Large scale reverse image search

A method comparison for almost identical image retrieval

Mathieu Gaillard[1], Előd Egyed-Zsigmond[1,2]

1 Université de Lyon
INSA Lyon,
mathieu.gaillard@insa-lyon.fr
2.Université de Lyon
LIRIS, INSA Lyon
elod.egyed-zsigmond@insa-lyon.fr

RESUME. Dans ce papier nous présentons une étude comparative de méthodes pour un système de recherche d'images inversée. Nous nous concentrons plus spécifiquement sur le cas de la recherche d'images quasi identiques dans de très grands ensembles d'images. Après une étude de l'état de l'art, nous avons implanté notre propre système de recherche d'images inversée en utilisant trois descripteurs basés sur des techniques de hachage perceptuel choisies pour leur extensibilité. Nous avons comparé la vitesse et la précision/rappel de ces méthodes contre plusieurs modifications (flou gaussien, redimensionnement, compression, rotation, recadrage). Nous proposons également un système à deux couches combinant : une première étape très rapide mais moyennement précise ; avec une étape, certes, plus lente mais beaucoup plus précise. Nous améliorons ainsi la précision globale de notre système tout en conservant sa rapidité de réponse.

ABSTRACT. In this paper, we presented our study and benchmark on Reverse Image Search (RIS) methods, with a special focus on finding almost similar images in a very large image collection. In our framework we concentrate our study on radius (threshold) based image search methods. We focused our study on perceptual hash based solutions for their scalability, but other solutions seem to give also good results.

We studied the speed and the accuracy (precision/recall) of several existing image features. We also proposed a two-layer method that combines a fast but not very precise method with a slower but more accurate method to provide a scalable and precise RIS system.

MOTS-CLES : recherche d'images inversé, pHash, optimisation, SI images

KEYWORDS: reverse image search, pHash, optimization, image information systems

1. Introduction

In this paper we deal with the reverse image retrieval problem in image centered information systems. The purpose of the reverse image retrieval is to retrieve images by similarity based on a query image. This study is mainly motivated by the need to find the original of an image in a large-scale image database given a slightly modified version of it. This kind of systems are extensively used in the context of Intellectual property and crime prevention. Many implementations already exist, for example Google Images (Google, 2017), TinEye (TinEye, 2017) and Microsoft PhotoDNA(Microsoft, 2017). At last we also expose a benchmark that we designed in order to evaluate and compare these systems.

In the introduction we present and define the problem. In the second section we will explain how reverse image search engines generally work. In the third section we will specially present the perceptual hashes as a technical solution to address our problem. We will also introduce different existing implementations. Finally, in the fourth section, we will expose our benchmark for these systems and the result of our experimentations.

In this document we study a reverse image search engine that is capable of indexing a huge amount of images and then allows the user to search for the original version of an image, even if this one is slightly modified. The number of indexed images can be more than a million. The time to index does not really matter as long as it is done within a reasonable interval. On the contrary the time to search should be as short as possible in such a way that it is possible to search 10 000 images in less than an hour. If several technical solutions are able to address this problem, we want to be able to compare them in order to select the one that best fits our needs.

We define a modification as an operation that does not alter the essential content of an image (Zauner, 2010) A non-exhaustive list of modifications is given later in this section.

The definition of a similar image varies depending on what photometric and geometric variations are deemed acceptable. This depends on the application (Philbin, Isard, & Zisserman, 2007). For our application, two images are considered perceptually similar if a human interprets and understands them as the same image. For example, an image and a slightly modified version of it are perceptually similar whereas two images with similar colors and shapes can be conceptually different therefore perceptually different.

In this section a non-exhaustive list of image modifications is exposed: Scaling: not necessarily with the same aspect ratio ; Lossy compression: JPEG ; Filter: blur, noise, artistic color filters ; Rotation ; Flipping: horizontally or vertically ; Cropping ; Shifting ; Random deletion: rectangle ; Random insertion: text, watermark, logo ;

After a modification by one of those listed above, an image can be considered as similar to the original one. These modifications do not alter the essential content of images if they are done carefully. Obviously, it is no more the case if the parameters take extreme values. For example, if we crop the half of an image, this one becomes

perceptually different. On a computer to perform these modifications on a collection of images we can use ImageMagick. (ImageMagick, 2017)

2. State of the art

Reverse image search (RIS) is a type of content-based image retrieval (CBIR). According to (Chutel & Sakhare, 2014), It is a search engine technology that takes images as input and returns results related to the query image. The search analyses the actual content of images rather than the metadata such as keywords or descriptions associated with them. The aim of Reverse Image Search Engine is to find the similar, exact image on web based on the given query image though the search images are cropped, transformed or it may have illumination changed. This Reverse Image Search engine can be used for detecting unauthorized use of brands and copyright images. Other common usage modes are to locate the source of an image, find a higher resolution version, discover other webpages where the image appears or get some more information about the image. (TinEye, 2017)

The following is a list of terms related to the reverse image search topic: near identical image detection, image retrieval, content-based image retrieval, information retrieval.

