Fast Spectral Ranking for Similarity Search

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

[1]Inria Rennes   [2]Center for Machine Perception, CTU in Prague

{ahmet.iscen,ioannis.avrithis,teddy.furon}@inria.fr

{giorgos.tolias,chum}@cmp.felk.cvut.cz

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. Therefore, nearest neighbor search cannot be expected to be optimal for this task. Even if a nearest neighbor graph is computed offline, exploring the manifolds online remains expensive. This work introduces an explicit embedding reducing manifold search to Euclidean search followed by dot product similarity search. We show this is equivalent to linear graph filtering of a sparse signal in the frequency domain, and we introduce a scalable offline computation of an approximate Fourier basis of the graph. We improve the state of art on standard particular object retrieval datasets including a challenging one 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 [43, 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 [44, 58]. A recent study [23] uses a particularly challenging dataset [65] 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 [42, 51] help reducing this cost by computing a k-NN graph offline. Given this graph, *random walk*[1] processes [37, 68] provide a principled means of ranking. Iscen

---

[1]We avoid the term *diffusion* [11, 23] in this work.

(a) Input signal **y**

(b) Output signal **x**
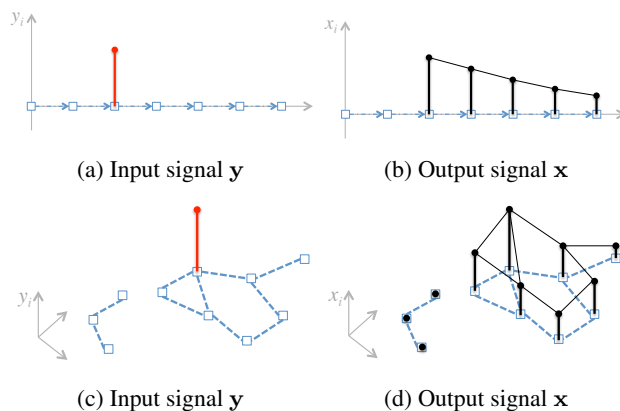
(c) Input signal **y**

(d) Output signal **x**

Figure 1: (a), (b) Output signal is $x_i := \beta \sum_{t=0}^{\infty} \alpha^t y_{i-t}$ with $\alpha \in [0,1)$ and $\beta := 1 - \alpha$. This is a low-pass filter determined by $x_i = \alpha x_{i-1} + (1 - \alpha)y_i$, with impulse response $h_t = \beta \alpha^t u_t$ and transfer function $H(z) := \beta \sum_{t=0}^{\infty} (az^{-1})^t = \beta/(1 - \alpha z^{-1})$. This assumes a directed graph G with vertices $V = \mathbb{Z}$ and edges $E = \{(i, i+1) : i \in \mathbb{Z}\}$, shown in blue. (c), (d) Using a weighted undirected graph G instead. Information "flows" in all directions, controlled by edge weights. In retrieval, the sample in red is the query, and the output x is its similarity to all samples.

et al. [23] make such a process efficient by solving a linear system online with *conjugate gradients* (CG), also handling multiple regions at the same cost. However, query times are still in the order of one second at large scale. Our work does more work offline than graph construction and shows that query expansion is more than just "post-processing".

Given a dataset of size n and a query vector, a sparse *observation vector* $\mathbf{y} \in \mathbb{R}^n$ is first constructed based on a similarity search. The final *ranking vector* $\mathbf{x}^* \in \mathbb{R}^n$ results from a linear operator $T : \mathbb{R}^n \to \mathbb{R}^n$ applied on $\mathbf{y}$ [23]. Our work computes and stores a sparse rank-r approximation of T represented by $n \times r$ matrix Φ offline, such that $\mathbf{x} = \Phi\Phi^\top \mathbf{y} \approx \mathbf{x}^*$. This is obtained online by projecting $\mathbf{y}$ onto a r-dimensional *embedding space* followed by a search of the NN of $\Phi^\top \mathbf{y}$ among the n rows of Φ in $\mathbb{R}^r$. This is achieved

without ever forming a dense representation for T. Figure 1 illustrates a 1d signal processing miniature of this idea. We make the following contributions:

1. Image retrieval is cast as *linear filtering* over a graph, efficiently performed in the *frequency domain*.

2. A truly scalable solution computes an approximate Fourier basis of the graph offline, accompanied by performance bounds.

3. *Manifold search is reduced to a two-stage similarity search* thanks to an explicit embedding. This is useful in many other applications than retrieval.

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

The text is structured as follows. Sections 2 and 3 present a description and an analysis of our method respectively. Section 4 gives a number of interpretations and connections to different fields. Section 5 discusses our contributions against related work. We report experimental findings in Section 7 and draw conclusions in Section 8.

2. Method

This section presents the problem and the proposed algorithm in abstract form, requiring as input a graph and a matrix function. Concrete choices relevant to image retrieval are discussed in the end.

2.1. Problem

We are given a weighted undirected graph $G = (V, E, w)$ with n vertices $V = \{v_1, \ldots, v_n\}$, ℓ edges $E \subset V^2$, and a positive weight (similarity) function w. The graph is determined by its $n \times n$ symmetric nonnegative *adjacency matrix* W with elements $w_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in E$ and zero otherwise. Graph G contains no self-loops, *i.e.* W has zero diagonal. We assume W is sparse with $2\ell \leq kn$ nonzero elements for given $k \ll n$.

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

We are also given a *transfer function* $h : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S}$ is the set of real symmetric square matrices including scalars, $\mathbb{R}$. Our results hold for any function h under conditions given in section 3, but our standard choice is

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

parametrized by $\alpha \in [0, 1)$ and provided that $I - \alpha A$ is nonsingular. Accordingly, we define the $n \times n$ matrices $L_\alpha := \beta^{-1}(D - \alpha W)$ and $\mathcal{L}_\alpha := D^{-1/2}L_\alpha D^{-1/2} = \beta^{-1}(I - \alpha \mathcal{W})$, where $\beta := 1 - \alpha$. Both are positive-definite [23] and indeed $\mathcal{L}_\alpha^{-1} = h_\alpha(\mathcal{W})$.

