Visual Image Retrieval and Localization

Yannis Kalantidis Giorgos Tolias Evaggelos Spyrou Phivos Mylonas
Yannis Avrithis
Image, Video and Multimedia Systems Laboratory
National Technical University of Athens
School of Electrical and Computer Engineering
Iroon Polytexneiou 9, 15780 Zografou, Greece
e-mail: {ykalant, gtolias, espyrou, fmylonas, iavr}@image.ntua.gr

1 Introduction

The popularity of social networks and web-based personal image collections has resulted to a continuously growing volume of publicly available photos and videos. Users are uploading, describing, taggging and annotating their personal photos. Moreover, a recent trend is to also ”geotag” them, that is to mark the location they were taken onto a web-based map. Consequently, this growth of image collections has created the need for fast, robust and efficient sytems, able to analyze large-scale diverse and heterogeneous visual content. This growing need for automatic metadata generation, concept detection, search and retrieval has boosted research efforts towards these directions. The work presented herein is a web-based system that aims not only to the retrieval of visually similar images, but also to determine the location they were taken by exploiting the available socially created metadata. This system makes use of a visual vocabulary and a bag-of words approach, in order to describe the visual properties of an image. Moreover, geometric constraints are applied, in order to extend the bag-of-words model towards more accurate results.

We begin by describing some related work in the field of image retrieval, in order to present both the relation and the novelties of the presented system in comparison with the existing techiques. Numerous extensions to the bag-of-words model have been proposed recently. For example, [26] tries to explore techniques to map each region to a weighted set of words and thus allow the inclusion of features that were lost during the quantization phase of previous systems. In [6], images are segmented into regions which are then classified into visual words, using a variety of features and a mapping between visual words and keywords is learned using an Expectation-Maximazation method. In [17], an approach that uses Wavelets to extract image features and Hidden Markov Models (HMMs) to learn the association of those features to the keywords describing the images is presented. [3] proposes a randomized data mining method that finds clusters of spatially overlapping images and applies it on on large databases in order to retrieve is capable to retrieve near-duplicates of images. Theapproach of [11] uses a visual words' description of images and then tries to create a more accurate description by using Hamming embedding and weak geometric consistency constraints.

In certain retrieval problems, the extraction of global features (from the entire image) or local (from regions/patches of the image) presents good results. However, in the case of applications that aim to retrieve images based on the objects they contain, these techniques present serious limitations. Thus, most recent algorithms begin with the determination of some interest points within an image. These points carry some properties such as invariance to various image transformations, illumination etc. They

continue by defining regions in the neighborhood of these points and extract the descriptors within them. We should note here, that while some of the papers presented herein deal solely with object detection, the techniques mentioned are also applicable in the area of image retrieval, when the goal is to retrieve images based on the objects/places they contain. In [4], a representation of local image structure and a matching scheme, both insensitive to many appearance changes is presented. This method is applied on two-view matching problems of images from different modalities. Moreover, [7] presents a method to learn and recognize object class models from unlabeled and unsegmented cluttered scenes in a scale invariant manner. In this work, objects are modeled as flexible constellations of parts. A probabilistic representation is used for all aspects of the object: shape, appearance, occlusion and relative scale. An entropy-based feature detector is then applied for region selection within the image. Also, in [8], object recognition is based on affine invariant regions. Segmentation and recognition are achieved simultaneously. In [14], the problem of near-duplicate image retrieval is tackled with a parts-based representation of images using distinctive local descriptors extracted from points of interest, which are invariant under several transformations. Moreover, the work presented in [15], uses parts affinely rigid by construction. Object detectors are trained by identifying groups of neighboring local affine regions whose appearance and spatial configuration remains stable across multiple instances. In [24], a novel feature matching method aims to tackle efficiently high-dimensional problems. The work presented in [27] is a large-scale object retrieval system. Therein, the query is a region of an image and the system retrieves images that contain the same object as the one contained in the user's query.

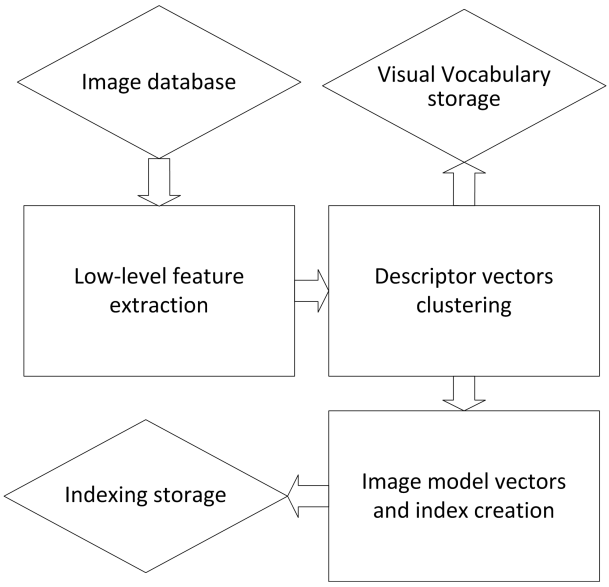
The architecture of the presented system is depicted in Fig. 1. In the *offline* procedure (Fig. 1(a)), a visual vocabulary is created, in such a way that contains the most common visual local patterns of the dataset. Each database image is afterwards represented in terms of this visual vocabulary through a model vector. This process takes place only once and for all the images of the given database. In the *online* procedure(Fig. 1(b)), the user uploads or chooses a query image from the database. From this image, points of interest and visual descriptors are extracted and the model vector representing the image is calculated using the same process. By comparing the model vector to those of the database, the system retrieves similar images.

The presentation of the presented visual retrieval system is organised as follows: Section 2 presents the local features extracted from images. The process of creating a visual vocabulary and indexing is presented in Section 3. The matching procedure between two images is described in Section 4 followed by the geometric consistency check in Section 5. The implemented application that locates an image is presented in Section 6 followed by the demo features and a demo walk-through in section 8.

