VIRaL: Visual Image Retrieval and Localization

Yannis Kalantidis · Giorgos Tolias · Yannis Avrithis · Marios Phinikettos · Evaggelos Spyrou · Phivos Mylonas · Stefanos Kollias

October 22, 2010

Abstract New applications are emerging every day exploiting the huge data volume in community photo collections. Most focus on popular subsets, *e.g.* images containing landmarks or associated to Wikipedia articles. In this work we are concerned with the problem of accurately finding the location where a photo is taken without needing any metadata, that is, solely by its visual content. We also recognize landmarks where applicable, automatically linking them to Wikipedia. We show that the time is right for automating the geo-tagging process, and we show how this can work at large scale. In doing so, we do exploit redundancy of content in popular locations—but unlike most existing solutions, we do not restrict to landmarks. In other words, we can compactly represent the visual content of all thousands of images depicting *e.g.* the Parthenon and still retrieve any single, isolated, non-landmark image like a house or a graffiti on a wall. Starting from an existing, geo-tagged dataset, we cluster images into sets of different views of the the same scene. This is a very efficient, scalable, and fully automated mining process. We then align all views in a set to one reference image and construct a 2D *scene map*. Our indexing scheme operates directly on scene maps. We evaluate our solution on a challenging one million urban image dataset and provide public access to our service through our online application, VIRaL.

1 Introduction

Images in community photo collections have scaled to billions over the last few years. Searching into such huge collections traditionally depends on text and other community generated data; state-of-the-art visual image retrieval has not yet scaled enough. On the other hand, a number of data mining and clustering approaches have emerged that exploit data like location, time, user (photographer) and tags. Such approaches typically focus on popular subsets where visual representation can indeed help, *e.g.* images containing landmarks or associated to Wikipedia[1] articles.

National Technical University of Athens
9, Iroon Polytexneiou Str., Zografou, Athens, Greece
E-mail: {ykalant,gtolias,iavr,finik,espyrou,fmylonas,stefanos}@image.ntua.gr

[1] `http://www.wikipedia.org`

What is more interesting, new applications are emerging, for instance location estimation as in Hayes and Efros [13], virtual tourism as in Snavely *et al*. [41], and landmark recognition as in Zheng *et al*. [46]. Such applications are becoming part of larger geographical systems, creating a new experience. For instance, Flickr[2] and Panoramio[3] photos can already be seen directly overlaid in panoramas of Bing Maps Streetside Photos[4] and Google Street View[5], respectively. While matching and alignment can be done automatically with certain accuracy, photos have to be already geo-tagged. History Pin[6] goes further to collect a user-generated archive of the world's historical images and stories. The process here is manual and users need to "pin" photos on Street View by themselves.

GPS-enabled devices can provide geo-tags automatically, but most photos are still uploaded without geo-tags, let alone historical material. Automating the geo-tagging process would be a leap for such applications. Unlike [13], which only estimates a geolocation probability map, we are interested in *location recognition*, that is, accurate matching to images of the same scene. This is typically possible in urban scenes, due to their unique structural details. Along with triangulation, it may also lead to exact localization as in Zhang and Kosecka [45]. Unfortunately, current approaches to location recognition either do not scale well, or focus on popular locations like landmarks.

In our recent work (Avrithis *et al*. [2]) we have shown how large image clusters of popular places may help in boosting the efficiency of retrieval, while a distortion bound can guarantee that isolated images are still retrieved as in a generic retrieval engine. This has opened the way to a scalable solution that is still able to retrieve *non-landmark* photos. In this work we present our retrieval framework, we use it to localize a new landmark or non-landmark photo, and recognize landmarks or points of interest when they do appear. We also present our online application VIRaL[7]—Visual Image Retrieval and Localization—that provides public access to all services via an integrated interface.

Given a large set of geo-tagged images, we group them by location first, constructing *geographical clusters*. The objective here is to identify images that potentially depict views of the same scene. *E.g*., two images taken $2km$ apart are unlikely to depict the same building. We then use sub-linear indexing to compute pairwise visual (dis)similarities efficiently within geographical clusters and group images depicting the same scene into *visual clusters*. Given a visual cluster, we align all images to a reference image by homography estimation and construct a 2D *scene map* by grouping together similar local features of all images of the visual cluster. We extend the entire indexing, retrieval, and spatial matching scheme to operate on scene maps rather than images. This not only provides memory savings, but also increases recall significantly.

We experiment on a challenging one million urban image dataset containing images from 22 European cities. The clustering and mining process is very efficient and entirely automatic. It took about two days on a 8-core CPU, while the baseline visual index was already available. At query time, filtering relevant scene maps takes place in the order of milliseconds, whereas verification and re-ranking according to geometry takes a

[2] http://www.flickr.com
[3] http://www.panoramio.com
[4] http://www.bing.com/toolbox/blogs/maps/archive/2010/02/11/new-bing-maps-application-streetside-photos.aspx
[5] http://google-latlong.blogspot.com/2010/06/seeing-new-sights-with-photo-overlays.html
[6] http://www.historypin.com/
[7] http://viral.image.ntua.gr

couple of seconds. Given even a single verified match in the dataset, the location of the query image is inferred and displayed on the map along with all similar images found. Finally, locations and text (title, tags) of similar and nearby images are cross-validated with relevant information in Geonames[8] entries and geo-referenced Wikipedia articles. Whenever a known landmark or point of interest appears in the photo, the relevant article is automatically linked to the photo.

2 Related Work

2.1 Location Recognition

In one of the earliest works on multiview matching in urban scenes, Johansson and Cipolla [17] estimate homographies between pairs of images and provide automatic *pose estimation*. Using edges and corners as image features, this approach and the later one by Robertson and Cipolla [35] are limited to simple geometric structures like those in building facades. Using SIFT features [26], Zhang and Kosecka [45] search directly in the descriptor space for the closest reference view in a small image database, thereby providing coarse *location recognition* in urban environments. Pose estimation follows by RANSAC using either a homography or a fundamental matrix model, whereas exact *localization* in 3D relies on triangulation using the query and two reference views.

Using MSER regions [27] and fast nearest neighbor search, Steinhoff *et al*. [43] build on the previous model to achieve pose estimation that is fast enough for real-time, *continuous positioning* on a mobile device, with accuracy comparable to GPS. Here the dataset scales to 600 reference images of an urban environment covering an area of a few city blocks. Schindler *et al*. [37] are among the first to use inverted file indexing by means of a vocabulary tree [30] for *city-scale* location recognition, scaling up to $30,000$ images covering $20km$ of streetside views.

Hayes and Efros [13] advance to *world-scale* geographic estimation by searching into a database of six million geo-tagged images downloaded from Flickr. The price to pay is that images are now represented by global features like color/texton histograms, GIST descriptors[31], *etc*. Matching accuracy is not even comparable to that of local features and the output is a *geolocation probability map*. Kalogerakis *et al*. [18] build on the previous result by exploiting the time each photo is taken, much like [10]. The output remains a probability map and anyhow this only works for *image sequences* rather than a single image query. More recently, precise location recognition approaches have emerged that can work at world scale, but all are restricted to landmarks. Some of them are examined in the following subsections.

2.2 Landmark Recognition

Kennedy *et al*. [19] are probably among the first to mine popular locations and landmarks from a large scale (10^7) Flickr dataset including metadata like tags, geo-tags and photographers. While clustering photo locations and frequent tags helps construct *tag maps* for arbitrary areas in the world, subsequent visual clustering performs rather

[8] `http://www.geonames.org`

poorly due to the global features employed. Likewise, Crandall *et al*. [10], detect geographical regions of high density corresponding to popular locations and automatically mine landmark names from tags. Relevant photos are then seen as a ground truth dataset for a learning problem. This dataset turns out quite noisy; visual features alone underperform text and in some cases are only comparable to chance. Li *et al*. [24] slightly improve performance using a multi-class SVM classifier. Seen as an *object recognition* task, this is a difficult problem with 30 million images, of which 2 million are labelled in one of 500 categories. Clearly, indexing approaches outperform this learning alternative.

On the other hand, Simon *et al*. [39] focus more on visual clustering without location data, but follow a more principled optimization approach to select a number of *canonical views* and construct a *scene summary* for browsing. Clearly, this cannot scale easily to more than 10^4 images. *Image webs* is a related idea by Heath *et al*. [14]. Parallelism is again the key in the high computational cost involved. Chum and Matas [6], extend to *web-scale* visual clustering without using location data as well, relying on hashing to detect near-duplicates. This leads to a dramatic increase in performance, under the assumption that a popular location with a large number of associated photos is likely to be discovered.

Quack *et al*. [34] divide the geographic areas of interest into overlapping square tiles; similarly to [19] and contrary to [39] and [6], they perform visual clustering inside each tile only, making the problem more tractable. On the other hand, they perform exhaustive pairwise homography estimation, probably loosing the computational advantage. Even though landmarks, objects or events are mined in an offline process, location recognition of a new image is severely limited, due to exhaustive linear search. Gammeter *et al*. [11] improve this by inverted file indexing, but the mining process is still quadratic in the number of images in each geo-cluster. There is now an inverse search by Wikipedia articles, while objects of interest are automatically detected and labelled in photos. Finally, Zheng *et al*. [46] perform a similar combination of geographic and visual clustering, as well as an inverse search by travel guide articles containing landmark names. Again there is no indexing during mining and the huge computational cost is simply handled by parallel computing.

2.3 Reconstructing 3D Scenes

Another interesting application is vision-based reconstruction and navigation of a 3D scene from a collection of widely separated views. Targeting small unordered sets of personal photo collections, Schaffalitzky and Zisserman [36] provide one of the earliest approaches. Here local features are connected in tracks and pairwise image matches are connected into a global view of the dataset. Such *structure from motion* is enhanced by Snavely *et al*. [41] to scale to datasets of 10^3 images acquired by text queries from Flickr. On top of that, scene rendering and object-based navigation are now targeting *virtual tourism* applications.

While working on datasets of similar scale, Li *et al*. [23] attempt to speed up the reconstruction process by a hierarchical approach, eventually constructing an *iconic scene graph*. Because of the use of global descriptors, the increased speed comes at a loss of accuracy. On the other hand, Snavely *et al*. [42] employ the idea of a *skeletal graph* to speed up by summarizing data. An extreme application is reconstruction of *city-scale* models by Agarwal *et al*. [1] from Flickr datasets in the order of 10^5

photos. Here, a massively parallel architecture is designed to take advantage of cloud computing.

