Constant Time Feature Matching for ID Document Type Identification with On-the-Fly Type Subset Selection

E.E. Limonova [1,2], A.V. Trusov [1,2,3], D.Z. Rybalko [2], N.S. Skoryukina [1,2], K.B. Bulatov [2]
[1] Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, 119333, Russia, Moscow, Vavilova 44, kor.2;
[2] Smart Engines Service LLC, 117312, Russia, Moscow, pr. 60-letiya Oktyabrya 9;
[3] Moscow Institute of Physics and Technology, 141701, Russia, Dolgoprudny, Institutskiy per. 9

Abstract

Identity document recognition is becoming more and more common in our daily lives. As security measures and document standards improve, the number of documents that need to be recognized is also increasing. So, one of the essential tasks of identity document recognition systems is to identify the document type from thousands of possible variants. However, in many cases, we have supplementary information and can reduce a set of possible types on-the-fly to improve processing speed and quality. In this paper, we discuss ID document recognition with on-the-fly type subset selection. The main challenges in such a system are responding within a limited time and achieving computational and memory efficiency for subset handling. We propose a solution based on a feature-matching approach using binary keypoint descriptors and adjusted multi-index hashing, which uses two new heuristics to ensure a constant number of comparisons for each request. We experimentally evaluate this method on the MIDV-500 and MIDV-2019 datasets and demonstrate that it offers an excellent combination of accuracy, configuration time, and search time compared to commonly used hierarchical clustering, hierarchical navigable small-world graphs, multi-probe locality-sensitive hashing, and straightforward brute-force solutions.

Keywords: identity documents, ID document classification, on-the-fly selection, descriptor matching, multi-index hashing, approximate nearest neighbors.

Citation: Limonova EE, Trusov AV, Rybalko DZ, Skoryukina NS, Bulatov KB. Constant Time Feature Matching for ID Document Type Identification with On-the-Fly Type Subset Selection. Computer Optics 2025; 49(6): 1061-1070. DOI: 10.18287/COJ1758.

Introduction

Recognition systems have become commonplace and are used in many areas. Recognition of identity documents is needed in banking, different government services, when buying railway or airplane tickets, going through border control, and many other cases. At the same time, such systems must provide high recognition quality, work quickly, and process information safely since identity documents contain personal data. Moreover, there is an increasing demand for compact solutions that run on general-purpose user devices like mobile phones. In particular, such solutions can help implement the regulatory restrictions necessary to address the protection of private information.

Identity document recognition systems implement a multi-stage pipeline to satisfy all these requirements. For example, it can involve locating a document within an image, determining its type, and then extracting field content. This way, we simplify data extraction as we know field places and formats (e.g., Machine-Readable Zone [1] field with a specific format, date fields, etc.). Thus, we significantly reduce computations in the recognition system, provide higher inference speed, and use additional information, allowing for better recognition quality.

However, modern identity document recognition systems face a growing demand for universality and worldwide support. It means that they are forced to maintain not just a few or dozens but hundreds of types of documents, which significantly complicates the task of locating and classifying them.

In such circumstances, using the most available information about the task becomes essential. In this paper, we consider selecting the subset of potential document types on-the-fly, because we can obtain this additional information during the recognition process. For example, it can come from the Machine-Readable Zone, from previous frames (such as when recognizing a document in a video), or directly from the user.

Type identification for ID documents is typically conducted using a feature-based approach [2, 3]. This approach involves extracting features from images and comparing them to predefined sample images that represent all supported document types [4].

Various data structures, such as hierarchical clustering, are commonly used to provide fast nearest-neighbor search for obtained features. However, many of these structures do not allow for the dynamic reduction of the feature set. This limitation necessitates a complete rebuild for each new subset, which can be time-consuming and memory-intensive.

In this paper, we consider the ID document recognition system with on-the-fly document type subset selection and propose a new approach to the feature-matching configuration and inference stages to handle it efficiently. Our approach is based on multi-index hashing [5], which enhances the original method with two heuristics to provide fast configuration and approximate search with a limited response time.

Let us summarize our contribution:

- we introduce a new property for the ID document recognition systems: on-the-fly type subset selection and discuss the feature-matching approaches in this context;
- we propose a bucket size constraint heuristic for multi-index hashing to ensure the uniform distribution across the buckets;
- we propose a heuristic on the examined bucket number for multi-index hashing to ensure a constant number of operations to limit the response time;
- we demonstrate the quality and computational efficiency of our heuristic multi-index hashing method on public datasets MIDV-500 [6] and MIDV-2019 [7] and show that it has an optimal balance between the processing speed and classification accuracy in comparison to multi-probe locality-sensitive hashing [8], hierarchical k-means trees [9], hierarchical navigable small world graphs [10] and brute-force search.

1. Related Work

Document recognition systems usually use a feature-matching approach to locate and identify documents. At the system setup stage, document types are indexed and stored in the database. Then, an input image is processed to extract features and searched in the database.

Recognition systems that search for an exact match between the input document and the already indexed document can use document hashing [11] or global descriptors [12, 13]. Systems that process documents with arbitrary content and aim to identify their types can vary significantly [14, 3]. For example, ID document type determination usually relies on keypoints and their descriptors as features, searching for the most similar sets of feature points in the database using nearest-neighbor search methods [4, 2].

The problem of nearest neighbor search is relatively well-studied. In applications that use binary descriptors such as ORB [15], BAD [16], and RFD [17], the group of methods supporting search in Hamming space is of interest. Primary methods are hierarchical clustering [18] and hierarchical k-means trees [9]. The main idea is to divide the data into k classes sequentially and build a tree for a fast search. The most popular implementation of this method is available in the FLANN library and can be used from OpenCV. There are also methods based on binary search trees [19, 20], which allow, for example, the computationally efficient addition of new descriptors to a tree for real-time scene processing. However, these methods are difficult to adapt to the task of locating and identifying the type of identity documents, as they do not allow searching only over a subset of document types.

