Fast Spectral Ranking for Similarity Search

Ahmet Iscen\textsuperscript{1} Yannis Avrithis\textsuperscript{2} Giorgos Tolias\textsuperscript{1} Teddy Furon\textsuperscript{2} Ondřej Chum\textsuperscript{1}
\textsuperscript{1}VRG, FEE, CTU in Prague \textsuperscript{2}Inria Rennes
\{ahmet.iscen,giorgos.tolias,chum\}@cmp.felk.cvut.cz
\{ioannis.avrithis,teddy.furon\}@inria.fr

Abstract

Despite the success of deep learning on representing images for particular object retrieval, recent studies show that the learned representations still lie on manifolds in a high dimensional space. This makes the Euclidean nearest neighbor search biased for this task. Exploring the manifolds online remains expensive even if a nearest neighbor graph has been computed offline.

This work introduces an explicit embedding reducing manifold search to Euclidean search followed by dot product similarity search. This is equivalent to linear graph filtering of a sparse signal in the frequency domain. To speed up online search, we compute an approximate Fourier basis of the graph offline. We improve the state of art on particular object retrieval datasets including the challenging Instre dataset containing small objects. At a scale of $10^5$ images, the offline cost is only a few hours, while query time is comparable to standard similarity search.

1. Introduction

Image retrieval based on deep learned features has recently achieved near perfect performance on all standard datasets [45, 14, 15]. It requires fine-tuning on a properly designed image matching task involving little or no human supervision. Yet, retrieving particular small objects is a common failure case. Representing an image with several regions rather than a global descriptor is indispensable in this respect [46, 60]. A recent study [24] uses a particularly challenging dataset [67] to investigate graph-based query expansion and re-ranking on regional search.

Query expansion [7] explores the image manifold by recursive Euclidean or similarity search on the nearest neighbors (NN) at increased online cost. Graph-based methods [44, 53] help reducing this cost by computing a $k$-NN graph offline. Given this graph, random walk\textsuperscript{1} processes [39, 70] provide a principled means of ranking. Iscen et al. [24] transform the problem into finding a solution $x$ of a linear system $Ax = y$ for a large sparse dataset-dependent matrix $A$ and a sparse query-dependent vector $y$. Such a solution can be found efficiently on-the-fly with conjugate gradients (CG). Even for an efficient solver, the query times are still in the order of one second at large scale.

In this work, we shift more computation offline: we exploit a low-rank spectral decomposition $A \approx U \Lambda U^\top$ and express the solution in closed form as $x = U \Lambda^{-1} U^\top y$. We thus treat the query as a signal $y$ to be smoothed over the graph, connecting query expansion to graph signal processing [50]. Figure 1 depicts 1d and graph miniatures of this interpretation. We then generalize, improve and interpret this spectral ranking idea on large-scale image retrieval. In particular, we make the following contributions:

1. We cast image retrieval as linear filtering over a graph, efficiently performed in the frequency domain.
2. We provide a truly scalable solution to computing an approximate Fourier basis of the graph offline, accompanied by performance bounds.

\textsuperscript{1}We avoid the term diffusion [11, 24] in this work.

Figure 1: The low-pass filtering of an impulse over the real line (top) and a graph (bottom). In a weighted undirected graph the information “flows” in all directions, controlled by edge weights. In retrieval, the impulse in red is the query, and the output $x$ is its similarity to all samples.
3. We reduce manifold search to a two-stage similarity search thanks to an explicit embedding.

4. A rich set of interpretations connects to different fields.

The text is structured as follows. Section 2 describes the addressed problem while Sections 3 and 4 present a description and an analysis of our method respectively. Section 5 gives a number of interpretations and connections to different fields. Section 6 discusses our contributions against related work. We report experimental findings in Section 8 and draw conclusions in Section 9.

2. Problem

In this section we state the problem addressed by this paper in detail. We closely follow the formulation of [24].

2.1. Representation

A set of $n$ descriptor vectors $\mathcal{V} = \{v_1, \ldots, v_n\}$, with each $v_i$ associated to vertex $u_i$ of a weighted undirected graph $G$ is given as an input. The graph $G$ with $n$ vertices $V = \{v_1, \ldots, v_n\}$ and $\ell$ edges is represented by its $n \times n$ symmetric nonnegative adjacency matrix $W$. Graph $G$ contains no self-loops, i.e. $W$ has zero diagonal. We assume $W$ is sparse with $2\ell \ll n(n-1)$ nonzero elements.

