Hybrid Diffusion: Spectral-Temporal Graph Filtering for Manifold Ranking

Ahmet Iscen[1] Yannis Avrithis[2] Giorgos Tolias[1]
Teddy Furon[2] Ondřej Chum[1]

[1]VRG, FEE, CTU in Prague [2]Univ Rennes, Inria, CNRS, IRISA

Abstract. State of the art image retrieval performance is achieved with CNN features and manifold ranking using a k-NN similarity graph that is pre-computed off-line. The two most successful existing approaches are *temporal filtering*, where manifold ranking amounts to solving a sparse linear system online, and *spectral filtering*, where eigen-decomposition of the adjacency matrix is performed off-line and then manifold ranking amounts to dot-product search online. The former suffers from expensive queries and the latter from significant space overhead. Here we introduce a novel, theoretically well-founded *hybrid filtering* approach allowing full control of the space-time trade-off between these two extremes. Experimentally, we verify that our hybrid method delivers results on par with the state of the art, with lower memory demands compared to spectral filtering approaches and faster compared to temporal filtering.

1 Introduction

Most image retrieval methods obtain their initial ranking of the database images by computing similarity between the query descriptor and descriptors of the database images. Descriptors based on local features [1, 2] have been largely replaced by more efficient CNN-based image descriptors [3, 4]. Regardless of the initial ranking, the retrieval performance is commonly boosted by considering the manifold structure of the database descriptors, rather than just independent distances of query to database images. Examples are query expansion [5, 6] and diffusion [7–9]. *Query expansion* uses the results of initial ranking to issue a novel, enriched, query [5] on-line only. *Diffusion* on the other hand, is based on the k-NN graph of the dataset that is constructed off-line, so that, assuming novel queries are part of the dataset, their results are essentially pre-computed. Diffusion can then be seen as infinite-order query expansion [8].

The significance of the performance boost achieved by diffusion has been recently demonstrated at the "Large-Scale Landmark Recognition"[1] challenge in conjunction with CVPR 2018. The vast majority of top-ranked teams have used query expansion or diffusion as the last step of their method.

Recently, efficient diffusion methods have been introduced to the image retrieval community. Iscen *et al*. [8] apply diffusion to obtain the final ranking,

[1] https://landmarkscvprw18.github.io/

in particular by solving a large and sparse system of linear equations. Even though an efficient *conjugate gradient* (CG) [10] solver is used, query times on large-scale datasets are in a range of several seconds. A significant speed-up is achieved by *truncating* the system of linear equations. Such an approximation, however, brings a slight degradation in the retrieval performance. Their method can be interpreted as graph filtering in the *temporal* domain.

In the recent work of Iscen *et al.* [9], more computation is shifted to the off-line phase to accelerate the query. The solution of the linear system is estimated by low-rank approximation of the k-NN graph Laplacian. Since the eigenvectors of the Laplacian represent a Fourier basis of the graph, this is interpreted as graph filtering in the *spectral* domain. The price to pay is increased space complexity to store the embeddings of the database descriptors. For comparable performance, a 5k-10k dimensional vector is needed per image.

In this paper, we introduce a *hybrid* method that combines spectral filtering [9] and temporal filtering [8]. This hybrid method offers a trade-off between speed (*i.e.*, the number of iterations of CG) and the additional memory required (*i.e.*, the dimensionality of the embedding). The two approaches [8, 9] are extreme cases of our hybrid method. We show that the proposed method pairs or outperforms the previous methods while either requiring less memory or being significantly faster – only three to five iterations of CG are necessary for embeddings of 100 to 500 dimensions.

While both temporal and spectral filtering approaches were known in other scientific fields before being successfully applied to image retrieval, to our knowledge the proposed method is novel and can be applied to other domains.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. Previous work on temporal and spectral filtering is detailed in Sections 3.1 and 3.2 respectively, since the paper builds on this work. The proposed method is described in Section 4 and its behavior is analyzed in Section 5. Experimental results are provided in Section 6. Conclusions are drawn in Section 7.

2 Related work

Query expansion (QE) has been a standard way to improve recall of image retrieval since the work of Chum *et al.* [5]. A variety of approaches exploit local feature matching and perform various types of verification. Such matching ranges from selective kernel matching [11] to geometric consensus [5, 12, 13] with RANSAC-like techniques. The verified images are then used to refine the global or local image representation of a novel query.

Another family of QE methods are more generic and simply assume a global image descriptor [14–18, 7, 6]. A simple and popular one is average-QE [5], recently extended to α-QE [4]. At some small extra cost, recall is significantly boosted. This additional cost is restricted to the on-line query phase. This is in contrast to another family of approaches that considers an off-line pre-processing of the database. Given the nearest neighbors list for database images, QE is performed by adapting the local similarity measure [15], using reciprocity constraints [16, 17] or graph-based similarity propagation [18, 7, 8]. The graph-based approaches, also known as diffusion, are shown to achieve great performance [8] and to be a good way for feature fusion [18]. Such on-line re-ranking is typically orders of magnitude more costly than simple average-QE.

The advent of CNN-based features, especially global image descriptors, made QE even more attractive. Average-QE or α-QE are easily applicable and very effective with a variety of CNN-based descriptors [4, 3, 19, 20]. State-of-the-art performance is achieved with diffusion on global or regional descriptors [8]. The latter is possible due to the small number of regions that are adequate to represent small objects, in contrast to thousands in the case of local features.

Diffusion based on tensor products can be attractive in terms of performance [21, 22]. However, in this work, we focus on the page-rank like diffusion [8, 23] due to its reasonable query times. An iterative on-line solution was commonly preferred [7] until the work of Iscen *et al.* [8], who solve a linear system to speed up the process. Additional off-line pre-processing and the construction and storage of additional embeddings reduce diffusion to inner product search in the spectral ranking of Iscen *et al.* [9]. This work lies exactly in between these two worlds and offers a trade-off exchanging memory for speed and vice versa.

