Triangulation embedding and democratic aggregation for image search

Hervé Jégou, Andrew Zisserman

Triangulation embedding and democratic aggregation for image search

Hervé Jégou
Inria
Rennes

Andrew Zisserman
Dept. of Engineering Science
University of Oxford

Abstract

We consider the design of a single vector representation for an image that embeds and aggregates a set of local patch descriptors such as SIFT. More specifically we aim to construct a dense representation, like the Fisher Vector or VLAD, though of small or intermediate size.

We make two contributions, both aimed at regularizing the individual contributions of the local descriptors in the final representation. The first is a novel embedding method that avoids the dependency on absolute distances by encoding directions. The second contribution is a "democratization" strategy that further limits the interaction of unrelated descriptors in the aggregation stage.

These methods are complementary and give a substantial performance boost over the state of the art in image search with short or mid-size vectors, as demonstrated by our experiments on standard public image retrieval benchmarks.

1. Introduction

CONSIDER the problem of representing a set of vectors describing an image, for example a set of SIFT descriptors [18], by a single set-vector such that a simple comparison of two such set-vectors with cosine similarity reflects the similarity of the original sets. This is what is done in the literature in the many papers on large scale image retrieval, where the first step is to describe an image by a set of vectors (bag-of-features) each representing sub-parts (patches) of the image, and this set is then converted into a single vector based on an aggregation strategy, such as the bag-of-visual-words (BOW) representation [30], BOW with multiple- [12, 14] or soft-assignment [25, 32], locality-constrained linear coding [33], VLAD [13] or the Fisher vector [22, 23]. A similar approach is also employed in large scale image classification, but we will concentrate on image retrieval here.

All these methods can be decomposed into two steps: the *embedding step* individually maps each vector of the set to a high-dimensional space; whilst the *aggregating step* produces a single vector from the set of mapped vectors, for instance using sum- or max-pooling [5]. In this paper, we revisit these two steps and make a novel contribution to each. Our overall objective is to design a **"democratic" kernel,** such that each vector of the set contributes almost equally to the set similarity. This objective is addressed separately in both the embedding and aggregating stages.

First, we aim to design the embedding step ϕ such that, for any pair of vectors (x, y) describing two patches, the similarity $\phi(x)^\top \phi(y)$ is close to unity if the patches match, and close to zero if they do not, *i.e.*, the magnitude of $\phi(x)^\top \phi(y)$ should be small for unrelated patches. To this end, our first contribution is to introduce a *triangulation embedding* (T-embedding) that encodes the input vector with respect to a set of anchor points using only directions, not magnitudes. In contrast to most similar existing techniques [13, 17, 26], we discard the magnitude information between the input vector and the anchor points, as we consider this unreliable. From this point of view, our method can be seen as a way to localize the vector with a triangulation strategy.

Our second contribution is an aggregating strategy that explicitly takes into account the interference between the vectors of a set to remove it, and tends to give equal weight to each vector in the final score between two sets. This involves an optimization problem to find weights linearly balancing the contribution of each mapped vector in the final vector representation, and is solved with a modified Sinkhorn algorithm [15, 29]. This method is especially effective for relatively short representations, where it is essential to cancel the interference between the mapped vectors.

As will be demonstrated on public benchmarks for large scale image search, both these contributions give a significant improvement over previous techniques: our T-embedding outperforms the Fisher vector by a large margin for a given dimensionality, and our aggregation strategy offers a similar gain, which is also complementary to the so-called power-law normalization [23].

This paper is organized as follows. Section 2 introduces notation and motivates our contributions. Section 3 introduces our T-embedding and Section 4 our aggregation strategy. The experiments are presented in Section 5. Appendices are provided as supplementary material and available with code on the project page[1].

[1] `http://tinyurl.com/democratic-kernel`

2. Preliminaries

Let us consider two sets $\mathcal{X}$ and $\mathcal{Y}$ such that $\mathrm{card}(\mathcal{X}) = n$ and $\mathrm{card}(\mathcal{Y}) = m$. Each set consists of a set of vectors, such as local descriptors associated with an image. We first consider match kernels, in a framework derived from Bo and Sminchisescu [4][2], that have the form:

$$K(\mathcal{X}, \mathcal{Y}) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} k(x, y) = \psi(\mathcal{X})^\top \psi(\mathcal{Y}), \quad (1)$$

where $k(x, y)$ is a kernel between individual vectors of the sets. The right term indicates that we consider more specifically a vector representation for sets, such that two images are compared based on the inner product between their representations $\psi(\mathcal{X})$ and $\psi(\mathcal{Y})$. The match kernel is also written as

$$K(\mathcal{X}, \mathcal{Y}) = \mathbf{1}_n^\top \, \mathsf{K}(\mathcal{X}, \mathcal{Y}) \, \mathbf{1}_m \quad (2)$$

where $\mathbf{1}_n = \underbrace{[1, \ldots, 1]}_{\times n}$, and we define the $n \times m$ matrix

$$\mathsf{K}(\mathcal{X}, \mathcal{Y}) = \begin{bmatrix} k(x_1, y_1) & \ldots & k(x_1, y_m) \\ \vdots & \ddots & \vdots \\ k(x_n, y_m) & \ldots & k(x_n, y_m) \end{bmatrix}. \quad (3)$$

This matrix typically contains all the pairwise similarities between the local descriptors of two images. For any kernel K, we denote its normalized counterpart

$$K^\star(\mathcal{X}, \mathcal{Y}) = \alpha(\mathcal{X}) \, \alpha(\mathcal{Y}) \, K(\mathcal{X}, \mathcal{Y}), \quad (4)$$

where the normalizer $\alpha(.)$ is defined such that $K^\star(\mathcal{X}, \mathcal{X}) = 1$, *i.e.*, $\alpha(\mathcal{X}) = K(\mathcal{X}, \mathcal{X})^{-1/2}$.

2.1. Construction: embedding and aggregation

We divide the construction of K into two steps, namely *embedding* and *aggregation*. The embedding step $\phi : \mathbb{R}^{\mathsf{d}} \to \mathbb{R}^{\mathsf{D}}$ maps each $x \in \mathcal{X}$ as

$$x \mapsto \phi(x). \quad (5)$$

The aggregating step computes a single vector from the set $\{\phi(x_1), \ldots, \phi(x_n)\}$ of embedded vectors through a function ψ. This function is for instance a simple summation, in which case we denote it ψ_{s}:

$$\psi_{\mathrm{s}}(\mathcal{X}) = \sum_{x \in \mathcal{X}} \phi(x). \quad (6)$$

This simple definition of ψ is implicitly used in (1). In this case, $k(x, y) = \langle \phi(x) | \phi(y) \rangle$. The match kernel K is computed as the inner product between the aggregated vectors:

$$\psi_{\mathrm{s}}(\mathcal{X})^\top \psi_{\mathrm{s}}(\mathcal{Y}) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \phi(x)^\top \phi(y), \quad (7)$$

where each possible match (x, y) contributes to the overall set similarity, each with weight $\phi(x)^\top \phi(y)$.

This formulation, considered in particular by Bo and Sminchisescu [4] and Tolias *et al.* [31], encompasses many approaches. Let us first consider the embedding step. For a bag-of-visual-words vocabulary $\mathcal{C}$ of size $|\mathcal{C}| = \mathsf{D}$, a single descriptor x of $\mathcal{X}$ is mapped to a D-dimensional vector having one component equal to 1 (if not considering inverse document frequency) and the others to zero: $\phi_{\mathrm{BOW}}(x) = [\ldots, 0, 1, 0, \ldots]^\top$. The non-zero position is determined based on a nearest-neighbor assignment rule. With multiple assignment to visual words [14], several components are set to one, while soft assignment [23, 25, 32] gives different weights to a few components to account for the distances to centroids. Approaches such as local linear coding [33], the Fisher vector [22] or VLAD [13], also give alternative definitions of ϕ. Power-law normalization [11, 13, 23] modifies the function ψ by post-processing the aggregated vector.

