Noname manuscript No.
(will be inserted by the editor)

Hough Pyramid Matching: Speeded-up geometry re-ranking for large scale image retrieval

Yannis Avrithis and Giorgos Tolias
National Technical University of Athens
Iroon Polytexneiou 9 Zografou, Greece
{iavr,gtolias}@image.ntua.gr

the date of receipt and acceptance should be inserted later

Abstract Exploiting local feature shape has made geometry indexing possible, but at a high cost of index space, while a sequential spatial verification and re-ranking stage is still indispensable for large scale image retrieval. In this work we investigate an accelerated approach for the latter problem. We develop a simple spatial matching model inspired by Hough voting in the transformation space, where votes arise from single feature correspondences. Using a histogram pyramid, we effectively compute pair-wise affinities of correspondences without ever enumerating all pairs. Our *Hough pyramid matching* algorithm is linear in the number of correspondences and allows for multiple matching surfaces or non-rigid objects under one-to-one mapping. We achieve re-ranking one order of magnitude more images at the same query time with superior performance compared to state of the art methods, while requiring the same index space. We show that soft assignment is compatible with this matching scheme, preserving one-to-one mapping and further increasing performance.

1 Introduction

Sub-linear indexing of appearance has been possible with the introduction of discriminative local features and descriptor vocabularies [33]. Despite the success of the bag of words model, spatial matching is still needed to boost performance, especially at large scale. *Geometry indexing* is still in its infancy, either being limited to weak constraints [17], or having high index space requirements [2]. A second stage of *spatial verification* and *geometry re-ranking* is the de facto solution of choice, where RANSAC approximations dominate. Re-ranking is linear in the number of images to match, hence its speed is crucial.

Address(es) of author(s) should be given

Exploiting local shape of features (*e.g.* local scale, orientation, or affine parameters) to extrapolate relative transformations, it is either possible to construct RANSAC hypotheses by single correspondences [27], or to see correspondences as Hough votes in a transformation space [23]. In the former case one still has to count inliers, so even with fine codebooks and almost one correspondence per feature, the process is quadratic in the number of (tentative) correspondences. In the latter, voting is linear in the number of correspondences but further verification with inlier count appears unavoidable.

Flexible spatial models are more typical in recognition; these are either not invariant to geometric transformations, or use pairwise constraints to detect inliers without any rigid motion model [21]. The latter are at least quadratic in the number of correspondences and their practical running time is still prohibitive if our target for re-ranking is thousands of matches per second.

We develop a relaxed spatial matching model, which, similarly to popular pyramid match approaches [13], distributes correspondences over a hierarchical partition of the transformation space. Using local feature shape to generate votes, it is *invariant* to similarity transformations, free of inlier-count verification and *linear* in the number of correspondences. It imposes *one-to-one* mapping and is *flexible*, allowing non-rigid motion and multiple matching surfaces or objects. This model is compatible with *soft assignment* of descriptors to multiple visual words, preserving one-to-one mapping and further increasing performance.

Fig. 1 compares our *Hough pyramid matching* (HPM) to *fast spatial matching* (FSM) [27]. Both the foreground object and the background are matched by HPM, following different motion models; inliers from one surface are only found by FSM. We show experimentally that flexible matching outperforms RANSAC-based approximations under any fixed model in terms of precision. But our major achieve-

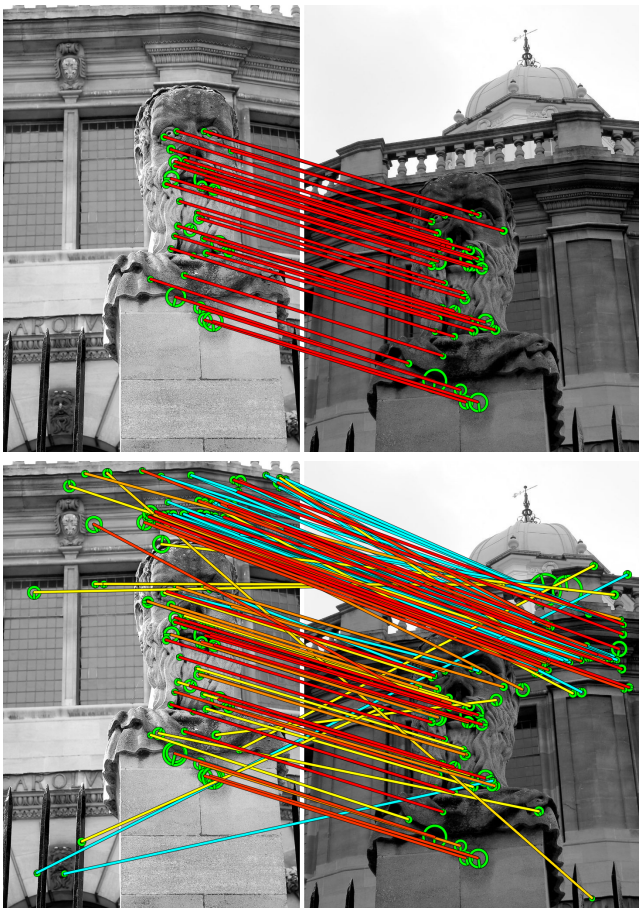

Fig. 1 Top: Inliers found by 4-dof FSM and affine-model LO-RANSAC for two images of *Oxford* dataset. Bottom: HPM matching, with all tentative correspondences shown. The ones in cyan have been erased. The rest are colored according to strength, with red (yellow) being the strongest (weakest).

ment is speed: in a given query time, HPM can re-rank one order of magnitude more images than the state of the art in geometry re-ranking. We give a more detailed account of our contribution in section 2 after discussing in some depth the most related prior work.

2 Related work

Given a number of correspondences between a pair of images, RANSAC [12], along with various approximations, is still one of the most popular spatial verification methods. It uses as evidence the count of inliers to a geometric model (*e.g*. homography, affine, similarity). Hypotheses are generated on random sets of correspondences, depending on model complexity. However, its performance is poor when the ratio of inliers is too low. Philbin *et al*. [27] generate hypotheses from *single correspondences* exploiting local feature shape. Matching then becomes deterministic by enumerating all hypotheses. Still, this process is quadratic in the number of correspondences.

Consistent groups of correspondences may first be found in the transformation space using the *generalized Hough transform* [3]. This is carried out by Lowe [23], but only as a prior step to verification. Tentative correspondences are found via fast nearest neighbor search in the descriptor space and used to generate votes in the transformation space. Correspondences are single, exploiting local shape as in [27]. Using a hash table, mapping to Hough bins is linear in the number of correspondences and performance depends on the number rather than the ratio of inliers. Still, multiple groups need to be verified for inliers and this may be quadratic in the worst case.

Leibe *et al*. [20] propose a probabilistic extension of the generalized Hough transform for object detection. In their voting scheme, observed visual words vote for object hypotheses based on their position relative to the object center. Votes in this case come from a number of training images, rather than a single matched image. Feature orientation is not taken into account so the method is not rotation invariant, but the principle is the same.

Jégou *et al*. [17] use a weaker geometric model whereby groups of correspondences only agree in their relative scale and—independently—orientation. Feature correspondences are found using a visual vocabulary. Scale and orientation of local features are quantized and stored in the inverted file. Hence, geometric constraints are integrated in the filtering stage of the search engine. However, because constraints are weak, this model does not dispense with geometry re-ranking after all.

In an attempt to capture information on the neighborhood of local features, MSER regions are employed to group features into bundles in [37]. Consistency between orderings of bundled features is used for geometric matching. However, the reference frame of the MSER regions is not used and features are projected on image axes instead, so rotation invariance is lost. The same weakness applies to Zhou *et al*. [39], who binarize spatial relations between pairs of local features. Similarly, Cao *et al*. [7] order features according to linear and circular projections, but rely on a training phase to learn the best spatial configurations.

Global feature geometry is integrated in the indexing process by Avrithis *et al*. [2]. A *feature map* is constructed for each feature, encoding positions of all other features in a local reference frame, similarly to shape context [5]. Re-ranking is still used, though it is much faster in this case. The additional space requirements for each feature map are reduced by the use of hashing, but it is clear that this approach does not scale well.

Closely related to our approach is the work of Zhang *et al*. [38], who set up a 2D Hough voting space based on relative displacements of corresponding features. Fixed size

bins incur quantization loss, while the model supports translation invariance only. Likewise, Shen *et al*. [32] apply several scale and rotation transformations to the query features and produce a 2D (translation) voting map for each database image. Queries are costly, both in terms of time and space needed for voting maps.

Whenever a single hypothesis may be generated by a single correspondence, Hough-based approaches are preferable to RANSAC-based ones, because they are nearly independent of the ratio of inliers and need to verify only a subset of hypotheses. However, both share the use of a fixed geometric model and need a parameter in verifying a hypothesis, *e.g*. bin size for Hough, or inlier threshold for RANSAC. Although there are attempts for parameter-free methods [29], an entirely different approach is to use *flexible* models, which is typical for recognition. In this case consensus is built among hypotheses only, operating on pairs of correspondences.

For instance, multiple groups of consistent correspondences are identified with the flexible, semi-local model of Carneiro and Jepson [8], employing pairwise relations between correspondences and allowing *non-rigid* deformations. Similarly, Leordeanu and Hebert [21] build a sparse adjacency (affinity) matrix of correspondences and greedily recover inliers based on its principal eigenvector. This spectral model can additionally incorporate different feature *mapping constraints* like one-to-one.

One-to-one mapping is maybe reminiscent of early correspondence methods on non-discriminative features, but can be very important when vocabularies are small, under the presence of repeating structures, or *e.g*. with *soft assignment* models [28]. Enqvist *et al*. [11] form a graph with correspondences as vertices and inconsistencies as edges. One-to-one mapping is easily incorporated by having multiple matches of a single feature form a clique on the graph. In contrast, we form a graph with features as vertices and correspondences as edges. Multiple matches then form a connected component of the graph.

Other solutions include for instance the early spectral approach by Scott and Longuet-Higgins [31], the quadratic programming approximation by Berg *et al*. [6], the context dependent kernel by Sahbi *et al*. [30], and the linear programming formulation by Jiang and Yu [18]. Most flexible models are iterative and at least quadratic in the number of correspondences. They are invariant to geometric transformations and resistant to ambiguous feature correspondences but not necessarily robust to outliers.

Relaxed matching processes like the one of Vedaldi and Soatto [36] offer an extremely attractive alternative in terms of complexity by employing distributions over hierarchical partitions instead of pairwise computations. The most popular is by Grauman and Darell [13], who map features to a histogram pyramid in the descriptor space, and then match them in a bottom-up process. The benefit comes mainly from approximating similarities by bin size. Lazebnik *et al*. [19] apply the same idea to image space but in such a way that geometric invariance is lost. Attempts to handle partial similarity usually resort to optimization methods like [22].

It should be noted that although a pyramid is a way to increase robustness over flat histograms, it is still approximate as correspondences may be lost at quantization boundaries, even at coarse levels. The effect has been analyzed in [14], where it has been shown that empirical distortion is substantially lower than the theoretical upper bound.

3 Contribution

While the above relaxed methods apply to *two* sets of features, we rather apply the same idea to *one* set of correspondences (feature pairs) and aim at grouping according to proximity, or affinity. This problem resembles *mode seeking* [9][35], but our solution is a non-iterative, bottom-up grouping process that is free of any scale parameter. We represent correspondences in the transformation space exploiting local feature shape as in [23], but we form correspondences using a vocabulary as in [27][17] rather than nearest neighbors in descriptor space. Like *pyramid match* [13], we approximate affinity by bin size, without actually enumerating correspondence pairs.

