American Journal of Computer Science and Technology
2021; 4(2): 38-45
http://www.sciencepublishinggroup.com/j/ajcst
doi: 10.11648/j.ajcst.20210402.12
ISSN: 2640-0111 (Print); ISSN: 2640-012X (Online)

Effective Geometric Hashing of the Feature Hyperspace for a Quick Accurate Search of the Most Similar Descriptors in Large Datasets

Dmitry Pozdnyakov

Deep Learning Department, Closed Joint Stock Company "Oxagile", Minsk, Belarus

Email address:
pozdnyakov@tut.by, dmitry.pozdnyakov@oxagile.com

To cite this article:
Dmitry Pozdnyakov. Effective Geometric Hashing of the Feature Hyperspace for a Quick Accurate Search of the Most Similar Descriptors in Large Datasets. *American Journal of Computer Science and Technology*. Vol. 4, No. 2, 2021, pp. 38-45. doi: 10.11648/j.ajcst.20210402.12

Received: June 25, 2021; **Accepted**: July 16, 2021; **Published**: July 24, 2021

Abstract: The algorithm of effective geometric hashing of the facial feature hyperspace for the accelerated search of the most similar face descriptors by their cosine similarity is described in the present study. The algorithm includes 6 required stages of processing descriptors extracted by a neural network from face images. The first stage is filtration of the descriptor database by selecting the most representative descriptor for each person from the set of descriptors corresponding to his/her different face images. The second stage is evaluation of a number of statistical values for all the components of the selected descriptors. The third stage is intermediate hashing through quantization of every descriptor component value so that almost the same quantity of descriptors corresponds to any quantum number. The fourth stage is statistical processing of the descriptor database to determine the most discriminative descriptor key components and their hierarchy. The fifth stage is calculation of the descriptor hash code for every most representative descriptor from the considered database. The sixth final stage is a special cataloging of data in the form of a multi-tiered directory ordered by the hash codes. The search acceleration is achieved through sparse processing of the whole directory when the hash code obtained for the requested person descriptor acts as a very selective search filter. The developed algorithm always provides the same absolute accuracy as the brute-force search. Through the example of the LFW dataset consideration, the average search acceleration by about 100 times is achieved under conditions that the descriptors have been extracted by a neural network trained on the WiderFace dataset with application of the additive angular margin loss function.

Keywords: Geometric Hashing, Brute Force, Search, Feature Hyperspace, Descriptor, Cosine Similarity

1. Introduction

To find out the presence of the requested person image in the database, the automated methods of face images comparison have become especially popular in recent years. And when there is a large number of face images in the database, the time spent on searching for the most similar face image from the database to the requested person face image is extremely critical. The problem of quick processing of the face images database even transformed into a separate domain of scientific research [1]. At present, the most popular method for encoding the distinctive facial features is neural network extraction of the corresponding descriptor (a face feature vector usually normalized to the unity) from the multidimensional face features hyperspace [2–5]. This approach provides the possibility to reduce the face images comparison to the formal comparison of their descriptors applying one metric or another. In particular, the typical Euclidean distance can be applied as the first approximation or, as the opposite, some exotic metrics can be used (see, for example, [6, 7]). Whereas, the most applied measure of similarity is the cosine distance [7, 8]. In any case, despite the metric type, the problem of searching for a face image from a dataset, that is most similar to the face image of the requested person, is reduced to the search of the nearest neighbor point in an appropriate metric sense to the determined point of the multidimensional face features hyperspace [9, 10]. And such a problem can be accurately

solved by means of exhaustive (brute-force) search through comparison of the descriptors by their cosine similarity. When the number of face images in the database is equal to M, then the calculation complexity of the search algorithm is obviously proportional to M. If M is a very large value, then it takes too much time to brute force, which is often unacceptable. Moreover, starting from some values of M, the search algorithm parallelization as well as its transfer to GPU [10–12] are not already effective ways of its acceleration because of hardware limitations. The natural method of solution to such a problem is a special organization of data in the database [13, 14] (indexing, hashing, etc. [15, 16]) that allows the search process to be drastically accelerated. There are very many concrete implementations of the approach like that (see, for example, [10, 18–31]). But all of them allow the nearest neighbor to be found either approximately or with the allocation of large memory volume for index information and the necessity of the database re-indexing with a very high computational complexity.