We define the $n \times n$ degree matrix $D := \text{diag}(W1)$ where $1$ is the all-ones vector, and the symmetrically normalized adjacency matrix $W' := D^{-1/2}WD^{-1/2}$ with the convention $0/0 = 0$. We also define the Laplacian and normalized Laplacian of $G$ as $L := D - W$ and $L' := D^{-1/2}LD^{-1/2} = I - W$, respectively. Both are singular and positive-semidefinite; the eigenvalues of $L$ are in the interval $[0, 2]$ [8]. Hence, if $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $W'$, its spectral radius $\rho(W') := \max_i |\lambda_i|$ is 1. Each eigenvector $u$ of $L$ associated to eigenvalue 0 is constant within connected components (e.g., $L1 = D1 - W1 = 0$), while the corresponding eigenvector of $L'$ is $D^{1/2}u$.

2.2. Transfer function

We define the $n \times n$ matrices $L_\alpha := \beta^{-1}(D - \alpha W)$ and $L^\alpha := D^{-1/2}L_\alpha D^{-1/2} = \beta^{-1}(I - \alpha W)$, where $\alpha \in [0, 1)$ and $\beta := 1 - \alpha$. Both are positive-definite. Given the $n \times 1$ sparse observation vector $y$ online, [24] computes the $n \times 1$ ranking vector $x$ as the solution of the linear system

$$L_\alpha x = y. \quad (1)$$

We can write the solution as $h_\alpha(W)y$, where

$$h_\alpha(W) := (1 - \alpha)(I - \alpha W)^{-1} \quad (2)$$

for a matrix $W$ such that $I - \alpha W$ is nonsingular; indeed, $L^{-1}_\alpha = h_\alpha(W)$. Here we generalize this problem by considering any given transfer function $h : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S}$ is the set of real symmetric matrices including scalars, $\mathbb{R}$. The general problem is then to compute

$$x^* := h(W)y \quad (3)$$

efficiently, in the sense that $h(W)$ is never explicitly computed or stored: $W$ is given in advance and we are allowed to pre-process it offline, while both $y$ and $h$ are given online. For $h_\alpha$ in particular, we look for a more efficient solution than solving linear system (1).

2.3. Retrieval application

The descriptors $\mathcal{V}$ are generated by extracting image descriptors from either whole images, or from multiple sampled rectangular image regions, which can be optionally reduced by a Gaussian mixture model as in [24]. Note that the global descriptor is a special case of the regional one, using a single region per image. In the paper, we use CNN-based descriptors [45].

The undirected graph $G$ is a $k$-NN similarity graph constructed as follows. Given two descriptors $v, z$ in $\mathbb{R}^d$, their similarity is measured as $s(v, z) = [v^T z]^\gamma$, where exponent $\gamma > 0$ is a parameter. We denote by $s(v_i|z)$ the similarity of $v_i$ if $v_i$ is a $k$-NN of $z$ in $\mathcal{V}$ and zero otherwise. The symmetric adjacency matrix $W$ is defined as $w_{ij} := \min(s(v_i|v_j), s(v_j|v_i))$, representing mutual neighborhoods. Online, given a query image represented by descriptors $\{q_1, \ldots, q_m\} \subset \mathbb{R}^d$, the observation vector $y \in \mathbb{R}^n$ is formed with elements $y_i := \sum_{j=1}^m s(v_i|q_j)$ by pooling over query regions. We make $y$ sparse by keeping the $k$ largest entries and dropping the rest.

3. Method

This section presents our fast spectral ranking (FSR) algorithm in abstract form first, then with concrete choices.

3.1. Algorithm

We describe our algorithm given an arbitrary $n \times n$ matrix $A \in \mathcal{S}$ instead of $\mathcal{W}$. Our solution is based on a sparse low-rank approximation of $A$ computed offline such that online, $x \approx h(A)y$ is reduced to a sequence of sparse matrix-vector multiplications. The approximation is based on a randomized algorithm [47] that is similar to Nyström sampling [12] but comes with performance guarantees [18, 68]. In the following, $r \ll n$, $p < r$, $q$ and $\tau$ are given parameters, and $\tilde{r} = r + p$.

1. (Offline) Using simultaneous iteration [62, §28], compute an $n \times \tilde{r}$ matrix $Q$ with orthonormal columns that represents an approximate basis for the range of $A$, i.e., $QQ^T A \approx A$. In particular, this is done as follows [18, §4.5]: randomly draw an $n \times \tilde{r}$ standard Gaussian matrix $B^{(0)}$ and repeat for $t = 0, \ldots, q - 1$:
(a) Compute QR factorization \( Q^{(t)} R^{(t)} = B^{(t)} \).
(b) Define the \( n \times \hat{r} \) matrix \( B^{(t+1)} := AQ^{(t)} \).

Finally, set \( Q := Q^{(q-1)} \), \( B := B^{(q)} = AQ \).

2. (Offline–Fourier basis) Compute a rank-\( r \) eigenvalue decomposition \( U A U^T \approx A \), where \( n \times r \) matrix \( U \) has orthonormal columns and \( r \times r \) matrix \( A \) is diagonal.