In what follows, we present our general framework for reverse image search.

Reverse image search with large databases imposes two challenging constraints on the methods used. Firstly, for each image, only a small amount of data (a fingerprint) can be stored; secondly, queries must be very cheap to evaluate. In order to be able to deal efficiently with millions of images, while still being able to keep a sizeable portion of the data in main memory, we need to generate an extremely compressed feature vector for each image. (Philbin et al., 2007)

Most approaches to reverse image search share a similar pattern. Firstly, an image representation and a distance measure are defined, which affects both the amount of data stored per image and the time complexity of a database search. When searching the database for similar images, algorithms of different time complexity are used, the most naive approach being computing the difference to every image in the database. (Philbin et al., 2007)

Reverse image search engines usually work in two phases: indexing and searching. In the indexing phase, the database is filled with feature vectors of images that should be found later on. The images are not necessarily stored in the database; this reduces the size of the database dramatically. In the searching phase, a new image is presented to the system and a feature vector representing this one is computed. Then the feature vector is compared to those in the database using the previously defined distance measure. Here are two manners to handle the results: a radius (threshold) based and a k nearest neighbors based method. The radius based method works well for content authentication emphasizing precision while the k nearest neighbors, usually is used when the needs of recall are more important. A general workflow for the radius based method is illustrated in Figure 1. There is a match if the distance between the query

vector and a vector representing an indexed image is less than or equal to a threshold. Usually the distances are normalized between 0 and 1. If there is a match the system will return the images corresponding to the concerned feature vectors as results to the similar image search for the query image. There are a lot of techniques to define a pair of image representations and distances. Some of them are discussed in the next section. The workflow is composed of these steps:

Feature vector extraction: A feature vector is computed from the image using one of the later-discussed techniques.

Matching: A feature vector is compared to those in the database. A sequential search is the easiest way to iterate over the database and can be parallelized or even distributed among many computers. There also exist some special data structures to fasten the search when the distance verifies certain properties. Instead of considering all indexed images, only a subset of them is compared to the query feature vector. For example, in Hamming space, the Multi-Index Hashing (MIH) has sub-linear run-time behavior for uniformly distributed codes (Norouzi, Punjani, & Fleet, 2013).

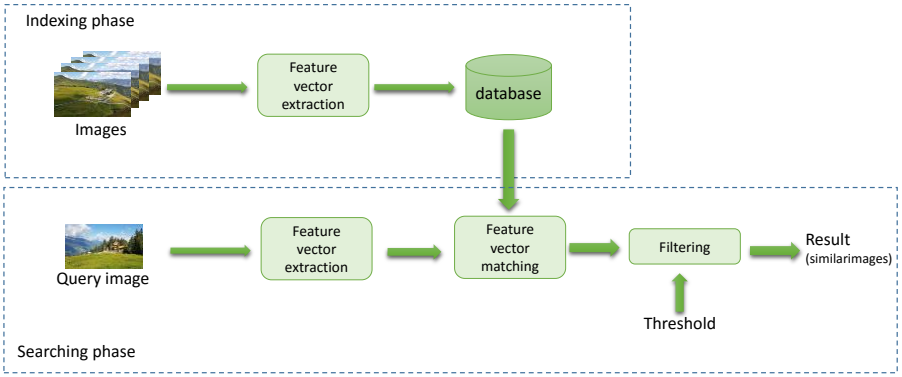

Figure 1 General workflow for reverse image search.

Therefore the determination of an adequate threshold, in accordance with the actual application scenario, is critical. (Zauner, 2010) Information retrieval research has shown that precision and recall follow an inverse relationship. (Datta, Joshi, Li, & Wang, 2008) If the threshold is too low, the precision is better at the expense of the recall because only the most relevant images are retrieved. On the contrary, if the threshold is too high, the recall is better but the precision is worse. When the application needs to authenticate an image, the precision is more important because we want to limit the number of false positives, whereas, when the application needs to identify content, the recall is more important because the user can deal with a small number of false positives. (Zauner, 2010) In any case, relevant and irrelevant images cannot be separated clearly; the boundary between these two sets is fuzzy.