Remark: The embedding step resembles the *coding* step as usually considered in the literature [13], and (6) is close to the *pooling* step [13]. We use another terminology to avoid confusion, because in our case all the operations applied on a per descriptor basis are included in the embedding stage. In this respect, the function ϕ already includes part of the pooling, including geometry-based pooling such as a spatial pyramid [16]. Consequently, in this formulation the dimensionality of $\phi(x)$ is typically the same as that of the final representation of the set $\mathcal{X}$.

2.2. Interferences in match kernels

The set vectorization underpinning (6), by casting a set of descriptor vectors into a single vector, has the advantage of producing a vector representation compatible with linear algebra, SVM and quantization, to mention but a few. However, this procedure gives unequal importance to the original descriptors in the final representation. More precisely, by comparing $\psi_{\mathrm{s}}(\mathcal{X})$ to itself, the contribution of a given vector x to the set similarity $\psi_{\mathrm{s}}(\mathcal{X})^\top \psi_{\mathrm{s}}(\mathcal{X})$ is given by $\phi(x)^\top \psi_{\mathrm{s}}(\mathcal{X})$:

$$\phi(x)^\top \sum_{x' \in \mathcal{X}} \phi(x') = \|\phi(x)\|^2 + \phi(x)^\top \sum_{x' \in \mathcal{X} : x' \neq x} \phi(x'). \quad (8)$$

This equation suggests two important properties for ϕ:

1. The left term $\|\phi(x)\|^2$ isolates the matching descriptor, whose contribution strongly (*i.e.*, quadratically) depends on its norm.

[2] The only minor difference is that we do not use the same normalizers: [4] normalize the vector representation by the number of features.

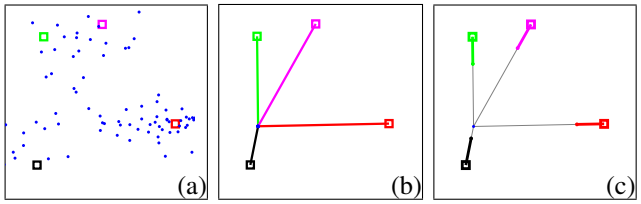

Figure 1. Illustration of our T-embedding for $\mathsf{d} = 2$ and $|\mathcal{C}| = 4$. (a) Distribution and learned anchor points. (b) Residual vectors associated with a given vector x; (c) Normalized residuals $\mathrm{R}(x)$.

2. The right term is the "noise" polluting the contribution of x due to its interaction with the other vectors.

This paper aims at addressing these two problems, both in the design of the embedding function ϕ (in Section 3) and in that of the aggregation function ψ (in Section 4).

3. Triangulation embedding

In this section, we introduce the T-embedding function $\phi_\triangle$. It offers several desirable properties motivated by the observations raised by (8), in particular that the inner product between two unrelated features is almost zero, except when the features are close enough with respect to other features drawn from the same distribution.

3.1. Construction

Given a distribution of vectors x on the d-dimensional unit sphere, we consider a set $\mathcal{C} = \{\mathsf{c}_1, \ldots, \mathsf{c}_{|\mathcal{C}|}\}_i$, $\mathsf{c}_i \in \mathbb{R}^\mathsf{d}$, of $|\mathcal{C}|$ representative anchor points. This set is typically learned by k-means and is similar to a visual vocabulary. Yet in our context it is more related to the anchor graph proposed for the purpose of binary encoding [17].

In contrast to most existing works, we focus on the directional information and discard the absolute distances to the anchor points. This strategy can be related to secant manifolds [9]. This is the key to circumvent the "bandwidth issue", *i.e.*, the dependence on absolute distances, which are generally not reliable [7]. As a result, our novel vector representation is implicitly defined by triangulation[3]. This is achieved by considering the set of normalized residual vectors

$$\mathrm{r}_j(x) = \left\{ \frac{x - \mathsf{c}_j}{\|x - \mathsf{c}_j\|} \right\} \quad \text{for} \quad j = 1 \ldots |\mathcal{C}|, \tag{9}$$

which preserves the angular information between x and c_j while discarding the absolute magnitude. Figure 1 illustrates this triangulation strategy. We assume that $x \neq \mathsf{c}_j$ for all j, which is guaranteed for ℓ_2-normalized SIFT vectors if c_j are obtained by k-means (k-means centroids, as the average of distinct vectors, strictly lie inside the unit ball). The concatenation $\mathrm{R}(x) = [r_1(x)^\top, \ldots, r_{|\mathcal{C}|}(x)^\top]^\top$ is an intermediate D-dimensional representation such that $\mathsf{D} = |\mathcal{C}| \times \mathsf{d}$. It is redundant and gives too much weight to the main directions. We subsequently whiten the representation [17] (center, rotate and scale based on eigenvalues). More precisely, denoting by Σ the covariance matrix associated with the random variable $\mathrm{R}(X)$, our T-embedding is obtained from $\mathrm{R}(x)$ as

$$\phi_\triangle(x) = \Sigma^{-1/2}(\mathrm{R}(x) - \mathrm{R}_0), \tag{10}$$

where both $\mathrm{R}_0 = \mathbb{E}_X[\mathrm{R}(X)]$ and Σ are empirically measured on a training set.

Figure 2 depicts, in the original space, the values associated with each eigenvector, *i.e.*, each component of the output descriptor $\phi_\triangle(x)$. By analogy to PCA or Laplacian Eigenmaps [3], the largest eigenvalues are associated with the "low frequencies": the corresponding components vary slowly as a function of the input descriptors. In contrast, eigenvectors associated with small eigenvalues correspond to high frequencies: A small variation of a given input feature has a larger impact on the corresponding output component, as can be seen in Figure 2 where the components are ordered from largest (left) to smallest eigenvalues (right).

As a result, our embedding compares descriptors at different resolutions. This is also the case in prior works like the pyramid match kernel [8] and the vocabulary tree [21], which implement varying resolutions by using different quantizers. In our case, there is no quantization artifact: the first components reflect the rough positions while the last are more localized. In order to improve the localization of the descriptors, we discard the d first components associated with the largest eigenvalues. This reduces the variance of the cosine similarity between unrelated descriptors, as discussed in Appendix A. The final dimensionality of the embedded descriptor $\phi_\triangle(x)$ is therefore $\mathsf{D} = \mathsf{d} \times (|\mathcal{C}| - 1)$.

3.2. Efficient computation with match kernels

We now consider a match kernel inherited from our T-embedding, as introduced in Section 2. Exploiting the linearity in equations (6) and (10), we compute the explicit set representation of a vector set $\mathcal{X}$ comprising n vectors as

$$\psi(\Phi_\triangle(\mathcal{X})) = \sum_{x \in \mathcal{X}} \phi_\triangle(x) \tag{11}$$

$$= \Sigma^{-1/2} \left(\sum_{x \in \mathcal{X}} \mathrm{R}(X) \right) - n\Sigma^{-1/2}\mathrm{R}_0. \tag{12}$$

Computing this representation for typical parameters ($\mathsf{d} = 128, n = 3000, |\mathcal{C}| = 16$) takes about 20 ms on a quad-core laptop with an efficient Matlab implementation. However, it is worth normalizing $\phi_\triangle$ before the aggregation to ensure that $\phi_\triangle(x)^\top \phi_\triangle(x) = 1$. In particular, this is important for the aggregation technique presented in Section 4. In this case the computation is slower, as each $R(x)/\|R(x)\|$ is projected separately with $\Sigma^{-1/2}$.

