searching over manifolds of image regions

Yannis Avrithis

Inria Rennes-Bretagne Atlantique

Menlo Park, May 2017

joint work with

Ahmet Iscen

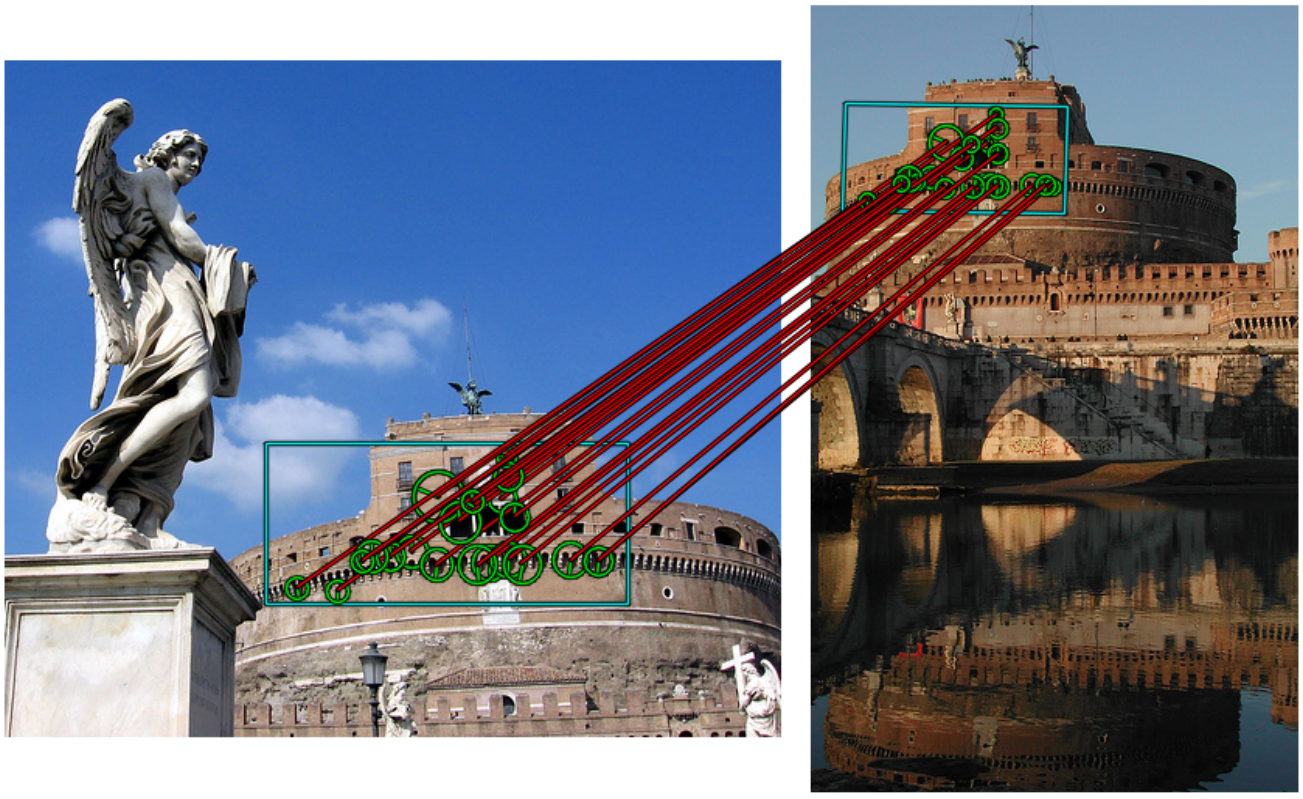
Teddy Furon

Giorgos Tolias

Ondrej Chum

motivation: visual search

challenges



- viewpoint
- lighting
- occlusion
- large scale

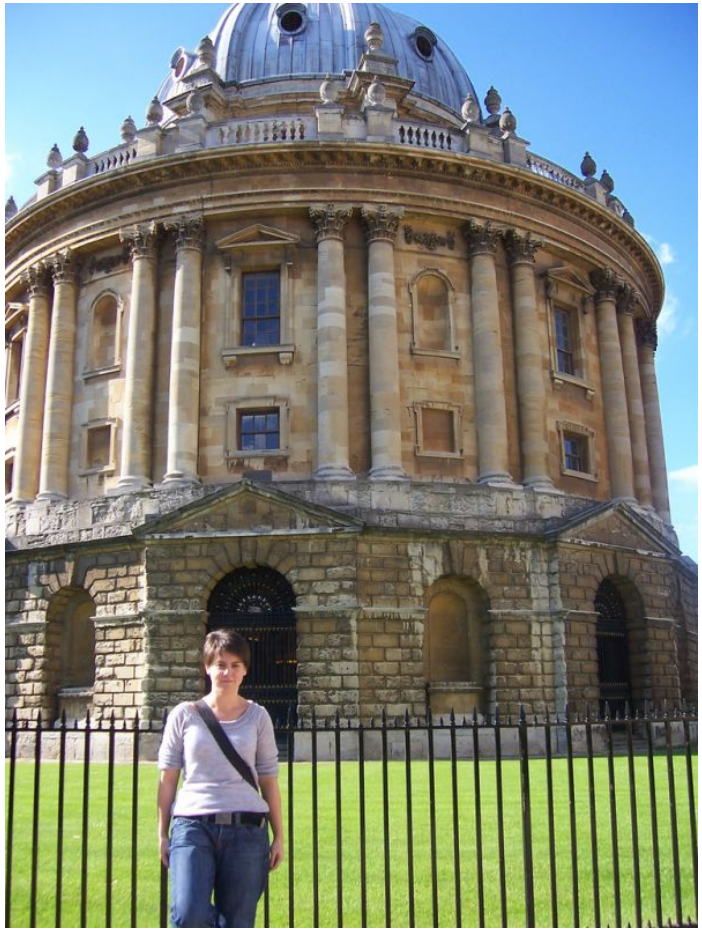
overview

- background
- problem formulation
- diffusion on region manifolds
- fast spectral ranking

background

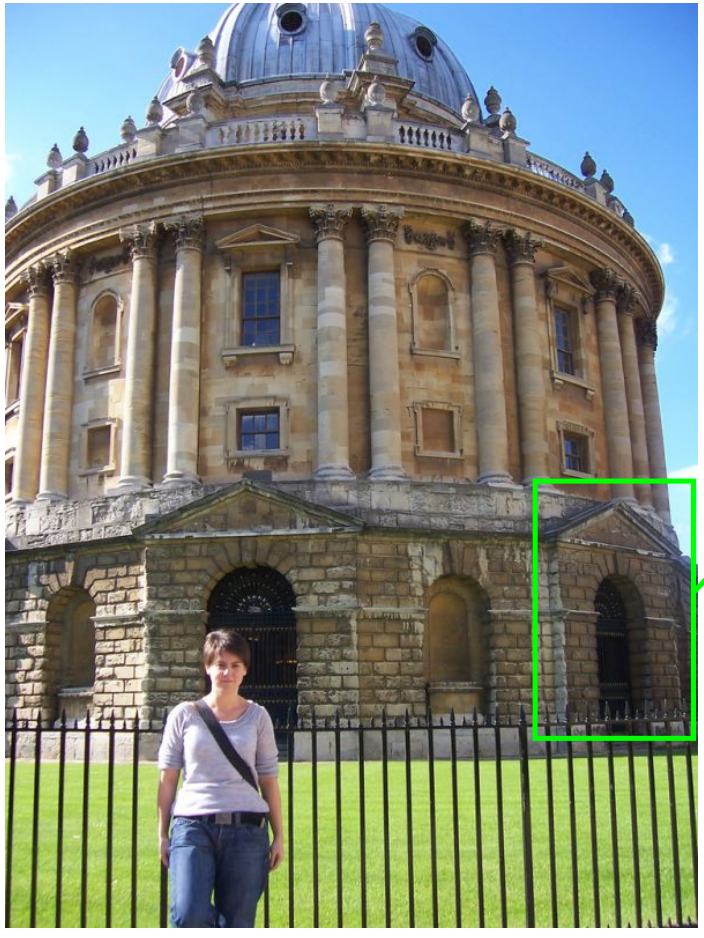
discriminative local features

[Lowe, ICCV 1999]

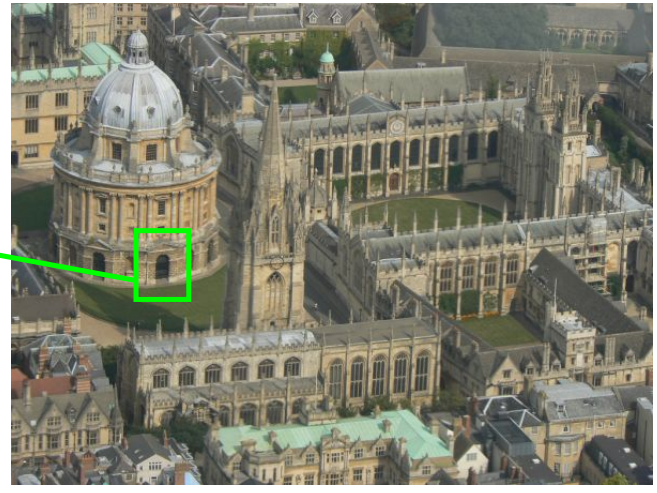


discriminative local features

[Lowe, ICCV 1999]

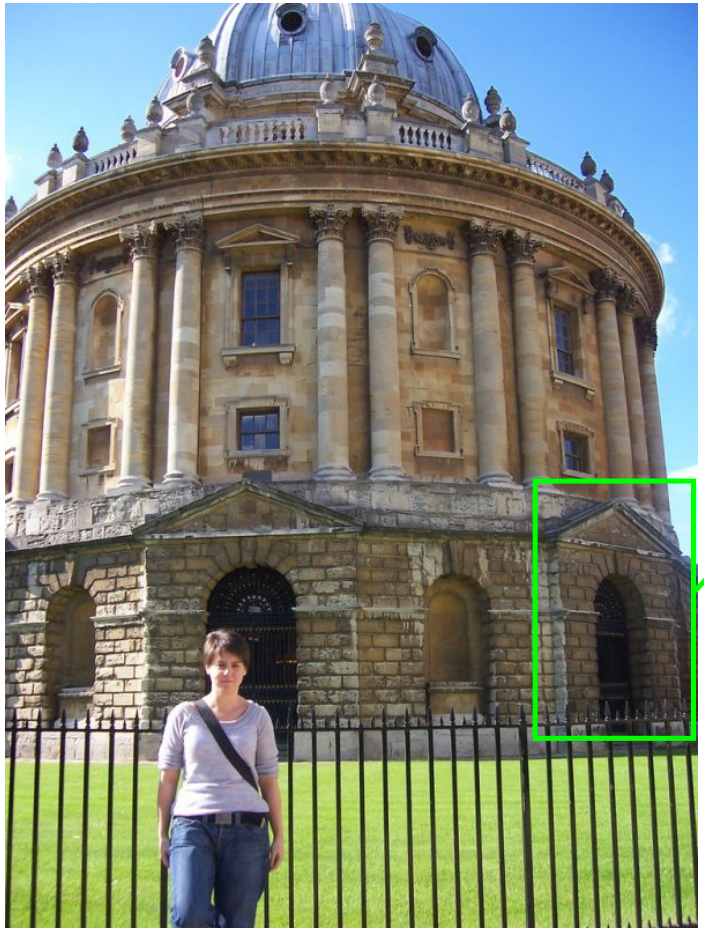


features

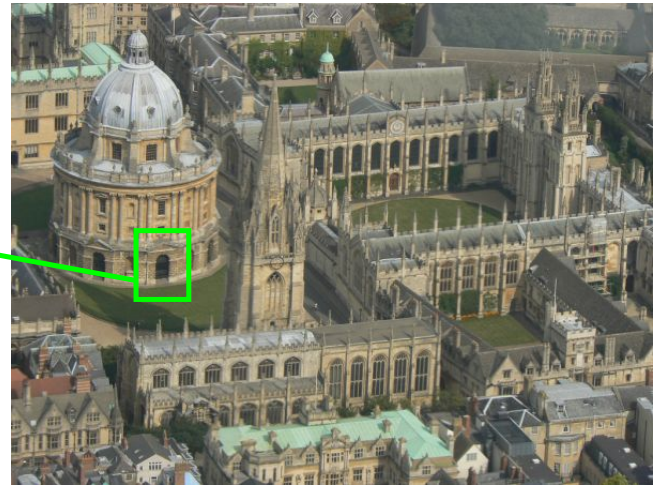


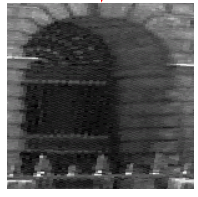
discriminative local features

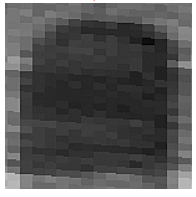
[Lowe, ICCV 1999]