Fast random walk with restart [24] is very relevant in the sense that it follows the same diffusion model as [8, 23] and is a hybrid method like ours. It first disconnects the graph into distinct components through clustering and then obtains a low-rank spectral approximation of the residual error. Apart from the additional complexity, parameters *etc.* of the off-line clustering process and the storage of both eigenvectors and a large inverse matrix, its online phase is also complex, involving the Woodbury matrix identity and several dense matrix-vector multiplications. Compared to that, we first obtain a *very* low-rank spectral approximation of the original graph, and then solve a sparse linear system of the residual error. Thanks to orthogonality properties, the online phase is nearly as simple as the original one and significantly faster.

3 Problem formulation and background

The methods we consider are based on a nearest neighbor graph of a dataset of n items, represented by $n \times n$ *adjacency matrix* W. The graph is undirected and weighted according to similarity: W is sparse, symmetric, nonnegative and zero-diagonal. We symmetrically normalize W as $\mathcal{W} := D^{-1/2} W D^{-1/2}$, where $D := \operatorname{diag}(W\mathbf{1})$ is the diagonal *degree matrix*, containing the row-wise sum of W on its diagonal. The eigenvalues of $\mathcal{W}$ lie in $[-1, 1]$ [25].

At query time, we are given a sparse $n \times 1$ *observation vector* $\mathbf{y}$, which is constructed by searching for the nearest neighbors of a query item in the dataset and setting its nonzero entries to the corresponding similarities. The problem is to obtain an $n \times 1$ *ranking vector* $\mathbf{x}$ such that retrieved items of the dataset are ranked by decreasing order of the elements of $\mathbf{x}$. Vector $\mathbf{x}$ should be close to $\mathbf{y}$ but at the same time similar items are encouraged to have similar ranks in $\mathbf{x}$, essentially by exploring the graph to retrieve more items.

3.1 Temporal filtering

Given a parameter $\alpha \in [0,1)$, define the $n \times n$ *regularized Laplacian* function by

$$\mathcal{L}_\alpha(A) := (I_n - \alpha A)/(1-\alpha) \tag{1}$$

for $n \times n$ real symmetric matrix A, where I_n is the $n \times n$ identity matrix. Iscen *et al.* [8] then define $\mathbf{x}$ as the unique solution of the linear system

$$\mathcal{L}_\alpha(\mathcal{W})\mathbf{x} = \mathbf{y}, \tag{2}$$

which they obtain approximately in practice by a few iterations of the *conjugate gradient* (CG) method, since $\mathcal{L}_\alpha(\mathcal{W})$ is positive-definite. At large scale, they truncate $\mathcal{W}$ by keeping only the rows and columns corresponding to a fixed number of neighbors of the query, and re-normalize. Then, (2) only performs re-ranking within this neighbor set.

We call this method *temporal*[2] *filtering* because if $\mathbf{x}$, $\mathbf{y}$ are seen as signals, then $\mathbf{x}$ is the result of applying a linear graph filter on $\mathbf{y}$, and CG iteratively applies a set of recurrence relations that determine the filter. While $\mathcal{W}$ is computed and stored off-line, (2) is solved online (at query time), and this is expensive.

3.2 Spectral filtering

Linear system (2) can be written as $\mathbf{x} = h_\alpha(\mathcal{W})\mathbf{y}$, where the *transfer function* h_α is defined by

$$h_\alpha(A) := \mathcal{L}_\alpha(A)^{-1} = (1-\alpha)(I_n - \alpha A)^{-1} \tag{3}$$

for $n \times n$ real symmetric matrix A. Given the eigenvalue decomposition $\mathcal{W} = U\Lambda U^\top$ of the symmetric matrix $\mathcal{W}$, Iscen *et al.* [9] observe that $h_\alpha(\mathcal{W}) = U h_\alpha(\Lambda) U^\top$, so that (2) can be written as

$$\mathbf{x} = U h_\alpha(\Lambda) U^\top \mathbf{y}, \tag{4}$$

which they approximate by keeping only the largest r eigenvalues and the corresponding eigenvectors of $\mathcal{W}$. This defines a low-rank approximation $h_\alpha(\mathcal{W}) \approx U_1 h_\alpha(\Lambda_1) U_1^\top$ instead. This method is referred to as *fast spectral ranking* (FSR) in [9]. Crucially, Λ is a diagonal matrix, hence h_α applies element-wise, as a scalar function $h_\alpha(x) := (1-\alpha)/(1-\alpha x)$ for $x \in [-1,1]$.

At the expense of off-line computing and storing the $n \times r$ matrix U_1 and the r eigenvalues in Λ_1, filtering is now computed as a sequence of low-rank matrix-vector multiplications, and the query is accelerated by orders of magnitude compared to [8]. However, the space overhead is significant. To deal with approximation errors when r is small, a heuristic is introduced that gradually falls back to the similarity in the original space for items that are poorly represented, referred to as FSRw [9].

[2] "Temporal" stems from conventional signal processing where signals are functions of "time"; while "spectral" is standardized also in graph signal processing.

We call this method *spectral filtering* because U represents the Fourier basis of the graph and the sequence of matrix-vector multiplications from right to left in the right-hand side of (4) represents the Fourier transform of $\mathbf{y}$ by $U^\top$, filtering in the frequency domain by $h_\alpha(\Lambda)$, and finally the inverse Fourier transform to obtain $\mathbf{x}$ in the time domain by U.

4 Hybrid spectral-temporal filtering

Temporal filtering (2) is performed once for every new query represented by $\mathbf{y}$, but $\mathcal{W}$ represents the dataset and is fixed. Could CG be accelerated if we had some very limited additional information on $\mathcal{W}$?

On the other extreme, spectral filtering (4) needs a large number of eigenvectors and eigenvalues of $\mathcal{W}$ to provide a high quality approximation, but always leaves some error. Could we reduce this space requirement by allocating some additional query time to recover the approximation error?

The answer is positive to both questions and in fact these are the two extreme cases of *hybrid spectral-temporal filtering*, which we formulate next.