[3] Triangulation only relies on angles to determine the position of a point, in contrast to trilateration that finds point locations by measuring distances.

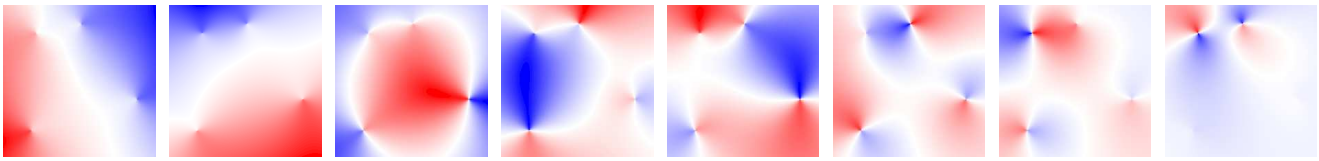

Figure 2. Pictorial representation of the function $\phi_\triangle$ for the same example as in Figure 1. The 8 components of $\phi_\triangle(x)$, where the spatial position in each map indicates the 2-D input vector x, while the magnitude represents the output value (red=negative, blue=positive, white=0) of a given component of $\phi_\triangle$. The components associated with the two largest eigenvalues (on the left) are smoothly varying, and are discarded in the final representation. Observe the locality conveyed by the other components.

3.3. Properties

By construction, and apart from singular cases of $\mathcal{C}$, our T-embedding satisfies several desirable properties. First, the function $\phi_\triangle : \mathbb{R}^{\mathsf{d}} \to \mathbb{R}^{\mathsf{D}} : x \mapsto \phi_\triangle(x)$ is injective. Second, it is continuous everywhere except at the location of the centroids (and as noted above, for a vector distribution on the unit sphere (as SIFT descriptors are) and anchor points learned with k-means, it is continuous everywhere because the centroids are strictly inside the sphere). This property ensures that an infinitesimal change of the input vector does not produce an abrupt change in the output space. Similarly, the embedding function is differentiable everywhere.

Note that VLAD, LLC and the Fisher vector are also injective, but bag-of-words is not. Among these three embedding techniques, only Fisher is continuous.

In addition to these formal mathematical properties, another key characteristic of our embedding, as will be demonstrated next, is that the inner product $\langle \phi_\triangle(x) | \phi_\triangle(y) \rangle$ between two unrelated vectors is close to 0 with high probability. In contrast, when applied to SIFT descriptors, similar patches compared with their embedded descriptors have a similarity much greater than 0.

Quantitative analysis on a patch dataset. We collect empirical statistics of the cosine similarity for related and unrelated image patches. For this purpose, we use the datasets *Liberty* and *Notredame* provided by Brown *et al.* [34]. Each dataset consist of about 500k patches from multiple images grouped into 150k clusters, where a cluster corresponds to the same physical scene point. These datasets are usually employed for learning patch descriptors [28, 34], but here we use them for learning and evaluating embeddings, and simply use RootSIFT [1] as the patch descriptor. Learning (*e.g.*, of Σ for our method) is performed on Liberty for all methods. We test on Notredame by considering 150k pairs of matching descriptors (a pair per cluster) and the same number of pairs for unrelated descriptors (we take two descriptors from two different clusters). Figure 3 shows the cosine similarity for related/unrelated patches for the original descriptors (RootSIFT), and after they are individually mapped with Fisher vector encoding (without aggregation) and our T-embedding ($\phi_\triangle$).

Both Fisher and T-embedding increase the contrast between unrelated and related descriptors. In this respect, T-embedding is better than Fisher. First, on average the similarity between unrelated patches is closer to 0 with $\phi_\triangle$, and few unrelated pairs deviate from this behavior. Moreover, in Fisher, a large proportion (note the log scale) of correct matches are given a similarity close to 0. This proportion is comparatively much lower in T-embedding.

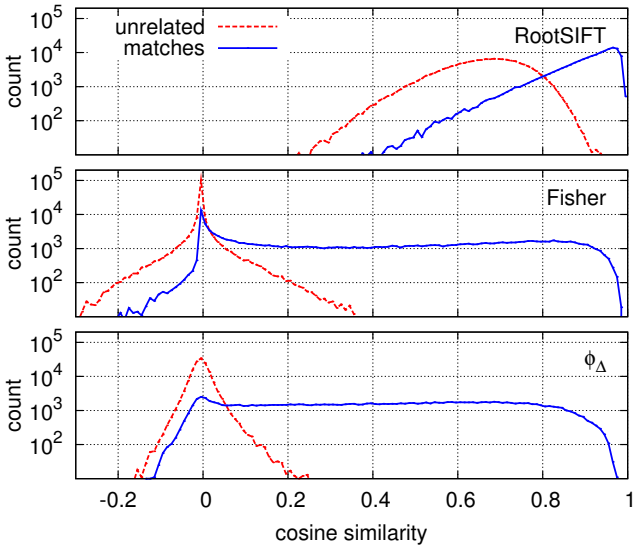

Figure 3. Histogram of the cosine similarity between related (plain) and unrelated (dashed) patches, for RootSIFT descriptors (top), embedded with Fisher kernel (middle, $|\mathcal{C}| = 16$) and our T-embedding (bottom, $|\mathcal{C}| = 16$). The count is shown in log-scale.

A high $\phi_\triangle$ cosine similarity associated with two local descriptors x and y reliably reflects the confidence that we have in the visual resemblance of the corresponding patches. As a byproduct of this observation, it is possible to determine how close patches are based on their absolute $\phi_\triangle$ cosine similarity, as visually illustrated in Figure 4 by detecting similar patterns (bursts [11]) in a given image. The quality of the similarity measure for descriptors mapped with T-embedding is evaluated with a ROC curve in Appendix A. Supervised learning of patch descriptors [28, 34] would further improve the separation in all cases.

4. Democratic aggregation

Our T-embedding reduces the interferences in (8) by giving a cosine similarity that is almost 0 for unrelated pairs of descriptors, while providing a comparatively higher posi-

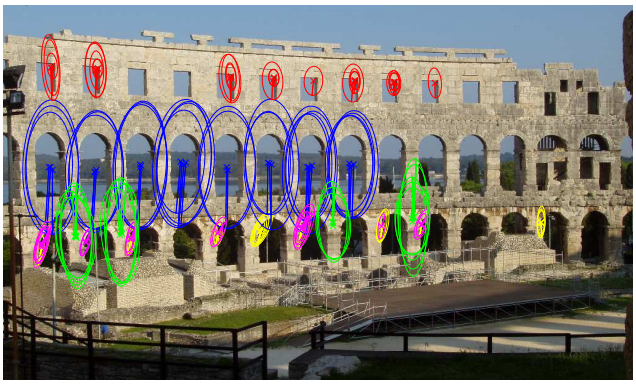

Figure 4. Detection of self-similar structures by thresholding the cosine similarity between embedded descriptors. To produce it, we have simply thresholded the gram matrix K (with threshold 0.5) between descriptors mapped with $\phi_{\triangle}$, and performed connected component analysis of the associated graph. We display the five largest components (one color for each).

tive score to the true matches. However, at this stage, the descriptors are still considered independently. In the following, we further limit interferences by explicitly analyzing and reducing them in the aggregation stage.

A match kernel $\mathcal{K}$ is defined as *democratic* if and only if, for any set $\mathcal{X}$ s.t. $\mathrm{card}(\mathcal{X}) = n$, the corresponding matrix K satisfies

$$\mathsf{K}(\mathcal{X},\mathcal{X})\,\mathbf{1}_n = C\,\mathbf{1}_n, \tag{13}$$

where the scaling factor C may (or not) depend on $\mathcal{X}$. In other words, a democratic kernel ensures that all the vectors in $\mathcal{X}$ contribute equally to the set self-similarity. In the rest of this section, we present the optimization problem aiming at producing a democratic kernel from an arbitrary one in the aggregation stage. Then we discuss convergence issues and present a strategy to achieve convergence.