features

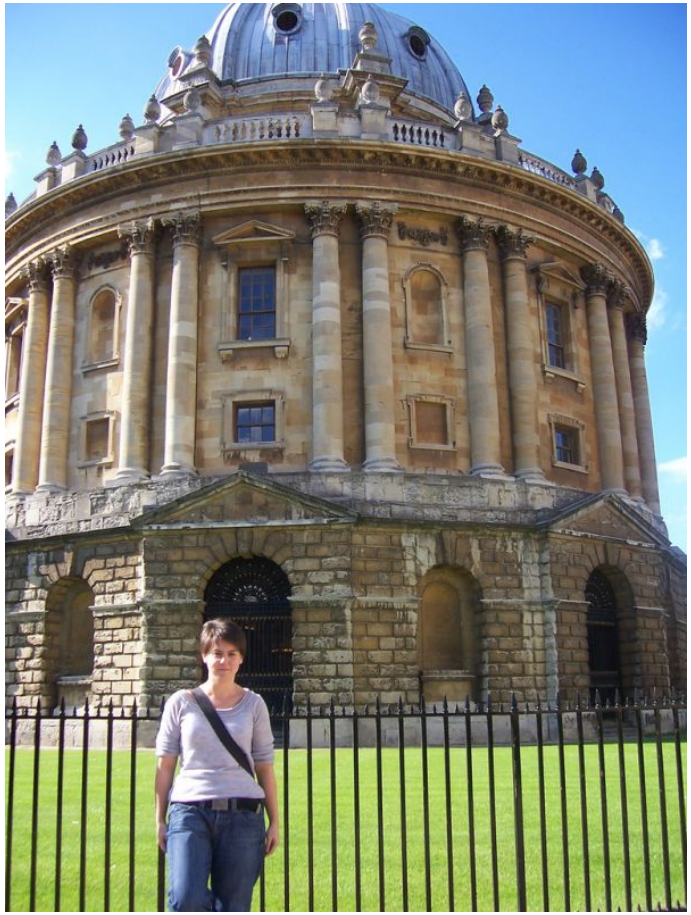






normalized features

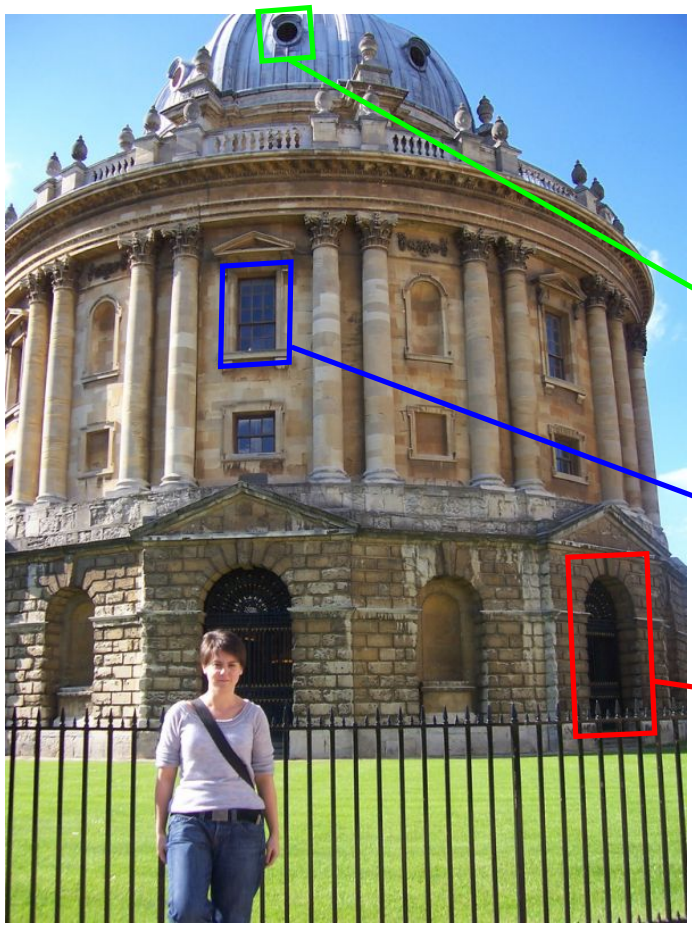
descriptor matching



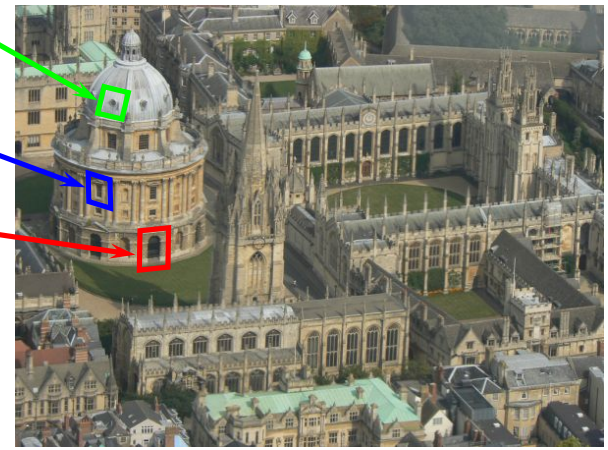
query

15

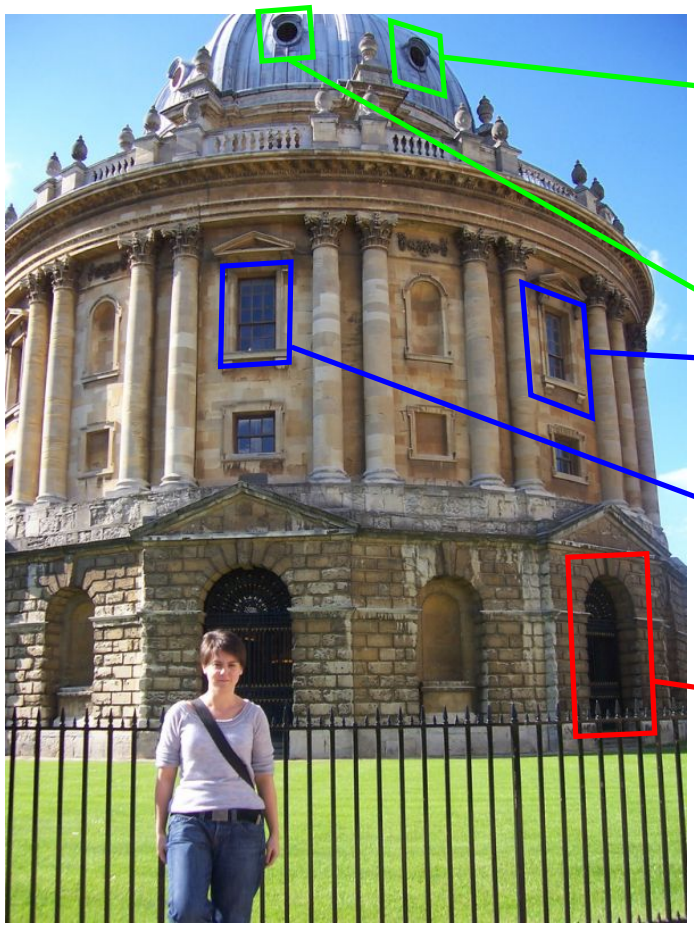
descriptor matching



query

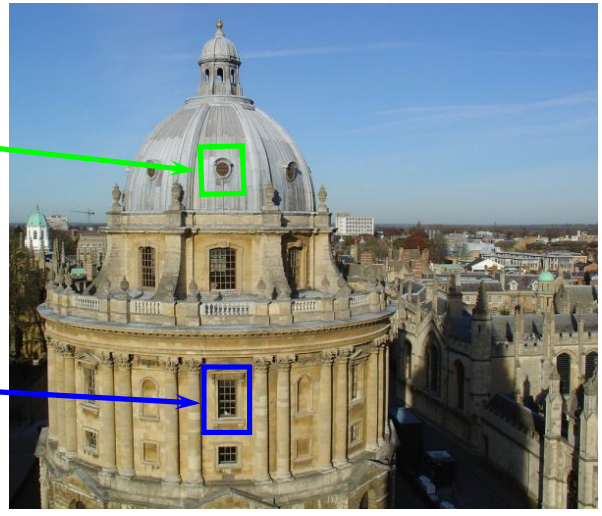


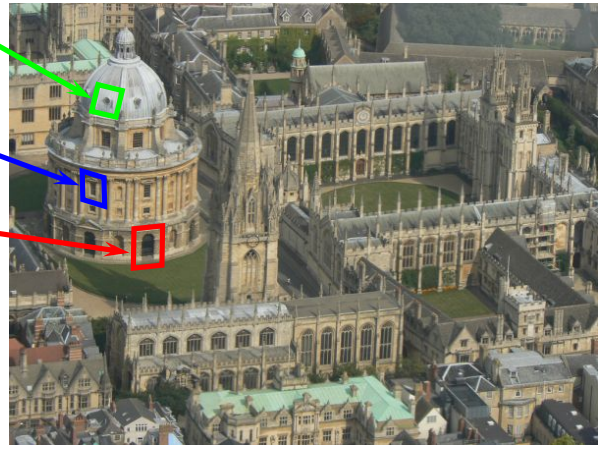
descriptor matching



query

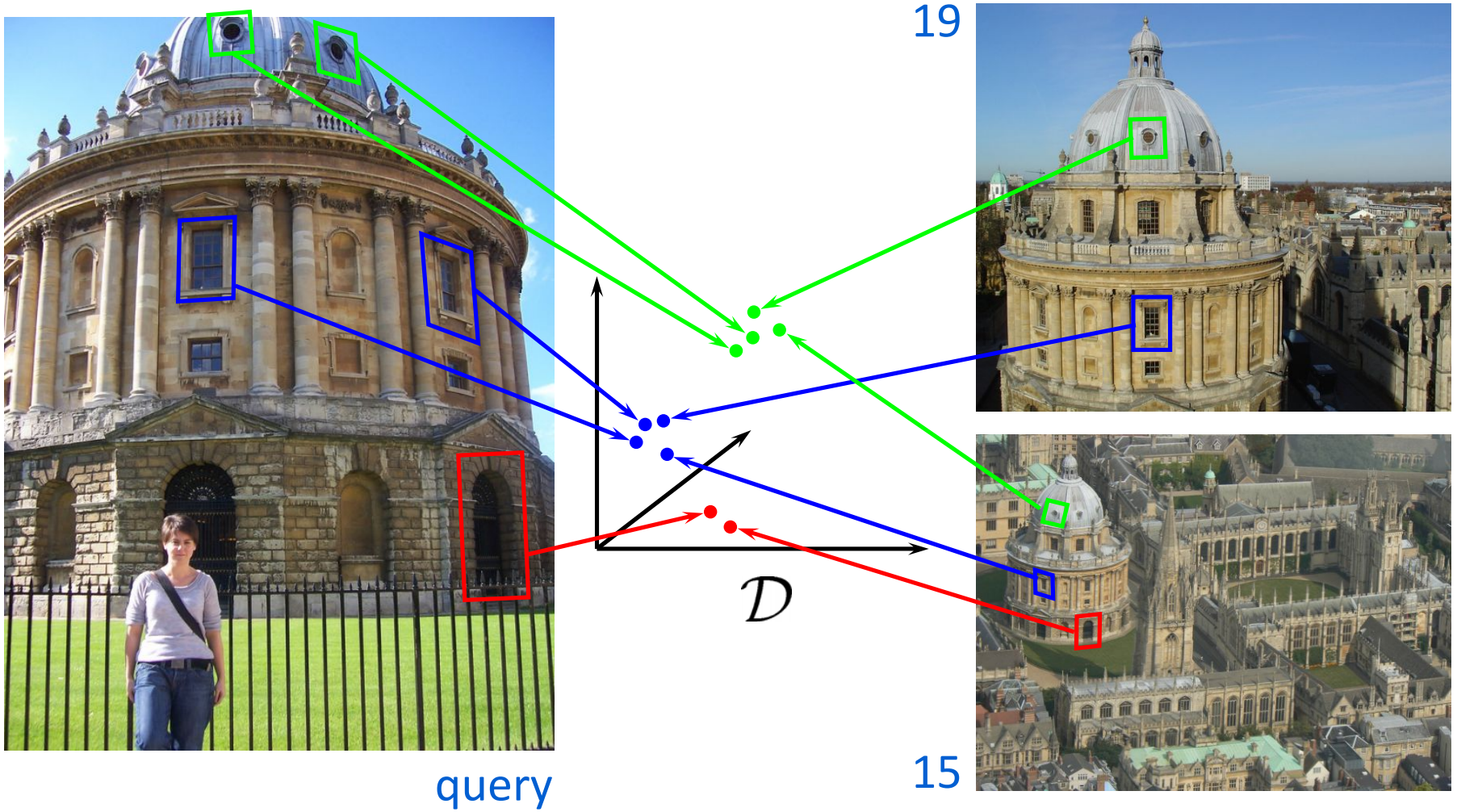
19





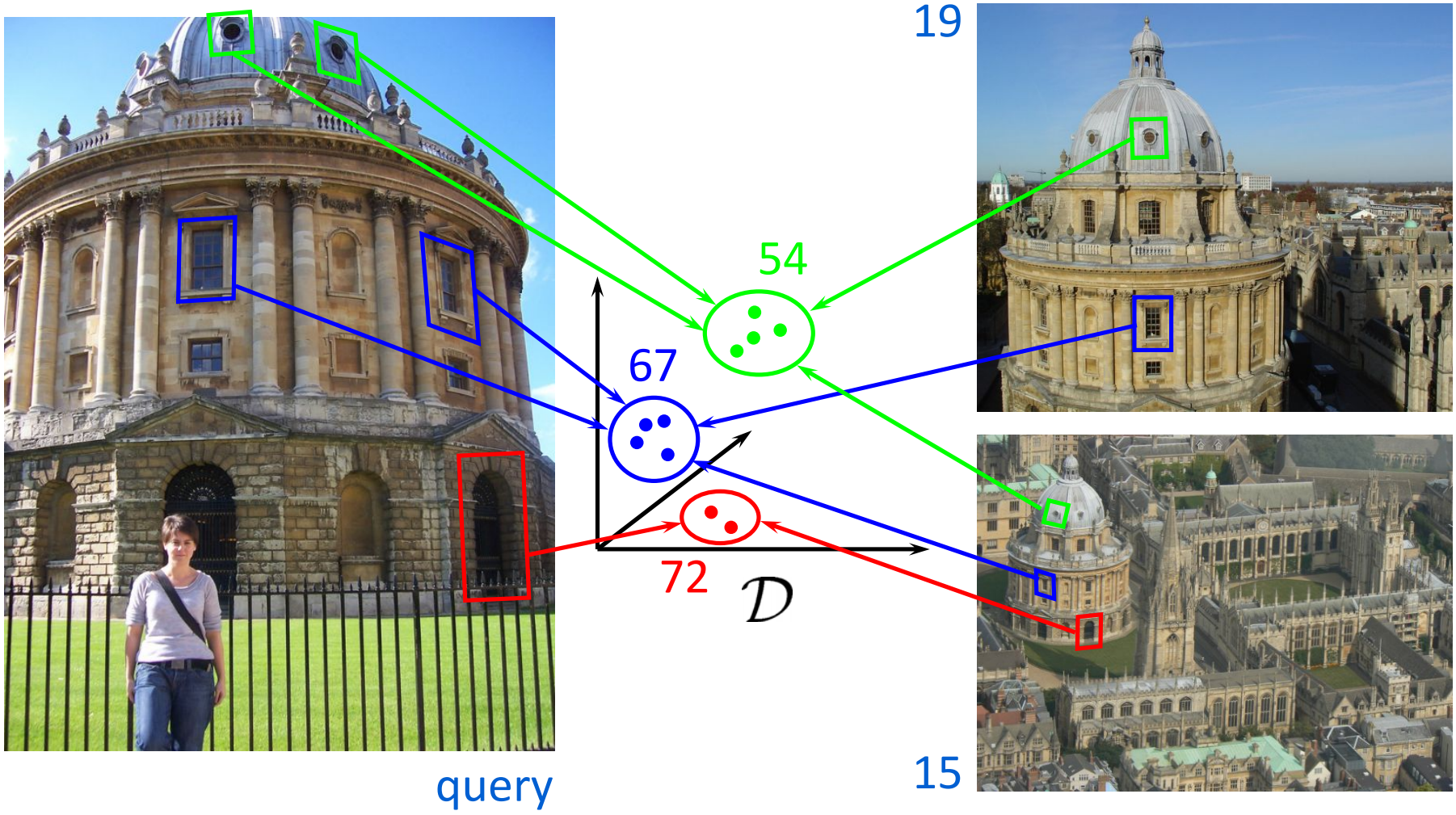
15

descriptor matching



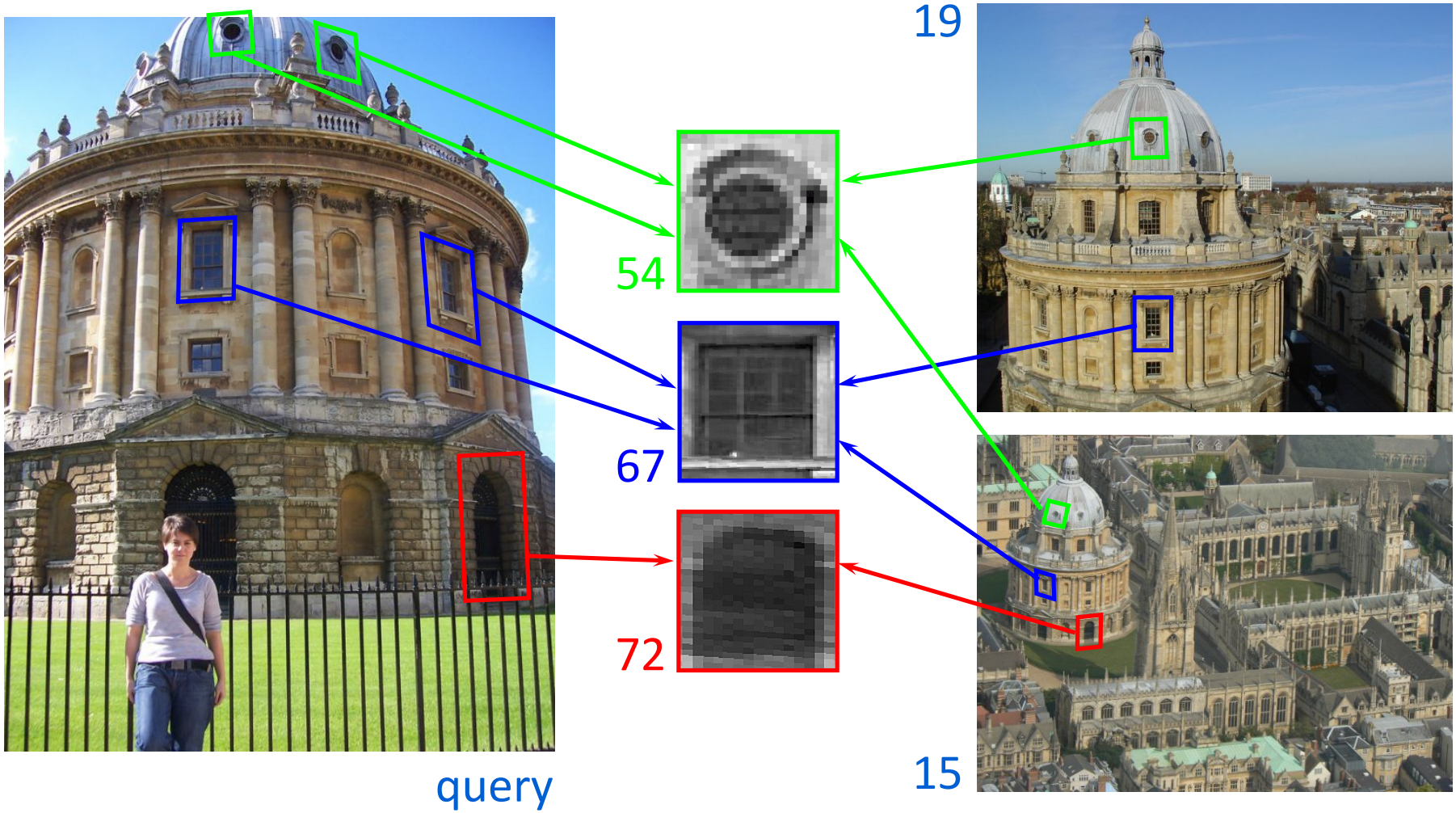
vector quantization $\rightarrow$ visual words

[Sivic and Zisserman, ICCV 2003]



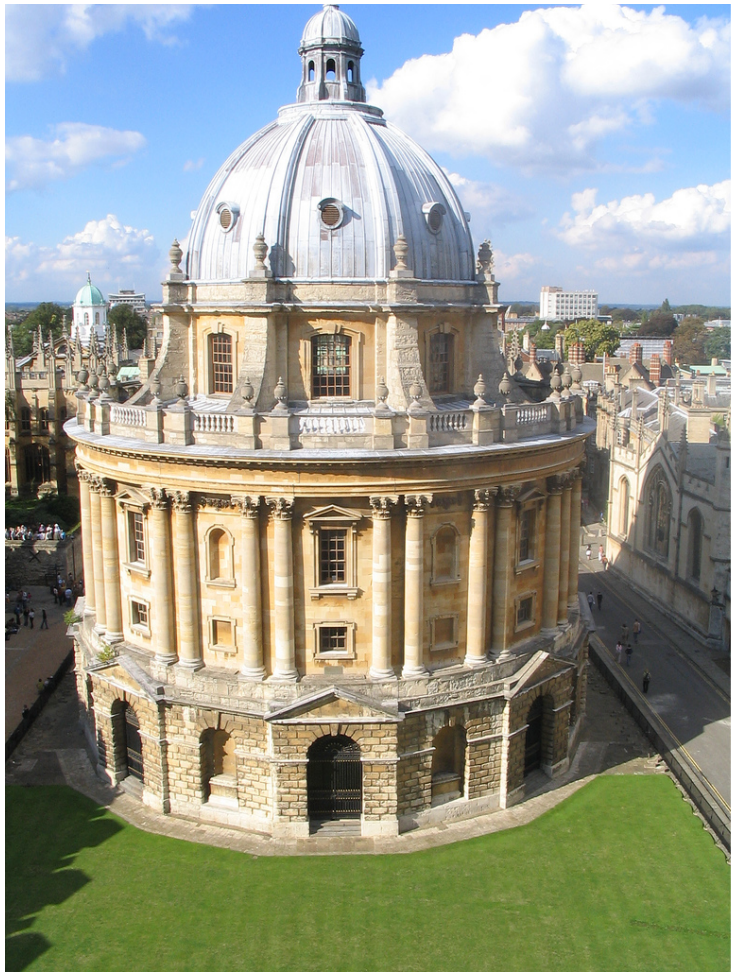
vector quantization $\rightarrow$ visual words

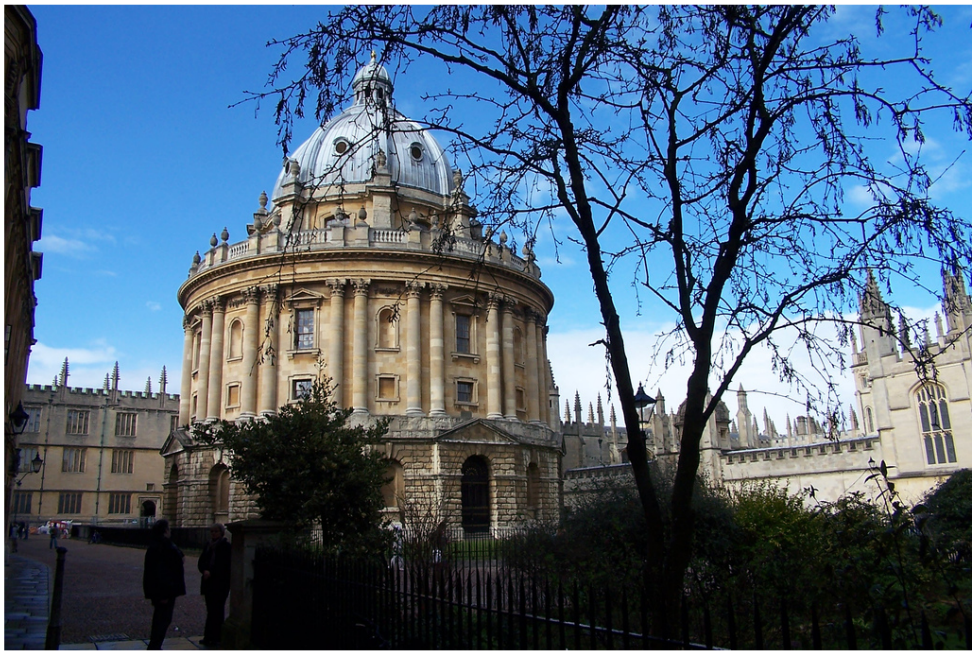
[Sivic and Zisserman, ICCV 2003]



spatial matching

[Philbin et al. CVPR 2007]

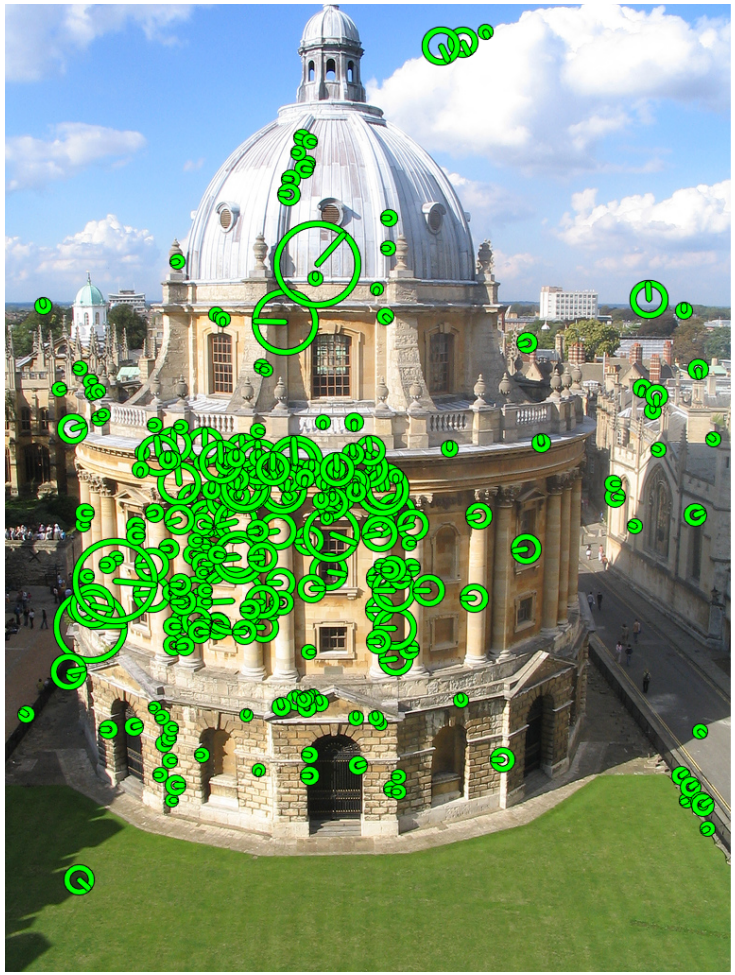


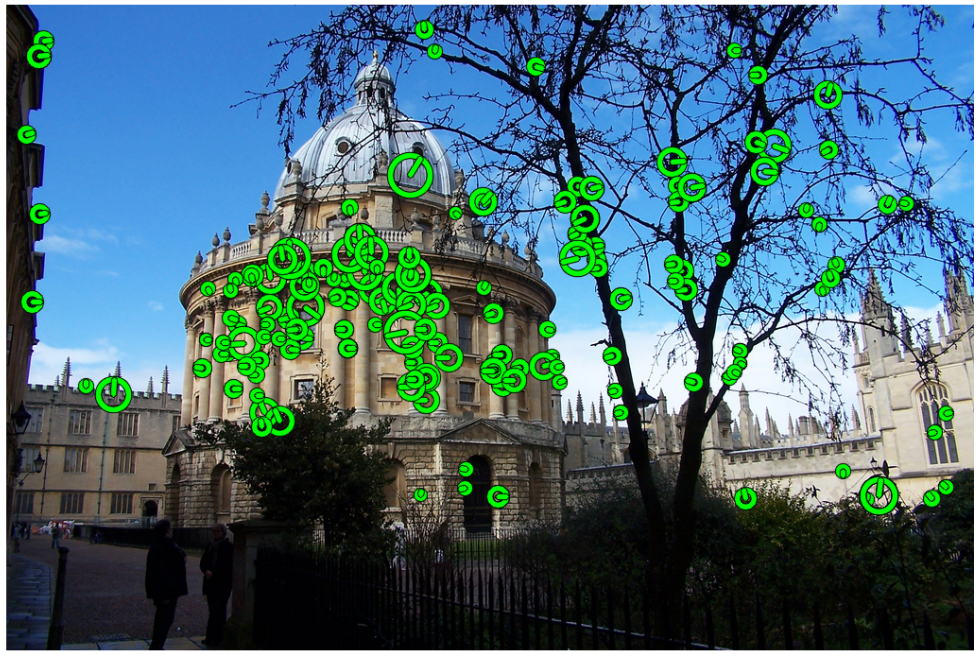


original images

spatial matching

[Philbin et al. CVPR 2007]