There are also alternative methods. For example, in [21], the authors propose transforming binary descriptors into lower-dimensional real-valued vectors, performing a nearest neighbor search in Euclidean space using a kd-tree [22], and then refining the Hamming metric for a few neighbors. In [10], the authors proposed a hierarchical structure for navigable small-world graphs. The search is performed layer-by-layer and gradually converges to the nearest neighbor of the query node. However, they also have a problem considering a subset of the database or graph.

Other approaches, such as multi-index hashing [5] and multi-probe locality-sensitive hashing [8], are based on hash tables. They divide input descriptors into subsequences and use them to index the tables after hashing. To find all the neighbors, one must look through several hash buckets and test the distance to the query. Since we need to check all the elements in the bucket, we can easily discard elements with a type that is not currently enabled. Therefore, methods based on multi-index hashing are well-suited for feature-matching in document recognition systems.

2. On-the-fly ID document type subset selection problem

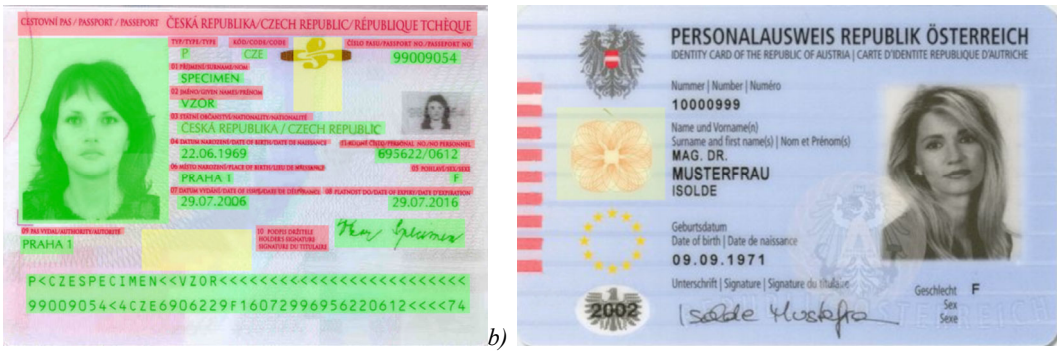
We consider the input to be a digital image that may contain an identity document of some type. Such a document encompasses three primary types of elements, exemplified in Figure 1a: background, static, and content elements.

Background elements may include patterns or designs intrinsic to the document. Background can contain complex features, for instance, the classic "guilloché" pattern (yellow zone in Fig. 1 *b*). It is commonly found in identity or registration documents, protecting against forgery. At the same time, bank plastic cards often exhibit dynamic backgrounds, which vary significantly from one series to another. These backgrounds generally do not convey informational attributes and are primarily decorative.

Static elements frequently appear as textual segments within the document. These often encompass headings and field labels, such as "name" or "address", providing structural context to the document. Other static elements in the document can include boundary lines and checkboxes, which are crucial in defining the document's structure.

Content elements, comprising data fields that contain information that changes from document to document. Such fields typically contain textual data; however, in some instances, alternative forms like barcode fields are utilized. Barcodes can occasionally function as static elements, thus helping in document classification.

So, the recognition system must locate the document in the image and determine its type based on the static and sometimes background elements, ignoring the content. In this context, additional knowledge that can restrict possible document types is crucial. For instance, when purchasing a ticket, we may be required to present various documents, such as ID cards, passports, and driving licenses, but not medical insurance.



a) *b)*

Fig. 1. a) Identity document elements: static (red), personal data content (green) and parts of background (yellow); b) "guilloché" pattern in the background (yellow). Images are taken from MIDV-2019 [7]

We aim to solve the classification task for the input image, which can be defined as follows. Let us define all possible the image classes as $C = \{C_i\}, i \in \{0, \dots, N\}$, where N is a number of document types. Then

- C_i is the class of images with the i-th document type, for $i \in \{1, \dots, N\}$;
- C_0 is a class for other images.

At the inference time, we have a current image Q to be classified, the subset $D \in C$, and also know that $Q \in D$. Also, as in Skoryukina et al. [4] for each class C_i, $i \in \{1, \dots, N\}$ we define the ideal template T_i.

The metrics to assess the quality of the solution are:

- classification accuracy,
- working time,
- required memory.

The working time can be divided into 3 times:

1. **Startup time:** initialization time at the system startup, when only the set C is given; this initialization can be done only once and does not affect user experience;
2. **Config time:** initialization time, when we obtain the set D; this kind of initialization is performed at the inference stage and directly affects the response time of the system; however, if we are going to recognize the number of images of the set D (for example, if we have a video as an input), it can be done once per a set of images;
3. **Matching time:** the working time of the method, the classification time per image.

3. Document Type Identification Pipeline

We use quite a straightforward document identification pipeline based on feature-matching inspired by [4] and [2]. At the system creation stage, we know one ideal template image for each class of documents. Each image is processed as described in [4]. We use YACIPE [23] for keypoint extraction and 128-bit RFDoc [24] for keypoint description. We keep $M = 500$ best keypoints according to the YACIPE score for each ideal template image as described in [4].

The inference consists of the following stages:

1. **Preprocessing.** We convert the input Q to grayscale and scale its longest side to 1920.
2. **Document Location.** We use [25] to find the approximate location of the document quadrangle. Then, we perform a projective transformation to rectify the document. In this work, we launch further processing for the 3 best quadrangles and the whole image if no quadrangles were found.
3. **Feature extraction.** We extract keypoints from the document area determined at the previous stage and compute their descriptors for four orientations (0 °, 90 °, 180 °, and 270 °) using the fast method from [26].
4. **Feature matching**. In this step, for each keypoint descriptor, we search for the closest descriptor among all descriptors from ideal template images. For such a descriptor, we remember a mapping (edge) that will be used in the next step.
5. **Model estimation.** We look through the set of possible document types D and for each ideal template $T_i \in D$, find the homography $Q \to T_i$ using the edges from the previous stage and RANSAC-based framework PESAC [27]. We use parameters for 2D homography estimation from [27]. Finally, we choose the template with the highest inlier rate and identify the document type.