3. Technical solutions

As stated in the last section, a technical solution for reverse image search consists of an image representation and a distance measure between these representations. (Philbin et al., 2007) In addition to that, the image representation should be easy to store in a database. For example, to compute a similarity distance between two images first their feature vectors are extracted and then the distance between them is measured. The more the result is near to 0, the more the images are considered as similar. In our study we have considered several image features based on colors, textures, contours, interest points, neural networks and tested libraries such as LIRE[1](Lux & Chatzichristofis, 2008), OpenCV[2] and pHash[3]. We focused our study on perceptual hashing because, at a first glance, it appeared to be significantly faster than the descriptors from the LIRE library with an almost identical accuracy, on basic modifications (scaling, compression, grayscale filter).

As our goal was to search large image collections, we focused mainly on the perceptual hash functions implemented in the library pHash because it was really faster than the others giving comparable accuracy. Moreover, by using an existing library we save implementing time in order to focus our resources on experimentations.

3.1. Perceptual hashing

3.1.1 Definition

A perceptual hash function is a type of hash function that has the property to be analogous if inputs are similar. This allows us to make meaningful comparisons between hashes in order to measure the similarity between the source data. Perceptual hash functions are an interdisciplinary field of research. Cryptography, digital watermarking and digital signal processing are part of this field of research.

The definition of a hash function according to (Menezes, Oorschot, & Vanstone, 1997) is:

> *A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values.*

In the case of a perceptual hash, some more properties should be present according to (Zauner, 2010):

> *Let H denote a hash function which takes one media object (e.g. an image) as input and produces a binary string of length l. Let x denote a particular media object and $\hat{x}$ denote a modified version of this media object which is "perceptually similar" to x. Let y denote a media object that is "perceptually*

[1] http://www.lire-project.net
[2] http://opencv.org
[3] http://phash.org

different" from x. Let x' and y' denote hash values. $\{0/1\}^l$ *represents binary strings of length l. Then the four desirable properties of a perceptual hash are identified as follows.*

A uniform distribution of hash-values; the hash-value should be unpredictable.

$$P(H(x) = x') \approx \frac{1}{2^l}, \forall x' \in \{0/1\}^l$$

Pairwise independence for perceptually different media objects.

$$P(H(x) = x' | H(Y) = y') \approx P(H(x) = x'), \forall x', y' \in \{0/1\}^l$$

Invariance for perceptually similar media objects.

$$P(H(x) = H(\hat{x})) \approx 1$$

Distinction of perceptually different media objects. It should be impossible to construct a perceptually different media object that has the same hash-value as another media object.

$$P(H(x) = H(y)) \approx 0$$

Most of the time to achieve these properties the perceptual hash function extract some features of media objects that are invariant under slight modifications to construct a perceptual hash. For example, knowing how a compression algorithm works, it is possible to find some invariant features and then design a perceptual hash based on them. Some examples of perceptual hash functions for images are detailed later in this section.

3.2. Implementations

In this section, we present the three implementations of perceptual hash functions that we used in our benchmark: the DCT based , the Marr-Hildert Operator based and the Radial Variance based perceptual hash functions from (Zauner, 2010).

The Discrete Cosine Transformation (DCT) based perceptual hash from (Zauner, 2010) takes advantage of the property that low-frequency DCT coefficients are mostly stable under image modifications to construct a 64 bits image hash. The Hamming distance is used to compare them. The fact that the hashes are encoded on 64 bits and the use of the Hamming distance is a wise choice because a hash can fit in a processor register. Moreover, to fasten the calculation of the Hamming distance, it is possible to use the special popcount (Sun & Mundo, 2016) instruction from the x86 processor family.

The Marr-Hildreth (MH) operator, also denoted as the Laplacian of Gaussian (LoG), is a special case of a discrete Laplace filter. It is an edge and contour detection based image feature extractor. The MH operator generates vectors encoded on 576 bits.

The Radial Variance hash (Standaert et al., 2005) is based on the Radon transform that is the integral transform which consists of the integral of a function over a straight line. It is robust against various image processing steps (e.g. compression) and more robust than the DCT and MH based perceptual hash functions against geometrical transformations (e.g. rotation up to 2°).

4. Benchmarking

As stated in (Zauner, 2010) not much research has been published dealing with the benchmarking of perceptual hash functions. Therefore, they propose their own benchmark for perceptual hash function: Rihamark. This one allows the user to compare several perceptual hash functions against several attacks and to analyze the results with graphics. The benchmark is modular so that it is possible to add new perceptual hash functions, attacks functions or analyzer functions. This benchmark is for example very useful to choose which perceptual hash function is best suited for a specific usage.

