Retrieving Landmark and Non-Landmark Images from Community Photo Collections

Yannis Avrithis
National Technical University of Athens
Iroon Polytexneiou 9
Zografou, Greece
iavr@image.ntua.gr

Yannis Kalantidis
National Technical University of Athens
Iroon Polytexneiou 9
Zografou, Greece
ykalant@image.ntua.gr

Giorgos Tolias
National Technical University of Athens
Iroon Polytexneiou 9
Zografou, Greece
gtolias@image.ntua.gr

Evaggelos Spyrou
National Technical University of Athens
Iroon Polytexneiou 9
Zografou, Greece
espyrou@image.ntua.gr

ABSTRACT

State of the art data mining and image retrieval in community photo collections typically focus on popular subsets, *e.g.* images containing landmarks or associated to Wikipedia articles. We propose an image clustering scheme that, seen as vector quantization, compresses a large corpus of images by grouping visually consistent ones while providing a guaranteed distortion bound. This allows us, for instance, to represent the visual content of all thousands of images depicting the Parthenon in just a few dozens of *scene maps* and still be able to retrieve any single, isolated, non-landmark image like a house or a graffiti on a wall.

Starting from a geo-tagged dataset, we first group images geographically and then visually, where each visual cluster is assumed to depict different views of the the same scene. We align all views to one reference image and construct a 2D scene map by preserving details from all images while discarding repeating visual features. Our indexing, retrieval and spatial matching scheme then operates directly on scene maps. We evaluate the precision of the proposed method on a challenging one-million urban image dataset.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis

General Terms

Algorithms and Experimentation

Keywords

image retrieval, image clustering, sub-linear indexing, geo-tagging

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 to permit searching into such huge collections. On the other hand, a number of data mining and clustering approaches have emerged that exploit data such as 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. What is more interesting, new applications have emerged, for instance location estimation as in Hayes and Efros [10], virtual tourism as in Snavely *et al.* [32], and landmark recognition as in Zheng *et al.* [36].

Unlike [10], which only estimates a geolocation probability map, we are interested in developing a retrieval method that allows *location recognition* of any kind of image, landmark and non-landmark, provided that the dataset contains at least one instance of that scene. Such matching, typically possible in urban scenes, may allow automatic geo-tagging of a new image. Along with triangulation, it may also lead to exact localization as in Zhang and Kosecka [35]. Unfortunately, current approaches to location recognition either do not scale well, or focus on popular locations like landmarks.

Our work lies between generic image retrieval and clustering. It turns out that, seeing clustering as vector quantization, it is *distortion* that determines how isolated data are treated. Therefore, while large image clusters of popular places help in boosting the efficiency of retrieval, a distortion

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

bound can guarantee that isolated images are still retrieved as in a generic retrieval engine.

For instance, a common approach is to cluster a given set of geo-tagged images by location, as in Crandall *et al.* [6]. Our objective here is to identify images that potentially depict views of the same scene and then compute visual similarities. Rather then spatial density, time, or number of photographers that typically measure popularity, it is maximal distance that counts. *E.g.*, two images taken $2km$ apart are unlikely to depict the same building. Likewise, spatial matching by RANSAC [7] or as in Philbin *et al.* [26] relies on the number of inlier visual features found between two images. In this case *e.g.* 20 images, each having at least 15 inliers with a reference image, may all depict similar views of a single scene.

We use the *kernel vector quantization* (KVQ) approach of Tipping and Schölkopf [34], along with an appropriate metric for each of the two cases above. Contrary to *e.g.* k-means or agglomerative clustering, it guarantees that no image in a geographic cluster is taken too far away from a specific location, and no image in a visual cluster has too few inliers with a specific image.

To speed up the mining process, we apply visual clustering only to the set of images in a single geographical cluster and we use sub-linear indexing to compute the required pairwise dissimilarities. Given a visual cluster, we align all images to a reference image by homography estimation and construct a 2D *scene map* by grouping similar local features, which gives rise to another application of KVQ: no feature in a cluster should be too far from a specific feature, where by “far” we mean either spatially or in terms of appearance.

Finally, 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 increases recall as well. At query time, it takes milliseconds to filter relevant scene maps, and a couple of seconds to re-rank according to geometry. The mining process is entirely automatic. We experiment on a challenging one-million urban image dataset from 22 cities.

2. RELATED WORK

2.1 Location Recognition

In one of the earliest works on multiview matching in city scenes, Johansson and Cipolla [13] estimate homographies between pairs of images and provide automatic *pose estimation.* Using SIFT features [20], Zhang and Kosecka [35] search directly in the descriptor space for the closest feature matches. They find the closest reference view in a small image database, thereby providing coarse *location recognition* in urban environments.

Steinhoff *et al.* [33] 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 $200 \times 200m^2$. Schindler *et al.* [28] are among the first to use inverted file indexing by means of a vocabulary tree [24] for *city-scale* location recognition, scaling up to $30,000$ images covering $20km$ of streetside views.