Now, given a $n \times 1$ *observation vector* $\mathbf{y}$ on the graph vertices, the problem is to compute $n \times 1$ *ranking vector*

$$\mathbf{x}^* := h(\mathcal{W})\mathbf{y} \quad (2)$$

efficiently, in the sense that $h(\mathcal{W})$ is not explicitly computed or stored. For h_α in particular, we look for a more efficient solution than solving linear system

$$\mathcal{L}_\alpha \mathbf{x} = \mathbf{y} \quad (3)$$

as in [23]. The idea is that $\mathcal{W}$ and h are given in advance and we are allowed to pre-process them *offline*, while $\mathbf{y}$ ($\mathbf{x}$) is given (resp. computed) *online*. Moreover, $\mathbf{y}$ is sparse in practice as discussed in section 2.3.

2.2. Algorithm

We describe our algorithm given an arbitrary $n \times n$ matrix $A \in \mathcal{S}$ (instead of $\mathcal{W}$) and a transfer function h. Our solution is based on a sparse low-rank approximation of A computed offline such that online, $\mathbf{x} \approx h(A)\mathbf{y}$ is reduced to a sequence of sparse matrix-vector multiplications. The approximation is based on a randomized algorithm [45] that is similar to *Nyström sampling* [12] but comes with performance guarantees [18, 66]. In the following, $r \ll n$, $p < r$, q and τ are given parameters, and $\hat{r} = r + p$.

1. (*Offline*) Using *simultaneous iteration* [60, §28], compute an $n \times \hat{r}$ matrix Q with orthonormal columns that represents an approximate basis for the range of A, *i.e.* $QQ^\top A \approx A$. In particular, this is done as follows [18, §4.5]: randomly draw an $n \times \hat{r}$ 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*) Compute an approximate rank-r eigenvalue decomposition $U\Lambda U^\top \approx A$, where $n \times r$ matrix U has orthonormal columns and $r \times r$ matrix Λ is diagonal. In particular, roughly following [18, §5.3]:
    (a) Form the $\hat{r} \times \hat{r}$ matrix $C := Q^\top B = Q^\top AQ$.
    (b) Compute its eigendecomposition $\hat{V}\hat{\Lambda}\hat{V}^\top = C$.
    (c) Form (V, Λ) 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 τ largest entries and dropping the rest.

4. (*Online*) Given $\mathbf{y}$, compute

$$\mathbf{x} := Uh(\Lambda)U^\top\mathbf{y}. \qquad (4)$$

Observe that $U^\top$ projects $\mathbf{y}$ onto $\mathbb{R}^r$. With Λ being diagonal, $h(\Lambda)$ is computed element-wise, so h can be given online. Finally, multiplying by U and ranking $\mathbf{x}$ amounts to dot product similarity search in $\mathbb{R}^r$.

2.3. Retrieval

The above algorithm solves any problem of the form (2) satisfying our assumptions. We now apply it to image retrieval, following [23]. We are given a dataset of N images, each represented by m *region descriptors* in $\mathbb{R}^d$. Global image description is just the special case $m = 1$. The entire dataset is represented by a total of $n = Nm$ descriptors $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, with each $\mathbf{v}_i$ associated to vertex v_i of G. Descriptors are obtained by sampling a set of rectangular regions on CNN activations of a network, followed by pooling, normalization, PCA, whitening, and optionally reduced by a Gaussian mixture model [23].

Given descriptors $\mathbf{v}, \mathbf{z} \in \mathbb{R}^d$, we measure their *similarity* by $s(\mathbf{v}, \mathbf{z}) = (\mathbf{v}^\top\mathbf{z})^\gamma$, where exponent $\gamma > 0$ is a parameter. We denote by $s(\mathbf{v}_i|\mathbf{z})$ the similarity $s(\mathbf{v}_i, \mathbf{z})$ if $\mathbf{v}_i$ is a k-NN of $\mathbf{z}$ in $\mathcal{V}$ and zero otherwise. The sparse $n \times n$ *similarity matrix* S is defined element-wise as $s_{ij} := s(\mathbf{v}_i|\mathbf{v}_j)$, and the symmetric *adjacency matrix* W as $w_{ij} := \min(s_{ij}, s_{ji})$, representing *mutual* neighborhoods.

Given a query image represented by descriptors $\{\mathbf{q}_1, \dots, \mathbf{q}_m\} \subset \mathbb{R}^d$, we form the *observation vector* $\mathbf{y} \in \mathbb{R}^n$ with elements $y_i := \sum_{j=1}^m s(\mathbf{v}_i|\mathbf{q}_j)$ by pooling over query regions. We make $\mathbf{y}$ sparse by keeping the k largest entries and dropping the rest, and compute the ranking vector $\mathbf{x} \in \mathbb{R}^n$ by (4), containing the *ranking score* x_i of each dataset region $\mathbf{v}_i$. To obtain a score per image, we perform a linear pooling operation [23] represented as $\overline{\mathbf{x}} := \Sigma\mathbf{x}$ where Σ is a sparse $N \times n$ *pooling matrix*. The $N \times r$ matrix $\overline{U} := \Sigma U$ is indeed computed offline so that we directly compute $\overline{\mathbf{x}} = \overline{U}h(\Lambda)U^\top\mathbf{y}$ online.

Computing $\mathbf{y}$ involves Euclidean search in $\mathbb{R}^d$, which happens to be dot product because vectors are ℓ^2-normalized. Applying $\overline{U}$ and ranking $\mathbf{x}$ 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 $\mathbf{y}$ and rows of $\overline{U}$, whence the cost, are the same for global or regional search.