4. Multi-index hashing

Multi-index hashing [5] is a method for exact nearest neighbor search for binary descriptors with the Hamming metric. Its idea is to divide the input binary sequence f of length b into m subsequences $f^{(k)}$, $k = 1, \dots, m$ of length b/m, and use the Pigeonhole principle to restrict the radius of the neighbor search. If

$$||f - g|| \le r,$$

there is at least one k such that:

$$||f^{(k)} - g^{(k)}|| \le \lfloor r/m \rfloor$$

So, we can create m hash tables L_m with subsequences $f^{(k)}$ as indexes and pairs of descriptors and document types (f, C) as values. To search neighbors of the descriptor f with the radius r in such a structure, we look through all the elements in buckets $L_k[g]$, $k = 1,\dots,m$ for all g such that $||f^{(k)} - g|| \le \lfloor r/m \rfloor$ and test whether the distance to f is not greater than r.

There is also an algorithm variant in which we increase the radius until we find a pre-defined number of descriptors to ensure the required number of neighbors.

If we assume that the distribution of descriptors across buckets is uniform, then the complexity of the search and testing is as follows:

$$cost(r) \le (1 + n/2^s)m2^{H(r/b)s},$$

where $s = b/m$ is the subsequence length, n is the number of descriptors in dataset and $H(\varepsilon) = \varepsilon\log_2\varepsilon - (1 - -\varepsilon)\log_2(1-\varepsilon)$ is the entropy of Bernoulli distribution with probability ε as shown in [5].

However, the actual distribution of the descriptors across the buckets can depend on the descriptor type, task features, and available data, and may be non-uniform. In practice, large buckets can lead to a significant increase in response time.

5. Heuristic multi-index hashing

The main idea of the proposed heuristic multi-index hashing is to limit a maximum number of descriptor comparisons p^{max}, i.e., Hamming distance computations, to constrain the response time strictly.

Let us define descriptor length as b, the number of its subsequences as m, and the length of one subsequence as $s = b/m$. For the input descriptor $f = \{f^{(k)}\}$, $k = 1,\dots,m$ and radius for neighbor search r we have to consider hash buckets $L_m[g^{(k)}]$, such that

$$||f^{(k)} - g^{(k)}|| \le \lfloor r/m \rfloor$$

However, the number of descriptors to test in these buckets is not bound by the above in the case of non-uniform descriptor distribution and may be large. So, we introduce two heuristics to limit it and satisfy the restriction above.

5.1. Bucket size constraint

Our first heuristic is targeted to constrain the bucket size. We define Bucket Size BS as the threshold for the number of elements in a bucket to test. So, there are two cases:

1. Small bucket. There are fewer than BS elements in the bucket, and we examine all of them.
2. Large bucket. There are more than BS elements in the bucket, and we only take BS elements from it. In the implementation of the proposed method from each large bucket, we sample BS elements so that they are equally spaced.

We take $BS = p^{max}/m$ to ensure at most p^{max} Hamming distance computations. The theoretical justification for this heuristic is that when $BS > n/2^s$, we remove only the elements that do not correspond to the uniform distribution. The number of these elements is expected to be small, as having a significant number of descriptors with similar values can negatively impact matching accuracy and may suggest the need for fine-tuning the descriptors.

5.2. Bucket number constraint

Our second heuristic limits the number of buckets to examine. We consider two variants.

1. Direct Match (DM): we look thorough the elements in $L_m[f^{(k)}]$ only. This way, we guarantee that we examine all the neighbors in radius $m - 1$ from the input descriptor f.
2. Almost Match (AM): after examining the elements in $L_m[f^{(k)}]$ we also examine neighboring buckets $L_m[g^{(k)}]$ with the distance 1:

$$||f^{(k)} - g^{(k)}|| \le 1$$

However, we ensure that the number of Hamming distance computations is limited, so we:

- ignore large buckets;
- stop when we reach p^{max}/m tests.

This heuristic can be explained as follows: when determining the document type, we aim to identify the closest descriptor for each keypoint rather than to find all the nearest neighbors. If no similar descriptor is found, it likely indicates that this keypoint does not appear on any template, making it not informative for type identification. Additionally, large buckets suggest that specific bit sequences appear in many descriptors, offering little information. As a result, these large buckets are excluded in favor of testing more distinctive buckets.

5.3. Initialization

5.3.1. System startup

We initialize m tables on system startup, as in the original multi-index hash algorithm, using descriptors from all available document templates. So, we put into tables pairs of (f_i^j, i), $i = \{1, \dots, N\}$, $j = \{1, \dots, M\}$. Thus, we can determine the type of document C_i from which each descriptor originates. So, in general, we need to keep mMN entries.

5.3.2. Configuration for a subset.

During matching, we need to access the elements of hash tables L_m only for a subset of document types D. To do this, we create new hash tables $L_{m'}$ and add pointers to the necessary elements in the original tables. We can estimate the current bucket sizes in $L_{m'}$ during this operation and apply the bucket size constraint. It ensures we keep at most BS elements in each bucket. We perform $O(Dm)$ operations, so the complexity of the configuration does not depend on the total number of document types.