We impose a *one-to-one* mapping constraint such that each feature in one image is mapped to at most one feature in the other. Indeed, this makes our problem similar to that of [21], in the sense that we greedily select a pairwise compatible subset of correspondences that maximize a non-negative, symmetric affinity matrix. However we allow *multiple groups* (clusters) of correspondences. Contrary to [11][21], our voting model is non-iterative and linear in the number of correspondences.

To summarize our contribution, we derive a *flexible* spatial matching scheme whereby all tentative correspondences contribute, appropriately weighted, to a similarity score between two images. What is most remarkable is that *no verification*, model fitting or inlier count is needed as in [23], [27] or [8]. Besides significant performance gain, this yields a dramatic speed-up. Our result is a very simple algorithm that requires no learning and can be easily integrated into any image retrieval process.

Our *Hough pyramid matching* has been introduced in [34]. In this work, we extend our matching algorithm to account for *soft assignment* of visual words on the query side, providing an alternative solution to enforce one-to-one mapping. We further study the distribution of votes in the transformation space and derive a non-uniform space quantization scheme turning out to a considerable speed-up while leaving performance almost unaffected. We give more de-

tails on the matching processing itself, more examples, and a number of additional experiments and comparisons.

4 Problem formulation

In a nutshell, we are looking for one or more transformations that will make parts of one image align to parts of another. A number of transformation models is possible, but we choose to develop our method for similarity, *i.e*. a four parameter transformation consisting of translation, rotation and scale. Starting with our image representation, we formalize our goal as an optimization problem below.

We assume an image is represented by a set P of *local features*, and for each feature $p \in P$ we are given its descriptor, position and local shape. We restrict discussion to scale and rotation covariant features, so that the local shape and position of feature p are given by the 3×3 matrix

$$F(p) = \begin{bmatrix} M(p) & \mathbf{t}(p) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}, \tag{1}$$

where $M(p) = \sigma(p)R(p)$ and $\sigma(p), R(p), \mathbf{t}(p)$ stand for isotropic scale, orientation and position, respectively. $R(p)$ is an orthogonal 2×2 matrix with $\det R(p) = 1$, represented by an angle $\theta(p)$. In effect, $F(p)$ specifies a *similarity transformation* with respect to a normalized patch *e.g*. centered at the origin with scale $\sigma_0 = 1$ and orientation $\theta_0 = 0$.

Given two images P, Q, an *assignment* or *correspondence* $c = (p, q)$ is a pair of features $p \in P, q \in Q$. The relative transformation from p to q is again a similarity transformation given by

$$F(c) = F(q)F(p)^{-1} = \begin{bmatrix} M(c) & \mathbf{t}(c) \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}, \tag{2}$$

where $M(c) = \sigma(c)R(c)$, $\mathbf{t}(c) = \mathbf{t}(q) - M(c)\mathbf{t}(p)$; and $\sigma(c) = \sigma(q)/\sigma(p)$, $R(c) = R(q)R(p)^{-1}$ are the relative scale and orientation respectively from p to q. This is a 4-dof transformation represented by a parameter vector

$$f(c) = (x(c), y(c), \sigma(c), \theta(c)), \tag{3}$$

where $[x(c)\, y(c)]^{\mathrm{T}} = \mathbf{t}(c)$ and $\theta(c) = \theta(q) - \theta(p)$. Hence assignments can be seen as points in a d-dimensional transformation space $\mathcal{F}$; $d = 4$ in our case, while affine-covariant features would have $d = 6$.

An initial set C of candidate or *tentative* correspondences is constructed according to proximity of features in the descriptor space. There are different criteria, *e.g*. by nearest neighbor search given a suitable metric, or using a visual *vocabulary* (or *codebook*). Here we consider the simplest vocabulary approach where two features correspond when assigned to the same visual word:

$$C = \{(p, q) \in P \times Q : u(p) = u(q)\}, \tag{4}$$

where $u(p)$ is the *visual word* (or *codeword*) of p. This is a many-to-many mapping; each feature in P may have multiple assignments to features in Q, and vice versa. Given assignment $c = (p, q)$, we define its visual word $u(c)$ as the common visual word $u(p) = u(q)$.

Each correspondence $c = (p, q) \in C$ is given a weight $w(c)$ measuring its relative importance; we typically use the inverse document frequency (idf) of its visual word. Given a pair of assignments $c, c' \in C$, we assume an *affinity* score $\alpha(c, c')$ measures their similarity as a non-increasing function of their distance in the transformation space. Finally, we say that two assignments $c = (p, q)$, $c' = (p', q')$ are *compatible* if $p \neq p'$ and $q \neq q'$, and *conflicting* otherwise. For instance, c, c' are conflicting if they are mapping two features of P to the same feature of Q.

Our problem is then to identify a subset of pairwise compatible assignments that maximizes the sum of the weighted, pairwise affinity over all assignment pairs. This subset of C determines an *one-to-one* mapping between inlier features of P, Q, and the maximum value is the *similarity score* between P, Q. It can be easily shown that this is a binary quadratic programming problem [25] and we only target a very fast, approximate solution. In fact, we want to group assignments according to their affinity without actually enumerating pairs.

The *spectral matching* (SM) approach of [21] is an approximate solution where the binary constraint is relaxed and optimization is reduced to eigenvector computation. Note that being compatible does not exclude assignments from having low affinity. This is a departure of our solution from that of [21], as it allows *multiple groups* of assignments, corresponding to high-density regions in the transformation space. Spectral matching is a method we compare to in our experiments.

5 Hough Pyramid Matching

We assume that transformation parameters are normalized or non-linearly mapped to $[0, 1]$ (see section 7). Hence the transformation space is $\mathcal{F} = [0, 1]^d$. While we formulated our problem with $d = 4$, matching can apply to arbitrary transformation spaces (motion models) of any dimension (degrees of freedom).

We construct a hierarchical partition

$$\mathcal{B} = \{B_0, \ldots, B_{L-1}\} \tag{5}$$

of $\mathcal{F}$ into L levels. Each $B_\ell \in \mathcal{B}$ is a partition of $\mathcal{F}$ into 2^{kd} *bins* (hypercubes), where $k = L - 1 - \ell$. The bins are obtained by uniformly quantizing each transformation parameter, or partitioning each dimension into 2^k equal intervals of length 2^{-k}. B_0 is at the finest (bottom) level; B_{L-1} is at the coarsest (top) level and has a single bin. Each partition

B_ℓ is a refinement of $B_{\ell+1}$. Conversely, each bin of B_ℓ is the union of 2^d bins of $B_{\ell-1}$.

Starting with the set C of tentative correspondences of images P, Q, we distribute correspondences into bins with a histogram *pyramid*. Given a bin b, let

$$h(b) = \{c \in C : f(c) \in b\} \tag{6}$$

be the set of correspondences with parameter vectors falling into b, and $|h(b)|$ its count. We use this count to approximate affinities over bins in the hierarchy, making greedy decisions upon conflicts to compute a similarity score of P, Q. This computation is linear-time in the number of tentative correspondences $n = |C|$.

5.1 Matching process

We recursively split correspondences into bins in a *top-down* fashion, and then group them again recursively in a *bottom-up* fashion. We expect to find most groups of consistent correspondences at the finest (bottom) levels, but we do go all the way up the hierarchy to account for flexibility.

Large groups of correspondences formed at a fine level are more likely to be true, or *inliers*. Conversely, isolated correspondences or groups formed at a coarse level are expected to be false, or *outliers*. It follows that each correspondence should contribute to the similarity score according to the size of the groups it participates in and the level at which these groups are formed. We use the *count* of a bin to estimate a group size, and its *level* to estimate the pairwise affinity of correspondences within the group: indeed, bin sizes (hence distances within a bin) are increasing with level, hence affinity is decreasing.

In order to impose a *one-to-one* mapping constraint, we detect conflicting correspondences at each level and greedily choose the best one to keep on our way up the hierarchy. The remaining are marked as *erased*. Let X_ℓ denote the set of all erased correspondences up to level ℓ. If $b \in B_\ell$ is a bin at level ℓ, then the set of correspondences we have kept in b is $\hat{h}(b) = h(b) \setminus X_\ell$. Clearly, a single correspondence in a bin does not make a group, while each correspondence links to $m-1$ other correspondences in a group of m for $m > 1$. Hence we define the *group count* of bin b as

$$g(b) = [|\hat{h}(b)| - 1]_+, \tag{7}$$

where $[x]_+ = \max\{0, x\}$.

Now, let $b_0 \subseteq \ldots \subseteq b_\ell$ be the sequence of bins containing a correspondence c at successive levels up to level ℓ such that $b_k \in B_k$ for $k = 0, \ldots, \ell$. For each k, we approximate the affinity $\alpha(c, c')$ of c to any other correspondence $c' \in b_k$ by a fixed quantity. This quantity is assumed a non-negative, non-increasing *level affinity* function of k, say $\alpha(k)$. We focus here on the decreasing exponential form

$$\alpha(k) = 2^{-\lambda k}, \tag{8}$$

where λ controls the relative importance between successive levels, *i.e.* how *relaxed* the matching process is. For $\lambda = 1$, affinity is inversely proportional to bin size, which is in fact an upper bound on the actual distance between parameter vectors. For $\lambda > 1$, lower levels of the pyramid become more significant and the matching process becomes less flexible.

Observe that there are $g(b_k) - g(b_{k-1})$ new correspondences joining c in a group at level k. Similarly to standard pyramid match [13], this gives rise to the following *strength* of c up to level ℓ:

$$s_\ell(c) = g(b_0) + \sum_{k=1}^{\ell} \alpha(k)\{g(b_k) - g(b_{k-1})\}. \tag{9}$$

We are now in position to define the *similarity score* between images P, Q. Indeed, the *total strength* of correspondence c is simply its strength at the top level, $s(c) = s_{L-1}(c)$. Then, excluding all erased assignments $X = X_{L-1}$ and taking weights into account, we define the similarity score by the weighted sum

$$s(C) = \sum_{c \in C \setminus X} w(c)s(c). \tag{10}$$

On the other hand, we are also in position to choose the best correspondence in case of conflicts and impose one-to-one mapping. In particular, let $c = (p, q)$, $c' = (p', q')$ be two conflicting assignments. By definition (4), all four features p, p', q, q' share the same visual word, so c, c' are of equal weight: $w(c) = w(c')$. Now let $b \in B_\ell$ be the first (finest) bin in the hierarchy with $c, c' \in b$. It then follows from (9) and (10) that their contribution to the similarity score may only differ up to level $\ell - 1$. We therefore choose the strongest one up to level $\ell - 1$ according to (9). In case of equal strength, or at level 0, we pick one at random.