3. Analysis

We refer to our algorithm as *fast spectral ranking* (FSR) with 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 $\hat{r} = n$, $Q = I$, $B = A$ in stage 2.
- FSR.EXACT: same as FSR.RANK-r for $r = n$.

Given observation vector $\mathbf{y} \in \mathbb{R}^n$, $n \times n$ matrix $A \in \mathcal{S}$, and function h, we denote by $\mathbf{x} = \text{FSR.v}(\mathbf{y}|A, h) \in \mathbb{R}^n$ the vector obtained by variant v of the algorithm. We consider here functions h for which there exists a series expansion

$$h(A) = \sum_{t=0}^\infty c_t A^t \qquad (5)$$

for $A \in \mathcal{S}$. We denote by $\mathcal{H}$ the family of such functions.

3.1. Correctness

Algorithm FSR.EXACT is not useful: U is a dense $n \times n$ matrix in this case and even if space is not an issue, one is still better off solving linear system (3) as in [23] rather than using (4). However, showing its correctness is instructive in light of its interpretation in section 4.

Theorem 1. *If $h \in \mathcal{H}$ and series (5) converges, then* $\text{FSR.EXACT}(\mathbf{y}|A, h) = h(A)\mathbf{y}$. *In particular,* $\text{FSR.EXACT}(\mathbf{y}|\mathcal{W}, h_\alpha) = \mathbf{x}^*$.

Proof. Matrix A is real symmetric hence diagonalizable, and FSR.EXACT computes its exact eigenvalue decomposition $U\Lambda U^\top = A$, where U is orthogonal. Because the series converges, $h(A) = Uh(\Lambda)U^\top = U\,\text{diag}(h(\lambda_1), \dots, h(\lambda_n))U^\top$ [1, §9.14,9.2]. In particular, $h_\alpha \in \mathcal{H}$ and has the *geometric progression* expansion

$$h_\alpha(A) := \beta(I - \alpha A)^{-1} = \beta\sum_{t=0}^\infty (\alpha A)^t, \qquad (6)$$

which converges absolutely if $\varrho(\alpha A) < 1$ [1, §9.6,9.19]. This holds for $A = \mathcal{W}$ because $\alpha < 1$ and $\varrho(\mathcal{W}) = 1$. $\qquad\square$

3.2. 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 $n\hat{r}$. The online complexity is determined by the number of nonzero entries in matrix 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 $O(nr)$.
- FSR.SPARSE: The offline complexity is $O(qn(k + \hat{r})\hat{r} + \tau\log\tau)$ time and $O(n\hat{r})$ space; its online complexity is $O(\tau)$.

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.

3.3. Error bound

We present main ideas for bounding the approximation error of FSR.RANK-r and FSR.APPROX coming from literature. The approximation $QQ^\top A \approx A$ of stage 1 is studied in [18, §9.3,10.4]: an average-case bound on $\left\| A - QQ^\top A \right\|$ decays exponentially fast in the number of iterations q to $|\lambda_{r+1}|$. Stage 2 yields an approximate eigenvalue decomposition of A: Since A is symmetric, $A \approx QQ^\top A QQ^\top = QCQ^\top \approx QV\Lambda V^\top Q^\top = U\Lambda U^\top$. The latter approximation $C \approx V\Lambda V^\top$ is essentially a best rank-r approximation of $C = Q^\top AQ$. This is also studied in [18, §9.4] for the truncated SVD case of a non-symmetric matrix. It involves an additional term of $|\lambda_{r+1}|$ in the error.

We are actually approximating $h(A)$ by $Uh(\Lambda)U^\top$, so that $|h(\lambda_{r+1})|$ governs the error instead of $|\lambda_{r+1}|$. A similar situation appears in [59, §3.3]. Therefore, our method makes sense only when the restriction of h to scalars is non-decreasing. This is the case for h_α.

4. Interpretation

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

4.1. Random walks

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

$$\mathbf{x}^{(t)} := \alpha A \mathbf{x}^{(t-1)} + (1 - \alpha)\mathbf{y}. \tag{7}$$

If A is a stochastic *transition matrix* and $\mathbf{x}^{(0)}, \mathbf{y}$ 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 α or jumps to a vertex according to distribution $\mathbf{y}$ with probability $1 - \alpha$. This is called a *Markov chain with restart* [2] or *random walk with restart* [38]. State $\mathbf{x}^{(t)}$ converges to $\mathbf{x}^* = h_\alpha(A)\mathbf{y}$ as $t \to \infty$ provided $\varrho(\alpha A) < 1$ [67]. In fact, (7) is equivalent to *Jacobi solver* [17] on linear system (3) [23].

If $\mathbf{y} = \mathbf{e}_i$, the i-th canonical vector, then $\mathbf{x}^*$ is used to rank the vertices of G, expressing a measure of "similarity" to v_i [68]. Parameter α controls how much $\mathbf{x}^*$ is affected by *boundary condition* $\mathbf{y}$ [62]: $\mathbf{x}^*$ equals $\mathbf{y}$ for $\alpha = 0$, while in the limit $\alpha \to 1$, $\mathbf{x}^*$ tends to a dominant eigenvector of A. Indeed, for $\alpha = 1$, (7) becomes a power iteration.