What is interesting is that while the above applications are probably the most computationally intensive, none actually uses existing geo-tags to guide the clustering process. This is a waste not only because each clustering sub-problem would then be smaller, but also because geo-tagged photos typically depict outdoor scenes more often, compared *e.g.* to a text query for the term "`rome`". Futhermore, despite the effort spent in constructing a model, the output is not used in any way to help retrieval or location recognition of a new photo.

2.4 Sub-linear Indexing

It is evident that local feature matching may provide accurate location recognition, so scaling up largely depends on the efficiency of the employed image indexing and retrieval scheme. Using a *bag of words* representation, Sivic and Zisserman [40] show how text retrieval techniques like codebooks, inverted file indexing, and TF-IDF weighting can apply to visual search. Nister and Stewenius [30] extend to hierarchical codebooks and construct a *vocabulary tree* that is also used to assign features to visual words. Philbin *et al.* [33] show that, being more flexible, flat k-means in fact outperforms the vocabulary tree. To construct a large (1M) codebook they employ the randomized kd-tree of Silpa-Anan and Hartley [38] to assign points to cluster centers at each iteration of k-means. Moreover, they exploit local feature shape to speed up spatial re-ranking.

Chum *et al.* [9] go a step further to exploit image similarities in the dataset and boost recall by employing a number of strategies for *query expansion*. Also employed in [1], this is a form of query-time clustering. It assumes multiple different views of the same scene in the dataset, which is typical in geo-tagged datasets from Flickr. More recent advances in image indexing include the work of Jegou *et al.* [15], Perdoch *et al.* [32] and Jegou *et al.* [16], focusing on different aspects of geometric consistency, visual codebooks and memory usage respectively. Furthermore, Chum *et al.* [8] focus on small object retrieval, while Avrithis *et al.* [3] achieve sub-linear indexing of global geometry. In general, while all recent methods are very fast, there is a trade-off between indexing accuracy and memory requirements. Our choices are discussed in section 5.

3 View Clustering

As it is common in a number of recent approaches, we follow a two-layer clustering scheme according to location (latitude, longitude) and visual similarity (number of inliers arising from spatial matching). The two layers are termed *geo-clustering* and *visual clustering*, respectively. The objective of the latter is to identify photos depicting *views* of the same *scene*. The final outcome is therefore a set of *view clusters* and the overall process is termed *view clustering*. The idea of the two layers is that views of the same scene are not expected in photos taken too far apart, so geo-clustering helps reduce the computational cost of visual clustering. We use the *kernel vector quantization* (KVQ) approach of Tipping and Schölkopf [44] for clustering. We first summarize some properties of KVQ below. We then discuss our specific two-layer clustering scheme and give examples of geo-clusters and visual clusters mined from our urban photo dataset. Finally, we discuss our choices in comparison to other solutions.

3.1 Kernel Vector Quantization

Seeing KVQ as an encoding process, the maximal intra-cluster distance is the maximum level of *distortion*. KVQ guarantees an upper bound on distortion and adjusts the number of clusters accordingly. Given a metric space (X, d) and a finite dataset $D \subseteq X$, the objective is to select a subset $Q(D)$ that is as small as possible, under the constraint that all points in D are not too far away from some point in Q. "Too far" is measured by metric d and the maximal distance is specified by a given *scale* parameter $r > 0$. An optimal solution would require combinatorial optimization; in practice, we can obtain a sufficiently sparse solution by simply solving a linear programming problem and applying a subsequent pruning step. Details are given in [44] and [2].

Given a point $x \in X$, define *cluster* $C(x) = \{y \in D : d(x, y) < r\}$ as the set of all points $y \in D$ that lie within distance r from x. The *codebook* $Q(D)$ obtained by KVQ has the following properties. (i) $Q(D) \subseteq D$, that is, *codebook vectors* are points of the original dataset. Alternatively, we shall refer to such points as *cluster centers*. (ii) By construction, the *maximal distortion* is upper bounded by r, that is, $\max_{y \in C(x)} d(x, y) < r$ for all $x \in Q(D)$. (iii) The *cluster collection* $\mathcal{C}(D) = \{C(x) : x \in Q(D)\}$ is a *cover* for D, that is, $D = \bigcup_{x \in Q(D)} C(x)$. However, it is *not* a partition as $C(x) \cap C(y) \neq \emptyset$ in general for $x, y \in D$. That is, clusters are *overlapping*.

The latter property is particularly useful for geo-clustering where it is not desirable to spatially separate views of the same scene. For visual clustering, it is useful in case of gradual transitions of views that would otherwise be arbitrarily separated. Contrary *e.g.* to k-means, the number of clusters is automatically adjusted to the maximal distortion r. KVQ requires pairwise distances between all points in D; their computation is quadratic in the dataset size $|D|$.

3.2 Geo-clustering

Given a set of photos[9], we represent each photo $p \in P$ by tuple (ℓ_p, F_p), where ℓ_p is the capture location of the photo (latitude and longitude) and F_p its set of local visual features. The latter includes feature position and shape, along with visual word labels, as detailed in section 5.1. We perform geo-clustering by applying KVQ to P in metric space $(\mathcal{P}, d_g)$ with scale parameter r_g, where $\mathcal{P}$ is the set of all possible photos and metric d_g is the *great circle distance* [2]. Given a photo $p \in P$, define a *geo-cluster* as $C_g(p) = \{q \in P : d_g(p, q) < r_g\}$. That is, the set of all photos $q \in P$ that lie within geographic distance r_g from p. Similarly, given the resulting codebook $Q_g(P)$, define the *geo-cluster collection* $\mathcal{C}_g(P) = \{C_g(p) : p \in Q_g(P)\}$.

In practice, we use spatial *bucketing* by quantizing coordinates on a uniform grid and keep one sample from each bucket to perform KVQ. The grid interval is small compared to r_g so geo-clusters are largely unaffected. The computational cost is considerably reduced however, and eventually depends on spatial grid resolution rather than $|P|$. This cost is negligible compared to that of the remaining clustering steps, *e.g.* it takes a few seconds to complete geo-clustering on set of photos P, with $|P| = 10^5$ geo-tagged photos. If more speed-up is needed, one may always index coordinates *e.g.* by a kd-tree and locate spatial neighbors in logarithmic time.

[9] We shall use the terms *photo*, *image* and *view* interchangeably in the following.

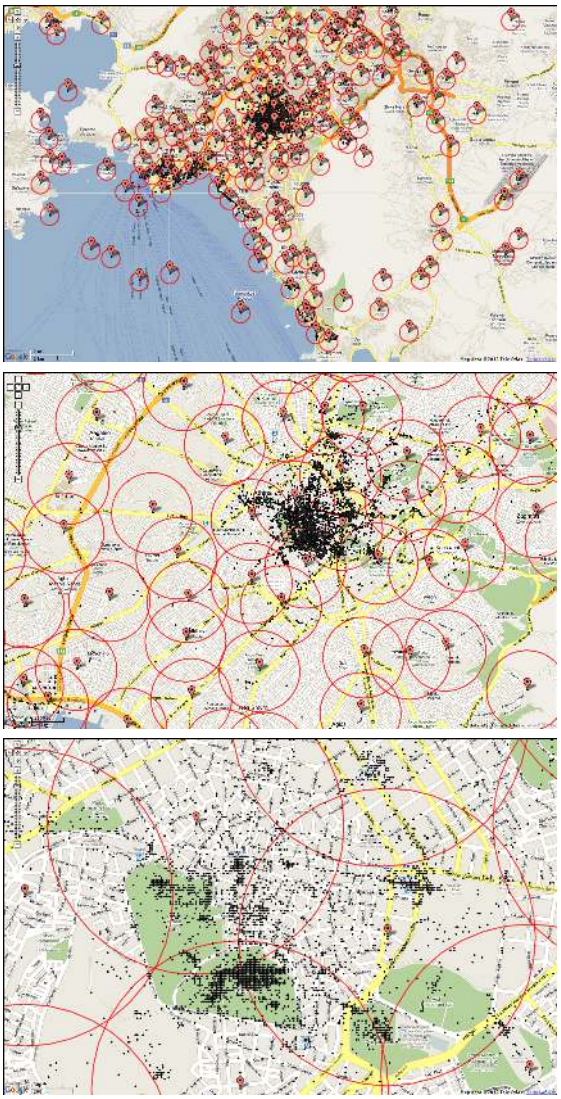


Fig. 1 Map of Athens illustrating geo-clusters at three different zoom levels. Black dots, red markers and red circles stand for photos, codebook vectors and cluster boundaries, respectively.

In Figure 1, we illustrate a map of Athens depicting all geo-clusters at three different zoom levels, for $r_g = 700m$. Observe the density of photos *e.g.* in the city center and particularly in the area of the Acropolis. Overlapping helps keep such dense areas in a single cluster for subsequent visual clustering. Photos taken even *e.g.* $1km$ away from a landmark may be included in the same cluster. The total number and position of clusters is automatically inferred from the data.

Fig. 2 Photos associated to the centers of the most populated visual clusters from Pantheon, Rome.

3.3 Visual Clustering

As in [39], we will say that any two photos $p, q \in P$ are *connected* if at least one rigid object is visible in both, possibly under different viewpoints. A *scene* is then defined as a subset $S \subseteq P$ of connected photos. That is, for all $p, q \in S$, we may visually match common objects under rigid 3D geometry regardless of viewpoint. Local visual features and descriptors are employed for this purpose, as detailed in section 5.1. The output of visual matching is typically the number of *inliers* $I(p, q)$ between visual feature sets F_p, F_q of photos p, q respectively.

We now apply KVQ to each geo-cluster $G \in \mathcal{C}_g(P)$ in space $(\mathcal{P}, d_v)$ with scale parameter r_v. Since $I(F_p, F_q)$ is a similarity measure, any decreasing function will do as a metric, *e.g.* $d_v(p, q) = \exp\{-I(F_p, F_q)\}$. The exact formula of $d_v(p, q)$ is not important; in effect, the scale parameter specifies a threshold $\tau = -\log r_v$ in the number of inliers. Let $Q_v(G)$ be the resulting codebook, and define *visual cluster* $C_v(p) = \{q \in G : d_v(p, q) < r_v\}$ for $p \in G$ and *visual cluster collection* $\mathcal{C}_v(G) = \{C_v(p) : p \in Q_v(G)\}$, similarly to geo-clustering. Repeating over all geo-clusters, the complete codebook $Q(P)$ over the entire dataset is the union $Q(P) = \bigcup_{G \in \mathcal{C}_g(P)} Q_v(G)$. Finally, the set of all *view clusters* $\mathcal{C}(P)$ is defined accordingly as $\mathcal{C}(P) = \{C_v(p) : p \in Q(P)\}$.