5.2 Examples and discussion

A toy 2D example of the matching process is illustrated in Figures 2, 3, 4. We assume that assignments are conflicting when they share the same visual word, as denoted by color. As shown in Fig. 2, three groups of assignments are formed at level 0: $\{c_1, c_2, c_3\}$, $\{c_4, c_5\}$ and $\{c_6, c_9\}$. The first two are then joined at level 1. Assignments c_7, c_8 are conflicting, and c_7 is erased at random. Assignments c_5, c_6 are also conflicting, but are only compared at level 2 where they share the same bin; according to (9), c_5 is stronger because it participates in a group of 5. Hence group $\{c_6, c_9\}$ is broken up,

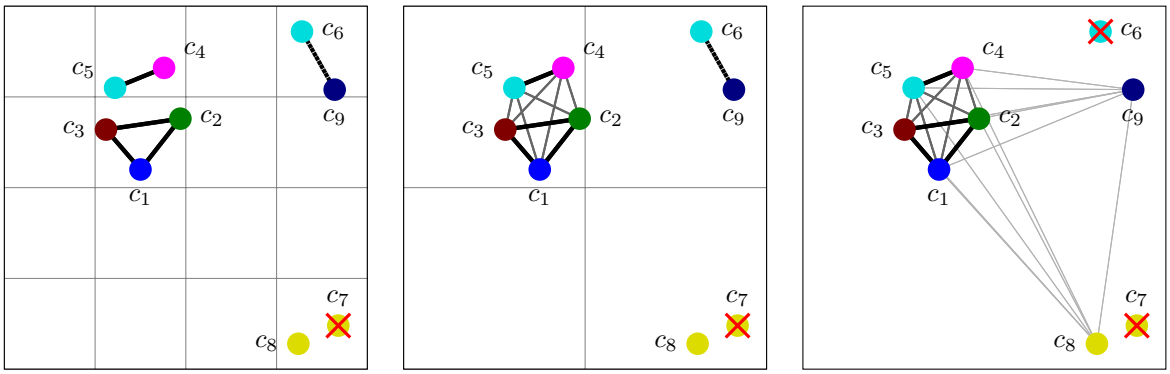

Fig. 2 Matching of nine assignments on a 3-level pyramid in 2D space. Colors denote visual words, and edge strength denotes affinity. The dotted line between c_6, c_9 denotes a group that is formed at level 0 and then broken up at level 2, since c_6 is erased.

	p	q	similarity score
c_1			$(2+\frac{1}{2}2+\frac{1}{4}2)w(c_1)$
c_2			$(2+\frac{1}{2}2+\frac{1}{4}2)w(c_2)$
c_3			$(2+\frac{1}{2}2+\frac{1}{4}2)w(c_3)$
c_4			$(1+\frac{1}{2}3+\frac{1}{4}2)w(c_4)$
c_5			$(1+\frac{1}{2}3+\frac{1}{4}2)w(c_5)$
c_6			0
c_7			0
c_8			$\frac{1}{4}6w(c_8)$
c_9			$\frac{1}{4}6w(c_9)$

Fig. 3 Assignment labels, features and scores referring to Fig. 2. Here vertices and edges denote features (in images P, Q) and assignments, respectively. Assignments c_5, c_6 are conflicting, being of the form $(p, q), (p, q')$. Similarly for c_7, c_8. Assignments $c_1, \dots, c_5$ join groups at level 0; c_8, c_9 at level 2; and c_6, c_7 are erased.

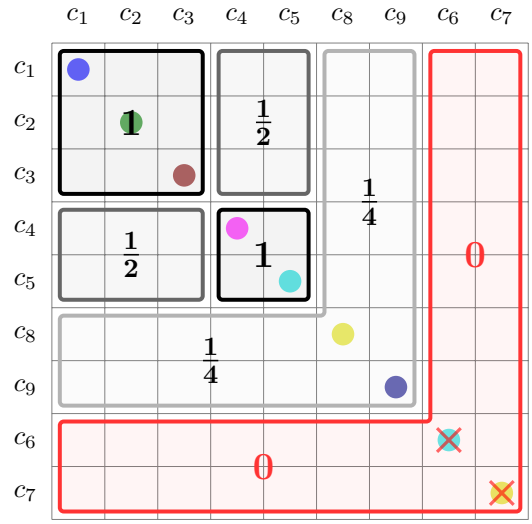

Fig. 4 Affinity matrix equivalent to the strengths of Fig. 3 according to (9). Assignments have been rearranged so that groups appear in contiguous blocks. Groups formed at levels $0, 1, 2$ are assigned affinity $1, \frac{1}{2}, \frac{1}{4}$ respectively. Assignments are placed on the diagonal, which is excluded from summation.

c_6 is erased and finally c_8, c_9 join $c_1, \dots, c_5$ in a group of 7 at level 2.

Apart from the feature/assignment configuration in images P, Q, Fig. 3 also illustrates how the similarity score of (10) is formed from individual assignment strengths, where we have assumed that $\lambda = 1$, so that $\alpha(k) = 2^{-k}$. For instance, assignments $c_1, \dots, c_5$ have strength contributions from all 3 levels, while c_8, c_9 only from level 2. Fig. 4 shows how these contributions are arranged in a (theoretical) $n \times n$ *affinity matrix* A. In fact, summing affinities over a row of A and multiplying by the corresponding assignment weight yields the assignment strength, as illustrated in Fig. 3—note though that the diagonal is excluded due to (7).

Finally, observe that the upper triangular part of A, responsible for half the similarity score of (10), corresponds to the set of edges among assignments shown in Fig. 2, the edge strength being proportional to affinity. This reveals the *pairwise* nature of the approach [8][21], including the fact that one assignment cannot form a group alone.

Another example is that of Fig. 1, where we match two real images of the same scene from different viewpoints. All tentative correspondences are shown, but colored according to the strength they have acquired through the matching process. There are a few mistakes, which is expected since HPM is really a fast approximation. However, it is clear that the strongest correspondences, contributing most to the similarity score, are true inliers. There may be no non-rigid mo-

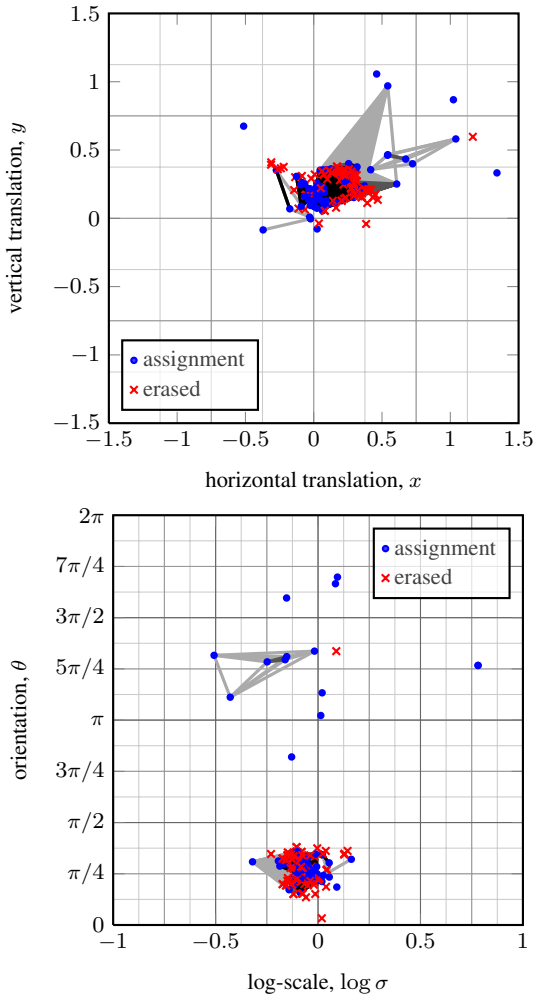

Fig. 5 Correspondences of the example in Fig. 1 as votes in 4D transformation space. Two 2D projections are depicted, separately for translation (x, y) (above) and log-scale / orientation $(\log \sigma, \theta)$ (below). Translation is normalized by maximum image dimension. Orientation is shifted by $5\pi/16$ (see section 7). There are $L = 5$ levels and we are zooming into the central 8×8 (16×16) bins above (below). Edges represent links between assignments that are grouped in levels $0, 1, 2$ only. Level affinity α is represented by three tones of gray with black corresponding to $\alpha(0) = 1$.

tion or relative motion of different objects like two cars, yet the 3D scene geometry is such that not even a homography can capture the motion of all visible surfaces. Indeed, the inliers to an affine model with RANSAC are only a small percentage of the ones shown here.

In analogy to the toy example of Fig. 2, Fig. 5 illustrates matching of assignments in the Hough space. Observe how assignments get stronger and contribute more by grouping according to proximity, which is a form of consensus. HPM takes no more than 0.6ms to match this particular pair of images, given the features, visual words, and tentative correspondences.

5.3 The algorithm

The matching process is outlined more formally in algorithm 1. It follows a recursive implementation: code before the recursive call of line 12 is associated to the *top-down* splitting process, while after that to *bottom-up* grouping. For brevity, variables or functions that are used in some block of code without definition or initialization are considered global. For instance weights $w(c)$ are global to HPM, strengths $s(c)$ are global to HPM-REC and ERASE, and so on *e.g.* for $\mathcal{B}$, L, X. On the other hand, C is redefined locally. In fact, $|C|$ is decreasing as we go down the hierarchy.

Algorithm 1 Hough Pyramid Matching

1: **procedure** HPM(assignments C, levels L)
2: $X \leftarrow \emptyset$; $\mathcal{B} \leftarrow$ PARTITION(L) ▷ erased; partition
3: **for all** $c \in C$ **do** $s(c) \leftarrow 0$ ▷ strengths
4: HPM-REC($C, L-1$) ▷ recurse at top
5: **return** score $\sum_{c \in C \setminus X} w(c)s(c)$ ▷ see (10)
6: **end procedure**
7:
8: **procedure** HPM-REC(assignments C, level ℓ)
9: **if** $|C| < 2 \vee \ell < 0$ **return**
10: **for all** $b \in B_\ell$ **do** $h(b) \leftarrow \emptyset$ ▷ histogram
11: **for all** $c \in C$ **do** $h(\beta_\ell(c)) \leftarrow h(\beta_\ell(c)) \cup c$ ▷ quantize
12: **for all** $b \in B_\ell$ **do** HPM-REC($h(b), \ell - 1$) ▷ recurse down
13: **for all** $b \in B_\ell$ **do**
14: $X \leftarrow X \cup$ ERASE($h(b)$)
15: $h(b) \leftarrow h(b) \setminus X$ ▷ exclude erased
16: **if** $|h(b)| < 2$ **continue** ▷ exclude isolated
17: **if** $\ell = L - 1$ **then** $a \leftarrow 1$ **else** $a \leftarrow 1 - 2^{-\lambda}$
18: **for all** $c \in h(b)$ **do** $s(c) \leftarrow s(c) + a2^{-\lambda\ell} g(b)$ ▷ (9)
19: **end for**
20: **end procedure**

Assignment of correspondences to bins is linear-time in $|C|$ in line 11, though equivalent to (6). B_ℓ partitions $\mathcal{F}$ for each level ℓ, so given a correspondence c there is a unique bin $b \in B_\ell$ such that $f(c) \in b$. We then define a constant-time mapping $\beta_\ell : c \mapsto b$ by uniformly quantizing parameter vector $f(c)$ at level ℓ. Storage in bins is sparse and linear-space in $|C|$; complete partitions B_ℓ are never really constructed.

Computation of strengths in lines 17-18 of algorithm 1 is equivalent to (9). In particular, substituting (8) into (9) and manipulating similarly to [13] yields

$$s(c) = g(b_0) + \sum_{k=1}^{L-1} 2^{-\lambda k}(g(b_k) - g(b_{k-1})) \tag{11}$$

$$= a \sum_{k=0}^{L-2} 2^{-\lambda k} g(b_k) + 2^{-\lambda(L-1)} g(b_{L-1}), \tag{12}$$

where $a = 1 - 2^{-\lambda}$.

Given a set of assignments in a bin, optimal detection of conflicts can be a hard problem. In function ERASE of algorithm 2, we follow a very simple approximation whereby

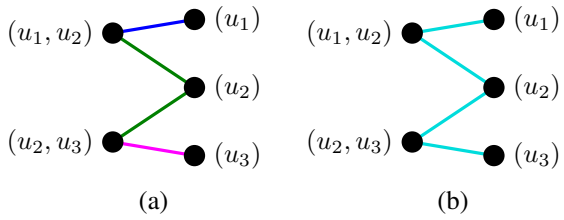

Fig. 6 Detection of conflicts based on (a) visual word classes and (b) component classes, under soft assignment. Vertices on the left (right) represent query (database) features. Labels denote assigned visual words, which are multiple for the query features. Same color denotes assignments in conflict.

two assignments are conflicting when they share the same visual word. This avoids storing features; and makes sense because with a fine vocabulary, features are uniquely mapped to visual words, *e.g.* 92% in our test sets—see section 8 for details.