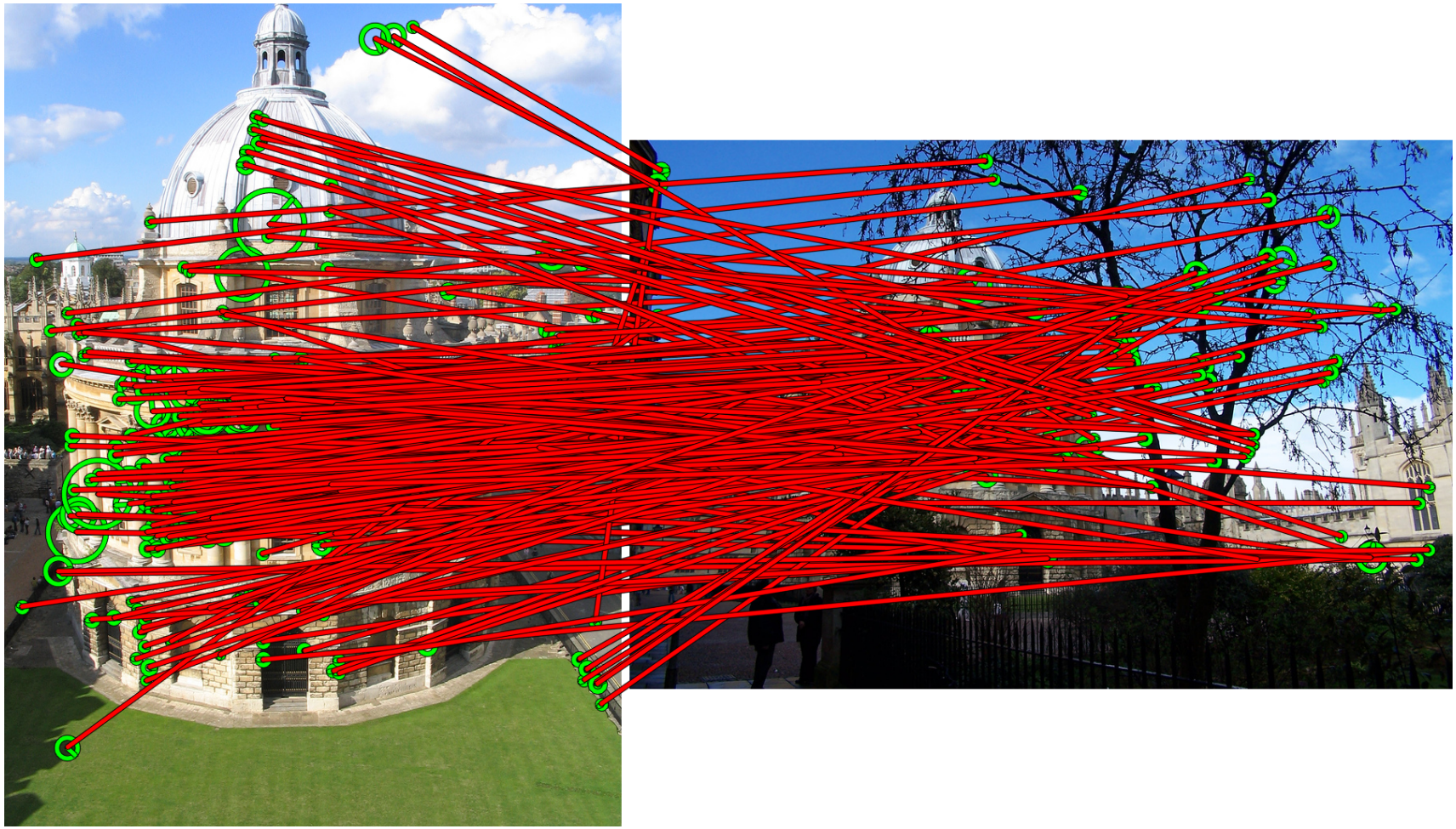




local features

spatial matching

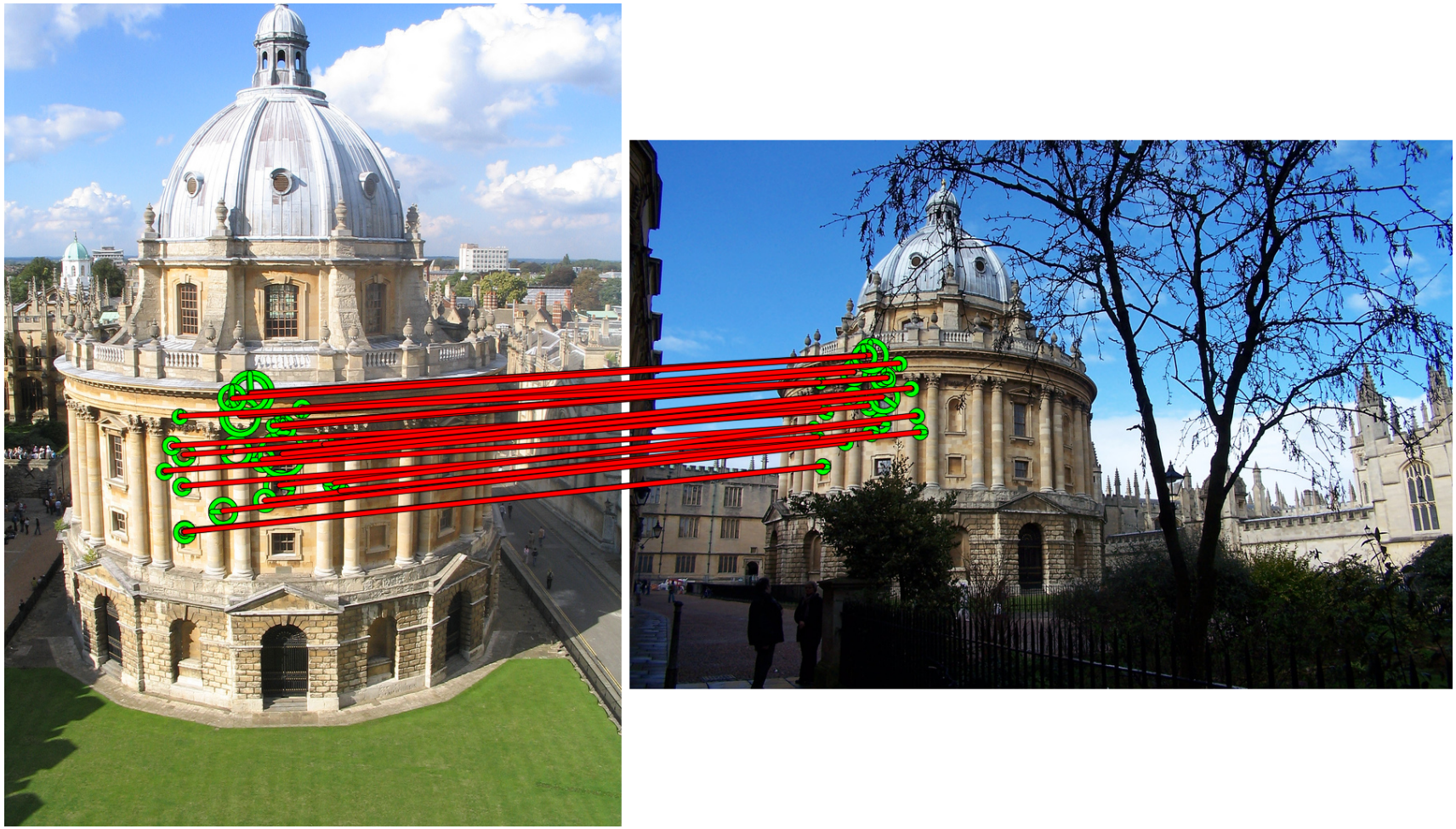
[Philbin et al. CVPR 2007]



tentative correspondences

spatial matching

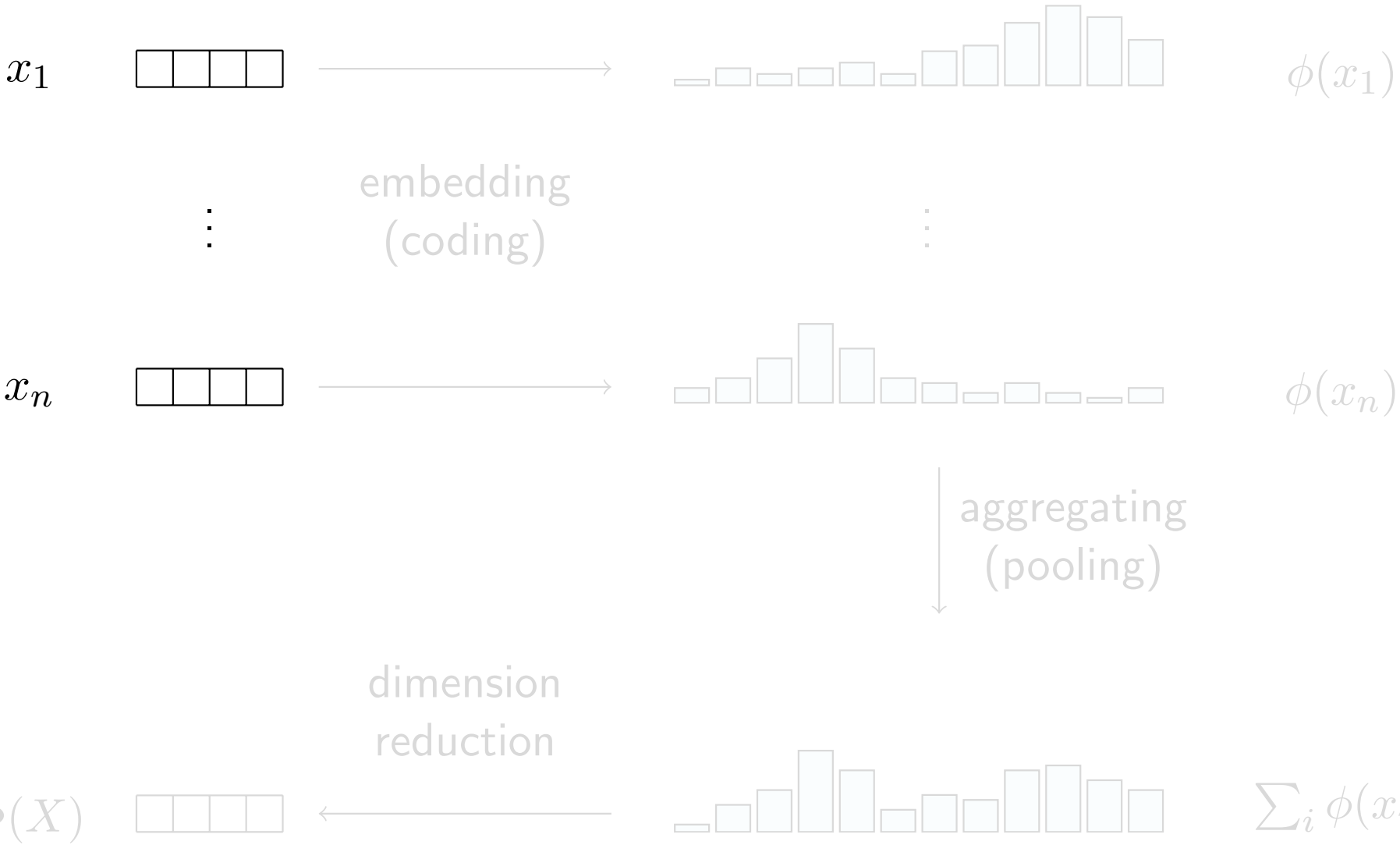
[Philbin et al. CVPR 2007]



inliers

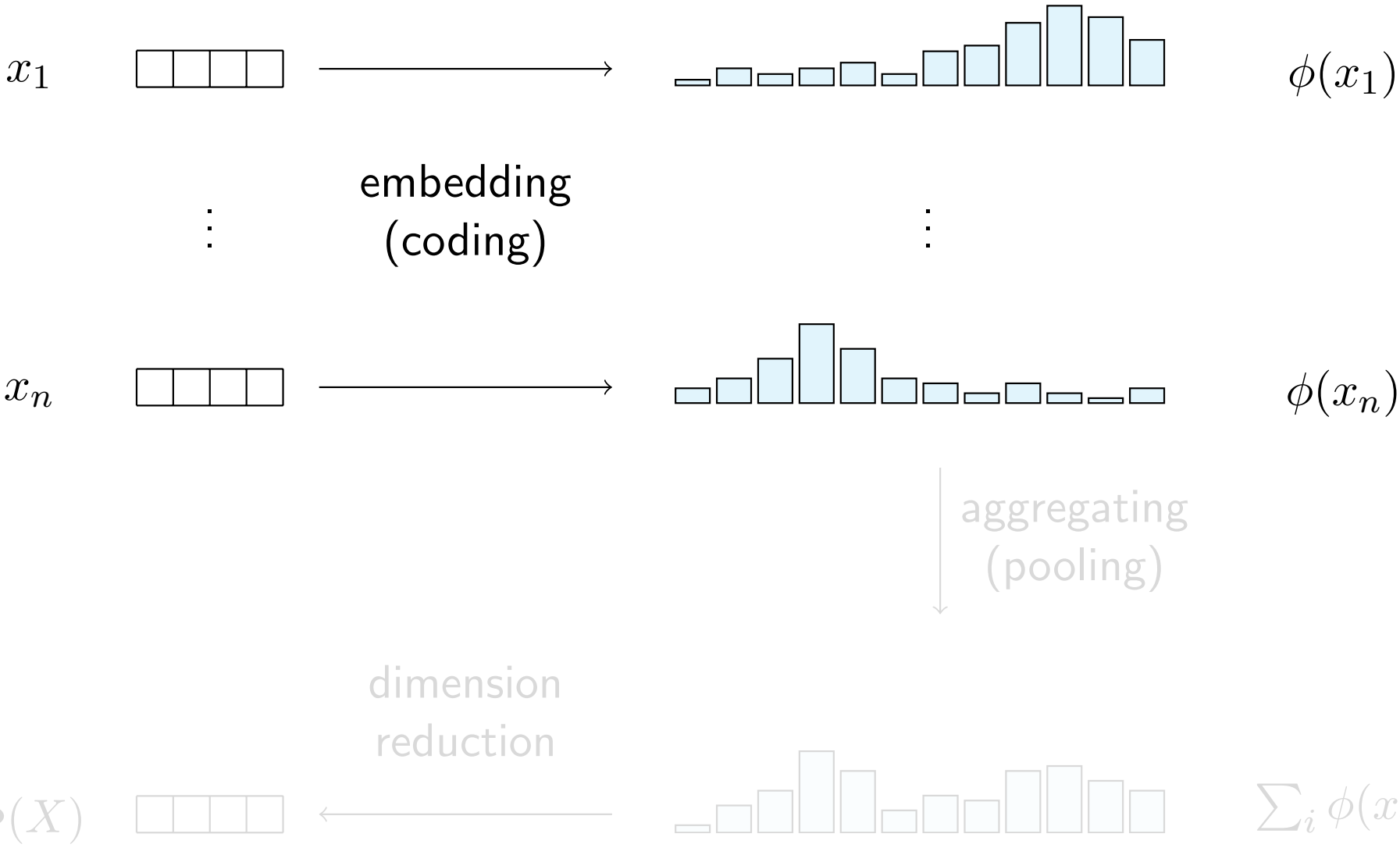
explicit feature maps (VLAD, Fisher)

[Jégou et al. CVPR 2010, Perronnin et al. CVPR 2010]



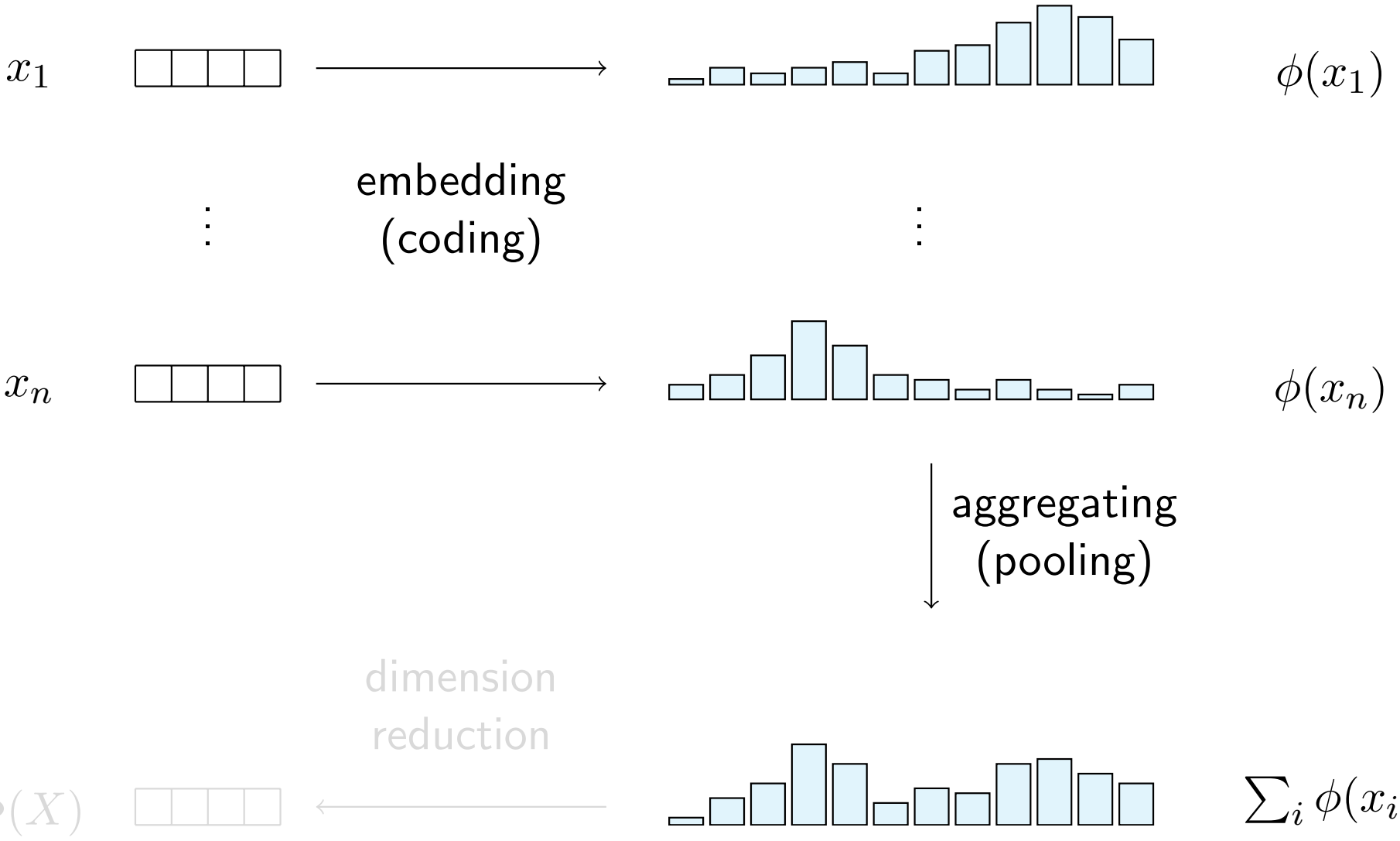
explicit feature maps (VLAD, Fisher)