4.2. Random fields

Given a positive-definite $n \times n$ *precision matrix* $A \in \mathcal{S}$ and a *mean vector* $\boldsymbol{\mu} \in \mathbb{R}^n$, a *Gaussian Markov random field* (GMRF) [47] with respect to an undirected graph G is a random vector $\mathbf{x} \in \mathbb{R}^n$ with normal density $p(\mathbf{x}) := \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A^{-1})$ iff A has the same nonzero off-diagonal entries as the adjacency matrix of G. Its *canonical parametrization* $p(\mathbf{x}) \propto e^{-E(\mathbf{x}|\mathbf{b},A)}$ where $E(\mathbf{x}|\mathbf{b}, A) :=$

$\frac{1}{2}\mathbf{x}^\top A\mathbf{x} - \mathbf{b}^\top \mathbf{x}$ is a quadratic *energy*. Its expectation $\boldsymbol{\mu} = A^{-1}\mathbf{b}$ is the minimizer of this energy. Now, $\mathbf{x}^* = \mathcal{L}_\alpha^{-1}\mathbf{y}$ (3) is the expectation of a GMRF with energy

$$f_\alpha(\mathbf{x}) := E(\mathbf{x}|\mathbf{y}, \mathcal{L}_\alpha) = \frac{1}{2}\mathbf{x}^\top \mathcal{L}_\alpha \mathbf{x} - \mathbf{y}^\top \mathbf{x}. \tag{8}$$

A *mean field* method on this GMRF is equivalent to *Jacobi* or *Gauss-Seidel* solvers on (3) [64]. Yet, *conjugate gradients* (CG) [35] is minimizing $f_\alpha(\mathbf{x})$ more efficiently [23, 5].

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

$$\alpha \sum_{i,j} w_{ij} \|\hat{x}_i - \hat{x}_j\|^2 + (1 - \alpha)\|\mathbf{x} - \mathbf{y}\|^2, \tag{9}$$

where $\hat{\mathbf{x}} := D^{-1/2}\mathbf{x}$. The pairwise *smoothness term* encourages $\mathbf{x}$ to vary little across edges with large weight whereas the unary *fitness term* to stay close to observation $\mathbf{y}$ [67]. Again, α controls the trade-off: $\mathbf{x}^*$ equals $\mathbf{y}$ for $\alpha = 0$, while for $\alpha \to 1$, $\mathbf{x}^*$ tends to be constant over connected components like dominant eigenvectors of $\mathcal{W}$.

4.3. Regularization and kernels

The first term of (8) is interpreted as a *regularization operator* related to a kernel $K = \mathcal{L}_\alpha^{-1}$ [56, 55, 30]. In a finite graph, a *kernel* can be seen either as an $n \times n$ matrix K or a function $\kappa : V^2 \to \mathbb{R}$ operating on pairs of vertices. More generally, if $h(x) > 0$ for $x \in \mathbb{R}$, which holds for h_α, then $K := h(\mathcal{W})$ is positive-definite and there is an $n \times n$ matrix Φ such that $K = \Phi^\top \Phi$, or $\kappa(v_i, v_j) = \phi(v_i)^\top \phi(v_j)$ where *feature map* $\phi : V \to \mathbb{R}^n$ is given by $\phi(v_i) := \Phi\mathbf{e}_i$. A particular choice for Φ is

$$\Phi := h(\Lambda)^{1/2}U^\top \tag{10}$$

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

The goal of *out-of-sample extension* is to compute a "similarity" $\hat{\kappa}(\mathbf{z}_1, \mathbf{z}_2)$ between two unseen vectors $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^d$ not pertaining to the graph. Here we define

$$\hat{\kappa}(\mathbf{z}_1, \mathbf{z}_2) := \psi(\mathbf{z}_1)^\top \Phi^\top \Phi\psi(\mathbf{z}_2) \tag{11}$$

given any mapping $\psi : \mathbb{R}^d \to \mathbb{R}^n$, *e.g.* $\psi(\mathbf{z})_i := s(\mathbf{v}_i|\mathbf{z})$ discussed in section 2.3. This extended kernel is also positive-definite and its embedding $\hat{\phi}(\mathbf{z}) = \Phi\psi(\mathbf{z})$ is a linear combination of the dataset embeddings. For $r \ll n$, our method allows rapid computation of κ or $\hat{\kappa}$ for any given function h, without any dense $n \times n$ matrix involved.

4.4. Paths on graphs

Many *nonlinear dimension reduction* methods replace Euclidean distance with an approximate *geodesic distance*,

assuming the data lie on a *manifold* [32]. This involves the *all-pairs shortest path* (APSP) problem and Dijkstra's algorithm is a common choice. Yet, it is instructive to consider a naïve algorithm [9, §25.1]. We are given a *distance matrix* where missing edges are represented by ∞ 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*[2] of weights over paths of length t, recovering the ordinary matrix power $\mathcal{W}^t$, and the weighted sum over all lengths $\sum_{t=0}^{\infty} c_t \mathcal{W}^t$, where coefficients $(c_t)_{t \in \mathbb{N}}$ allow for convergence [62], [50, §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* [62], which justifies function h_α (6).

4.5. Graph signal processing

In *signal processing* [36], a discrete-time *signal* of period n is a vector $\mathbf{s} \in \mathbb{R}^n$ where indices are represented by integers modulo n, that is, $s_{\bar{i}} := s_{(i \bmod n)+1}$ for $i \in \mathbb{Z}$. A *shift* (or translation, or delay) of $\mathbf{s}$ by one sample is the mapping $s_{\bar{i}} \mapsto s_{\overline{i-1}}$. If we define the $n \times n$ circulant matrix $C_n := (\mathbf{e}_2 \, \mathbf{e}_3 \ldots \mathbf{e}_n \, \mathbf{e}_1)$[3], a shift can be represented by $\mathbf{s} \mapsto C_n \mathbf{s}$ [48]. A linear, time (or shift) invariant *filter* is the mapping $\mathbf{s} \mapsto H\mathbf{s}$ where H is an $n \times n$ matrix with a series representation $H := h(C_n) = \sum_{t=0}^{\infty} h_t C_n^t$. Matrix C_n has the eigenvalue decomposition $U \Lambda U^\top$ where $U^\top$ is the $n \times n$ *discrete Fourier transform* matrix $\mathcal{F}$. If the series $h(C_n)$ converges, filtering $\mathbf{s} \mapsto H\mathbf{s}$ is written as

$$\mathbf{s} \mapsto \mathcal{F}^{-1} h(\Lambda) \mathcal{F} \mathbf{s}. \tag{12}$$

That is, $\mathbf{s}$ is mapped to the *frequency domain*, scaled element-wise, and mapped back to the time domain.