Algorithm 2 Erase using visual word classes

1: **procedure** ERASE(assignments C)
2: $x \leftarrow \emptyset$; $U \leftarrow \emptyset$;
3: **for all** $c \in C$ **do** $U \leftarrow U \cup u(c)$ ▷ common visual words
4: **for all** $u \in U$ **do** $e(u) \leftarrow \emptyset$ ▷ visual word classes
5: **for all** $c \in C$ **do** $e(u(c)) \leftarrow e(u(c)) \cup c$
6: **for all** $u \in U$ **do** $x \leftarrow x \cup e(u) \setminus \arg\max_{c \in e(u)} s(c)$
7: **return** erased assignments x ▷ all but strongest
8: **end procedure**

For all assignments $h(b)$ of bin b we first construct the set U of common visual words. This is done in line 3, where $u(c)$ is the visual word assigned to c. Then, in lines 4-5, we define for each visual word $u \in U$ the *visual word class* $e(u)$, that is, the set of all assignments mapped to u. According to our assumption, all assignments in a class are (pairwise) conflicting. Therefore, we keep the strongest assignment in each class, erase the rest and update X.

It is clear that all operations in each recursive call on bin b are linear in $|h(b)|$. Since B_ℓ partitions $\mathcal{F}$ for all ℓ, the total operations per level are linear in $n = |C|$. Hence the time complexity of HPM is $O(nL)$.

Algorithm 3 Erase using connected component classes

1: **procedure** ERASE-CC(connected components Z)
2: $x \leftarrow \emptyset$;
3: **for all** $z \in Z$ **do** $e(z) \leftarrow E(z)$ ▷ component classes
4: **for all** $z \in Z$ **do** $x \leftarrow x \cup e(z) \setminus \arg\max_{c \in e(z)} s(c)$
5: **return** erased assignments x ▷ all but strongest
6: **end procedure**

6 Matching under soft assignment

The use of a vocabulary always incurs quantization loss. A common strategy for partially recovering from this loss is to assign descriptors to multiple visual words, in practice a small number of nearest neighbors in the vocabulary [28]. This *soft assignment* is preferably applied on the query image only, since inverted file memory requirements are left unaffected [17]. We explore here the use of soft assignment with HPM, following the latter choice both in the filtering and re-ranking phase.

In the case of hard assignment, the detection of conflicting assignments has been based on the observation that a high percentage of features are uniquely mapped to visual words. Unfortunately, this is not the case with soft assignment: the percentage of uniquely mapped features drops significantly, *e.g.* 85% (80%) for 3 (5) nearest neighbors in our test sets—see section 8 for details.

Figure 6(a) illustrates the detection of conflicts using visual word classes under soft assignment. The database image has three features in this example, assigned to three different visual words u_1, u_2, u_3 respectively, while the query image has two features soft-assigned to u_1, u_2 and u_2, u_3 respectively. There are three visual word classes in this case, and keeping one assignment from each class according to algorithm 2 leads to a one-to-many mapping, with one of the query features mapping to two database features.

We therefore introduce an alternative solution which can always preserve one-to-one mapping, without significant increase in complexity. Given two images P, Q, the union of features $V = P \cup Q$ and the set of assignments C can be seen as an undirected graph $G = (V, C)$, where each feature is a vertex and each assignment is an edge. In fact, the graph is bipartite, as each edge always joins a vertex of P to a vertex of Q.

Under this representation, we compute the set Z of *connected components* (maximally connected subgraphs) of G. In practice, using the union find algorithm with a disjoint-set data structure, this task is quasi-linear in the number of edges (assignments), hence in the number of vertices (features). The assignments of each component are used to define a *component class*. Component classes replace the visual word classes of algorithm 2. In particular, all assignments in the same component are considered pairwise conflicting, and only one is kept. Observe that isolated vertices are features participating in no assignments, implying that their components have no edges and are ignored.

Figure 6(b) illustrates conflicts using this scheme for the example of 6(a). There is only one connected component in this case, only one assignment is kept, and one-to-one mapping is preserved. This holds in general: all assignments of one feature always belong to the same component, so keeping one assignment from each component yields at most one

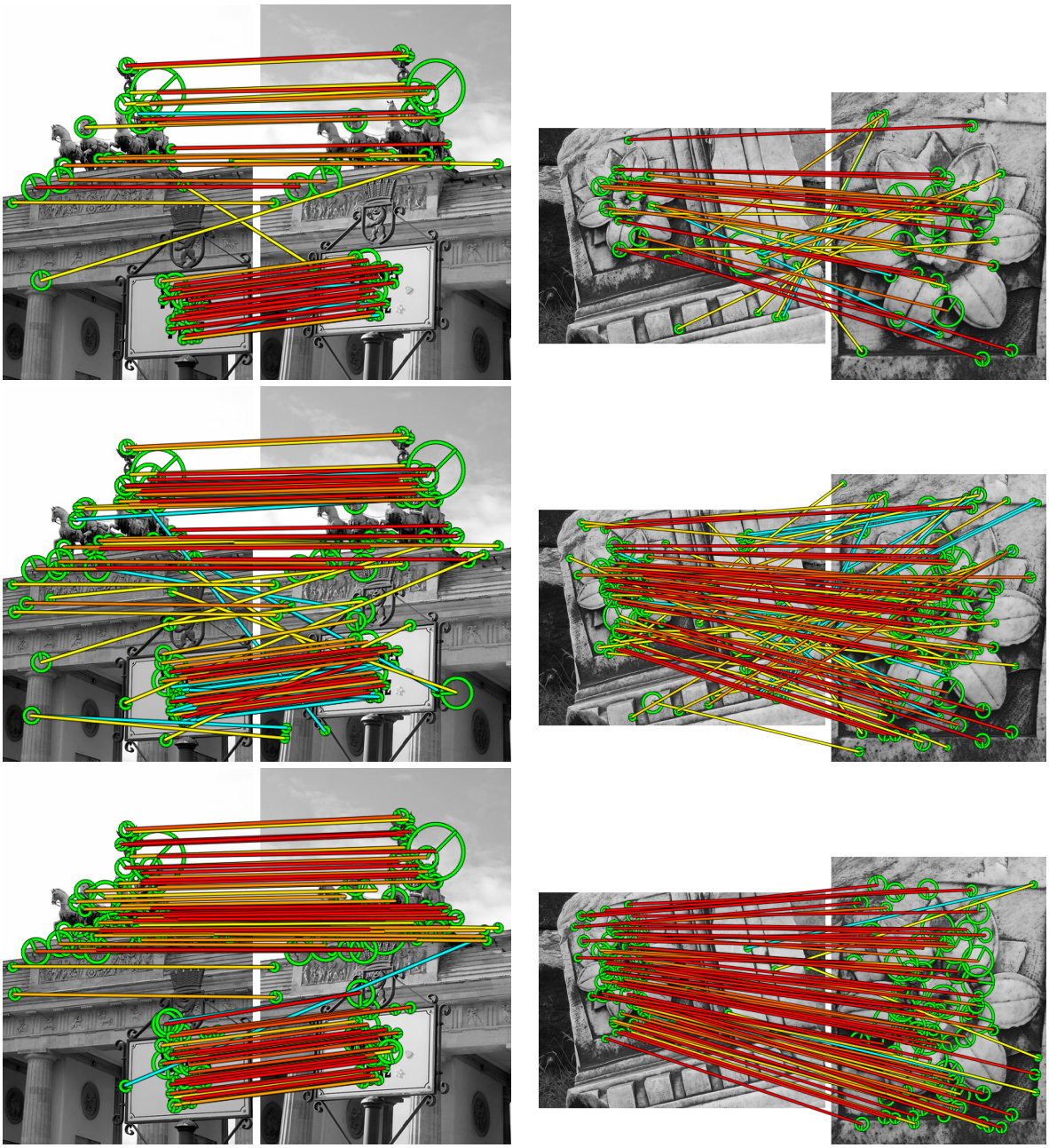

Fig. 7 Examples of HPM matching. (Top) hard assignment, visual word classes. (Middle) Soft assignment with 3 nearest neighbors, component classes. (Bottom) direct descriptor matching with ratio test [23], component classes. The color scheme is the same as in Fig. 1.

assignment per feature. Of course, the scheme of Fig. 6(b) is more strict than necessary; for instance, it misses one second assignment that may be valid. On the other hand, this is beneficial in reducing additional votes of background features due to soft assignment.

The modified function ERASE-CC is summarized in Algorithm 3, where visual word classes $e(u)$ are replaced by component classes $e(z)$ and $E(z)$ in line 3 stands for the set of edges of z (which is a graph itself). Both kinds of classes are treated as *equivalence* classes with one representative only selected from each. As in the example of 6, this scheme can be too strict, especially when the vocabulary is not fine enough. On the other hand, it is applicable even in the absence of a vocabulary, for instance when feature correspondences are determined by direct computations in the descriptor space.

Matching of two real images under different assignment and ERASE schemes is shown in Fig. 7. It is clear that HPM can work perfectly well when descriptors are not quantized and visual words are not available. In fact, this yields the

best matching quality. Hence HPM is not limited to image retrieval and can be applied to different matching scenaria, although finding correspondences from descriptors is much more expensive than matching itself. The use of a vocabulary with soft assignment yields fewer 'inliers' (assignments with significant strength contribution); hard assignment yields even less.

7 Implementation

7.1 Indexing and re-ranking

HPM turns out to be so fast in practice that we integrate it into a large scale image retrieval engine to perform geometric verification and re-ranking. We construct an inverted file structure indexed by visual word and for each feature in the database we store quantized location, scale, orientation and image id. Given a query, this information is sufficient to perform the initial, sub-linear filtering stage either by bag of words (BoW) [33] or weak geometric consistency (WGC) [17].

A number of top-ranking images are *marked* for re-ranking. For each query feature, we retrieve tentative assignments from the inverted file once more, but now only for marked images. For each assignment c found, we compute the parameter vector $f(c)$ of the relative transformation and store it in a collection indexed by marked image id. Given all assignments and parameter vectors, we match each marked image to the query using HPM. Finally, we normalize scores by marked image BoW ℓ_2 norm and re-rank.

7.2 Quantization

We treat each relative transformation parameter x, y, σ, θ of (3) separately. Translation $\mathbf{t}(c)$ in (2) refers to the coordinate frame of the query image, Q. If r is the maximum dimension of Q in pixels, we only keep assignments with horizontal and vertical translation $x, y \in [-3r, 3r]$. We also filter assignments such that $\sigma \in [1/\sigma_m, \sigma_m]$, where $\sigma_m = 10$ is above the range of any feature detector, so anything above that may be considered noise. We compute logarithmic scale, normalize all ranges to $[0, 1]$ and quantize parameters uniformly.

We also quantize local feature parameters of the *database* images: with $L = 5$ levels, each parameter is quantized into 16 bins. Our space requirements per feature, as summarized in Table 1, are then exactly the same as in [17]. On the other hand, *query* feature parameters are not quantized. It is therefore possible to have more than 5 levels in our histogram pyramid.

image id	x	y	$\log\sigma$	θ	total
16	4	4	4	4	32

Table 1 Inverted file memory usage per local feature, in bits. We use delta encoding for image id, so 2 bytes are sufficient.