[Jégou et al. CVPR 2010, Perronnin et al. CVPR 2010]



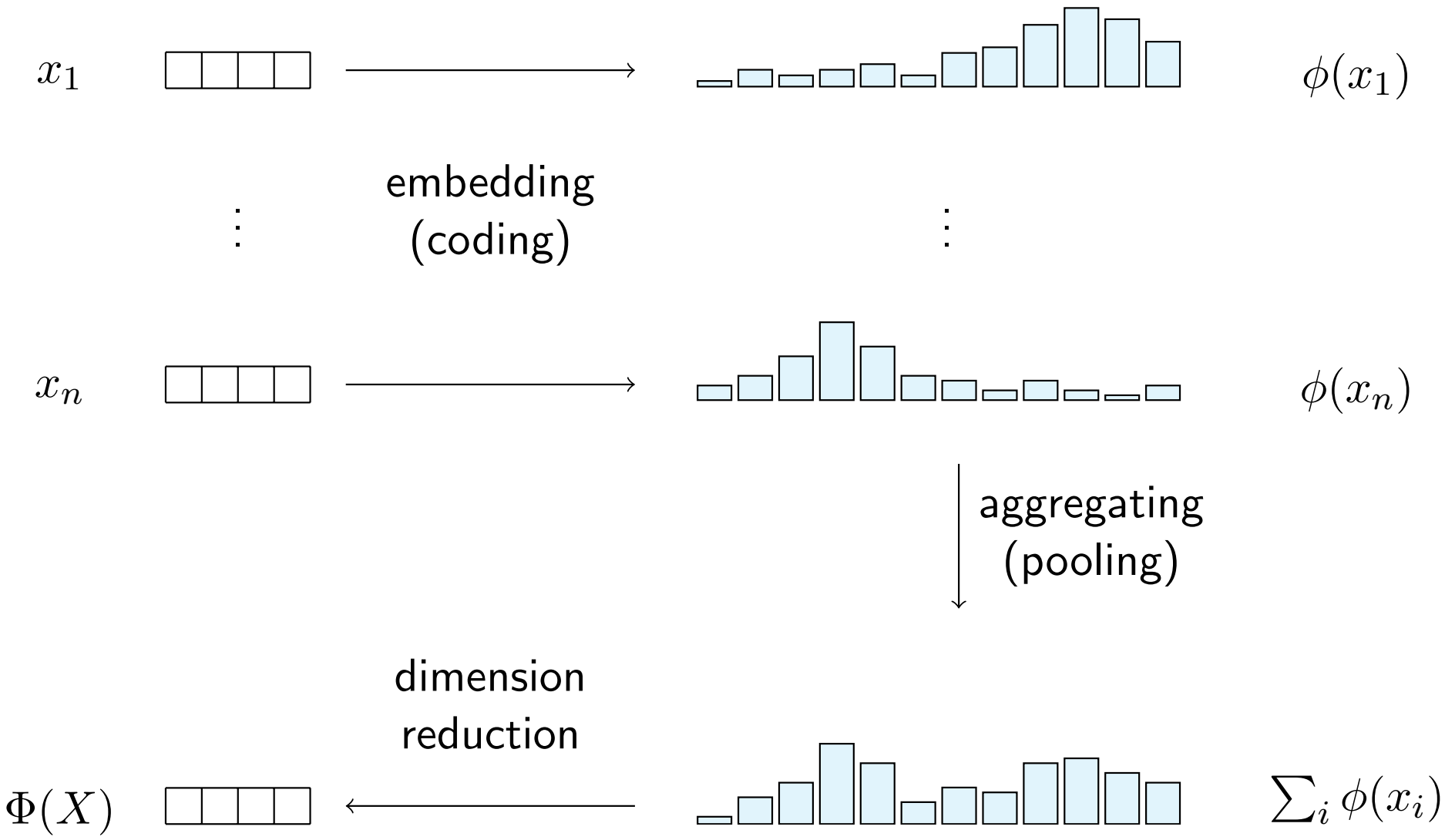
explicit feature maps (VLAD, Fisher)

[Jégou et al. CVPR 2010, Perronnin et al. CVPR 2010]



explicit feature maps (VLAD, Fisher)

[Jégou et al. CVPR 2010, Perronnin et al. CVPR 2010]



to aggregate or not to aggregate?

[Tolias et al. ICCV 2013]

$k = 128$ as in VLAD

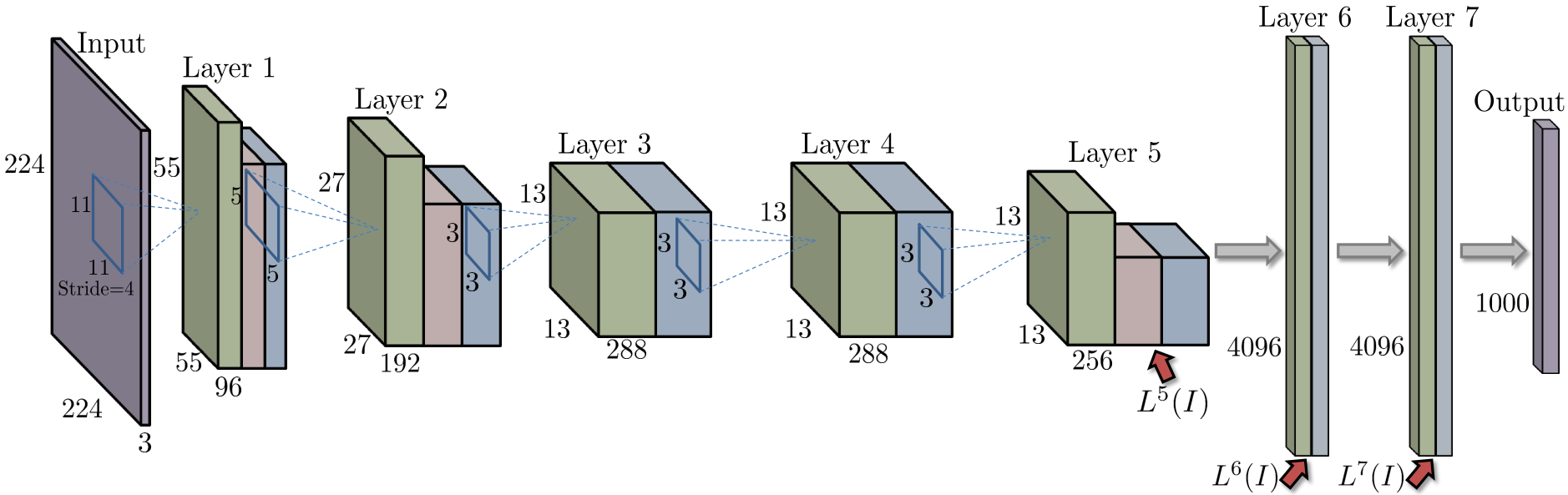
to aggregate or not to aggregate?

[Tolias et al. ICCV 2013]

$k = 65$k as in HE

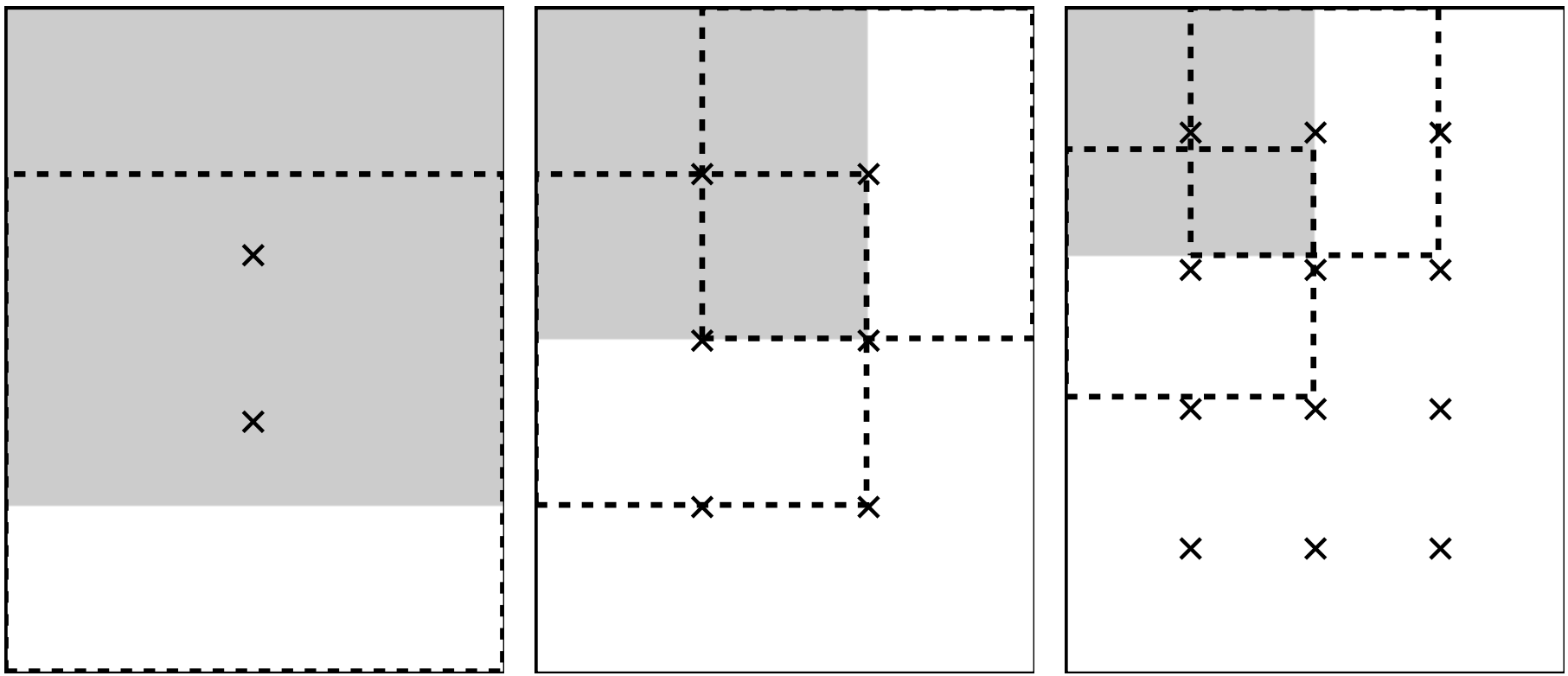
neural codes

[Babenko et al. ECCV 2014]



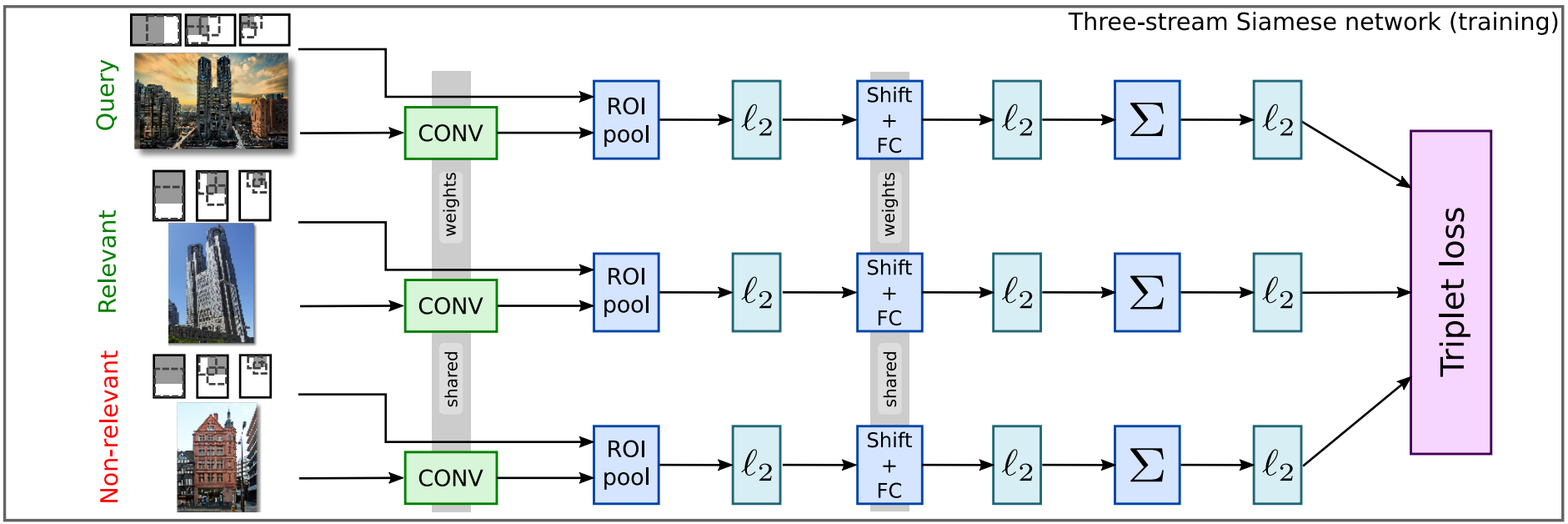
regional descriptors

[Tolias et al. ICLR 2016]



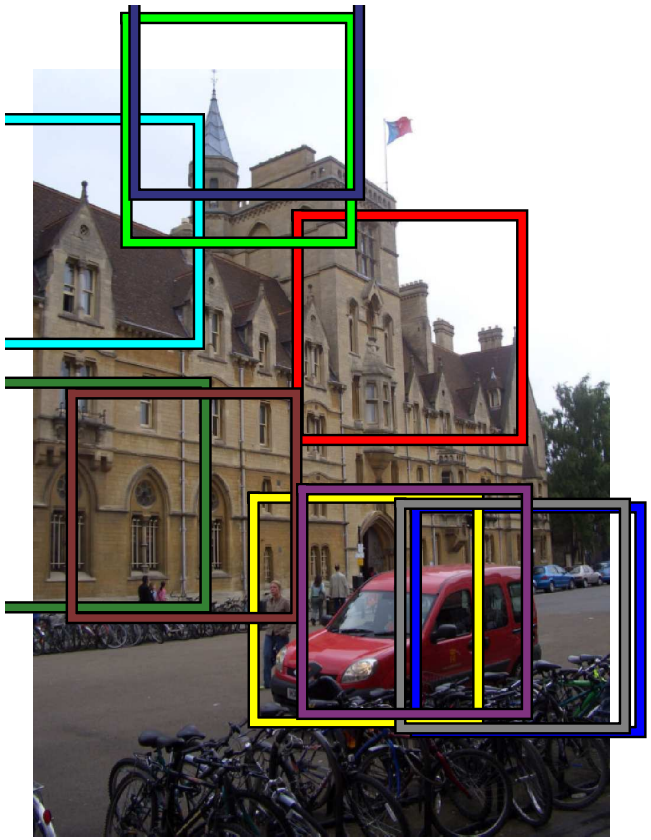
fine tuning: regional descriptors

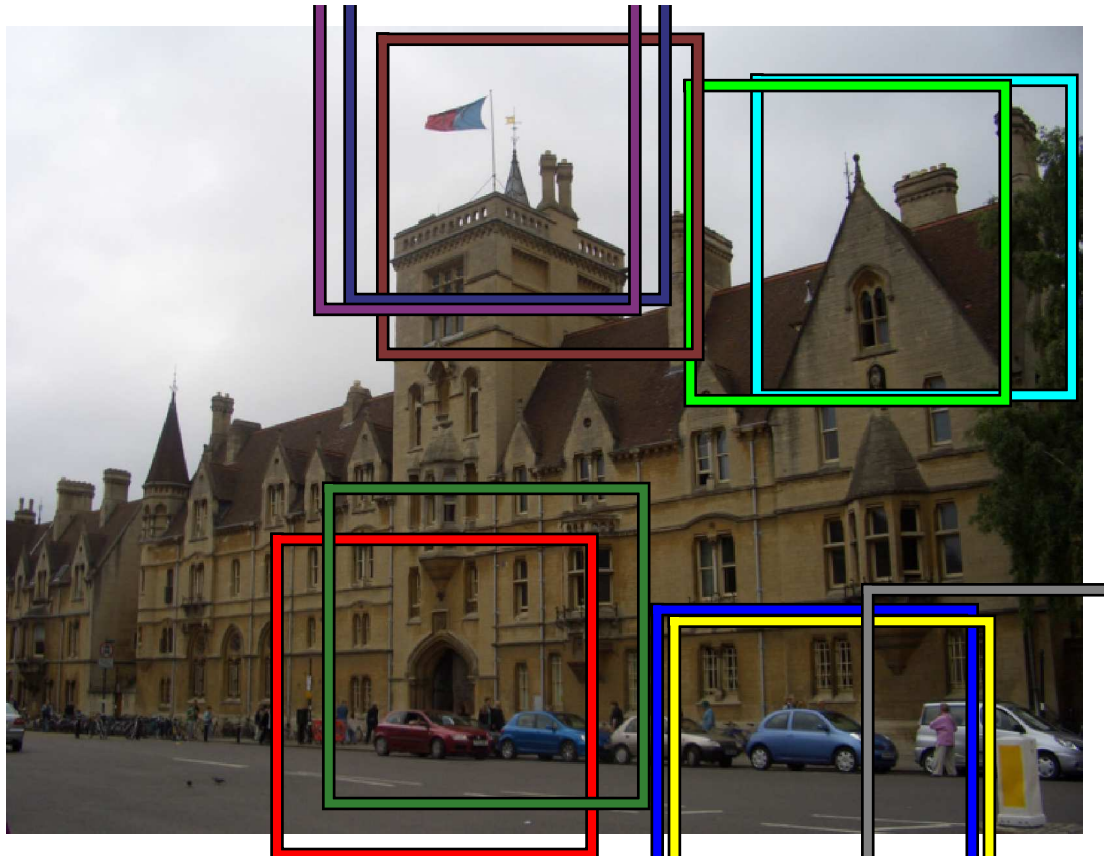
[Gordo et al. ECCV 2016]



fine tuning: global descriptor

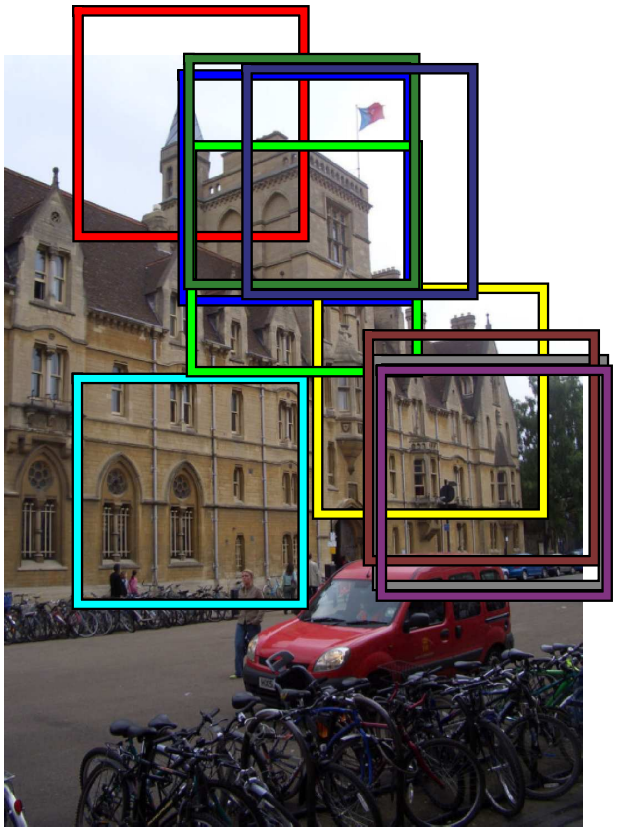
[Radenović et al. ECCV 2016]

