

Image Identification using Local Descriptors (Task 1)

Eduardo Valle
ETIS — ENSEA/UCP
6 ave. du Ponceau BP 44F
95014 Cergy-Pontoise France
+33 1 30 73 66 10
valle@ensea.fr

Sylvie Philipp-Foliguet
ETIS — ENSEA/UCP
6 ave. du Ponceau BP 44F
95014 Cergy-Pontoise France
+33 1 30 73 66 10
philipp@ensea.fr

Matthieu Cord
LIP6 — UPMC
104 ave. du Président Kennedy
75016 Paris France
+33 1 44 27 71 39
matthieu.cord@lip6.fr

ABSTRACT

Our participation in the Task 1 of the ImageEVAL competition was motivated by our previous work in image identification for cultural institutions. Our approach is based in local descriptors computed around points of interest. We used KD-Trees with the best-bin-first traversal in order to match the descriptors. To eliminate false positives, we ensured the consistency in the matching with the RANSAC algorithm and an affine transform model. We obtained good results.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *search process*.

General Terms

Algorithms, Measurement, Performance.

Keywords

ImageEVAL, image identification, SIFT, KD Tree, best-bin-first, RANSAC.

1. INTRODUCTION, MOTIVATION

Our motivation in participating in the Task 1 of ImageEVAL comes from our previous work on image identification for cultural institutions.

Image identification is the task of taking a given query image and finding out the original from where it possibly derives, together with any set of relevant metadata, such as titles, authors, copyright information, etc.

More formally, if we have a set of candidate images I , a set of transformations T , and a query image q , we want to find the set of possible original images from q , $O = \{o_1, o_2, \dots, o_n\}$ such as:

$$q = t(o_i) \text{ for some } t \in T$$

The set of transformations may include rotations, scale changes,

shearings, croppings, occlusions, compressions, photometric changes, colorimetric changes, gamma correction, addition of noise, blurring, and any combinations of those. If the set I contain more than one image that can be transformed into q , multiple valid answers should be considered, and the set O will have more than one element.

Institutions possessing large sets of images, like museums, archives and press agencies, are often asked to perform the identification of clippings, images in books, thesis and even postcards, where the references are missing, too summary, outdated or incorrect. In those cases, the visual information is the only reliable evidence for the identification of the document, in order to retrieve the original and its metadata. But because the collection is too large, it will take many hours for a trained operator, using only conventional search tools, to retrieve the desired image.

Another application for image identification is the detection of copyright infringement. In this context, the users believe that a database contains images of their possession. They want to confront this suspicious dataset against their own image set and find all intersections.

Content-based image retrieval methods are a strong candidate for image identification, because they rely only on the visual aspects of the documents, needing not keywords, annotations or watermarks, which can be unavailable, lost or removed.

2. DESCRIPTION OF TASK 1

The detailed description of the tasks of ImageEVAL can be found in [9]. Here we give a summary description.

Task 1 was divided in two subtasks 1.1 and 1.2. Both considered the following transformations:

- Three rotations;
- Two “translations” (actually two croppings);
- A shearing;
- A conversion to greyscale;
- A conversion to the negative colours;
- A dessaturation of the colours;
- A half-toning using the Floyd & Steinberg transform;
- A softening using the mean filter;
- An aggressive JPEG compression;
- An addition of impulsive noise;
- Addition of a text label inside the image;
- Inclusion of a border around the image;
- Embedding of an image inside another.

In subtask 1.1, we had a dataset of 2.500 images submitted to the 16 transformations (plus the identity). The final database had a size of 42.500 images. The subtask consisted of using the *original* (untransformed) image as a query, and retrieving all the related transformations from the database.

In subtask 1.2, the dataset consisted of the 2.500 *original* images. From the query, a *transformed* image from the dataset, the original image should be retrieved.

3. OUR APPROACH

Local descriptor-based methods have been shown to be of great value for image identification, due to their strong robustness to occlusions, cropping and geometric distortions. [7][1] Instead of creating a single descriptor per image, those methods will identify a large number of PoI (Points of Interest), and compute a local descriptor around each one of those points.

In order to perform the identification, first, in an offline phase, which is done only once, the image set is prepared: each image has its PoI detected and described, and the descriptors are indexed in a large database in order to facilitate the matching. Then, in the online phase, the PoI of the query image are detected and described, and each descriptor is matched with the descriptors in the database. Each matched descriptor votes to the image to which it belongs.