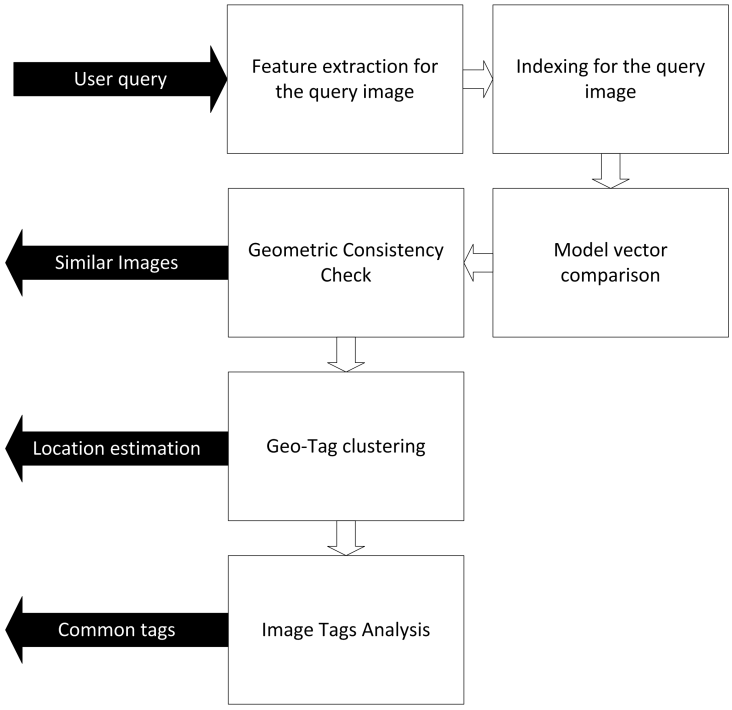
2 Local features

For the representation of the visual content of a given image a set of interest points is selected and visual features are extracted locally, from their surrounding area. Since the goal is to choose scale invariant interest points, their localization is carried out on a gaussian scale-space. In our system, the SURF (Speeded-Up Robust Features)[1] features have been selected to capture the visual properties of the images. These features have been proven to achieve high repeatability and distinctiveness. Apart from that, their extraction speed is very fast, when compared e.g. with the SIFT features [19]. An example of the extracted SURF features is depicted in figure 2.

For the localization of interest points, a fast approximation of the Hessian matrix is adopted, which exploits the use of integral images. Then the local maxima of the Hessian matrix determinant are chosen as interest points. This blob response maxima process is carried out on several octaves of a Gaussian



(a)



(b)

Figure 1: The architecture of the proposed retrieval system. Online (1(a)) and offline (1(b)) process.

scale-space. The correct scale is automatically selected also from the Hessian determinant, as introduced in [18]. For the exact point localization, an efficient non-maximum suppression algorithm is used at a $3 \times 3 \times 3$ intra-scale neighbourhood [22].

The SURF descriptor captures the intensity content distribution around the interest point detected

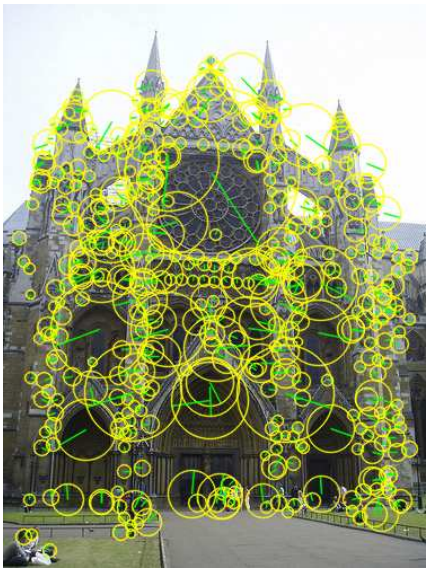


Figure 2: Interest points extracted with the SURF Fast Hessian detector. The size of the yellow circle and the green line denote the scale and the dominant orientation, respectively.

with the aforementioned process. The first order Haar wavelet responses are computed with the use of integral images, resulting to a 64-dimensional feature vector. In order to achieve rotation invariance, a dominant orientation is determined. This is selected as the direction that maximizes the sum of the Haar-wavelet responses in a sliding window of size $\pi/3$ around the interest point neighborhood.

To compute the descriptor, a square area around the interest point with $20 \times s$ side length is selected and divided in 4×4 blocks, with s denoting the interest point scale. Thus, the descriptor is also scale invariant. At each one of the 16 blocks, 4 values that correspond to the sum of the x,y, $|x|$ and $|y|$ first order Haar wavelet responses in a 5×5 grid in the block are extracted. To make the descriptor robust to contrast changes, the descriptor vector is turned into a unit vector.

It clear that the selection of the aforementioned low-level feature extraction scheme combines speed with robustness to scale, rotation and contrast changes. The fact that the extraction time is very small comparing to other approaches allows the use of this scheme to a real time system as the one presented herein. Robustness to changes of the image guarantees that the system would be able to match two images depicting the same object under certain visual changes.

3 Visual Vocabulary and Indexing

In this section we present the method we follow in order to create a visual vocabulary. The words contained in this vocabulary will be used for the representation of the visual properties of a given image. To understand the notion of a visual vocabulary, one should consider it as an equivalent to a typical language vocabulary, with an image corresponding to a part of a text. In the same way that text may be decomposed to a set of words, an image can also be decomposed to a set of *visual* words. Then, in order to compare two images, their corresponding visual words may be compared instead. Thus, it is interesting to create a visual vocabulary in such a way that parts of images could be meaningfully assigned to visual

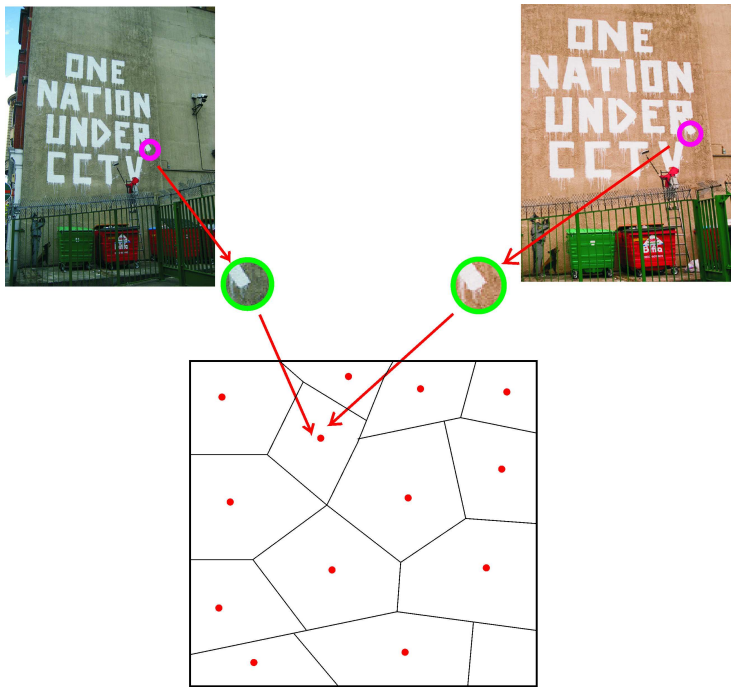


Figure 3: Two points extracted from different images, corresponding to the same visual word.

words. Fig. 3 depicts two regions of interest extracted from two different images, that correspond to the same visual word. The visual vocabulary is presented as the Voronoi cells of the clustered space of visual words. We should note here that due to their polysemy, visual words cannot be as accurate as natural language words. This means that a given visual word cannot be directly corresponded to a concept, but it can represent a part of a significantly large number of concepts.