Graph signal processing [48, 52] generalizes the above concepts to graphs by replacing C_n by $\mathcal{W}$, an appropriately normalized adjacency matrix of an arbitrary graph. If $U \Lambda U^\top$ is the eigenvalue decomposition of $\mathcal{W}$, we realize that (4) treats $\mathbf{y}$ as a (sparse) *signal* and filters it in the frequency domain via transfer function h to obtain $\mathbf{x}$. Function h_α in particular is a *low-pass filter* that reconstructs $\mathbf{x}$ even from a single nonzero sample $\mathbf{y}$ over the graph. By varying α from 0 to 1, the frequency response varies from all-pass to sharp low-pass, allowing only the DC component.

---

[2]In fact, similar to *softmax* due to the exponential and normalization.

[3]Observe that C_n is the adjacency matrix of the directed graph of Figure 1 after adding an edge from the rightmost to the leftmost vertex.

5. Related work

The history of the particular case $h = h_\alpha$ is the subject of the excellent study of *spectral ranking* [62]. The fundamental contributions originate in the social sciences and include the eigenvector formulation by Seeley [49], damping by α (6) by Katz [28] and the boundary condition $\mathbf{y}$ (3) by Hubbell [22]. The most well-known follower is PageRank [37]. In machine learning, h_α has been referred to as the *von Neumann* [26, 50] or *regularized Laplacian* kernel [55]. Along with the *diffusion kernel* [31, 30], it has been studied in connection to *regularization* [56, 55].

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 [57, 5]. A similar mechanism appears in *semi-supervised learning* [67, 71, 69, 6] or *interactive segmentation* [16, 29] 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 [68, 67, 38] is an alternative interpretation to retrieval [11]. Yet, directly solving a linear system is superior [23]. Offline matrix decomposition has been studied for RWR [59, 13, 25]. All three methods are limited to h_α while sparse LU decomposition [13, 25] assumes an uneven distribution of vertex degrees [27], 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 throught the kernel $K = \mathcal{L}_\alpha^{-1}$.

In the general case, the spectral formulation (4) has been known in machine learning [6, 50, 34, 70, 63] and in graph signal processing [48, 52, 19]. The latter is becoming popular in the form of *graph-based convolution* in deep learning [4, 21, 10, 3, 33, 41]. 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, 53] or translation-invariant neighborhood templates in the spatial domain [33, 41].

We cast *retrieval as graph filtering* by constructing an appropriate observation vector. We actually perform the computation in the *frequency domain* via a scalable solution. 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* [61, 46]. This roughly corresponds to a single iteration of our step 1. Our solution is thus more accurate, while h can be specified online.

6. 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 [27]. For this reason we find the connected components with the *union-find* algorithm [9] and reorder vertices such that A is block diagonal: $A = \mathrm{diag}(A_1, \ldots, A_c)$. For each $n_l \times n_l$ matrix A_l, we apply offline stages 1-3 to obtain an approximate rank-r_l eigenvalue decomposition $\hat{U}_l \hat{\Lambda}_l \hat{U}_l^\top \approx A_l$ with $r_l = \max(\rho, \lceil rn_l/n \rceil)$ if $n_l > \rho$, otherwise we compute an exact decomposition. Integer ρ is a given parameter. We form (U_l, Λ_l) by keeping up to ρ slices from each pair $(\hat{U}_l, \hat{\Lambda}_l)$ and complete with up to r slices in total, associated to the largest eigenvalues of the entire set $\mathrm{diag}(\hat{\Lambda}_1, \ldots, \hat{\Lambda}_c)$. Online, we partition $(\mathbf{y}_1; \ldots; \mathbf{y}_c) = \mathbf{y}$, compute each $\mathbf{x}_l$ from $\mathbf{y}_l$ by (4) and form back vector $\mathbf{x} = (\mathbf{x}_1; \ldots; \mathbf{x}_c)$.

Sparse neighborhoods. Denote by η_i the ℓ_2-norm of the i-th row of U. FSR.EXACT yields $\boldsymbol{\eta} = \mathbf{1}$ but this is not the case for FSR.APPROX. Larger (smaller) values appear to correspond to densely (sparsely) populated parts of the graph. For small rank r, norms η_i are more severely affected for uncommon than common vectors in the dataset. We propose replacing each element x_i of (4) by

$$x_i' = x_i + (1 - \eta_i)\mathbf{v}_i^\top \mathbf{q}, \tag{13}$$

for global descriptors, with a straightforward extension for regional ones. This is referred to as FSRw.APPROX and is a weighted combination of manifold search and Euclidean search. It approaches the former for common elements and to the latter for uncommon ones. Our experiments show that this is essential at large scale.

7. Experiments

This section introduces our experimental setup, investigates the performance and behavior of the proposed method and its application to large-scale image retrieval.

7.1. Experimental Setup

Datasets. We use three image retrieval benchmarks: Oxford Buildings (Oxford5k) [39], Paris (Paris6k) [40] and Instre [65], with the evaluation protocol introduced in [23] for the latter. We conduct large-scale experiments by following a standard protocol of adding 100k distractor images from Flickr [39] 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 [23]. Global description is R-MAC with 3 different scales [58], including the full image as a separate region. Regional descrip-
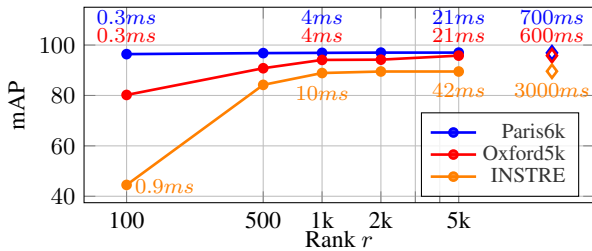

Figure 2: Performance of regional search with FSR.RANK-r. Runtimes are reported in text labels. $\diamond$ refers to FSR.EXACT performed with conjugate gradients as in [23]

tors 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 [43]. In particular we work with d-dimensional vectors extracted from VGG [54] ($d = 512$) and ResNet101 [20] ($d = 2,048$) networks fine-tuned specifically for image retrieval [43, 15].