Taking into account the aforesaid, a logical question arises: is it possible, in principle, by means of quite a simple calculation without the need of allocation of large memory volume to index the database elements in such a way that the application of some search algorithm guarantees the nearest neighbor could be found accurately with such an average search time value which at least several times lower than the time of exhaustive search over all the database elements [32]? The present study is just devoted to this question on an example of the solution to the problem of descriptor-assisted search for a face image from the database that is most similar to the face image of the requested person.

2. Theory and Results

2.1. General Issues

As a methodological basis for indexing the descriptors from a face image database, a well-known algorithm of geometric hashing [16] will be applied after a number of its modifications.

So, let the descriptors $\mathrm{d}_{ij}=(d_1, d_2, \ldots, d_n, \ldots, d_N)_{ij}^{\mathrm{T}}$ be the vectors with unity length from the face features space with dimension N=512 (it can be higher or lower), which are extracted from the face images by a neural network, as an example, ArcFace with the ResNet backbone [5] that itself has been trained on a large face image dataset, for example, WiderFace [33].

The geometric hashing algorithm does not allow the nearest neighbor vector to be found accurately in the case when a requested vector belongs to the vicinity of multidimensional bound between feature vector clusters. Thus, it is necessary to apply some modifications to the classical geometric hashing algorithm. Further, such a well-known face image dataset as LFW [34] will be considered to concretize all the examples. All the stages, which are necessary for the preparation of the face image database, descriptors hashing, and accelerated index search, will be described below.

2.2. Descriptors Filtration

The search of the most representative descriptor $\mathrm{D}_i \in \{\mathrm{d}_{ij}\}$ for each i-th person is realized over all the descriptors in the database according to [35]. Namely, the most representative descriptor is matched to every person as the only personal feature vector identifier. As a result, the number of descriptors in the database will coincide with the number of persons in the database. In general, the database can contain many times more images than the number of persons (several face images can correspond to one person).

2.3. Statistical Evaluations

Some additional values should be calculated

$$\mathrm{MD} = \left(\underset{i}{\mathrm{med}}\{D_{1,i}\}, \ldots, \underset{i}{\mathrm{med}}\{D_{n,i}\}, \ldots, \underset{i}{\mathrm{med}}\{D_{N,i}\} \right)^{\mathrm{T}}, \quad (1)$$

$$\overline{\Delta \mathrm{D}} = \left(\overline{\Delta D_1}, \ldots, \overline{\Delta D_n}, \ldots, \overline{\Delta D_N} \right)^{\mathrm{T}}, \quad (2)$$

$$\overline{\Delta D_n} = I^{-1} \sum_{i=1}^{I} \left| D_{n,i} - MD_n \right|. \quad (3)$$

In (1), to evaluate the mode of the Gaussian distribution, its median estimation is applied instead of the average estimation to reduce the influence of statistical outliers on calculation results (if there are very many descriptors in the database, the average evaluation could be applied to reduce the median calculation complexity).

2.4. Intermediate Hashing

Taking into account the Gaussian distribution of face features for each component of the feature vector, the quantization of the vector components values is realized for every component of every descriptor by means of erf-function in such a way so that the statistically equivalent number of face images from the database corresponds to every integer quantum number. In particular, each descriptor (hash key) D_i is associated with the corresponding vector (intermediate hash code) P_i (see (4) and (5) below).

To determine the optimal bit depth for the components of the intermediate hash code (number of partitions/cells/bins) Q (q=0, …, Q), let us proceed from the following assumptions. If it is necessary to find accurately the nearest neighbor for the requested vector at least locally for some concrete vector component in the framework of the projective approximation [35], not only the descriptors with the same hash code component q_0 must be compared with the requested descriptor but also the descriptors from the neighbor cells ($q_0 \pm 1$) should be compared to exclude the problem of boundaries vicinity. For example, when some component is divided into 3 cells, it is necessary to check all the descriptors in 1 out of 3 outcomes. In this case, the gain in the index search will be insignificant in comparison with brute force. To obtain considerable acceleration by the index search, the component domain should be divided into many cells. And the more cells the domain is divided into, the less part of the descriptors belongs to cells $q_0 - 1$, q_0, and $q_0 + 1$ relative to all descriptors from other cells. Due

to that, the average index search speed for one vector component is $(Q+1)^2/(3Q+1)$ times higher in comparison with brute force. When R representative descriptor components are considered, the search based on the hash codes are $(Q+1)^{2R}/(3Q+1)^R$ times faster than the exhaustive search, on average. But because of the application of the projective approximation, the decrease of the cell size leads to the reduction of the general accuracy in correspondence with the increase of a number of descriptor components used for hashing (the increase of R-value sharply leads to the increase of the probability of finding the most similar descriptor to the requested one outside the set of search). Thus, the optimal partitions are those for which Q takes values from 4 to 7. Further, the balanced partition at Q=4 will be considered (in most cases the number of not checked cells is equal to the number of checked cells when the same cell is neglected). This provides the highest general accuracy for the search algorithm.