3.1 Visual Vocabulary Construction

To create the visual vocabulary, a clustering process is followed. More specifically, the well-known K-means clustering algorithm [20] is applied on the SURF descriptors corresponding to a very large number of points of interest. If the number of the points to be clustered is significantly large, clustering using the K-means algorithm becomes a very slow task. For example, clustering of 5M of points (which are typically extracted from 10K of images) requires a few days of processing. However, to efficiently deal with large scale retrieval problems, the size of the vocabulary should be in the order of a few tenths of thousands of visual words [12],[28]. Thus, in order to rapidly create an appropriate vocabulary, the clustering process is performed on a smaller subset, carefully selected to contain the most representative images. After constructing the visual vocabulary, each image has to be represented with a description that captures its relation to all the words of it. We should also emphasize here that in order to create a visual vocabulary able to perform well in more than one domains, the images from which the regions of interest are extracted, have to be an diverse and as possible heterogeneous and moreover its size has to be significantly large.

3.2 Nearest Neighbor search using a k-d tree

The goal is to describe a given image based on the visual words it contains. This description will be in a vector form and will be denoted as "model vector". In particularly, for the formulation of a model vector, we need to find the visual word that is the closest in terms of descriptor vector to each one of the image's points. To do this fast and efficiently we rely on the k-d tree structure. The structure of k-d trees has been widely used in information retrieval during the last decades [10], [2]. This data structure is a binary tree, which stores a finite number of k-dimensional data points and has been widely applied in the field of computer learning [21] and neural networks [25]. Within the presented work, k-d trees are used in order to find the closest visual word of every point of interest, which is typically a very difficult and time consuming task due to the large dimension of points.

Given N k-dimensional elements, the k-d tree is constructed by partitioning the space iteratively, one dimension at a time. At each iteration, the feature space is divided into two subspaces along the selected dimension. This is repeated until each subspace contains a single point. This process creates a tree which allows a very fast search for all data points. The height of this tree is $log(n)$.

In the case of the presented system, a k-d tree is created by the centroids of the clusters that are created by the clustering process. The dimension of the centroids is equal to 64. These centroids comprise the visual words of the visual vocabulary. This tree is created once and for all the images that we would like to index. Then, within the process of formulating the model vector, for each point of the given image, its nearest neighbor is determined using the k-d tree.

3.3 Model Vector Formulation

After constructing the visual vocabulary, a given image is represented in terms of it using a model vector. We assign the most similar visual word of the vocabulary to each descriptor of an image point. Then, a histogram is constructed for each image, which counts the occurrences of the visual words of the vocabulary within it. The model vector of an image I is a N_{vw}-dimensional vector, where N_{vw} is the size of the visual vocabulary:

$$mv_I = \left[tf_I(0), tf_I(1), \ldots, tf_I(N_{vw})\right] \tag{1}$$

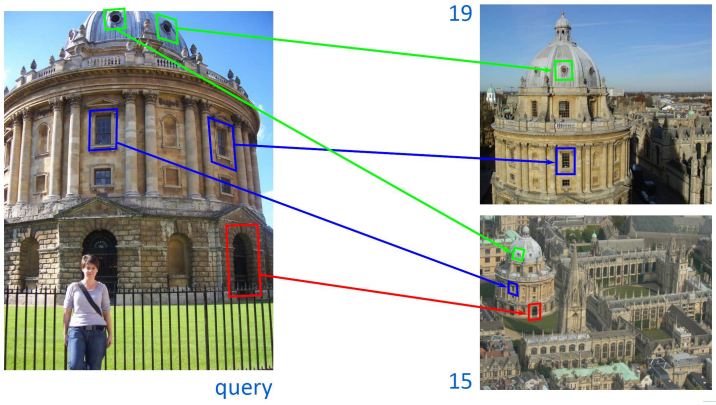
where $tf_{I,i}$ denotes the number of times that the visual word i was selected as a nearest neighbor of one of the interest points extracted from image I. In order to find the closest visual word to a point, the aforementioned k-d tree structure is used.

The histogram of visual word appearance frequencies is then normalized and its non-zero values are stored in an index which resembles to the technique of inverted files, widely used to fast text retrieval [31],[32]. Each image is then represented by its corresponding visual words and the frequencies they occur. From this point, when it is mentioned that a visual word appears within an image, this would mean that this visual word is the nearest neighbor of one or more of the image's interest points.

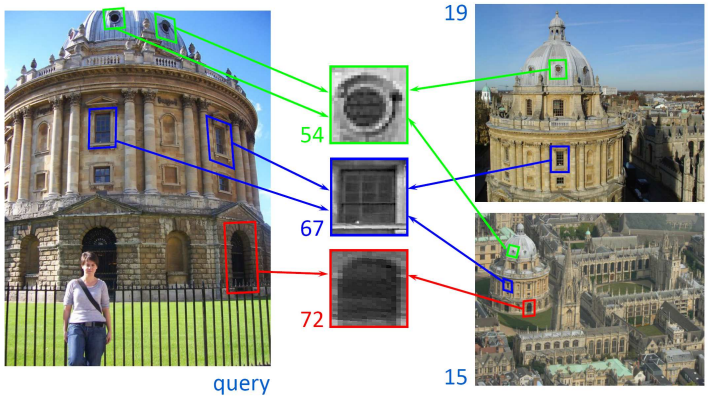
In presence of large vocabularies of over 10^4 visual words, the model vector is very sparse. It can have at most as many non zero values as the image's interest points, when every one of them is assigned to a different visual word. So, in practice, only the non-zero values of the model vector are stored to save storage and gain speed.