Currently we didn't find an already implemented benchmark for a reverse image search engine. It is important to have a benchmark to test all the technical solutions previously detailed. As previously said there exists a benchmark but only for perceptual hash functions. However, it could be adapted to reverse image search. In fact, (Zauner, 2010) proposes some metrics to evaluate content identification systems but no implementation is provided along with it. Their approach comes from a biometrics background because they model the search as m authentication tests. Basically they calculate the False Accept and the False Reject Rate (FAR/FRR) and then plot the Receiver Operating Characteristic (ROC) curve. They also explain how to compare several perceptual hash functions based on their respective ROC curves.

We designed a new framework to benchmark the reverse image search engines. Our approach is based on the evaluation measures of information retrieval systems described in (Manning, Raghavan, & Schütze, 2009). We model the reverse image search engine as an information retrieval system that returns an unranked set of documents for a query. If many documents are retrieved, the user is in charge of choosing the best suited image.

4.1. Metrics

4.1.1 Effectiveness

To process a query, the reverse image search engine classifies the indexed images by relevance. In addition, we introduce a threshold for each similarity measure to be able to precisely select the images that are to be considered as similar to the query image. Thus, each image, whether relevant for the query or not, can be retrieved or not. This notion can be made clear by examining the following contingency table from (Manning et al., 2009).

	Relevant	Nonrelevant
Retrieved	True positive	False positive
Not retrieved	False negative	True positive

The effectiveness of the system is measured with the following metrics from (Manning et al., 2009)

Precision (P) is the fraction of retrieved documents that are relevant.

$$Precision = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)} = P(relevant|retrieved)$$

Recall (R) is the fraction of relevant documents that are retrieved.

$$Recall = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)} = P(retrieved|relevant)$$

A single measure that trades off precision versus recall is the F measure, which is the weighted harmonic mean of precision and recall:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \ ; \ F_{\beta=1} = \frac{2PR}{P + R}$$

It is possible to change the weights in the harmonic mean of the F measure in order to tune it. This is done by changing the β parameter. Values of $\beta < 1$ emphasize precision, while values of $\beta > 1$ emphasize recall. This is important in order to benchmark the system in accordance with its application.

4.1.2. Performance

It is important to measure the time taken by the system for indexing and searching. Actually, the system should be able to index millions of images and search across them as fast as possible. The complexity of both the indexing and searching phases depends on the number of images and is not necessarily linear. In fact, the data structure used to store the feature vectors can have a nonlinear complexity. Therefore, we propose to measure the indexing and searching time for a certain number of images.

4.2. Protocol

In order to compare the effectiveness and the speed of different image search methods, we created a comparison protocol. We chose 25 000 images from the mirflickr dataset[4].

In order to be able to calculate automatically the precision and recall of the results, we applied 6 small modifications on each image, that gave us a dataset with 175 000

[4] Http://press.liacs.nl/mirflickr/ (consulted in 2017)

images. We measured the index and search speed as well as the results precision and recall.

The modifications (illustrated on Figure 2) applied on the images are:

1. Gaussian blur (r=4, Σ=2)
2. Black and white transformation
3. Resize to half height and width
4. Compression into jpeg with a quality=10
5. Clockwise rotation by 5°
6. Crop by 10% at the right side of the image.

Figure 2 Illustrations of the image modifications

These modifications represent the basic cases of small changes images usually undergo over the web.

The benchmark was centered on high speed image search methods. We used only one perceptual hash function to retrieve the results. Our first benchmark protocol for a generic reverse image search engine is detailed in this section.

1. Select N+M images that are representative to an application with no duplicated images. In our case N=24 000 and M=1000 (the first 1000 images from the dataset in alphabetical order)
2. Split them into 2 sets of N base images and M non-indexed images.
3. Select K transformations and from the N base images, generate K new image sets containing K*N transformed images. In our case K=6, and the transformations are those enumerated above.

4. Index the N base images and the N*K transformed images according to the different image descriptor extraction methods. For us: Discrete Cosine Transform (DCT), Marr-Hildert Operator (MH) and Radial Variance (RV) based perceptual hashes (Zauner, 2010).
5. Make search queries with:
    a. The M images from the non-indexed image set.
    b. The N images from the base image set.
    c. The K*N images from the transformed sets.
6. Analyze the search results and compute the mean precision, the mean recall and the mean F measure of all queries.
    a. For the M images of the non-indexed set, there should be no relevant result image. Thus the result should be empty.
    b. When querying with one of the other (K+1)*N already indexed images, the relevant results are the K+1 images that are the transformations of the query image. Thus the result of each query should contain K+1 images that come from the same base image as the query image.