4.1. Democratization

A kernel as in (1) is normally not democratic. To achieve this property, we modify it by including additional weights linearly associated with each vector, as

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \lambda_{\mathcal{X}}(x)\,\lambda_{\mathcal{Y}}(y)\,k(x,y). \tag{14}$$

Each scalar $\lambda_{\mathcal{X}}(x)$ (respectively $\lambda_{\mathcal{Y}}(y)$) only depends on x and the set $\mathcal{X}$ (respectively y and $\mathcal{Y}$). Considering the set $\mathcal{X} = \{x_1,\ldots,x_n\}$, the corresponding weights $\lambda_i,\ i = 1\ldots n$, are determined by solving, when possible, the set of equations

$$\forall x_i \in \mathcal{X}, \lambda_i \times \sum_{x_j\in\mathcal{X}} \lambda_j\,k(x_i,x_j) = C \tag{15}$$

under the constraint $\forall i,\ \lambda_i > 0$. The problem is summarized in matrix form as

$$\Lambda\mathsf{K}\Lambda\mathbf{1}_n = C\mathbf{1}_n, \tag{16}$$

where $\Lambda = \mathrm{diag}(\boldsymbol{\lambda}) = \mathrm{diag}(\lambda_1,\ldots,\lambda_n)$ is a matrix whose diagonal is strictly positive. Note, (14) is equivalent to defining a new match kernel $k'(x,y) = \lambda_{\mathcal{X}}(x)\,\lambda_{\mathcal{Y}}(y)\,k(x,y)$. Consider the particular case of the match kernel "embed+aggregate" introduced in Section 2. Equation (14) is re-written as

$$\begin{aligned}\mathcal{K}(\mathcal{X},\mathcal{Y}) &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \lambda_{\mathcal{X}}(x)\,\lambda_{\mathcal{Y}}(y)\,\phi(x)^\top\phi(y) \qquad (17)\\ &= \left(\sum_{x\in\mathcal{X}}\lambda_{\mathcal{X}}(x)\phi(x)\right)^\top \left(\sum_{y\in\mathcal{Y}}\lambda_{\mathcal{Y}}(y)\phi(y)\right),\end{aligned}$$

where it can be seen that the democratization amounts to defining an alternative function ψ, denoted ψ_{d}. More precisely, we replace the aggregation function ψ_{s} in (6) by the weighted summation:

$$\psi_{\mathrm{d}}(\Phi(\mathcal{X})) = \sum_{\phi_i\in\Phi(\mathcal{X})} \lambda_i\phi_i. \tag{18}$$

The weighted vector is ℓ_2-normalized to produce the normalized match kernel.

4.2. Modified Sinkhorn scaling algorithm

It is worth noticing that this problem resembles that of projection to a doubly stochastic matrix [29]: It is equivalent if $C = 1$ and K is positive. Under additional assumptions (matrix K has total support and is fully indecomposable [15]), the Sinkhorn's algorithm converges to a unique solution satisfying $\forall i,\ \lambda_i > 0$. It is a fixed-point algorithm that proceeds by alternately normalizing the rows and columns. We adopt a symmetric variant analyzed by Knight [15] and weaken the impact of each iteration, as recently suggested [14], by using a power exponent smaller than 0.5 for a smoother convergence.

Appendix B gives pseudo-code for this optimization strategy. Sinkhorn is an algorithm that converges quickly. We stop it after 10 iterations for efficiency reasons. Experimentally, no benefit comes from using more iterations.

4.3. General case: convergence issue and a solution

In the case of an arbitrary kernel $k(.,.)$, the assumptions required for convergence with Sinkhorn are generally not satisfied (Matrix K nonnegative and fully indecomposable [15]). Thus, a positive solution does not necessarily exist. Any optimization algorithm may produce *negative weights* for kernels with negative values, which typically happen if $\sum_j k(x_i,x_j) < 0$. This is not desirable because it means that the weight computation is sensitive to new/deleted vectors in the set. We solve this problem by adopting the following pre-processing step.

Enforcing positivity. After ℓ_2-normalizing $\phi_{\triangle}(x)$ so that the energy is identical for all vectors, we solve the convergence issue by setting all negative values to 0 in K. The

weights computed with this new matrix K^+ are positive with Sinkhorn's algorithm because all rows/columns sums are positive. The resulting embedding ψ_{d} is not strictly a democratic kernel but tends towards more "democracy".

4.4. Discussion

Consider the right term in (8). If the embedding perfectly removes it (no interaction between the descriptors of the same set), then our democratization is a calibration such that all the norms $\|\phi(x)\|$ are equal. Appendix C also shows that our strategy is equivalent to the square-root componentwise normalization in the case of bag-of-visual-words vectors without inverse document frequency weighting.

5. Experiments

This section presents results for our democratic kernel. The novel ingredients that form our method, namely T-embedding and democratic aggregation, can be used separately. Therefore we evaluate their impact separately, by performing experiments (a) with T-embedding only; (b) with democratic aggregation applied to Fisher embedding; (c) with our two methods. Throughout this section, we only use the normalized kernel $\mathcal{K}^\star$, meaning that the image vector is normalized to have unit Euclidean norm.

5.1. Datasets and evaluation protocol

We adopt public datasets and corresponding evaluation protocols that are often used in the context of large scale image search. All the learning stages, *i.e.*, k-means clustering and learning the projection for our T-embedding, are performed off-line using a distinct image collection, that does not contain the indexed database nor the query images.

Oxford5k [24] consists of 5062 images of buildings and 55 query images corresponding to 11 distinct buildings in Oxford. The search quality is measured by the mean average precision (mAP) computed over the 55 queries. Images are annotated as either relevant, not relevant, or *junk*, which indicates that it is unclear whether a user would consider the image as relevant or not. Following the recommended protocol, the *junk* images are removed from the ranking. For the experiments on Oxford5k, all the learning stages are performed on the **Paris6k** dataset [25]. **Oxford105k** is the combination of Oxford5k with 100k negative images, in order to evaluate the search quality on a large scale.

INRIA Holidays [12]. This dataset includes 1491 photos of different locations and objects, 500 of them being used as queries. The search quality is measured by mAP, with the query removed from the ranked list. To obtain the vocabulary, we use the independent dataset Flickr60k provided with Holidays. For Holidays and Oxford5k, we perform the experiments three times for our methods (for three distinct vocabularies) and report the mean performance.

5.2. Implementation notes

Local descriptors are extracted with the Hessian-affine detector [19] and described by SIFT [18]. We have used the same descriptors as provided in a previous paper [2]. We use the RootSIFT variant [1], in all our experiments.

Power-law normalization. Images contain "visual bursts" [11], meaning that numerous descriptors are almost identical within the same image, as observed in Figure 4. These descriptors tend to dominate the similarity even in democratic kernels. As a common post-processing step [11, 23], we apply power-law normalization on the vector image representation, and subsequently ℓ_2-normalize it. This processing is parametrized by a constant α that controls the value of the exponent when modifying a component a such that $a := |a|^\alpha \mathrm{sign}(a)$. We standardly set $\alpha = 0.5$ to ensure a fair comparison between the methods. Note that this section also includes a specific analysis for this parameter.