In particular, roughly following [18, \$5.3]:

(a) Form the \( \hat{r} \times \hat{r} \) matrix \( C := Q^T B = Q^T AQ \).
(b) Compute its eigendecomposition \( \hat{V} \Lambda \hat{V}^T = C \).
(c) Form \((V, \Lambda)\) by keeping from \((\hat{V}, \hat{\Lambda})\) the slices (rows/columns) corresponding to the \( r \) largest eigenvalues.

(d) Define the matrix \( U := QV \).

3. (Offline) Make \( U \) sparse by keeping its \( \tau \) largest entries and dropping the rest.

4. (Online) Given \( y \) and \( h \), compute
\[
\mathbf{x} := U h(\lambda) U^T y. \tag{4}
\]

Observe that \( U^T \) projects \( y \) onto \( \mathbb{R}^r \). With \( \Lambda \) being diagonal, \( h(\lambda) \) is computed element-wise. Finally, multiplying by \( U \) and ranking \( x \) amounts to dot product similarity search in \( \mathbb{R}^r \). The online stage is very fast, provided \( U \) only contains few leading eigenvectors and \( y \) is sparse. We consider the following variants:

- **FSR.sparse**: This is the complete algorithm.
- **FSR.approx**: Drop sparsification stage 3.
- **FSR.rank-\( r \)**: Drop approximation stage 1 and sparsification stage 3. Set \( r = n, Q = I, B = A \) in stage 2.
- **FSR.exact**: same as FSR.rank-\( r \) for \( r = n \).

To see why FSR.exact works, consider the case of \( h_\alpha(W) \). Let \( W \approx U A U^T \). It follows that \( h_\alpha(W)/\beta = (I - \alpha W)^{-1} \approx U (I - \alpha \Lambda)^{-1} U^T \), where \( (I - \alpha \Lambda)^{-1} \) is computed element-wise. Then, \( x^* \approx \beta U (I - \alpha \Lambda)^{-1} U^T y \).

The general case is discussed in section 4.

### 3.2. Retrieval application

Returning to the retrieval problem, we compute the ranking vector \( x \in \mathbb{R}^n \) by (4), containing the ranking score \( x_i \) of each dataset region \( v_i \). To obtain a score per image, we perform a linear pooling operation [24] represented as \( \mathbf{X} := \Sigma \mathbf{x} \) where \( \Sigma \) is a sparse \( N \times n \) pooling matrix. The \( N \times r \) matrix \( \mathbf{U} := \Sigma \mathbf{U} \) is indeed computed offline so that we directly compute \( \mathbf{X} = \mathbf{U} h(\lambda) \mathbf{U}^T \mathbf{y} \) online.

Computing \( y \) involves Euclidean search in \( \mathbb{R}^d \), which happens to be dot product because vectors are \( \ell^2 \)-normalized. Applying \( \mathbf{U} \) and ranking \( x \) amounts to a dot product similarity search in \( \mathbb{R}^r \). We thus reduce manifold search to Euclidean followed by dot product search. The number of nonzero elements of \( y \) and rows of \( \mathbf{U} \), whence the cost, are the same for global or regional search.

### 4. Analysis

We derive the asymptotic space and time complexity of different algorithm variants and derive necessary condition for correctness and error bounds of approximate variants.

#### 4.1. Complexity

The offline complexity is mainly determined by the number of columns \( \hat{r} \) of matrix \( Q \): Stage 1 reduces the size of the problem from \( n^2 \) down to \( \hat{r} \). The online complexity is determined by the number of nonzero entries in matrix \( \mathbf{U} \). A straightforward analysis leads to the following:

- **FSR.approx**: The offline complexity is \( O(qn(k + \hat{r})\hat{r}) \) time and \( O(n\hat{r}) \) space; its online (time and space) complexity is \( O(n) \).
- **FSR.sparse**: The offline complexity is \( O(qn(k + \hat{r})\hat{r} + n \log r) \) time and \( O(n\hat{r}) \) space; its online complexity is \( O(n) \).

Stage 1 is “embarrassingly parallelizable” meaning that it is dramatically accelerated on parallel and distributed platforms. Since the online stage 4 amounts to NN search, any approximate method applies, making it sublinear in \( n \).

#### 4.2. Correctness

We derive here the conditions on \( h \) and \( A \) under which our algorithm is correct under no truncation, i.e.,
\( \text{FSR.exact}(y\mid A, h) = h(A)y \).
We also show, that \( h_\alpha \) and \( \mathcal{W} \) satisfy these conditions, which is an alternative proof of correctness to the one in Section 3.1.