The main bottleneck the clustering process above is the computation of pairwise distances, which is typically quadratic in the size of the dataset. This is not an issue in geo-clustering but is critical in visual clustering. Our solution here is *geo-cluster specific* sub-linear indexing. In particular, we use an inverted file indexed by both visual word and geo-cluster. Given a query image $q \in G$, we find all matching images $p \in G$ with $I(F_p, F_q) > \tau$ in constant time that is typically less than one second. The entire computation is now linear in $|G|$.

To illustrate the effect of visual clustering on a set of photos, we give an example from Pantheon, Rome, following the examples appearing in [39] and [34]. In particular, we select all Flickr photos geo-tagged in Rome. We then separate a *seed set* of photos with tag `pantheon` and expand this set by adding all Rome photos that are visually matching any other photo in the seed set. We end up with a total of $1,146$ images that we consider to be a single geo-cluster. The resulting visual clusters are 258. The average visual cluster size is 30 images and an image belongs to 4 visual clusters on average, due to overlapping.

Figure 2 depicts photos corresponding to cluster centers for the most populated clusters. Comparing to [39], the objective here is neither summarization nor canonical

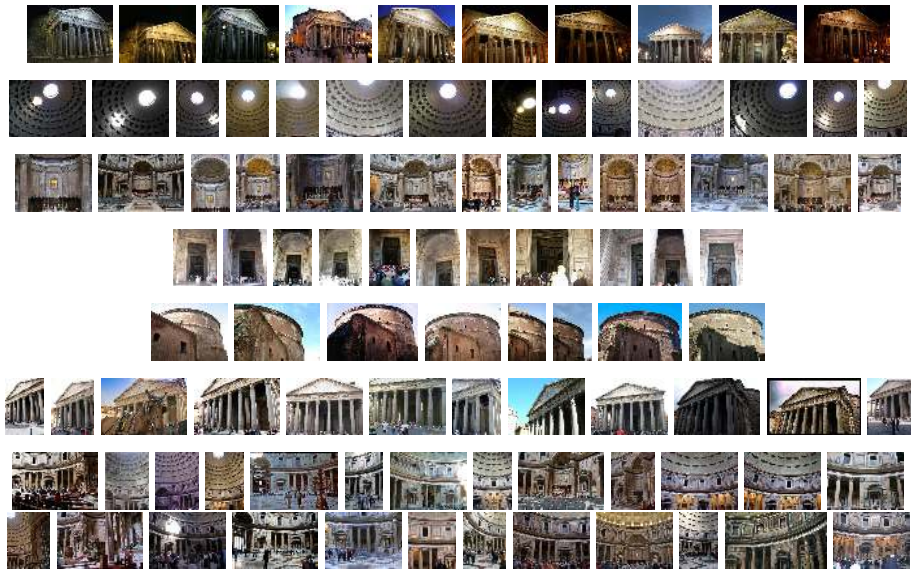


Fig. 3 Photos in a sample of visual clusters from Pantheon. The first image (on the left) of each cluster corresponds to the cluster center.

view selection, and there is no requirement for *orthogonality* between cluster centers. On the other hand, the maximal distance between photos in a single visual cluster is such that we can subsequently align all of them in a scene map. Figure 3 depicts images in a sample of visual clusters. Due to the strict matching process, images in each visual cluster are quite similar. The last cluster at the bottom appear to be diverse, but close observation reveals that all images are *connected*—that is, share a common rigid image part—with the first image in the cluster, that is the cluster center.

3.4 Discussion

Different strategies are followed for clustering in existing research works. For instance, Crandall *et al*. [10] and Li *et al*. [24] use mean-shift to perform geo-clustering alone and mine high-density locations corresponding to popular places. On the other hand, a second layer of visual clustering follows in other approaches, using different algorithms including k-means ([19]) and agglomerative clustering ([34],[11],[46]). For geo-clustering, [19] and [46] use the same algorithm as for visual clustering, whereas [34] and [11] simply quantize locations into overlapping rectangular tiles. There are also [23], [39] and [6] which perform visual clustering alone. Naturally, this does not scale well.

The main drawback of k-means and agglomerative clustering is that there is no control over the maximal intra-cluster distance. This is crucial because it may lead to geo-clusters with photos taken too far apart, or visual clusters with photos that have too few inliers. Note that k-means requires a vector space anyway, so it cannot use the number of inliers as a similarity measure. On the contrary, KVQ controls distortion and works in arbitrary metric spaces.

Mean-shift [5], used in [10] and [24], has a similar property of controlling distortion: in this case the upper bound is the bandwidth parameter of the kernel function, or the *scale of observation*. However, mean-shift needs *seeding* and *e.g.* [10] uses spatial *bucketing* and samples one photo from each bucket as a seed. There is no such need in KVQ and this is fortunate because bucketing also assumes a vector space and would not apply to visual clustering. The fixed tiles of [34] also control scale/distortion in geo-clustering, but KVQ has the advantage of adjusting to data.

A similar use of KVQ in retrieval may be found in Lampert [20]. As a branch-and-bound method, [20] relies on visual similarities within the dataset and would reduce to linear search without visual clustering. With our inverted file index on the other hand, we can still work with isolated images in sub-linear time and yet have the advantage of clustering wherever similarities permit.

Finally, the bottleneck of *pairwise distance computation* in the visual clustering process is typical in most related work. The same problem appears in Quack *et al.* [34] who use quite small spatial tiles of $200m$ because they need to perform exhaustive pairwise homography estimation within each geographic tile. This will fail to capture scenes that extend spatially to more than $200m$, which is quite often. The same quadratic cost appears *e.g.* in [11],[46],[39], while for [19] this is a reason for *not* using local features. We use larger geo-clusters with $r_g = 700m$, yet achieve a very fast implementation. This implementation is not as fast as the on of Chum [6], but we do have the advantage of geo-clustering. This lowers the cost and allows one query per image in each geo-cluster. On the other hand, [6] employs hashing with low recall, and is thus limited to popular locations—isolated photos are unlikely to be discovered.

4 Scene Maps

So far, we know that the image associated to the center of a view cluster shares at least one rigid object with all other images in the cluster. We treat it as a *reference* image for the cluster and align to it all other images by computing a relative homography transformation, as detailed in section 5. We collect all aligned visual features and construct a compact representation that we call a *scene map*, because it is a 2D spatial map of features associated to different views of the same scene. It is now possible to match a query image to an entire scene map under the same geometry. We thus use scene maps directly for retrieval, instead of images. This saves on memory and computations at query time, makes matching more robust by increasing inliers and also increases recall, because for each matched scene map we return all its views. We present scene map construction here, and then discuss our model in relation to existing work.

4.1 Scene Map Construction

For each reference image $p \in Q(P)$ and corresponding view cluster $C_v(p)$ we construct a feature collection $F(p)$ as the union of features over all images $q \in C_v(p)$, after aligning with the reference. In particular, let H_{qp} be the estimated homography from q to p and assume each visual feature is represented by a tuple (x, w) with x being the

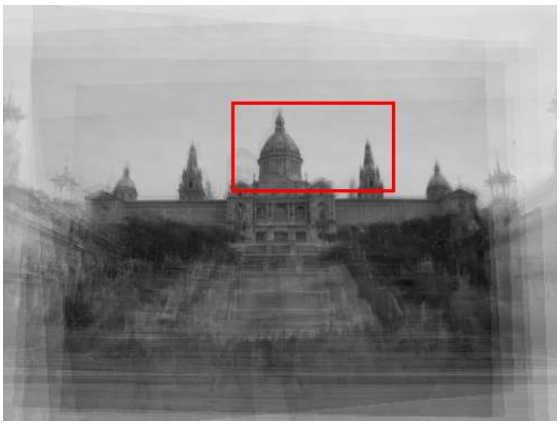
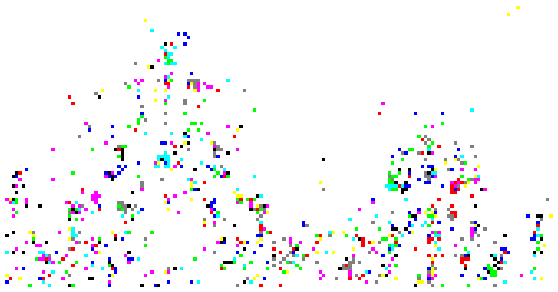
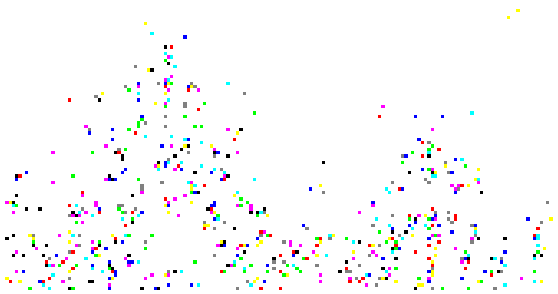


Fig. 4 Scene map construction from 10 photos of Palau Nacional, Montjuic, Barcelona.



(a)



(b)

Fig. 5 Detail of point cloud in Montjuic scene map corresponding to the highlighted region of Figure 4, (a) *before* and (b) *after* vector quantization. Colors represent different visual words, modulo 9.

position and w the visual word label. Then this collection is constructed as

$$F(p) = \bigcup_{q \in C_v(p)} \{(H_{qp}x, w) : (x, w) \in F_q\}. \quad (1)$$

Here, x is assumed a 3-vector with the homogeneous coordinates of feature position. The *scene map* $S(p)$ is a sparse representation of $F(p)$ such that a query will (ideally) match a scene map whenever it would match any single image in the map. This gives rise to vector quantization once more. As detailed in our previous work [2], it turns out that we can break this up into a number of smaller subproblems. In particular, we partition $F(p)$ into a number of disjoint sets $F_w(p) = \{(x, u) \in F(p) : u = w\}$, one for each a visual word w, and apply KVQ separately to each $F_w(p)$ in $(\mathbb{R}^2, d_2)$ with scale parameter r_x. Here feature positions are in 2D Euclidean coordinates and d_2 is the Euclidean metric. Finally, we join the resulting codebooks $Q_x(F_w(p))$ into a single scene map, $S(p) = \bigcup_{w \in \mathcal{W}} Q_x(F_w(p))$. We set scale parameter to $r_x = \theta$, where θ is the error threshold used in spatial matching. Hence, a feature f will be in the spatial cluster $C_x(f')$ of another feature f' whenever f, f' are inliers in spatial matching.