So, $P_{n,i}$ for $D_{n,i}$ are defined by

$$\Theta_{n,i} = \frac{D_{n,i} - MD_n}{\overline{\Delta D_n}} \tag{4}$$

and

$$\begin{cases} P_{n,i} = 0 \ \ \forall\, \Theta_{n,i} \le -1.055; \\ P_{n,i} = 1 \ \ \forall\, \Theta_{n,i} \in (-1.055, -0.318); \\ P_{n,i} = 2 \ \ \forall\, \Theta_{n,i} \in [-0.318, 0.318]; \\ P_{n,i} = 3 \ \ \forall\, \Theta_{n,i} \in (0.318, 1.055); \\ P_{n,i} = 4 \ \ \forall\, \Theta_{n,i} \ge 1.055. \end{cases} \tag{5}$$

As mentioned earlier, the numerical values of intervals in (5) are obtained by inversion of the erf-function so that every interval contains, on average, almost the same quantity of descriptors in accordance with the values of their vector components. To increase the algorithm robustness to statistical outliers, the partition into intervals is done through the average deviation of every descriptor component from the mode of its normal distribution instead of the root-mean-square deviation of the descriptor component from the mode (see (4)).

As an example, for one of the descriptors (– 0.002924, 0.017627, 0.007543, 0.036980, …)^T corresponding to one of the face images from LFW, the intermediate hash code (1, 2, 2, 4, …)^T which is N-dimensional vector with the integer values for its components can be obtained.

2.5. Descriptor Key Components and Their Hierarchy

Further, to determine the descriptor hash code, which is an ordered selection of the key components from the descriptor intermediate hash code, a pairwise comparison of the whole set of the descriptors is realized by their cosine similarity Cs (this corresponds to the dot product for the descriptors normalized to the unity length). To do that, some cosine similarity threshold level Cth is selected, for example, equal to 0.5. Then, over all the descriptors in the database the number of descriptor pairs D_i and $D_{i'}$ is calculated for which the cosine similarity is equal or higher than the considered level ($D_i\, D_{i'} \ge Cth$). During that, for each descriptor component the part from the selected descriptors pairs ε_n is calculated under the condition that the difference between the corresponding hash code component values is equal to one or less ($|P_{n,i} - P_{n,i'}| \le 1$), i.e. it is ascertained if the descriptors belong to the same local hash neighborhoods. Further, for all descriptor components the condition of equality of ε_n to 1 is checked ($\exists\ n$: ε_n=1?). If the condition is not fulfilled for any descriptor component, the value of Cth is incremented, for example, by 0.05 and the check is performed again. And so on until some descriptor component is found for which ε_n=1. In the case when the condition of ε_n=1 is fulfilled for several values of n, at the next increment step the previous increment is reduced, as an example, by half, and by analogy with the bisection method [36] such a level $Cth^{(1)}$ is found for which the only descriptor component is equal to one, but it is not necessary to find exactly the level of jump of ε_n from some value to 1. When, in some cases, due to insufficient diversity of face images in the database leading to a significant discreetness of the number of calculated parameters, it is impossible to divide the current interval during the reasonable number of iterations so that between $Cth^{(1)}$ and $Cth^{(2)}$ the condition of ε_n=1 fulfills for the only descriptor component, for such a case of multiple jumps of ε to 1 at the same level Cth it is supposed that $Cth^{(2)}$=$Cth^{(3)}$=… with any hierarchy by corresponding values of n. Then, taking into account all preliminary iterations by Cth that could enforce a condition of ε_n=1 for several values of n, such all the levels $Cth^{(r)}$ (r=1, 2, …, R) should be found that meet the condition: all faces images from the database with cosine similarity between their descriptors greater than or equal to $Cth^{(r)}$ have r descriptor components for which the corresponding components of their hash codes belong to the same local hash neighborhoods and vice versa. Thus, the corresponding subset of the intermediate hash code key components ordered by $Cth^{(r)}$ is nothing else than the descriptor hash code.

