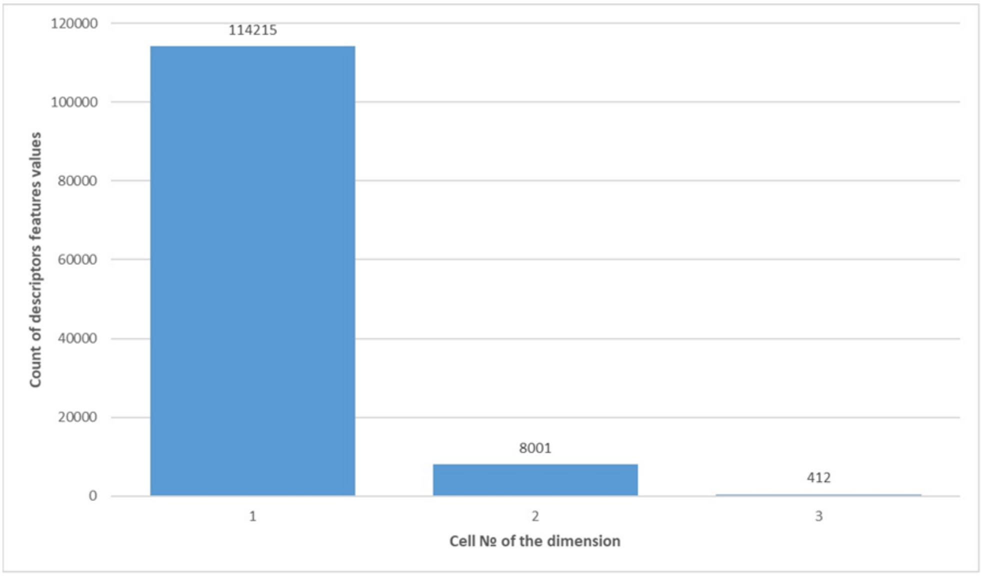

Figure 6: Statistics of dimension values distribution by segments for the Configuration 2

Subsequently, 20 000 image descriptors were extracted from the storage and loaded into the MDC. Segment boundary optimization was then performed based on this subset. Once the boundaries were determined, the remaining descriptors were loaded, and the value distribution statistics were calculated in the same manner. Using only a portion of the data for boundary optimization significantly reduces the time required for this step.

Figures 7 and 8 illustrate the distribution statistics after the optimization for the Configuration 1 and the Configuration 2, respectively.

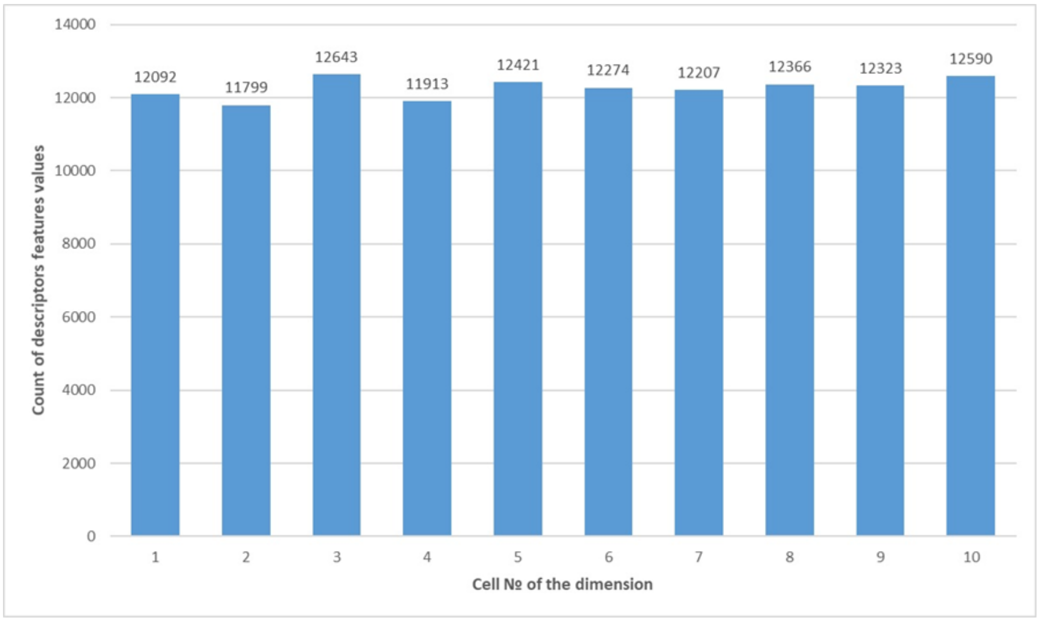

Figure 7: Statistics of dimension values distribution by segments for the Configuration 1