In order to provide an example of scene map construction, we use a visual cluster containing 30 images of Palau Nacional, Montjuic, Barcelona, 10 of which are overlaid in Figure 4, after alignment. Out of $11,623$ features in total, $9,924$ are retained in the scene map after quantization, giving a compression rate of 15%. In terms of inverted file entries (unique visual words), the figures are $11,165$, $8,616$, and 23%, respectively. Detail of this scene map's point cloud is shown in Figure 5. It is evident that features are sparser *after* vector quantization.

4.2 Discussion

The above formulation bears similarities with several models in different contexts. To name a few, Lowe [25] performs local feature *view clustering* by linking similar features that are matched in adjacent views of an object, applying this representation to 3D object recognition. Simon *et al*. [39] organize matching features of multiple images into *tracks*, where a track corresponds to a single 3D point of a scene. They use this representation to produce a visual summary of the scene by means of a set of *canonical views*. Gammeter *et al*. [11] perform a similar alignment in visual clusters with the objective of isolating bounding boxes of depicted landmarks. In image retrieval, Chum *et al*. [9] collect the verified images from a query and build a *latent model* of the scene by averaging term frequency vectors. This model is used on the query side to perform *query expansion*. Leibe *et al*. [21] construct a set of *spatial occurrence distributions* in an implicit shape model for object detection.

Comparing our model to [9], the latter does not encode feature position and is constructed dynamically on the query side, whereas scene maps reside on the database side and are static. Unlike the *object-based* approach of [11] we want to keep information from all image regions. Matching features are *linked* into connected components in [25], [39], and we need a similar compact representation, that is, more compact than storing features of individual views. However, we also need to control the size of such components, so that components in a scene map behave like features in a single image. One way is to keep a minimal subset $S(p) \subseteq F(p)$ such that no feature in $F(p)$ is too distant from its nearest neighbor in $S(p)$. This justifies our use of KVQ in this case as well.

5 Visual Matching and Indexing

All processes used to match, align and index images or scene maps are described in this section. These include: (i) the *baseline* visual representation, similarity, indexing and spatial matching process. This is also a stand-alone retrieval solution that can be used for location recognition as well. It forms the baseline for comparisons in our experiments in section 8 and in fact, it is the underlying process of the current implementation of VIRaL application. (ii) The *geo-cluster specific* indexing process, used for distance computations during visual clustering (section 3.3). (iii) The visual *alignment* process that is employed during scene map construction (section 4). (iv) The *scene map* similarity, indexing and spatial matching process. This is an extension of the baseline, and can handle matching between images or scene maps.

5.1 Baseline

In the baseline process all images are treated individually. Images are represented by local visual features and descriptors, which are quantized up to visual word against a visual codebook. More details on features and visual codebook are given in section 8. We construct a bag of words representation and measure similarity by histogram intersection and TF-IDF weighting. We then index images in an inverted file structure, so query time is sub-linear in the number of images in the dataset. Ranking is based on appearance only, not taking into account the spatial layout of local features.

A top-ranking shortlist of images is subsequently checked for geometric consistency with the query image to verify there is common rigid object, or two views of the same scene. We use a variant of fast spatial matching [33] over a 4-DOF similarity model. This model makes a *single correspondence* assumption. In particular, tentative correspondences between features of the query and an image in the list are generated by matching visual words. Given a single correspondence, we use the position, scale and orientation of the two features to compute similarity transformations T_1, T_2 that map the features to a unit circle centered at the origin. Under no gravity-vector assumption, an initial transformation hypothesis is $T_2^{-1}T_1$. We count inliers over the tentative correspondences and iterate over hypotheses. Whenever a new maximum is found, we compute a least squares estimate of an affine transform from the given inliers and store the best model so far—this corresponds to the "simple" method of Locally Optimized RANSAC (LO-RANSAC) [7]. We have found that images with at least $\tau = 10$ inliers with the query typically depict the same object or scene.

Geo-cluster specific indexing. This is a simple variation of the baseline process, where the inverted file is indexed by both visual word and geo-cluster. It is used during visual clustering where, making a query for each image in a geo-cluster, we collect all verified images giving $I(p, q) > \tau$ inliers. Because a geo-cluster is quite small compared to the entire dataset, querying the index is significantly faster. Irrelevant images are fewer so the top-ranking list can be shorter and spatial matching is faster as well. Typically the query time is constant and on average an order of magnitude faster than the baseline. It is also independent of the size of dataset.

5.2 Alignment

In order to construct a scene map from the images in a view cluster, we need to align their corresponding features first. We do this by estimating homography models between matching images. Alignment is performed between a single reference image and all other images in a view cluster; that is, it is *linear* in the size of the cluster. Initial estimates are readily available from the responses of each query: for each pair of matching images (p, q) in a geo-cluster, we store the best affine model A_{qp} that transforms q to p. Therefore, when view clustering is complete, we only need a final step of local optimization to estimate the homography.

More specifically, given a reference image p in view cluster $C_v(p)$, we align p to each image $q \in C_v(p)$. We start from the stored affine model A_{qp} and perform a single step of the "iterative" method of LO-RANSAC. The complete set of all points with error smaller than threshold $K\theta$ are used to estimate a homography with the Direct Linear Transformation (DLT) algorithm [12]. We reduce the threshold and iterate until it is equal to θ. We have found a maximum of 3 iterations to be enough for our experiments. The final homography that aligns q to p is stored as H_{qp}.

5.3 Scene Map Indexing

Once all scene maps have been computed, we build a separate index for them. Even if a scene map is typically larger than a single image, it has exactly the same representation, that is, a set of features. We therefore treat scene maps as images for indexing and retrieval. By construction, we have already subsets $Q_x(F_w(p)$ of scene map $S(p)$ corresponding to each visual word w. The cardinalities of these subsets give directly a term frequency vector for $S(p)$. We then index all scene maps by visual word in an inverted file. At query time, we compute a similar vector for the query image, and retrieve relevant scene maps by histogram intersection and TDF-IF weighting.

A shortlist of top-ranking scene maps is again verified using the single correspondence assumption, as in the baseline process. Even if the initial estimate is a similarity transformation originating from the position, scale and orientation of local features, we can still recover a correct affine transform by least squares fitting given at least three inliers. To speed up the re-ranking process, we terminate and consider the scene map verified if at least τ_h inliers have been found. On the other hand, we discard an image if no more than τ_ℓ inliers have been found for a predefined percentage of all hypotheses. Moreover, we discard a hypothesis if the the inliers found for a predefined percentage of correspondences are fewer than τ_ℓ.

Whenever a scene map $S(p)$ is found relevant, all images $q \in C_v(p)$ are considered relevant as well. This is exactly how recall is increased. To avoid the additional cost of individual matching with each image, we consider all of them at the same rank, which slightly affects precision.

5.4 Discussion

Our baseline visual indexing scheme is closest to [33], because it strikes a very good balance between recall performance and memory footprint. On the other hand, our scene map indexing is is beneficial in terms of both. It is analogous to the latent model

for query expansion of [9], but executing offline on the database side. To draw an analogy, recall that scene maps are statically computed in the off-line indexing process and constrained within geo-clusters. On the other hand, a model is built dynamically at query time in [9], increasing the computational cost. Without any constraint it is prone to drift, especially when iterative. Most importantly, query expansion cannot help at all when relevant images are too few (or just one) and initial query fails. We compare our approach against two query expansion schemes in section 8.

6 Location and Landmark Recognition

Since retrieved images are likely to depict the same scene with the query photo, they are also likely to be taken at a nearby location. Also, whenever any of the retrieved images is associated to a known landmark or point of interest, we may infer a similar association for the query photo as well. We explore these ideas below to provide automated location and landmark recognition, respectively. We then discuss our choices in relation to existing solutions.

6.1 Location Recognition

Once a list of verified images or scene maps is retrieved, we exploit their geo-tags to recognize the location where the query photo is taken. Of course, geo-tags of the images in the dataset have different levels of accuracy, and some may be completely wrong. We make here the assumption that even in the presence of outliers, there is a subset of photos that are correctly geo-tagged, and these geo-tags are not too far apart. Hence, we apply agglomerative clustering to the retrieved image locations and terminate when the minimum inter-cluster distance is above a certain threshold. If there is at least one cluster that contains more locations (photos) than all the others, then the centroid of these locations is provided as the estimate of the query photo location. Otherwise, one cluster is chosen at random.

We employ the *reciprocal nearest neighbor* (RNN) [21] algorithm for clustering, using the group average criterion and Euclidean distance—more precise geographical distance is not necessary here because locations are assumed nearby. Typically, we set the termination threshold to $200m$ to represent the area around a building, landmark, or depicted scene in general. The choice of an agglomerative approach is appropriate here because it allows the extent of clusters to adjust to how retrieved locations are spread around depicted scenes, yet it does not allow two clusters to merge if they are too distant. The number of clusters is inferred from the data, while computation is fast enough to apply at query time. Choosing the most populated cluster makes our estimate robust. Outliers, either due to wrong geo-tags or errors in visual matching, are discarded and do not affect location recognition.

6.2 Frequent Tags

Recognition of landmarks or points of interest relies on existing user tags and photo titles [10]. Titles are typically more reliable, but tags can also be helpful, despite being

[10] Photo titles and user tags are the ones provided by users at the Flickr website.

rather noisy. To provide for robustness and efficiency, we represent terms using a *codebook* and extract a set of frequent tags using this representation. We first filter all tags of the entire dataset through a manually created *stoplist* containing terms that are too generic (*e.g.* `paris`, `france`, `holidays`), describe the conditions of the photo-shoot (*e.g.* `night shot`, `black and white`), or are typically irrelevant to the content of the photos (*e.g.* `nikon`, `geo-tagged`).

We then construct the codebook in an offline clustering process that is initialized using data from the Wikipedia Search[11] web service of Geonames. For each city in the dataset, we have collected all entries in the geographical bounding box of the city center, as specified in section 8.1. Each entry corresponds to one landmark or point of interest, so we create one cluster for each. To deal with typos or language diversity, we compare strings using the Levenshtein distance [22]. Starting with a list of all tags in the dataset, we iteratively pick one tag at random, remove it from the list, and compare it to the current set of clusters. If it is within a specific distance T from some cluster, we insert it in that cluster (choosing one cluster at random in case of multiple candidates); otherwise, we create a new cluster represented by this tag. We repeat until the list is empty.