It is possible to repeat this protocol for several different thresholds in order to choose the best one suited for an application. In order to be able to measure the precision and the recall of our image retrieval information system, we decided to apply thresholds to the similarity scores between the query and the result images. We obtained this way a precise result set with a given item count. Having a threshold to distinguish the similar and different images enables our method to be considered also as an information retrieval system that returns an unranked set of images for a query.

4.3. Results

4.3.1. Improving searching time in Hamming space

In the case of our study we implemented and benchmarked 3 solutions to search for 64 bit hashes in Hamming space: a CPU based, a GPU based and a MIH (Norouzi, Punjani, & Fleet, 2013) based solution. For a large number of hashes (at least 50M) the MIH solution is the most efficient, followed by the GPU and finally the CPU. The two latter methods are memory bound thus the memory bandwidth and cache are both performance factors.

4.3.2. Effectiveness against modifications

In order to evaluate the effectiveness of the DCT, MH and RV based perceptual hash from pHash (Zauner, 2010) against modifications, we indexed the N base images and then made K search queries each with all N modified images.

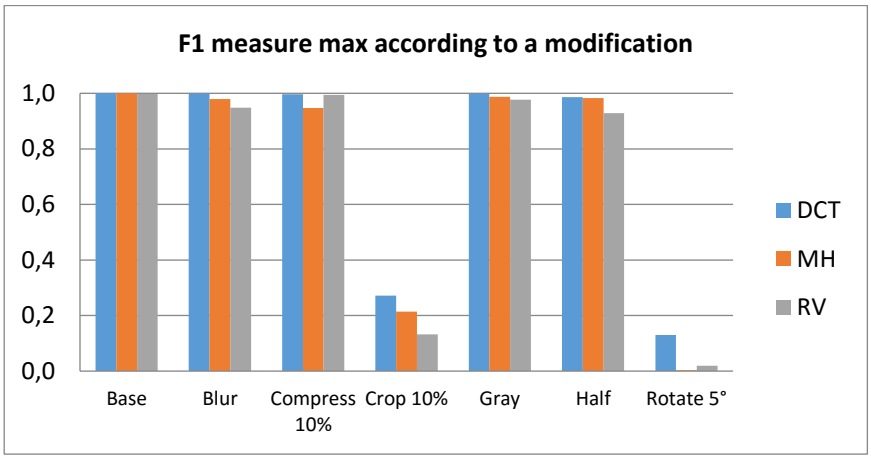

Figure 3 Maximum F1-measure of DCT, MH and RV perceptual hash function against modifications

We computed the F1-measure for various thresholds and took the maximum. The functions are robust against Gaussian blur (r=4, Σ=2), JPEG compression (quality 10%), grayscale filter, and scale to half the size. However the functions are not robust against crop (10% on the right) and rotate (5° clockwise) modifications as illustrated on Figure 3.

4.3.3. One-layer system

In a first implementation, we tested the speed and accuracy of a Reverse Image Search system based on one perceptual hash function. The experiment was carried out in order to get the best threshold values for the different methods (DCT (Figure 4), MH (Figure 5) and RV (Figure 6)).

We implemented the protocol as Linux shell commands and C++, using processor based and GPU based optimizations based on (Sun & Mundo, 2016) and other online available libraries[5].

The first results (see Table 1) showed that the DCT based hash was clearly faster than the Marr-Hildert Operator and Radial Variance based hash. It was also more accurate against the 6 chosen modifications. We tested the descriptor accuracy, calculating the mean precision, recall and F-measure of the returned results. We carried out the calculations for different threshold values. The threshold here is a normalized descriptor distance value, below which two images are considered as similar.