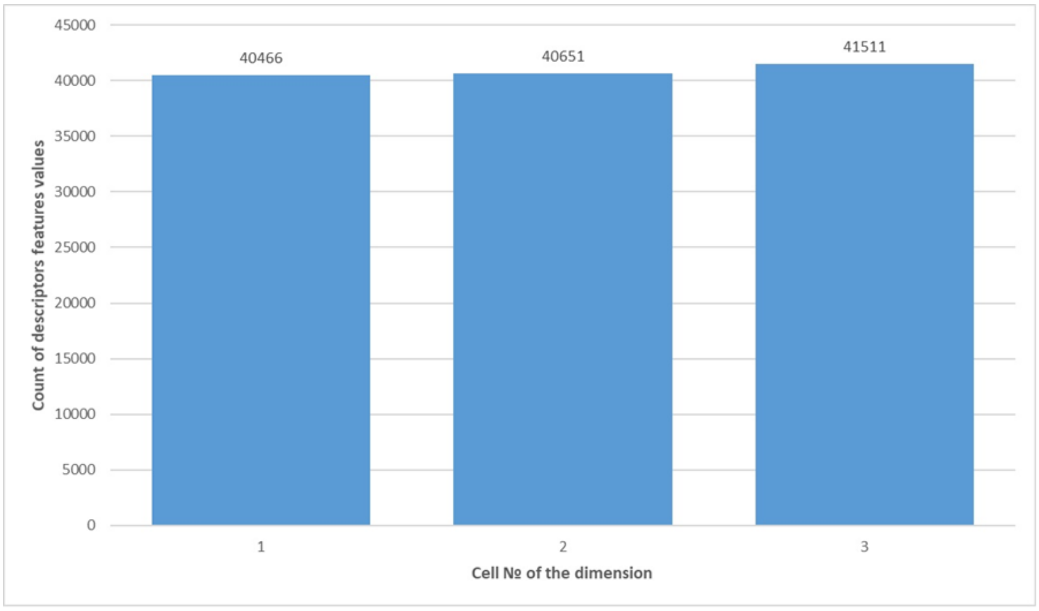

Figure 8: Statistics of dimension values distribution by segments for the Configuration 2

In both configurations, the number of values distributed across segments is close to the expected. This indicates that the model parameters have a limited impact on the outcomes achieved during the optimization process.

4.7. Search

After the optimizations, the MDC was filled with image descriptors of the considered COCO2017 dataset for each configuration. 100 images from the dataset were randomly selected. The previously mentioned modifications were built for them. For all modifications of the original images, descriptors were built and added to the model. Next, a search was carried out for images (originals and modifications) of the selection under consideration. For all images in the selection, the measure of difference from the desired image is calculated.

The experimental results with the average (arithmetic mean), maximum and minimum values of the metrics under consideration for searching for original images of the Configuration 1 are presented in Table 5, and for modified images in the Table 6, for the Configuration 2 in the Table 7 and Table 8, respectively. These results are the average results of performing the 10 rounds of the experiment. 10 rounds are the one stage of the experiment.

Similar results for a brute-force search are presented in Table 9 and Table 10, respectively. Of course, count of waves metric is not calculated for this type of the search. Brute-force search model is located in the RAM memory instead of DB.

Table 5
Results of the search of the original images, Configuration 1

	Labor intensity (c) [count of descriptors]	Quality (q) [measure from 0 to 1]	Count of waves (cw)	Percentage of all descriptors (p) [%]	Search time (min)
Min	1.1	1	0	0.001	0.012
Max	491.6	1	0	0.400	0.054
Average	60.27	1	0	0.049	0.019

Table 6
Results of the search of the original and modified images, Configuration 1

	Labor intensity (c) [count of descriptors]	Quality (q) [measure from 0 to 1]	Count of waves (cw)	Percentage of all descriptors (p) [%]	Search time (min)
Min	3.6	0.067	0	0.003	0.012
Max	5 446.2	1	1	4.434	0.415
Average	783.65	0.970	0.36	0.638	0.063

Table 7
Results of the search of the original images, Configuration 2

	Labor intensity (c) [count of descriptors]	Quality (q) [measure from 0 to 1]	Count of waves (cw)	Percentage of all descriptors (p) [%]	Search time (min)
Min	1.1	1	0	0.001	0.014
Max	1 541.5	1	0	1.255	0.102
Average	241.03	1	0	0.188	0.032

Table 8
Results of the search of the original and modified images, Configuration 2