Since this indexing process is inspired by techniques applied in the task of text search, in addition to the term frequency (tf), which is the frequency of a given term in a document, inverse document frequency (idf), can also be used. This case is studied in section 3.4.

The process of querying an image database without and with a visual vocabulary is depicted in Fig. 4. In the first case the comparison of the local descriptors is performed immediately for two images



(a)



(b)

Figure 4: Matching of two images without a visual vocabulary (4(a)) and with a visual vocabulary (4(b)).

and after exhaustive comparisons in the whole database, the closest regions are found. In the latter case, for every image of the database all points have been assigned to appropriate visual words of the visual vocabulary. Thus, for a new query, its points have to be assigned to the closest visual words of the vocabulary. After this process, two images are considered to be similar if their points are assigned to similar visual words.

3.4 Inverse Document Frequency and Stop List

Inverse Document Frequency is another technique used in information and text retrieval [29], which during the last few years has been applied to image retrieval, either along with language processing [13], or using visual dictionaries [30] [5].

The model vector of an image has taken, up to now, only the frequency of the appearance of a visual word as a nearest neighbor to any of the interest points into account. We define the *Inverse Document Frequency* or *idf* as:

$$idf_k = log\frac{N}{n_k} \tag{2}$$

where N is the total number of the images of the given collection (the size of the database) and n_k the

number of the appearences of the visual word VW_k as the nearest neighbor to all points of all images in the database. Thus, idf acts as a weighting scheme, which identifies the most and less frequent visual words of the entire collection. The model vector can now be formulated as:

$$v_I = \left[idf_0 \cdot f_I(0), \ldots, idf_K \cdot f_I(K)\right] \tag{3}$$

with idf_k being the idf value of the visual word k.

It is obvious that the most common visual words, the ones with the smallest idf values, are not discriminative and their absence would facilitate retrieval. On the other hand, the rarest visual words are in most of the cases a result of noise and can be distracting to the retrieval. To overcome these problems, in some cases a stop list is created that includes the most and the least frequent visual words of the image collection. Using this list, the presence of its visual words is ignored, thus resulting to even sparser model vectors and smaller image representations.

4 Matching

In order to compute the similarity between images, two similarity measures are used. The first one is the most commonly used inner product of the model vectors. If v_Q and v_I are the model vectors for the query image Q and a database image I respectively, their matching score can be computed with

$$s_2(Q, I) = \langle v_Q, v_I \rangle = \sum_{i=0}^{K} v_Q(i) v_I(i) \tag{4}$$

where N_{VW} is the visual vocabulary size, and $v_x(i)$ corresponds to the value or term frequency of the visual word i in image x. The second similarity measure, that also proved to yield better results, is histogram intersection, discussed also in [5]. Regarded as histograms, the similarity of the two model vectors for the query and a database image can be computed as:

$$s_1(Q, I) = \sum_{i=0}^{K} \min(v_Q(i), v_I(i)) \tag{5}$$

For both matching measures, and since the vectors can be very sparse, the inverted file scheme is used in order to decrease matching time.

When a user query reaches the system, then the local low-level features are extracted from the query image and the model vector is computed. Then the similarity of the query model vector with all database model vectors is computed, and the N most similar images, the images with the highest such value, are either returned to the user as similar, or become candidates for geometric consistency check, as explained below.

5 Geometric Consistency Check

When the retrieval process considers only the model vectors that represent the visual content of images, sometimes fails to produce accurate results. This is because the bag-of-features approach totally ignores the geometry of the extracted interest points. Two images can contain similar visual words, but in a totally different spatial layout one from the other. Thus, the inclusion of a geometry consistence check would be very useful. The method that was adopted is the RANSAC algorithm [9]. This method can find the

homography between two images given a set of tentative point-to-point correspondences, in presence of many false such correspondences that are also called *outliers*. In fact the RANSAC algorithm estimates the homography that maximizes the number of *inliers* that is the set of correspondences that support the model.The ransac algorithm is described in section 5.1.

It is obvious that RANSAC algorithm as presented before, relies a lot in the correspondences of the points, which will be provided initially. These correspondences are not available, thus need to be calculated. A method that finds the nearest neighbors is not efficient, since it is a very time-consuming procedure that needs to be computed online. However, we can exploit the correspondences between points and visual words, in order to create tentative point correspondences between two images. This requires an additional indexing process, according to which, for every image and every visual word that appears in it, we store the locations of the points that yield this visual words as nearest neighbor. We should note here that this process is very fast.

This procedure, however, introduces many false correspondences, due to the quantization effect of the bag-of-words approach. So if for example a visual word appears 4 times in one image and 5 in the other, then with out method $4 \times 5 = 20$ correspondences will be formed, instead of 4 correct ones. Taking this into account, a rejection procedure follows, called *neighbor check*, that rejects correspondences between points whose neighborhoods does not match[30]. This means that in order to keep a tentative correspondence as valid, we require some of its spatially neighboring points to also have a valid correspondence between them. An example of the RANSAC inliers between two images in the presence of partial occlusion is shown in Fig. 5.

Since this method is computationally expensive, for the purpose of the implemented system we chose not to apply it in the whole database, but in the most similar images, in terms of their model vectors. It is important to clarify that RANSAC does not retrieve images but *re-ranks* them, based on the spatial layout of their interest points. This approach appears to minimize the number of false positives. Using an appropriate threshold on the retrieved results, a higher precision can be easily achieved.

Figure 5: The RANSAC inliers found between the two images.

5.1 Homography estimation using RANSAC

The well known RANSAC (RANdom SAmple Consensus) algorithm is applied between two images in order to determinate the homography between them, in presence of many outliers. By homography, we mean the perspective transform that maps any given point x_i of the first to the corresponding point x'_i of the latter. Given the set of correspondences between two images, that is the pairs $x_i \leftrightarrow x'_i$, we can define the homography matrix H as:

$$x'_i = Hx_i \tag{6}$$

In general, estimating this homography has proven very useful in tasks such as stereoscopic camera calibration, a case where the images captured by both cameras differ only by means of a perspective transform. This simple idea is extended herein and instead of the two images taken by a stereoscopic camera, we consider the case of the same object captured by different points of views. We should expect that in this case the the homography will result to a large number of false correspondences since the variation of the viewpoint is significantly higher. To overcome this, the RANSAC algorithm is applied, since it is able to estimate correctly the homography even with a large number of false correspondences present. The goal of this method is not only to estimate the homography matrix, but also to classify the points into two categories: Those that represent correct correspondences (inliers) and those that represent false correspondences (outliers).