Implementation. We adopt the same parameters for graph construction and search as in [23]. A monomial kernel is used for pairwise descriptor similarity, *i.e.* $s(\mathbf{v}, \mathbf{z}) = (\mathbf{v}^\top \mathbf{z})^3$. We use $\alpha = 0.99$, and keep the top $k = 50$ and $k = 200$ mutual neighbors when constructing 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 [23]. 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 some other component is just copied from y_i. This choice works well because the largest component holds nearly all data in practice. Time measurements are reported with a 4-core Intel Xeon 2.00GHz CPU. Runtimes refer to the *search time* excluding the construction of the observation vector, since this task is common to all baseline and our methods.

7.2. Retrieval Performance

Rank-r. We analyze the performance of FSR.RANK-r for varying values of r. Parameter r affects the quality of the approximation and defines the dimensionality of the embedding space. The initial vectors in $\mathbb{R}^d$ are projected to an r-dimensional space where manifold search is enabled.

Figure 2 reports performance for regional search. The optimal value of r depends on the structure of the dataset. In all cases the optimal performance appears already before $r = 1\mathrm{k}$. In particular for Paris6k, performance reaches its peak with as small as $r = 100$. Compared to FSR.EXACT as implemented in [23], it achieves the same mAP but 150 times faster on Oxford5k and Paris6k, while 300 times faster on Instre. Global search demonstrates a similar behavior but is skipped due to lack of space.

We achieve 97.0 mAP on Paris6k which is near-perfect

(La Défense, AP: 92.1)   #5   #32   #51   #70   #71   #76   #79   #126

(Pyramide du Louvre, AP: 92.7)   #2   #4   #8   #61   #68   #72   #75   #108

Figure 3: Two queries with the *lowest* AP from Paris6k and the corresponding top-ranked negative images based on the ground-truth. The left-most image contains the query object, and its AP is reported underneath it. Images considered to be negative based on the ground-truth are shown with their rank underneath. Ranks are marked in blue for incorrectly labeled images that are visually relevant (overlapping), and red otherwise.

performance. Figure 3 presents the two queries with the lowest average precision and their top-ranked negative images based on the ground-truth. It appears that in most cases the ground-truth is incorrect, as these images have visual overlap with the query bounding box. In both cases, we retrieve the object of interest with variations thanks to manifold search. The first true (visually) negative image for "La Défense" query appears at rank 126. This image does not contain "La Grande Arche" but it depicts buildings from the surroundings due to "topic drift" of manifold search. The same outcome is seen for the other query, where the first visually negative image containing "Palais du Louvre" is at rank 108.

Regional search performs better than the global one [23] at the cost of more memory and slower query time. Our approach unlocks this bottleneck thanks to the offline pooling $\overline{U} = \Sigma U$. 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. The mAP equals 89.5 with FSR.RANK-5000 and regional search on Instre. The drop is small with FSR.APPROX (89.2) while the offline computation is 20 times faster: it takes 3 hours instead of 60 hours with 570k Instre regional descriptors, using 16-core Intel Xeon 2.00GHz CPU. This is important at large scale because the off-line complexity of FSR.RANK-r increases polynomially.

7.3. Large-scale experiments

We now apply our approach to a larger scale by using only 5 descriptors per image thanks to a GMM reduction [23]. This choice improves the scalability while minimizing the accuracy losses.

FSRw.**APPROX** becomes crucial, especially at large scale, because vectors of sparsely populated parts of the graph are not well represented. Figure 4 shows the comparison be-
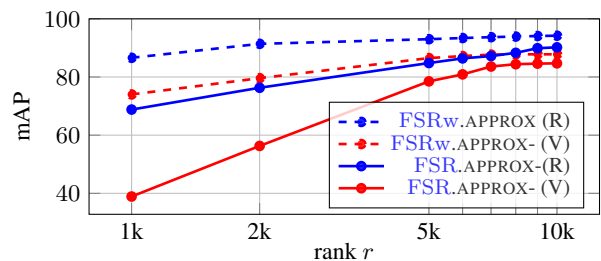

Figure 4: Oxford105k with FSR.APPROX and FSRw.APPROX using Resnet101(R) and VGG(V).

tween FSRw.APPROX and FSR.APPROX. We achieve 94.2 and 90.2 with FSRw.APPROX and FSR.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 5. Low values of r penalize sparsely populated parts of the graph, *i.e.* landmarks with less similar instances in the dataset. FSRw.APPROX partly solves this issue.

The search time is $0.14s$. and $0.3s$. per query for $r = 5k$ and $r = 10k$ respectively in Oxford105k. It is two orders of magnitude faster than FSR.EXACT: The implementation of [23] requires about 14 seconds per query. Iscen *et al.* [23] reduce this timing thanks to 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 4% and our method is still 1 second faster.