7.3 Orientation prior

Because most images on the web are either portrait or landscape, previous methods use prior knowledge for relative orientation in their model [27][17]. We use the prior of WGC in our model by incorporating the weighting function of [17] in the form of additional weights in the sum of (10). Before quantizing, we also shift orientation by $5\pi/16$ because most relative orientations are near zero and we need them to group together early in the bottom-up process. In particular, this shift includes (a) $\pi/4$, so that the main mode of the distribution in $(-\pi/4, \pi/4]$ fits in one bin at pyramid level 2, plus (b) $\pi/16$, so that the peak around $\theta = 0$ fits in one bin at level 0.

8 Experiments

In this section we evaluate HPM against state of the art *fast spatial matching* (FSM) [27] in pairwise matching and in re-ranking in large scale search. In the latter case, we experiment on two filtering models, namely baseline *bag-of-words* (BoW) [33] and *weak geometric consistency* (WGC) [17].

8.1 Experimental setup

8.1.1 Datasets

We experiment on three publicly available datasets, namely *Oxford Buildings* [27] and *Paris* [28], *Holidays* [15] and on our own *World Cities* dataset[1]. *Oxford buildings* comprises a test set of 5K images and a distractor set of 100K images; the former is referred to as *Oxford* 5K or just *Oxford*, while the use of the latter is referred to as *Oxford* 105K. *World Cities* is downloaded from Flickr and consists of 927 annotated photos taken in Barcelona city center and 2 million images from 38 cities used as a distractor set. The annotated photos, referred to as *Barcelona* dataset, are divided into 17 groups, each depicting the same building or scene. We have selected 5 queries from each group, making a total of 85 queries for evaluation. We refer to *Oxford* 5K, *Paris*, *Holidays* and *Barcelona* as test sets. In contrast to *Oxford* 105K, the *World Cities* distractors set mostly depicts urban scenes exactly like the test sets, but from different cities.

[1] http://image.ntua.gr/iva/datasets/world_cities

8.1.2 Features and vocabularies

We extract SURF features and descriptors [4] from each image, setting strength threshold to 2.0 for the detector. We build vocabularies with *approximate k-means* (AKM) [27]. In most cases we use a *generic* vocabulary of size 100K constructed from a subset of the 2M distractors, which does not include the cities of the annotated test sets. This is close to the situation in real retrieval system. The imbalance factor [17] of this vocabulary is 1.22. However, for comparison purposes, we also employ *specific* vocabularies of different sizes constructed from the test sets. Unless otherwise stated, we use the generic vocabulary below. Our measurements of features uniquely mapped to visual words refer to the *Barcelona* dataset with the 100K generic vocabulary, where the average number of features per image is 594.

8.2 Matching experiment

Enumerating all possible image pairs of the *Barcelona* test set, there are $74,075$ pairs of images depicting the same building or scene. The similarity score should be high for those pairs and low for the remaining $785,254$; we therefore apply different thresholds to classify pairs as matching or non-matching, and compare to the ground truth. We match all possible pairs with 6-dof RANSAC, 8-dof RANSAC, 4-dof FSM (translation, scale, rotation), SM [21] and HPM.

We use pairs of correspondences to estimate 4-dof (similarity) transformations in our implementation of FSM. In the cases of RANSAC and FSM we perform a final stage of LO-RANSAC as in [27] to recover an affine (homography) transform, and use the sum of inlier idf values as a similarity score. On the other hand, SM and HPM do not need model fitting or inlier count. The similarity score of HPM is given by (10), with parameter λ (8) and number of levels L set according to our experiments in section 8.3.1. We do not use soft assignment in this experiment.

SM works on the affinity matrix A containing pairwise affinities between assignments. These are exactly the quantities that we approximate in HPM without enumerating them. In our implementation of SM, pairwise affinity is determined by proximity in the transformation space. In particular, parameters of relative transformations are initially non-linearly mapped to $[0,1]$, as described in section 8.3. Then, given two assignments c and c' with normalized parameter vectors $f(c), f(c')$ respectively, their affinity is computed as

$$\alpha(c,c') = \begin{cases} \frac{w(c)+w(c')}{||f(c)-f(c')||_2}, & \text{if} ||f(c)-f(c')||_2 < \tau \\ 0, & \text{otherwise,} \end{cases} \quad (13)$$

where parameter τ is set to 0.15 in practice. The similarity score of SM is the sum over all affinities of the submatrix of A chosen by the optimization process of [21].

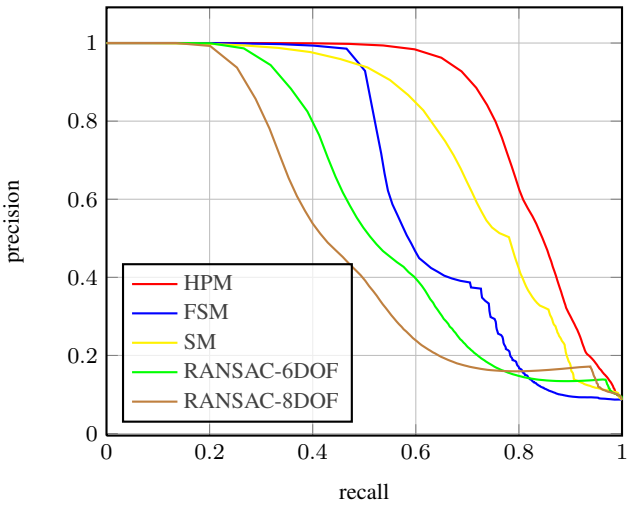

Fig. 8 Precision-recall curves over all image pairs of *Barcelona* test set with no distractors.

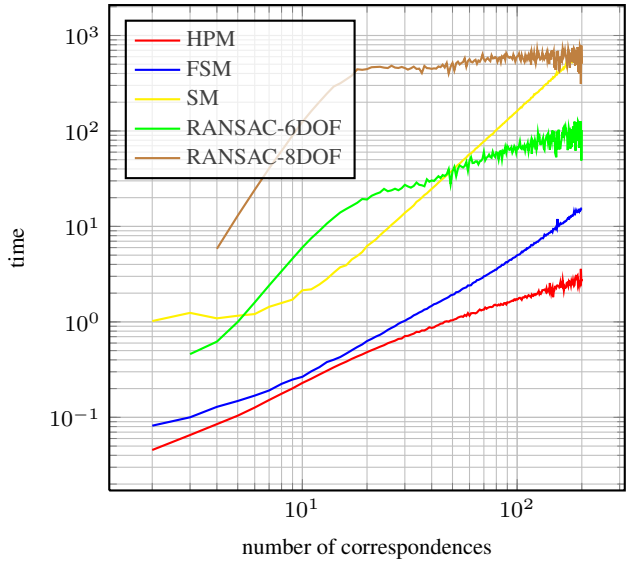

Fig. 9 Matching time versus number of correspondences over all image pairs of *Barcelona* test set with no distractors.

Given the above settings, we rank pairs according to score and construct the *precision-recall* curves of Fig. 8. On the other hand, Fig. 9 compares matching time versus number of correspondences over all tested image pairs. We use our own C++ implementations for all algorithms. Times are measured on a 2 GHz QuadCore processor, but our implementations are single-threaded.

HPM clearly outperforms all methods in terms of precision-recall performance, at the same time being linear in the number of correspondences and faster than all other methods. It is remarkable that this is achieved in a 4-dof transformation space only. This is attributed to the fact that its relaxed matching process does not assume any fixed model.

λ	0.8	1.0	1.2	1.4	1.6	1.8	2.0
map	0.532	0.546	0.553	0.556	0.557	**0.558**	0.557

Table 2 mAP for varying λ on *Barcelona* test set with 2M *World Cities* distractors. Filtering is performed with BoW and re-ranking on the top 1K images.

L	2	3	4	5	6
pyramid	0.473	0.498	0.536	0.556	**0.559**
flat	0.448	0.485	0.524	**0.534**	0.509

Table 3 mAP for pyramid and flat matching at different levels L on *Barcelona* with 2M *World Cities* distractors. Filtering is performed with BoW and re-ranking on the top 1K images.

SM is second in performance, but is quite slow, while FSM is the second choice in terms of speed. We adaptively estimate the inlier ratio and the target number of trials for RANSAC, but we also enforce a limit of 1000 on the number of trials, which explains the saturation effect in Fig. 9. Both 6-dof and 8-dof RANSAC are outperformed by other methods, while 8-dof RANSAC is the slowest.

8.3 Re-ranking

We experiment on retrieval using BoW and WGC with ℓ_2 normalization for filtering. Both are combined with HPM and 4-dof FSM for geometry re-ranking. We measure performance via *mean Average Precision* (mAP). We also compare re-ranking times and total query times, including filtering. All times are measured on a 2GHz QuadCore processor with our own C++ single-threaded implementations. We do not follow any early aborting scheme as in [27].

8.3.1 Tuning

Parameter λ. To quantify the effect of the level affinity parameter λ (8), we measure mAP on the *Barcelona* test set with 2M distractors, using BoW for filtering and re-ranking the top 1000 images using HPM with $L = 5$ levels. The latter choice is justified below. Table 2 presents mAP performance for varying λ. The performance is maximized for $\lambda = 1.8$, boosting the contribution of lower levels. Observe however that the effect of the parameter above $\lambda = 1.2$ is not significant.

Levels. Quantizing local feature parameters at 6 levels in the inverted file, we measure HPM performance versus pyramid levels L, as shown in Table 3. We also perform re-ranking on the single finest level of the pyramid for each L. We refer to the latter as *flat* matching; this is still different than [23] in the sense that it enforces one-to-one matching under a fine vocabulary and aggregates votes over the entire transformation space according to (9), even if $L = 1$.

Observe that the benefit of HPM in going from 5 to 6 levels is small, while flat matching actually drops in performance. Our choice for $L = 5$ then makes sense, apart from saving space—see section 7. For the same experiment, mAP is 0.341 and 0.497 for BoW and BoW+FSM respectively. It is thus interesting to observe that even the flat scheme yields considerable improvement. This is due to the flexibility of the model and its ability to handle multiple groups, which are a departure from [23]. Relaxed matching with the Hough pyramid further improves performance. Suggestively we report that re-ranking time for HPM with 3, 4 and 5 levels takes 391ms, 559ms and 664ms respectively, while flat matching takes around 230ms.

8.3.2 Results

Retrieval examples. Real retrieval examples along with comparisons for one query are available in our research page online[2]. In particular, for the same query image, positive images appearing within the top 20 retrieved images are 1, 8 and 11 for BoW, BoW+FSM and BoW+HPM respectively, with query time being respectively 282ms, 5054ms and 976ms. Most impressive is the fact that when re-ranking 10K images with HPM, all top 20 images are positive.

Distractors. Fig. 10 compares HPM to FSM and baseline, for a varying number of distractors up to 2M. Both BoW and WGC are used for the filtering stage and as baseline. HPM turns out to outperform FSM in all cases. We also re-rank 10K images with HPM, since this takes less time than 1K with FSM. This yields the best performance, especially in the presence of distractors. Interestingly, filtering with BoW or WGC makes no difference in this case. In Table 4 we summarize results for the same experiment with *orientation priors* for WGC and HPM. When these are used together, prior is applied to both. Again, BoW and WGC are almost identical in the HPM10K case. Using a prior increases performance in general, but this is dataset dependent. The side effect is limited rotation invariance.

Timing. Varying the number of re-ranked images, we measure mAP and query time for FSM and HPM. Once more, we consider both BoW and WGC for filtering. A combined plot is given in Fig. 11. HPM appears to re-rank *ten times* more images in less time than FSM. With BoW, its mAP is 10% higher than FSM for the same re-ranking time, on average. At the price of 7 additional seconds for filtering, FSM eventually benefits from WGC, while HPM is clearly unaffected. Indeed, after about 3.3 seconds, mAP performance of BoW+HPM reaches *saturation* after re-ranking 7K images, while WGC does not appear to help.