This method is robust, because the descriptors are many. Even if some descriptors are matched incorrectly, giving votes for the wrong images, only a correctly identified image will receive a significant amount of votes. [Figure 1]

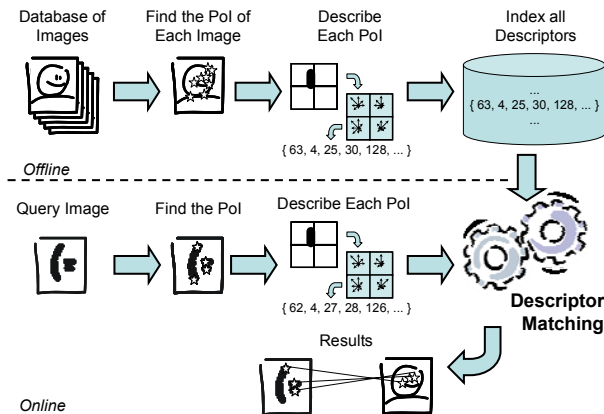


Figure 1. Image identification using PoI.

Unfortunately, the multiplicity of descriptors brings also a performance penalty, since hundreds, even thousands of matches must be found in order to identify a single image. The matter is made worse when the descriptors are high dimensional, which makes each match operation very expensive.

3.1 Points of Interest / Descriptors

All points of interest (PoI) methods are based on a common ground: the detection and description of some points of the image, which provide more information than the others.

To detect PoI, one can use many criteria: local contrast, local maximization/minimization of certain functions (Laplacian, gradient, etc.), threshold over a curvature function (Harris, Hessian, etc.), etc.

What is required from the PoI is repeatability: a high fraction of the points should be found at the same locations, even if the image suffers geometric, photometric or colorimetric deformations. Once a point is detected, a descriptor must be generated for indexation purposes. Usually, only a small patch around the point is analyzed, generating a local descriptor. If the method is to be robust, the descriptor must be invariant to the concerned deformations.

We have chosen SIFT (*Scale Invariant Feature Transform*), which is one of the most robust descriptors available. [6]

The SIFT PoI are local extremes (minima and maxima) in a scale-space composed by differences of Gaussians of progressively larger standard deviations, as explained below. Their description is based on a normalized histogram of the gradient directions around the point. To achieve rotation invariance, the patch is rotated towards the most important direction of the gradient.

The SIFT algorithm is a complex one. Its multiple steps carry out the task of detecting highly repeatable points and giving them a highly discriminating description. At several stages, a threshold or a parameter must be chosen (which is done mostly empirically in [5]). We feel that it is important to give a detailed description of the method in order to provide the reader an idea of what is at stake:

1. If in colour, the image is converted to greyscale. To avoid aliasing, the image is pre-processed, by doubling its size and then smoothing it with a Gaussian filter.
2. A set of progressively stronger Gaussian filters is applied to the image generated in step 1. This generates a series of images $G_1..G_n$. The standard deviations are chosen according to the equation below, where σ_1 is the standard deviation applied to the first image:
$$\sigma_i = \sigma_1 \times 2^{(i-1)}$$
3. Adjacent images are subtracted to generate a series of differences of Gaussian $D_1..D_{n-1}$, e.g., $D_1=G_2-G_1$, $D_2=G_3-G_2$, etc.
4. PoI candidates are the local extremes in the 3D grid formed by the differences of Gaussians series. A point is extreme if and only if its greyscale value is greater or lesser than all its 26 neighbours. [Figure 2]

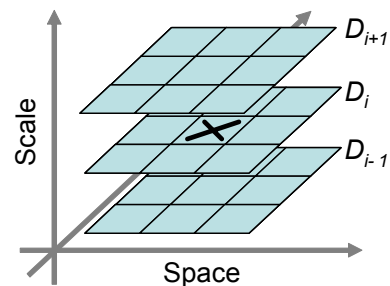


Figure 2. The value of interest point X must be greater or lesser than all its 26 neighbours

5. The scale of the point is determined by the scale of the plan where it was found. Its precise position is computed by interpolation of the scale-space grid.
6. A minimum contrast threshold is applied to filter out low contrast points.

7. Edge responses are eliminated to promote corner points (points of high curvature value), using the ratio of the eigenvalues of the Hessian matrix.
8. A direction is assigned to each surviving point. This is done by choosing the smoothed image G_i closest to the scale of the point, taking the gradient of this image, and computing a histogram of the values around the point. The peak direction is selected. If there are other significant peaks, the point can get multiple directions.