Sparse embeddings. We set r to a large enough value, such as 10k, to avoid compromising the search accuracy. This results in r-dimensional embeddings. Most descriptors belong only to few manifolds and each embedding vector has high energy in the components of the corresponding manifolds. Figure 6 presents the results of FSRw.APPROX. Remarkably, the drop equals $\%2$ mAP for embeddings that are $\approx 90\%$ sparse. Making the embeddings sparser yields memory savings.
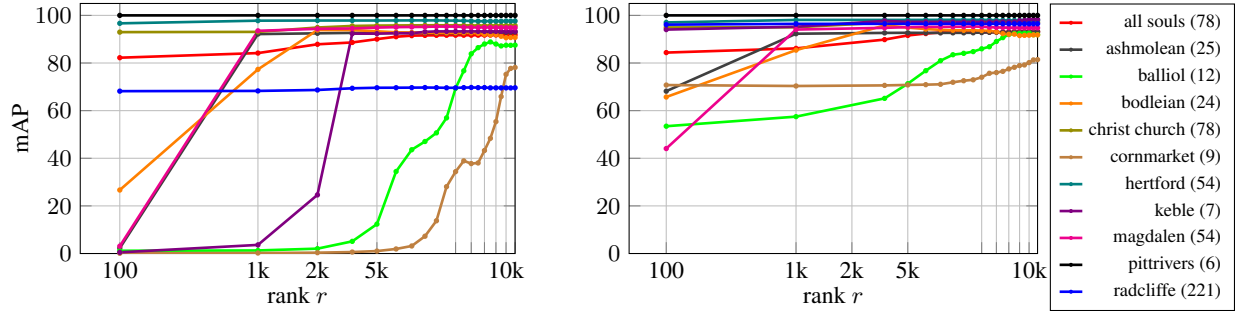
Figure 5: mAP reported separately for each landmark in Oxford105k with FSR.APPROX (left) and FSRw.APPROX (right). Number of positive images per landmark queries is shown in the labels.
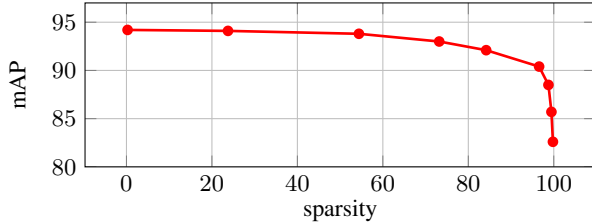
Legend:
- all souls (78)
- ashmolean (25)
- balliol (12)
- bodleian (24)
- christ church (78)
- cornmarket (9)
- hertford (54)
- keble (7)
- magdalen (54)
- pittrivers (6)
- radcliffe (221)

Figure 6: Effect of sparsification of U in Oxford105k with FSRw.APPROX and Resnet101 descriptors. The τ largest values of U are kept to achieve different levels of sparsity.

Quantized descriptors. Construction of the observation vector requires storage of the initial descriptors. We further use product quantization (PQ) [24] to compress those and perform approximate search method in Euclidean space. When using PQ with 64 and 256 bytes, we achieve mAP equal to 91.1 and 94.2 in Oxford105k, respectively, while without PQ 94.4 is achieved. This comparison is performed with FSRw.APPROX.

7.4. 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 [23]. We do not experiment large-scale without GMM since there is not much improvement any more and it is less scalable. Our method reaches performance similar to that of FSR.EXACT as evaluated with CG [23]. 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.

8. Discussion

This work reproduces the excellent results of online linear system solution [23] 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, embar-

Method	$m \times d$	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
Global descriptors - Euclidean search						
R-MAC [43]	512	47.7	77.7	70.1	84.1	76.8
R-MAC [15]	2,048	62.6	83.9	80.8	93.8	89.9
Global descriptors - Manifold search						
Diffusion [23]	512	70.3	85.7	82.7	94.1	92.5
FSR.RANK-r	512	70.3	85.8	85.0	93.8	92.4
Diffusion [23]	2,048	80.5	87.1	87.4	96.5	95.4
FSR.RANK-r	2,048	80.5	87.5	87.9	96.4	95.3
Regional descriptors - Euclidean search						
R-match [44]	21×512	55.5	81.5	76.5	86.1	79.9
R-match [44]	21×2,048	71.0	88.1	85.7	94.9	91.3
Regional descriptors - Manifold search						
Diffusion [23]	5×512	77.5	91.5	84.7	95.6	93.0
FSR.APPROX	5×512	78.4	91.6	86.5	95.6	92.4
Diffusion [23]	21×512	80.0	93.2	90.3	96.5	92.6
FSR.APPROX	21×512	80.4	93.0	-	96.5	-
Diffusion [23]	5×2,048	88.4	95.0	90.0	96.4	95.8
FSR.APPROX	5×2,048	88.5	95.1	93.0	96.5	95.2
Diffusion [23]	21×2,048	89.6	95.8	94.2	96.9	95.3
FSR.APPROX	21×2,048	89.2	95.8	-	97.0	-

Table 1: Performance comparison to the baseline methods and to the state of the art on manifold search [23]. Points at 512D are extracted with VGG [43] and at 2048D with ResNet101 [15]. Regional representation with $m = 5$ descriptors per image uses GMM. Large-scale regional experiments use the FSRw.APPROX variant. Dataset truncation is used in [23] at large scale.

rassingly 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.

Although it is not the focus of this work, both Euclidean and dot product search stages can be handled by any efficient search method [24]. Our method is generic and may be used for problems other than search, including clustering or unsupervised learning.

References

[1] K. M. Abadir and J. R. Magnus. *Matrix algebra*. Cambridge University Press, 2005. 3

[2] P. Boldi, V. Lonati, M. Santini, and S. Vigna. Graph fibrations, graph isomorphism, and PageRank. *RAIRO-Theoretical Informatics and Applications*, 40(2):227–253, 2006. 4

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *arXiv preprint arXiv:1611.08097*, 2016. 5

[4] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 5

[5] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In *ECCV*, pages 402–418, 2016. 4, 5

[6] O. Chapelle, J. Weston, and B. Scholkopf. Cluster kernels for semi-supervised learning. *NIPS*, pages 601–608, 2003. 5

[7] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, October 2007. 1

[8] F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997. 2

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. Massachusetts Institute of Technology, 2009. 5, 6

[10] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3837–3845, 2016. 5

[11] M. Donoser and H. Bischof. Diffusion processes for retrieval revisited. In *CVPR*, 2013. 1, 5