Let us consider the essence of the algorithm on a particular example of the LFW dataset processing. The results of the calculation of ε_n for all values of n are presented for different values of Cth in the figures below. It follows quite an obvious fact from the figures: the higher the cosine similarity for a descriptor pair, the greater number of descriptor components belong to the same local hash neighborhoods with a higher probability. Now, it is easy to understand the essence of the contradiction between the acceleration and general accuracy for the search algorithm. The more descriptor key components are considered (the higher search speed because of the search subset reduction), the higher probability of absence of a descriptor with quite a high cosine similarity to the requested descriptor in the considered subset because the increase of R-value leads to the increase of the threshold-barrier level Cth for descriptors to get into the search subset [5].

In practice, due to a very large spatial variance even between descriptors for different face images of the same person ($R \to N \Leftrightarrow Cth \to 1$), there are always descriptor components for which the corresponding hash codes do not belong to the same hash neighborhoods ($\exists\ n$: $|P_{n,i} - P_{n,i'}| > 1$).

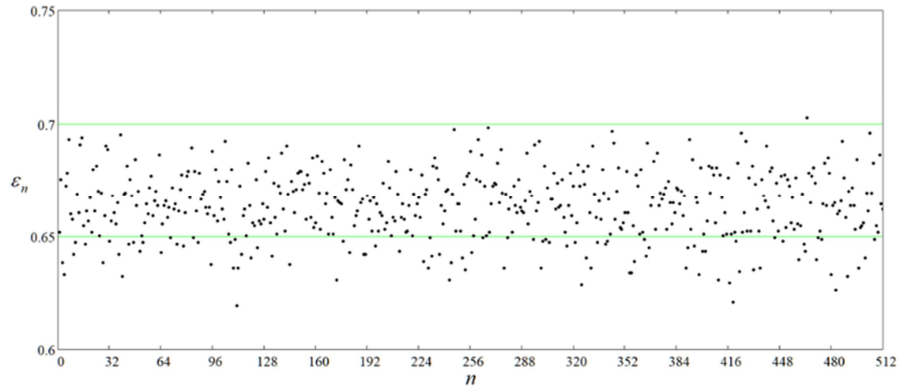

Figure 1. *ε_n at Cth=0.4.*

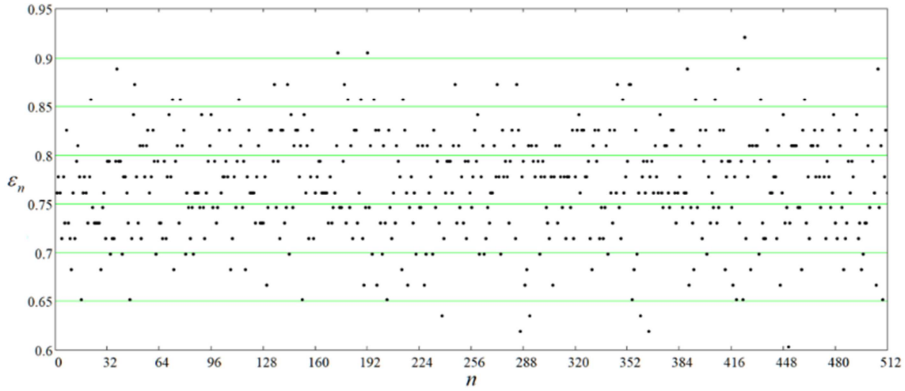

Figure 2. *ε_n at Cth=0.5.*

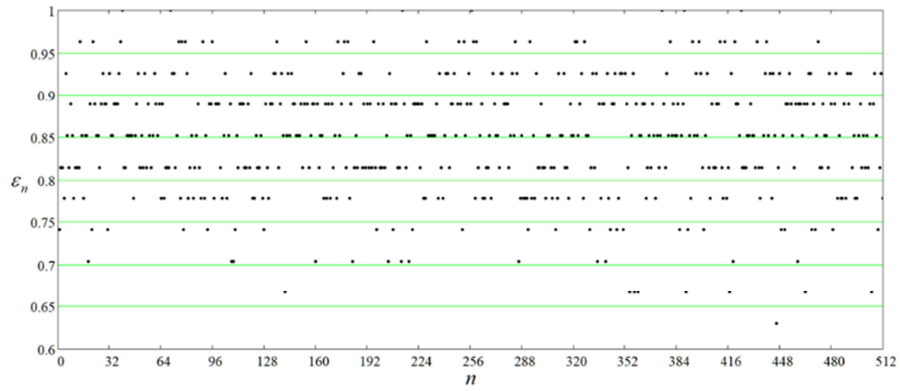

Figure 3. *ε_n at Cth=0.61.*

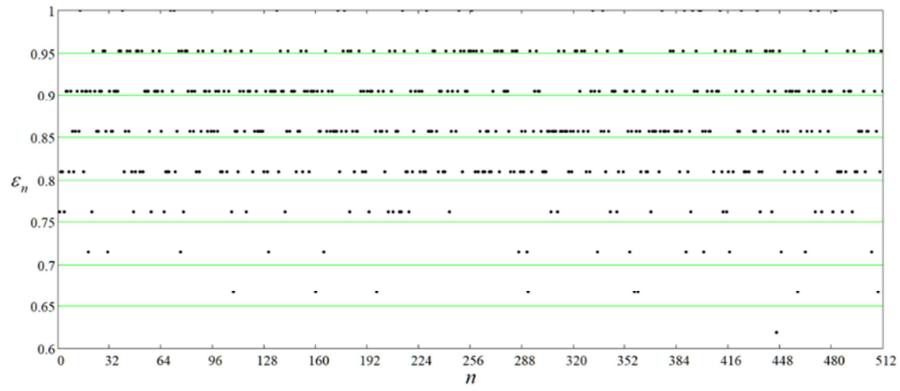

Figure 4. *ε_n at Cth=0.7.*

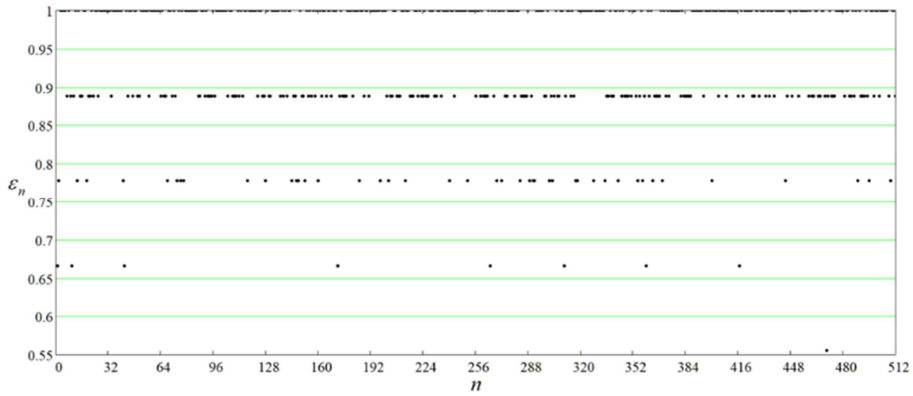

Figure 5. *ε_n at Cth=0.8.*

So, for the LFW dataset, the following subset ordered by descriptors components has been obtained:

Table 1. *Descriptor key components for the LFW dataset.*

r	n	Cth $^{(r)}$
1	423	0.57
2	388	0.58
3	213	0.59
4	39	0.60
5	256	0.60
6	69	0.61
7	374	0.61

2.6. Descriptor Hash Codes