4.1 Derivation

We begin with the eigenvalue decomposition $\mathcal{W} = U \Lambda U^\top$, which we partition as $\Lambda = \mathrm{diag}(\Lambda_1, \Lambda_2)$ and $U = (U_1 \; U_2)$. Matrices Λ_1 and Λ_2 are diagonal $r \times r$ and $(n-r) \times (n-r)$, respectively. Matrices U_1 and U_2 are $n \times r$ and $n \times (n-r)$, respectively, and have the following orthogonality properties, all due to the orthogonality of U itself:

$$U_1^\top U_1 = I_r, \quad U_2^\top U_2 = I_{n-r}, \quad U_1^\top U_2 = \mathbf{O}, \quad U_1 U_1^\top + U_2 U_2^\top = I_n. \tag{5}$$

Then, $\mathcal{W}$ is decomposed as

$$\mathcal{W} = U_1 \Lambda_1 U_1^\top + U_2 \Lambda_2 U_2^\top. \tag{6}$$

Similarly, $h_\alpha(\mathcal{W})$ is decomposed as

$$\begin{aligned} h_\alpha(\mathcal{W}) &= U h_\alpha(\Lambda) U^\top &(7)\\ &= U_1 h_\alpha(\Lambda_1) U_1^\top + U_2 h_\alpha(\Lambda_2) U_2^\top, &(8)\end{aligned}$$

which is due to the fact that diagonal matrix $h_\alpha(\Lambda)$ is obtained element-wise, hence decomposed as $h_\alpha(\Lambda) = \mathrm{diag}(h_\alpha(\Lambda_1), h_\alpha(\Lambda_2))$. Here the first term is exactly the low-rank approximation that is used by spectral filtering, and the second is the approximation error

$$\begin{aligned} e_\alpha(\mathcal{W}) &:= U_2 h_\alpha(\Lambda_2) U_2^\top &(9)\\ &= (1-\alpha)\left(U_2 (I_{n-r} - \alpha \Lambda_2)^{-1} U_2^\top + U_1 U_1^\top - U_1 U_1^\top\right) &(10)\\ &= (1-\alpha)\left(\left(U_2 (I_{n-r} - \alpha \Lambda_2) U_2^\top + U_1 U_1^\top\right)^{-1} - U_1 U_1^\top\right) &(11)\\ &= (1-\alpha)\left(\left(I_n - \alpha U_2 \Lambda_2 U_2^\top\right)^{-1} - U_1 U_1^\top\right) &(12)\\ &= h_\alpha(U_2 \Lambda_2 U_2^\top) - (1-\alpha) U_1 U_1^\top. &(13)\end{aligned}$$

We have used the definition (3) of h_α in (10) and (13). Equation (12) is due to the orthogonality properties (5). Equation (11) follows from the fact that for any invertible matrices A, B of conformable sizes,

$$(U_1 A U_1 + U_2 B U_2)^{-1} = U_1 A^{-1} U_1 + U_2 B^{-1} U_2, \tag{14}$$

which can be verified by direct multiplication, and is also due to orthogonality.

Now, combining (8), (13) and (6), we have proved the following.

Theorem 1. *Assuming the definition (3) of transfer function h_α and the eigenvalue decomposition (6) of the symmetrically normalized adjacency matrix $\mathcal{W}$, $h_\alpha(\mathcal{W})$ is decomposed as*

$$h_\alpha(\mathcal{W}) = U_1 g_\alpha(\Lambda_1) U_1^\top + h_\alpha(\mathcal{W} - U_1 \Lambda_1 U_1^\top), \tag{15}$$

where

$$g_\alpha(A) := h_\alpha(A) - h_\alpha(\mathbf{O}) = (1-\alpha)\left((I_n - \alpha A)^{-1} - I_n\right) \tag{16}$$

for $n \times n$ real symmetric matrix A. For $x \in [-1, 1]$ in particular, $g_\alpha(x) := h_\alpha(x) - h_\alpha(0) = (1-\alpha)\alpha x/(1-\alpha x)$.

Observe that Λ_2, U_2 do not appear in (15) and indeed it is only the largest r eigenvalues Λ_1 and corresponding eigenvectors U_1 of $\mathcal{W}$ that we need to compute. The above derivation is generalized from h_α to a much larger class of functions in appendix A.

4.2 Algorithm

Why is decomposition (15) of $h_\alpha(\mathcal{W})$ important? Because given an observation vector $\mathbf{y}$ at query time, we can express the ranking vector $\mathbf{x}$ as

$$\mathbf{x} = \mathbf{x}^s + \mathbf{x}^t, \tag{17}$$

where the first, *spectral*, term $\mathbf{x}^s$ is obtained by spectral filtering

$$\mathbf{x}^s = U_1 g_\alpha(\Lambda_1) U_1^\top \mathbf{y}, \tag{18}$$

as in [9], where g_α applies element-wise, while the second, *temporal*, term $\mathbf{x}^t$ is obtained by temporal filtering, that is, solving the linear system

$$\mathcal{L}_\alpha(\mathcal{W} - U_1 \Lambda_1 U_1^\top)\mathbf{x}^t = \mathbf{y}, \tag{19}$$

which we do by a few iterations of CG as in [8]. The latter is possible because $\mathcal{L}_\alpha(\mathcal{W} - U_1 \Lambda_1 U_1^\top)$ is still positive-definite, like $\mathcal{L}_\alpha(\mathcal{W})$. It's also possible without an explicit dense representation of $U_1 \Lambda_1 U_1^\top$ because CG, like all Krylov subspace methods, only needs *black-box* access to the matrix A of the linear system, that is, a mapping $\mathbf{z} \mapsto A\mathbf{z}$ for $\mathbf{z} \in \mathbb{R}^n$. For system (19) in particular, according to the definition (1) of $\mathcal{L}_\alpha$, we use the mapping

$$\mathbf{z} \mapsto \left(\mathbf{z} - \alpha\left(\mathcal{W}\mathbf{z} - U_1 \Lambda_1 U_1^\top \mathbf{z}\right)\right)/(1-\alpha), \tag{20}$$

where product $\mathcal{W}\mathbf{z}$ is efficient because $\mathcal{W}$ is sparse as in [8], while $U_1 \Lambda_1 U_1^\top \mathbf{z}$ is efficient if computed right-to-left because U_1 is an $n \times r$ matrix with $r \ll n$ and Λ_1 is diagonal as in [9].