6. Experimental evaluation

6.1. Experimental Setup

6.1.1. Datasets

We conduct experiments on public MIDV-500 [6] and MIDV-2019 [7] datasets. MIDV-500 contains frames of 500 video clips with identity documents of 50 different types (17 ID cards, 14 passports, 13 driving licences, and 6 other identity documents of different countries) in 1080×1920 resolution and corresponding ideal distortion-free document images (one per type). MIDV-2019 expands this dataset with images with complex lighting conditions and projective distortions in a higher resolution 2160×3840. It adds 200 more video clips. Examples of images are shown in Fig. 2.

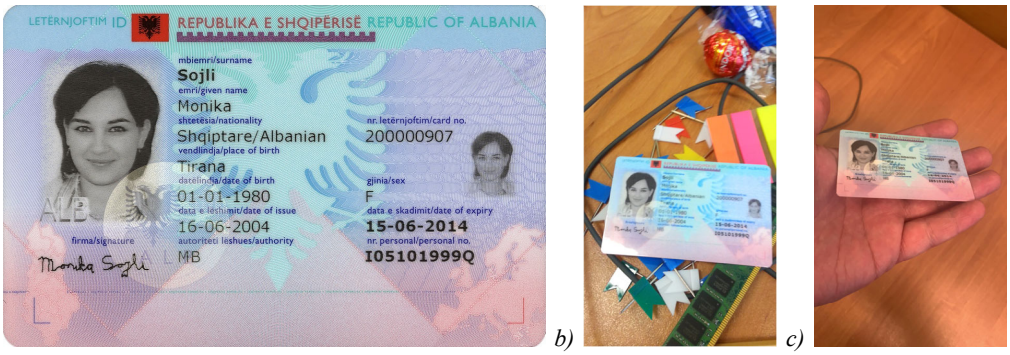


Fig. 2. The sample images: the ideal template image (a), the image from MIDV-500 (b), and the image from MIDV-2019 (c)

6.1.2. Methods

In our experiments, we consider different methods for the nearest neighbor search in the feature-matching stage of the above pipeline:

- **HMIH**: proposed heuristic multi-index hashing with descriptor length of $b = 128$ bits, $s = 16$ bits per subsequence and $m = 8$ tables. We use $r = 48$.
- **BF**: brute-force matching (linear search through all descriptors from ideal templates); it requires no time for configuration;
- **HC**: hierarchical clustering from FLANN matcher (with hierarchical k-means tree inside) in OpenCV with default parameters; we constructed an index for the current document types subset at the configuration stage,
- **LSH**: multi-probe local sensitive hashing from FLANN matcher in OpenCV with default parameters; we constructed tables for the current document types subset at the configuration stage.
- **HNSW**: hierarchical navigable small world graph search from FAISS library [28].

6.1.3. Hardware

We measure configuration and matching time on an AMD Ryzen 9 5950X CPU with the x86_64 architecture without multi-threading. Time is scaled over the number of keypoints in the query image and averaged over all the images in the dataset.

6.2. Measurements and results

In this section, we estimate optimal parameters for the proposed heuristics and then measure document classification accuracy and matching time.

In our first experiment, we investigate the proposed HMIH algorithm's Bucket Size (BS) parameter. The algorithm was tested in Almost Match (AM) mode with BS values of 16, 32, 64, and 128 and in Direct Match (DM) mode without

any constraints. To find the appropriate parameter value, we measured the average matching time per image and classification accuracy per frame for all frames in the MIDV-2019 dataset. In this experiment, we use all 50 document types at system startup and vary the size of the current subset.

In this case, the "frame" refers to a single frame from the corresponding MIDV dataset clip (there are 30 frames per clip). This frame is the input (Q) of the pipeline, presented in Section 3. However, feature-matching obtains one of the alternative quadrangles, found by [25] algorithm as an input, or the whole frame. So, we denote as "image" the source of query keypoints and descriptors to be matched, and there can be several images per frame.

In Fig. 3, we present the experimental results. This figure shows that the Direct Match mode is significantly faster than the Almost Match mode; however, it leads to a noticeable decrease in classification accuracy. For instance, Direct Match is about 4 times faster than Almost Match with BS $= 32$, but the accuracy of DM is 3 – 4% lower than that of AM.

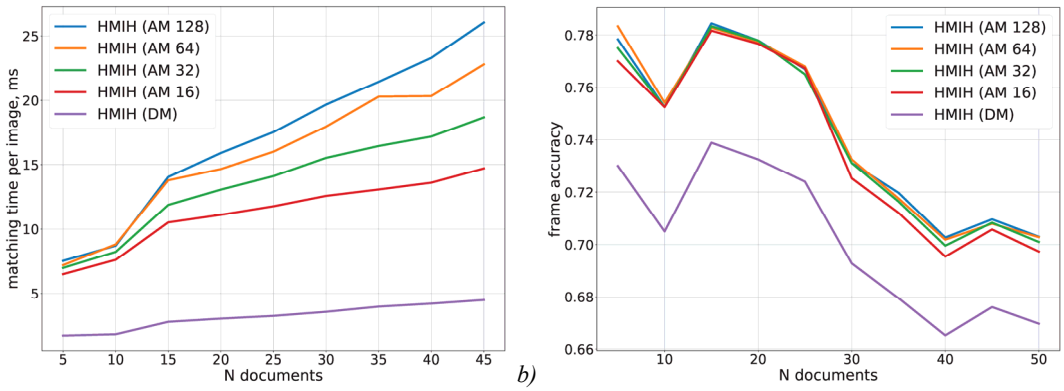


Fig. 3. Bucket Size impact on of the average matching time (a) and classification accuracy (b) for MIDV-2019 dataset

Also, Fig. 3 shows that all AM algorithms result in almost the same quality. However, the quality of BS = 16 is slightly worse than the others. So, we chose BS = 32 as a baseline parameter for the HMIH algorithm in AM mode. Another reason to choose this parameter is the bucket size distribution presented in Fig. 4. We can see that this threshold separates the majority of small buckets from a few large ones. In the case of the MIDV-500 dataset, 8 HMIH tables contain only 219 buckets, whose size is greater than 32, but those buckets contain 14.8k descriptors out of 175 k descriptors in total. Thus, less than 0.3 % of non-empty buckets contain more than 8 % of descriptors. Consequently, the actual distribution differs from the assumed uniformity in the theoretical complexity assessment. As a result, the bucket size constraint aligns the real data more closely with the considered model and can significantly influence the running time.