In general, when the goal is to fit a model into data, our initial random sample should contain a number of points equal to the number of the model variables. In the homography case, the model requires at least 4 initial points. Given the correspondences of the points between two images, the basic RANSAC algorithm that makes an estimation of the homography between two images is as follows:

- 4 points are selected in a random manner. This is the minimum number of points needed to define a homography. These points are initially assumed to be inliers.
- Using the model that was estimated during the previous step and a distance parameter t, the number of the points that satisfy the model is calculated. Their distance to the point estimated by the homography model should be smaller than the predefined threshold t:

  $$d^2_{vertical} < t^2 \tag{7}$$

  where $t^2 = F_m^{-1}(\alpha)\sigma^2$ and $F_m(k^2) = \int_0^{k^2} \chi^2_m(\xi)d\xi$ denote the probability density distribution for the error, which we assume that follows χ^2_m with m degrees of freedom. The points for which the aforementioned relation stands are the inliers of the model. A typical value for α is 0.95. This means that the probability that a given point is an inlier is equal to 95%.
- random quadruplets of points are sampled repeatedly, and after every iteration the maximum number of inliers up to that point is kept together with the corresponding model.
- This process is repeated until a predefined number of iterations is reached, or the value of the probability that new inliers could be found using another model falls beneath $\eta\%$. This probability is defined as:

  $$\eta = (1 - P_I)^k \tag{8}$$

  where k denotes the number of the correspondences between two images and P_I is the probability that a sample consisting solely from inliers is selected:

  $$P_I = \frac{\binom{I}{m}}{\binom{N}{m}} = \prod_{j=0}^{m-1} \frac{I-j}{N-j} \approx \epsilon^m \tag{9}$$

$\epsilon = I/N$ denotes the ratio inliers / points. A typical value for ϵ is equal to 0.99.

- When the aforementioned relation is satisfied, the homography model is redefined, using all inliers selected from the previous steps.

6 Application: Geographical Location of Images

During the last few years one of the trends in the image collections of the World Wide Web is the inclusion of metadata which aim to provide a more complete description. Typical metadata contain a free text description, some representative tags and some metadata related to the geographical position that the image was taken (Geo-tag). A *Geo-tag* consists of the geographic coordinates: the longitude and the latitude, and is either extracted automatically through GPS from some cameras or manually defined by the user. In the latter case case, users mark their position on a map, using web-based applications. A very popular example is Flickr[1]. It is estimated that each month, almost 3M of geotagged images are uploaded.

If a user queries with a landmark image in a large database of Geo-tagged images, then the top retrieved results will probably contain the landmark that the query image depicts. Those correctly retieved images are expected to have near identical Geo-tag values. Little variance is expected since geo-data can be defined by users and also because the same building can be photographed by different distances (using the appropriate camera lens). However, the estimated Geo-tag for the query image is expected to be within the larger consistent subset of the result images.

In the presented system we exploited the available metadata of geo-tagged images. Several experiments have been performed in a collection of 60000 images collected from Flickr, depicting monuments and characteristic buildings of London and Berlin. We should note that all images used in the experiments had been geotagged by Flickr users.

Within the implemented application, a user is able to execute a query of an image depicting among others a monument of London or Berlin. The system responds not only with visually similar images, but also with an estimation of the location of the query image, and frequent tags that can accompany it. The complete demo features are listed in section 7.

To find the most consistent subset mentioned above, an agglomerative clustering algorithm is used, on the 2-dimensional geographic coordinates of the retrieved images. An agglomerative clustering merges in one the points that lay in distance closer to a predefined value t. Setting t to represent the area around a building or monument, each cluster of Geo-tags will denote a location. The query landmark images, that are in most of the times a large subset of the retrieved results, will probably yield the largest cluster. The algorithm used herein is the Reciprocal Nearest Neighbor (RNN) [23],[16].

Images uploaded by users contain free text annotation in terms of tags describing the visual content of the image. This information can be exploited to provide the the user with some tag recommendations for an image query. Using a stop list analogy the most and less frequent tags are suppressed and the most common tags in the final set of retrieved images are the system's tag recommendations.

To visualize the query results, the graphical interface of the Google Maps application has been used, as depicted in Fig. 6. In this map, the red marker denotes the geographical position of the query image, and the blue markers denote the positions of the images that match the query in terms of their visual features. Some indicative retrieval results are also presented in Fig. 7. Apart from the visual features of the retrieved images, their most common tags have been also extracted. Those tags are depicted in Fig 6.

[1] `http://www.flickr.com`

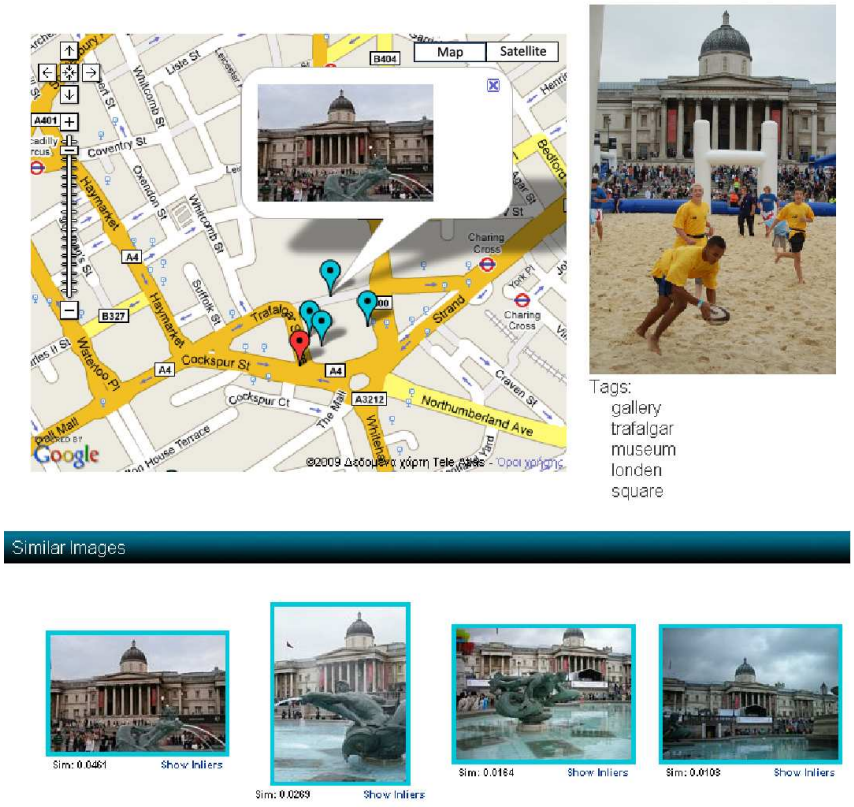


Figure 6: Upper row: A Google map is depicted on the left, estimating the location of the query image depicted on the right. Under these images, the most common tags are estimated. Lower row: Visually similar retrieved images.