Hayes and Efros [10] take the leap to *world-scale* geographic estimation by searching into a database of 6 million geo-tagged images downloaded from Flickr[2]. The price to pay is that images are now represented by global features like color/texton histograms, GIST descriptors etc. Matching accuracy is not even comparable to that of local features and the output is a *geolocation probability map.* Kalogerakis *et al.* [14] build on the previous result by exploiting the time each photo is taken, much like [6]. Using human travel priors they significantly improve accuracy; however, the output remains a probability map and anyhow this only works for image *sequences* rather than a single image query.

2.2 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 [31] go beyond individual features and show how text retrieval techniques like codebooks, inverted file indexing, and TF-IDF weighting can apply to visual search. Nister and Stewenius [24] extend to hierarchical codebooks and construct a *vocabulary tree* that is also used to assign features to visual words. Philbin *et al.* [26] 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 [29] 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.* [5] 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.* [12] and Perdoch *et al.* [25], mainly related to geometry and visual codebooks.

2.3 Structure from Motion

Another interesting application is vision-based reconstruction and navigation of a 3D scene from a collection of widely separated views. A recent example of such *structure from motion* is Snavely *et al.* [32], which scales to datasets of 10^3 images acquired by text queries from Flickr. At similar scales, Li *et al.* [17] attempt to speed up reconstruction by a hierarchical approach. Due to the use of global descriptors, the increased speed comes at a loss of accuracy. On the other hand, Agarwal *et al.* [1] reconstruct *city-scale* models from Flickr datasets in the order of 10^5 photos. A vocabulary tree is again used for indexing, and query expansion like in [5] helps make the matching image graph dense enough. 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 then each clustering sub-problem would 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

[2]http://www.flickr.com

in any way to help retrieval or location recognition of a new photo.

2.4 Landmark Recognition

Kennedy *et al.* [15] 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 poorly due to the global features employed. Likewise, Crandall *et al.* [6], 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.* [18] slightly improve performance using a multi-class SVM classifier. Temporal information also helps, as in [14]. 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.* [30] 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.* [11]. Parallelism is again the key in the high computational cost involved. Chum and Matas [3], again without location data, extend to *web-scale* visual clustering, 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.* [27] divide the areas of interest into overlapping square tiles instead of performing location clustering. Similarly to [15] and contrary to [30] and [3], they perform visual clustering inside each geographic cluster only, making the problem more tractable. On the other hand, they perform pairwise homography estimation without indexing and subsequent agglomerative clustering, thus probably loosing the computational advantage. Tags and user information are then employed to mine landmarks, objects or events and link to Wikipedia articles. In the absence of indexing, the use of this kind of mining for location recognition of a new image is severely limited, since one has to resort to exhaustive linear search. Gammeter *et al.* [8] 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, frequent features in an image cluster are used to detect and automatically label the object of interest.

Finally, Zheng *et al.* [36] perform a similar combination of geo-clustering and visual (agglomerative) 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. In all the above approaches, clustering helps to either construct high-level summaries or limit search to a small percentage of images. It should be clear by now that this will only work for landmarks and other popular places or events.

3. VIEW CLUSTERING

As 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.

Different strategies are followed in existing work. For instance, [6] and [18] 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 ([15]) and agglomerative clustering ([27], [8], [36]). For geo-clustering, [15] and [36] use the same algorithm as for visual clustering, whereas [27] and [8] simply quantize locations into overlapping rectangular tiles. There are also [17], [30] and [3] which perform visual clustering alone. Naturally, this does not scale well.

The 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—global descriptors are employed in [15]. We rather use the *kernel vector quantization* (KVQ) approach of [34]. Seeing KVQ as an encoding process, the maximal intra-cluster distance is now the maximum level of *distortion*. KVQ guarantees an upper bound on distortion and adjusts the number of clusters accordingly.

Mean-shift, used in [6] and [18], 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 to either run initially for every point, or it requires some kind of *seeding*. For example, [6] 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 [27] 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 [16]. As a branch-and-bound method, [16] 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.

We summarize KVQ and its properties below. We then discuss our specific two-layer clustering scheme and give examples of geo-clusters and visual clusters.

3.1 Kernel Vector Quantization

Let (X, d) be a metric space and suppose we are given a finite data set $D \subseteq X$ of cardinality $|D| = n$, whose elements we may list as $D = \{x_1, \ldots, x_n\}$. The objective is to select a subset $Q(D) \subseteq D$ that is as small as possible, under the constraint that all points in D are not too far away from some point in Q. If

$$B_r(x) = \{y \in X : d(x,y) < r\} \tag{1}$$

is the open ball in X of radius r centered at x, and $\mathbb{1}_A : X \to \{0,1\}$ denotes the indicator function of set $A \subseteq X$, define *kernel function* $k : X \times X \to \mathbb{R}$ as