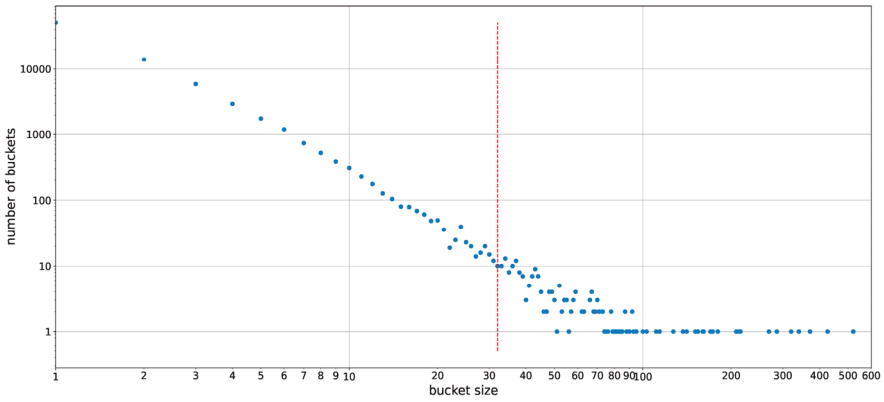
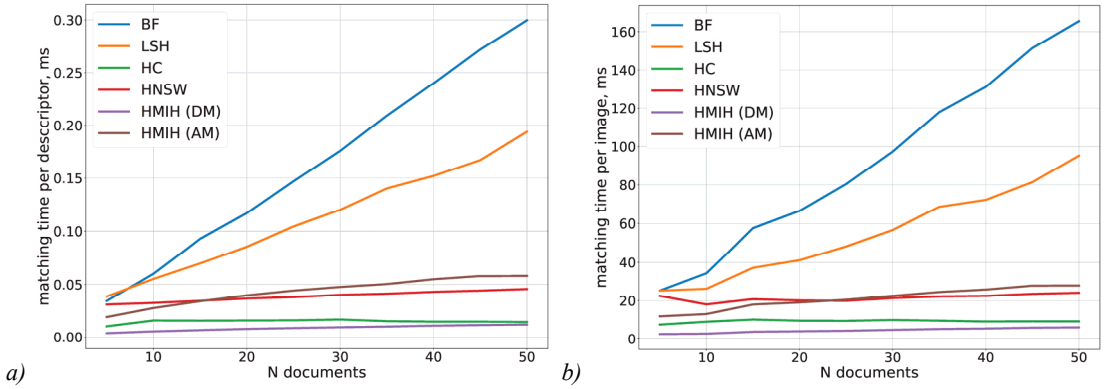


Fig. 4. Bucket Size distribution in HMIH constructed for 50 documents of MIDV-500 and MIDV-2019 datasets. The vertical line shows $BS = 32$

Our second experiment compares the matching time and classification accuracy of HMIH with other methods. We test HMIH in Almost Match (AM) and Direct Match modes with $BS = 32$. The matching times are shown in Fig. 5 and 6. We can see that HMIH in DM mode and hierarchical clustering have comparable matching times, with HMIH in DM mode being negligibly faster. The proposed HMIH in AM mode significantly outperformed multi-probe locality-sensitive hashing (LSH) and brute-force. However, HNSW was faster for medium and large document subsets. Surprisingly, brute-force has slightly fewer matching times for a small number of the current document types than LSH on MIDV-500. On MIDV – 2019, brute-force outperformed it almost everywhere. This result can be explained by the absence of computational overhead for brute-force and inefficient parameters of LSH for the considered problem.

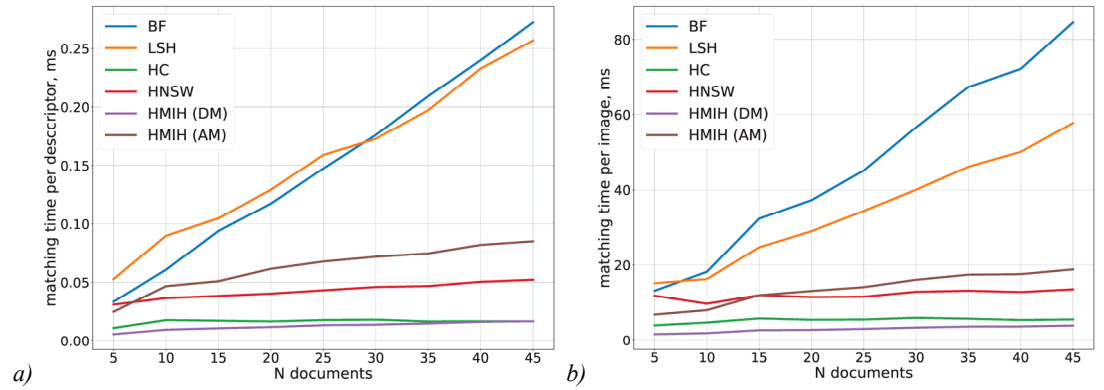
The classification accuracies are shown in Fig. 7. We demonstrate per-frame classification accuracy when we treat each frame from a video clip as a separate image. Regarding accuracy, both brute-force, LSH, and HNSW approaches perform similarly on both datasets. HMIH AM has slightly lower accuracy (–1%), and the HMIH DM and hierarchical

clustering are significantly worse (– 5 %). However, HMIH DM gives noticeably better results than hierarchical clustering for a large number of (more than 30) document types enabled on MIDV– 500 and only 2 % better results on MIDV – 2019.