Specific vocabularies. Table 5 summarizes performance on the *Oxford* dataset for specific vocabularies of varying

[2] http://image.ntua.gr/iva/research/relaxed_spatial_matching/

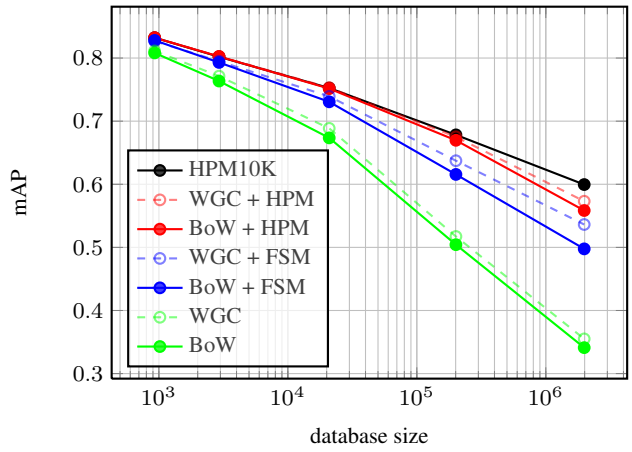

Fig. 10 mAP comparison for varying database size on *Barcelona* with up to 2M *World Cities* distractors. Filtering is performed with BoW or WGC and re-ranking on the top 1K images with FSM or HPM, except for HPM10K where BoW and WGC curves coincide.

method	no distractors		2M distractors	
	no prior	prior	no prior	prior
WGC+HPM10K	–	–	0.599	0.612
BoW+HPM10K	–	–	**0.601**	**0.613**
WGC+HPM	**0.832**	**0.851**	**0.573**	**0.599**
BoW+HPM	0.832	0.837	0.558	0.565
WGC+FSM	0.826	0.846	0.536	0.572
BoW+FSM	0.827	–	0.497	–
WGC	0.811	0.843	0.355	0.447
BoW	0.808	–	0.341	–

Table 4 mAP comparison on *Barcelona* with and without 2M *World Cities* distractors, with and without prior. Re-ranking on top 1K images, except for HPM10K.

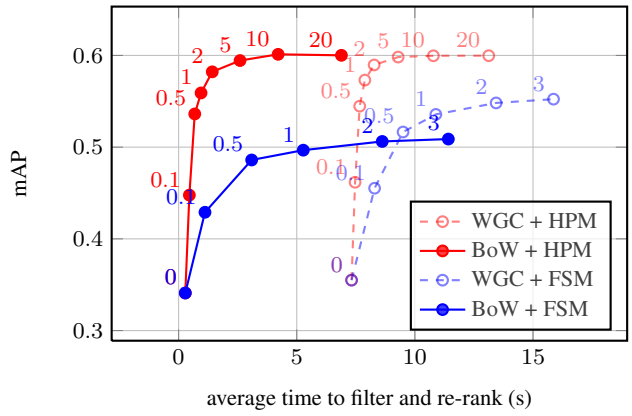

Fig. 11 mAP and total (filtering + re-ranking) query time for a varying number of re-ranked images. The latter are shown with text labels near markers, in thousands. Results on *Barcelona* with 2M *World Cities* distractors.

method	vocabulary size			
	100K	200K	500K	700K
BoW+HPM+P	0.640	0.683	0.701	0.690
BoW+HPM	0.622	**0.669**	**0.692**	**0.686**
BoW+FSM	**0.631**	0.642	0.677	0.653
BoW	0.545	0.575	0.619	0.614

Table 5 mAP comparison on *Oxford* dataset for specific vocabularies of varying size, without distractors. Filtering with BoW and re-ranking top 1K images with FSM and HPM. P = prior.

method	*Oxford*		*Paris*	
	0	2M	0	2M
BoW+HPM10K+P	–	**0.418**	–	**0.419**
BoW+HPM10K	–	0.403	–	0.418
BoW+HPM+P	0.546	0.381	0.595	0.402
BoW+HPM	**0.522**	**0.372**	**0.581**	**0.397**
BoW+FSM	0.503	0.317	0.542	0.336
BoW	0.430	0.201	0.539	0.282

Table 6 mAP comparison on *Oxford* and *Paris* datasets with 100K generic vocabulary, with and without 2M distractors. Filtering performed with BoW only. Re-ranking 1K images with FSM and HPM, as well as 10K with HPM. P = prior.

size, created from all *Oxford* images. HPM again has superior performance in all cases except for the 100K vocabulary. Our best score without prior (0.692) can also be compared to the best score (0.664) achieved by 5-dof FSM and specific vocabulary in [27], though the latter uses a 1M vocabulary and different features. The higher scores achieved in Perdoch *et al.* [26] are also attributed to superior features rather than the matching process.

More datasets. Switching back to our generic vocabulary, we perform large scale experiments on *Oxford* and *Paris* test sets and present results in Table 6. We consider both *good* and *ok* images as positive examples. We have shown the capacity of HPM to be one order of magnitude higher than that of FSM, so it makes sense again to also re-rank up to 10K images with HPM. Furthermore, focusing on practical query times, we limit filtering to BoW. HPM clearly outperforms FSM, while re-ranking 10K images significantly increases the performance gap at large scale. Our best score without prior on *Oxford* (0.522) can be compared to the best score (0.460) achieved by FSM in [28] with a 1M generic vocabulary created on the *Paris* dataset.

Comparison to other models. In order to be more comparable to previously published methods, we conduct an experiment using the modified version of Hessian-Affine detector of [26], where the gravity vector assumption is used to estimate the dominant orientation of features for descriptor extraction. Similarly to [26] and [32], we switch off rotation for spatial matching and perform our voting scheme in a pyramid of 3 dimensions. We use a 1M specific vocabulary trained on all images of *Oxford* 5K, when we test on

method	Ox 5K	Ox 105K	Paris	Holidays
HPM (this work)	**0.789**	**0.730**	0.725	**0.790**
Shen *et al*. [32]	0.752	0.729	**0.741**	0.762
Zhang *et al*. [38]	0.696	-	-	-
[38]+RANSAC	0.713	-	-	-
Cao *et al*. [7]	0.656	-	0.632	-
[7]+RANSAC	0.661	-	-	-
Perdoch *et al*. [26]	**0.789**	0.726	-	0.715
FSM [27]	0.647	0.541	-	-

Table 7 mAP comparison to other spatial models on *Oxford* 5K and *Oxford* 105K test set. Both for our method and all other methods a specific vobulary of size 1M is created from images of *Oxford* 5K.

method	strength	mAP
BoW only	-	0.341
out-of-bounds removed	1.0	0.373
one-to-one only	1.0	0.503
HPM	as in (9)	0.558

Table 8 mAP comparison on *Barcelona* with 2M *World Cities* distractors, illustrating the incremental effect of enforcing voting space bounds, enforcing one-to-one mapping, and using level-dependent correspondence strengths. The latter is exactly HPM.

Oxford 5K or *Oxford* 105K. Similarly, we use a 1M specific vocabulary for *Paris*. We further conduct experiments on *Holidays* dataset. Since this includes rotated images, the gravity vector assumption does not hold and we switch back to the use of SURF features as in our previous experiments. A specific vocabulary of 200K visual words is used in this case as in [32].

Table 7 presents mAP performance for a number of spatial matching or indexing models on the *Oxford* 5K, *Oxford* 105K, *Paris* and *Holidays* test sets. HPM achieves state of the art performance on all datasets, in fact outperforming all methods except for *Paris*, where it is outperformed by Shen *et al*. [32], and *Oxford* 5K, where Perdoch *et al*. [26] achieve the same performance. Our query time for re-ranking 1000 images is 210ms on a single threaded implementation, while the one reported in [26] is 238ms on 4 cores. This time of HPM is without the early aborting scheme of [27] or [26]. The reported query time in [32] is 89ms on *Oxford* 5K. Both time and space per query are (in the worst case) linear in the dataset size for this method, as voting maps are allocated for all images having common visual words with the query.

Non-uniform quantization. Relative transformation votes are not uniformly distributed in Hough space. Apart from orientation where we use a prior, this is true also for translation and scale. For instance, the distribution of relative log-scale and x parameter of translation appears experimentally close to the Laplacian distribution, as shown in Fig. 12 and 13 respectively. If we quantize Hough space uniformly, bins will not be equally populated and votes in sparse areas will not form groups as easily as in dense ones; and when they do so, their affinity will be lower.

We therefore investigate normalizing distributions, *i.e*., non-linear mapping of relative transformation parameters prior to quantization, so that vote distributions become uniform. In particular, we model translation x, y and log-scale $\log\sigma$ by Laplacian distributions, estimate their parameters via maximum likelihood on experimental data and use the learned CDFs to non-linearly map x, y and $\log\sigma$ to $[0,1]$. We discard assignments outside interval $[0.05, 0.95]$, that is assignments exhibiting extreme displacement or scale change compared to the population of our dataset. Finally, we uniformly quantize each parameter, which is equivalent to non-uniform quantization in the original voting space.

We report results only for the combination of translation and scale normalization, since we have seen that there is no apparent benefit in other cases (*e.g*., when each parameter is alone. Rotation θ of the relative transformation follows a similar distribution to the one shown in [17], which we do not normalize; we rather use the orientation prior in this case.

Fig. 14 shows mAP as measured on *Barcelona* test set for uniform and non-uniform quantization. Quite unexpectedly, non-uniform quantization slightly reduces performance but it also accelerates matching considerably. Votes in sparse areas appear to increase for distractor images as well, and this may explain why mAP is not improved. On the other hand, matching is faster because there are more single votes in lower levels of the pyramid, and single votes do not form groups. Non-uniform quantization is therefore a good choice for further speed-up. However, we still use uniform quantization in the remaining experiments, seeking maximum performance.

Effect of one-to-one-mapping. Our similarity score is a weighted sum over all correspondences between two images. Correspondences with transformations falling out of bounds (out of the HPM voting space) are discarded, as described in Section 7. Results of Table 8 show that, by just removing such correspondences, there is a small improvement in performance. In this case, set X of (10) includes only out-of-bounds correspondences, while strength $w(c)$ is set equal to 1.0. We further enforce one-to-one mapping with our erase procedure, but still keep strengths equal to 1.0. One-to-one mapping appears to impressively improve performance since many false matching correspondences are not accounted in the similarity score. Finally, using strengths as provided by HPM (9) there is further significant improvement.