$$k(x,y) = \mathbb{1}_{B_r(x)}(y) \tag{2}$$

to indicate whether points $x, y \in X$ lie within distance r, where $r > 0$ is typically given as an input *scale* parameter. Given a point $x \in X$, define the *empirical kernel map*

$$\boldsymbol{\phi}(x) = (k(x_1,x), \ldots, k(x_n,x))^\top. \tag{3}$$

The key observation is that if there is a *weight vector* $\mathbf{w} \in \mathbb{R}^n$ with components w_j such that for all $x \in D$,

$$\mathbf{w}^\top \boldsymbol{\phi}(x) > 0 \tag{4}$$

then all points $x \in D$ lie within distance r of some point $x_j \in D$ with a positive associated weight $w_j > 0$. To achieve a *sparse* solution for $\mathbf{w}$ satisfying (4), one typically uses the ℓ_1 norm in $\mathbb{R}^n$, giving rise to the following optimization problem:

$$\min_{\mathbf{w} \in \mathbb{R}^n} \quad \|\mathbf{w}\|_1 \tag{5}$$

$$\text{subject to} \quad \mathbf{w}^\top \boldsymbol{\phi}(x) \geq 1 \quad \forall x \in D. \tag{6}$$

Now, given a point $x \in D$, define *cluster* $C(x) = D \cap B_r(x) = \{y \in D : d(x,y) < r\}$ as the set of all points $y \in D$ that lie within distance r from x. A slightly adjusted penalizer instead of ℓ_1 norm then makes the penalty associated to weight w_j inversely proportional to the *support* $|C(x_j)|$ of x_j, thus favoring larger clusters. Defining vector $\boldsymbol{\gamma} \in \mathbb{R}^n$ with elements $\gamma_j = |C(x_j)|^{-1} = \|\boldsymbol{\phi}(x_j)\|_1^{-1}$, one ends up with the following linear programming problem:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta} \in \mathbb{R}^n} \quad \boldsymbol{\gamma}^\top(\boldsymbol{\alpha} + \boldsymbol{\beta}) \tag{7}$$

$$\text{subject to} \quad \mathbf{K}(\boldsymbol{\alpha} - \boldsymbol{\beta}) \geq 1 \tag{8}$$

$$\boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0, \tag{9}$$

where $\mathbf{w}$ has been decomposed as $\mathbf{w} = \boldsymbol{\alpha} - \boldsymbol{\beta}$ and $\mathbf{K}$ is the *Gram* matrix with elements $K_{ij} = k(x_i, x_j)$. Given the optimal solution $\mathbf{w}^\star = \boldsymbol{\alpha}^\star - \boldsymbol{\beta}^\star$ with components $w_j^\star$, the *codebook* $Q(D)$ of data set D is defined as

$$Q(D) = \{x_j \in D : w_j^\star > 0\}. \tag{10}$$

Clearly, $Q(D) \subseteq D$, that is, *codebook vectors* are points of the original data set. Alternatively, we shall refer to such points as *cluster centers*. By construction, the *maximal distortion* is upper bounded by r because $\max_{y \in C(x)} d(x,y) < r$ for all $x \in Q(D)$. Also, the *cluster collection*

$$\mathcal{C}(D) = \{C(x) : x \in Q(D)\} \tag{11}$$

is a *cover* for D because $D = \bigcup_{x \in Q(D)} C(x)$. 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*. This 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.

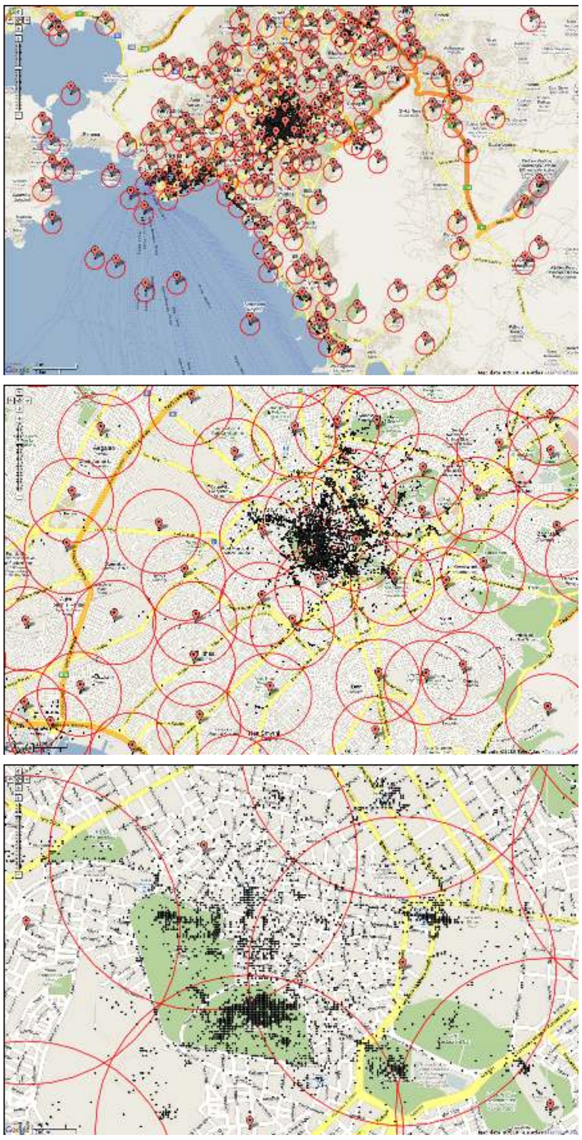


Figure 1: Maps of Athens illustrating geo-clusters at three different zoom levels. Black dots, red markers and red circles stand for photos, codebook vectors and cluster boundaries, respectively.