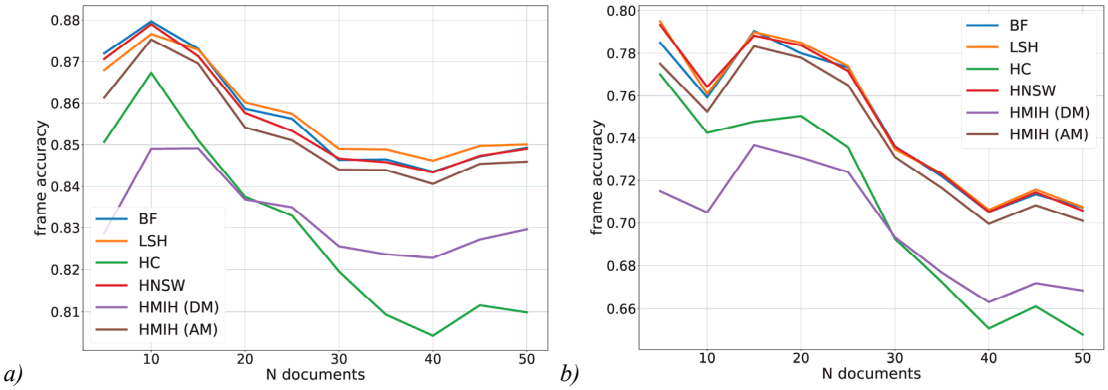
a) *b)*

Fig. 5. The average matching time for the MIDV-500 dataset depending on the number of documents in the subset: a) normalized time per input descriptor, b) the average matching time per image



a) *b)*

Fig. 6. The average matching time for the MIDV-2019 dataset depending on the number of documents in the subset: a) the normalized time per input descriptor, b) the average matching time per image



a) *b)*

Fig. 7. The dependence of frame classification accuracies on the number of documents in the current subset for the MIDV-500 (a) and MIDV-2019 (b) datasets

Finally, we measured configuration time for all considered methods. The results are demonstrated in Fig. 8. We can see that the startup time for HMIH is not so small. However, this time is needed only once and does not affect user experience. The hierarchical navigable small-world graph approach had the most significant configuration time and was more than 100 times slower than the HMIH configuration. The reason is that it creates the graph for the current document subset for each launch. The same is observed for hierarchical clustering, which is 3 times slower. The matching time and accuracy results indicate that hierarchical clustering is inferior to HMIH. The configuration time of HNSW makes it unusable for online graph construction for a subset of documents. LSH has inconsiderable configuration time, but the longest matching time is inappropriate for document type identification with default parameters.

Thus, the proposed HMIH with an Almost Match heuristic and a Bucket Size limit of 32, offers a great combination of accuracy and computational efficiency in the considered task and can be successfully applied in document recognition systems. Note that these results, of course, can further be enhanced by implementing a more advanced recognition pipeline (like adding detection of false responses [29]).

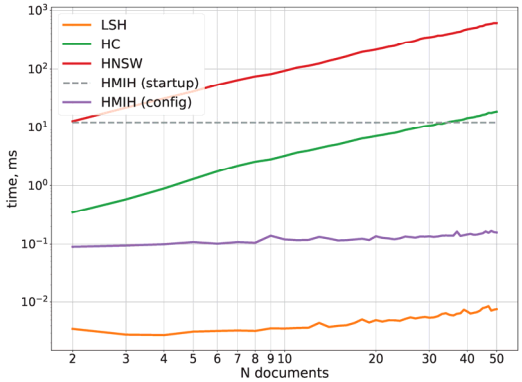


Fig. 8. Configuration time for different matching algorithms depending on the number of documents in the current subset in the MIDV-500 dataset

7. Discussion

In this paper, we examined an ID document recognition system that can select a subset of possible document types on-the-fly. For such a system, it is crucial to quickly configure a data structure for feature-based document type identification and enable searches within it in constant time. This challenge is highly relevant, as modern recognition systems must operate rapidly on various devices and support thousands of document types.

We proposed a solution based on multi-index hashing, enhanced by two heuristics to limit the number of descriptor comparisons. Our findings indicate that constraining the bucket size is significant in practice due to the non-uniform distribution of descriptors across the buckets. Experiments show that the heuristic constraint of the bucket size has a minimal impact on the accuracy of type identification with appropriately adjusted parameters. The reduced number of examined buckets results in a slight decrease in accuracy compared to a brute-force search, but significantly improves processing speed.

In our case, selecting a specific subset of document types does not include a complete and time-consuming rebuild of the search structure, as hierarchical clustering or navigable small-world graphs do. However, we face a longer configuration time than multi-probe local sensitive hashing, which gives a noticeably worse typing accuracy. It illustrates the classic trade-off between accuracy and time.

The main **limitation** of this study is the absence of experiments with a large number of document types. The scale of a real-world ID document recognition system is likely to exceed 50 documents. For example, the Brazilian Identity Document Dataset [30] contains only eight different document types, and the IDNet dataset [31] offers 20 types, but consists of synthesized images. Nevertheless, the observed trends suggest that even with a larger number of types, the proposed solution, based on heuristic multi-index hashing, is expected to maintain high matching accuracy with a relatively fast configuration.

The results of this research are promising for tasks that require feature selection followed by fast matching. In these situations, using a multi-index hash combined with proposed heuristics can dramatically reduce operational time, which is crucial for real-time applications. One example of such a task is visual localization, where we have prior knowledge about the continuity of the camera's trajectory and its speed. This understanding enables us to limit the database size to be searched, thereby minimizing the computational resources required. This area of research is especially important for **future work**.