8.4 Re-ranking with soft assignment

We experiment on retrieval using soft assignment for visual words [28] on the query side only [17]. We perform large

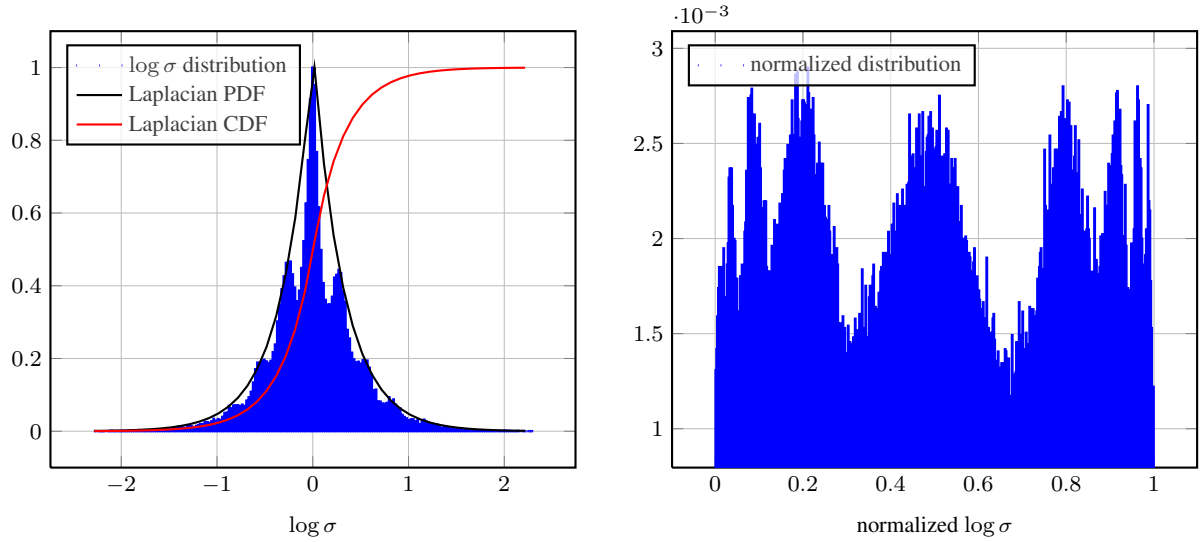

Fig. 12 (Left) Distribution of relative log-scale over all pairs of images of the *Barcelona* test set. Maximum likelihood estimation of Laplacian distribution yields location parameter $\mu = 0.003$ and scale parameter $\gamma = 0.323$. (Right) Normalized distribution after non-linearly mapping $\log \sigma$ to $[0, 1]$ via the Laplacian CDF.

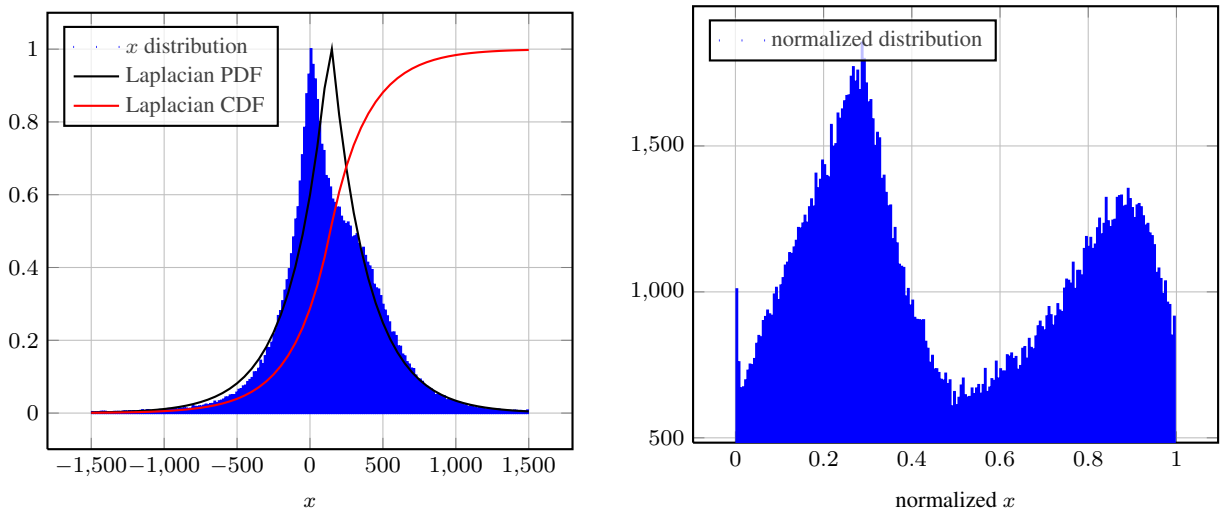

Fig. 13 (Left) Distribution of relative x over all pairs of images of the *Barcelona* test set. Maximum likelihood estimation of Laplacian distribution yields location parameter $\mu = 139$ and scale parameter $\gamma = 251$. (Right) Normalized distribution after non-linearly mapping x to $[0, 1]$ via the Laplacian CDF.

scale experiments and compare HPM to FSM using soft assignment with both. We use BoW for filtering in all remaining experiments.

Conflict detection. We compare the two methods of conflict detection, namely based on visual word classes (ERASE) and component classes (ERASE-CC). The results are shown in Fig. 15 versus number of nearest neighbors used in soft assignment. Component classes seem to outperform visual word classes in all cases, despite the fact that correct assignments may be discarded. In fact, the performance of the latter drops eventually, which is attributed to one-to-one mapping not being preserved. In the remaining soft assignment experiments we only use component classes for conflict detection.

Nearest neighbors. We compare HPM to FSM in terms of mAP performance and re-ranking time per query versus number of nearest neighbors, as shown in Fig. 16. The two methods appear to converge to the same mAP as nearest neighbors increase when re-ranking 1K images. However, HPM is clearly superior when re-ranking 10K images, even more so with orientation prior. HPM remains roughly one order of magnitude faster and this is much more significant under of soft assignment than in the baseline, because re-ranking time for FSM exceeds reasonable query

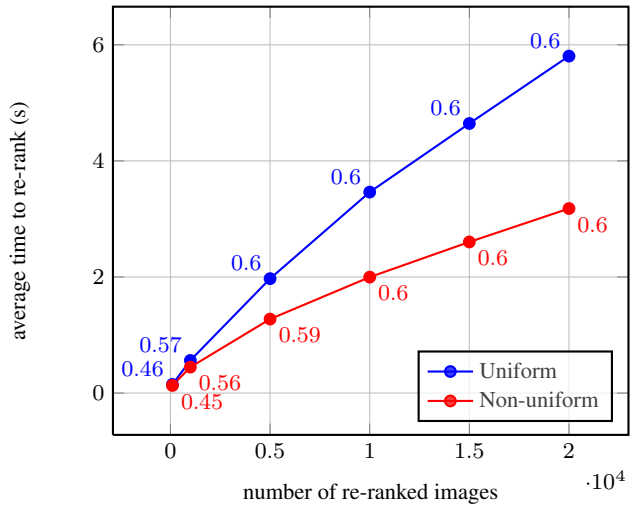

Fig. 14 Average re-ranking time versus number of re-ranked images for uniform and non-uniform quantization. mAP is shown with text labels near markers. Results on *Barcelona* test set with 2M *World Cities* distractors. Filtering with BoW.

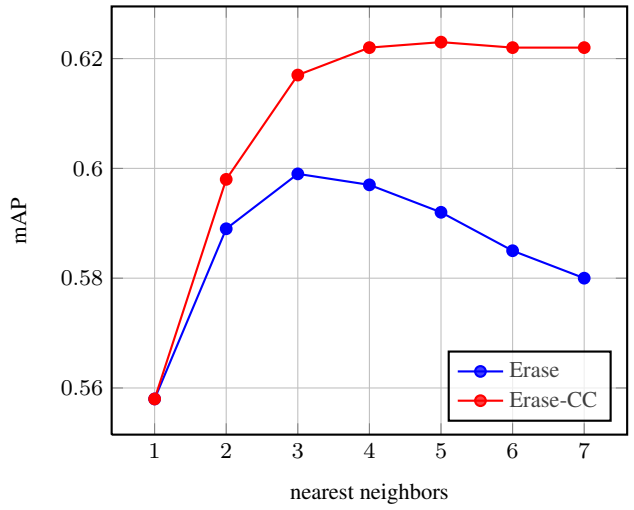

Fig. 15 Comparison of HPM conflicting assignment detection based on visual word classes (ERASE) and component classes (ERASE-CC). mAP is given versus number of nearest neighbors in soft assignment, as measured on *Barcelona* test set with 2M *World Cities* distractors. Filtering with BoW and re-ranking top 1K images.

times above 3 nearest neighbors. The number of tentative correspondences is nearly linear in the number of nearest neighbors when soft assignment is performed on one side only [28], so it is confirmed once again that HPM is linear in the number of correspondences.

Distractors, timing. Similarly to the hard assignment case, we compare HPM and FSM for a varying number of distractors up to 2M. mAP is shown in Fig. 17 where we apply soft assignment with 3 nearest neighbors for all methods. Again, the benefit of HPM over FSM is higher when re-ranking 10K images especially with prior, in which case the gain is nearly

NN	500K			700K		
	FSM	HPM	HPM+P	FSM	HPM	HPM+P
1	0.677	0.692	0.701	0.653	0.686	0.690
2	0.699	0.714	0.723	0.692	0.715	0.719
3	0.707	0.716	0.724	0.701	0.721	0.724
4	0.709	0.716	**0.726**	0.711	0.726	**0.730**
5	0.716	0.719	0.724	0.716	0.726	0.729
6	0.712	0.713	0.724	0.718	0.725	0.729

Table 9 mAP comparison on *Oxford* test set for specific vocabularies, without distractors. Filtering with BoW and re-ranking on the top 1K images with soft assignment using a varying number of nearest neighbors.

method	*Oxford*	*Paris*
BoW+HPM10K+SA+P	**0.461**	**0.456**
BoW+HPM10K+SA	0.445	0.434
BoW+HPM+SA+P	**0.426**	**0.427**
BoW+HPM+SA	0.414	0.414
BoW+FSM+SA	0.391	0.389
BoW+SA	0.251	0.316

Table 10 mAP comparison on *Oxford* and *Paris* datasets with 100K generic vocabulary, with 2M distractors. Filtering with BoW and re-ranking on the top 1K images with FSM and HPM, also 10K with HPM. Using soft assignment with 3 nearest neighbors.

10%. mAP versus re-ranking time for a varying number of re-ranked images is shown in Fig. 18. Similarly to hard assignment, HPM can re-rank one order of magnitude more images than FSM in the same amount of time with 10% higher mAP. In general, all methods benefit by 7-13% by the use of soft assignment compared to baseline BoW, but query times become unrealistic for FSM.

Specific vocabularies. Table 9 summarizes mAP performance for specific vocabularies on *Oxford* test set. We use the same vocabularies as in the hard assignment experiments. Our best score achieved on *Oxford* with hard assignment (0.701) now increases to 0.730. However, without distractors, the gain of HPM over FSM is not as high as in the hard assignment case.

More datasets. Finally, large scale soft assignment experiments with a generic vocabulary and 2M distractors on *Oxford* and *Paris* test sets are summarized in Table 10. This time the gain of HPM over FSM in mAP is roughly 5%, which becomes 7% with the prior.

9 Discussion

Clearly, apart from geometry, there are many other ways in which one may improve the performance of image retrieval. For instance, *query expansion* [10] increases recall of popular content, though it takes more time to query. The latter can be avoided by offline clustering and *scene map* construction [1], also yielding space savings. Methods related to visual word quantization like *soft assignment* [28] or *hamming*

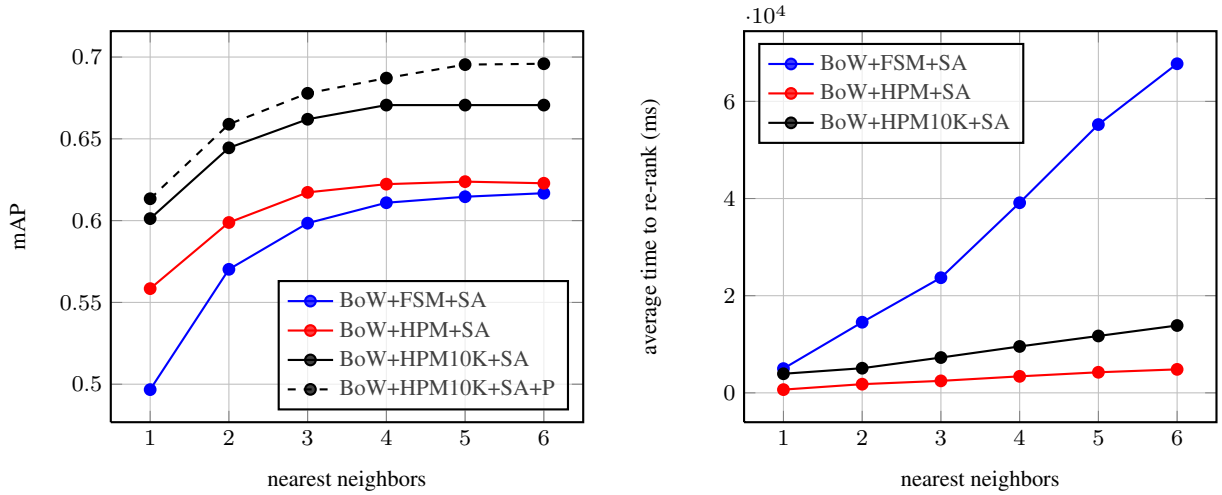

Fig. 16 mAP (left) and average re-ranking time per query (right) versus number of nearest neighbors in soft assignment, measured on *Barcelona* test set with 2M *World Cities* distractors. SA = soft assignment, P = prior.