The above is not the optimal solution in terms of codebook size $|Q(D)|$, for which we would have to resort to combinatorial optimization, so there is some degree of redundancy. A subsequent *pruning* step removes at random order from $Q(D)$ any redundant vector x such that the cluster collection of the remaining points is still a cover for D[3]. That is, any vector $x \in Q(D)$ such that $C(x) \subseteq \bigcup_{y \in Q(D) \setminus \{x\}} C(y)$. We assume pruning is always performed and we use the same symbols $Q(D)$, $\mathcal{C}(D)$ to denote the final codebook and cluster collection after this step, respectively.

[3]Such vectors are typically not more than 5% of the codebook size.

3.2 Geo-clustering

Define a set of photos P—we will use the terms *photo*, *image* and *view* interchangeably. Each photo $p \in P$ is represented 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 either descriptors or (most often) visual word labels in case descriptors have been quantized over a visual codebook. 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 $d_g : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is defined as the geodesic distance[4] between any two points on the surface of Earth. Let $d_g(p, q)$ be the geodesic distance between locations of photos $p, q \in \mathcal{P}$. For simplicity, we will use $B^g(p)$ to denote the open ball in $\mathcal{P}$ of radius r_g centered at p, assuming metric d_g. Given a photo $p \in P$, define a *geo-cluster* as

$$C_g(p) = P \cap B^g(p) = \{q \in P : d_g(p, q) < r_g\}. \tag{12}$$

Similarly, given the resulting codebook $Q_g(P)$, define the *geo-cluster collection*

$$\mathcal{C}_g(P) = \{C_g(p) : p \in Q_g(P)\}. \tag{13}$$

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 an entire city like Barcelona with $|P| = 10^5$ geo-tagged photos. We use a sparse representation of the Gram matrix $\mathbf{K}_g$ but compute it by enumerating all $(p, q) \in P^2$ (or just one p per bucket).

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.

3.3 Visual Clustering

As in [30], 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, *e.g.* SIFT [20], SURF [2], or MSER [21].

Descriptors may be matched pairwise according to a number of different strategies, *e.g.* mutual nearest neighbors, distance threshold, distance ratio or combinations [22]. Alternatively, they may by quantized up to visual word against a large (*e.g.* 10^5 or 10^6) visual codebook and matched when mapped to the same visual word, possibly checking consistency in spatial neighborhoods as well. Rigid geometry is typically verified by means of RANSAC [7] or simpler forms of spatial matching like [26], [25]. The geometric model may vary from similarity (or even of 3-DOF including translation and uniform scaling, without rotation) to affine, homography or fundamental matrix.

[4]http://en.wikipedia.org/wiki/Great-circle_distance

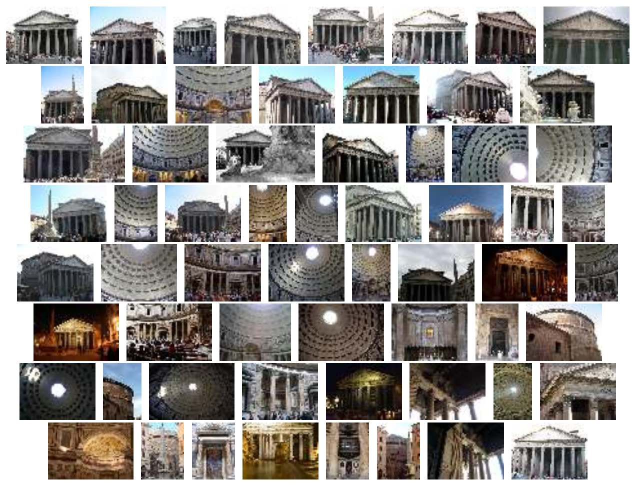


Figure 2: Photos associated to the centers of the most populated visual clusters from Pantheon, Rome.

