Hue class equalization to improve a hierarchical image retrieval system

Tristan d'Anzi, William Puech, Christophe Fiorio, Jérémie François

Hue class equalization to improve a hierarchical image retrieval system

Tristan D'Anzi[1] [2], William Puech[1], Christophe Fiorio [1] and Jérémie François[2]
[1] LIRMM, UMR 5506 CNRS, University of Montpellier, Montpellier, FRANCE
[2] PIC SIDE Company, Cap Oméga, Montpellier, FRANCE

***Abstract*— This paper proposes a filtering system within a large database in order to accelerate image retrieval. A first filter is applied to the database in order to have a small number of candidates. This filter consists of a global descriptor based on color classification. Instead of the use of static classification based on the HVS (Human Visual System), the classification is based on a uniform repartition of pixels from the database. Those classes are gathered from a learning database. With this classification a global descriptor is computed based on hue, saturation and lightness. An equiprobability of each pixel is assigned to each class, this allows us to have a more constant reduction for the requested image and to have better filtering of the candidates. A more powerful and time consuming method can be used then for identifying the best candidate.**

***Keywords*— Image retrieval, HSL, Global descriptor, Color classification, space search reduction .**

I. Introduction

Today more and more images are available on the Internet. The exponential increase of the number of images makes it more difficult to retrieve images. Our objective is to use an image taken from the Internet and check if this image or a similar image exists in our database. In order to retrieve an image from the database we use CBIR (Content-Based Image Retrieval) methods. These methods extract local or global features from an image and use it to retrieve from a database. These features allow us to compute local or global descriptors. A global descriptor describes the entire image whereas a local descriptor describes a part of the image. Some very effective local descriptors include features like, SIFT or SURF, these methods are high dimensional descriptors composed of key points characterized by histograms of gradients [7] or Haar-wavelet responses [2]. But using these descriptors to identify an image causes several problems when we compare two images with each other. Searching a similarity between two high dimensional descriptors takes a lot of time [3]. To reduce the computing time one can use a database of key points instead of having a database of images [5]. The search is then reduced to finding similar key points. This solution allows us to retrieve an image quickly, but adding a new image in this database can take a lot of time depending on the existing size of the database [5]. Some methods reduce the number of key points in order to accelerate the image retrieval process [9], while others try to optimize by pre-filtering the database, so as to reduce the number of items to compare.

In this paper we propose to use a global descriptor that helps filter a database quickly. This descriptor needs to be able to compute and compare quickly. Moreover it allows us to update the database with new images quickly. In order to do this a simple and fast computing descriptor is required. As Presented by Aït Younes *et al.* [1] and Liu and Yang [6] we propose to develop a global descriptor based on the colors contained within images. But instead of using pre-distributed and static classes, based for example on HVS (Human Visual System), we propose to determine classes from a learning database, the classes which are equiprobable. Aït Younes *et al.* [1] proposes a method based on the fuzzy representation of color. This method computes some dominant colors on a HSL image, using a fuzzy classification for the hue, the saturation and the lightness. A descriptor representing the dominants colors of the image is computed and used in order to retrieve this image from a database. Liu and Yang [6] proposes a method based on histogram differences using images in CLE $L*a*b$ color space. The latter was designed to be perceptually uniform for the HVS and it allows us to measure the difference between two colors with the Euclidean distance. Two histograms are computed, the first one is a color difference histogram and the second one is an edge orientation difference histogram.

However, these methods result in heterogeneous bin sizes, i.e. some classes represent a large part of the population whereas others represent a very small part of the population. This heterogeneous repartition could result in an unequal reduction of the database according to the requested image, i.e. images with pixels in large bins will be less distinguishable than images with pixels in small bins. So instead of using pre-distributed and static classes, we define classes from a learning database and we generate classes that are equiprobable (still, e.g. based on HVS): in order to have a more constant reduction we want an equitable classification of the hue.

First, in Section II, we try to compute a classification on the HSL colors space that achieves an equal repartition of the population of pixels, so as to achieve an optimum reduction of the database. Then in Section III some results are shown of this implemented method and then compared to another classification [1]. Finally in Section IV we conclude and present some perspectives.