In accordance with the considered example, the rules of calculation of parameters for each descriptor D by (4) and (5) along with the application of the obtained table-filter for the descriptor components define nothing else than the hash function to calculate the seven-digit quinary hash code H by the descriptor – hash key (in the general case, both the hash code bit width and its number system may be different). As an example of the descriptor considered earlier, there is its hash code in the table below.

Table 2. *Tabulated hash function for the LFW dataset.*

r	n	D_n	⇒ (4), (5) ⇒	H_r	→	H
1	423	−0.027835	→	0	⎫	
2	388	0.066036	→	4	⎪	
3	213	0.043252	→	4	⎪	
4	39	−0.010060	→	2	⎬	0442103
5	256	−0.034873	→	1	⎪	
6	69	−0.072940	→	0	⎪	
7	374	0.044692	→	3	⎭	

2.7. Database Cataloging

To accelerate access to the search objects in the database by their hash codes it is necessary to catalog the objects properly. In particular, for the considered example of the database (root directory), the subdirectories tree looks as shown in Figure 6. According to the figure, the root directory contains 5^7=78125 nested directories of the seventh deepest level. For the cataloged database of face images to be representative, it is necessary that every such subdirectory contains a sufficient number of files for different persons. Statistically, this condition can be more or less ensured when the database contains face images of at least 10^6 persons. However, the volume of the database should contain information about 10^7 or more different persons in order to exclude the high-reliability presence of empty directories at the deepest level. A value of a similar order can be reached from another point of view. The considered problem can be solved by modern PCs in the framework of the exhausting search within reasonable time intervals even for about 10^5 1-2 kilobytes descriptors. Additional opportunities arise when applying multiprocessor workstations with installed graphics accelerators. Hardware like that allows the database with the order of 10^6 descriptors to be processed during the same time intervals.