This process is similar to *canopy clustering* [28]; however, we use a single threshold and a specific initial set of clusters. Custom initialization is the reason we do not use KVQ, which would otherwise fit to this problem as well. All tags are assigned to a unique cluster at maximum distance T, while tags associated with known landmarks are represented by their Geonames form, which is considered canonical. Now, given the codebook, we assign each tag to a single codeword. As an offline process, we associate each photo to the codewords of its tags. Then, at query time, we collect all codewords of retrieved photos and keep the ones having at least two occurrences into a set of *frequent tags*. No string comparison is required for this process.

6.3 Landmark Recognition

We consider a landmark or point of interest to be any item associated to a Wikipedia article that is geo-referenced within the geographical bounding boxes of the dataset cities. To construct a list of such items, we use the Geonames source mentioned above, as well as the corresponding Wikipedia web service[12]. The two services are quite similar and typically 90% of entries are identical in the sense that they share the same Wikipedia article URL. There are differences however, so we have merged them into a single, combined list. For each item, we have stored the article name, url and geographical coordinates.

Now, given a query photo and its estimated location we select a list of articles located within specific distance from the photo location—typically $200m$, as in location recognition. Each article title is matched against all frequent tags, as well as all titles of retrieved photos. The Levenshtein distance is used once more, and each article is assigned the minimum distance found. We rank articles by ascending distance and select the top ranking ones below distance T as the set of *suggested tags*. These tags identify the landmarks found and the associated Wikipedia articles are automatically linked in the VIRaL result page. We could of course follow a similar codebook approach

[11] `http://www.geonames.org/export/wikipedia-webservice.html#wikipediaSearch`

[12] `http://de.wikipedia.org/wiki/Wikipedia:WikiProjekt_Georeferenzierung/Wikipedia-World/en`

to avoid string matching for titles but in practice the computational cost is negligible compared to visual search.

6.4 Discussion

Different approaches are followed in related work for tag processing and landmark recognition. A learning framework is followed in [10], [24], where geo-clustering is only used to construct a ground truth set. A classifier is trained on this set using text, visual features, or both. This approach is mostly limited by the dataset being too noisy, and recognition has scaled up to 500 landmarks, while no location recognition is supported. By contrast, our solution supports approximately $8,500$ landmarks or points of interest in the current dataset of 23 cities. It is also interesting that [10], [24] only use tags while we have observed that photo titles are typically more reliable.

Quack *et al*. [34] follow a quite exhaustive offline process: for each visual cluster they extract tags, query Google for related Wikipedia articles, download photos from several such articles, and then match the two photo sets to verify. However, photos within articles are not quite reliable and the mining process is too slow. By using article location and matching to estimated query location, we can narrow down text search so that verification is now achieved at query time. Article location is used during crawling in [11], which is quite similar to our approach in this sense, although assignment of articles to visual clusters is again offline.

Zheng *et al*. [46] also assign visual clusters to landmarks in an offline process, and support approximately $5,500$ landmarks from $1,300$ cities in 144 different countries. They search into a small subset of representative images in each cluster. By contrast, we use scene maps to search efficiently into the entire dataset, can recognize *any* point of interest within the supported areas and localize *any* photo, landmark or not.

7 Application: VIRaL

The proposed methods may be accessed through our online application, VIRaL. We use a dataset of 1.1M Flickr images depicting content from 23 European cities, along with their metadata (i.e. geographic location, user tags, image title and description). We have crawled this dataset from Flickr by requesting only geo-tagged photos and constraining the search with a bounding box around each city's center. A subset of this dataset is used for our experiments as described in detail in section 8.1.

The response to a visual query is a ranked list of visually similar images. The integrated process for visual retrieval follows the baseline approach described in subsection 5.1. A visualization of the detailed matching between query and each similar image is also possible. The query image gets localized on the map and associated with a set of frequent tags and a set of suggested tags as well (see section 6). Suggested tags come along with direct links to Wikipedia articles.

7.1 Walkthrough

The welcome screen of the online application (Figure 6) presents a random set of dataset images. There are two ways to browse the image dataset: through the welcome

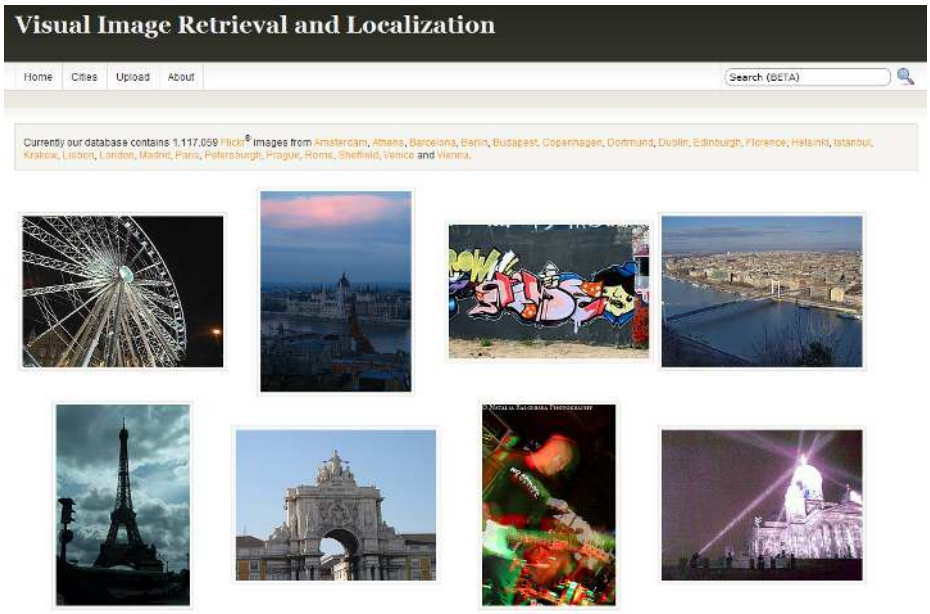


Fig. 6 VIRaL home page. Images shown are randomly selected from the entire dataset.

screen and through the *Cities* page. The latter presents random images from each city. After choosing a city name, VIRaL will only fetch random images from the selected city. Clicking on any of the images will trigger a new visual query.

Figure 7 presents a result page. In this case, the query is the second image in the second row of the welcome screen example of Figure 6. On the top left corner of the result page, we can see a map that contains a blue marker for each similar image and a red marker for the estimated location of the query image. The gray marker at the rightmost part of the map corresponds to a visually similar but incorrectly geo-tagged image, which does not participate in the location estimation and is thus treated as an outlier. On the top right of the result page we can see the query image along with the sets of frequent and suggested tags. At the bottom rows, VIRaL presents the retrieved images, ranked by decreasing similarity. The similarity value shown is the number of inliers, normalized in $[0, 1]$ with the use of a sigmoid function. Still referring to the example of Figure 7, the frequent tags are `terreiro do paço, praça do municipio, monument, stevie0020, arch`. The final set of suggested tags is `Praça do Comércio` and `Lisboa`. Both are automatically linked to Wikipedia and are valid suggestions as shown in Figure 8.

An uploaded image or image URL can also be used as a visual query. Of course, in order to get proper results, the query image must have been taken at one of the cities included in the VIRaL dataset. We have tuned the VIRaL application for high precision, in order to eliminate as many false positives as possible. To boost recall also, we have integrated a query expansion method, which is referred to as QE1 in section 8, and it produces the set of *Similar of Similar* images. This set is constructed in negligible query time, since similar images for the complete dataset have been computed off-line. Figure 9 depicts this set for the example query of Figure 7.

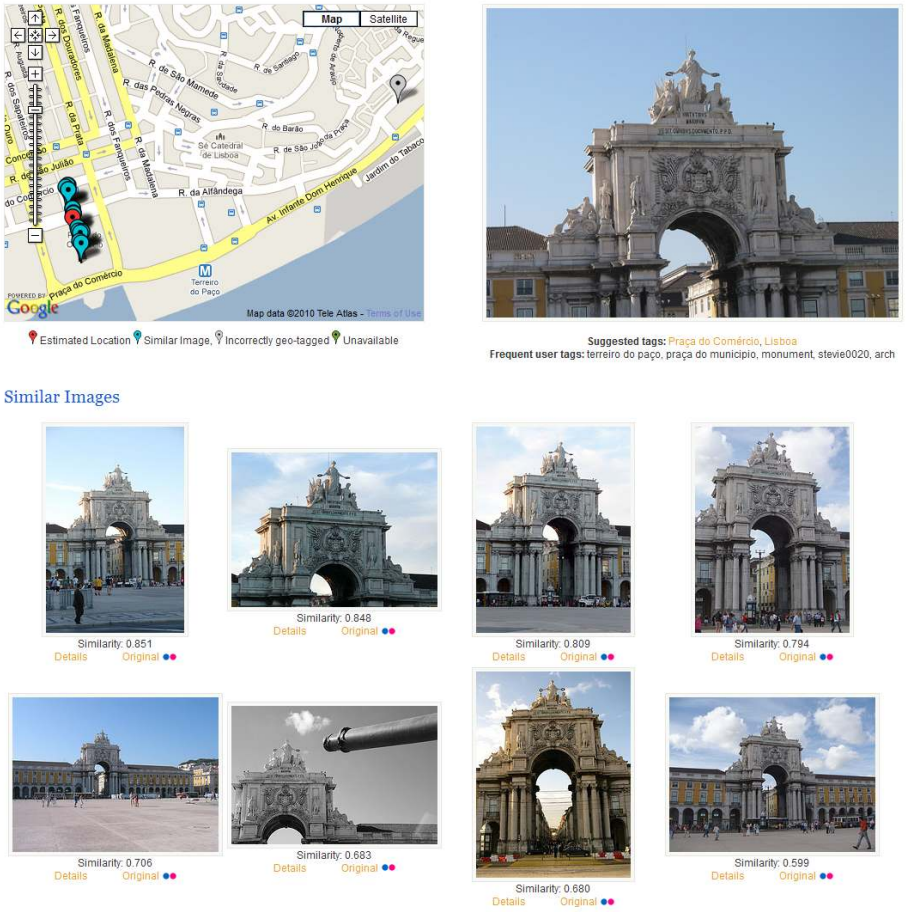


Fig. 7 Results of successful visual matching, location and landmark recognition. Top left: map depicting actual locations of the similar images (blue markers) and the estimated location of the query image (red marker). Top right: query image along with the sets of frequent and suggested tags. Bottom: visually similar images.