4.3 Discussion

What is there to gain from spectral-temporal decomposition (17) of $\mathbf{x}$*?*

First, since the temporal term (19) can recover the spectral approximation error, the rank r of U_1, Λ_1 in the spectral term (18) can be chosen as small as we like. In the extreme case $r = 0$, the spectral term vanishes and we recover temporal filtering [8]. This allows efficient computation of only a few eigenvectors/values, even with the Lanczos algorithm rather than the approximate method of [9]. Most importantly, it reduces significantly the space complexity, at the expense of query time. Like spectral filtering, it is possible to *sparsify* U_1 to compress the dataset embeddings and accelerate the queries online. In fact, we show in section 6 that sparsification is much more efficient in our case.

Second, the matrix $\mathcal{W} - U_1 \Lambda_1 U_1^\top$ is effectively like $\mathcal{W}$ with the r largest eigenvalues removed. This improves significantly the condition number of matrix $\mathcal{L}_\alpha(\mathcal{W} - U_1 \Lambda_1 U_1^\top)$ in the temporal term (19) compared to $\mathcal{L}_\alpha(\mathcal{W})$ in the linear system (2) of temporal filtering [8], on which the convergence rate depends. In the extreme case $r = n$, the temporal term vanishes and we recover spectral filtering [9]. In turn, even with small r, this reduces significantly the number of iterations needed for a given accuracy, at the expense of computing and storing U_1, Λ_1 off-line as in [9]. The improvement is a function of α and the spectrum of $\mathcal{W}$, and is quantified in section 5.

In summary, for a given desired accuracy, we can choose the rank r of the spectral term and a corresponding number of iterations of the temporal term, determining a trade-off between the space needed for the eigenvectors (and the off-line cost to obtain them) and the (online) query time. Such choice is not possible with either spectral or temporal filtering alone: at large scale, the former may need too much space and the latter may be too slow.

5 Analysis

How "easier" for CG is $\mathcal{W} - U_1 \Lambda_1 U_1^\top$ *in (19) compared to* $\mathcal{W}$ *in (2)?*

In solving a linear system $A\mathbf{x} = \mathbf{b}$ where A is an $n \times n$ symmetric positive-definite matrix, CG generates a unique sequence of iterates $\mathbf{x}_i$, $i = 0, 1, \dots$, such that the A-norm $\|\mathbf{e}_i\|_A$ of the error $\mathbf{e}_i := \mathbf{x}^* - \mathbf{x}_i$ is minimized over the Krylov subspace $\mathcal{K}_i := \langle \mathbf{b}, A\mathbf{b}, \dots, A^i \mathbf{b} \rangle$ at each iteration i, where $\mathbf{x}^* := A^{-1}\mathbf{b}$ is the exact solution and the A-norm is defined by $\|\mathbf{x}\|_A := \sqrt{\mathbf{x}^\top A \mathbf{x}}$ for $\mathbf{x} \in \mathbb{R}^n$.

A well-known result on the rate of convergence of CG that assumes minimal knowledge of the eigenvalues of A states that the A-norm of the error at iteration i, relative to the A-norm of the initial error $\mathbf{e}_0 := \mathbf{x}^*$, is upper-bounded by [26]

$$\frac{\|\mathbf{e}_i\|_A}{\|\mathbf{e}_0\|_A} \leq \phi_i(A) := 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^i, \tag{21}$$

where $\kappa(A) := \|A\| \left\|A^{-1}\right\| = \lambda_1(A)/\lambda_n(A)$ is the 2-norm *condition number* of A, and $\lambda_j(A)$ for $j = 1, \dots, n$ are the eigenvalues of A in descending order.

In our case, matrix $\mathcal{L}_\alpha(\mathcal{W})$ of linear system (2) has condition number

$$\kappa(\mathcal{L}_\alpha(\mathcal{W})) = \frac{1 - \alpha\lambda_n(\mathcal{W})}{1 - \alpha\lambda_1(\mathcal{W})} = \frac{1 - \alpha\lambda_n(\mathcal{W})}{1 - \alpha}. \tag{22}$$

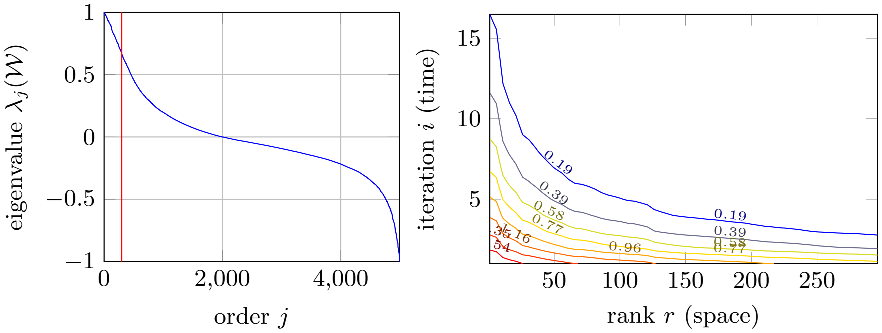
The first equality holds because for each eigenvalue λ of $\mathcal{W}$ there is a corresponding eigenvalue $(1 - \alpha\lambda)/(1 - \alpha)$ of $\mathcal{L}_\alpha(\mathcal{W})$, which is a decreasing function. The second holds because $\lambda_1(\mathcal{W}) = 1$ [25].