II. Proposed Hierarchical Image retrieval System

As illustrated in Fig. 1, the proposed method proceeds in three steps. The first step consists of learning the classes. The second step consists in computing a global descriptor based on the previously learned classes for each image of a testing

database. Then, the third step consists of performing a retrieval search of the image simply by computing its global descriptor and comparing it with those in the testing database.

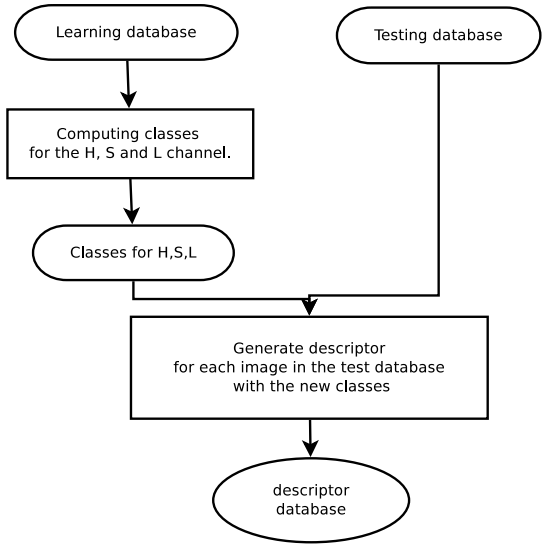


Fig. 1. Overview of the proposed method.

A. Classes computing

First we apply a preprocessing operation on the learning database is applied. This method uses images in HSL color space (Hue, Saturation and Lightning). Each channel has a value between 0 and 255. All the images are resized in order to have similar sizess in the database: all the images have a common size using the tallest dimension as a guide.

The learning database needs to be sufficiently large in order to represent all the variability of the images which can be found on the Internet.

Then the classes are computed. For each image, three histograms are computed: h_H for the hue, h_S for the saturation and h_L for the lightness. The hue of a pixel is not computed when it has a low saturation because the pixel looks like gray whatever its hue value. At the same time, pixels with low or high lightness can not give enough information on their hue or saturation. In order to avoid unusable hue and saturation values, a minimum saturation threshold S_{min} is fixed. For the lightness a minimum L_{Min} and maximum L_{Max} threshold are also defined. Any discarded pixel is considered gray, i.e. it has no hue, nor saturation, but only a lightness value.

Then, all histograms are organized into three new histograms H_H for all h_H, H_S for h_S and H_L for h_L. The next step consists of segmenting each histogram in order to have n_H classes for the hue, n_S for the saturation, and n_L for the lightness. Ideally, each bin of each histogram represents a similar part of the population:

$$H_i \simeq \frac{1}{n_H} H_H, \forall i \in [1..n_H], \quad (1)$$

$$S_i \simeq \frac{1}{n_S} H_S, \forall i \in [1..n_S], \quad (2)$$

$$L_i \simeq \frac{1}{n_L} H_L, \forall i \in [1..n_L], \quad (3)$$

where H_i, S_i and L_i the population class i of each channel.

The gray pixels are not used to compute hue and saturation classes. The saturation histogram has no value lower than $Smin$.

B. Classification method

In this section, an image descriptor is generated for each image in the testing database. This descriptor is inspired by the method presented in [1] without the fuzziness or choosing dominant colors. The fuzziness is ignored in order to simplify the descriptor computing. On another side, all the colors are used to build this descriptor. Fig. 2 shows how we proceed. This methods requires us to start with an image in RGB color space and with the learned classes previously computed. The first step is the same process as previously described to reduce the size of the image. Then, a conversion of this image from RGB color space to HSL color space is completed. The next step computes a matrix in order to build the descriptor. It is composed of 2 vectors, the first H encodes the hue, and is sized $n_H + 1$ (n_H classes plus one for the gray). The second descriptor Q is computed by using the saturation S and the lightness L. Its size is $n_Q = n_S \times n_L$.