7 Demo features

The ViRaL *web-based demo* provides the following features:

1. the user may select a city image and use it as visual query,
2. the execution of the visual query returns a set of visually similar images obtained from the image database,
3. returned images are geo-tagged and thus localized on Google Maps,
4. a set of user-generated textual tags are also returned to aid in the query image semantic description,
5. a similarity metric is also presented to the user below each retrieved image,
6. together with further implementation information, such as the calculated feature points, inlier features and their correspondences.

The web-interface is implemented in php, and the system core is in C++. For the creation and the querying to the database, MySql has been used. Currently, our database contains 60000 geo-tagged

Figure 7: Query Results. Images that are estimated to be in the same location of the query image have a light blue border.

images, downloaded from Flickr [2]. The system requires around 11Mb of memory and data space for the indexing of 1000 images. A typical query runs at about 15 seconds on a typical Intel Pentium Core2Duo P8400 processor, with 3Gb of RAM.

8 Demo walk-through

The first page a user encounters when entering the VIRaL system is shown in Fig. 8. It is an introductory screen, that suggests some sample queries to execute. On the top, under the headlines, there is the overhead menu, that gives the user two options, to either return to this introductory screen at any time (by clicking on *Flickr Dataset*) or to let the system pick a random sample of images from the dataset as query samples (by clicking on *Random Sample*).

The whole demo interface is simple, and the user can immediately start a query with any of the images displayed, just by clicking on them.

After executing the query, the user is redirected to a results page, similar to the one shown in Fig. 9 or Fig. 10 .

The query image is depicted on the top-right of the page. The most frequent user defined tags of the top retrieved images are also presented under the query image.

On the left of the query image, there is a fully operational Google Map, with the zoom and scroll features enabled. Inside we may observe three types of pins, specified by different colors. The light-blue pins correspond to the retrieved images that are predicted to have the same visual content as the query image. The grey pins correspond to the retrieved images that are predicted not to contain the same visual content as the query images due to their significantly irrelevant location. Finally, the red pin denotes the location of the the query image. It is obvious that the ideal result is a red pin lying near or within a cluster of light-blue pins, i.e. to those images that are predicted to depict the visual content of the query image (the same monument/building/landmark).

[2] `www.flickr.com`

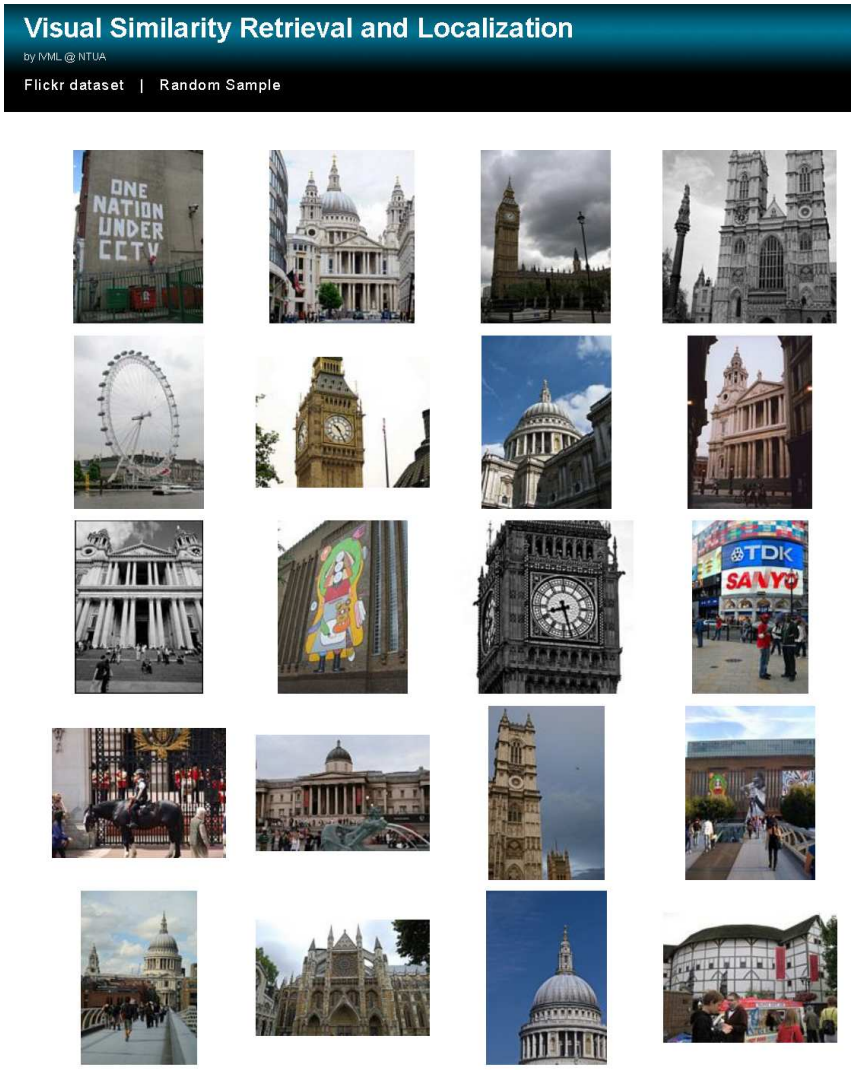


Figure 8: The first page a user encounters when enters the VIRaL system.

The most similar results are presented, sorted by their similarity. By clicking on any result image, the user is able to launch a new query with this image. Along with each retrieved image, its similarity value to the query image is presented. There is also the option to visualize the details of the retrieval process, by clicking on the option *Show Inliers*. This way, the user is redirected to a window similar to the one depicted in (Fig. 11).

The borders of each of the retrieved images have matching colors to the corresponding pins, in order to assist the user to perceive their relevance to the query image. Figure 10 demonstrates a retrieval example where although the visual similarity fails (since there are not other images that contain the same building view in the database) the location estimation is correct.