Now, let $\mathcal{W}_r := \mathcal{W} - U_1 \Lambda_1 U_1^\top$ for $r = 0, 1, \dots, n-1$, where Λ_1, U_1 represent the largest r eigenvalues and the corresponding eigenvectors of $\mathcal{W}$ respectively. Clearly, $\mathcal{W}_r$ has the same eigenvalues as $\mathcal{W}$ except for the largest r, which are replaced by zero. That is, $\lambda_1(\mathcal{W}_r) = \lambda_{r+1}(\mathcal{W})$ and $\lambda_n(\mathcal{W}_r) = \lambda_n(\mathcal{W})$. The latter is due to the fact that $\lambda_n(\mathcal{W}) \le -1/(n-1) \le 0$ [25], so the new zero eigenvalues do not affect the smallest one. Then,

$$\kappa(\mathcal{L}_\alpha(\mathcal{W}_r)) = \frac{1 - \alpha\lambda_n(\mathcal{W})}{1 - \alpha\lambda_{r+1}(\mathcal{W})} \le \kappa(\mathcal{L}_\alpha(\mathcal{W})). \tag{23}$$

This last expression generalizes (22). Indeed, $\mathcal{W} = \mathcal{W}_0$. Then, our hybrid spectral-temporal filtering involves CG on $\mathcal{L}_\alpha(\mathcal{W}_r)$ for $r \ge 0$, compared to the baseline temporal filtering for $r = 0$. The inequality in (23) is due to the fact that $|\lambda_j(\mathcal{W})| \le 1$ for $j = 1, \dots, n$ [25]. Removing the largest r eigenvalues of $\mathcal{W}$ clearly improves (decreases) the condition number of $\mathcal{L}_\alpha(\mathcal{W}_r)$ relative to $\mathcal{L}_\alpha(\mathcal{W})$. The improvement is dramatic given that α is close to 1 in practice. For $\alpha = 0.99$ and $\lambda_{r+1}(\mathcal{W}) = 0.7$ for instance, $\kappa(\mathcal{L}_\alpha(\mathcal{W}_r))/\kappa(\mathcal{L}_\alpha(\mathcal{W})) = 0.0326$.

More generally, given the eigenvalues $\lambda_{r+1}(\mathcal{W})$ and $\lambda_n(\mathcal{W})$, the improvement can be estimated by measuring the upper bound $\phi_i(\mathcal{L}_\alpha(\mathcal{W}_r))$ for different i and r. A concrete example is shown in Figure 1, where we measure the eigenvalues of



(a) (b)

Fig. 1. (a) In descending order, eigenvalues of adjacency matrix $\mathcal{W}$ of Oxford5k dataset of $n = 5,063$ images with global GeM features by ResNet101 and $k = 50$ neighbors per point (see section 6). Eigenvalues on the left of vertical red line at $j = 300$ are the largest 300 ones, candidate for removal. (b) Contour plot of upper bound $\phi_i(\mathcal{L}_\alpha(\mathcal{W}_r))$ of CG's relative error as a function of rank r and iteration i for $\alpha = 0.99$, illustrating the space(r)-time(i) trade-off for constant relative error.

the adjacency matrix $\mathcal{W}$ of a real dataset, remove the largest r for $0 \leq r \leq 300$ and plot the upper bound $\phi_i(\mathcal{L}_\alpha(\mathcal{W}_r))$ of the relative error as a function of rank r and iteration i given by (21) and (23). Clearly, as more eigenvalues are removed, less CG iterations are needed to achieve the same relative error; the approximation error represented by the temporal term decreases and at the same time the linear system becomes easier to solve. Of course, iterations become more expensive as r increases; precise timings are given in section 6.

6 Experiments

In this section we evaluate our hybrid method on popular image retrieval benchmarks. We provide comparisons to baseline methods, analyze the trade-off between runtime complexity, memory footprint and search accuracy, and compare with the state of the art.

6.1 Experimental setup

Datasets. We use the revisited retrieval benchmark [27] of the popular Oxford buildings [2] and Paris [28] datasets, referred to as $\mathcal{R}$Oxford and $\mathcal{R}$Paris, respectively. Unless otherwise specified, we evaluate using the *Medium* setup and always report mean Average Precision (mAP). Large-scale experiments are conducted on $\mathcal{R}$Oxford +$\mathcal{R}$1M and $\mathcal{R}$Paris +$\mathcal{R}$1M by adding the new 1M challenging distractor set [27].

Image representation. We use GeM descriptors [4] to represent images. We extract GeM at 3 different image scales, aggregate the 3 descriptors, and perform whitening, exactly as in [4]. Finally, each image is represented by a single vector with $d = 2048$ dimensions, since ResNet-101 architecture is used.

Baseline methods. We consider the two baseline methods described in Section 3, namely temporal and spectral filtering. *Temporal filtering* corresponds to solving a linear system with CG [8] and is evaluated for different numbers of CG iterations. It is used with truncation at large scale to speed up the search [8] and is denoted by *Temporal*†. *Spectral filtering* corresponds to FSR and its FSRw variant [9]. Both FSR variants are parametrized by the rank r of the approximation, which is equal to the dimensionality of the spectral embedding.

Implementation details. Temporal ranking is performed with the implementation[3] provided by Iscen *et al.* [8]. The adjacency matrix is constructed by using top $k = 50$ reciprocal neighbors. Pairwise similarity between descriptors $\mathbf{v}$ and $\mathbf{z}$ is estimated by $(\mathbf{v}^\top \mathbf{z})^3_+$. Parameter α is set to 0.99, while the observation vector $\mathbf{y}$ includes the top 5 neighbors. The eigendecomposition is performed on the largest connected component, as in [9]. Its size is 933,412 and 934,809 for $\mathcal{R}$Oxford +$\mathcal{R}$1M and $\mathcal{R}$Paris +$\mathcal{R}$1M, respectively. Timings are measured with Matlab implementation on a 4-core Intel Xeon 2.60GHz CPU with 200 GB of RAM. We only report timings for the diffusion part of the ranking and exclude the initial nearest neighbor search used to construct the observation vector.