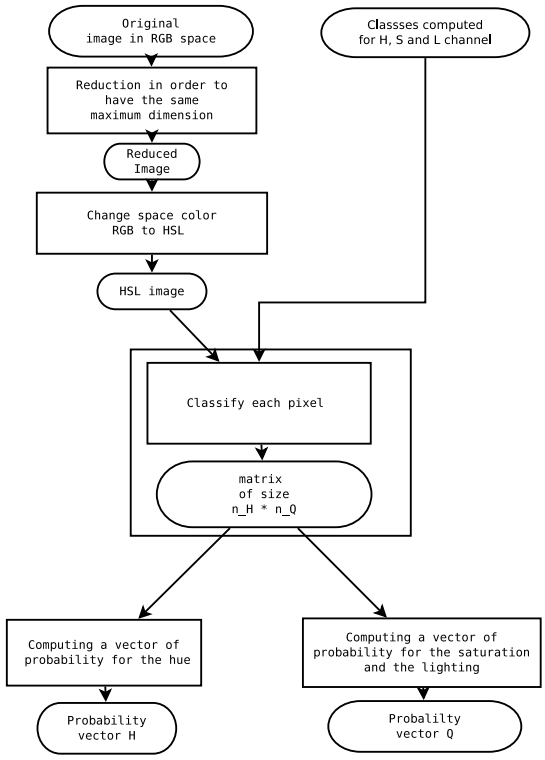


Fig. 2. Overview of the classification method.

For each pixel P of the image two values P_h and P_q are defined. P_h is the class of the hue and P_q is the class of

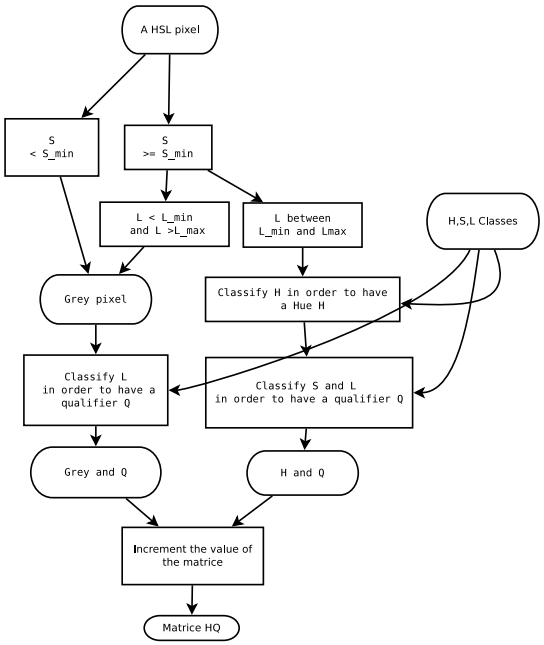


Fig. 3. Classification of pixels using H, S and L.

the qualifier. Gray pixels are hue encoded as $P_h = n_H + 1$. Qualifier classes Q are created using the saturation classes S and the lightness classes L. Classes Q are just the 1-D representation of a matrix SL of dimension $n_S \times n_L$. So Q is:

$$Q(i + j \times n_S) = S.L(i, j), \tag{4}$$

with $i \in [0..(n_S - 1)]$ and $j \in [0..(n_j - 1)]$

Gray pixels have no saturation, so a value of 0 are given to them (i.e. they belong to the class of saturation S_0).

All the pixel classes are saved in a matrix $\mathcal{HQ}$ of size $(n_H + 1) \times n_Q$, where $\mathcal{HQ}(P_h, P_q)$ is the number of pixel of these classes. All of these processes to compute this matrix is summarized in Fig. 3.