Along with the view of similar images, it is also possible to visualize the result of spatial matching by clicking on the *Details* link, positioned under each retrieved image. As depicted in Figure 10, the matching part of the two images that contains features in correspondence is inside bounding boxes.

Figure 11 illustrates the case of a typical scene of a building in Amsterdam that is correctly localized based on three similar images but does not correspond to any known landmark or point of interest. The two suggested tags are `Sint Antoniesbreestraat`, the name of the street, and `Zwanenburgwal`, the name of the canal.

Fig. 8 Wikipedia article suggested for the query image.

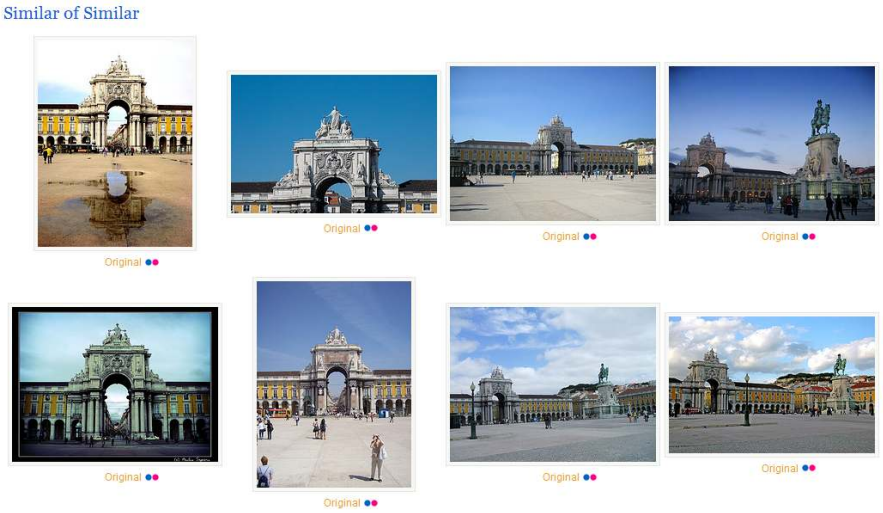


Fig. 9 Similar of Similar results, for the example query of Figure 7.

8 Experiments

8.1 Dataset

We experiment on a challenging one million urban image dataset, namely *European Cities 1M*[13]. It consists of a total of 1,037,574 geo-tagged images from 22 European cities, which is a subset of the dataset used in the VIRaL application. A subset of 1,081 images from Barcelona are annotated into 35 groups depicting the same scene, building or landmark. Well known landmarks of Barcelona are depicted in the 17 groups, while the rest 18 depict scenes or buildings around the city center. Samples of

[13] We have published the dataset online at `http://image.ntua.gr/iva/datasets/ec1m/`.

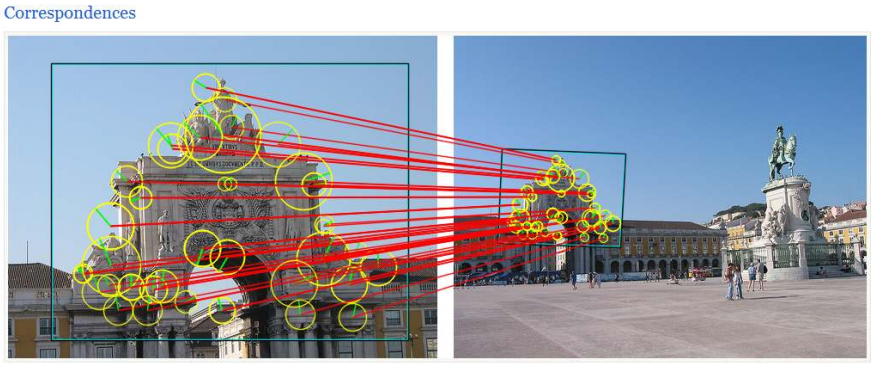


Fig. 10 Correspondences between the query image (left) and a similar image (right). Local features identified as inliers are depicted in yellow circles with scale and rotation (green line). Correspondences between inliers are drawn in red lines. The blue bounding box indicates the common region of the two images.

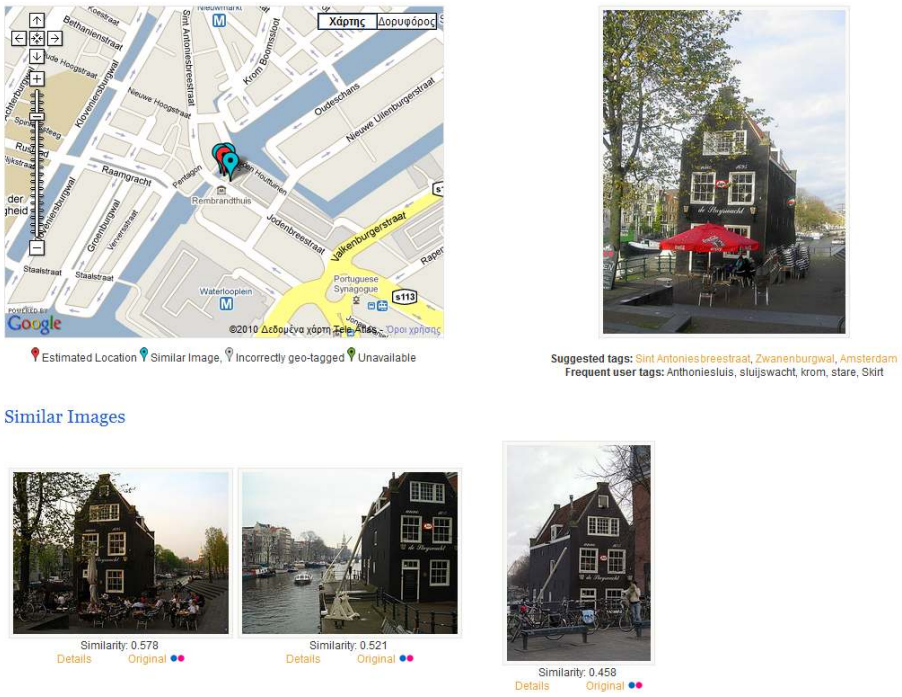


Fig. 11 Localization and recognition result for an indicative *non-landmark* query image.

the annotated set of images are presented in Figures 12 and 13, depicting landmarks and non-landmarks respectively. We will therefore refer to non-landmarks as scenes. Since only a subset of the annotated images are landmarks, annotation cannot rely on tags; it is rather a combination of visual query expansion and manual clean-up. We have also assigned geographic information and relative Wikipedia article(s), whenever

Fig. 12 Query images from the 17 groups of landmarks in the annotated dataset.

applicable, to each of the groups. This way the ground truth can be further used for geo-estimation and landmark recognition evaluation. Five images are selected as queries from each group. If the group contains less than 5 images, a frequent case for non-landmark scenes, all of the group images are used as a query. In total, we used 157 queries. Table 1 presents the names of the landmarks selected for the evaluation and the size of the corresponding group for each of them and for all the scenes contained in the annotated dataset.

Our 1M dataset contains 128,715 Barcelona images from Flickr. Since the 1,081 annotated images are a subset of these, we have removed the rest Barcelona photos, in order to be sure that no other image in the evaluation dataset depicts the same scene/building as the ground truth. The remaining 908,859 images are the distractors. Most of them depict urban scenery like the ground-truth, making a challenging distractor dataset.

8.2 Evaluation protocol

For all experiments, we used the medium Flickr image size, which is 500×500 pixels maximum. We extracted SURF features and descriptors [4] and kept a maximum of 1,000 features per image. We built a generic 75K visual codebook from features of urban scene images, that are not a part of the evaluation dataset. Larger codebooks did not perform well in scene map construction. To construct the vocabulary, we used approximate k-means [33], where nearest cluster centers at each iteration have been assigned using randomized kd-trees [38]. Specifically, we used the FLANN library of Muja and Lowe [29] both in vocabulary creation and to assign visual words to image

Fig. 13 Query images from the 18 groups of non-landmarks in the annotated dataset.

Landmark	Group size	Non-landmark	Group size
La Pedrera(a)	129	Scene1	5
Park Guell(a)	50	Scene2	3
Museu Nat. d' Art	17	Scene3	22
Columbus Monument	18	Scene4	2
Carrer B.I.-El Gotic	36	Scene5	30
Port Vell	18	Scene6	5
Sagrada Familia	29	Scene7	4
Casa Batllo	16	Scene8	3
Arc de Triomf	20	Scene9	17
La Pedrera(b)	71	Scene10	14
Hotel Arts	106	Scene11	22
Hosp. de San Pau(a)	116	Scene12	7
Hosp. de San Pau(b)	73	Scene13	4
Park Guell(b)	17	Scene14	2
Torre Agbar	93	Scene15	2
Placa de Catalunya	48	Scene16	5
Cathedral (side)	70	Scene17	4
		Scene18	3

Table 1 Ground truth group size for each landmark (17 items) and non-landmark (18 items) of the annotated dataset.

features. Our bag of words implementation uses histogram intersection similarity on L_1-normalized vectors and TF-IDF weighting. Details on indexing and spatial matching during visual clustering and scene map construction were presented in sections 3 and 4 respectively. We evaluate overall retrieval performance by measuring mean Average Precision (mAP).

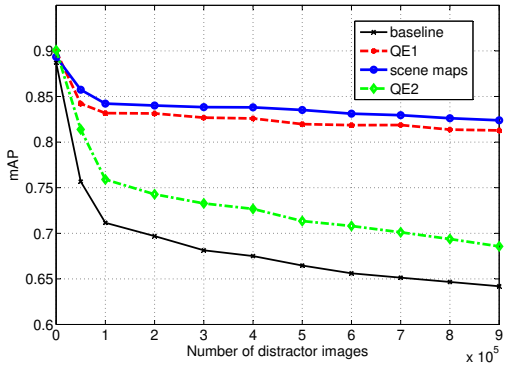


Fig. 14 Mean Average Precision measurements for the four methods on the *European Cities 1M* dataset under a varying number of distractors.

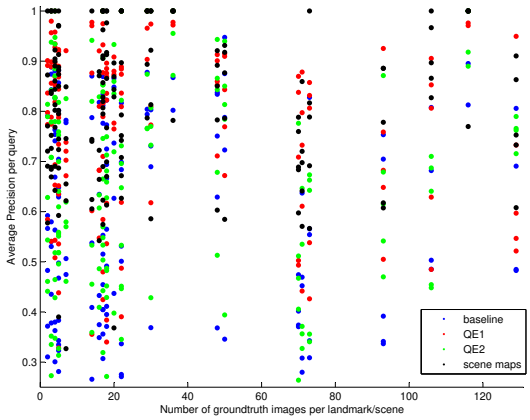


Fig. 15 Average Precision for each query vs. size of the corresponding ground truth group.