[3] https://github.com/ahmetius/diffusion-retrieval/

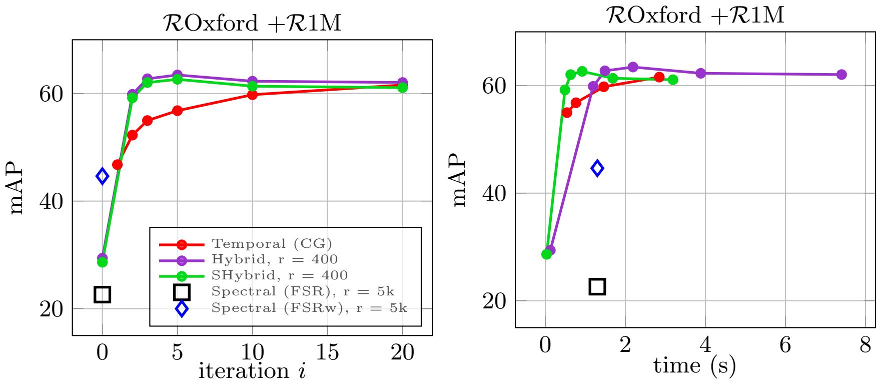


Fig. 2. mAP *vs.* CG iteration i and mAP *vs.* time for temporal, spectral, and hybrid filtering. Sparsified hybrid is used with sparsity 99%.

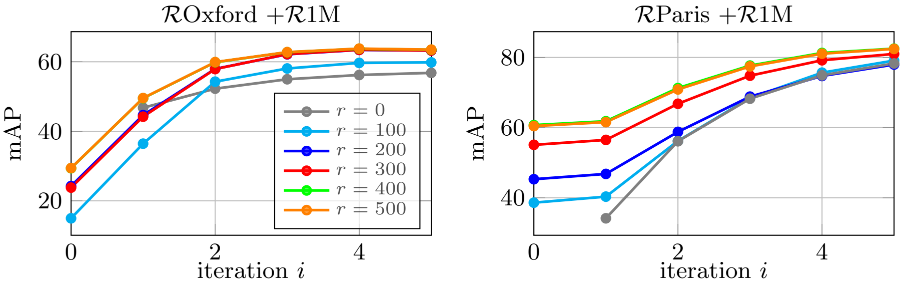


Fig. 3. mAP *vs.* CG iteration i for different rank r for our hybrid method, where $r = 0$ means temporal only.

6.2 Comparisons

Comparison with baseline methods. We first compare performance, query time and required memory for temporal, spectral, and hybrid ranking. With respect to the memory, all methods store the initial descriptors, *i.e.* one 2048-dimensional vector per image. Temporal ranking additionally stores the sparse regularized Laplacian. Spectral ranking stores for each vector an additional embedding of dimensionality equal to rank r, which is a parameter of the method. Our hybrid method stores both the Laplacian and the embedding, but with significantly lower rank r.

We evaluate on $\mathcal{R}$Oxford +$\mathcal{R}$1M and $\mathcal{R}$Paris +$\mathcal{R}$1M with global image descriptors, which is a challenging large scale problem, *i.e.* large adjacency matrix, where prior methods fail or have serious drawbacks. Results are shown in Figure 2. Temporal ranking with CG is reaching saturation near 20 iterations as in [8]. Spectral ranking is evaluated for a decomposition of rank r whose computation and required memory are reasonable and feasible on a single machine. Finally, the proposed hybrid method is evaluated for rank $r = 400$, which is a good compromise of speed and memory, as shown below.

	Temporal [8]	Temporal† [8]	Spectral (FSRw) [9]	SHybrid
mAP	61.6	59.0	42.1	62.6
Time (s)	2.8	1.0	1.3	0.9
Memory (MB)	205	205	35,606	264

Table 1. Performance, memory and query time comparison on $\mathcal{R}$Oxford +$\mathcal{R}$1M with GeM descriptors for temporal (20 iterations), truncated temporal (20 iterations, 75k images in shortlist), spectral ($r = 5k$), and hybrid ranking ($r = 400$, 5 iterations). Hybrid ranking is sparsified by setting the 99% smallest values to 0. Reported memory excludes the initial descriptors requiring 8.2 GB. U_1 is stored with double precision.

Spectral ranking variants (FSR, FSRw) are not performing well despite requiring about 250% additional memory compared to nearest neighbor search in the original space. Compared to hybrid, more than one order of magnitude higher rank r is required for problems of this scale. Temporal ranking achieves good performance but at much more iterations and higher query times. Our hybrid solution provides a very reasonable space-time trade-off.

Runtime and memory trade-off. We report the trade-off between number of iterations and rank r, representing additional memory, in more detail in Figure 3. It is shown that the number of iterations to achieve the top mAP decreases as the rank increases. We achieve the optimal trade-off at $r = 400$ where we only need 5 or less iterations. Note that, the rank not only affects the memory and the query time of the spectral part in a linear manner, but the query time of the temporal part too (20).

Sparsification of spectral embeddings is exploited in prior work [9]. We sparsify the embeddings of our hybrid method by setting the smallest values of U_1 to zero until a desired level of sparsity is reached. We denote this method by *SHybrid*. This sparse variant provides memory savings and an additional speedup due to the computations with sparse matrices. Figure 4 shows that performance loss remains small even for extreme sparsity *e.g.* 99%, while the results in Figure 2 show that it offers a significant speedup in the global descriptor setup.

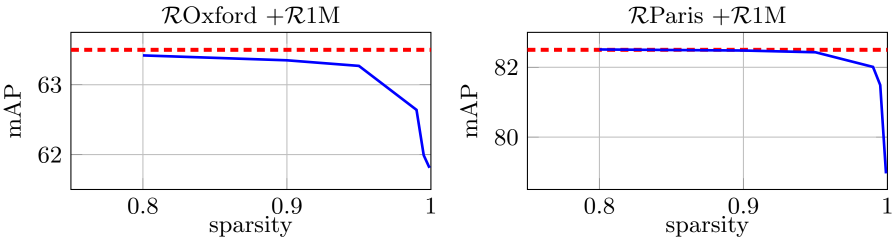


Fig. 4. mAP *vs.* level of sparsification on our hybrid method for $r = 400$. Dashed horizontal line indicates no sparsification.