[5] See our implementation on : https://github.com/mgaillard/pHashRis

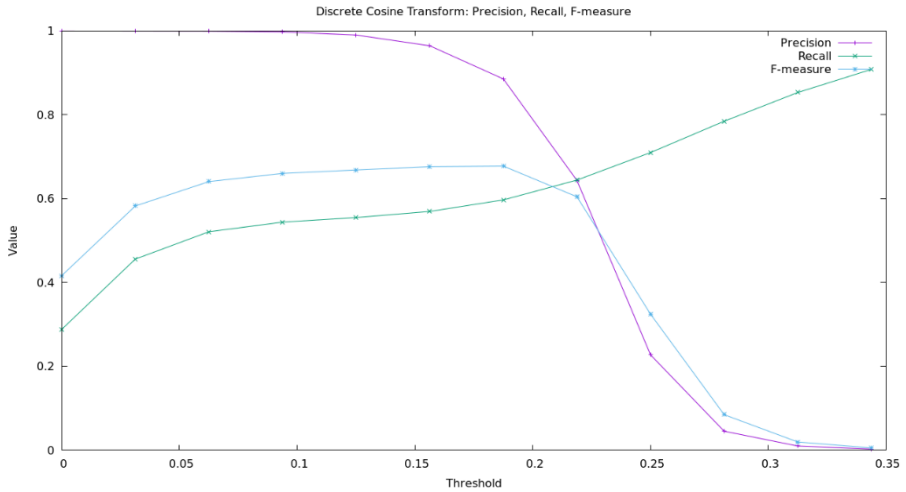

Figure 4 Precision/Recall/F-measure curves of the DCT based perceptual hash function search results according to different threshold values

The performance is evaluated through the index and search speed. Table 1 presents the measurements done for 125 000 images on an OVH dedicated virtual machine equipped with an Intel Xeon Haswell 8 cores at 3.1 GHz and 30 GB of RAM.

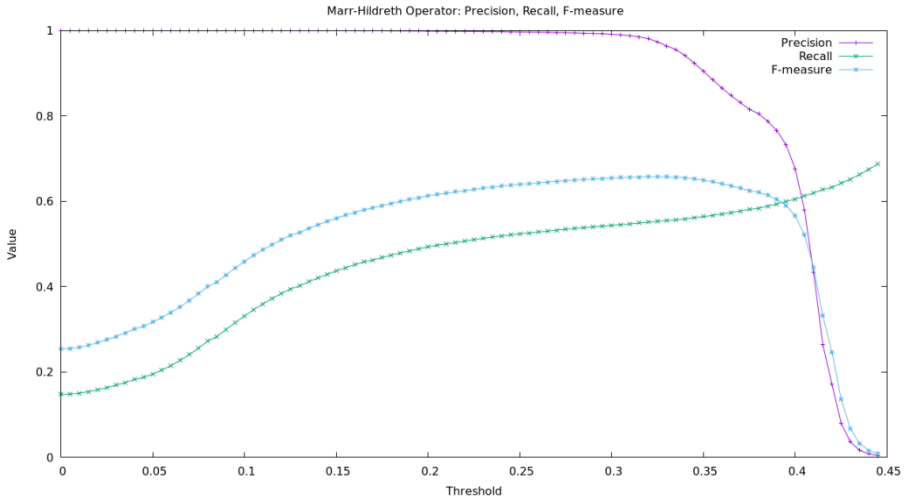

Figure 5 Precision/Recall/F-measure curves of the Marr-Hilderth based perceptual hash function search results according to different threshold values.

We also tested, the evolution of our accuracy measures with different degrees of the modifications. We noticed that the different hash methods were quite sensible to rotations (above 2°) or to cropping (above 5%), but resisted very well to compression, blur or resize.

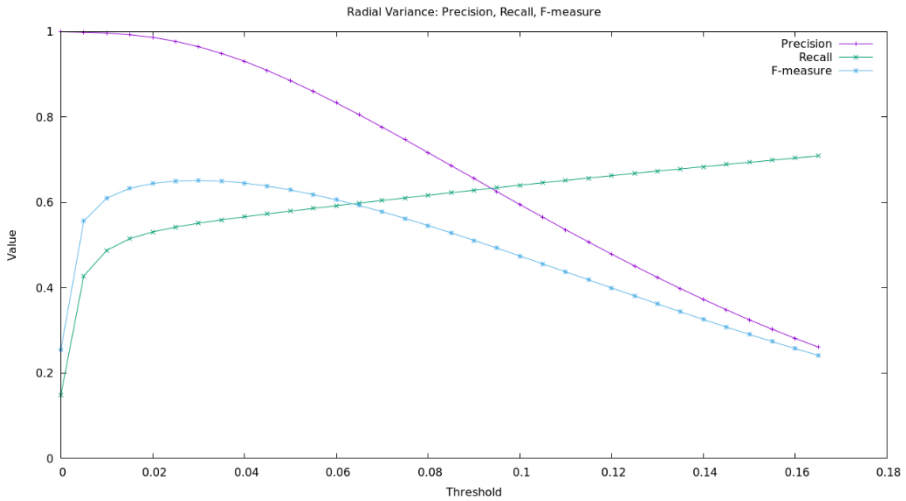

Figure 6 Precision/Recall/F-measure curves of the Radial Variance based perceptual hash function search results according to different threshold values.