Starting from the fact a real symmetric matrix \( A \) is diagonalizable, there exists an exact eigenvalue decomposition \( U A U^T = A \), where \( U \) is orthogonal. According to [1, \$9.14.9.2], we have \( h(A) = U h(\Lambda) U^T = U \text{diag}(h(\lambda_1), \ldots, h(\lambda_n)) U^T \) if and only if there exists a series expansion of \( h \) converging for this specific \( A \):
\[
h(A) = \sum_{i=0}^{\infty} c_i A^i. \tag{5}
\]

This holds in particular for \( h_\alpha \) admitting the geometric progression expansion
\[
h_\alpha(A) := \beta (I - \alpha A)^{-1} = \beta \sum_{i=0}^{\infty} (\alpha A)^i, \tag{6}
\]
which converges absolutely if \( \|\alpha A\| < 1 \) [1, \$9.6.9.19]. This holds for \( \mathcal{W} = \mathcal{W} \) because \( \alpha < 1 \) and \( \mathcal{W} = 1 \).

#### 4.3. Error bound

We present main ideas for bounding the approximation error of FSR.rank-\( r \) and FSR.approx coming from literature, and we derive another condition on \( h \) under which
our algorithm is valid under truncation. The approximation $Q^T A A Q = A$ of stage 1 is studied in [18, $\S$9.3,10.4]: an average-case bound on $\|A - Q^T A\|$ decays exponentially fast in the number of iterations $q$ to $\|\lambda_{q+1}\|$. Stage 2 yields an approximate eigenvalue decomposition of $A$: Since $A$ is symmetric, $A \approx Q^T A Q Q^T = Q C Q^T \approx Q V^T A^T Q^T = U A U^T$. The latter approximation $C \approx V^T A^T$ is essentially a best rank-$r$ approximation of $C = Q^T A U$. This is also studied in [18, $\S$9.4] for the truncated SVD case of a non-symmetric matrix. It involves an additional term of $\|\lambda_{q+1}\|$ in the error.

We are actually approximating $h(A)$ by $U h(A) U^T$, so that $|h(\lambda_{q+1})|$ governs the error instead of $\|\lambda_{q+1}\|$. A similar situation appears in [61, $\S$3.3]. Therefore, our method makes sense only when the restriction of $h$ to scalars is non-decreasing. This is the case for $h_\alpha$.

5. Interpretation

Our work is connected to studies in different fields with a long history. Here we give a number of interpretations both in general and in the particular case $h = h_\alpha$.

5.1. Graph signal processing

In signal processing [38], a discrete-time signal of period $n$ is a vector $s \in \mathbb{R}^n$ where indices are represented by integers modulo $n$, that is, $s_i := s_{i \mod n} + 1$ for $i \in \mathbb{Z}$. A shift (or translation, or delay) of $s$ by one sample is the mapping $s \mapsto s \ast \tau$. If we define the $n \times n$ circulant matrix $C_n := (e_2 e_3 \ldots e_n e_1)^T$, a shift can be represented by $s \mapsto C_n s$ [50]. A linear, time (or shift) invariant filter is the mapping $s \mapsto H s$ where $H$ is an $n \times n$ matrix with a series representation $H := h(C_n) := \sum_{r=0}^\infty h(r) C_n^r$. Matrix $C_n$ has the eigenvalue decomposition $U A U^T$ where $U^T$ is the $n \times n$ discrete Fourier transform matrix $F$. If the series $h(C_n)$ converges, filtering $s \mapsto H s$ is written as

$$s \mapsto F^{-1} h(\Lambda) F s.$$  
(7)

That is, $s$ is mapped to the frequency domain, scaled element-wise, and mapped back to the time domain.

Graph signal processing [50, 54] generalizes the above concepts to graphs by replacing $C_n$ by $W$, an appropriately normalized adjacency matrix of an arbitrary graph. If $U A U^T$ is the eigenvalue decomposition of $W$, we realize that (4) treats $y$ as a (sparse) signal and filters it in the frequency domain via transfer function $h$ to obtain $x$. Function $h_\alpha$ in particular is a low-pass filter, as illustrated in Figure 2. By varying $\alpha$ from 0 to 1, the frequency response varies from all-pass to sharp low-pass.

5.2. Random walks

Consider the iterating process: for $t = 1, 2, \ldots$

$$x(t) := \alpha A x(t-1) + (1 - \alpha) y.$$  
(8)

If $A$ is a stochastic transition matrix and $x(0)$, $y$ are distributions over vertices, this specifies a random walk on a (directed) graph: at each iteration a particle moves to a neighboring vertex with probability $\alpha$ or jumps to a vertex according to distribution $y$ with probability $1 - \alpha$. This is called a Markov chain with restart [2] or random walk with restart [40]. State $x(t)$ converges to $x^* = h_\alpha(A)y$ as $t \to \infty$ provided $\theta(A) < 1$ [69]. In fact, (8) is equivalent to Jacobi solver [17] on linear system (1) [24].