The choice of the difference of Gaussians is motivated by the fact it approximates the Laplacian function, which performs very well in the composition of a scale-space. Lowe depicts a clever scheme for computing the Gaussians without the heavy processing cost associated to large standard deviations. [5]

Next, it is necessary to give the points an invariant description, in order to allow indexation and retrieval. SIFT will proceed as follows:

1. The gradient image of the smoothed image closest to the scale of the point is taken, just like we have done in the step (8) of the detection.
2. A Gaussian weighting function is set around the point, with a standard deviation equal to half the size of the descriptor window.
3. The descriptor window is divided in zones. We illustrate here a descriptor with 4 zones (2x2), though the real descriptor has 16 zones (4x4). [Figure 3]
4. In each zone, we compute a histogram of the gradient values, with 8 orientations. To avoid boundary effects, a tri-linear interpolation is used to distribute the values of the samples.
5. The descriptor is composed by all entries of all histograms. This generates a quite large vector (32 in our illustration, 128 in the real implementation).
6. To minimize the effects of illumination changes, the vector is adapted. The affine variations are accounted by normalizing the vector. The saturation effects are reduced by applying a maximum threshold to the gradient magnitudes of the histogram, and renormalizing it.

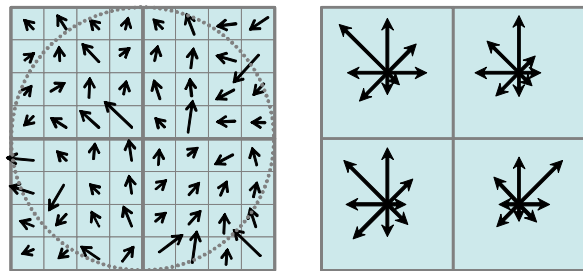


Figure 3. SIFT Descriptor. The gradient values around the point are weighted by a Gaussian function (dotted circle). Each zone has the histogram computed for 8 orientations

3.2 Matching the Descriptors

When using global descriptors, querying the base consists of finding the most similar descriptor, using a predetermined criterion. When employing local descriptors, however, each operation is composed of many sub-queries, one for each PoI present in the query image.

For each query point, one searches for the set of k points in the base (the most similar), and counts one vote for each corresponding image. One expects that, even if many images may receive some votes, only the pertinent ones will have a significant amount of votes.

Because the process consists of several inspections (a query image typically has a few hundred points), it is important to perform the search as fast as possible.

Unfortunately, the methods used search for the most similar descriptors, known as *nearest neighbours search*, have high computational costs on high dimensional spaces. In order to overcome this problem, one must use an *approximate nearest neighbour search*, in which one are not guaranteed to find the exact k points most similar to the query descriptor, but still expects to have most of the correct answers.

From the several methods described in the literature, we have chosen the *KD-Tree* with the *best bin first* search strategy, which has been successfully used with SIFT before. [2]

The *KD-Tree* is a classic data structure for multidimensional indexing. Basically, it is a binary search tree where each node of the tree splits the search space along a given dimension of the space. [4]

To build a *KD-Tree* from a base set B in the space D , we start by choosing the dimension to be split. The criterion of choice may vary from choosing the dimension accordingly to the level on the tree, choosing it randomly, or picking the dimension where the data are more spread. The point of splitting must then be chosen by selecting an element as the *pivot* of the splitting. The pivot may be the median element on the splitting dimension, or the element nearest to the middle of the range of the dimension, or something else. A node is created with the pivot and the splitting dimension, the left subtree is created recursively with all element lesser than or equal to the pivot and the right subtree is created with all elements greater than the pivot. In some implementations, if the number of elements in a subtree is smaller than a threshold, a leaf is created contained a *bucket* which simply contains all remaining elements. [Figure 4]

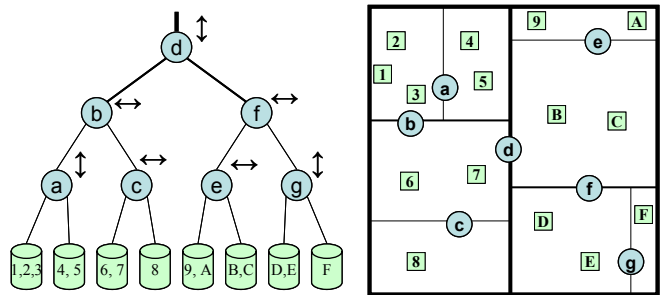


Figure 4. A sample 2-dimensional *KD-Tree* with buckets, and its associated space

In order to perform the nearest neighbours search in a *KD-Tree*, we start by finding the leaf nearest to the query. To do this, we traverse the tree from top to bottom, choosing always the subtree which is on the same side as the query on the splitting dimension. Once we reach a leaf, we find the nearest elements to the query and we compute the distance to those elements. If this distance is

smaller than the distance between the query and any other regions of the space, we have found the nearest neighbours. Otherwise, we explore the sub-trees corresponding to the other regions of the space in order to check if there are nearer elements. The search stops when no region can possibly contain an element nearer than the elements already found.