	Labor intensity (c) [count of descriptors]	Quality (q) [measure from 0 to 1]	Count of waves (cw)	Percentage of all descriptors (p) [%]	Search time (min)
Min	3	0.115	0	0.002	0.013
Max	49 728.6	1	1	40.486	1.903
Average	5 223.05	0.970	0.25	4.252	0.314

Table 9
Results of the search of the original images, brute-force

	Labor intensity (c) [count of descriptors]	Quality (q) [measure from 0 to 1]	Percentage of all descriptors (p) [%]	Search time (min)
Min	1 223.3	1	0.996	0.002
Max	122 053.4	1	99.369	0.496
Average	61 028.36	1	49.686	0.155

Table 10
Results of the search of the original and modified images, brute-force

	Labor intensity (c) [count of descriptors]	Quality (q) [measure from 0 to 1]	Percentage of all descriptors (p) [%]	Search time (min)
Min	28 016.7	0.085	22.810	0.066
Max	122 356	1	99.616	0.408
Average	93 525.48	0.966	76.143	0.246

4.8. Discussions

As part of the experiments, the impact of different experimentally selected MDC model parameters was evaluated with respect to the two main processes performed by the model: search speed and model construction speed.

The model construction time was better for Configuration 2 up to 30%. Since the construction process also includes the optimization stage, the reported results apply to both processes. However, the choice of DBMS had a significant impact on performance. PostgreSQL outperformed MySQL by more than a factor of 10 in Configuration 2 and by up to 30% in other cases. The quality of the optimization is similar for both configurations. So, the configuration and the DBMS choice have an impact on construction and optimization process.

A similar situation was observed during the search experiments phase. The retrieval quality for original images was equally high across all configurations. The reduction in the search quality of modified images is determined by the properties of the descriptor. We can see that for both search methods the corresponding average values are almost the same: MDC = 0.97 and brute-force = 0.966.

For the brute-force search, the expected result was obtained when searching for original images – approximately half of the storage had to be scanned. When searching for modified versions of images, the number of scanned descriptors was significantly higher, reaching around 75% of the entire storage.

For original image retrieval, the average labor intensity in Configuration 2 was more than four times higher, as each cell contains a larger number of descriptors compared to Configuration 1. However, the absolute value remained below 1 000, whereas in the brute-force approach it reached approximately 61 000. In terms of search time, Configuration 2 performed about twice as slow as Configuration 1, yet it was still five times faster than the brute-force search. Count of the waves is equal to 0 for both configurations.

When searching for both original and modified images, the average search time in Configuration 2 was more than seven times higher than in Configuration 1. Although search waves were triggered less frequently (an average of 0.25 waves compared to 0.36 in Configuration 1), the cells contained a greater number of descriptors. Still, the average search complexity in Configuration 2 was 5 223, which is significantly lower than that of the brute-force approach (93 525). In terms of search speed, Configuration 1 demonstrated the best results (averaging 0.063 seconds), while Configuration 2 performed worse than brute-force search (averaging 0.314 vs. 0.246 seconds). This is attributed to the fact that, during wave-based searches, Configuration 2 requires communication with the database, whereas the brute-force model was fully loaded into RAM.

However, special attention should be given to the maximum labor intensity observed in Configuration 2. This value approached 50% of the entire dataset loaded into the MDC, prompting further analysis. A theoretical calculation was performed to estimate the number of descriptors examined during the search process and one search wave, resulting in the following formula:

$$ws = \frac{D}{s^d} \times 3^d, \tag{3}$$

where *ws* – denotes the number of descriptors checked during a single search wave, and *D*, *s*, *d,* correspond to the total number of descriptors in the dataset, the number of segments per dimension, and the number of dimensions, respectively, as defined in formula (1).

Thus, for Configuration 2, the maximum number of descriptors that must be examined during a search wave equals the size of the entire dataset, which is unacceptable. This revealed the significant impact of model parameters on the execution of wave-based search. Formula (3) should be considered one of the key factors in selecting appropriate parameters. It is necessary to introduce a threshold ratio between the number of descriptors retrieved in a single wave and the total number of descriptors in the dataset.

5. Conclusions

This study introduced an experimental parametric configuration approach for content-based image retrieval models, with the Multidimensional Cube model serving as the primary case study. The proposed method evaluates how variations in model parameters influence both construction and search performance, allowing for the identification of key dependencies between parameter choices and functional behavior.

Using a dataset of 122 628 common image descriptors, two configurations were compared, and multiple DBMS platforms were tested as storage backends. The experiments revealed that while the parameter configuration had minimal impact on model construction time, the choice of DBMS could influence performance by up to 30%. In contrast, parameter selection significantly affected search efficiency – leading to differences of up to 7× in computational workload and up to 6× in search time.