If $y = e_i$, the $i$-th canonical vector, then $x^*$ is used to rank the vertices of $G$, expressing a measure of “similarity” to $v_i$ [70]. Parameter $\alpha$ controls how much $x^*$ is affected by boundary condition $y$ [64]: $x^*$ equals $y$ for $\alpha = 0$, while in the limit $\alpha \to 1$, $x^*$ tends to a dominant eigenvector of $A$. Indeed, for $\alpha = 1$, (8) becomes a power iteration.

5.3. Random fields

Given a positive-definite $n \times n$ precision matrix $A \in \mathcal{S}$ and a mean vector $\mu \in \mathbb{R}^n$, a Gaussian Markov random field (GMRF) [49] with respect to an undirected graph $G$ is a random vector $x \in \mathbb{R}^n$ with normal density $p(x) := \mathcal{N}(x|\mu, A^{-1})$ iff $A$ has the same nonzero off-diagonal entries as the adjacency matrix of $G$. Its canonical parametrization $p(x) \propto e^{-E(x|\mu, A)}$ where $E(x|b, A) := \frac{1}{2} x^T A x - b^T x$ is a quadratic energy. Its expectation $\mu = A^{-1} b$ is the minimizer of this energy. Now, $x^* = \mathcal{L}_\alpha^{-1} y$ (1) is the expectation of a GMRF with energy

$$f_\alpha(x) := E(x|y, \mathcal{L}_\alpha) = \frac{1}{2} x^T \mathcal{L}_\alpha x - y^T x.$$  
(9)

A mean field method on this GMRF is equivalent to Jacobi or Gauss-Seidel solvers on (1) [66]. Yet, conjugate gradients (CG) [37] is minimizing $f_\alpha(x)$ more efficiently [24, 5].

If we expand $f_\alpha(x)$ using $\beta \mathcal{L}_\alpha = \alpha \mathcal{L} + (1 - \alpha) I$, we find that it has the same minimizer as

$$\alpha \sum_{i,j} w_{ij} \| \hat{x}_i - \hat{x}_j \|^2 + (1 - \alpha) \| x - y \|^2,$$  
(10)
where $\hat{x} := D^{-1/2}x$. The pairwise smoothness term encourages $x$ to vary little across edges with large weight whereas the unary fitness term to stay close to observation $y$ [69]. Again, $\alpha$ controls the trade-off: $x^*$ equals $y$ for $\alpha = 0$, while for $\alpha \rightarrow 1$, $x^*$ tends to be constant over connected components like dominant eigenvectors of $W$.

### 5.4. Regularization and kernels

The first term of (9) is interpreted as a regularization operator related to a kernel $K = L^{-1}$ [58, 57, 31]. In a finite graph, a kernel can be seen either as an $n \times n$ matrix $K$ or a function $\kappa : V^2 \rightarrow \mathbb{R}$ operating on pairs of vertices. More generally, if $h(x) > 0$ for $x \in \mathbb{R}$, which holds for $h_\alpha$, then $\hat{K} := h(W)$ is positive-definite and there is an $n \times n$ matrix $\Phi$ such that $K = \hat{\Phi} \hat{\Phi}^\top$, or $\kappa(v_i, v_j) = \phi(v_i)^\top \phi(v_j)$ where feature map $\phi : V \rightarrow \mathbb{R}^n$ is given by $\phi(v_i) := \Phi e_i$. A particular choice for $\Phi$ is

$$\Phi := h(\Lambda)^{1/2} U^\top$$

(11)

where $U \Lambda U^\top$ is the eigenvalue decomposition of $W$. If we choose a rank-$r$ approximation instead, then $\Phi$ is an $r \times n$ matrix and $\phi$ is a low-dimensional embedding onto $\mathbb{R}^r$.

The goal of out-of-sample extension is to compute a “similarity” $\hat{\kappa}(z_1, z_2)$ between two unseen vectors $z_1, z_2 \in \mathbb{R}^d$ not pertaining to the graph. Here we define

$$\hat{\kappa}(z_1, z_2) := \psi(z_1)^\top \hat{\Phi}^\top \hat{\Phi} \psi(z_2)$$

(12)

given any mapping $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^n$, e.g., $\psi(z)_i := s(v_i|z)$ discussed in section 2. This extended kernel is also positive-definite and its embedding $\hat{\phi}(z) = \hat{\phi} \psi(z)$ is a linear combination of the dataset embeddings. For $r \ll n$, our method allows rapid computation of $\kappa$ or $\hat{\kappa}$ for any given function $h$, without any dense $n \times n$ matrix involved.