The best bin first is one possible strategy of traversing the KD-Tree, in which we visit first the nodes where it is most probable we will find the nearest matches. In this strategy we start by exploring the sub-trees which have the corresponding regions nearer to the query, and stop the search after a certain number of regions have been explored. The search is approximate and it is possible to trade-off between efficiency and precision by setting the number of regions to visit. [2]

3.3 Pre-processing

Before computing the PoI of the images, we have to smooth them, not only to remove the omnipresent JPEG artefacts, but also to avoid the disastrous effect of the high frequencies introduced by transformations such as the impulsive noise and the Floyd & Steinberg half-toning. Without this pre-processing, we will have a huge number of spurious PoI, spoiling the results.

The most efficient way to eliminate those high frequencies is to reduce the size of the image, using an interpolation algorithm. We experienced with both reducing the images to half the linear dimensions (a fourth of the pixels) and a fourth of the linear dimensions (a sixteenth of the pixels).

3.4 Checking the Consistency

The original SIFT algorithm includes a criterion of significance, which tries to retain only the matches which are discriminating. This is done by comparing, for every query point, the distance to the *most* similar target and the *second most* similar target. If those distances are too close, the match is ignored. Since the underlying hypothesis, that there is only one correct match in the base for every point, is false in our case, we had to eliminate this criterion.

Many images will receive a few votes, and it is not straightforward to eliminate the false positives by thresholding the number of votes. One way to get rid of the false positives is to check the geometric consistency of the matching of the PoI. In our work we adopted the RANSAC algorithm [3] checking if the points corresponded to an affine transform.

3.5 Treating the Negative Images

SIFT is invariant to all the transformations proposed, except the conversion to the negative colours. This is because, contrarily to the other photometric/colorimetric transforms, it radically changes the direction of the gradient (actually inverting it).

Since this was the *only* failure case, instead of seeking to make SIFT invariant to the inversion of the gradient, it seemed to us more reasonable to solve the problem by making two researches: one using the unmodified query, and one using the negative of the query.

3.6 Combining the Results

At the final point of the processing, we have four rankings of the images:

- Unmodified query, brute vote count;
- Negative query, brute vote count;
- Unmodified query, RANSAC-consistent vote count;
- Negative query, RANSAC-consistent vote count.

Theoretically, we could use only the RANSAC-consistent results, but since some legitimate images have too few points of interest (because they are too smooth) and some image embeddings cause large occlusions, removing most matches, sometimes there will be too few matches for the RANSAC algorithm to perform reliably.

We created two ways of combining these results. The first method, which we will call *naïve*, privileges the unmodified query:

1. Output all the unmodified query, RANSAC-consistent images, up to the 50 places available in the result list;
2. Output all the negative query, RANSAC-consistent images, up to the remaining available places in the result list;
3. Output all the unmodified query, brute images, up to the remaining available places in the result list.

The other method does not privilege the unmodified query. Instead, it uses the vote count to merge the two result lists, putting first the result images with higher vote count. We call this method *symmetric*:

1. First, merge the unmodified and negative query, RANSAC-consistent lists, intercalating the results, and putting first those images with the higher vote count, up to the 50 places available in the result list;
2. Then, merge the unmodified and negative query, brute lists, intercalating the results, and putting first those images with the higher vote count up to the remaining places available in the result list.

4. THE RUNS

In order to analyse the results, let us summarise the methods used in each one of the runs.

The information we give is the number of run NN (where the name of the run is *iev_t011_run_q01_etisNN_off* for the subtask 1.1 and *iev_t012_run_q01_etisNN_off* for the subtask 1.2), the size of the bucket in the KD-Tree, the number of descriptors examined per search in the KD-Tree, the percentage of the resize in the pre-processing of the images (relative to the linear dimensions), the method used in the merge of the partial results, the relative and the absolute positions of the run in the competition.