Then, a hue probability vector $\mathcal{H}$ of dimension $n_H + 1$ is extracted. Where H_i is the rate of the pixel of the image that belongs to the i^{th} color class. So H has the following proprieties:

$$\sum_{i=0}^{n_H} H_i = 1. \tag{5}$$

According to the same method, a qualifier vector $\mathcal{Q}$ is created, that also encodes probabilities:

$$\sum_{i=0}^{n_Q} Q_i = 1. \tag{6}$$

C. Similarity measure

In order to measure the similarity between two descriptors (and then between two images) and because the two vectors $\mathcal{H}$ and $\mathcal{Q}$ are probability vectors, the distance of Bhattacharyya [4] is used. It allows us to compute a distance between two discrete probability distributions. For the Hue, this distance is computed:

$$D_{BT}(A_{\mathcal{H}}, B_{\mathcal{H}}) = -ln(BC(A_{\mathcal{H}}, B_{\mathcal{H}})), \tag{7}$$

with:

$$BC(A_{\mathcal{H}}, B_{\mathcal{H}}) = \sum_{i=0}^{n_H} \sqrt{A_{\mathcal{H}_i} B_{\mathcal{H}_i}}, \tag{8}$$

where $A_{\mathcal{H}}$ and $B_{\mathcal{H}}$ are the hue probability vectors $\mathcal{H}$ for images A and B, as defined in eq. 5.

The distance $D_{BT}(A_{\mathcal{Q}}, B_{\mathcal{Q}})$ is also computed, with the vector $\mathcal{Q}$ of two images A and B (computed as in eq. 6). A more important weight is given to the distance on $\mathcal{H}$ because the hue has a greater dynamic than the saturation and the lightness i.e. the hue information is more useful to characterize an image (this is also the reason more classes are defined for the hue than for the saturation and the lightness). Finally two images A and B are considered similar if $D_{BT}(A_{\mathcal{H}}, B_{\mathcal{H}}) < \mathcal{H}_{min}$ or if $D_{BT}(A_{\mathcal{H}}, B_{\mathcal{H}}) < \mathcal{H}_{max}$ and $D_{BT}(A_{\mathcal{Q}}, B_{\mathcal{Q}}) < \mathcal{Q}_{min}$.

III. Experimental results

As presented in the Introduction, the goal of this work consists of presenting a method that reduces the number of images to compare in order to quickly retrieve the best image. Whatever the requested image the number of responses should be small. To test the proposed method a learning database, of 100 000 images taken from Flickr[1] are used to generate classes with equal parts of the pixel population. Then, these classes are used to classify another database of 100 000 independent test images (also from Flickr). Thes two databases are composed of color images.

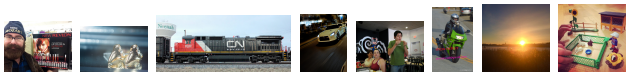


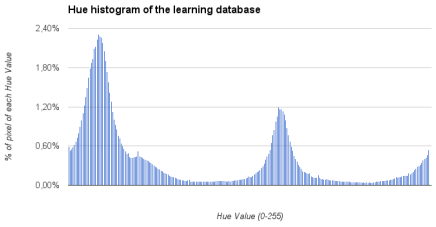
Fig. 4. Examples of images contained in the database.

In Fig. 4 some examples of images are presented. We can note variabilities of content.

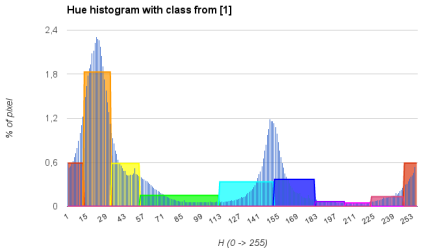
The hue is split into 9 classes in order to have an uniform repartition of the pixels read from the learning database. At the same time, saturation and lightness are each split into 3 classes. The histogram, illustrated in Fig. 5(a) shows the repartition of the hue in the learning database. These values are also classified using the method from [1] and Fig. 5(b) shows how the population isn't split into bins of a similar size. For example, the class "Orange" contains a large part of the population while The classes "Purple", "Magenta" and "Pink" represent only a little part of it. In contrast, our learned classes shown in the Fig 5(c) features a more equitable repartition of the classes, as shown by the areas of the classes. The original

[1] flickr.com

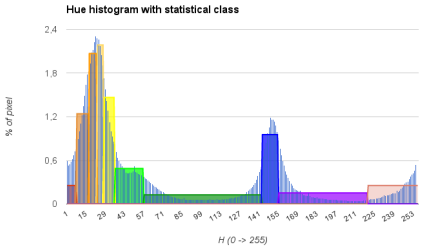
class ”Orange” is now subdivided into 3 classes whereas the classes ”Magenta”, ”Purple” and ”Pink” were grouped into only one.