At the details screen, the user is able to view at the query image at full size, and next to it a copy with the extracted SURF features printed on. For each feature the scale and orientation are also visible. Scale is denoted by the size of the corresponding circle, while orientation is determined by the green radius within it. Below this pair of images, the results of the geometric check, between the query image and

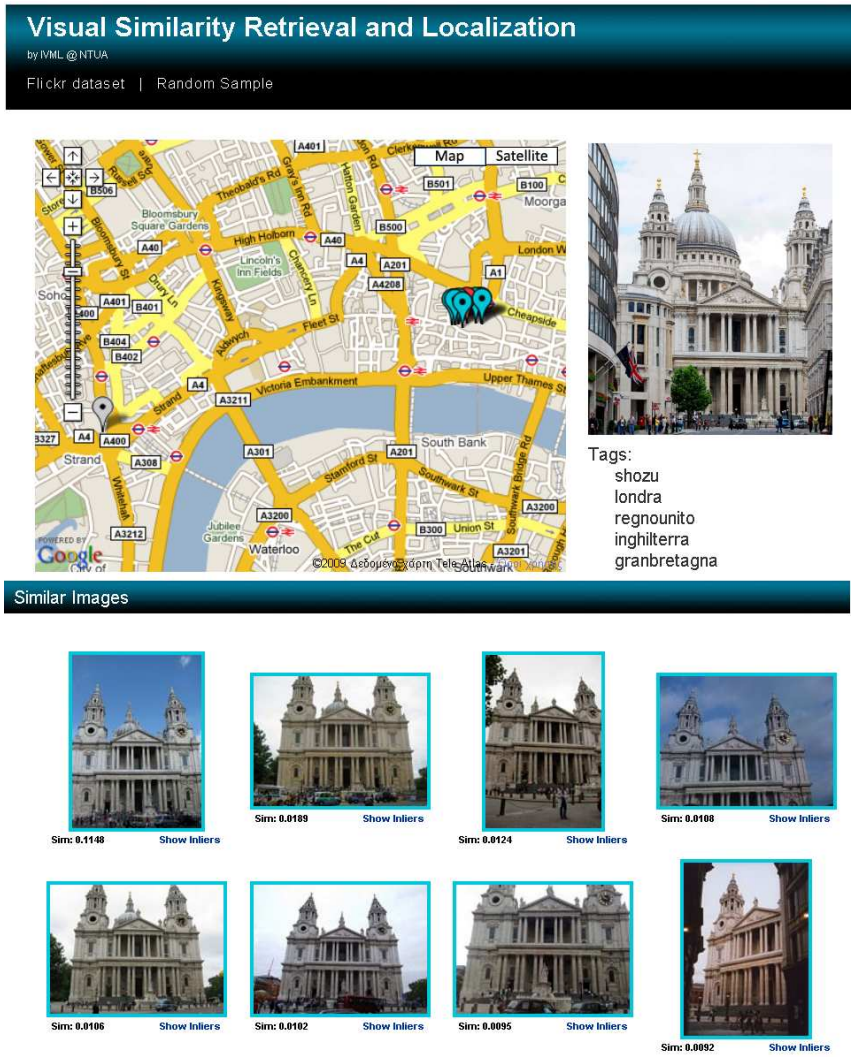


Figure 9: A page containing the results of a given query.

the one selected are presented. These are the inliers predicted by RANSAC. It is clearly visible at the example depicted in Fig. 11, that between two different views of the same monument, the percentage of the tentative correspondences that match some homography model is very high.

Acknowledgments

This research was partially supported by the European Commission under contract FP7-215453 - WeKnowIt.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[2] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] O. Chum and J. Matas. Web scale image clustering: Large scale discovery of spatially related images. Technical report, Technical Report CTU-CMP-2008-15, Czech Technical University in Prague, 2008.

[4] O. Chum and J. Matas. Geometric hashing with local affine frames. In *Computer Vision and Pattern Recognition*, volume 1, pages 879–884, 2006.

[5] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *Proceedings of the British Machine Vision Conference*, 2008.

[6] P. Duygulu, K. Barnard, JFG De Freitas, and D.A Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *Lecture Notes in Computer science*, pages 97–112, 2002.

[7] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern recognition*, volume 2, pages 1575–1589. IEEE Computer Society; 1999, 2003.

[8] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision*, 67(2):159–188, 2006.

[9] M.A. Fischler and R.C. Bolles. Random sample consensus - a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] J.H. Freidman, J.L. Bentley, and R.A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[11] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *ECCV, Oct*, 2008.

[12] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In Andrew Zisserman David Forsyth, Philip Torr, editor, *European Conference on Computer Vision*, LNCS. Springer, oct 2008. to appear.

[13] M.A.S.T. Kanade. Video Skimming and Characterization through the Combination of Image and Language Understanding. In *Proceedings of the 1998 International Workshop on Content-Based Access of Image and Video Databases (CAIVD'98)*, page 61. IEEE Computer Society Washington, DC, USA, 1998.

[14] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACMA C Multimedia*, volume 9, pages 869–876. IEEE Computer Society; 1999, 2004.

[15] S. Lazebnik, C. Schmid, and J. Ponce. Semi-local affine parts for object recognition. In *British Machine Vision Conference*, volume 2, pages 959–968, 2004.

[16] B. Leibe, A. Leonardis, and B. Schiele. Robust Object Detection with Interleaved Categorization and Segmentation. *International Journal of Computer Vision*, 77(1):259–289, 2008.

[17] J. Li and J.Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 2:1075–1088, 2003.

[18] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Print on Demand, 1994.

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

[20] J. MacQueen. B. 1967.Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 1–297.

[21] A.W. Moore. An intoductory tutorial on kd-trees. Technical report, Technical Report.

[22] A. Neubeck and L. Van Gool. Efficient Non-Maximum Suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, 2006.

[23] C.F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21(8):1313–1325, 1995.

[24] D. Omercevic, O. Drbohlav, and A. Leonardis. High-dimensional feature matching: Employing the concept of meaningful nearest neighbors. In *Ieee 11Th International Conference on Computer Vision*, pages 1–8, 2007.

[25] S.M. Omohundro. Efficient algorithms with neural network behavior. *Complex Systems*, 1(2):273–347, 1987.

[26] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization improving particular object retrieval in large scale image databases. *Image*, 9(14):15–17.

[27] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. C V P R*, volume 3613, pages 1575–1589, 2007.