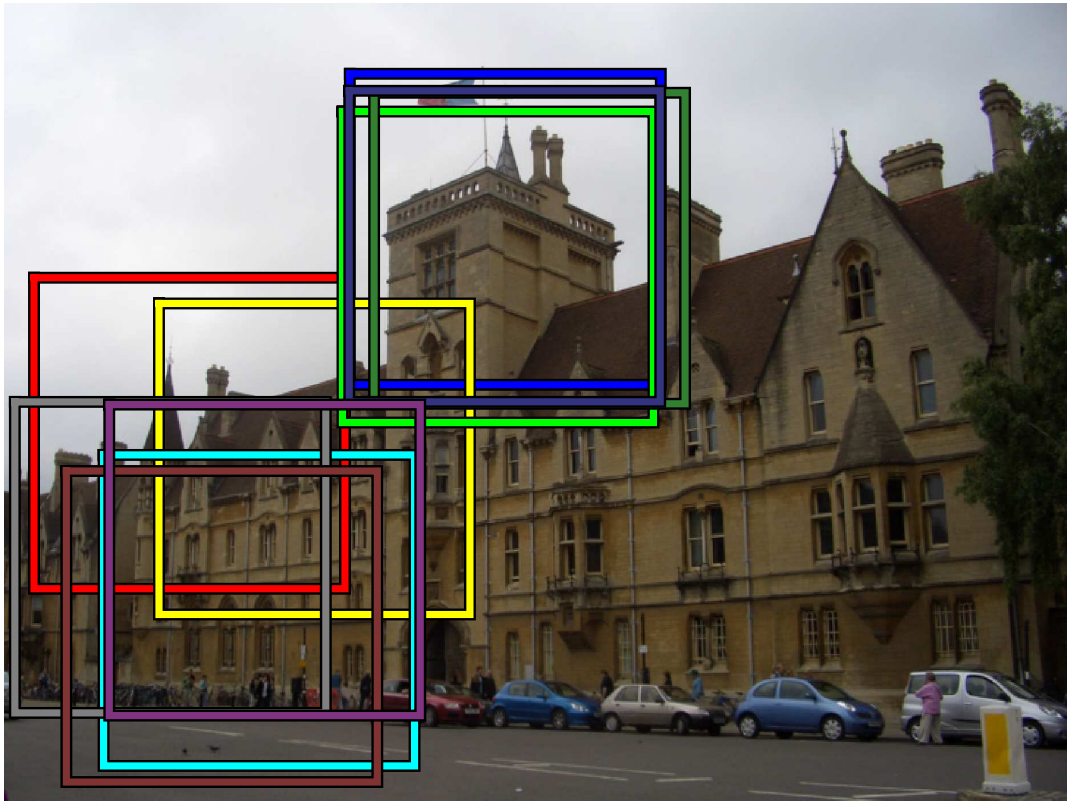




fine tuning: global descriptor

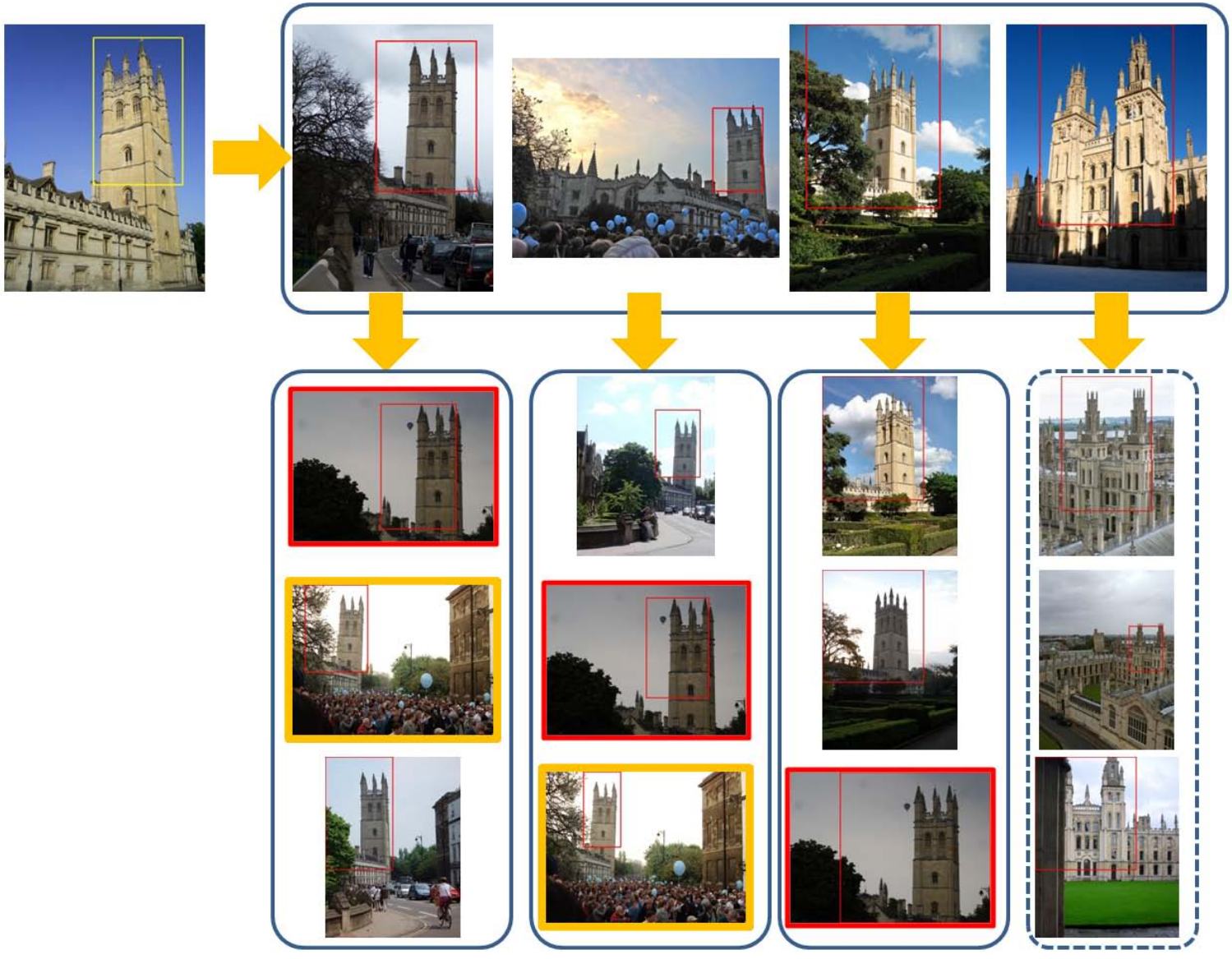
[Radenović et al. ECCV 2016]





query expansion

[Chum et al. ICCV 2007, Shen et al. CVPR 2012]



problem formulation

graph representation

- weighted undirected graph G with n vertices $V = \{v_1, \ldots, v_n\}$ and ℓ edges
- represented by symmetric non-negative $n \times n$ adjacency matrix W
- G has no self-loops: W has zero diagonal
- W is sparse with $2\ell \leq kn$ nonzero elements with $k \ll n$

symmetrically normalized representation

$n \times n$:

- degree matrix $D := \mathrm{diag}(W\mathbf{1})$
- normalized adjacency matrix $\mathcal{W} := D^{-1/2} W D^{-1/2}$
- Laplacian $L := D - W$
- normalized Laplacian $\mathcal{L} := D^{-1/2} L D^{-1/2} = I - \mathcal{W}$

Laplacian properties

- both L and $\mathcal{L}$ are singular and positive-semidefinite
- the eigenvalues of $\mathcal{L}$ are in $[0, 2]$
- each eigenvector $\mathbf{u}$ of L associated to eigenvalue 0 is constant within connected components of G (*e.g.* , $L\mathbf{1} = D\mathbf{1} - W\mathbf{1} = \mathbf{0}$); the corresponding eigenvector of $\mathcal{L}$ is $D^{1/2}\mathbf{u}$
- if $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of $\mathcal{W}$, its spectral radius $\varrho(\mathcal{W}) := \max_i |\lambda_i| = \lambda_1 = 1$

regularized Laplacian

$n \times n$:

- regularized Laplacian $L_\alpha := \beta^{-1}(D - \alpha W)$, where $\beta := 1 - \alpha$
- normalized regularized Laplacian
  $\mathcal{L}_\alpha := D^{-1/2} L_\alpha D^{-1/2} = \beta^{-1}(I - \alpha \mathcal{W})$
- both are positive-definite for $0 \leq \alpha < 1$

ranking on manifolds

[Zhou et al. NIPS 2003]

- $n \times 1$ observation vector $\mathbf{y}$ with $y_i = 1$ if v_i is a query and 0 otherwise
- diffusion or random walk: iterate for $t = 1, 2, \ldots$

$$\mathbf{x}^{(t)} := \alpha \mathcal{W} \mathbf{x}^{(t-1)} + (1 - \alpha)\mathbf{y}$$

- if $0 \leq \alpha < 1$, then as $t \to \infty$, $\mathbf{x}^{(t)}$ tends to $n \times 1$ ranking vector

$$\mathbf{x}^* := \mathcal{L}_\alpha^{-1} \mathbf{y}$$

- now, rank vertices $V = \{v_i\}$ by descending order of x_i

ranking on manifolds

[Zhou et al. NIPS 2003]

- $n \times 1$ observation vector $\mathbf{y}$ with $y_i = 1$ if v_i is a query and 0 otherwise
- diffusion or random walk: iterate for $t = 1, 2, \ldots$

$$\mathbf{x}^{(t)} := \alpha \mathcal{W} \mathbf{x}^{(t-1)} + (1 - \alpha)\mathbf{y}$$

- if $0 \leq \alpha < 1$, then as $t \to \infty$, $\mathbf{x}^{(t)}$ tends to $n \times 1$ ranking vector

$$\mathbf{x}^* := \mathcal{L}_\alpha^{-1} \mathbf{y}$$

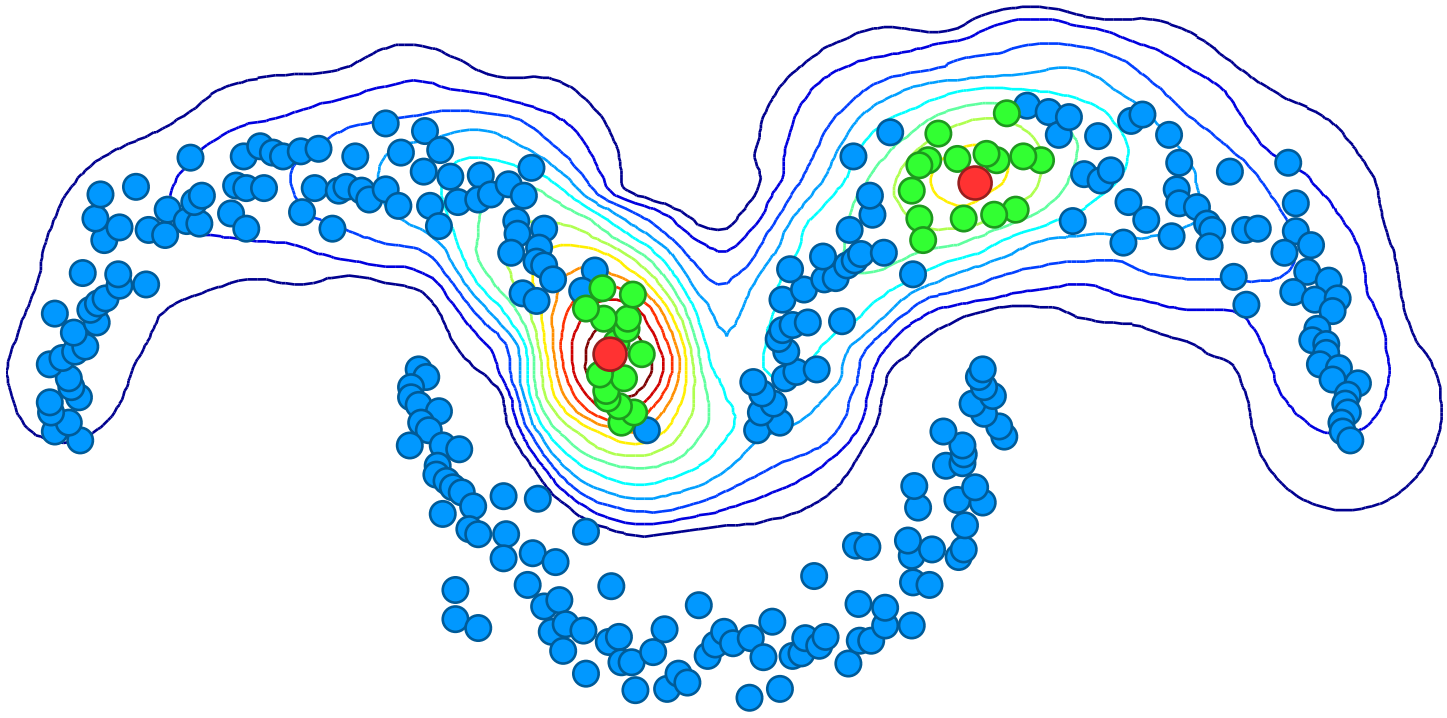
- now, rank vertices $V = \{v_i\}$ by descending order of x_i

image retrieval

- given N images, each represented by m region descriptors in $\mathbb{R}^d$
- dataset represented by $n := Nm$ descriptors $\mathcal{V} := \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$
- similarity $s(\mathbf{v}, \mathbf{z}) := (\mathbf{v}^\top \mathbf{z})_+^\gamma$ for $\mathbf{v}, \mathbf{z} \in \mathbb{R}^d$, with $\gamma > 0$
- k-NN similarity $s(\mathbf{v}_i|\mathbf{z}) := s(\mathbf{v}_i, \mathbf{z})$ if $\mathbf{v}_i$ is a k-NN of $\mathbf{z}$ in $\mathcal{V}$ and zero otherwise
- mutual neighbors: $W := \min(S, S^\top)$ where $s_{ij} := s(\mathbf{v}_i|\mathbf{v}_j)$

challenges

- how to handle unseen queries without recomputing W?
- how to rank images given region ranking scores?
- how to compute the ranking vector efficiently?
- how scale up beyond a few thousand images?



diffusion on region manifolds

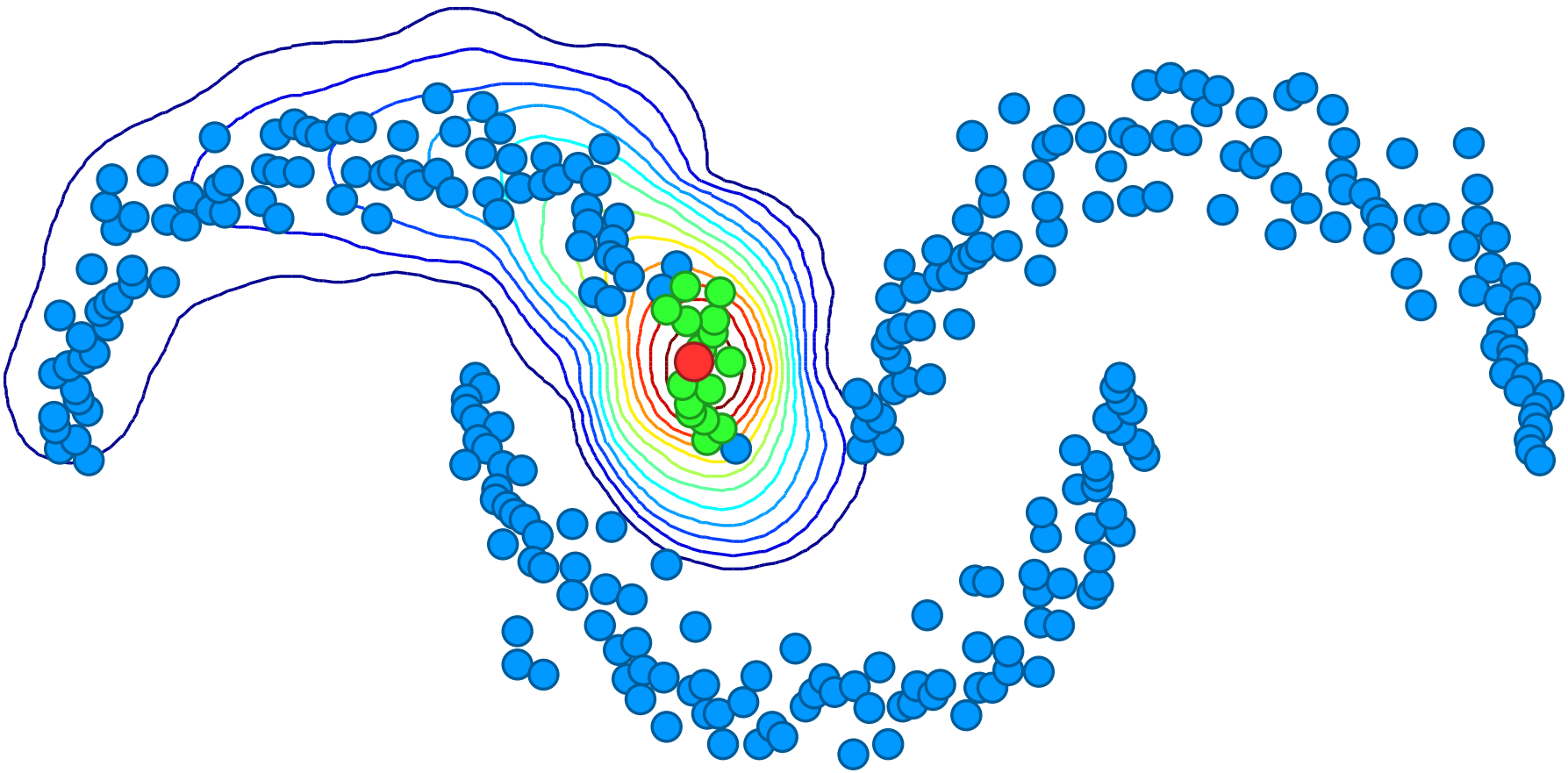
handling unseen queries

- keep W fixed, computed on dataset without queries
- given query image represented by $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\} \subset \mathbb{R}^d$, form the *observation vector* by pooling over regions