Whatever the choices, the output is typically the number of *inliers* $I(p, q)$ between visual feature sets F_p, F_q of photos p, q respectively. Since $I(F_p, F_q)$ is a similarity measure, any decreasing function will do as a metric, *e.g.*

$$d_v(p, q) = \exp\{-I(F_p, F_q)\} \tag{14}$$

The exact formula of $d_v(p, q)$ is not important, since kernel function k is discrete. Given a scale parameter r_v, the kernel function is equivalently

$$k_v(p, q) = \begin{cases} 1, & I(F_p, F_q) > \tau \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where $\tau = -\log r_v$. 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. Let $Q_v(G)$ be the resulting codebook, and define *visual cluster* $C_v(p) = G \cap B^v(p)$ for $p \in G$ and *visual cluster collection* $\mathcal{C}_v(G) = \{C_v(p) : p \in Q_v(G)\}$, similarly to (12) and (13), respectively. Again, we have used $B^v(p)$ to denote the open ball in $\mathcal{P}$ of radius r_v centered at p, assuming metric d_v. Repeating over all geo-clusters, the complete codebook $Q(P)$ over the entire data set is

$$Q(P) = \bigcup_{G \in \mathcal{C}_g(P)} Q_v(G). \tag{16}$$

Finally, the set of all *view clusters* $\mathcal{C}(P)$ is defined as

$$\mathcal{C}(P) = \{C_v(p) : p \in Q(P)\}. \tag{17}$$

The main bottleneck the clustering process above is construction of Gram matrix $\mathbf{K}$, which is typically quadratic in the data set size $|D|$. This is not an issue in geo-clustering but is critical in visual clustering. The same problem appears in Quack *et al.* [27] 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

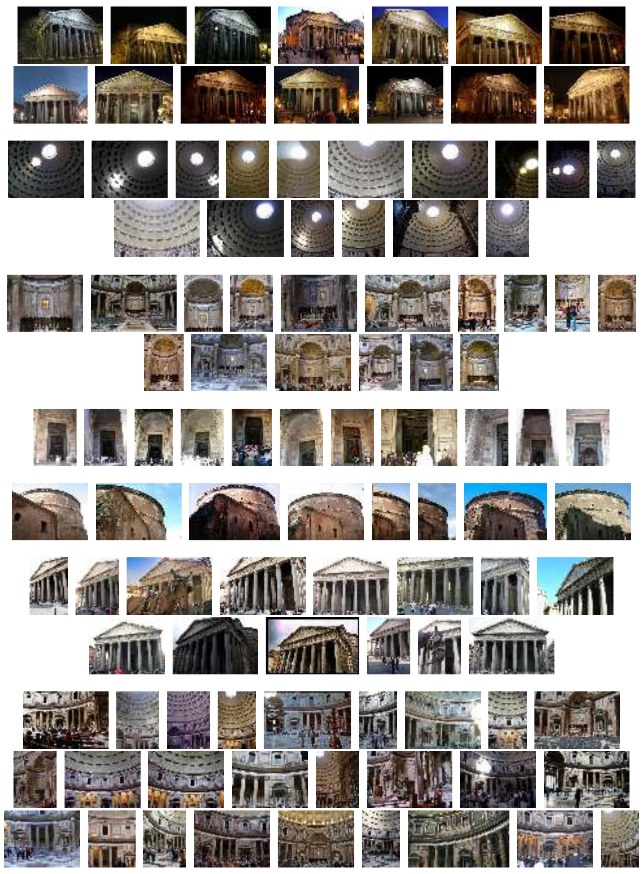


Figure 3: Photos in a sample of visual clusters from Pantheon. The first image in each cluster corresponds to the cluster center.

$200m$, which is quite often. The same quadratic cost appears *e.g.* in [8],[36],[30], while for [15] this is a reason for *not* using local features. We use larger geo-clusters with $r_g = 700m$, yet achieve a very fast implementation.

The key is *geo-cluster specific* sub-linear indexing. In particular, we use an inverted file indexed by both visual word and geo-cluster, with TF-IDF weighting and a variant of fast spatial matching [26] over a 4-DOF similarity model, as described in section 4. 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 a second. In effect, one such query returns a sparse representation of one entire empirical kernel map vector $\boldsymbol{\phi}$, or one row/column of $\mathbf{K}$. The construction of $\mathbf{K}$ is now linear in $|G|$. Compared to [3], we have the advantage of geo-clustering. This lowers the cost and allows one query per image in each geo-cluster. On the other hand, [3] employs hashing with low recall, and is thus limited to popular locations—isolated photos are unlikely to get discovered.

To illustrate the effect of visual clustering on a set of photos, we give an example from Pantheon, Rome, following the examples appearing in [30] and [27]. 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 1146 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. Unlike [30], the objective here is neither summarization nor canonical 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. Observe that 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.

4. SCENE MAPS

We do estimate homographies between matching images, eventually. However, this takes place only within each view cluster, and the number of tests is *linear* in the size of each cluster. Initial estimates of each homography are readily available from the responses of each query, so only the final step of local optimization is required. Given all homographies in a view cluster, we align all views to the reference image of the relevant codebook vector. We collect all aligned visual features and construct what we call a *scene map*, because it is a 2D spatial map of features associated to different views of the same scene. Because all views are aligned, it makes sense to match a query image to an entire scene map under the same geometry. Scene maps are then used 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.