8. Conclusion

This work proposed a new method for heuristic search of nearest descriptors based on multi-index hashing, which can be initialized on-the-fly for a selected subset of documents. The proposed heuristics include constraints on bucket size and the number of examined buckets, defined by two variants: direct match and almost match options.

We focused on a specific part of the ID document recognition system: document type identification. Our experiments were conducted on two datasets, MIDV – 2019 and MIDV – 500, which contain ID documents of 50 different types.

Our results showed that the brute-force, locality-sensitive hashing, and hierarchical navigable small-world graph approaches performed similarly on both datasets and achieved the highest accuracy across all examined document subsets. The proposed HMIH AM method with a bucket size of 32 had 1 % lower accuracy, while the HMIH DM and hierarchical clustering techniques were approximately 5 % less accurate. At the same time, the HMIH in DM mode and hierarchical clustering provided the fastest matching times, with HMIH DM being slightly faster. HMIH AM significantly outperformed locality-sensitive hashing and brute-force methods. It was marginally slower than hierarchical navigable small-world graphs when handling document subsets containing fewer than 15 documents. However, when analyzing a document subset, the HMIH configuration was over 100 times faster than the hierarchical navigable small-world graphs configuration and 3 times faster than the hierarchical clustering configuration. Locality-sensitive hashing configuration was much faster, but it required significantly more time for matching, making it less valuable.

Therefore, the proposed method can be used in computationally intensive ID document recognition systems, providing accurate type identification and time-efficient subset selection capability.

References

[1] ICAO, Doc 9303, Machine Readable Travel Documents.https://www.icao.int/publications/pages/publication.aspx?docnum=9303 (Accessed on 27.06.2025)

[2] A. M. Awal, N. Ghanmi, R. Sicre, T. Furon, Complex Document Classification and Localization Application on Identity Document Images, in: ICDAR 2017, Vol. 1, 2017, pp. 426–431, dOI: 10.1109/ICDAR.2017.77

[3] K. Bulatov, V. V. Arlazarov, T. Chernov, O. Slavin, D. Nikolaev, Smart IDReader: Document Recognition in Video Stream, in: ICDAR 2017, Vol. 6, 2017, pp. 39–44, dOI: 10.1109/ICDAR.2017.347.

[4] N. Skoryukina, V. V. Arlazarov, D. P. Nikolaev, Fast method of ID documents location and type identification for mobile and server application, in: ICDAR 2019, 2020, pp. 850–857, dOI: 10.1109/ICDAR.2019.00141.

[5] M. Norouzi, A. Punjani, D. J. Fleet, Fast Exact Search in Hamming Space With Multi-Index Hashing, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36, 2013, pp. 1107–1119, dOI: 10.1109/TPAMI.2013.231.

[6] V. V. Arlazarov, K. Bulatov, T. Chernov, V. L. Arlazarov, MIDV-500: A Dataset for Identity Document Analysis and Recognition on Mobile Devices in Video Stream, Computer Optics 43 (5) (2019) 818–824, dOI: 10.18287/2412-6179-2019-43-5-818-824.

[7] K. Bulatov, D. Matalov, V. V. Arlazarov, MIDV-2019: Challenges of the Modern Mobile-Based Document OCR, in: ICMV 2019, Vol. 11433, 2020, pp. 114332N1–114332N6, dOI: 10.1117/12.2558438.

[8] Q. Lv, W. K. Josephson, Z. Wang, M. Charikar, K. Li, Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search, in: Very Large Data Bases Conference, 2007.

[9] M. Muja, D. G. Lowe, Scalable Nearest Neighbor Algorithms for High Dimensional Data, IEEE Transactions on Pattern Analysis and Machine Intelligence, 36 2014, pp. 2227–2240, dOI: 10.1109/TPAMI.2014.2321376.

[10] 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 (4), 2018, pp. 824–836, dOI: 10.1109/TPAMI.2018.2889473.

[11] T. Nakai, K. Kise, M. Iwamura, Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval, in: Document Analysis Systems VII, 2006, pp. 541–552, dOI: 10.1007/11669487_48.

[12] P. Gomez-Krämer, K. Rouis, A. O. Diallo, M. Coustaty, Printed and scanned document authentication using robust layout descriptor matching, Multim. Tools Appl. 83 (2023), pp. 47477–47502, dOI: 10.1007/s11042-023-17021-1.

[13] S. Eskenazi, P. Gomez-Krämer, J.-M. Ogier, The Delaunay Document Layout Descriptor, in: Proceedings of the 2015 ACM Symposium on Document Engineering, 2015, pp.167–175, dOI: 10.1145/2682571.2797059.

[14] N. D. Moskin, K. A. Kulakov, A. A. Rogov, R. V. Abramov, Research the Stability of Decision Trees Using Distances on Graphs, Proceedings of the Instittute for Systems Analysis Russian Academy of Sciences (ISA RAS) 73 (1), 2023, pp. 94–100, dOI: 10.14357/20790279230111.

[15] E. Rublee, V. Rabaud, K. Konolige, G. R. Bradski, ORB: An efficient alternative to SIFT or SURF, 2011 International Conference on Computer Vision, 2011, pp. 2564–2571, dOI: 10.1109/ICCV.2011.6126544.

[16] I. Suárez, J. M. Buenaposada, L. Baumela, Revisiting Binary Local Image Description for Resource Limited Devices, IEEE Robotics and Automation Letters 6, 2021, pp. 8317–8324, dOI: 10.1109/LRA.2021.3107024.

[17] B. Fan, Q. Kong, T. Trzcinski, Z. Wang, C. Pan, P. V. Fua, Receptive Fields Selection for Binary Feature Description, IEEE Transactions on Image Processing 23, 2014, pp. 2583–2595, dOI: 10.1109/TIP.2014.2317981.