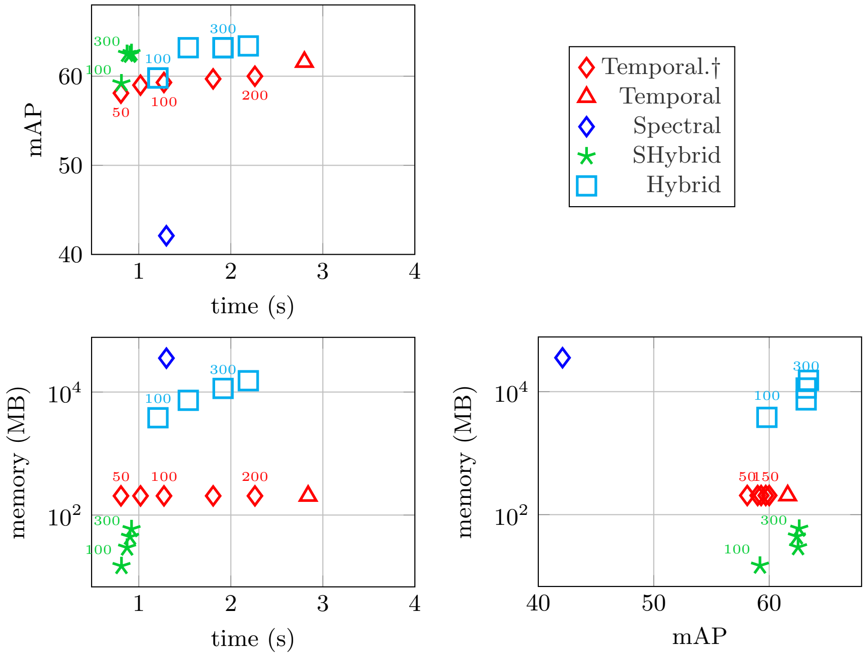


Fig. 5. Time (s) - memory (MB) - performance (mAP) for different methods. We show mAP *vs*. time, memory *vs*. time, and memory *vs*. mAP on $\mathcal{R}$Oxford +$\mathcal{R}$1M. Methods in the comparison: temporal for 20 iterations, truncated temporal for 20 iterations and shortlist of size 50k, 75k, 100k, 200k and 300k, spectral (FSRw) with $r = 5k$, hybrid with $r \in \{100, 200, 300, 400\}$ and 5 iterations, sparse hybrid with 99% sparsity, $r \in \{100, 200, 300, 400\}$ and 5 iterations. Text labels indicate the shortlist size (in thousands) for truncated temporal and rank for hybrid. Observe that the two plots on the left are aligned horizontally with respect to time, while the two at the bottom vertically with respect to memory.

Performance-memory-speed comparison with the baselines is shown in Table 1. Our hybrid approach enjoys query times lower than those of temporal with truncation or spectral with FSRw, while at the same time achieves higher performance and requires less memory than the spectral-only approach.

We summarize our achievement in terms of mAP, required memory, and query time in Figure 5. Temporal ranking achieves high performance at the cost of high query time and its truncated counterpart saves query time but sacrifices performance. Spectral ranking is not effective at this scale, while our hybrid solution achieves high performance at low query times.

Comparison with the state of the art. We present an extensive comparison with existing methods in the literature for global descriptors at small and large scale (1M distractors). We choose $r = 400$ and 5 iterations for our hybrid method, 20 iterations for temporal ranking, $r = 2k$ and $r = 5k$ for spectral ranking at small and large scale, respectively. Temporal ranking is also performed with

	$\mathcal{R}$Oxford		$\mathcal{R}$Oxf +$\mathcal{R}$1M		$\mathcal{R}$Paris		$\mathcal{R}$Paris +$\mathcal{R}$1M	
	Medium	Hard	Medium	Hard	Medium	Hard	Medium	Hard
NN-search	64.7	38.5	45.2	19.9	77.2	56.3	52.3	24.7
α-QE [27]	67.2	40.8	49.0	24.2	80.7	61.8	58.0	31.0
Temporal [8]	69.9	40.4	61.6	33.2	88.9	78.5	85.0	71.6
Temporal† [8]	-	-	59.0	31.4	-	-	79.7	65.2
FSR [9]	70.4	42.0	22.6	5.8	88.6	77.9	66.6	40.2
FSRw [9]	70.7	42.2	42.1	18.8	88.7	78.0	77.4	59.7
SHybrid (ours)	70.5	40.3	62.6	34.4	88.7	78.1	82.0	66.8

Table 2. mAP comparison with existing methods in the literature on small and large scale datasets, using Medium and Hard setup of the revisited benchmark.

truncation on a shortlist size of $75k$ images at large scale. The comparison is presented in Table 2. Our hybrid approach performs the best or second best right after the temporal one, while providing much smaller query times at a small amount of additional required memory.

7 Conclusions

In this work we have tested the two most successful manifold ranking methods of temporal filtering [8] and spectral filtering [9] on the very challenging new benchmark of Oxford and Paris datasets [27]. It is the first time that such methods are evaluated at the scale of one million images. *Spectral filtering*, with both its FSR and FSRw variants, fails at this scale, despite the significant space required for additional vector embeddings. It is possible that a higher rank would work, but it wouldn't be practical in terms of space. In terms of query time, *temporal filtering* is only practical with its truncated variant at this scale. It works pretty well in terms of performance, but the query time is still high.

Our new *hybrid filtering* method allows for the first time to strike a reasonable balance between the two extremes. Without truncation, it outperforms temporal filtering while being significantly faster, and its memory overhead is one order of magnitude less than that of spectral filtering. Unlike spectral filtering, it is possible to extremely sparsify the dataset embeddings with only negligible drop in performance. This, together with its very low rank, makes our hybrid method even faster than spectral, despite being iterative. More importantly, while previous methods were long known in other fields before being applied to image retrieval, to our knowledge our hybrid method is novel and can apply *e.g.* to any field where graph signal processing applies and beyond. Our theoretical analysis shows exactly why our method works and quantifies its space-time-accuracy trade-off using simple ideas from numerical linear algebra.

Acknowledgements: This work was supported by MSMT LL1303 ERC-CZ grant and the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

A General derivation

The derivation of our algorithm in section 4.1 applies only to the particular function (filter) h_α (3). Here, as in [9], we generalize to a much larger class of functions, that is, any function h that has a series expansion

$$h(A) = \sum_{i=0}^{\infty} c_i A^i. \tag{24}$$

We begin with the same eigenvalue decomposition (6) of $\mathcal{W}$ and, assuming that $h(\mathcal{W})$ converges absolutely, its corresponding decomposition

$$h(\mathcal{W}) = U_1 h(\Lambda_1) U_1^\top + U_2 h(\Lambda_2) U_2^\top, \tag{25}$$

similar to (8), where U_1, U_2 have the same orthogonality properties (5).