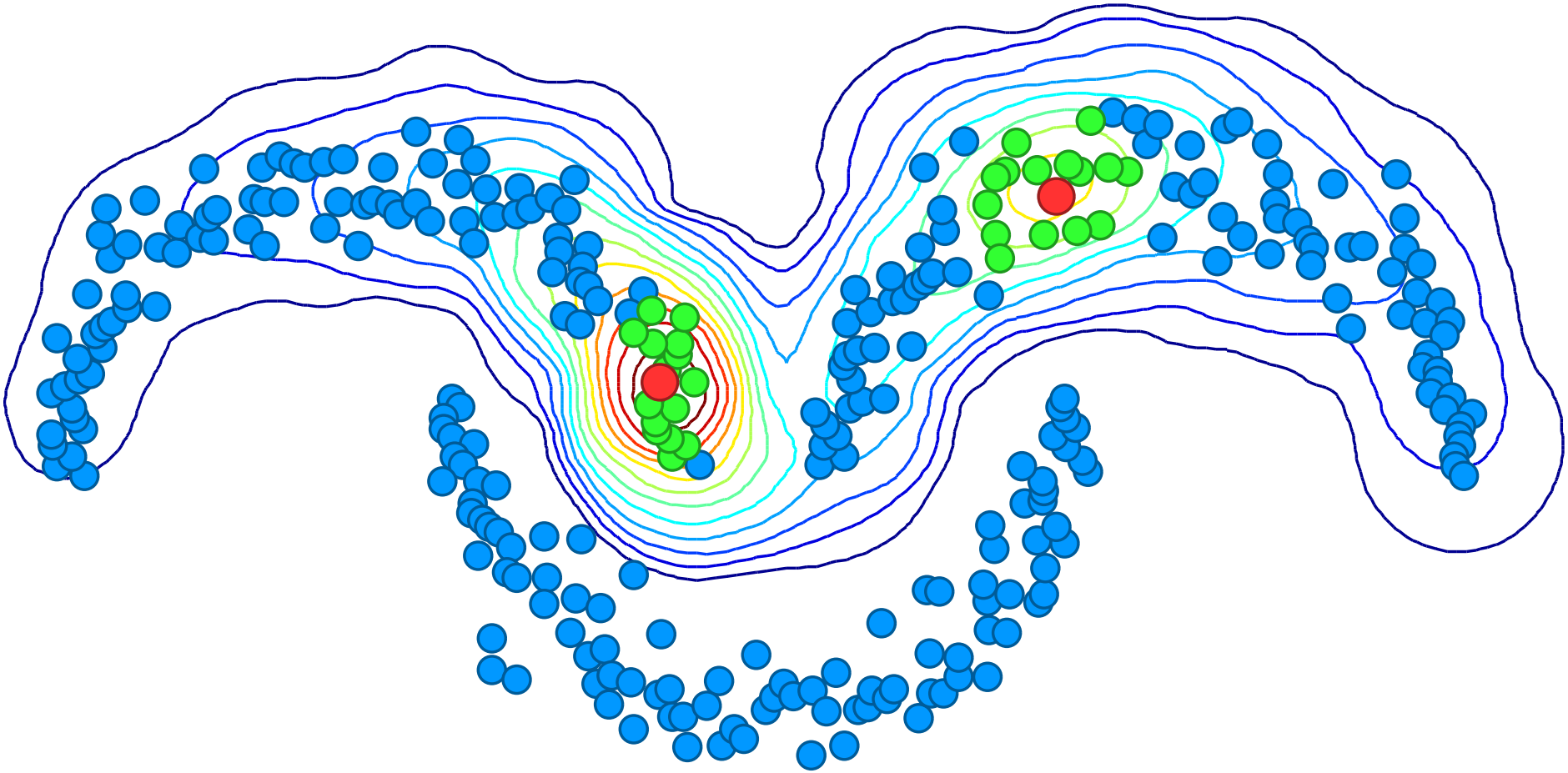
$$y_i := \sum_{j=1}^{m} s(\mathbf{v}_i | \mathbf{q}_j)$$

- make $\mathbf{y}$ sparse by keeping only the k largest entries
- now, computing the ranking vector is constant in m

one query vector



two query vectors



ranking images

- given region ranking scores $\mathbf{x}^*$ and dataset image represented by $\{\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_m}\}$, score image by

$$\sum_{j=1}^{m} w_j x^\star_{i_j}$$

- (uniform) sum pooling: $\mathbf{w} := \mathbf{1}_m$
- assuming $m < d$, generalized max pooling [Murray and Perronnin CVPR 2014, Iscen *et al.* 2014]:

$$\mathbf{w} := (\Phi^\top \Phi + \lambda I_m)^{-1} \mathbf{1}_m, \tag{1}$$

where $\Phi := (\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_m})$ and $\lambda > 0$

ranking images

- given region ranking scores $\mathbf{x}^*$ and dataset image represented by $\{\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_m}\}$, score image by

$$\sum_{j=1}^{m} w_j x^\star_{i_j}$$

- (uniform) sum pooling: $\mathbf{w} := \mathbf{1}_m$
- assuming $m < d$, generalized max pooling [Murray and Perronnin CVPR 2014, Iscen *et al.* 2014]:

$$\mathbf{w} := (\Phi^\top \Phi + \lambda I_m)^{-1} \mathbf{1}_m, \tag{1}$$

where $\Phi := (\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_m})$ and $\lambda > 0$

diffusion is an iterative solver

- given linear system

$$A\mathbf{x} = \mathbf{b},$$

Jacobi solver decomposes A as $\Delta + R$ where $\Delta := \mathrm{diag}(A)$ and iterates for $t = 1, 2, \ldots$

$$\mathbf{x}^{(t)} := \Delta^{-1}(\mathbf{b} - R\mathbf{x}^{(t-1)})$$

- given $\mathcal{L}_\alpha = \beta^{-1}(I - \alpha\mathcal{W})$, our system is

$$\mathcal{L}_\alpha \mathbf{x} = \mathbf{y}$$

- hence, $\mathbf{b} = (1-\alpha)\mathbf{y}$, $\Delta = I$, $R = -\alpha\mathcal{W}$ and

$$\mathbf{x}^{(t)} := \alpha\mathcal{W}\mathbf{x}^{(t-1)} + (1-\alpha)\mathbf{y}$$

diffusion is an iterative solver

- given linear system

$$A\mathbf{x} = \mathbf{b},$$

Jacobi solver decomposes A as $\Delta + R$ where $\Delta := \mathrm{diag}(A)$ and iterates for $t = 1, 2, \ldots$

$$\mathbf{x}^{(t)} := \Delta^{-1}(\mathbf{b} - R\mathbf{x}^{(t-1)})$$

- given $\mathcal{L}_\alpha = \beta^{-1}(I - \alpha\mathcal{W})$, our system is

$$\mathcal{L}_\alpha \mathbf{x} = \mathbf{y}$$

- hence, $\mathbf{b} = (1-\alpha)\mathbf{y}$, $\Delta = I$, $R = -\alpha\mathcal{W}$ and

$$\mathbf{x}^{(t)} := \alpha\mathcal{W}\mathbf{x}^{(t-1)} + (1-\alpha)\mathbf{y}$$

normalization is preconditioning

- (symmetric) preconditioning: solve a related system with A replaced by $C^{-1}AC^{-\top}$
- we could consider matrix L_α and solve instead

$$L_\alpha(D^{-1/2}\mathbf{x}) = D^{1/2}\mathbf{y}$$

- by normalizing L_α into $\mathcal{L}_\alpha$, we are actually performing preconditioning with $C = \mathrm{diag}(L_\alpha)^{1/2}$: diagonal scaling or Jacobi

normalization is preconditioning

- (symmetric) preconditioning: solve a related system with A replaced by $C^{-1}AC^{-\top}$
- we could consider matrix L_α and solve instead

$$L_\alpha(D^{-1/2}\mathbf{x}) = D^{1/2}\mathbf{y}$$

- by normalizing L_α into $\mathcal{L}_\alpha$, we are actually performing preconditioning with $C = \mathrm{diag}(L_\alpha)^{1/2}$: diagonal scaling or Jacobi

efficient solution

- use conjugate gradient (CG) method to solve linear system

$$\mathcal{L}_\alpha \mathbf{x} = \mathbf{y}$$

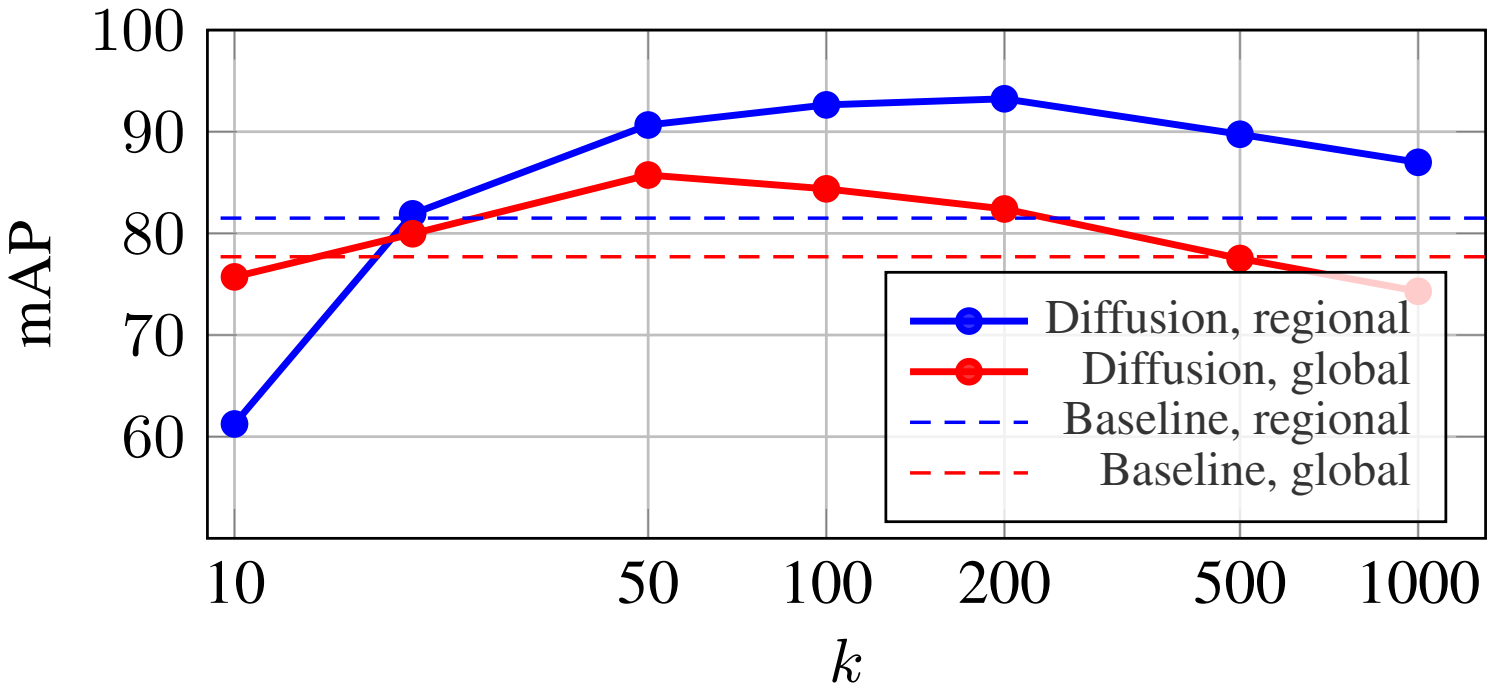
- implicitly, we are minimizing quadratic function

$$f_\alpha(\mathbf{x}) := \frac{1}{2}\mathbf{x}^\top \mathcal{L}_\alpha \mathbf{x} - \mathbf{x}^\top \mathbf{y}$$

experiments

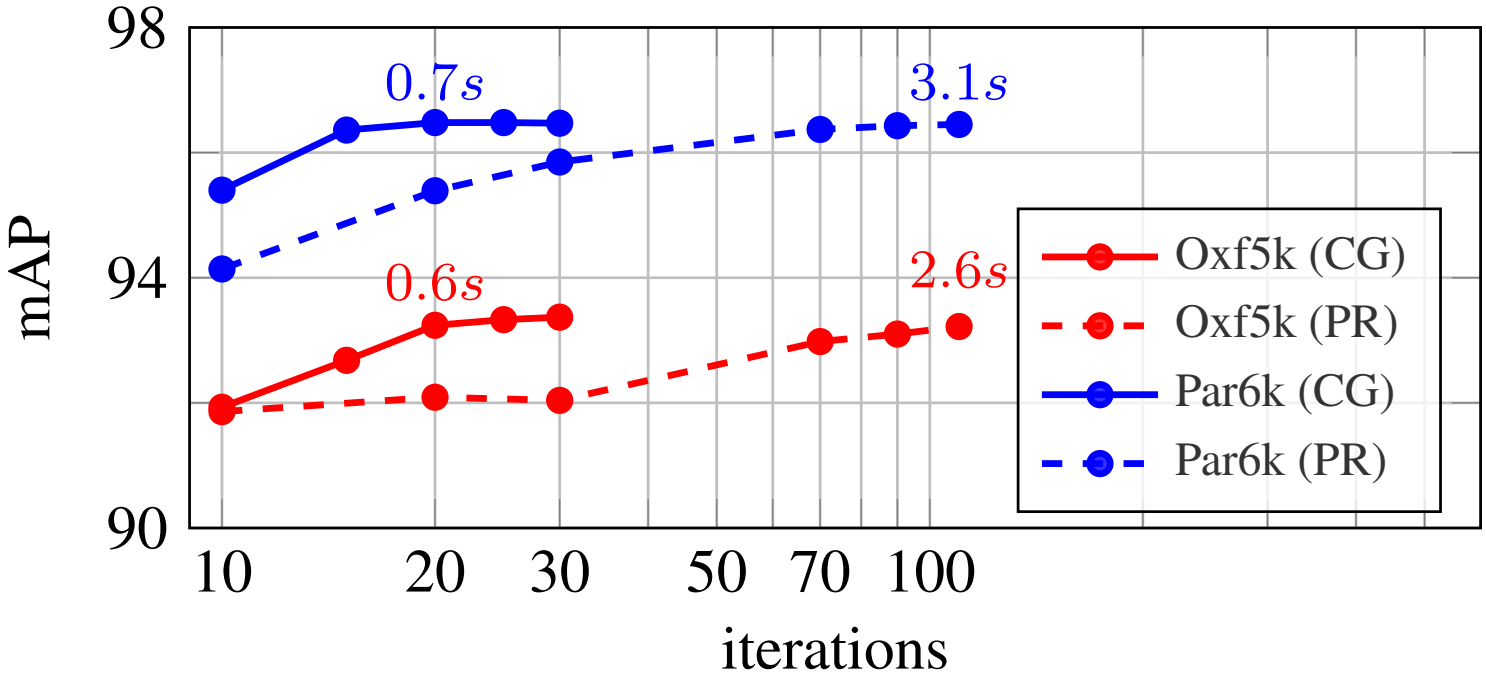
- datasets: Oxford5k, Paris6k, Oxford105k, Paris 106k, INSTRE (27k images, 250 classes)
- networks: VGG ($d = 512$) [Radenović *et al.* ECCV 2016], ResNet101 ($d = 2048$) [Gordo *et al.* ECCV 2016]
- region descriptors: 3 scales (21 regions/image) as in R-MAC [Tolias *et al.* ICLR 2016]
- supervised whitening [Radenović *et al.* ECCV 2016]
- parameters: $\gamma = 3$, $\alpha = 0.99$, $k = 50$ (global), $k = 200$ (regional)

dependence on neighbors, k (Oxford5k)

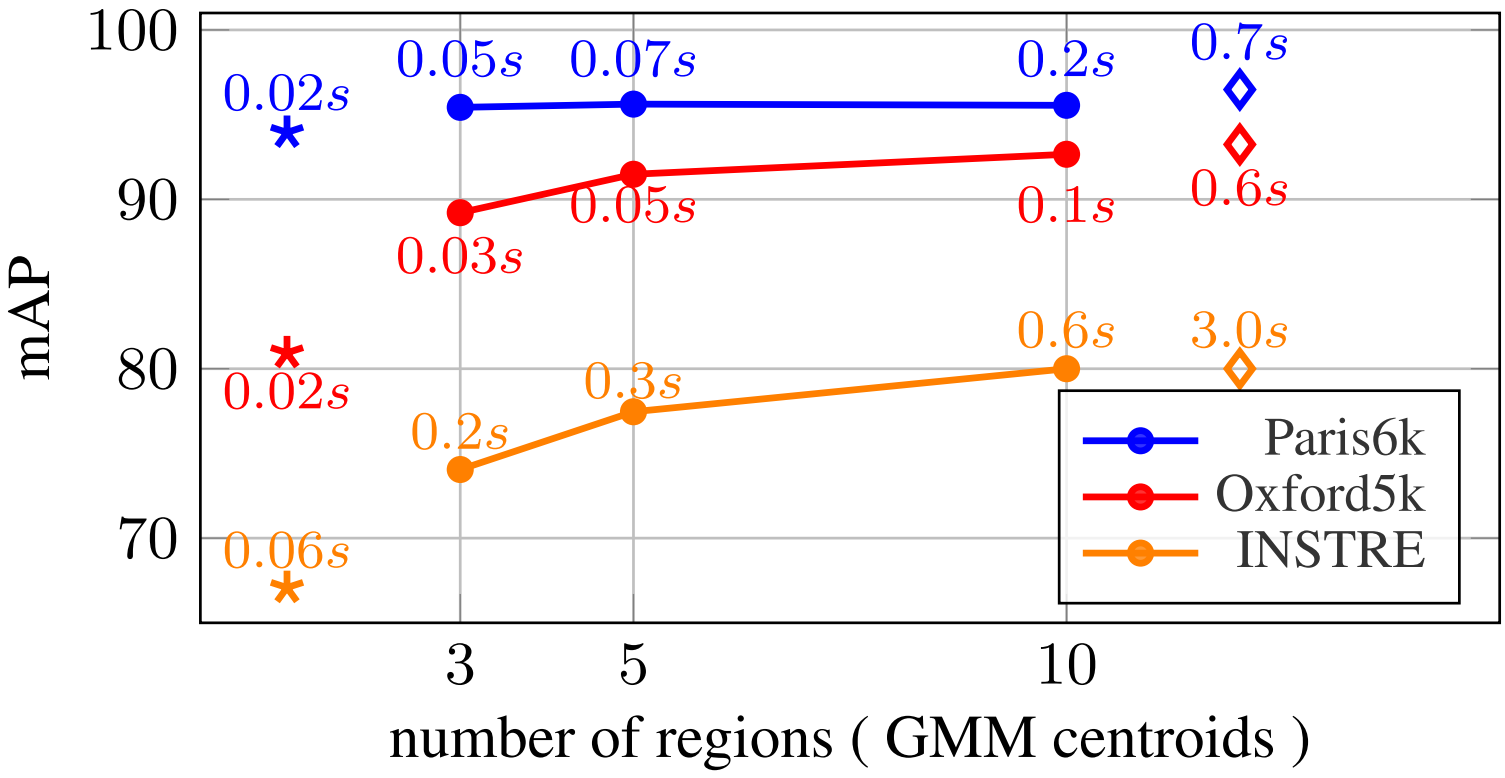


"small patterns appear more frequently than entire images"

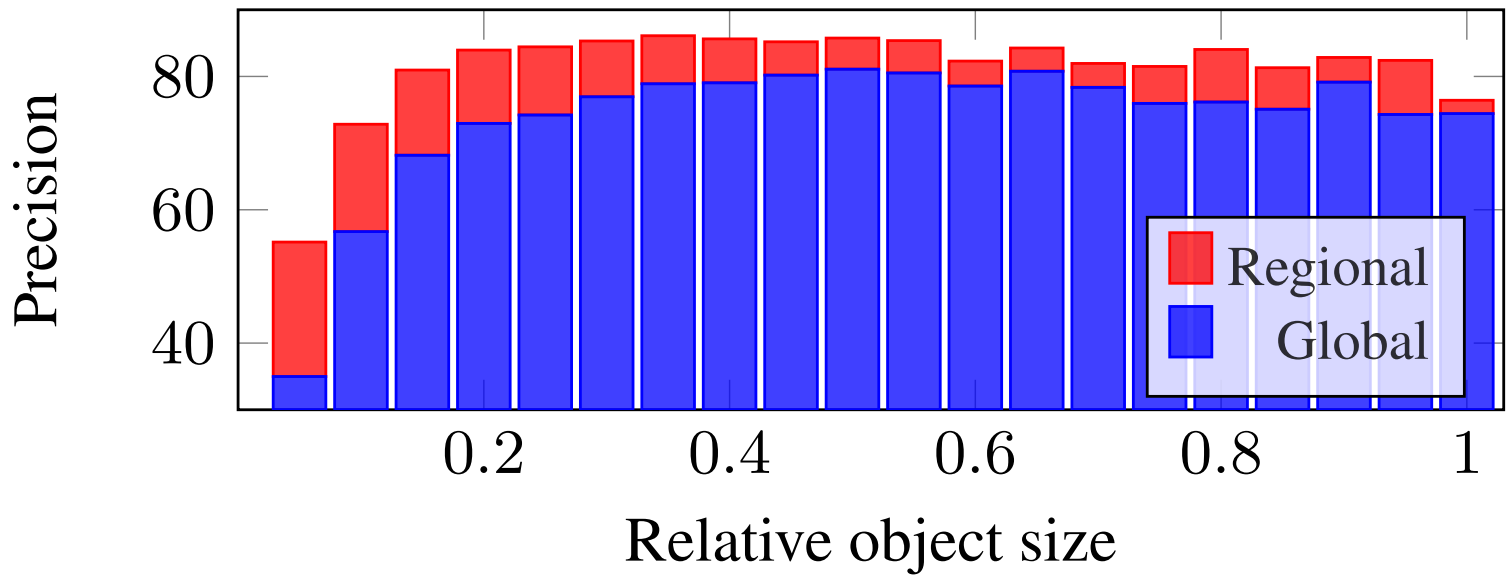
efficient (regional) diffusion with CG



global → regional



small objects (INSTRE)