### 5.5. Paths on graphs

Many nonlinear dimension reduction methods replace Euclidean distance with an approximate geodesic distance, assuming the data lie on a manifold [33]. This involves the all-pairs shortest path (APSP) problem and Dijkstra’s algorithm is a common choice. Yet, it is interesting to consider a naive algorithm [9, §25.1]. We are given a distance matrix where missing edges are represented by $\infty$ and define similarity weight $w_{ij} = e^{-d_{ij}}$. A path weight is a now a product of similarities and “shortest” means “of maximum weight”. Defining matrix power $A^\otimes t$ as $A^t$ with $+$ replaced by max, the algorithm is reduced to computing $\max_{t} W^\otimes t$ (element-wise). Element $i, j$ of $W^\otimes t$ is the weight of the shortest path of length $t$ between $v_i, v_j$.

Besides their complexity, shortest paths are sensitive to changes in the graph. An alternative is the sum\(^3\) of weights over paths of length $t$, recovering the ordinary matrix power $W^t$, and the weighted sum over all lengths $\sum_{t=0}^\infty c_t W^t$, where coefficients $(c_t)_{t \in \mathbb{N}}$ allow for convergence [64], [52, §9.4]. This justifies (5) and reveals that coefficients control the contribution of paths depending on length. A common choice is $c_t = \beta \alpha^t$ with $\beta = 1 - \alpha$ and $\alpha \in [0, 1)$ being a damping factor [64], which justifies function $h_\alpha$ (6).

### 6. Related work

The history of the particular case $h = h_\alpha$ is the subject of the excellent study of spectral ranking [64]. The fundamental contributions originate in the social sciences and include the eigenvector formulation by Seeley [51], damping by $\alpha$ (6) by Katz [29] and the boundary condition $y (1)$ by Hubbell [22]. The most well-known follower is PageRank [39]. In machine learning, $h_\alpha$ has been referred to as the von Neumann [27, 52] or regularized Laplacian kernel [57]. Along with the diffusion kernel [32, 31], it has been studied in connection to regularization [58, 57].

Random fields are routinely used for low-level vision tasks where one is promoting smoothness while respecting a noisy observation, like in denoising or segmentation, where both the graph and the observation originate from a single image [59, 5]. A similar mechanism appears in semi-supervised learning [69, 73, 71, 6] or interactive segmentation [16, 30] where the observation is composed of labels over a number of samples or pixels. In our retrieval scenario, the observation is formed by the neighbors in the graph of an external query image (or its regions).

The random walk or random walk with restart (RWR) formulation [70, 69, 40] is an alternative interpretation to retrieval [11]. Yet, directly solving a linear system is superior [24]. Offline matrix decomposition has been studied for RWR [61, 13, 26]. All three methods are limited to $h_\alpha$ while sparse LU decomposition [13, 26] assumes an uneven distribution of vertex degrees [28], which is not the case for $k$-NN graphs. In addition, we reduce manifold search to two-stage Euclidean search via an explicit embedding, which is data dependent through the kernel $K = L^{-1}$.

In the general case, the spectral formulation (4) has been known in machine learning [6, 52, 36, 72, 65] and in graph signal processing [50, 54, 19]. The latter is becoming popular in the form of graph-based convolution in deep learning [4, 21, 10, 3, 34, 43]. However, with few exceptions [4, 21], which rely on an expensive decomposition, there is nothing spectral when it comes to actual computation. It is rather preferred to work with finite polynomial approximations of the graph filter [10, 3] using Chebyshev polynomials [19, 55] or translation-invariant neighborhood templates in the spatial domain [34, 43].

We cast retrieval as graph filtering by constructing an appropriate observation vector. We actually perform the computation in the frequency domain via a scalable so-

---

\(^3\)In fact, similar to softmax due to the exponential and normalization.
lution. Comparing to other applications, retrieval conveniently allows offline computation of the graph Fourier basis and online reuse to embed query vectors. An alternative is to use random projections [63, 48]. This roughly corresponds to a single iteration of our step 1. Our solution is thus more accurate, while \( h \) is specified online.

7. Practical considerations

**Block diagonal case.** Each connected component of \( G \) has a maximal eigenvalue \( 1 \). These maxima of small components dominate the eigenvalues of the few (or one) “giant” component that contain the vast majority of data [28]. For this reason we find the connected components with the “ant” component that contain the vast majority of data [28]. Each connected component of \( G \) has a maximal eigenvalue \( 1 \).

As shown in Figure 4, the effect of \( r \) depends on the dataset. In all cases the optimal performance is already reached at \( r = 1 \). On Paris6k in particular, this happens as soon as \( r = 100 \). Compared to FSR.EXACT as implemented in [24], it achieves the same mAP but 150 times faster on Oxford5k and Paris6k and 300 times faster on Instre. Global search demonstrates a similar behavior.

We achieve 97.0 mAP on Paris6k, which is near-perfect. Figure 3 shows the two queries with the lowest AP and their top-ranked negative images. In most cases the ground-truth is incorrect, as these images have visual overlap with the query bounding box. The first correct negative image for “La Défense” appears at rank 126, where buildings from the surroundings are retrieved due to “topic drift”. The same happens with “Pyramide du Louvre”, where the first correct negative image is at rank 108.