(a)



(b)



(c)

Fig. 5. Hue histogram on the learning database (a). Classes from [1] (b). and classes computed with our method (c).

The classes computed with the learning database and the classes from [1] are shown in the Table 1, where hues are adjusted to the interval $[0, 255]$. Fig. 6 represents the same classes on a chromatic circle. As can be expected, the hues from [1] are more regularly spaced than ours, but their population is not split in uniform to classes.

	Learning Classes		Classes From [1]	
	Min	Max	Min	Max
# 1	-36 (219)	6	-9(246)	11
# 2	7	15	12	31
# 3	16	21	32	50
# 4	22	26	51	110
# 5	27	34	111	150
# 6	35	55	151	180
# 7	56	141	181	202
# 8	142	153	203	221
# 9	154	218	222	245

Table 1. Value of classes limits.

Fig. 7 shows how the population is split in the learning classes. The red line represents an optimal repartition (1/9 of the population, for 9 classes). As observed, the repartition with the learning is near to the optimal (so far on the learning database).

In table 2 shows the matrix HQ for a single image. The method presented in [1] characterized the image with only

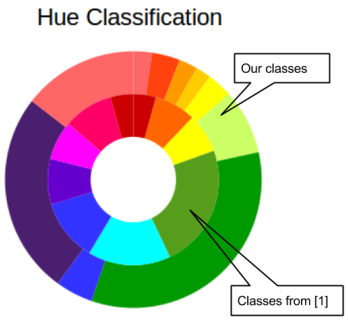


Fig. 6. Color classes representation on a circle within the inner circle classes from [1] and on the outer circle from our method

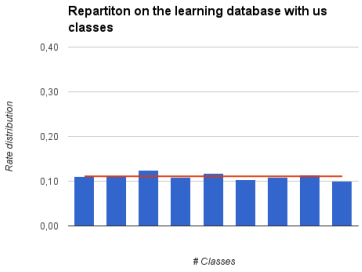


Fig. 7. Histogram of the repartition of hues on the learning database.

3 colors: orange, yellow and blue. No other class is present in the image. However, our method generates seven detailed classes to describe the image.

The proposed method is now validated against a test database of 100 000 individual images. The new repartition is similar to that of the training set, as presented in Table 3 and in Fig.8(a). Similarly, in Table 3 and Fig.8(a) shows how the classes from [1] are unequally distributed for this population.

Fig. 9(a) shows a repartition that is close to the optimal. Compared to the classes from [1], the proposed method generates more balanced repartition of the population. In the Fig. 9(b). It shows classes that have a repartition very far from the optimal one.

In order to quantify how well the classification works, the Shannon entropy is used. It measures how efficient an encoding is. In this paper the encoding is the classification and the effectiveness of it is measurements, i.e. if all the classes are equitably represented in the whole population. The entropy H is defined by the following equation:

$$H(X) = \sum_{i=1}^{n} P_i log_2 P_i, \tag{9}$$

where n is the number of classes and P_i the probability of a pixel to be an element of the i^{th} class. With fully equiprobable classes, each class would be $\frac{1}{9}^{th}$ of the database, so the optimal entropy would equal $\sum_1^9 \frac{1}{9} log_2(\frac{1}{9}) = log_2(\frac{1}{9})$.