	Index 125k images	125k queries on 125k images
DCT	3 min 55 sec	3 min 59 sec
MH	31 min 13 sec	32 min 20 sec
RV	1 min 48 sec	3h 47 min 31 sec

Table 1, image indexing and search time measures for the DCT, MH and RV based perceptual hash methods.

4.3.4. Two layered system

A second experiment was carried out in order to enhance the precision of our reverse image search engine. We used two successive layers of reverse image search. First, a very fast layer whose recall is high and precision is not perfect based on one perceptual hash function. Second a more accurate layer whose precision and recall are both near to 1. The aim of the first layer is to drastically reduce the number of images to be processed by the second layer which is more accurate but more expensive in searching time. The second layer is based on SIFT descriptors and on a distance between them. The distance is defined as follows:

When comparing two images A and B, let I be the number of interest points in image who has less interest points and J be the number of interest points in other image. We compute a matching that gives us for each interest point of image A the 2 nearest neighbors in the image B. Among these matches, we select only the G good matches that pass the ratio test proposed by (Lowe, 2004) in section 7.1 (In our

implementation 0,8). Finally, the distance is D = 1 – G/I. The two images are considered similar if the distance is less than or equal to 0,9.

In order to compare the results with or without the second, SIFT based layer, we run the protocol first without it in order to estimate a reasonable threshold for the perceptual hash layer and then with the SIFT layer in order to be able to compare the results. We choose the threshold for the perceptual hash layer so that the average precision of the first layer is around 0.4 thus the SIFT layer has to deal with 20 images on average which is reasonable. Because the search time is higher, we run it with these parameters: N=2000, M=200.

The results of the comparisons between the precision and recall evolutions in a one layer (DCT based perceptual hash) and a two layers (DCT based perceptual hash and SIFT based descriptor) information retrieval system are presented in Figure 7.

The precision is enhanced without affecting too much the recall at the expense of searching time and index size. We can see that for a threshold of around 0.25, the precision of the first step DCT perceptual hash based method is around 0.7. That means that in our case, when 7 images are to be returned, around 15 images are retrieved. The SIFT based comparison, even in its brute force implementation runs very fast on such a small number of images and enables to improve the global precision considerably (to more than 0,95). The implementation of our benchmarks can be accessed on GitHub[6].

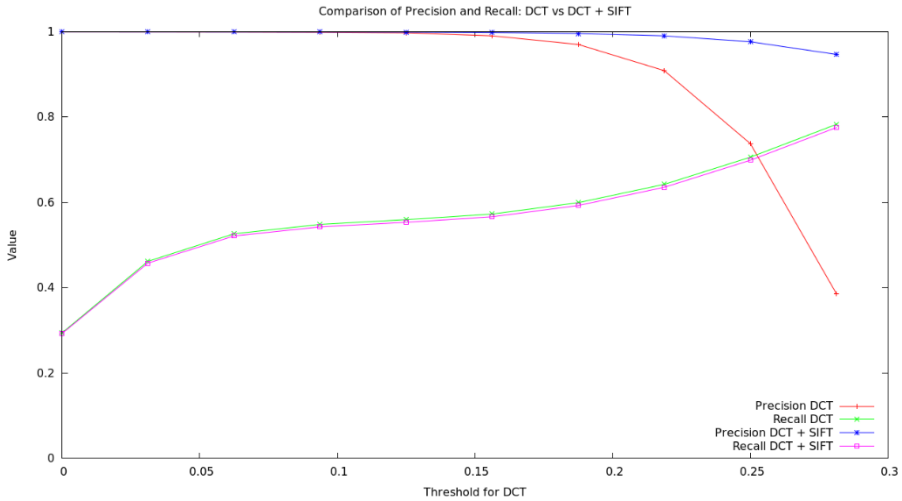

Figure 7 Precision and recall evaluation comparison between a one layer (DCT) RIS system and a two layers (DCT + SIFT) RIS system

[6] https://github.com/mgaillard/pHashRis

5. Conclusion

In this paper, we presented our study and benchmark on Reverse Image Search (RIS) methods, with a special focus on finding almost similar images in a very large image collection.

In our framework for reverse image search we state that there are mainly two algorithms for the search phase. One searches for the k nearest neighbors and the other searches for all images in a radius (within a distance threshold). We studied more the latter so the metrics of the benchmark should be adapted in the case of a k nearest neighbor method because the results can no longer be modeled as an unranked set of documents.

We focused our study on perceptual hash based solutions for their scalability, but other solutions seem to give also good results.

We studied the speed and the accuracy (precision/recall) of several existing image features. We also proposed a two-layer method that combines a fast but not very precise method with a slower but more accurate method to provide a scalable and precise RIS system.