8.3 Results

The mining process that leads to retrieval with scene maps is entirely automated. Geo-clustering on the European Cities 1M dataset takes less than 5 minutes and generates $1,677$ geo-clusters. Visual clustering creates $493,693$ visual clusters. Clustering takes approximately 22 minutes; however, all queries required to compute visual dissimilarity matrices take approximately 52 hours, clearly being the most time consuming process. Construction of all scene maps takes another 5 hours. It is noteworthy that $351,391$ visual clusters are single images, hence do not need scene map construction. Given larger datasets with more cities, the above times would increase linearly, while of course computation can be made parallel. The inverted index of the new retrieval engine requires 1.20GB of memory instead of 1.61GB for the baseline, providing a compression of 25%. It is worth mentioning that all experiments are performed with our own C++ implementation on a 2GHz Quad Core processor with 8GB of memory. The total

Method	Avg. query time	mAP
Baseline BoW	1.03s	0.642
QE1	20.30s	0.813
QE2	2.51s	0.686
Scene maps	1.29s	0.824

Table 2 Average query time and mean Average Precision (mAP) of the four methods on the *European Cities 1M* dataset including all distractors.

number of tags in the entire dataset is $7,764,264$ and the codebook contains $188,989$ and $181,752$ terms, before and after using a stoplist, respectively. A total number of $2,396,926$ tags corresponds to the terms removed by the stoplist.

Visual similarity evaluation. To evaluate the performance of the proposed method in terms of visual image retrieval, we compute mean average precision (mAP) on the aforementioned *European Cities 1M* dataset. We compare our scene map retrieval efficiency against a baseline bag of words and two *query expansion* methods. The first (QE1) is the naive iterative approach, where we re-query using the retrieved images and then merge the results. In our experiments, this expansion was carried out 3 times iteratively for each query. For the second (QE2) we create a scene map using the initial query's result and re-query once more. All methods use the same spatial re-ranking approach as described in section 4. The mAP measurements on the 157 ground truth queries for all four methods under varying size of distractor set are depicted in Figure 14. Observe that our method using scene maps (SM) outperforms all other methods in terms of mean average precision.

As shown in Table 2, our method does not differ much from the baseline method in terms of speed, which is clearly the fastest. The proposed method offers slightly faster filtering of the inverted index because there are less scene maps than images, however it requires slightly more time to re-rank, because scene maps have more features compared to images. In general, filtering time only depends on the number of relevant scene maps, while re-ranking time is constant. So query time is not a bottleneck when going to larger scale. It is noteworthy that both query expansion methods require far more time while yielding worse results. QE2 query corresponds roughly to two baseline queries and a scene map construction, and QE1 to several baseline queries, resulting to quite impractical query times.

The annotated dataset used contains variable sized groups of images depicting the same scene. Small ones usually correspond to non-landmark scenes while large ones to well known landmarks. Achieving high recall scores is challenging when we deal with a large group of similar images. Re-ranking is only performed on the top ranked images and this can lead to missing quite a few images with the baseline method. Figure 15 shows mAP values for each query, against the size of the corresponding annotated group. Observe that scene maps can yield total recall even for scenes containing more than 100 images. For the same scenes, the otherwise powerful QE1 fails to retrieve all the scene instances, since some images were lost from the initial query before the expansion. Furthermore, almost total recall is observed in the small clusters for scene maps, the images of which are usually contained in a very small number of scene maps, usually one or two.

Figures 16 and 17 show a query image, of a non-landmark and a landmark respectively and top ranked retrieved and geometrically verified images. Geometrically

Fig. 16 Sample queries and ranked geometrically verified images for a non-landmark image with the four methods. Query image is on the left and retrieved images from the database on the right. Each row correspond to one of the evaluated methods.

Fig. 17 Sample queries and ranked geometrically verified images for a landmark image with the four methods. Query image is on the left and retrieved images from the database on the right. Each row correspond to one of the evaluated methods.

verified images are more for scene maps leading to higher recall. Tables 3 and 4 contain mAP values for each group of the landmarks and non-landmarks ground truth respectively. Remarkable is the fact that for many groups scene map achieved perfect mAP equal to 1.0 while other methods achieved a worse ranking of the similar images.

Location recognition evaluation. All *European Cities 1M* dataset images are geo-tagged. Thus, given the outcome of visual retrieval, location recognition is performed as described in section 6. To evaluate the proposed scheme, we compare the

Landmark	Method			
	Baseline	QE1	QE2	Scene maps
La Pedrera(a)	0.326	0.588	0.377	0.901
Park Guell(a)	0.795	0.794	0.812	0.847
Museu Nat. d' Art	0.590	0.702	0.602	0.637
Columbus Monument	0.505	0.658	0.558	0.698
Carrer B.I.-El Gotic	0.449	0.917	0.555	0.739
Port Vell	0.332	0.746	0.380	0.480
Sagrada Familia	0.857	0.889	0.864	0.881
Casa Batllo	0.759	0.792	0.767	0.798
Arc de Triomf	0.840	0.889	0.847	0.882
La Pedrera(b)	0.651	0.921	0.939	0.903
Hotel Arts	0.560	0.773	0.573	0.633
Hosp. de San Pau(a)	0.317	0.580	0.423	0.838
Hosp. de San Pau(b)	0.421	0.776	0.502	0.709
Park Guell(b)	0.500	0.886	0.526	0.634
Torre Agbar	0.310	0.617	0.378	0.630
Placa de Catalunya	0.794	0.853	0.798	0.812
Cathedral (side)	0.487	0.864	0.546	0.972

Table 3 Mean Average Precision per landmark for the four methods. For each landmark 5 query images were used.

Scene	Method			
	Baseline	QE1	QE2	Scene maps
Scene1	0.618	0.648	0.654	0.884
Scene2	0.667	0.847	0.730	1.000
Scene3	0.399	0.458	0.451	0.880
Scene4	1.000	1.000	1.000	1.000
Scene5	1.000	1.000	1.000	1.000
Scene6	0.800	0.969	0.848	0.802
Scene7	0.876	0.979	0.940	1.000
Scene8	1.000	1.000	1.000	1.000
Scene9	0.339	0.557	0.357	0.754
Scene10	0.351	0.482	0.428	0.687
Scene11	0.557	0.843	0.575	0.633
Scene12	0.577	0.857	0.639	0.755
Scene13	0.681	0.846	0.746	1.000
Scene14	0.875	1.000	0.880	0.885
Scene15	1.000	1.000	1.000	1.000
Scene16	0.791	0.883	0.798	0.812
Scene17	1.000	1.000	1.000	1.000
Scene18	0.800	0.972	0.810	1.000

Table 4 Mean Average Precision per scene for the four methods. For each scene 5 query images were used (less if the total group size is below 5).

resulting estimation against the hand-picked geographic location information of each annotated group of images in our *European Cities 1M*. Localization accuracy in comparison to baseline and other methods is shown in Table 5. As we see, localization percentage is already high even for the baseline method. Still, our method using scene maps reaches the highest percentage.

Samples of query images depicting well known landmarks and the corresponding localization result on the map are presented in Figure 18. The first 6 cases achieve successful recognition. However in the last two cases we present two examples, coming from the evaluation queries, of unsuccessful recognition based on the ground truth geo-

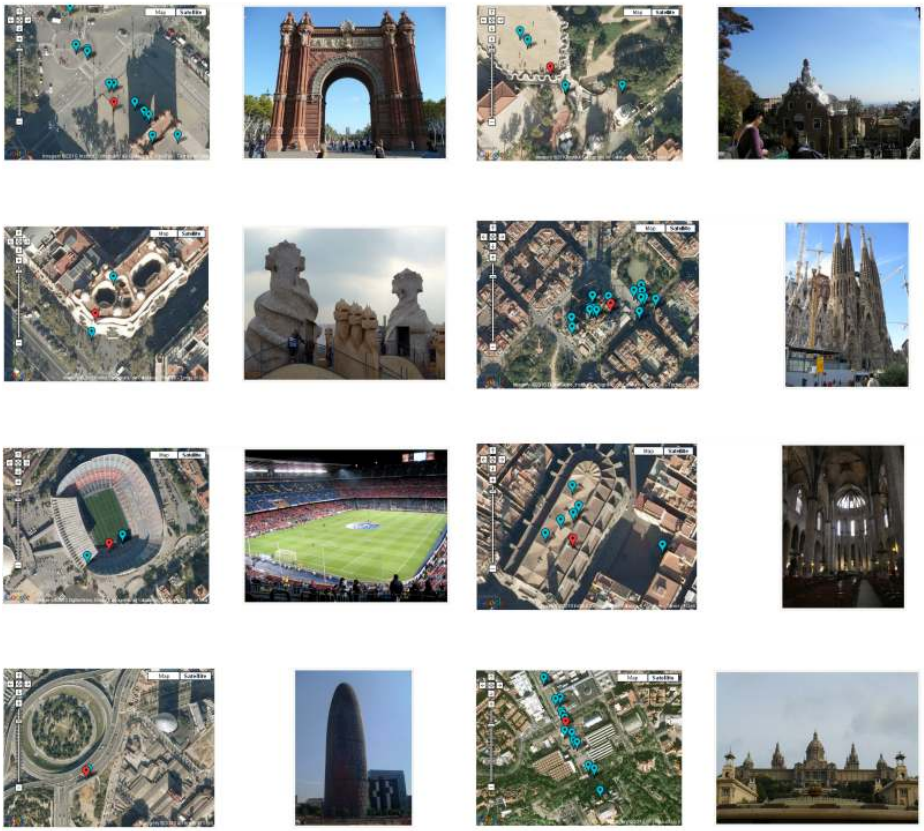


Fig. 18 Samples of query images and location recognition results on the map. For each pair there is the map on the left and the initial query image on the right. Blue marker: Retrieved image. Red marker: Geo-tag estimation.

Table 5 Percentage of correctly localized queries within at most $150m$ from the ground truth location.