A formal relationship was derived linking configuration parameters to the number of descriptors retrieved per wave during the search. Thresholds were proposed for these values and incorporated into the official MDC configuration guidelines, thereby enhancing its practical utility.

The proposed approach demonstrates how empirical parameter tuning can significantly improve CBIR model performance and may serve as a general framework for evaluating model suitability in domain-specific applications.

Further research will focus on detailed analysis of MDC's operational phases, refinement of the configuration algorithm, and incorporation of adaptive constraints. Additionally, we plan to assess the applicability of MDC in the medical domain – specifically in diabetic patient monitoring – where retrieval accuracy and speed are critical for timely intervention. Proper parametric configuration in such cases may have a direct impact on diagnostic quality and clinical outcomes.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] H.F. Korth, S. Sudarshan, A.S. Professor, Database System Concepts, McGraw—Hill Educa-tion, 2019.

[2] S. Danylenko, S. Smelyakov, Development of a multidimensional data model for efficient content-based image retrieval in big data storage, Radioelectronic and Computer Systems 2025 (2025) 137–152. doi: https://doi.org/10.32620/reks.2025.1.10.

[3] Y. Zhang, Similarity Image Retrieval Model based on Local Feature Fusion and Deep Metric Learning, in: 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), IEEE, 2020: pp. 563–566. doi: https://doi.org/10.1109/itoec49072.2020.9141871.

[4] N. Nasaruddin, K. Muchtar, A. Afdhal, A.P.J. Dwiyantoro, Deep anomaly detection through visual attention in surveillance videos, Journal of Big Data 7 (2020). doi: https://doi.org/10.1186/s40537-020-00365-y.

[5] K. Smelyakov, A. Chupryna, D. Darahan, S. Midina, Effectiveness of modern text recognition solutions and tools for common data sources, in: CEUR Workshop Proceedings, 2021, 2870, pp. 154–165. URL: https://ceur-ws.org/Vol-2870/.

[6] D. Gupta, R. Loane, S. Gayen, D. Demner-Fushman, Medical image retrieval via nearest neighbor search on pre-trained image features, Knowledge-Based Systems 278 (2023). doi: https://doi.org/10.1016/j.knosys.2023.110907.

[7] M. Marinov, I. Valova, Y. Kalmukov, Comparative Analysis of Existing Similarity Measures used for Content-based Image Retrieval, in: 2019 X National Conference with International Participation (ELECTRONICA), IEEE, 2019: pp. 1–4. doi: https://doi.org/10.1109/electronica.2019.8825645.

[8] M. Li, H. Wang, H. Dai, M. Li, C. Chai, R. Gu, F. Chen, Z. Chen, S. Li, Q. Liu, G. Chen, A Survey of Multi-Dimensional Indexes: Past and Future Trends, IEEE Transactions on Knowledge and Data Engineering 36 (2024) 3635–3655. doi: https://doi.org/10.1109/tkde.2024.3364183.

[9] Y.A. Malkov, D.A. Yashunin, Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (2020) 824–836. doi: https://doi.org/10.1109/tpami.2018.2889473.

[10] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the Twentieth Annual Symposium on Computational Geometry, ACM, New York, NY, USA, 2004. doi: https://doi.org/10.1145/997817.997857.

[11] D. Zhang, M.M. Islam, G. Lu, J. Hou, Semantic Image Retrieval Using Region Based Inverted File, in: 2009 Digital Image Computing: Techniques and Applications, IEEE, 2009: pp. 242–249. doi: https://doi.org/10.1109/dicta.2009.48.

[12] A. Babenko, V. Lempitsky, The inverted multi-index, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012: pp. 3069–3076. doi: https://doi.org/10.1109/cvpr.2012.6248038.

[13] G. Tereshchenko, I. Kyrychenko, V. Vysotska, Z. Hu, Y. Ushenko, M. Talakh, Hybrid System for Image Storage and Retrieval in Big Data Environments, in: International Journal of Image, Graphics and Signal Processing(IJIGSP) 17(3) 55-84, 2025. doi:10.5815/ijigsp.2025.03.04.

[14] O. Tverdokhlib, V. Vysotska, O. Nagachevska, Y. Ushenko, D. Uhryn, Y. Tomka, Intelligent Processing Censoring Inappropriate Content in Images, News, Messages and Articles on Web Pages Based on Machine Learning, in: International Journal of Image, Graphics and Signal Processing(IJIGSP) 17(1). 107-164, 2025. doi:10.5815/ijigsp.2025.01.08.

[15] COCO, Common Objects in Context. URL: https://cocodataset.org/.