Rotation and Normalization (RN). The power-law normalization suppresses visual bursts, but not the frequent co-occurrences that also corrupt the similarity measure [6]. In VLAD, this problem is addressed [10] by whitening the vectors. However, the whitening learning stage requires a lot of input data and the smallest eigenvalues generate artifacts. This makes such processing suitable only when producing very short representations. As an alternative [27], we apply power-normalization *after* rotating the data with a PCA rotation matrix learned on image vectors (from the learning set), i.e. no whitening. This produces a similar effect to that of whitening, but is more stable and not dependent on PCA eigenvalues. To avoid the full eigendecomposition and the need to use too many images for the learning stage, we compute the first 1000 eigenvectors and apply Gram-Schmid orthogonalization on the reminder of the space (orthogonal complement to these first eigenvectors) to produce a complete basis. After this rotation, we apply the regular power-law normalization, which then jointly addresses the bursts and co-occurrences by selecting a basis capturing both phenomenons on the first components.

5.3. Impact of the methods and parameters

Our methods introduce no extra parameter compared with existing techniques, apart from constants with no impact on performance, like the number of iterations in Sinkhorn. The main parameters are the vocabulary size $|\mathcal{C}|$ and the parameter α associated with power-law normalization. The analysis of these parameters is shown in Figure 5 for Holidays. The analysis for Oxford5k is in Appendix D (supplementary material). The conclusions drawn are identical on both datasets. To complement these curves, Table 1 shows the impact of our methods step by step for a fixed vocabulary size on Oxford5k, Oxford105k and Holidays.

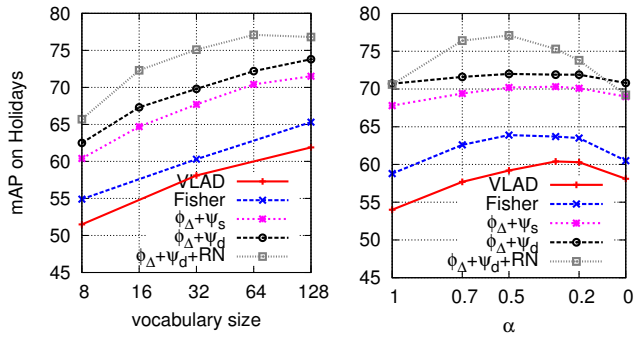

Figure 5. Impact of the parameters on the Holidays' performance for different image vector representations: VLAD, Fisher, our T-embedding $\phi_\triangle$ with sum aggregation ψ_s, and democratic aggregation ψ_d (without and with RN). *Left:* as a function of vocabulary size $|\mathcal{C}|$; *Right:* as a function of the power-law normalization exponent α. Note, $\alpha = 0$ amounts to binarizing the vector.

Vocabulary size. For all representations, including T-embedding and democratic aggregation, the performance is an increasing function of the vocabulary size. For reference, we give the performance of the improved Fisher baseline. Note the large gain provided by our embedding for a fixed vocabulary size. Our aggregation method gives a complementary gain. The improvement tends to be smaller for larger vocabularies: This is expected, as for larger vocabularies the interaction between the descriptors is less important than for small ones. For $|\mathcal{C}| > 128$, the benefit of democratic aggregation is not worth the computational overhead.

Our aggregation strategy gives a significant boost in performance with $\phi_\triangle$. As to be expected, it improves the performance when no power-law is applied. Moreover, the analysis of the parameter α also reveals that our aggregation method ψ_d is complementary to the power-law normalization, as both methods improves the score.

Power-law normalization and RN. Power-law normalization is less important with our methods (the right curves $\phi_\triangle + \psi_\mathrm{s}$ and $\phi_\triangle + \psi_\mathrm{d}$ are more flat), except if we employ RN: This normalization gives a large improvement in performance when used with the (standard) parameter $\alpha = 0.5$.

Dimensionality reduction. In order to get shorter representations, we keep the first D' components, after RN normalization, of the vector produced by our embeddings. Table 1 reports the performance for short vectors of varying dimensionality, $D' = 128$ to 1024. Despite a drop in performance due to dimensionality reduction, our best configuration ($\phi_\triangle + \psi_\mathrm{d}$+RN) still outperforms the 5120-dimensional Fisher vector with $D' = 512$.

5.4. Comparison with the state of the art

Baselines. We consider as baselines recent works targeting the same application scenario and similar representations, *i.e.*, that represent an image by a vector that may be subsequently reduced [13]. We compare with works recently published on similar mid-size vector representations [2, 13]. We also compare with our re-implemented (improved) version of VLAD and Fisher vectors that integrates RootSIFT. This baseline, by itself, approaches or outperforms the state of the art by combining most of the effective ingredients.

Table 1. Impact of our methods on the performance. First we evaluate the Fisher and combine it with democratic aggregation. Then we consider T-embedding $\phi_\triangle$ with sum (ψ_s) and democratic (ψ_d) aggregation, and show the boost given by RN. Finally, we present results after dimensionality reduction to short vectors. $|\mathcal{C}| = 64$.

method ↓	dim. red. to → D'	mAP Holidays	mAP Oxford5k	mAP Ox105k
Fisher baseline	–	63.9	50.7	44.9
Fisher + ψ_d	–	63.8	52.0	45.9
$\phi_\triangle$ + ψ_s	–	70.4 ±0.3	58.9 ±0.3	52.3
$\phi_\triangle$ + ψ_d	–	72.2 ±0.2	61.2 ±0.4	55.9
$\phi_\triangle$ + ψ_s + RN	–	74.5 ±0.4	63.3 ±0.9	55.5
$\phi_\triangle$ + ψ_d + RN	–	**77.1** ±0.7	**67.6** ±0.2	**61.1**
$\phi_\triangle$ + ψ_d + RN	→ 1,024	72.0 ±0.2	56.2 ±0.1	50.2
$\phi_\triangle$ + ψ_d + RN	→ 512	70.0 ±0.6	52.8 ±0.4	46.1
$\phi_\triangle$ + ψ_d + RN	→ 256	65.7 ±0.3	47.2 ±0.2	40.8
$\phi_\triangle$ + ψ_d + RN	→ 128	61.5 ±0.7	40.0 ±0.1	33.9

Results. Table 2 shows that our method outperforms the compared methods by a large margin on all datasets. The gain over a recent paper [2] using a larger vocabulary is **+11.8%** in mAP on both Holidays and Oxford5k. Compared with our improved Fisher baseline using the same vocabulary size, the gain is **+13.2%** in mAP for Holidays, **+16.9%** for Oxford5k and **+16.2%** for Oxford105k. Even when reducing the dimensionality to $D' = 1,024$ components, we outperform all other methods by a large margin, with a much smaller vector representation. Only when reducing the vector to $D' = 128$ components, our method gives on average slightly lower results than those reported by Arandjelović and Zisserman [2].

6. Conclusion

The key motivation for this paper is to reduce the interference between local descriptors when combining them to produce a vector representation of an image. It is addressed by two novel and complementary methods. The first is a T-embedding that reduces the impact of unrelated matches on the image similarity. The second method explicitly limits the interference between descriptors when aggregating them. The resulting representation compares favorably with state-of-the-art encoding methods for image search, such as the Fisher kernel, even when our representation is reduced to 1,000 components.

Table 2. Comparison with the state of the art for short and intermediate vector dimensionality on Holidays, Oxford5k and Oxford105k datasets. The two last rows show the performance after reducing our vector from 8,064 to 1,024 or 128 components.