The two-layer method can be extended with multiple functions in each layer. It is especially possible to use several perceptual hash functions in the first layer each of them tailored to a special modification. For example, a DCT based perceptual hash is robust against JPEG compression, Gaussian blur, scaling but is weak against rotation. To compensate for this weakness we could use a Color Moment based perceptual hash (Tang, Dai, & Zhang, 2012) which is robust against rotations. The second, SIFT based, layer has also some weaknesses. For example, images on which colors are uniform have a small number of interest points. To address this problem, it could be relevant to use other more accurate methods.

We foresee also the implementation of the LSH method from (Philbin et al., 2007) to reduce even more the hash sizes. The test of neural network based methods, such as the DSRH method from (Yao, Long, Mei, & Rui, 2016) is also a possible improvement idea although it needs a quite heavy learning phase.

We implemented our method in a near duplicate image search application[7] that can detect near duplicate image groups very quickly. This application can integrate information systems containing many images. One of our application fields is the illegal copy detection in large image sets. In this situation, the application has two inputs: the original images and the image set to check to search their copies in. The output will provide for each original image the list of its near duplicates found in the images set to check.

[7] https://github.com/mgaillard/ImageDuplicateFinder

References

Chutel, P. M., & Sakhare, A. (2014). Evaluation of compact composite descriptor based reverse image search. In *2014 International Conference on Communication and Signal Processing* (pp. 1430–1434). IEEE. http://doi.org/10.1109/ICCSP.2014.6950085

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2008). Image retrieval. *ACM Computing Surveys*, *40*(2), 1–60. http://doi.org/10.1145/1348246.1348248

Google. (2017). Google Images. Retrieved February 18, 2017, from https://images.google.com/

ImageMagick. (2017). ImageMagick @ Convert, Edit, Or Compose Bitmap Images. Retrieved February 18, 2017, from https://www.imagemagick.org/

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. http://doi.org/10.1023/B:VISI.0000029664.99615.94

Lux, M., & Chatzichristofis, S. A. (2008). Lire: lucene image retrieval. In *Proceeding of the 16th ACM international conference on Multimedia - MM '08* (p. 1085). New York, New York, USA: ACM Press. http://doi.org/10.1145/1459359.1459577

Manning, C. D., Raghavan, P., & Schütze, H. (2009). An Introduction to Information Retrieval. *Information Retrieval*, (c). http://doi.org/10.1109/LPT.2009.2020494

Menezes, A. J., Oorschot, P. C. Van, & Vanstone, S. a. (1997). Handbook of Applied Cryptography. *Electrical Engineering*, *106*, 780. http://doi.org/10.1.1.99.2838

Microsoft. (2017). PhotoDNA Cloud Service. Retrieved February 18, 2017, from https://www.microsoft.com/en-us/photodna

Norouzi, M., Punjani, A., & Fleet, D. J. (2013). Fast Exact Search in Hamming Space with Multi-Index Hashing. Retrieved from http://arxiv.org/abs/1307.2982

Philbin, J., Isard, M., & Zisserman, A. (2007). Scalable Near Identical Image and Shot Detection. *Analysis*, 549–556. http://doi.org/10.1145/1282280.1282359

Standaert, F.-X., Lefebvre, E., Rouvroy, G., Macq, B., Quisquater, J.-J., & Legat, J.-D. (2005). Practical evaluation of a radial soft hash algorithm. In *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II* (p. 89–94 Vol. 2). IEEE. http://doi.org/10.1109/ITCC.2005.229

Sun, C., & Mundo, C. C. del. (2016). Revisiting POPCOUNT Operations in CPUs/GPUs. In *The International Conference for High Performance Computing, Networking, Storage and Analysis* (p. 2p (poster)). Salt Lake City, Utah, USA.

Tang, Z., Dai, Y., & Zhang, X. (2012). Perceptual Hashing for Color Images Using Invariant Moments. *Appl. Math. Inf. Sci*, *6*, 643–650.

TinEye. (2017). TinEye Reverse Image Search. Retrieved February 18, 2017, from https://tineye.com/

Yao, T., Long, F., Mei, T., & Rui, Y. (2016). Deep Semantic-Preserving and Ranking-Based Hashing for Image Retrieval. *International Joint Conference on Artificial Intelligence (IJCAI)*, (c), 3931–3937.

Zauner, C. (2010). *Implementation and benchmarking of perceptual image hash functions*. *Master's thesis*. Upper Austria University of Applied Sciences, Hagenberg Campus. Retrieved from http://phash.org/docs/pubs/thesis_zauner.pdf