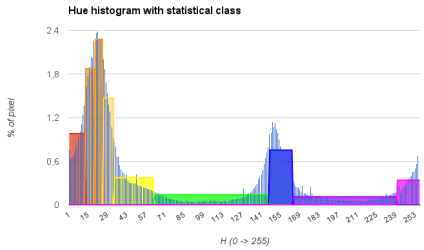
	Classes From [1]									
	L_0			L_1			L_2			Vector H
	S_0	S_1	S_2	S_0	S_1	S_2	S_0	S_1	S_2	
	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	
Red	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Orange	0,16	1,27	6,63	5,65	11,97	14,42	0,43	2,70	1,18	44,41
Yellow	0,01	0,00	0,00	1,99	1,51	1,62	2,41	2,23	2,00	11,77
Green	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Cyan	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Blue	0,00	0,00	0,00	2,97	0,00	0,00	0,06	0,00	0,00	3,03
Purple	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Magenta	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Pink	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Gray	27,22	0,00	0,00	3,88	0,00	0,00	9,69	0,00	0,00	40,79
Vector Q	27,39	1,27	6,63	14,48	13,49	16,04	12,59	4,93	3,18	100,00

	Our Classes									
	L_0			L_1			L_2			Vector H
	S_0	S_1	S_2	S_0	S_1	S_2	S_0	S_1	S_2	
	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	
Red-Pink	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Orange-Red	0,00	0,00	0,03	0,00	0,00	0,03	0,00	0,00	0,00	0,07
Orange	0,00	0,02	2,50	0,00	0,07	1,86	0,00	0,00	0,00	4,45
Orange-Yellow	0,00	0,03	3,12	0,01	0,52	4,74	0,00	0,01	0,77	9,20
Yellow	0,00	0,00	0,26	0,77	2,03	2,25	1,80	6,52	17,06	30,69
Yellow Green	0,00	0,00	0,00	0,16	0,01	0,01	2,90	2,43	6,24	11,77
Green	0,00	0,00	0,00	0,00	0,00	0,00	0,02	0,00	0,00	0,02
Blue	0,00	0,00	0,00	0,41	0,00	0,00	2,59	0,00	0,00	3,01
Purple	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Gray	27,05	0,00	0,00	0,74	0,00	0,00	13,00	0,00	0,00	40,79
Vector Q	27,05	0,06	5,92	2,09	2,63	8,90	20,33	8,96	24,07	100

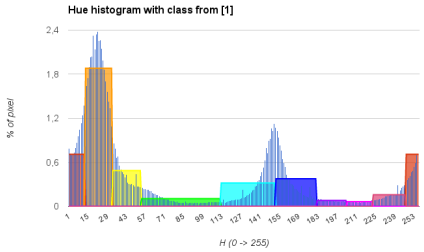
Table 2. Image with 2 matrix repartition.

	Population Between Classes		
	Learning database	Test database	
	Us classes		Classes from [1]
#1	11.03%	11.60%	14.84%
#2	11.18%	11.44%	37.54%
#3	12.43%	12.44%	9.37 %
#4	10.95%	11.79%	6.51 %
#5	11.73%	10.20%	13.29%
#6	10.28%	10.86%	11.33 %
#7	10.94%	10.03%	1.80 %
#8	11.46%	12.16%	1.32 %
#9	10.00%	9.49%	4.29%

Table 3. Computed Classes.



(a)



(b)

Fig. 8. Test database: class repartition with our method (a), versus that of [1] (b)

Let us compare each image from the test database to check how many images are considered similar using the learning classification and [1].

The classes of the proposed method have a smaller number of (false) similar images and a lower variance (see table 5). This shows that the computed classification is more accurate than that of [1]. Fig 10 shows a histogram of the numbers of similar images. It is observed that a lot of images have a small number of similar images when using the learned classes. With the classes from [1] the number of similar images is larger when a lot of images are similar.

Let us use a bigger database of 1 000 000 images and search 10 000 inside to check how many images are considered similar using the learning classification.

We can notice (see table 6) that we have always a low variance and that the number of similar images increase by a factor of 10 corresponding to the increase factor of the size of database. The goad to keep a few part of the database in order to compare is preserved, regardless of the size of the database only 0.5 % of the database is kept as potential candidates.

IV. Conclusion

In this paper, we propose a method that computes thresholds of classes for the hue, saturation and the lightness of the pixels based on a learning database. The resulting classes provide an equitable repartition of the pixels, which turns to be a more efficient way to classify images than a static set of classes based on HVS [1]. This method does not directly retrieve an image like CBIR methods based on local descriptors, but it does filter a large database very quickly and reduces drastically the number of candidates to consider. As a pre-filtering system, it speeds-up significantly the images retrieval, as more powerful and more time consuming methods can be used on a reduced number of candidates. But this reduction is constant in percentage. If the size of the database rises, the number of candidates rises too. In the current version, this descriptor does not reduce enough when very large databases are filtered and it is not infallible to some corruptions.