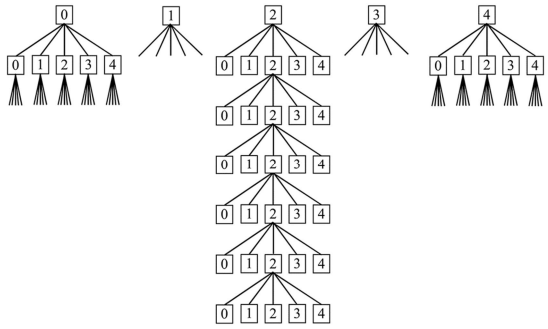

Figure 6. *Seven-tiered directory tree containing at the deepest seventh level both the images files and corresponding to them descriptors files comprising their hash codes as a part of their names.*

The transition to the databases with the order of 10^7 persons will require rather expensive solutions based on workstations united in a computing cluster. Whereas the proposed hashing algorithm can be applied to accelerate the search time by about $(25/13)^7 \approx 100$ times (Q=4) on average so that to process the face databases with volumes of more than 10^7 descriptors during reasonable time intervals even on PCs. At that, there is no need to re-index databases with such volumes after adding new elements to them because single elements have a negligible statistical weight within the framework of the proposed approach. Re-indexing of the databases is reasonable to carry out by the counter of new elements after their total number achieves a commensurate value with the number of the database primary elements.

2.8. Some Remarks

Now, let us consider the algorithm of the hash code-assisted search for the most similar face image from the database to the requested person face image on a particular example. So, let the LFW database with corresponding calculated descriptors be prepared in accordance with the algorithm described above. Suppose, that a search request for a face image has been generated by its descriptor. And let, for definiteness, it be the descriptor considered earlier. In this case, its hash code (0442103) can be obtained by means of the procedure (hash function) described above. Further, for all collisions of the descriptors by their hash code and the local neighborhoods of its components (see Table 3) the exhaustive search over such filtered descriptors (≈ 0.006 of their total number for the considered case) from the database is carried out to find a descriptor with the maximum cosine similarity to the requested one.

Table 3. *Initial neighborhoods for the hash code components.*

r	$H_r^{(1)}$
1	0..1
2	3..4
3	3..4
4	1..3
5	0..2
6	0..1
7	2..4

If the search result is a descriptor with the cosine similarity not less than $Cth^{(7)}$, then the search is considered successful and completed (in the general case, there may be a request to search for several most similar descriptors to the requested one). Otherwise, the search continues with the extension of the local hash neighborhoods on the left and right up to the exhaustive search at the current level r with the subsequent transition to the next level r (from large r-values to smaller ones) until either the found maximum cosine similarity value becomes equal to or greater than $Cth^{(r)}$, or all the descriptors in the database are processed. For example, let the maximum cosine similarity C_{max} be equal to 0.52 for the considered collisions. Then, the search should be performed over additional collisions for H_7 (see Table 4).

Table 4. *Additional neighborhoods for the hash code components.*

r	$H_r^{(2)}$
1	0..1
2	3..4
3	3..4
4	1..3
5	0..2
6	0..1
7	1

Let the same result of comparison for the cosine similarity be achieved (C_{max}=0.52). And so several times in a row, that is such a result has been obtained for the corresponding collisions listed below in Table 5 from left to right.

Table 5. *Further sequence of neighborhoods for the hash code components.*

r	$H_r^{(3)}$	$H_r^{(4)}$	$H_r^{(5)}$	$H_r^{(6)}$	$H_r^{(7)}$	$H_r^{(8)}$	$H_r^{(9)}$
1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
2	3..4	3..4	3..4	3..4	3..4	3..4	3..4
3	3..4	3..4	3..4	3..4	3..4	3..4	3..4
4	1..3	1..3	1..3	1..3	1..3	1..3	0, 4
5	0..2	0..2	0..2	0..2	3	4	0..4
6	0..1	2	3	4	0..4	0..4	0..4
7	0	0..4	0..4	0..4	0..4	0..4	0..4

As for the next collisions sequence (see Table 6) let the descriptor for which C_{max}=0.586 > $Cth^{(2)}$ be found. This obtained result is saved and the search is stopped.