4.1 Spatial Matching

When a query is issued during visual clustering, the top ranked images after TF-IDF voting are geometrically verified based on the *single correspondence assumption* of [26]. In particular, tentative correspondences between the visual features of the query and each image in the list are generated by matching visual words. Given two corresponding features with local shape described by two circular regions (typical with scale and rotation covariant features like SIFT and SURF), we find similarity transformations T_1, T_2 that map the regions 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 and whenever a new maximum is found, we find a least squares estimate of an affine transform from the given inliers and store the best model so far—this is the “simple” method of Locally Optimized RANSAC (LO-RANSAC) [4]. For each non-zero entry of matrix $\mathbf{K}$, that is 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.

When visual clustering is complete we align all images in each cluster using a homography model. Specifically, each image $p \in Q(P)$ is treated as a *reference* image in the corresponding view cluster $C_v(p) \in \mathcal{C}(P)$. We know by construction of view clusters that each image $q \in C_v(p)$ has been

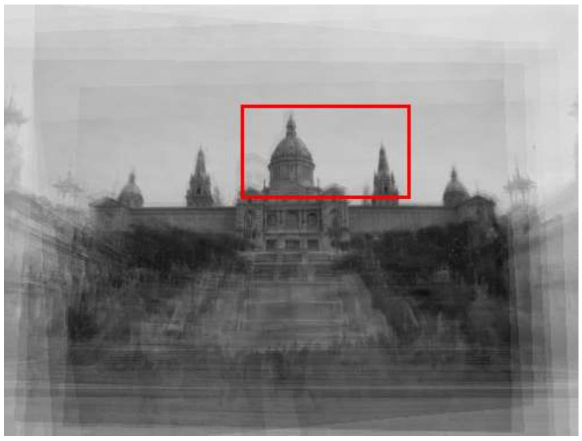
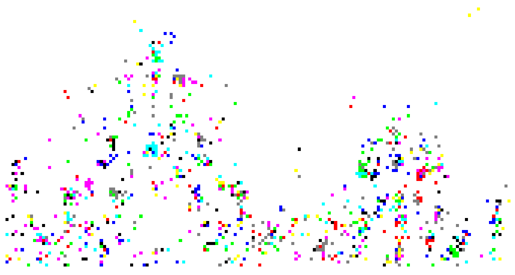
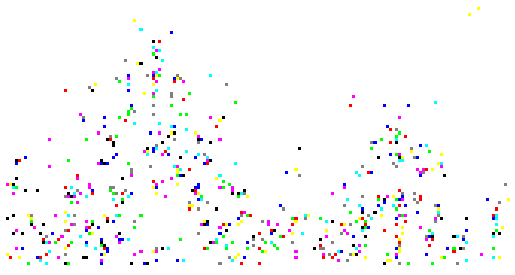


Figure 4: Scene map construction from 10 photos of Palau Nacional, Montjuic, Barcelona.



(a)



(b)

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

geometrically verified, giving $I(p,q) > \tau$ inliers. We now align q to p and compute a relevant homography H_q. For this, 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_θ are used to estimate a homography with the Direct Linear Transformation (DLT) algorithm [9]. 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}.

4.2 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, if each visual feature is represented by a tuple (x, w) with x being the position / local shape and w the visual word label, then this collection is constructed as

$$F(p) = \bigcup_{q \in C_v(p)} \{(H_{qp}x, w) : (x, w) \in F_q\}. \tag{18}$$

Here, x is assumed either a 3-vector in projective space $\mathbb{P}^2$ with the homogeneous coordinates of feature position, or a 3×3 matrix containing local shape as well. In the former case, local shape should be computed and stored separately to be used in spatial matching as described above.

The above formulation bears similarities with several models in different contexts. To name a few, Lowe [19] 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.* [30] organize matching features of multiple images into *tracks*, where a track is a *connected component* of features and corresponds to a single 3D point of a scene. From these tracks, they construct an incidence matrix, compute similarities and produce a visual summary of the scene by means of a set of *canonical views*. Gammeter *et al.* [8] perform a similar alignment in visual clusters with the objective of isolating bounding boxes of depicted landmarks. In image retrieval, Chum *et al.* [5] 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*.

In our case, the objective is to construct a compact representation of $F(p)$ that we will refer to as the *scene map* $S(p)$, such that retrieval is performed on scene maps directly, rather than database images. Ideally, a query should match a scene map whenever it matches any single image in the map. This will make it possible to retrieve images that would not match by themselves, effectively increasing recall. It is similar to the latent model of [5], which however 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 [8] we want to keep information from all image regions. Matching features are *linked* into connected components in [19], [30], 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. We should therefore keep a minimal subset $S(p) \subseteq F(p)$ such that no feature in $F(p)$ is too distant from its nearest neighbor in $S(p)$.