One way to improve the method would be to apply multiple descriptors in sequence, so as to further reduce the number

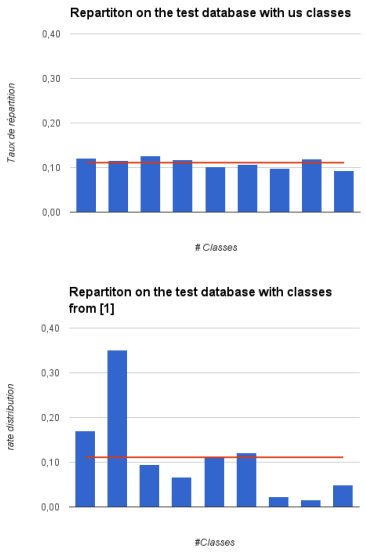


Fig. 9. Histogram of the hues with the learning classes (a) and with the classes from [1](b) on the testing database.

	Shannon entropy (in bits/hue)
Optimal Repartition $log_2(\frac{1}{9})$	3.170
Our classes in the learning database	3.167
Our classes in the testing database	3.163
Classes from [1] in the testing database	2.699

Table 4. Entropy values.

of remaining candidates, even with very large databases. These descriptors should be orthogonal to each other for best effect, i.e. only a very few candidates shall remain after their independent results are merged and even if each of them generates a significant number of candidates.

Another element to address is to improve the filtering with corrupted images. The method proposed here is able to cope with some changes in the image like rotation, flipping or low jpeg compression because it is statistic, hence it is insensitive to geometric changes, like moving the pixels around. But other common image processing can have a significant impact , like

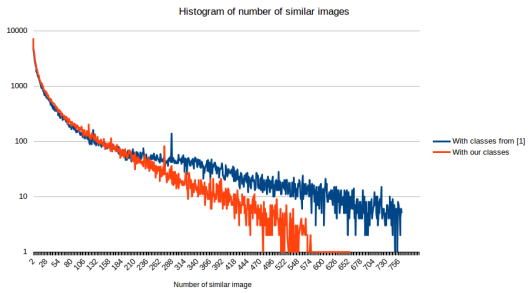


Fig. 10. Histogram of similar images (Y axis is in logarithmic scale).

	mean of similar images	variances
Our threshold	53.18	6362
Threshold extract of [1]	88.96	21001

Table 5. Similar image images on a database of 100 000 images.

	mean of similar images	variances
Our threshold	503.27	53752

Table 6. Similar images on a database of 1 000 000 images.

many of the usual filters from Instagram-filter[2] that modify the hue, the saturation and/or the lightness. The resulting corrupted image is much harder to recover from the database, even though we expect that fuzzy classification could help in this respect.

The ultimate goal is of course to highly reduce the number of candidates while being infallible to common corruptions, and this is again a case where a combination of miscellaneous descriptors should prove very efficient.

REFERENCES

[1] A. Aït Younes, I. Truck, and H. Akdag. Image Retrieval using Fuzzy Representation of Colors. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11(3):287–298, 2007.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[3] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *In Proc. IEEE Conf. Comp. Vision Patt. Recog*, pages 1000–1006, 1997.

[4] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[5] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. *CoRR*, abs/1102.3828, 2011.

[6] G. Liu and J. Yang. Content-based image retrieval using color difference histogram. *Pattern Recognition*, 46(1):188 – 198, 2013.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[8] R. Schettini, G. Ciocca, and S. Zuffi. Color in databases: Indexation and similarity. In *Proc. of first International Conference on Color in Graphics and Image Processing (CGIP 2000)*, pages 244–249, 2000.

[9] D. Zuchun. An Effective Keypoint Selection Algorithm in SIFT. *nternational Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(2):155–164, 2013.

[2]instagram.com