[18] M. Muja, D. G. Lowe, Fast Matching of Binary Features, 2012 Ninth Conference on Computer and Robot Vision, 2012, pp. 404–410, dOI: 10.1109/CRV.2012.60.

[19] D. Schlegel, G. Grisetti, HBST: A Hamming Distance Embedding Binary Search Tree for Feature-Based Visual Place Recognition, IEEE Robotics and Automation Letters 3, 2018, pp. 3741–3748, dOI: 10.1109/LRA.2018.2856542.

[20] D. Schlegel, G. Grisetti, Visual localization and loop closing using decision trees and binary features, 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4616–4623, dOI: 10.1109/IROS.2016.7759679.

[21] B. Fan, Q. Kong, B. Zhang, H. Liu, C. Pan, J. Lu, Efficient nearest neighbor search in high dimensional hamming space, Pattern Recognit. 99, 2020, p. 107082, dOI: 10.1016/j.patcog.2019.107082.

[22] Y.-C. Liaw, M.-L. Leou, C.-M. Wu, Fast exact k nearest neighbors search using an orthogonal search tree, Pattern Recognition 43, 2010, pp. 2351–2358, dOI: 10.1016/j.patcog.2010.01.003.

[23] A. Lukoyanov, D. Nikolaev, I. Konovalenko, Modification of YAPE keypoint detection algorithm for wide local contrast range images, in: ICMV 2017, Vol. 10696, 2018, pp. 1069616–1–1069616–8, dOI: 10.1117/12.2310243.

[24] D. P. Matalov, E. E. Limonova, N. S. Skoryukina, V. V. Arlazarov, Memory effcient local features descriptor for identity document detection on mobile and embedded devices, IEEE Access 11, 2022, pp. 1104–1114, dOI: 10.1109/ACCESS.2022.3233463.

[25] D. V. Tropin, S. A. Ilyukhin, D. P. Nikolaev, V. V. Arlazarov, Approach for document detection by contours and contrasts, in: ICPR 2020, 2021, pp. 9689–9695, dOI: 10.1109/ICPR48806.2021.9413271.

[26] A. V. Trusov, E. E. Limonova, V. V. Arlazarov, Fast computation of RFD-like descriptors in four orientations, IEEE Access 11, 2023, pp. 19725–19740, dOI: 10.1109/ACCESS.2023.3249100.

[27] E. O. Rybakova, A. V. Trusov, E. E. Limonova, N. S. Skoryukina, K. B. Bulatov, D. P. Nikolaev, PESAC, the Generalized Framework for RANSAC-Based Methods on SIMD Computing Platforms, IEEE Access 11, 2023, pp. 82151–82166, dOI: 10.1109/ACCESS.2023.3301777.

[28] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, H. Jégou, The Faiss library (2024).arXiv:2401.08281 .

[29] N. S. Skoryukina, E. A. Shalnova, V. V. Arlazarov, Method for Detecting False Responses of Localization and Identification Algorithms Using Global Features, ITiVS (4), 2023, pp. 28–36, dOI: 10.14357/20718632230403.
[30] A. de Sá Soares, R. B. das Neves Junior, B. L. D. Bezerra, BID dataset: a challenge dataset for document processing tasks, Conference on Graphics, Patterns and Images (SIBGRAPI), SBC, 2020, pp. 143–146.
[31] H. Guan, Y. Wang, L. Xie, S. Nag, R. Goel, N. E. N. Swamy, Y. Yang, C. Xiao, J. Prisby, R. Maciejewski, J. Zou, IDNet: A novel dataset for identity document analysis and fraud detection, arXiv preprint arXiv:2408.01690 (2024).

Author's information

Elena Evgenevna Limonova (b. 1993) completed Master's degree (2017) in Applied Mathematics and Physics at Moscow Institute of Physics and Technology. Received her Ph.D. degree in Computer Science in 2023 at FRC "Computer Science and Control" RAS. Currently, she works as a researcher at the FRC "Computer Science and Control" RAS and as a programmer at Smart Engines Service LLC. Research interests: computer vision, artificial neural networks, image recognition on mobile devices. E-mail:*limonova@smartengines.com*

Anton Vsevolodovich Trusov (b. 1997) completed Master's degree in Applied Mathematics and Physics at Moscow Institute of Physics and Technology in 2021, since then he has been a Ph.D. student at this institute. He also works as a programmer at FRC "Computer Science and Control" and at Smart Engines Service LLC. Research interests: computer vision algorithms, artificial neural networks, high-performance computing, and adversarial deep learning. E-mail:*trusov.av@smartengines.com*

Daniil Zlatomirovich Rybalko (b. 1999) obtained Masters degree from MIPT in 2023. Currently works as software engineer at Smart Engines Service LCC. Research interests: image processing, pattern recognition, computer vision. E-mail:*d.rybalko@smartengines.com*

Natalya Sergeevna Skoryukina (b.1991), PhD degree in technical sciences (2025). She graduated from National University of Science and Technology "MISiS" in 2013, majoring in Applied Mathematics. Research fellow at the Federal Research Center "Computer Science and Control," Russian Academy of Sciences. Research interests: image analysis, computer vision.*E-mail: skleppy.inc@smartengines.com*

Konstantin Bulatovich Bulatov (b. 1991) received the Specialist degree in Applied Mathematics from the National University of Science and Technology "MISiS" in 2013, and the Ph.D. degree in Computer Science from the FRC "Computer Science and Control" RAS in 2020. Since 2014, he has been employed with the FRC CSC RAS. Since 2016, he has been employed with Smart Engines Service LLC, Moscow. Research interests: computer vision, image processing, and document recognition systems. E-mail: *kbulatov@smartengines.com*

Received June 27, 2025. The final version – September 18, 2025