The above discussion gives rise to vector quantization once more. In particular, we choose to apply KVQ to $F(p)$ in space $(\mathcal{F}, d_f)$ where $\mathcal{F}$ is the set of all possible features and metric d_f should measure distance, both spatial and in appearance. Since each feature $f \in \mathcal{F}$ is represented by tuple (x, w), $\mathcal{F}$ is a *product space* $\mathcal{X} \times \mathcal{W}$, where $\mathcal{X}$ refers to position / local shape and $\mathcal{W}$ to appearance. Consequently, d_f may be defined as a product metric, where spatial distance is measured by the Euclidean metric in $\mathcal{X}$, while distance in appearance by the *discrete* metric in $\mathcal{W}$. The latter choice

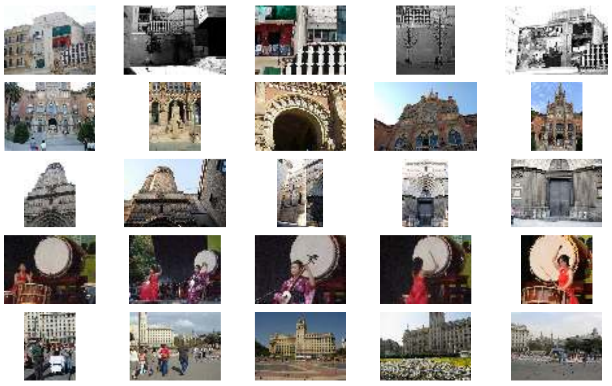


Figure 6: A sample of five groups of Barcelona query images from *European Cities 1M* dataset used in our experiments.

simplifies the process a lot: in effect we can partition $F(p)$ into a number of disjoint sets

$$F_w(p) = \{(x, u) \in F(p) : u = w\}, \tag{19}$$

each corresponding to a visual word w, and apply KVQ separately to each $F_w(p)$ in $(\mathcal{X}, d_x)$ with scale parameter r_x. Now, either $\mathcal{X} = \mathbb{P}^2$ or $\mathcal{X} = \mathbb{R}^{3\times3}$. In either case, if $t_x \in \mathbb{R}^2$ denotes the *position* component of x in the 2D image plane, metric d_x is defined as $d_x(y, z) = \|t_y - t_z\|_2$ for $y, z \in \mathcal{X}$. Finally, join the resulting codebooks $Q_x(F_w(p))$ into a single scene map:

$$S(p) = \bigcup_{w \in \mathcal{W}} Q_x(F_w(p)). \tag{20}$$

In fact, 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.

For an example of scene map construction, we use a visual cluster containing 30 images of Palau Nacional, Montjuic, Barcelona, 10 of which are depicted in overlay after alignment in Figure 4. Of the $11,623$ 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.

4.3 Indexing and Retrieving Scene Maps

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 in (20). 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, retrieve relevant scene maps by histogram intersection and TDF-IF weighting, and re-rank.

A short list of top-ranking scene maps is verified geometrically. We make use again of the single correspondence assumption and follow the "simple" method of LO-RANSAC whenever a new maximum is encountered. Note that even if the initial estimate is a similarity transformation, we can still recover the correct homography transformation by least squares fitting given at least four inliers.

To speed up the re-ranking process, we terminate iterations and regard as relevant an image that yields at least τ_h inliers. We also consider an image as irrelevant if no more than τ_ℓ inliers are found after having checked a predefined percentage of all hypotheses. Finally, we discard the current hypothesis if the symmetric transfer error is computed for more than a predefined percentage of correspondences and the maximum number of inliers is less 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. To draw an analogy *e.g.* to the latent model for query expansion of [5], recall that scene maps are statically computed in the off-line indexing process and constrained within geo-clusters. On the other hand, in [5] a model is built dynamically at query time, 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.

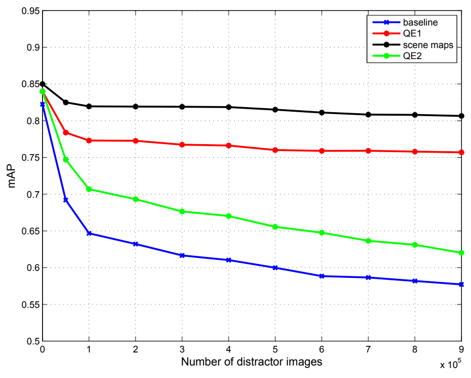


Figure 7: mAP measurements for the four benchmarked methods on the *European Cities 1M* dataset under a varying number of distractors.

5. EXPERIMENTS

5.1 Dataset

We experiment on a challenging one-million urban image dataset, namely *European Cities 1M*[5]. It consists of a total of $1,037,574$ geo-tagged images from 22 European cities that we have crawled from Flickr using geographic queries covering a window of each city center. A subset of 927 images from Barcelona are annotated into 17 groups of images depicting the same scene, building or landmark. Since not all are landmarks, annotation cannot rely on tags; it is rather

[5]The dataset will soon be publicly available at http://image.ntua.gr/iva/datasets

Method	Avg. query time	mAP
Baseline BoW	1.03s	0.5772
QE1	20.3s	0.7574
QE2	2.51s	0.6202
Scene maps	1.29s	0.8065

Table 1: Average query time and mAP of the four benchmarked methods on the *European Cities 1M* dataset including all distractors.

a combination of visual query expansion and manual clean-up. Five images are selected as queries from each group, for a total of 85 queries. To ensure that no other images in the dataset depict the same scene as the ground truth, we have removed the entire set of $128,715$ Barcelona images from the dataset; the remaining $908,859$ images are the *distractors*. Most of them depict urban scenery like the ground-truth, making a challenging distractor dataset. Sample query images are shown in Figure 6.