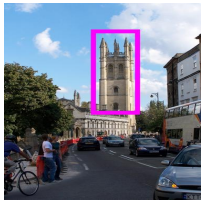
examples







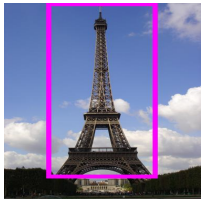
(AP: 43.1→84.9) 0.5→100 0.6→100 1.8→100 0.6→98.7 2.6→100 2.6→100

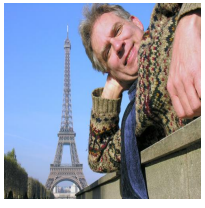
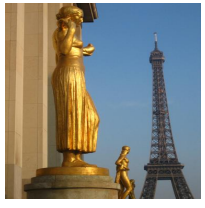
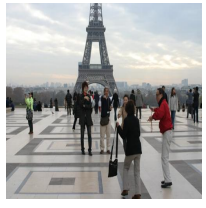
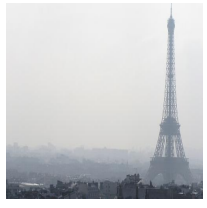
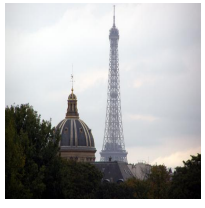






(AP: 24.0→89.9) 3.1→100 4.2→100 4.3→100 4.7→100 5.9→100 5.9→100



(AP: 56.5→94.3) 8.2→94.5 12.9→91.5 18.8→91.6 14.7→85.4 13.3→83.6 15.9→86.1

scaling up

- compact representation: reduce regions/image (21 $\rightarrow$ 5) by a Gaussian mixture model (GMM)
- approximate k-NN graph construction [Dong *et al.* WWW 2011]: 96 hours $\rightarrow$ 45 minutes on Oxford105k, mAP loss less than 1%
- truncate affinity matrix to 10k images for Oxford105k and Paris 106k: 14 $\rightarrow$ 1 second for re-ranking, constant in n and d

state of the art (global)

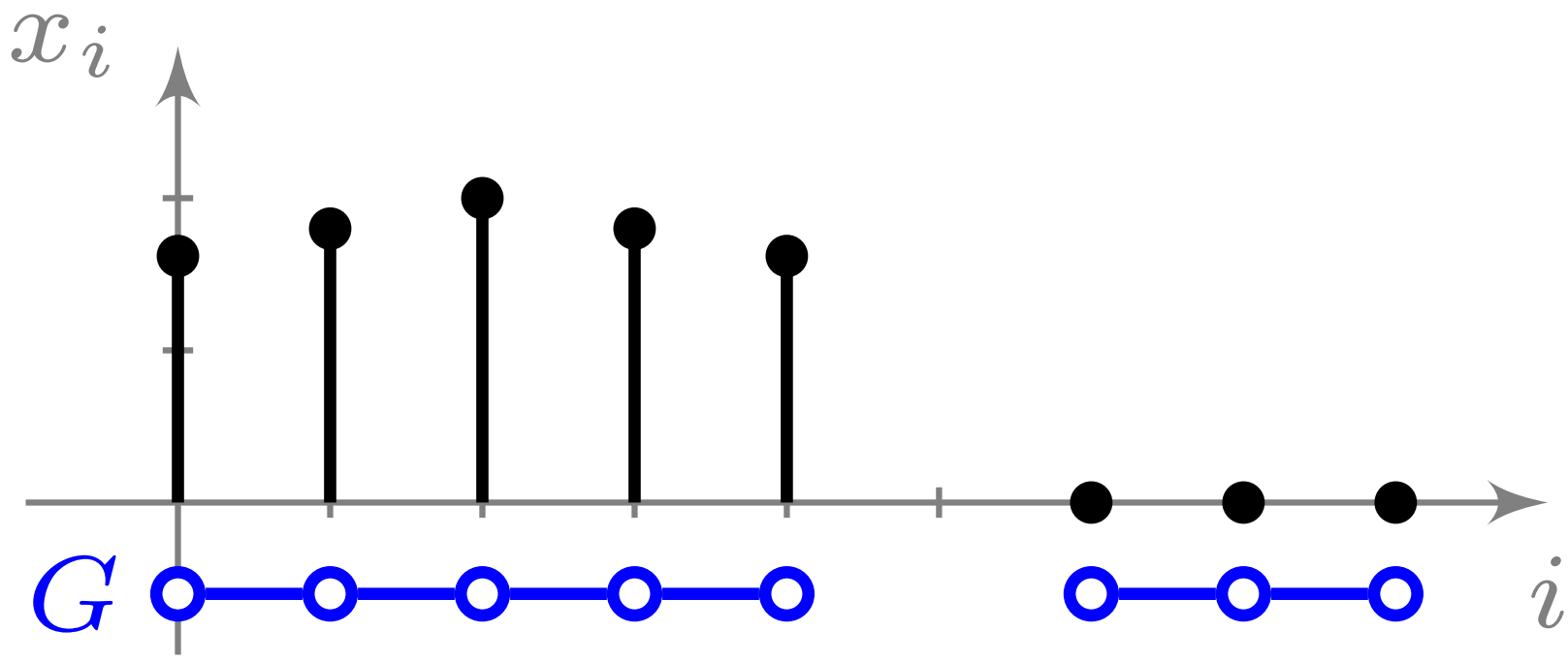
Method	d	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
global descriptors - nearest neighbor search						
CroW†	512	-	68.2	63.2	79.8	71.0
R-MAC	512	47.7	77.7	70.1	84.1	76.8
R-MAC	2,048	62.6	83.9	80.8	93.8	89.9
NetVLAD†	4,096	-	71.6	-	79.7	-
global descriptors - query expansion						
R-MAC+AQE	512	57.3	85.4	79.7	88.4	83.5
R-MAC+SCSM	512	60.1	85.3	80.5	89.4	84.5
R-MAC+HN	512	64.7	79.9	-	92.0	-
Global diffusion	512	70.3	85.7	82.7	94.1	92.5
R-MAC+AQE	2,048	70.5	**89.6**	**88.3**	95.3	92.7
R-MAC+SCSM	2,048	71.4	89.1	87.3	95.4	92.5
Global diffusion	2,048	**80.5**	87.1	87.4	**96.5**	**95.4**

state of the art (regional)

Method	$m \times d$	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
regional descriptors - nearest neighbor search						
R-match	21×512	55.5	81.5	76.5	86.1	79.9
R-match	21×2,048	71.0	88.1	85.7	94.9	91.3
regional descriptors - query expansion						
HQE	2.4k×128	74.7	89.4†	84.0†	82.8†	-
R-match+AQE	21×512	60.4	83.6	78.6	87.0	81.0
Regional diffusion*	5×512	77.5	91.5	84.7	95.6	93.0
Regional diffusion*	21×512	80.0	93.2	90.3	96.5	92.6
R-match+AQE	21×2,048	77.1	91.0	89.6	95.5	92.5
Regional diffusion*	5×2,048	88.4	95.0	90.0	96.4	**95.8**
Regional diffusion*	21×2,048	**89.6**	**95.8**	**94.2**	**96.9**	95.3

more challenges

- how to trade-off offline with online cost?
- how to get rid of truncation?
- how to generalize beyond the particular model?



fast spectral ranking

faster than CG?

- want to solve $\mathcal{L}_\alpha \mathbf{x} = \mathbf{y}$
- could invert $\mathcal{L}_\alpha$ offline, but it wouldn't be sparse
- could approximate $\mathcal{L}_\alpha^{-1}$ by $\Phi\Phi^\top$ where Φ is a (sparse) $n \times r$ matrix with $r \ll n$; then

$$\mathbf{x} \approx \Phi\Phi^\top \mathbf{y}$$

- but how to compute Φ without ever inverting $\mathcal{L}_\alpha$?
- still, there is no generalization; even α is given in advance

faster than CG?

- want to solve $\mathcal{L}_\alpha \mathbf{x} = \mathbf{y}$
- could invert $\mathcal{L}_\alpha$ offline, but it wouldn't be sparse
- could approximate $\mathcal{L}_\alpha^{-1}$ by $\Phi\Phi^\top$ where Φ is a (sparse) $n \times r$ matrix with $r \ll n$; then

$$\mathbf{x} \approx \Phi\Phi^\top \mathbf{y}$$

- but how to compute Φ without ever inverting $\mathcal{L}_\alpha$?
- still, there is no generalization; even α is given in advance

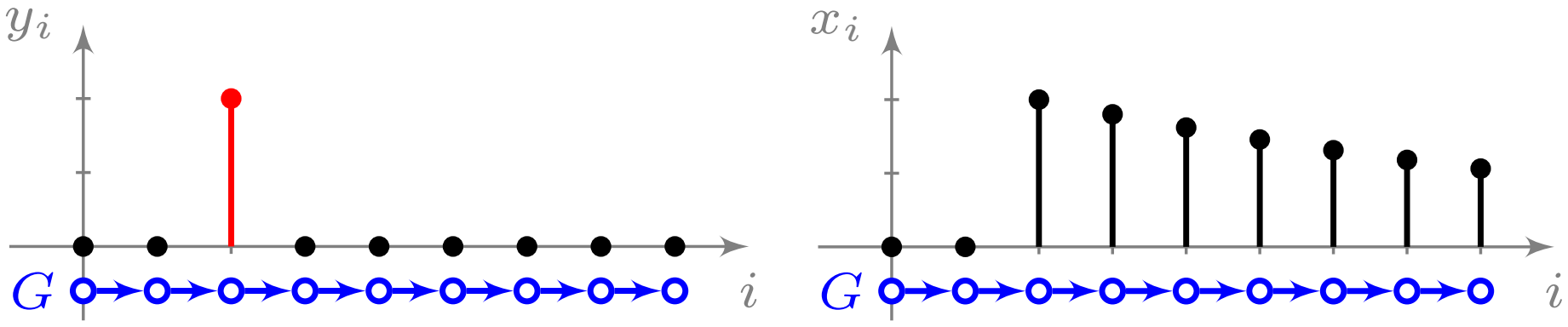
faster than CG?

- want to solve $\mathcal{L}_\alpha \mathbf{x} = \mathbf{y}$
- could invert $\mathcal{L}_\alpha$ offline, but it wouldn't be sparse
- could approximate $\mathcal{L}_\alpha^{-1}$ by $\Phi\Phi^\top$ where Φ is a (sparse) $n \times r$ matrix with $r \ll n$; then

$$\mathbf{x} \approx \Phi\Phi^\top \mathbf{y}$$

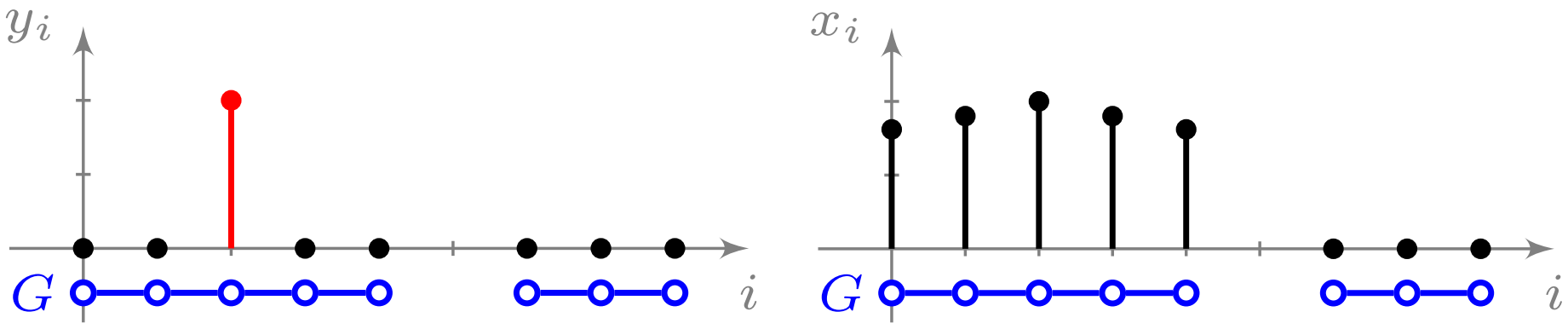
- but how to compute Φ without ever inverting $\mathcal{L}_\alpha$?
- still, there is no generalization; even α is given in advance

ranking as low-pass filtering



- output given by $x_i := \beta \sum_{t=0}^{\infty} \alpha^t y_{i-t}$
- or by recurrence $x_i = \alpha x_{i-1} + (1-\alpha) y_i$
- impulse response $h_i = \beta \alpha^i u_i$
- transfer function $H(z) := \beta \sum_{t=0}^{\infty} (az^{-1})^t = \beta / (1 - \alpha z^{-1})$

ranking as low-pass filtering



- using a weighted undirected graph G instead
- information "flows" in all directions, controlled by edge weights

transfer function

- in general, a function $h : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S}$ is the set of real symmetric square matrices including scalars, $\mathbb{R}$
- given $\mathcal{W}$ (offline) and $h, \mathbf{y}$ (online), the problem is now to compute

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

- our standard choice is

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

- recalling that $\mathcal{L}_\alpha = \beta^{-1}(I - \alpha\mathcal{W})$,

$$\mathcal{L}_\alpha^{-1} = h_\alpha(\mathcal{W})$$

transfer function

- in general, a function $h : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S}$ is the set of real symmetric square matrices including scalars, $\mathbb{R}$
- given $\mathcal{W}$ (offline) and $h, \mathbf{y}$ (online), the problem is now to compute

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

- our standard choice is

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

- recalling that $\mathcal{L}_\alpha = \beta^{-1}(I - \alpha\mathcal{W})$,

$$\mathcal{L}_\alpha^{-1} = h_\alpha(\mathcal{W})$$

(exact) spectral ranking

- given $A \in \mathcal{S}$ offline, compute the exact eigenvalue decomposition

$$U \Lambda U^\top = A$$

- given $h, \mathbf{y}$ online, compute

$$\mathbf{x} := U h(\Lambda) U^\top \mathbf{y}$$

where $h(\Lambda)$ is computed element-wise!

(exact) spectral ranking

- given $A \in \mathcal{S}$ offline, compute the exact eigenvalue decomposition

$$U \Lambda U^\top = A$$

- given $h, \mathbf{y}$ online, compute

$$\mathbf{x} := U h(\Lambda) U^\top \mathbf{y}$$

where $h(\Lambda)$ is computed element-wise!

when/why does it work?

- let $\mathcal{H}$ be the family of functions h with a series expansion

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

- if $h \in \mathcal{H}$ and the series converges, then

$$h(A) = U h(\Lambda) U^\top = U \operatorname{diag}(h(\lambda_1), \ldots, h(\lambda_n)) U^\top$$

- in particular, $h_\alpha \in \mathcal{H}$, having the *geometric progression* expansion

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

  which converges absolutely if $\varrho(\alpha A) < 1$
- but, this holds for $A = \mathcal{W}$ because $\alpha < 1$ and $\varrho(\mathcal{W}) = 1$

when/why does it work?

- let $\mathcal{H}$ be the family of functions h with a series expansion

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

- if $h \in \mathcal{H}$ and the series converges, then

$$h(A) = U h(\Lambda) U^\top = U \operatorname{diag}(h(\lambda_1), \ldots, h(\lambda_n)) U^\top$$

- in particular, $h_\alpha \in \mathcal{H}$, having the *geometric progression* expansion

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

  which converges absolutely if $\varrho(\alpha A) < 1$
- but, this holds for $A = \mathcal{W}$ because $\alpha < 1$ and $\varrho(\mathcal{W}) = 1$

low-rank approximation: stage 1

- given A, compute an $n \times \hat{r}$ matrix Q with $Q^\top Q = I_{\hat{r}}$ that represents an approximate basis for the range of A:

$$QQ^\top A \approx A$$

- how? simultaneous iteration: randomly draw an $n \times \hat{r}$ standard Gaussian matrix $B^{(0)}$ and repeat for $t = 0, \ldots, q-1$:
  1. compute QR factorization $Q^{(t)} R^{(t)} = B^{(t)}$
  2. define the $n \times \hat{r}$ matrix $B^{(t+1)} := AQ^{(t)}$
- finally, set $Q := Q^{(q-1)}$, $B := B^{(q)} = AQ$

low-rank approximation: stage 1

- given A, compute an $n \times \hat{r}$ matrix Q with $Q^\top Q = I_{\hat{r}}$ that represents an approximate basis for the range of A:

$$QQ^\top A \approx A$$

- how? simultaneous iteration: randomly draw an $n \times \hat{r}$ standard Gaussian matrix $B^{(0)}$ and repeat for $t = 0, \ldots, q-1$:
  1. compute QR factorization $Q^{(t)} R^{(t)} = B^{(t)}$
  2. define the $n \times \hat{r}$ matrix $B^{(t+1)} := AQ^{(t)}$
- finally, set $Q := Q^{(q-1)}$, $B := B^{(q)} = AQ$

low-rank approximation: stage 2

- compute an approximate rank-r eigenvalue decomposition

$$U \Lambda U^\top \approx A$$

  where U is $n \times r$ with $U^\top U = I_r$ and Λ is $r \times r$ diagonal

- how? [Halko *et al.* SIAM 2011]
  1. form the $\hat{r} \times \hat{r}$ matrix $C := Q^\top B = Q^\top A Q$
  2. compute its eigendecomposition $\hat{V} \hat{\Lambda} \hat{V}^\top = C$
  3. form (V, Λ) by keeping from $(\hat{V}, \hat{\Lambda})$ the rows/columns corresponding to the r largest eigenvalues
  4. define $U := QV$

low-rank approximation: stage 2

- compute an approximate rank-r eigenvalue decomposition

$$U\Lambda U^\top \approx A$$

  where U is $n \times r$ with $U^\top U = I_r$ and Λ is $r \times r$ diagonal
- how? [Halko *et al.* SIAM 2011]
  1. form the $\hat{r} \times \hat{r}$ matrix $C := Q^\top B = Q^\top A Q$
  2. compute its eigendecomposition $\hat{V}\hat{\Lambda}\hat{V}^\top = C$
  3. form (V, Λ) by keeping from $(\hat{V}, \hat{\Lambda})$ the rows/columns corresponding to the r largest eigenvalues
  4. define $U := QV$

when/why does it work?

- an average-case bound on $\left\|A - QQ^\top A\right\|$ decays to $|\lambda_{r+1}|$ exponentially fast in q [Halko *et al.* SIAM 2011]
- since $QQ^\top A \approx A$ and A is symmetric,

$$A \approx QQ^\top AQQ^\top = QCQ^\top \approx QV\Lambda V^\top Q^\top = U\Lambda U^\top$$

- the approximation $C \approx V\Lambda V^\top$ involves an additional term of $|\lambda_{r+1}|$ in the error [Halko *et al.* SIAM 2011]
- when approximating $h(A)$ by $Uh(\Lambda)U^\top$, $|h(\lambda_{r+1})|$ governs the error instead: h should be nondecreasing when restricted to scalars

when/why does it work?