Again, the first term is exactly the low-rank approximation that is used by spectral filtering, and the second is the approximation error

$$\begin{align} e_\alpha(\mathcal{W}) &:= U_2 h(\Lambda_2) U_2^\top \tag{26} \\ &= \sum_{i=0}^{\infty} c_i U_2 \Lambda_2^i U_2^\top \tag{27} \\ &= \sum_{i=0}^{\infty} c_i \left(U_2 \Lambda_2 U_2^\top \right)^i - c_0 U_1 U_1^\top \tag{28} \\ &= h(U_2 \Lambda_2 U_2^\top) - h(0) U_1 U_1^\top. \tag{29} \end{align}$$

Again, we have used the series expansion (24) of h in (27) and (29). Now, equation (28) is due to the fact that

$$(U_2 \Lambda_2 U_2^\top)^i = U_2 \Lambda_2^i U_2^\top \tag{30}$$

for $i \geq 1$, as can be verified by induction, while for $i = 0$,

$$U_2 \Lambda_2^0 U_2^\top = U_2 U_2^\top = I_n - U_1 U_1^\top = (U_2 \Lambda_2 U_2^\top)^0 - U_1 U_1^\top. \tag{31}$$

In both (30) and (31) we have used the orthogonality properties (5).

Finally, combining (25), (29) and (6), we have proved the following.

Theorem 2. *Assuming the series expansion (24) of transfer function h and the eigenvalue decomposition (6) of the symmetrically normalized adjacency matrix $\mathcal{W}$, and given that $h(\mathcal{W})$ converges absolutely, it is decomposed as*

$$h(\mathcal{W}) = U_1 g(\Lambda_1) U_1^\top + h(\mathcal{W} - U_1 \Lambda_1 U_1^\top), \tag{32}$$

where

$$g(A) := h(A) - h(\mathbf{O}) \tag{33}$$

for $n \times n$ real symmetric matrix A. For $h = h_\alpha$ and for $x \in [-1, 1]$ in particular, $g_\alpha(x) := h_\alpha(x) - h_\alpha(0) = (1-\alpha)\alpha x/(1-\alpha x)$.

This general derivation explains where the general definition of function g (33) is coming from in (16) corresponding to our treatment of h_α in section 4.1.

References

1. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV. (2003)
2. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
3. Gordo, A., Almazan, J., Revaud, J., Larlus, D.: End-to-end learning of deep visual representations for image retrieval. IJCV **124** (2017)
4. Radenović, F., Tolias, G., Chum, O.: Fine-tuning CNN image retrieval with no human annotation. IEEE Trans. PAMI (2018)
5. 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)
6. Arandjelovic, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: CVPR. (2012)
7. Donoser, M., Bischof, H.: Diffusion processes for retrieval revisited. In: CVPR. (2013)
8. Iscen, A., Tolias, G., Avrithis, Y., Furon, T., Chum, O.: Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In: CVPR. (2017)
9. Iscen, A., Avrithis, Y., Tolias, G., Furon, T., Chum, O.: Fast spectral ranking for similarity search. In: CVPR. (2018)
10. Hackbusch, W.: Iterative solution of large sparse systems of equations. Springer Verlag (1994)
11. Tolias, G., Jégou, H.: Visual query expansion with or without geometry: refining local descriptors by feature aggregation. Pattern recognition **47** (2014) 3466–3476
12. Chum, O., Mikulik, A., Perdoch, M., Matas, J.: Total recall II: Query expansion revisited. In: CVPR. (2011)
13. Joly, A., Buisson, O.: Logo retrieval with a contrario visual query expansion. In: ACM Multimedia. (2009)
14. Shen, X., Lin, Z., Brandt, J., Wu, Y.: Spatially-constrained similarity measure for large-scale object retrieval. IEEE Trans. PAMI **36** (2014) 1229–1241
15. Jégou, H., Harzallah, H., Schmid, C.: A contextual dissimilarity measure for accurate and efficient image search. In: CVPR. (2007)
16. Delvinioti, A., Jégou, H., Amsaleg, L., Houle, M.: Image retrieval with reciprocal and shared nearest neighbors. In: VISAPP. (2014)
17. Qin, D., Gammeter, S., Bossard, L., Quack, T., Van Gool, L.: Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In: CVPR. (2011)
18. Zhang, S., Yang, M., Cour, T., Yu, K., Metaxas, D.N.: Query specific fusion for image retrieval. In: ECCV. (2012)
19. Tolias, G., Sicre, R., Jégou, H.: Particular object retrieval with integral max-pooling of cnn activations. ICLR (2016)
20. Kalantidis, Y., Mellina, C., Osindero, S.: Cross-dimensional weighting for aggregated deep convolutional features. arXiv preprint arXiv:1512.04065 (2015)
21. Bai, S., Bai, X., Tian, Q., Latecki, L.J.: Regularized diffusion process on bidirectional context for object retrieval. IEEE Trans. PAMI (2018)
22. Bai, S., Zhou, Z., Wang, J., Bai, X., Jan Latecki, L., Tian, Q.: Ensemble diffusion for retrieval. In: ICCV. (2017)
23. Zhou, D., Weston, J., Gretton, A., Bousquet, O., Schölkopf, B.: Ranking on data manifolds. In: NIPS. (2003)

24. Tong, H., Faloutsos, C., Pan, J.Y.: Fast random walk with restart and its applications. In: Proceedings of the IEEE International Conference on Data Mining. (2006) 613–622
25. Chung, F.R.: Spectral graph theory. Volume 92. American Mathematical Soc. (1997)
26. Trefethen, L.N., Bau III, D.: Numerical linear algebra. SIAM (1997)
27. Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Revisiting oxford and paris: Large-scale image retrieval benchmarking. In: CVPR. (2018)
28. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR. (2008)