	$\|\mathcal{C}\|$	D	mAP		
method ↓			Hol.	Ox5k	Ox105k
BOW [13]	20k	20,000	43.7	35.4	–
BOW [13]	200k	200,000	54.0	36.4	–
VLAD [13]	64	4,096	55.6	37.8	–
Fisher [13]	64	4,096	59.5	41.8	–
VLAD-intra [2]	256	32,536	65.3	55.8	–
VLAD-intra [2]	256	$\rightarrow 128$	62.5	44.8	37.4
Our methods					
$\phi_\triangle + \psi_\mathrm{s}$ + RN	16	1,920	69.5	53.1	45.6
$\phi_\triangle + \psi_\mathrm{s}$ + RN	64	8,064	74.5	63.3	55.5
$\phi_\triangle + \psi_\mathrm{d}$ + RN	16	1,920	72.3	57.1	49.5
$\phi_\triangle + \psi_\mathrm{d}$ + RN	64	8,064	**77.1**	**67.6**	**61.1**
$\phi_\triangle + \psi_\mathrm{d}$ + RN	16	$\rightarrow 128$	61.7	43.3	35.3
$\phi_\triangle + \psi_\mathrm{d}$ + RN	64	$\rightarrow 1{,}024$	72.0	56.0	50.2

Acknowledgments. This work was done within the Project Fire-ID, supported by the ANR French research agency, and also supported by ERC grant VisRec no. 228180. We warmly thank Karen Simonyan for providing features and Miaojing Shi for the complementary results on UKB and Holidays+Flickr1M in Appendix E. We also thank Florent Perronnin and Naila Murray for preliminary discussions, and encourage reading their related paper [20] on generalized max pooling.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] R. Arandjelović and A. Zisserman. All about VLAD. In *CVPR*, Jun. 2013.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, Jun. 2003.

[4] L. Bo and C. Sminchisescu. Efficient match kernels between sets of features for visual recognition. In *NIPS*, 2009.

[5] Y. Boureau, F. Bach, Y. Lecun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010.

[6] O. Chum and J. Matas. Unsupervised discovery of co-occurrence in sparse high dimensional data. In *CVPR*, 2010.

[7] Y. Fu, M. Liu, and T. S. Huang. Conformal embedding analysis with local graph modeling on the unit hypersphere. In *CVPR*, 2007.

[8] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[9] C. Hegde, A. C. Sankaranarayanan, W. Yin, and R. G. Baraniuk. A convex approach for learning near-isometric linear embeddings. *JMLR*, 2012. *in submission*.

[10] H. Jégou and O. Chum. Negative evidences and cooccurences in image retrieval: The benefit of PCA and whitening. In *ECCV*, 2012.

[11] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

[12] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, Feb. 2010.

[13] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local descriptors into compact codes. In *PAMI*, 2012.

[14] H. Jégou, C. Schmid, H. Harzallah, and J. Verbeek. Accurate image search using the contextual dissimilarity measure. *PAMI*, Jan. 2010.

[15] P. A. Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

[16] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *CVPR*, Jun. 2006.

[17] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.

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

[19] K. Mikolajczyk and C. schmid. Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004.

[20] N. Murray and F. Perronnin. Generalized max pooling. In *CVPR*, 2014.

[21] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[22] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[23] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.

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

[25] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[26] N. L. Roux and F. Bach. Local component analysis. In *Proc. Intl Conf. on Learning Representations*, 2013.

[27] B. Safadi and G. Quénot. Descriptor optimization for multimedia indexing and retrieval. In *CBMI*, 2013.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 2014.

[29] R. Sinkhorn. A relationship between arbitrary positive matrices and double stochastic matrices. *Annals of Mathematics and Statistics*, 35:876–879, 1964.

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

[31] G. Tolias, Y. Avrithis, and H. Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *ICCV*, 2013.

[32] J. van Gemert, C. Veenman, A. Smeulders, and J. Geusebroek. Visual word ambiguity. *PAMI*, Jul. 2010.

[33] J. Wang, J. Yang, F. L. K. Yu, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[34] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, 2007.

Democratic match kernels for image search
— Supplemental material: Appendices —

Hervé Jégou
Inria
Rennes, France

Andrew Zisserman
Dept. of Engineering Science
University of Oxford

Abstract

The appendices provided in this supplemental material complement our paper in several aspects. We provide additional experiments, results and interpretations. We give pseudo-code of our democratization strategy and show how the democratic kernel relates to square-rooting normalization (powerlaw with $\alpha = 0.5$) in the case of bag-of-words vectors without inverse document frequency terms. Then, we report additional results, in particular on the UKB benchmark and Holidays merged with 1 million images. Finally, we provide complexity measurements.

Appendix A – ROC curves and discussion

Figure 1 gives the receiver operating curves associated with Fisher and T-embedding. We consider the same setup as in recent papers for evaluation of learned local descriptors [7], which is also used in Section 3.3. We learn on Liberty and test on NotreDame. The baseline is RootSIFT. Here, we use this framework to evaluate the impact of the encoding technique on the hypothesis test. More precisely we compare local descriptors *individually encoded* by the Fisher vector (mean components, no power-law component-wise normalization) and our encoding technique $\phi_\triangle$. We also evaluate our strategy *without* removing the low frequency terms.

The first observation is that the Fisher vector is not as good as RootSIFT with respect to the hypothesis test, *i.e.*, when vectors are compared individually with cosine similarity. Observe, also, that the benefit of T-embedding is especially significant for low false positive rates, which explains why T-embedding is more suitable than RootSIFT to detect visual bursts based on a high similarity threshold. Despite its relative poor performance on these curves, recall that the Fisher vector offers better performance than RootSIFT once aggregated with a sum.

This leads us to the following interpretation: the main benefit of these embedding functions is not the underlying quality of the comparison metric (although this also matters), but mainly the fact that they limit the interferences between descriptors once aggregated, thanks to the mapping to a higher dimensional space. This is also supported by the comparison of T-embedding before and after removing the first components. While the ROC curve of T-embedding is better when keeping the low frequencies (and better than RootSIFT), the recognition performance for image search is better when we remove them.

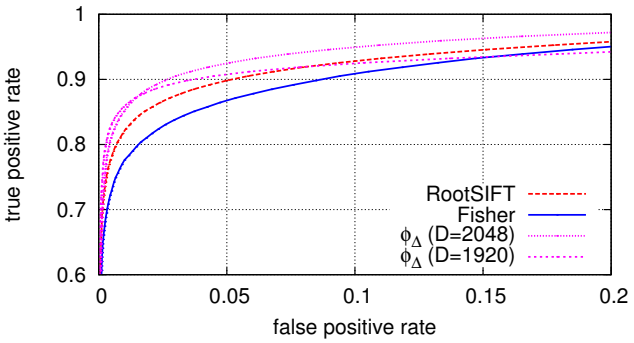

Figure 1. ROC curves: comparison of RootSIFT, Fisher (applied on RootSIFT, k=16), and T-embedding (k=16) before (D=2048) and after (D=1920) removing the first D components associated with the low frequencies.

Impact of removing the components associated with low frequencies. The previous interpretation is consistent with the fact that, before removing the low frequencies in our T-embedding, the variance of the cosine similarity for unrelated descriptors is comparable to that of RootSIFT after PCA. In this case, there is a lot of interferences between the aggregated descriptors.

After removing these components, the distribution of cosine similarities, for unrelated descriptors, is more comparable to that obtained with high-dimensional vectors distributed on the unit hypersphere: It is Gaussian-like with a small variance. These remarks are illustrated by Figure 2, which shows how the distributions of related and unrelated patches evolve when removing a varying number of components: 0 (=none removed), 64, 128, 192 and 256. Observe

that our removal strategy has a dramatic impact of the distributions of cosine similarities, in particular for the first d components, where d is the input (SIFT) dimensionality.

The strategy has also an effect on the true matches: the corresponding similarities decrease when removing components. Even though a significant proportion of the matches remain associated with strong similarities, the hypothesis test on individual descriptors is weakened. Our strategy therefore aims at optimizing a trade-off between

- reducing the interferences in the match kernel
- and keeping the hypothesis test as good as possible.