8. Experimental Setup

Datasets. We use three image retrieval benchmarks: Oxford Buildings (Oxford5k) [41], Paris (Paris6k) [42] and Instre [67], with the evaluation protocol introduced in [24] for the latter. We conduct large-scale experiments by following a standard protocol of adding 100k distractor images from Flickr [41] to Oxford5k and Paris6k, forming the so called Oxford105k and Paris106k. Mean average precision (mAP) evaluates the retrieval performance in all datasets.

Image Descriptors. We apply our method on the same global and regional image descriptors as in [24]. In particular, we work with \( d \)-dimensional vectors extracted from VGG [56] \((d = 512)\) and ResNet101 [20] \((d = 2,048)\) networks fine-tuned specifically for image retrieval [45, 15]. Global description is R-MAC with 3 different scales [60], including the full image as a separate region. Regional descriptors consist of the same regions as those involved in R-MAC but without sum pooling, resulting in 21 vectors per image on average. Global and regional descriptors are processed by supervised whitening [45].

Implementation. We adopt the same parameters for graph construction and search as in [24]. The pairwise descriptor similarity is defined as \( s(v, z) = |v^\top z|^3 \). We use \( \alpha = 0.99 \), and keep the top \( k = 50 \) and \( k = 200 \) mutual neighbors in the graph for global and regional vectors, respectively. These choices make our experiments directly comparable to prior results on manifold search for image retrieval with CNN-based descriptors [24]. In all our FSR.APPROX experiments, we limit the algorithm within the largest connected component only, while each element \( x_i \) for vertex \( v_i \) in any other component is just copied from \( y_i \). This choice works well because the largest component holds nearly all data in practice. Following [24], generalized max-pooling [35, 23] is used to pool regional diffusion scores per image. Reported search times exclude the construction of the observation vector \( y \), since this task is common to all baseline and our methods. Time measurements are reported with a 4-core Intel Xeon 2.00GHz CPU.

8.1. Experimental Setup

Datasets. We use three image retrieval benchmarks: Oxford Buildings (Oxford5k) [41], Paris (Paris6k) [42] and Instre [67], with the evaluation protocol introduced in [24] for the latter. We conduct large-scale experiments by following a standard protocol of adding 100k distractor images from Flickr [41] to Oxford5k and Paris6k, forming the so called Oxford105k and Paris106k. Mean average precision (mAP) evaluates the retrieval performance in all datasets.

Image Descriptors. We apply our method on the same global and regional image descriptors as in [24]. In particular, we work with \( d \)-dimensional vectors extracted from VGG [56] \((d = 512)\) and ResNet101 [20] \((d = 2,048)\) networks fine-tuned specifically for image retrieval [45, 15]. Global description is R-MAC with 3 different scales [60], including the full image as a separate region. Regional descriptors consist of the same regions as those involved in R-MAC but without sum pooling, resulting in 21 vectors per image on average. Global and regional descriptors are processed by supervised whitening [45].

Implementation. We adopt the same parameters for graph construction and search as in [24]. The pairwise descriptor similarity is defined as \( s(v, z) = |v^\top z|^3 \). We use \( \alpha = 0.99 \), and keep the top \( k = 50 \) and \( k = 200 \) mutual neighbors in the graph for global and regional vectors, respectively. These choices make our experiments directly comparable to prior results on manifold search for image retrieval with CNN-based descriptors [24]. In all our FSR.APPROX experiments, we limit the algorithm within the largest connected component only, while each element \( x_i \) for vertex \( v_i \) in any other component is just copied from \( y_i \). This choice works well because the largest component holds nearly all data in practice. Following [24], generalized max-pooling [35, 23] is used to pool regional diffusion scores per image. Reported search times exclude the construction of the observation vector \( y \), since this task is common to all baseline and our methods. Time measurements are reported with a 4-core Intel Xeon 2.00GHz CPU.

8.2. Retrieval Performance

Rank-\( r \). We evaluate the performance of FSR.RANK-\( r \) for varying rank \( r \), which affects the quality of the approximation and defines the dimensionality of the embedding space. As shown in Figure 4, the effect of \( r \) depends on the dataset. In all cases the optimal performance is already reached at \( r = 1 \). On Paris6k in particular, this happens as soon as \( r = 100 \). Compared to FSR.EXACT as implemented in [24], it achieves the same mAP but 150 times faster on Oxford5k and Paris6k and 300 times faster on Instre. Global search demonstrates a similar behavior.