For the task 1.1, we have:

# Run	Bucket	Comp.	Resize	Merge	Rel. Pos.	Abs. Pos.
01	5.000	20.000	25%	Naïve	2	13
02	5.000	20.000	25%	Symmetric	1	6
03	10.000	40.000	25%	Symmetric	3	14
04	10.000	40.000	25%	Naïve	4	16

For the task 1.2, we have:

# Run	Bucket	Comp.	Resize	Merge	Rel. Pos.	Abs. Pos.
01	5.000	20.000	25%	Naïve	3	11
02	5.000	20.000	25%	Symmetric	2	8
03	5.000	20.000	50%	Symmetric	1	1
04	5.000	20.000	50%	Naïve	1	1

The best runs in each subtask are in **boldface**.

5. ANALYSIS

The analysis from the results of both subtasks leads to several immediate conclusions:

The reduction to 25% of the size is excessive, while the reduction to 50% is adequate as a pre-treatment;

The symmetric method of merging the partial results works better than the naïve method;

Using smaller bucket sizes in the KD-Tree slightly improves the results, even if fewer descriptors get to be visited.

Some results are not that evident, but the careful observation of the problematic images in each subtask can be revealing.

We believe that if very few matches remained (due to excessive smoothness, large occlusions or low contrast), the RANSAC algorithm performed badly; especially because of the need of at least 7 matches to determine reliably an affine model (though in theory 3 should suffice).

We also observed some randomness in the results of the subtask 1.1, which suggest that our RANSAC algorithm needs better parameterisation, perhaps exploring more intensively the space of possible solutions.

6. CONCLUSIONS AND FUTURE WORK

We are generally satisfied with our general approach, which gave us nice results in the Task 1, both in terms of efficacy and of efficiency (though the later was measured rather informally).

A better parameterisation of the RANSAC, or the use of a model fitting algorithm which performs better in the presence of few samples would have granted better results. It would also have been better to use to 50% resize in the task 1.1.

One question we have posed is which backup strategy to use in images which generate very few, if any, PoI, and which sometimes are of interest for cultural databases (like some paintings of modern art which consist of large areas of flat colour).

Our research since the competition has been concentrated in the development of better descriptor matching algorithms. We had both developed a much more efficient version of the KD-Tree,

using overlapping segments [8], and an entirely new method (still unpublished) based on space-filling curves.

7. ACKNOWLEDGEMENTS

Mr. Valle is sponsored by a CAPES scholarship through the CAPES/COFECUB program.

8. REFERENCES

- [1] Amsaleg, L., Gros, P., Berrani, S-A., “Robust Object Recognition,” in *Images and the Related Database Problems, Multimedia Tools and Applications*, v. 23 n. 3, p. 221–235, August 2004
- [2] Beis, J.S. and Lowe, D., “Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces”, in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, p.1000, June, 1997.
- [3] Fischler, M.A. and Bolles, R.C., “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography” in *Communications of the ACM*, Vol. 24, pp. 381-395, 1981.
- [4] Friedman, J.H., Bentley, J.L. and Finkel, R.A., “An Algorithm for Finding Best Matches in Logarithmic Expected Time” in *ACM Transactions on Mathematical Software*, Vol. 3, N. 3, pp. 209–226, 1977.
- [5] Lowe, D., “Distinctive Image Features from Scale-Invariant Keypoints,” in *International Journal of Computer Vision*, Vol. 60, N. 2, pp. 91–110, 2004.
- [6] Mikolajczyk, K. and Schmid, C., “A performance evaluation of local descriptors” in *International Conference on Computer Vision & Pattern Recognition*, Vol. 2, pp. 257–263, June, 2003.
- [7] Valle, E., Cord, M. and Phillip-Foliguet, S., “Content-Based Retrieval of Images for Cultural Institutions Using Local Descriptors,” in *Geometric Modeling and Imaging — New Trends (GMAI'06)*, 2006.
- [8] Valle, E., Cord, M. and Phillip-Foliguet, S., “Descriptor Matching for Image Identification on Cultural Databases,” in *International Conference on Image Processing (ICIP'07)*, 2007. (In Prelo)
- [9] Techno-Vision. *ImageEVAL — Information about the evaluations. V.3*. Update for the official campaign. August/2006.
- [10] Young, H. P. and Levenglick, A.. “A consistent extension of Condorcet's election principle.” *SIAM Journal on Applied Mathematics*, 35(2):285–300, 1978.