By removing more than d=128 components, the benefit of variance reduction is less significant for unrelated patches: compare the Figures associated with 128 and 256 removed components, which have comparable shapes for unrelated matches. In contrast, the similarity of true matches continue to suffer. In particular, more patches have a similarity close to 0. These observations are consistent with our preliminary experiments, where the optimal performance, with respect to search accuracy, is approximately obtained when removing d components.

As a final remark, recall that the embeddings we consider are learned in an unsupervised manner: The ROC curves would be certainly improved by using metric learning techniques such as those proposed for descriptor learning [6]. However, as discussed in this section, the ROC performance is not directly related to the image search performance because of the interferences. This shows the limit of the ROC evaluation setup in the context of match kernels.

Appendix B – Modified Sinkhorn: Pseudo-code

The algorithm below gives pseudo-code for our democratization strategy, solved by a symmetric variant of the Sinkhorn scaling algorithm. We set $\gamma = 0.3$, typically. Note however that using a value lower than $\gamma = 0.5$ (like in the regular Knight variant) is critical only if we re-compute the kernel matrix from the weighted descriptors at each iteration of the Sinkhorn algorithm.

Input: Gram matrix K % of size $n \times n$
parameters γ and $\mathrm{n_{iter}}$
Output: Weight vector $\boldsymbol{\lambda}$
Initialization: $\boldsymbol{\lambda} = \mathbf{1}_n$

For i=1 to $\mathrm{n_{iter}}$
$\boldsymbol{\sigma} = \mathrm{diag}(\boldsymbol{\lambda}) \times \mathsf{K} \times \mathrm{diag}(\boldsymbol{\lambda}) \times \mathbf{1}_n$ % Sums of rows
$\forall i,\ \lambda_i := \lambda_i / \sigma_i^\gamma$ % Update

---

Note, we provide a package associated with paper, which includes a Matlab implementation of this algorithm: `http://tinyurl.com/democratic-kernel`.

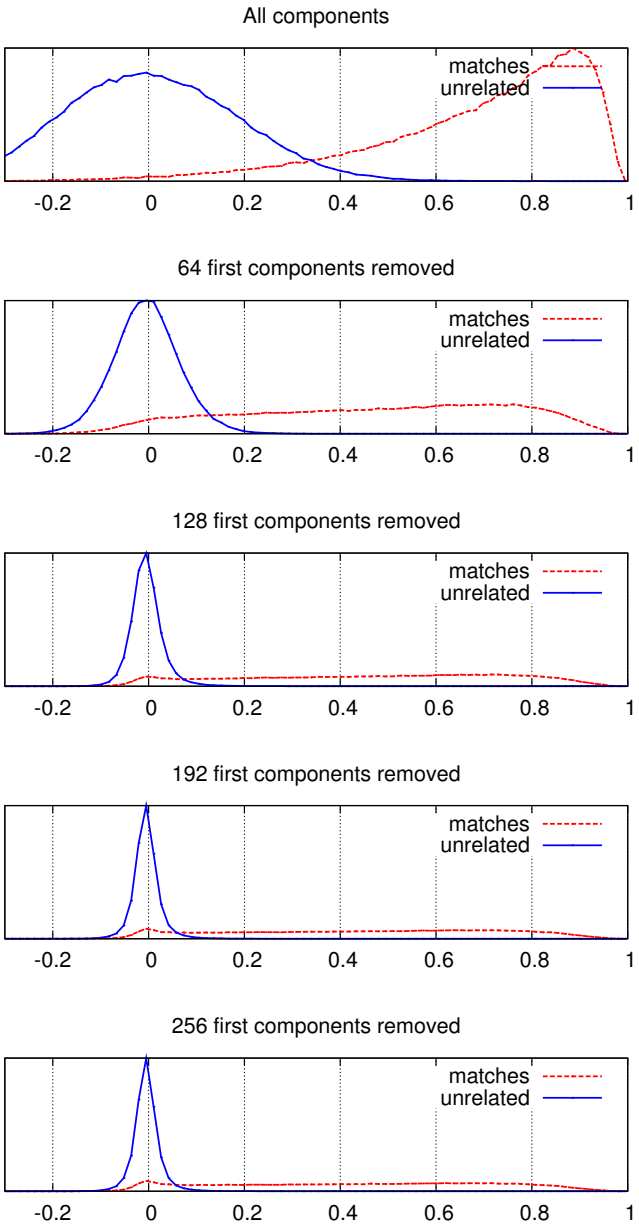

Figure 2. Impact on the cosine similarity distributions of removing highly-energetic components in our T-embedding (k=16).

Appendix C – Bag-of-visual-words: Link between democratization with square-root normalization

In this section, we discuss the particular case of our method when applied to the bag-of-word representation with cosine similarity. We do not consider inverse-document-frequency terms. In this case, ϕ_{BOW} is defined by

$$\phi_{\mathrm{BOW}}(x) = [0, \ldots, 0, 1, 0, \ldots, 0]^\top. \qquad (1)$$

These mapped vectors are summed up to produce the

bag-of-visual-words vector

$$\text{BOW}(\mathcal{X}) = \alpha(\mathcal{X}) \times [m_1(\mathcal{X}), \dots, m_j(\mathcal{X}), \dots, m_n(\mathcal{X})]^\top, \tag{2}$$

where $m_j(\mathcal{X})$ is the number of descriptors assigned to visual word j in $\mathcal{X}$.

The match kernel matrix $\mathsf{K}_{\text{BOW}}(\mathcal{X}, \mathcal{X})$ is, up to a permutation (to assume that the vectors are ordered by increasing visual word indices), block diagonal with only 1 in the blocks:

$$\mathsf{K}_{\text{BOW}}(\mathcal{X}, \mathcal{Y}) = \begin{array}{cccc} \boxed{\mathbf{1}} & \boxed{\mathbf{0}} & \dots & \boxed{\mathbf{0}} \\ \boxed{\mathbf{0}} & \boxed{\mathbf{1}} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \boxed{\mathbf{0}} & \dots & \dots & \boxed{\mathbf{1}} \end{array} \begin{array}{c} \updownarrow m_1 \\ \updownarrow m_2 \\ \vdots \\ \updownarrow m_n \end{array} \tag{3}$$

This matrix is positive, which means that the strategy to enforce only positive values has no effect. Similarly, the mapped vectors are already normalized to unit norm. A trivial solution to

$$\Lambda \mathsf{K} \Lambda \mathbf{1}_n = C \mathbf{1}_n, \tag{4}$$

is such that

$$\lambda(x) = 1/\sqrt{m_j} \tag{5}$$

for a vector assigned to visual word j such that $m_j \neq 0$. The resulting vector

$$\psi_{\text{d}}(\mathcal{X}) \propto [\sqrt{m_1}, \dots, \sqrt{m_i}, \dots, \sqrt{m_n}]^\top \tag{6}$$

is a democratic kernel that plainly satisfies the "democratic" condition of Equation 4. Interestingly, it is the same vector as the one obtained by applying square-root component-wise normalization [3, 5], which significantly improves bag-of-words performance.

Consider now the symmetric version of the Sinkhorn algorithm (Algorithm 1), where we set $\gamma = 0.5$. The first iteration computes the sum of each row. If the i^{th} vector is assigned to the visual word j, then the sum is m_j and $\lambda_i = 1/\sqrt{m_j}$. In other terms, the algorithm reaches the fixed-point in a single iteration. For other values of $\gamma < 0.5$, the algorithm also converges to this fixed point.

Appendix D – Parameter and method evaluation

We complement the parameter study of Section 5.3 by providing the same experiments performed on the Oxford5k dataset. We also include the variant method $\phi_\triangle + \psi_s + RN$ in the plot. The results are shown in Figure 3 for the vocabulary size $|\mathcal{C}|$ and in Figure 4 for the exponent α involved in power-law normalization.