[12] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec):2153–2175, 2005. 2

[13] Y. Fujiwara, M. Nakatsuji, M. Onizuka, and M. Kitsuregawa. Fast and exact top-k search for random walk with restart. *Proceedings of the VLDB Endowment*, 5(5):442–453, 2012. 5

[14] A. Gordo, J. Almazan, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. *ECCV*, 2016. 1

[15] A. Gordo, J. Almazan, J. Revaud, and D. Larlus. End-to-end learning of deep visual representations for image retrieval. *arXiv preprint arXiv:1610.07940*, 2016. 1, 6, 8

[16] L. Grady. Random walks for image segmentation. *IEEE Trans. PAMI*, 28(11):1768–1783, 2006. 5

[17] W. Hackbusch. *Iterative solution of large sparse systems of equations*. Springer Verlag, 1994. 4

[18] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. 2, 4

[19] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. 5

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[21] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 5

[22] C. H. Hubbell. An input-output approach to clique identification. *Sociometry*, 1965. 5

[23] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum. Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. In *CVPR*, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[24] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 33(1):117–128, January 2011. 8

[25] J. Jung, K. Shin, L. Sael, and U. Kang. Random walk with restart on large graphs using block elimination. *ACM Transactions on Database Systems*, 41(2):12, 2016. 5

[26] J. Kandola, J. Shawe-Taylor, and N. Cristianini. Learning semantic similarity. In *NIPS*, 2002. 5

[27] U. Kang and C. Faloutsos. Beyond 'caveman communities': Hubs and spokes for graph compression and mining. In *Proceedings of the IEEE International Conference on Data Mining*, pages 300–309. IEEE, 2011. 5, 6

[28] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953. 5

[29] T. H. Kim, K. M. Lee, and S. U. Lee. Generative image segmentation using random walks with restart. In *ECCV*, pages 264–275. Springer, 2008. 5

[30] R. Kondor and J.-P. Vert. Diffusion kernels. *Kernel Methods in Computational Biology*, pages 171–192, 2004. 4, 5

[31] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *ICML*, 2002. 5

[32] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007. 5

[33] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *arXiv preprint arXiv:1611.08402*, 2016. 5

[34] B. Nadler, S. Lafon, R. R. Coifman, and I. G. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *NIPS*, 2005. 5

[35] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006. 4

[36] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing: Pearson New International Edition*. Pearson Higher Ed, 2010. 5

[37] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. 1999. 1, 5

[38] J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. In *International Conference on Knowledge Discovery and Data Mining*. ACM, 2004. 4, 5

[39] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, June 2007. 6

[40] 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*, June 2008. 6

[41] G. Puy, S. Kitic, and P. Pérez. Unifying local and non-local signal processing with graph cnns. *arXiv preprint arXiv:1702.07759*, 2017. 5

[42] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *CVPR*, 2011. 1

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

[44] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4:251–258, 2016. 1, 8

[45] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2009. 2

[46] S. Roux, N. Tremblay, P. Borgnat, P. Abry, H. Wendt, and P. Messier. Multiscale anisotropic texture unsupervised clustering for photographic paper. In *IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2015. 5

[47] H. Rue and L. Held. *Gaussian Markov random fields: theory and applications*. CRC Press, 2005. 4

[48] A. Sandryhaila and J. M. Moura. Discrete signal processing on graphs. IEEE *Transactions on Signal Processing*, 61(7):1644–1656, 2013. 5

[49] J. R. Seeley. The net of reciprocal influence. a problem in treating sociometric data. *Canadian Journal of Experimental Psychology*, 3:234, 1949. 5

[50] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004. 5

[51] X. Shen, Z. Lin, J. Brandt, and Y. Wu. Spatially-constrained similarity measure for large-scale object retrieval. *IEEE Trans. PAMI*, 36(6):1229–1241, 2014. 1

[52] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. 5

[53] D. I. Shuman, P. Vandergheynst, and P. Frossard. Chebyshev polynomial approximation for distributed signal processing. In *International Conference on Distributed Computing in Sensor Systems and Workshops*, pages 1–8. IEEE, 2011. 5

[54] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2014. 6

[55] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines*, pages 144–158. Springer, 2003. 4, 5

[56] A. J. Smola, B. Scholkopf, and K.-R. Muller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649, 1998. 4, 5

[57] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning Gaussian conditional random fields for low-level vision. In *CVPR*, pages 1–8. IEEE, 2007. 5

[58] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. *ICLR*, 2016. 1, 6

[59] H. Tong, C. Faloutsos, and J. Y. Pan. Fast random walk with restart and its applications. In *Proceedings of the IEEE International Conference on Data Mining*, pages 613–622, 2006. 4, 5

[60] L. N. Trefethen and D. Bau III. *Numerical linear algebra*. SIAM, 1997. 2

[61] N. Tremblay and P. Borgnat. Graph wavelets for multiscale community mining. IEEE *Transactions on Signal Processing*, 62(20):5227–5239, 2014. 5

[62] S. Vigna. Spectral ranking. *arXiv preprint arXiv:0912.0238*, 2009. 4, 5

[63] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010. 5

[64] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 649, 2008. 4

[65] S. Wang and S. Jiang. Instre: a new benchmark for instance-level object retrieval and recognition. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11:37, 2015. 1, 6

[66] R. Witten and E. Candes. Randomized algorithms for low-rank matrix factorizations: Sharp performance bounds. *arXiv preprint arXiv:1308.5697*, 2013. 2

[67] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2003. 4, 5

[68] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *NIPS*, 2003. 1, 4, 5

[69] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, 2003. 5

[70] X. Zhu, J. Kandola, J. Lafferty, and Z. Ghahramani. Graph kernels by spectral transforms. *Semi-Supervised Learning*, pages 277–291, 2006. 5

[71] X. Zhu, J. D. Lafferty, and Z. Ghahramani. Semi-supervised learning: From Gaussian fields to Gaussian processes. Technical report, 2003. 5