We achieve 97.0 mAP on Paris6k, which is near-perfect. Figure 3 shows the two queries with the lowest AP and their top-ranked negative images. In most cases the ground-truth is incorrect, as these images have visual overlap with the query bounding box. The first correct negative image for “La Défense” appears at rank 126, where buildings from the surroundings are retrieved due to “topic drift”. The same happens with “Pyramide du Louvre”, where the first correct negative image is at rank 108.
Regional search performs better than global [24] at the cost of more memory and slower query. We unlock this bottleneck thanks to the offline pooling $\bar{U} = \Sigma U$. Indeed, global and regional search on Instre take 0.040s and 0.042s respectively with our method, while the corresponding times for FSR.exact are 0.055s and 3s.

**Approximate eigendecomposition** keeps the off-line stage tractable at large scale. With 570k regional descriptors on Instre, FSR.rand-5000 and FSR.approx yield a mAP of 89.5 and 89.2 respectively, with offline cost 60 and 3 hours respectively, using 16-core Intel Xeon 2.00GHz CPU. This is important at large scale because the off-line complexity of FSR.rand-r is polynomial.

When new images are added, one can express them according to existing ones, as in (12). We evaluate such extension by constructing the graph on a random subset of 100%, 90%, 70%, 50%, 30% and 10% of Instre, yielding 80.5, 80.1, 78.3, 75.8, 70.2 and 40.6 mAP respectively on the entire dataset, with global search. The drop is graceful until 30%; beyond that, the graph needs to be updated.

### 8.3. Large-scale experiments

We now apply our approach to a larger scale by using only 5 descriptors per image using GMM reduction [24]. This choice improves scalability while minimizing the accuracy loss.

FSRw.approx becomes crucial, especially at large scale, because vectors of sparsely populated parts of the graph are not well represented. Figure 5 shows the comparison between FSRw.approx and FSR.exact. We achieve 90.2 and 94.2 with FSR.approx and FSRw.approx respectively, with $r = 10k$ and ResNet101 descriptors.

We further report the performance separately for each of the 11 queries of Oxford105k dataset. Results are shown in Figure 6. Low values of $r$ penalize sparsely populated parts of the graph, i.e., landmarks with less similar instances in the dataset. FSRw.approx partially solves this issue.

**The search time** is 0.14s and 0.3s per query for $r = 5k$ and $r = 10k$ respectively on Oxford105k. It is two orders of magnitude faster than FSR.exact: The implementation of [24] requires about 14s per query, which is reduced to 1s with dataset truncation: manifold search is a re-ranking only applied to top-ranked images. We do not use any truncation. This improves the mAP by 3% and our method is still one order of magnitude faster.

**Sparse embeddings.** Most descriptors belong only to few manifolds and each embedding vector has high energy in the corresponding components. Setting $r = 10k$, large enough to avoid compromising accuracy, Figure 7 shows the effect of sparsifying the embeddings with FSRw.sparse on Oxford105k. Remarkably, we can make up to 90% memory savings with only 2% drop of mAP.

**Quantized descriptors.** Construction of the observation vector requires storing the initial descriptors. We further...
use product quantization (PQ) [25] to compress them. Using FSRw.APPROX on Oxford105k, mAP drops from 94.4 with uncompressed descriptors to 94.2 and 91.1 with 256- and 64-byte PQ codes, respectively.

### 8.4. Comparison to other methods

Table 1 compares our method with the state-of-the-art. We report results for \( r = 5k, \) FSR.RANK-\( r \) for global description, FSR.APPROX for regional description, and FSRw.APPROX in large-scale (with 100k distractors) and regional experiments. GMM reduces the number of regions per image from 21 to 5 [24]. We do not experiment at large-scale without GMM since there is not much improvement and it is less scalable. Our method reaches performance similar to that of FSR.EXACT as evaluated with CG [24]. Our benefit comes from the dramatic speed-up. For the first time, manifold search runs almost as fast as Euclidean search. Consequently, dataset truncation is no longer needed and this improves the mAP.

### 9. Discussion

This work reproduces the excellent results of online linear system solution [24] at fraction of query time. We even improve performance by avoiding to truncate the graph online. The offline stage is linear in the dataset size, embarrassingly parallelizable and takes a few hours in practice for the large scale datasets of our experiments. The approximation quality is arbitrarily close to the optimal one at a given embedding dimensionality. The required dimensionality for good performance is large but in practice the embedded vectors are very sparse. This resembles an encoding based on a large vocabulary, searched via an inverted index. Our method is generic and may be used for problems other than search, including clustering and unsupervised or semi-supervised learning.

### Acknowledgments

The authors were supported by the MSMT LL1303 ERC-CZ grant. The Tesla K40 used for this research was donated by the NVIDIA Corporation. The authors would like to thank James Pritts for fruitful discussions during this work.
References


[45] F. Radenović, G. Tolias, and O. Chum. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *ECCV*, 2016. 1, 2, 6, 8