Method	Distance threshold		
	$< 50m$	$< 100m$	$< 150m$
Baseline BoW	82.5%	91.6%	94.2%
QE1	86.3%	93.5%	96.2%
QE2	86.7%	93.3%	96.5%
Scene maps	87.8%	94.2%	97.1%

tag which is the exact location of the landmark. Final estimation is far from the ground truth location. This is derived from the fact that geo-tags of user images correspond to the location where the photo was taken from. Thus, these are unsuccessful examples of localizing the landmark but successful ones of localizing the photo.

Table 6 Percentage of correct Wikipedia article suggestions for each landmark and average percentage for the four methods.

Landmark	Method			
	Baseline	QE1	QE2	Scene maps
La Pedrera(a)	100%	100%	100%	100%
Park Guell(a)	100%	100%	100%	100%
Museu Nat. d' Art	40%	100%	60%	80%
Columbus Monument	100%	100%	100%	100%
Carrer del Bisbe Irurit-El Gotic	100%	100%	100%	100%
Port Vell	80%	100%	80%	100%
Sagrada Familia(b)	100%	100%	100%	100%
Casa Batllo	100%	100%	100%	100%
Arc de Triomf	100%	100%	100%	100%
La Pedrera(b)	60%	100%	80%	80%
Hotel Arts	40%	40%	40%	60%
Hospital de Sant Pau(a)	100%	100%	100%	100%
Hospital de Sant Pau(b)	80%	80%	80%	100%
Park Guell(b)	100%	100%	100%	100%
Torre Agbar	100%	100%	100%	100%
Placa de Catalunya	100%	100%	100%	100%
Cathedral (side)	80%	80%	80%	80%
Average	**87%**	**95%**	**90%**	**95%**

Landmark recognition evaluation. Since most photographers are taking pictures of well known landmarks, we can safely assume that some of the annotated groups of images in our European Cities 1M dataset can be linked with Wikipedia articles. Given that the metadata of the images in our European Cities 1M dataset contain user tags, we can use the method proposed in section 6 to analyze them and effectively identify the landmark and suggest Wikipedia articles for each query.

The performance of the approach is shown in Table 6, where we see the percentage of correctly discovered links. Experiments are carried on 17 of the groups, that is the dataset subset which depict landmarks and has corresponding Wikipedia articles. We regard a query link suggestion as correct, if the ground truth article link is one of those suggested from the landmark recognition process. As the table shows, recognition for landmark queries is really efficient both with the use of scene maps and query expansion. Samples of query images and the corresponding suggested and frequent tags are presented in Figure 19. These are examples of successful landmark recognition.

9 Discussion

While mining from user generated content in community photo collections is becoming popular and new applications are emerging, several possibilities are still unexplored. Sub-linear indexing is not typically exploited in landmark recognition applications, while geo-tags are not typically exploited in large scale 3D reconstruction applications. We have combined both, along with a novel scene representation that is directly encoded in our retrieval engine. The result is a considerable increase in retrieval performance, even compared to query expansion methods, at the cost of a slight increase in query time. Memory requirements for the index are also considerably reduced compared to a baseline system. Contrary to landmark recognition applications, we can still

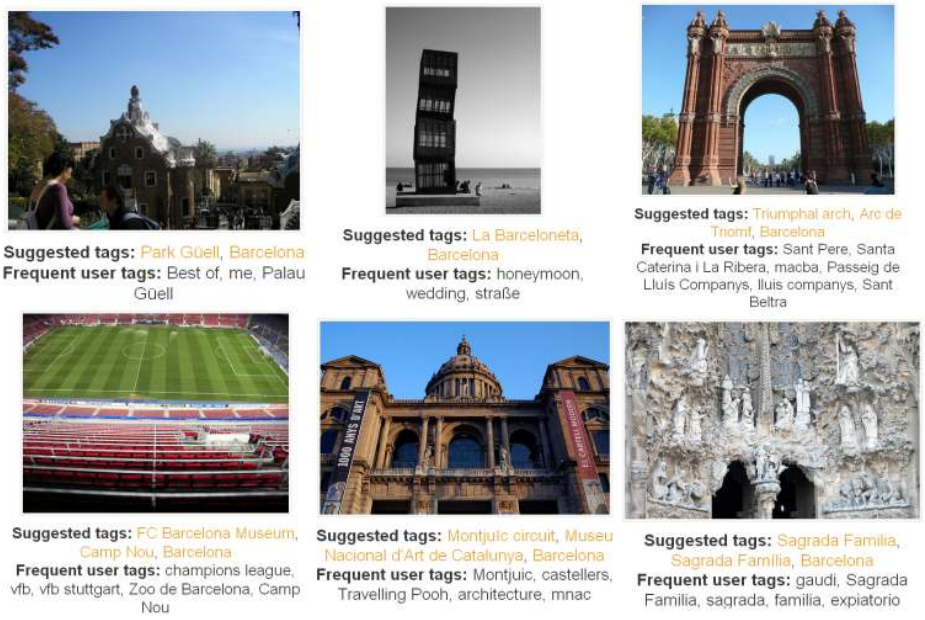


Fig. 19 Samples of query images with suggested and frequent tags. Landmarks are recognized successfully and corresponding Wikipedia links are provided.

retrieve any isolated image from the original database, allowing location recognition at any region where geo-tagged photos are available. Our mining process is even faster than other implementations that employ massive parallelism without exploiting geo-tags. We also recognize landmarks and points of interest by cross-validating location, photo title, frequent tags and geo-referenced Wikipedia article titles in an efficient online process. Our VIRaL application is publicly available online and provides the baseline visual search together with location and landmark recognition.

In the future we would like to investigate more precise methods in measuring dissimilarity of feature appearance during scene map construction. This will enable much more compression of the index, hence increased scalability, as well as more robust matching. Though our visual clustering does not target perceptual summarization or browsing, it may still be the first stage of such a process, exploiting its compact representation and maximum distortion guarantee. Another immediate application can be exact localization *i.e.* pose detection. Finally, regarding our online VIRaL application, we intend to incorporate our scene maps indexing scheme in the interface. All evaluation results on location and landmark recognition presented in this paper, together with a summary of the proposed approach, can be found online in our project homepage[14].

Acknowledgments: This work was partially supported by the European Commission under contract FP7-215453 WeKnowIt.

[14] http://www.image.ntua.gr/iva/research/scene_maps

References

1. S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R. Szeliski. Building Rome in a day. In *International Conference on Computer Vision*, 2009.
2. Y. Avrithis, Y. Kalantidis, G. Tolias, and E. Spyrou. Retrieving landmark and non-landmark images from community photo collections. In *ACM Multimedia*, Firenze, Italy, October 2010.
3. Y. Avrithis, G. Tolias, and Y. Kalantidis. Feature map hashing: Sub-linear indexing of appearance and global geometry. In *ACM Multimedia*, Firenze, Italy, October 2010.
4. H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision*, 2006.
5. Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 17(8):790–799, 1995.
6. O. Chum and J. Matas. Large-scale discovery of spatially related images. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 32(2):371–377, Feb 2010.
7. O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *German Association for Pattern Recognition*, page 236. Springer Verlag, 2003.
8. O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *Computer Vision and Pattern Recognition*, 2009.
9. O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *International Conference on Computer Vision*, 2007.
10. D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world's photos. In *International World Wide Web Conference*, 2009.
11. S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. I know what you did last summer: Object-level auto-annotation of holiday snaps. In *International Conference on Computer Vision*, 2009.
12. R. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge university press Cambridge, UK, 2000.
13. J. Hays and A.A. Efros. IM2GPS: Estimating geographic information from a single image. In *Computer Vision and Pattern Recognition*, 2008.
14. K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *Computer Vision and Pattern Recognition*, 2010.
15. H. Jegou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, pages 1–21, 2010.
16. H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition*, 2010.
17. B. Johansson and R. Cipolla. A system for automatic pose-estimation from a single image in a city scene. In *IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, 2002.
18. E. Kalogerakis, O. Vesselova, J. Hays, A.A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *International Conference on Computer Vision*, 2009.
19. L. Kennedy, M. Naaman, S. Ahern, R. Nair, and T. Rattenbury. How flickr helps us make sense of the world: Context and content in community-contributed media collections. In *ACM Multimedia*, volume 3, pages 631–640, 2007.
20. C.H. Lampert. Detecting objects in large image collections and videos by efficient subimage retrieval. In *International Conference on Computer Vision*, 2009.
21. 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.
22. V.I. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1(1):8–17, 1965.
23. X. Li, C. Wu, C. Zach, S. Lazebnik, and J.M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *European Conference on Computer Vision*, pages 427–440. Springer, 2008.
24. Y. Li, D.J. Crandall, and D.P. Huttenlocher. Landmark classification in large-scale image collections. In *International Conference on Computer Vision*, 2009.
25. D.G. Lowe. Local feature view clustering for 3D object recognition. In *Computer Vision and Pattern Recognition*, 2001.

26. D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
27. J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.
28. A. McCallum, K. Nigam, and L.H Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *6Th ACM International Conference on Knowledge Discovery and Data Mining*, page 178, 2000.
29. M. Muja and D.G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision*, 2009.
30. D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition*, 2006.
31. A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.
32. M. Perdoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *Computer Vision and Pattern Recognition*, 2009.
33. J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition*, 2007.
34. T. Quack, B. Leibe, and L. Van Gool. World-scale mining of objects and events from community photo collections. In *CIVR*, pages 47–56, 2008.
35. D. Robertson and R. Cipolla. An image-based system for urban navigation. In *British Machine Vision Conference*, 2004.
36. F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or how do i organize my holiday snaps. In *European Conference on Computer Vision*, 2002.
37. G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Computer Vision and Pattern Recognition*, 2007.
38. C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition*, 2008.
39. I. Simon, N. Snavely, and S.M. Seitz. Scene summarization for online image collections. In *International Conference on Computer Vision*, 2007.
40. J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision*, pages 1470–1477, 2003.
41. N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *Computer Graphics and Interactive Techniques*, pages 835–846, 2006.
42. N. Snavely, S.M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *Computer Vision and Pattern Recognition*, 2008.
43. U. Steinhoff, D. Omercevic, R. Perko, B. Schiele, and A. Leonardis. How computer vision can help in outdoor positioning. In *European Conference on Ambient Intelligence*, 2007.
44. M. Tipping and B. Schölkopf. A kernel approach for vector quantization with guaranteed distortion bounds. In *Artificial Intelligence and Statistics*, pages 129–134, 2001.
45. W. Zhang and J. Kosecka. Image based localization in urban environments. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.
46. Y. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *Computer Vision and Pattern Recognition*, 2009.