Table 6. *Final sequence of neighborhoods for the hash code components.*

r	$H_r^{(10)}$	$H_r^{(11)}$	$H_r^{(12)}$
1	0..1	0..1	0..1
2	3..4	3..4	3..4
3	2	1	0
4	0..4	0..4	0..4
5	0..4	0..4	0..4
6	0..4	0..4	0..4
7	0..4	0..4	0..4

Now, a reasonable question arises: how often the cases, when $C_{max} < Cth^{(7)}$, take place during the initial search? Indeed, in this case, the search speed can reduce significantly up to the brute force. If such cases are common, the search by hash codes will not give considerable acceleration compared with brute force. However, the observations based on the practical application of neural network technologies like ArcFace [5] are as follows. In rather rare cases for a specific implementation of ArcFace, descriptors corresponding to high-quality face images of certain persons may have the cosine similarity less than 0.57 to the most representative descriptors for these persons. Whereas in some extremely rare cases, descriptors corresponding to the face images of some persons may have the cosine similarity of more than 0.61 to the most representative descriptors for some other persons. All these observations allow us to formulate the following hypothesis:

if the inequality $C_{max} < Cth^{(7)}$ is true during the initial search, then with a high probability there are no face images of the requested person in the database.

The overlapping effect for descriptor clusters of different persons [5] does not have such a statistical significance to lead to a crucial acceleration reduction for the hash code assisted search. In particular, for the LFW dataset, the average statistical acceleration of the search by more than 90 times has been achieved.

3. Conclusion

The modified geometric hashing algorithm has been proposed in the present study to conduct the search for a face image from the database that is most similar to the face image of the requested person. The proposed algorithm is as accurate as the brute-force search but at times faster. Its undeniable advantage is absolute accuracy in comparison with other approximate methods of quick search. In addition, it is easy to implement it and perform simple data processing,

simple hashing, and special database organization. Beginning from a certain required threshold database volume, the addition of new elements to it does not require immediate re-indexing of the database in contrast to other approaches of accelerated search with absolute accuracy. In particular, the new database elements do not influence the output hash code. That is, the database expansion does not break the robustness of the quick search algorithm. An additional advantage of the proposed algorithm is the absence of the necessity to apply expensive hardware solutions to process large volume datasets.

4. Recommendations

It is obvious that the search algorithm considered in the present study is very efficient and quite general, and it can be applied for wider purposes. However, the measure of its efficiency in the case when other type loss functions will be used is unclear [2]. Moreover, its efficiency is much more unclear when not intraclass (face-vs-face, dog-vs-dog, cat-vs-cat, bird-vs-bird etc.) but interclass comparison (person-vs-dog-vs-cat-vs-bird-vs-etc.) by descriptors will be made. To resolve these issues their additional research is required.

References

[1] Hou B.-W., Zheng R., Yang G.-Sh. (2014) Quick search algorithms based on ethnic facial image database. 2014 IEEE 5th International Conference on Software Engineering and Service Science, 573-576.

[2] Wang M., Deng W. (2018) Deep face recognition: a survey. https://arxiv.org/pdf/1804.06655.pdf, 1-24.

[3] Liu W., Wen Y., Yu Zh., Li M., Raj B., Song L. (2017) SphereFace: deep hypersphere embedding for face recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition, 6738-6746.

[4] Wang H., Wang Y., Zhou Zh., Ji X., Gong D., Zhou J., Li Zh., Liu W. (2018) CosFace: large margin cosine loss for deep face recognition. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 5265-5274.

[5] Deng J., Guo J., Xue N., Zafeiriou S. (2020) ArcFace: additive angular margin loss for deep face recognition. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4685-4694.

[6] Cao Q., Ying Y., Li P. (2013) Similarity metric learning for face recognition. 2013 IEEE International Conference on Computer Vision, 2408-2415.

[7] Vezzetti E., Marcolin F. (2015, Bentham Books) Similarity measures for face recognition.

[8] Nguyen H. V., Bai L. (2011) Cosine similarity metric learning for face verification. 2010 Asian Conference on Computer Vision, 709-720.

[9] https://en.wikipedia.org/wiki/Nearest_neighbor_search.

[10] Aumüller M. (2020) Algorithm engineering for high-dimensional similarity search problems. 18th International Symposium on Experimental Algorithms, 160, 1: 1-3.

[11] Johnson J., Douze M., J´egou H. (in press) Billionscale similarity search with GPUs. IEEE Transactions on Big Data. Also in (2017) https://arxiv.org/pdf/1702.08734.pdf, 1-12.