[28] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.

[29] Y. Rui, TS Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, 1997.

[30] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1470, Washington, DC, USA, 2003. IEEE Computer Society.

[31] D.M.G. Squire, W. Müller, H. Müller, and T. Pun. Content-based query of image databases: inspirations from text retrieval. *Pattern Recognition Letters*, 21(13-14):1193–1198, 2000.

[32] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.

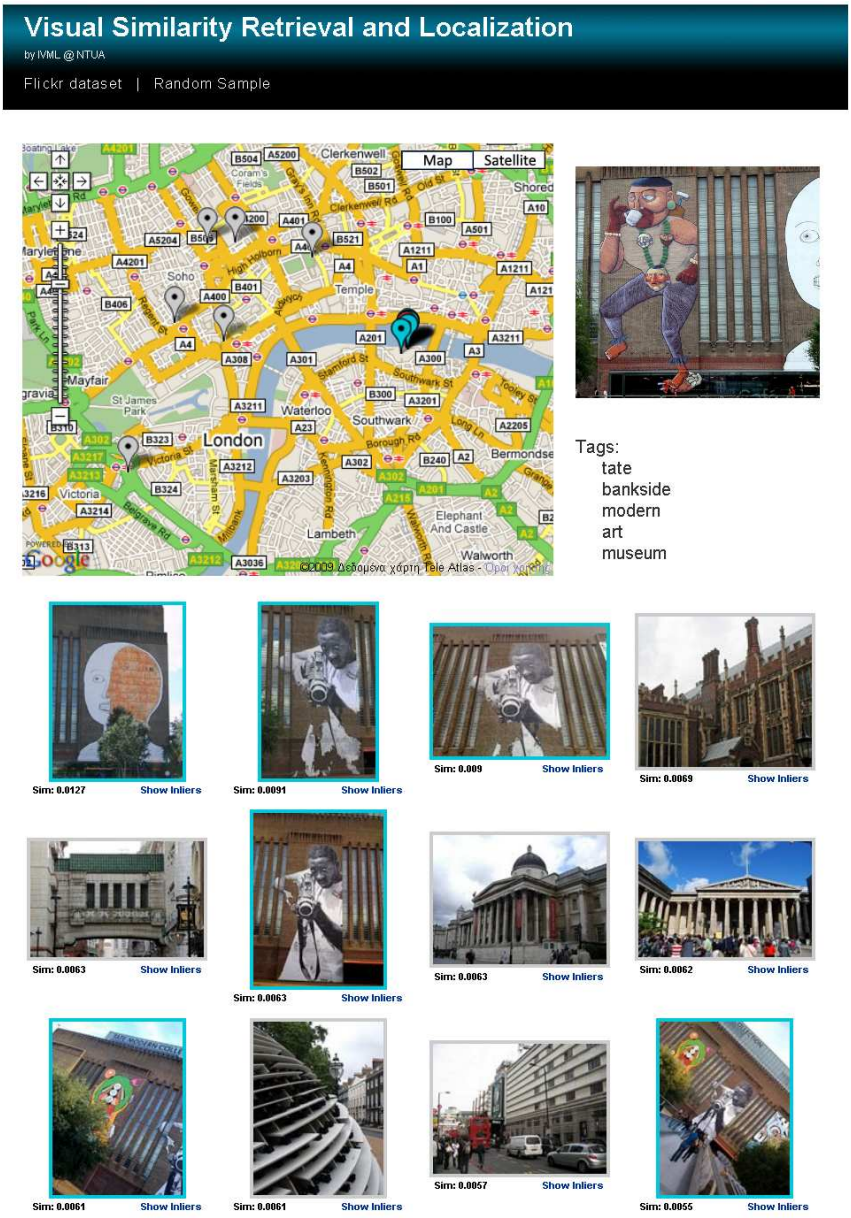


Figure 10: A page containing the results of another query.

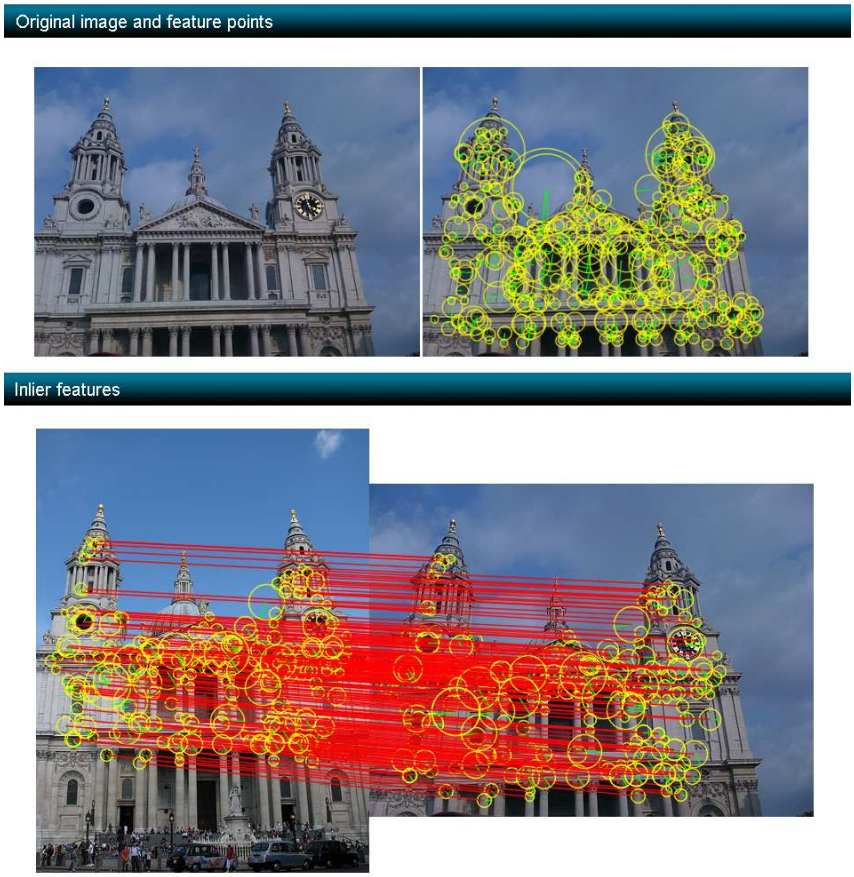


Figure 11: A page containing the query image, the SURF features that have been extracted and the correspondences of the inliers among those features to a given image of the dataset.