The conclusions drawn in our main paper are identical for both datasets:

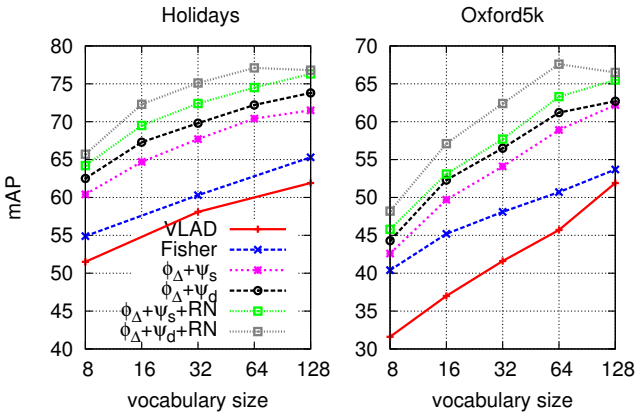

Figure 3. Impact of the vocabulary size $|\mathcal{C}|$ on the performance on Holidays and Oxford5k for several strategies. VLAD, Fisher, our T-embedding $\phi_\triangle$ with sum aggregation ψ_{s} and democratic pooling ψ_{d}. All methods use the same input descriptors with RootSIFT post-processing [1]. Parameter $\alpha = 0.5$.

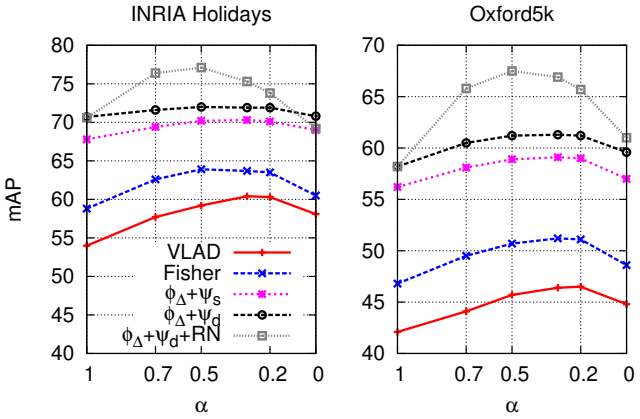

Figure 4. Impact of power-law normalization (parameter α) on the performance. We fix $|\mathcal{C}| = 64$ for all methods. Note that $\alpha = 0$ amounts to binarizing the vector.

- The performance grows as a function of the vocabulary size for all methods, apart from $|\mathcal{C}| = 128$ with the best variant. For this last point, a possible explanation is that the number of learning images (10k for Holidays, the 6k images from Paris for Oxford105k) is no large enough to accurately learn the rotation matrix.
- Our democratic aggregation strategy is complementary with power-law normalization, and improves upon our embedding;
- RN is worth applying and combining with power-law normalization in most cases.
- When using RN, the best performance is noticeably attained for $\alpha = 0.5$.

Table 1 gives a enlarged set of results for accuracy as a function of the vocabulary size on Oxford5k. The conclusion is consistent for all vocabulary sizes, except for ψ_{d}+RN which is better for $|\mathcal{C}| = 64$ than for $|\mathcal{C}| = 128$. How-

	Oxford5k			
$\|\mathcal{C}\|$	ψ_s	ψ_d	ψ_s+RN	ψ_d+RN
8	42.6 *±1.9*	44.3 *±2.0*	45.8 *±2.8*	48.2 *±2.2*
16	49.7 *±0.5*	52.3 *±0.3*	53.1 *±0.4*	57.1 *±1.0*
32	54.1 *±0.4*	56.5 *±0.4*	57.7 *±0.1*	62.4 *±0.7*
64	58.9 *±0.3*	61.2 *±0.4*	63.3 *±0.9*	67.5 *±0.2*
128	62.2 *±0.5*	62.7 *±0.9*	65.5 *±1.4*	66.5 *±1.9*
	Holidays			
$\|\mathcal{C}\|$	ψ_s	ψ_d	ψ_s+RN	ψ_d+RN
8	60.4 *±0.0*	62.5 *±0.2*	64.2 *±0.2*	65.7 *±0.1*
16	64.7 *±0.6*	67.3 *±0.4*	69.5 *±0.8*	72.3 *±0.6*
32	67.7 *±0.3*	69.8 *±0.4*	72.4 *±0.5*	75.1 *±0.5*
64	70.4 *±0.4*	72.2 *±0.2*	74.5 *±0.4*	77.1 *±0.7*
128	71.5 *±1.2*	73.8 *±0.8*	76.3 *±0.6*	76.8 *±1.3*

Table 1. Performance of different combinations. For all the methods, we report the average over 3 distinct vocabularies, which are the same for all the methods. We set $\alpha = 0.5$.

$\|\mathcal{C}\|$	UKB (score/4)	Holidays+Flickr1M (mAP)
D=1920	3.53	51.9
$\rightarrow$ 1024	3.51	49.4
$\rightarrow$ 512	3.49	46.9
$\rightarrow$ 256	3.45	43.7
$\rightarrow$ 128	3.40	38.7

Table 2. $\phi_\triangle + \psi_d$: Performance on UKB and Holidays+Flickr1M before (D=1920) and after dimensionality reduction to 1024, 512, 256 and 128 components. $|\mathcal{C}| = 16$. Note that, depending on the desired output dimensionality, the choice of $|\mathcal{C}| = 16$ is not optimal, as it depends on the target final dimensionality.

ever note the larger variance: On Oxford5k, the best result mAP=68.6% is actually obtained with a vocabulary of size $|\mathcal{C}| = 128$ (Best score with $|\mathcal{C}| = 64$: mAP=67.7%).

Appendix E – Performance on UKB and Holidays+Flickr1M

The performance of our best method, namely our T-embedding associated with Sinkhorn democratization, is presented in Table 2 for two additional datasets, namely

- **The University of Kentucky Benchmark** (UKB) [4]. This dataset contains 10,200 images, organized by group of 4 images. All images are submitted in turn. The performance measure is the average number of images returned in the first 4 positions.

- **Holidays+Flickr1M**: Following common practice [2], we merge the Inria Holidays dataset with another set of 1 million images retrieved from Flickr. The performance measure is mean average precision.

$\|\mathcal{C}\|$	$\phi_\triangle + \psi_s$	$\phi_\triangle + \psi_d$
8	0.086	1.754
16	0.096	2.603
32	0.107	4.720
64	0.147	18.740
128	0.237	56.753

Table 3. CPU timings (in seconds) for encoding. We do not include the cost of extracting the SIFT descriptors.

Appendix F – Complexity analysis

Table 3 reports the timings measured to compute our representations for different vocabulary sizes $\mathsf{k_c}$. The measures are obtained on the query images of Oxford5k. The measurements are carried out on a Xeon E5-2650/2.00GHz (32 cores). We report the CPU times (larger than elapsed ones because CPU time cumulates all active threads). On a quad-core laptop with multi-threading, the timing is typically 20ms per image for $\phi_\triangle + \psi_\mathrm{s}$.

The computation of $\phi_\triangle$ is fast. The bottleneck is democratic aggregation, when adopted. This is partially due to the low degree of optimization: Democratic aggregation is done in plain Matlab, while we have optimized the computation of $\phi_\triangle$ with a mex file. This also suggests that further optimization strategies should be considered for larger vocabularies. A simple effective one is to threshold the gram matrix by setting to 0 all values below a threshold (typically, 0.1), to get it sparse at a small accuracy cost.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

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

[3] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

[4] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[5] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010.

[6] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 2014.

[7] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, 2007.