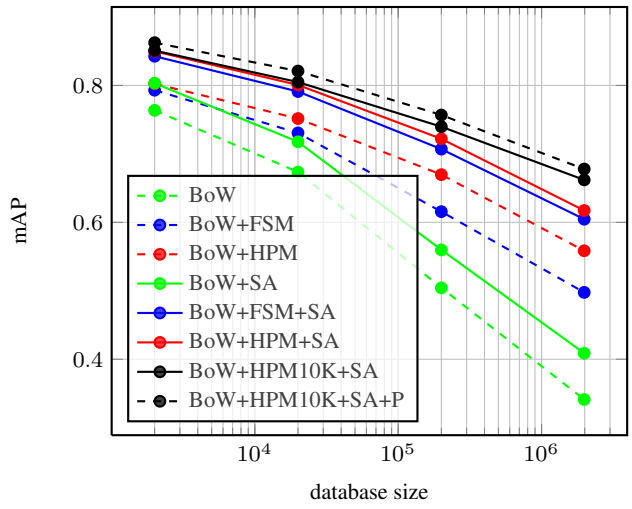

Fig. 17 mAP comparison versus database size on *Barcelona* with up to 2M *World Cities* distractors. SA = soft assignment, with 3 nearest neighbors for all methods.

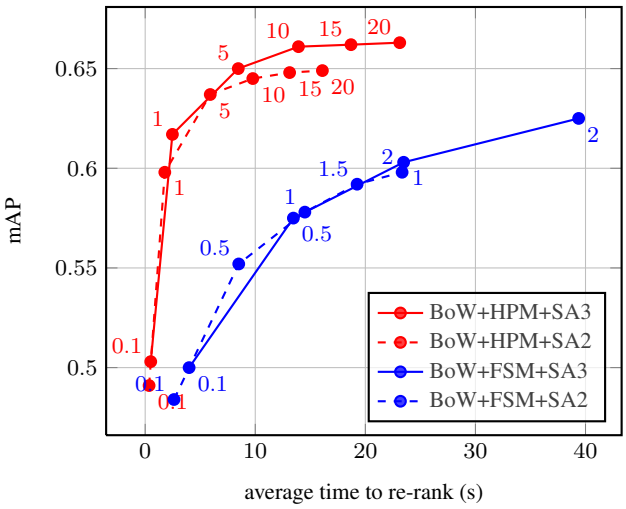

Fig. 18 mAP versus re-ranking time for a varying number of re-ranked images. The latter are shown with text labels near markers, in thousands. Results on *Barcelona* with 2M *World Cities* distractors, using soft assignment with 2 and 3 nearest neighbors for both methods.

embedding [17] also increase recall, at the expense of query time and index space. Vocabulary *learning* [24] transfers this problem to the vocabulary itself, given large amounts of indexed images. Other *priors* like the gravity vector [26] also help, usually at the cost of some invariance loss.

Experiments in the literature have shown that the effect of such methods is additive. Comparing to our prior work [34], we have investigated here the case of soft assignment and we have confirmed this finding. In fact, integrating soft assignment has been maybe the most interesting among other methods because it is a non-trivial problem and it has a considerable impact on query times in general. Improving or learning a vocabulary, query expansion or other priors are straightforward to integrate with HPM and expected to yield further gain.

We have developed a very simple spatial matching algorithm that requires no learning and can be easily implemented and integrated in any image retrieval engine. It boosts performance by allowing flexible matching and matching of multiple objects or surfaces. Matching is not as parameter-dependent as in other methods: λ is a relative quantity and there is no such thing as absolute scale, fixed bin size, or threshold parameter. Ranges and relative importance of transformation parameters are fixed so that they apply to most practical cases, while fitting the voting grid to true distributions does not influence performance much.

By dispensing with the need to count inliers to geometric model hypotheses, HPM also yields a dramatic speed-up in comparison to RANSAC-based methods. It is arguably the first time a geometry re-ranking method is shown to reach saturation in as few as three seconds, which is a practical query time. The practice so far has been to stop re-ranking at a point such that queries do not take too long, without studying further potential improvement using graphs like those in Figure 11.

One limitation of HPM that is shared with *e.g.* [23], [27], [26] is that matching depends on the precision of the local shape (*e.g.* scale, orientation) of the features used, apart from their position. On the other hand, without this information, matching is typically either not invariant (*e.g.* [38] is only invariant to translation) or more costly (*e.g.* RANSAC uses combinations of more than a single correspondence, and [32] resorts to searching for a number of scales/rotations on a discrete grid).

Another limitation may be that matching of small objects is influenced by background features through the aggregation of votes in (9), which is exactly what gives HPM its flexibility. We expect this influence to be higher than RANSAC-like methods that rather seek a single hypothesis that maximizes inliers. In fact, seeking for modes in the voting space is still possible under our pyramid framework and, at an additional cost, may give precise object localization and support partial matching.

It is a very interesting question whether there is more to gain from *geometry indexing*. Experiments on larger scale datasets or new methods may provide clearer evidence. Either way, a final re-ranking stage always seems unavoidable, and HPM can provide a valuable tool. In fact, our first objective with HPM has been for indexing and although this is too expensive for an online query, HPM can indeed perform an *exhaustive* re-ranking over our entire 2M distractor set in a few minutes. Further scaling up is a very challenging task we intend to investigate.

Another far-fetched prospect is to apply HPM to recognition problems. Our very assumption of inferring transformations from local feature shape makes this prospect rather limited for object category recognition, but not necessarily for specific objects. Whenever feature matching becomes increasingly ambiguous, *e.g.* with coarser vocabularies, repeating or 'bursty' patterns [16], HPM clearly favors low complexity over optimality. It would be interesting to explore this trade-off. Affine covariant features and geometric models more complex than similarity can be another subject of investigation, though flexibility of our model may not leave much space for improvement.

More can be found at our project page[3], including the entire 2M *World Cities* dataset.

[3] http://image.ntua.gr/iva/research/relaxed_spatial_matching

References

1. Avrithis, Y., Kalantidis, Y., Tolias, G., Spyrou, E.: Retrieving landmark and non-landmark images from community photo collections. In: ACM Multimedia. Firenze, Italy (2010) 16
2. Avrithis, Y., Tolias, G., Kalantidis, Y.: Feature map hashing: Sub-linear indexing of appearance and global geometry. In: ACM Multimedia. Firenze, Italy (2010) 1, 2
3. Ballard, D.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition (1981) 2
4. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: ECCV (2006) 10
5. Belongie, S., Malik, J., Puzicha, J.: Shape context: A new descriptor for shape matching and object recognition. In: NIPS, vol. 12, pp. 831–827 (2000) 2
6. Berg, A., Berg, T., Malik, J.: Shape matching and object recognition using low distortion correspondences. In: CVPR (2005) 3
7. Cao, Y., Wang, C., Li, Z., Zhang, L., Zhang, L.: Spatial-bag-of-features. In: CVPR, pp. 3352–3359 (2010) 2, 14
8. Carneiro, G., Jepson, A.: Flexible spatial configuration of local image features. PAMI pp. 2089–2104 (2007) 3, 6
9. Cheng, Y.: Mean shift, mode seeking, and clustering. PAMI **17**(8), 790–799 (1995) 3
10. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A.: Total recall: Automatic query expansion with a generative feature model for object retrieval. In: ICCV (2007) 16
11. Enqvist, O., Josephson, K., Kahl, F.: Optimal correspondences from pairwise constraints. In: ICCV (2009) 3
12. Fischler, M., Bolles, R.: 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) 2
13. Grauman, K., Darrell, T.: The pyramid match kernel: Efficient learning with sets of features. Journal of Machine Learning Research **8**, 725–760 (2007) 1, 3, 5, 7
14. Indyk, P., Thaper, N.: Fast image retrieval via embeddings. In: Workshop on Statistical and Computational Theories of Vision (2003) 3
15. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV (2008) 10
16. Jegou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: CVPR (2009) 18
17. Jegou, H., Douze, M., Schmid, C.: Improving bag-of-features for large scale image search. IJCV **87**(3), 316–336 (2010) 1, 2, 3, 8, 10, 11, 14, 16
18. Jiang, H., Yu, S.X.: Linear solution to scale and rotation invariant object matching. In: CVPR (2009) 3
19. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, vol. 2, p. 1 (2006) 3
20. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. IJCV **77**(1), 259–289 (2008) 2
21. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: ICCV, vol. 2, pp. 1482–1489 (2005) 1, 3, 4, 6, 11
22. Lin, Z., Brandt, J.: A local bag-of-features model for large-scale object retrieval. ECCV pp. 294–308 (2010) 3
23. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004) 1, 2, 3, 9, 12, 18
24. Mikulik, A., Perdoch, M., Chum, O., Matas, J.: Learning a fine vocabulary. In: ECCV (2010) 16
25. Olsson, C., Eriksson, A., Kahl, F.: Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In: CVPR (2007) 4

26. Perdoch, M., Chum, O., Matas, J.: Efficient representation of local geometry for large scale object retrieval. In: CVPR (2009) 13, 14, 16, 18
27. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007) 1, 2, 3, 10, 11, 12, 13, 14, 18
28. Philbin, J., Chum, O., Sivic, J., Isard, M., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR (2008) 3, 8, 10, 13, 14, 15, 16
29. Raguram, R., Frahm, J.M.: Recon: Scale-adaptive robust estimation via residual consensus. In: ICCV (2011) 3
30. Sahbi, H., Audibert, J.Y., Rabarisoa, J., Keriven, R.: Context-dependent kernel design for object matching and recognition. In: CVPR (2008) 3
31. Scott, G., Longuet-Higgins, H.: An algorithm for associating the features of two images. Proceedings of the Royal Society of London **244**(1309), 21 (1991) 3
32. Shen, X., Lin, Z., Brandt, J., Avidan, S., Wu, Y.: Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking. In: CVPR. IEEE (2012) 3, 13, 14, 18
33. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV, pp. 1470–1477 (2003) 1, 10
34. Tolias, G., Avrithis, Y.: Speeded-up, relaxed spatial matching. In: ICCV (2011) 3, 17
35. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. In: ECCV (2008) 3
36. Vedaldi, A., Soatto, S.: Relaxed matching kernels for robust image comparison. In: CVPR (2008) 3
37. Wu, Z., Ke, Q., Isard, M., Sun, J.: Bundling features for large scale partial-duplicate web image search. In: CVPR (2009) 2
38. Zhang, Y., Jia, Z., Chen, T.: Image retrieval with geometry-preserving visual phrases. In: CVPR, pp. 809–816. IEEE (2011) 2, 14, 18
39. Zhou, W., Lu, Y., Li, H., Song, Y., Tian, Q.: Spatial coding for large scale partial-duplicate web image search. In: ACM Multimedia. Firenze, Italy (2010) 2