[12] FAISS (Facebook AI Similarity Search). https://github.com/facebookresearch/faiss/wiki.

[13] https://en.wikipedia.org/wiki/Voronoi_diagram.

[14] https://en.wikipedia.org/wiki/K-d_tree.

[15] https://en.wikipedia.org/wiki/Search_engine_indexing.

[16] https://en.wikipedia.org/wiki/Geometric_hashing.

[17] Benchmarking nearest neighbors. https://github.com/erikbern/ann-benchmarks.

[18] Li W., Zhang Y., Sun Y., Wang W., Li M., Zhang W., Lin X. (2019) Approximate nearest neighbor search on high dimensional data – experiments, analyses, and improvement. IEEE Transactions on Knowledge and Data Engineering, 32 (8), 1475-1488.

[19] Hyvonen V., Pitkanen T., Tasoulis S., Jaasaari E., Tuomainen R., Wang L., Corander J., Roos T. (2016) Fast k-NN search. https://arxiv.org/pdf/1509.06957.pdf, 1-10.

[20] Aumüller M., Bernhardsson E., Faithfull A. (2020) ANN-benchmarks: a benchmarking tool for approximate nearest neighbor algorithms. Information Systems, 87, 1-13.

[21] Abu-Aisheh Z., Raveaux R., Ramel J.-Y. (2020) Efficient k-nearest neighbors search in graph space. Pattern Recognition Letters, 134, 77-86.

[22] Hwang Y., Baek M., Kim S., Han B., Ahn H.-K. (2018) Product quantized translation for fast nearest neighbor search. The Thirty-Second AAAI Conference on Artificial Intelligence, 3295-3301.

[23] Abdelhadi A. M. S., Bouganis Ch.-S., Constantinides G. A. (2019) Accelerated approximate nearest neighbors search through hierarchical product quantization. 2019 International Conference on Field-Programmable Technology, 90-98.

[24] Subramanya S. J., Devvrit F., Simhadri H. V., Krishaswamy R., Simhadri H. V. (2019) DiskANN: fast accurate billion-point nearest neighbor search on a single node. 33rd Conference on Neural Information Processing Systems, 13748-13758.

[25] Guo R., Sun Ph., Lindgren E., Geng Q., Simcha D., Chern F., Kumar S. (2020) Accelerating large-scale inference with anisotropic vector quantization. 37th International Conference on Machine Learning, 119, 3887-3896.

[26] Ren J., Zhang M., Li D. (2020) HM-ANN: efficient billion-point nearest neighbor search on heterogeneous memory. 34th Conference on Neural Information Processing Systems, 1-13.

[27] Cariou C., Moan S. L., Chehdi K. (2020) Improving k-nearest neighbor approaches for density-based pixel clustering in hyperspectral remote sensing images. Remote Sensing, 12 (22), 3745-3771.

[28] Li M., Zhang Y., Sun Y., Wang W., Tsang I. W., Lin X. (2020) I/O efficient approximate nearest neighbor search based on learned functions. 2020 IEEE 36th International Conference on Data Engineering, 289-300.

[29] Dong Y., Indyk P., Razenshteyn I., Wagner T. (2020) Learning space partitions for nearest neighbor search. Eighth International Conference on Learning Representations, 1-16.

[30] Aghbari Z. A., Ismail T., Kamel I. (2020) SparkNN: a distributed in-memory data partitioning for KNN queries on big spatial data. Data Science Journal, 19 (1), 35-48.

[31] Chen H., Chillotti I., Dong Y., Poburinnaya O., Razenshteyn I., Riazi M. S. (2020) SANNS: scaling up secure approximate k-nearest neighbors search. Proceedings of the 29th USENIX Security Symposium, 2111-2128.

[32] Pan Y., Pan Z., Wang Y., Wang W. (2020) A new fast search algorithm for exact k-nearest neighbors based on optimal triangle-inequality-based check strategy. Knowledge-Based Systems, 189, 105088.

[33] WIDERFACE: a face detection benchmark. http://shuoyang1213.me/WIDERFACE.

[34] Labeled faces in the wild. http://vis-www.cs.umass.edu/lfw.

[35] Pozdnyakov D. (2020) Determination of the most representative descriptor among a set of feature vectors for the same object. https://arxiv.org/ftp/arxiv/papers/2007/2007.03021.pdf, 1-8.

[36] https://en.wikipedia.org/wiki/Bisection_method.