5.2 Evaluation protocol

In all experiments, we have used the medium Flickr image size, which is 500×500 pixels maximum. We have extracted SURF features and descriptors [2] and kept a maximum of $1,000$ features per image. We have built a 75K visual codebook trained from a set of images of urban scenes that are not part of our evaluation dataset. We have used flat k-means, where, as in [26], nearest cluster centers at each iteration have been found with the randomized kd-tree of Silpa-Anan and Hartley [29], and specifically using the FLANN library of Muja and Lowe [23]. Larger codebooks did not perform well in scene map construction. All features in all images in the data set have been mapped to visual words again using FLANN. Our bag of words implementation uses histogram intersection similarity on L_1-normalized vectors and TF-IDF weighting. Details on the remaining processes of indexing, spatial matching etc. during visual clustering and scene map construction are given in sections 3 and 4, respectively. We evaluate overall retrieval performance via mean average precision (mAP). All experiments are performed with our own C++ implementation on a 2GHz Quad Core processor with 8GB of memory.

5.3 Results

The mining process is entirely automated. We start from a baseline system where the dataset is already indexed so can perform queries. Then, 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 1.2GB of memory instead of 1.61GB for the baseline, providing a compression of 25%.

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 results and then merge the results. In our experiments, this expansion was carried out iteratively, for three levels per 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 85 ground truth queries for all four methods under varying size of distractor set are depicted in Figure 7. Observe that our method using scene maps outperforms all other methods in terms of mean average precision. Surprisingly, it even performs better than the QE1 method. The explanation for this can be found in the use of *geo-cluster specific* sublinear indexing (see Section 3.3) during scene map construction. While in QE1 the expanded set of similar images comes from multiple queries in the whole database of 1 Million images, when creating a scene map visual similarities are obtained through querying the index of a single geo-cluster.

As shown in Table 1, 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. 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.

6. DISCUSSION

While mining from user generated content in community photo collections and new applications are becoming popular, 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 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 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.

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. Our approach has already given good results on location and landmark recognition. Some quantitative evaluation results, both for landmark and non-landmark scenes, can be found online in our project homepage[6].

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, 2009.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006.

[3] O. Chum and J. Matas. Large-scale discovery of spatially related images. *PAMI*, 32(2):371–377, 2010.

[4] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *DAGM*, page 236. Springer Verlag, 2003.

[5] 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 *ICCV*, 2007.

[6] D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the world's photos. In *WWW*, 2009.

[7] M. Fischler and R. 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.

[8] S. Gammeter, L. Bossard, T. Quack, and L. V. Gool. I know what you did last summer: Object-level auto-annotation of holiday snaps. In *ICCV*, 2009.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry.* Cambridge university press Cambridge, UK, 2000.

[10] J. Hays and A. A. Efros. IM2GPS: Estimating geographic information from a single image. In *CVPR*, 2008.

[11] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. J. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *CVPR*, 2010.

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

[13] B. Johansson and R. Cipolla. A system for automatic pose-estimation from a single image in a city scene. In *Proc. IASTED Int. Conf. Signal Processing, Pattern Recognition and Applications*, 2002.

[14] E. Kalogerakis, O. Vesselova, J. Hays, A. A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *ICCV*, 2009.

[15] 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.

[16] C. Lampert. Detecting objects in large image collections and videos by efficient subimage retrieval. In *ICCV*, 2009.

[17] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, pages 427–440. Springer, 2008.

[18] Y. Li, D. J. Crandall, and D. P. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, 2009.

[19] D. Lowe. Local feature view clustering for 3D object recognition. In *CVPR*, 2001.

[20] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[21] 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.

[22] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[23] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *ICCV*, 2009.

[24] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[25] M. Perdoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *CVPR*, 2009.

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

[27] 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.

[28] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

[29] C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. In *CVPR*, 2008.

[30] I. Simon, N. Snavely, and S. Seitz. Scene summarization for online image collections. In *ICCV*, 2007.

[31] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[32] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *Computer Graphics and Interactive Techniques*, pages 835–846, 2006.

[33] 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.

[34] M. Tipping and B. Schölkopf. A kernel approach for vector quantization with guaranteed distortion bounds. In *Artificial Intelligence and Statistics*, pages 129–134, 2001.

[35] W. Zhang and J. Kosecka. Image based localization in urban environments. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[36] 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 *CVPR*, 2009.

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