- an average-case bound on $\left\|A - QQ^\top A\right\|$ decays to $|\lambda_{r+1}|$ exponentially fast in q [Halko *et al.* SIAM 2011]
- since $QQ^\top A \approx A$ and A is symmetric,

$$A \approx QQ^\top AQQ^\top = QCQ^\top \approx QV\Lambda V^\top Q^\top = U\Lambda U^\top$$

- the approximation $C \approx V\Lambda V^\top$ involves an additional term of $|\lambda_{r+1}|$ in the error [Halko *et al.* SIAM 2011]
- when approximating $h(A)$ by $Uh(\Lambda)U^\top$, $|h(\lambda_{r+1})|$ governs the error instead: h should be nondecreasing when restricted to scalars

back to image retrieval

- given $\mathcal{W}$ offline, compute the rank-r eigenvalue decomposition

$$U\Lambda U^\top \approx \mathcal{W}$$

- given $h, \mathbf{y}$ online, compute

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

- score per image obtained by sparse $N \times n$ pooling matrix Σ

$$\overline{\mathbf{x}} := \Sigma \mathbf{x}$$

- the $N \times r$ matrix $\overline{U} := \Sigma U$ is computed offline so that, online,

$$\overline{\mathbf{x}} = \overline{U}h(\Lambda)U^\top \mathbf{y}$$

back to image retrieval

- given $\mathcal{W}$ offline, compute the rank-r eigenvalue decomposition

$$U\Lambda U^\top \approx \mathcal{W}$$

- given $h, \mathbf{y}$ online, compute

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

- score per image obtained by sparse $N \times n$ pooling matrix Σ

$$\overline{\mathbf{x}} := \Sigma \mathbf{x}$$

- the $N \times r$ matrix $\overline{U} := \Sigma U$ is computed offline so that, online,

$$\overline{\mathbf{x}} = \overline{U}h(\Lambda)U^\top \mathbf{y}$$

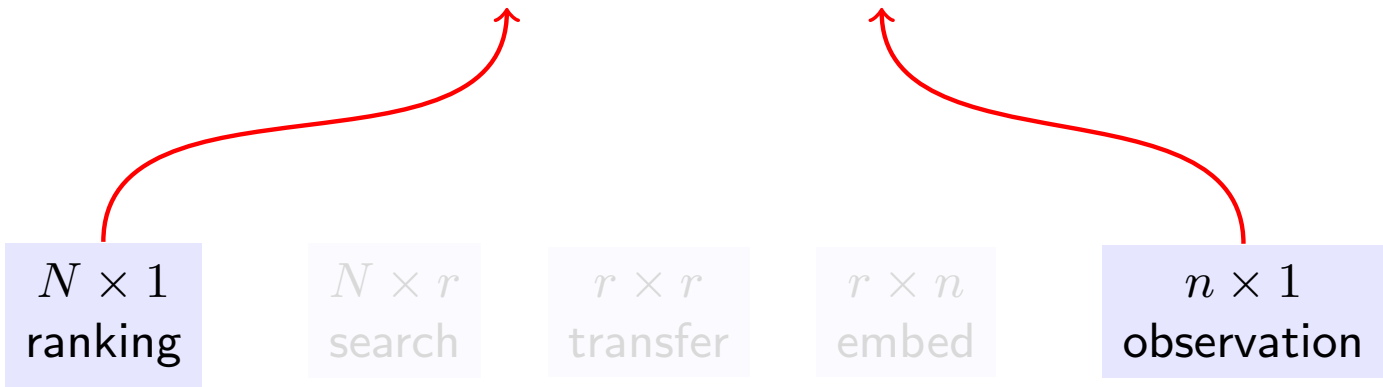
(fast) spectral ranking

$$\overline{\mathbf{x}} = \overline{U} h(\Lambda) U^\top \mathbf{y}$$

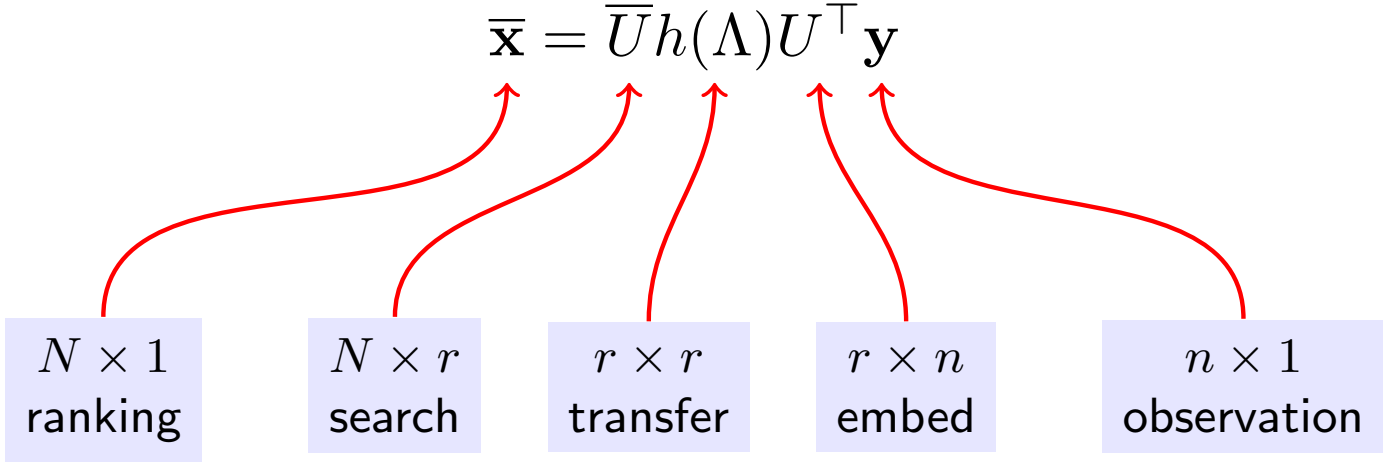
$N \times 1$	$N \times r$	$r \times r$	$r \times n$	$n \times 1$
ranking	search	transfer	embed	observation

(fast) spectral ranking

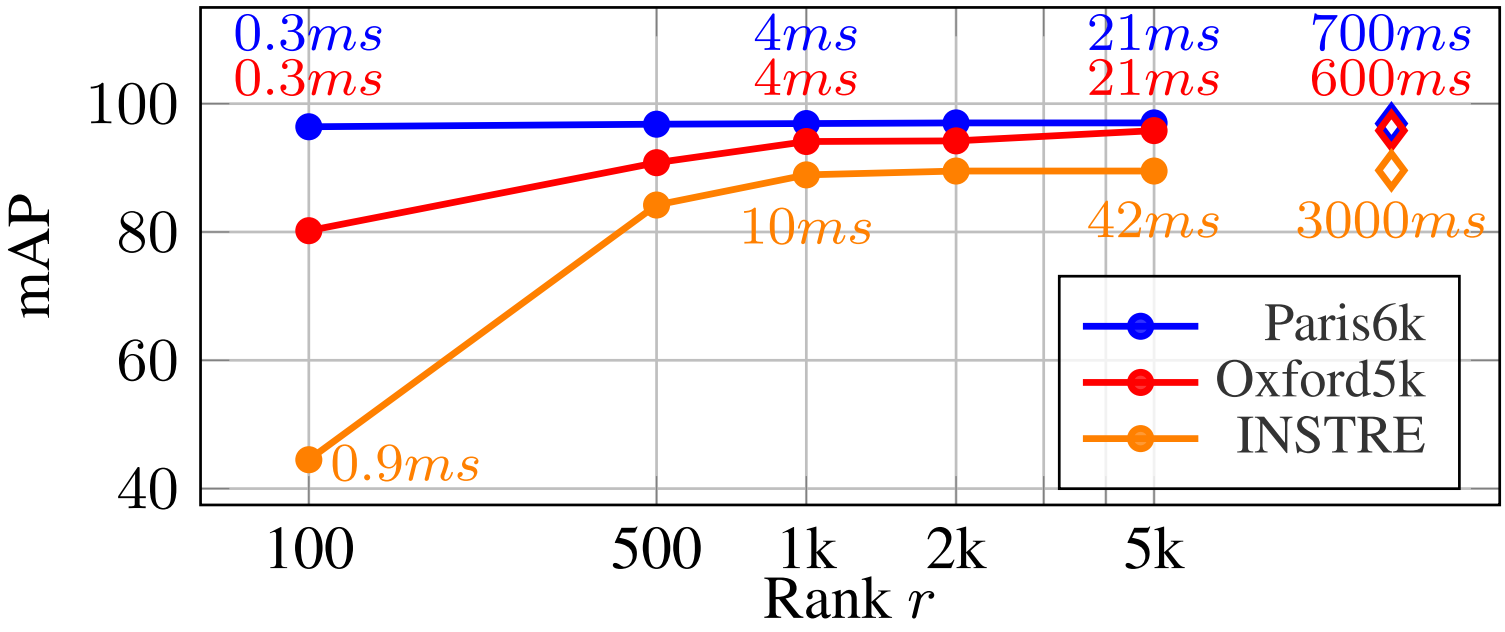
$$\overline{\mathbf{x}} = \overline{U} h(\Lambda) U^\top \mathbf{y}$$



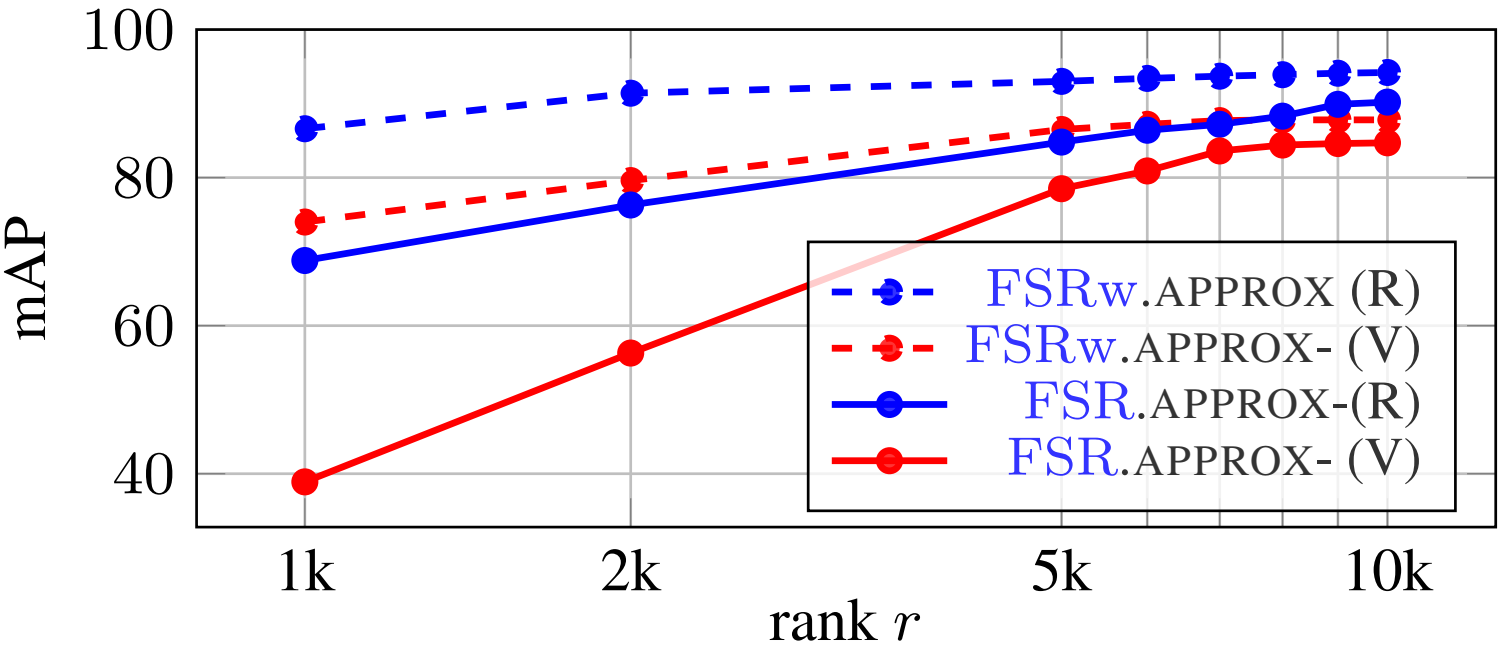
(fast) spectral ranking



small scale


0.3ms
0.3ms
4ms
4ms
21ms
21ms
700ms
600ms
10ms
42ms
3000ms
0.9ms
100
80
60
40
mAP
100
500
1k
2k
5k
Rank r
Paris6k
Oxford5k
INSTRE

Oxford105k


100
80
60
40
mAP
1k
2k
5k
10k
rank r
FSRw.APPROX (R)
FSRw.APPROX- (V)
FSR.APPROX-(R)
FSR.APPROX- (V)

practical considerations

- search independently in each connected component of G; otherwise maximal eigenvalue of each component dominates the eigenvalues of the few (or one) "giant" component
- "weighted" FSR: if η_i is the ℓ^2-norm of the i-th row of U, adjust ranking vector as

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

falling back on original dot-product similarity for sparsely populated parts of the graph

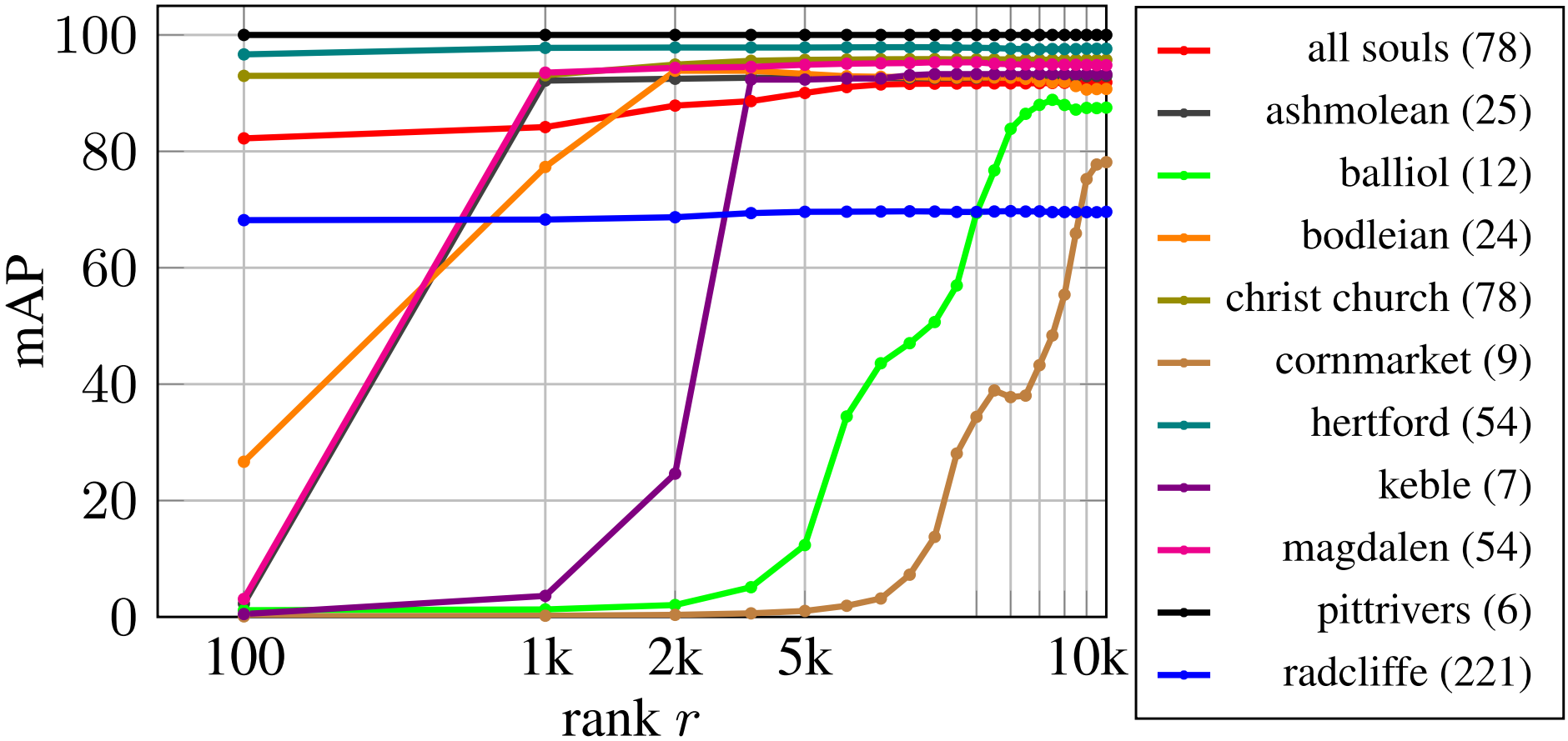
practical considerations

- search independently in each connected component of G; otherwise maximal eigenvalue of each component dominates the eigenvalues of the few (or one) "giant" component
- "weighted" FSR: if η_i is the ℓ^2-norm of the i-th row of U, adjust ranking vector as

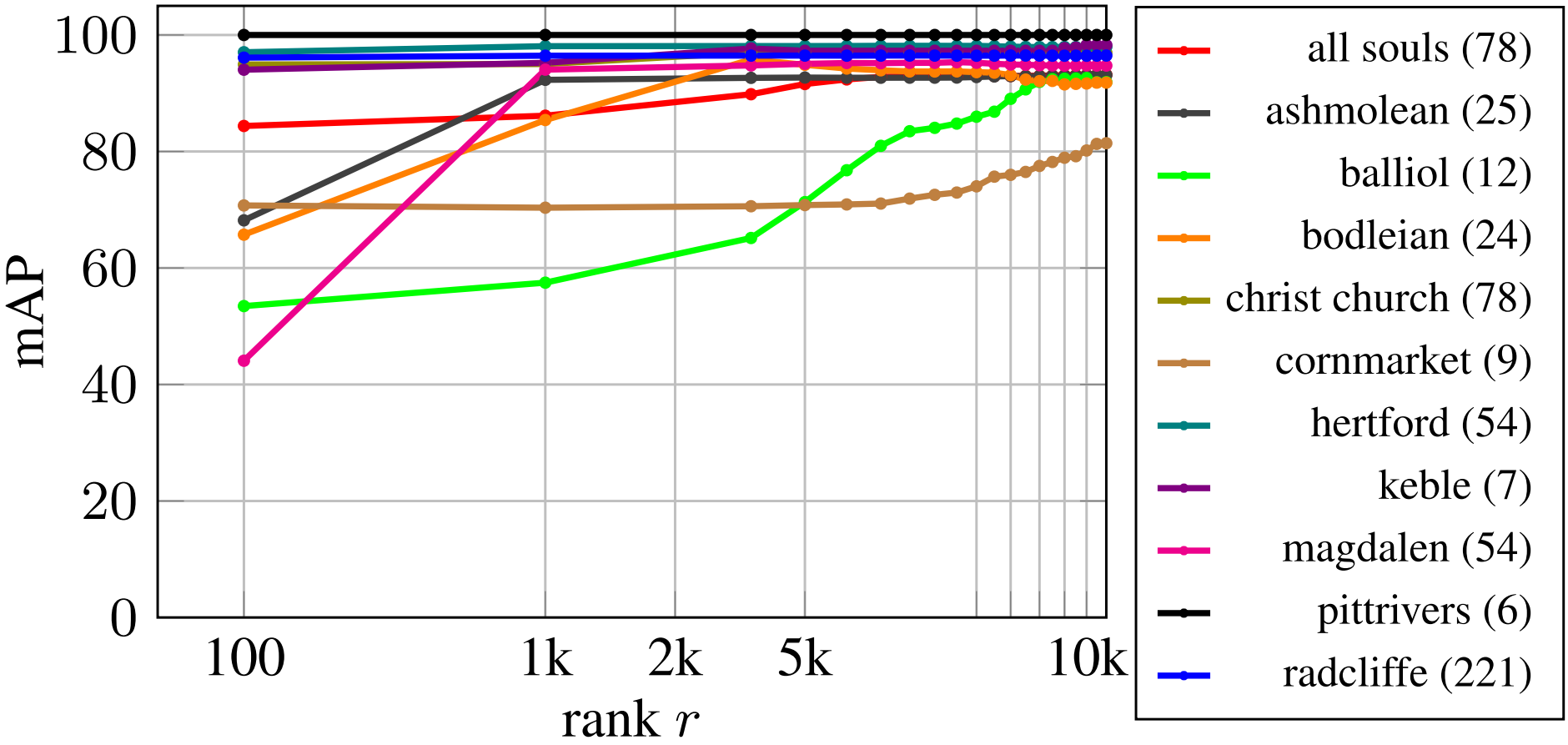
$$x'_i = x_i + (1 - \eta_i)\mathbf{v}_i^\top \mathbf{q}$$

falling back on original dot-product similarity for sparsely populated parts of the graph

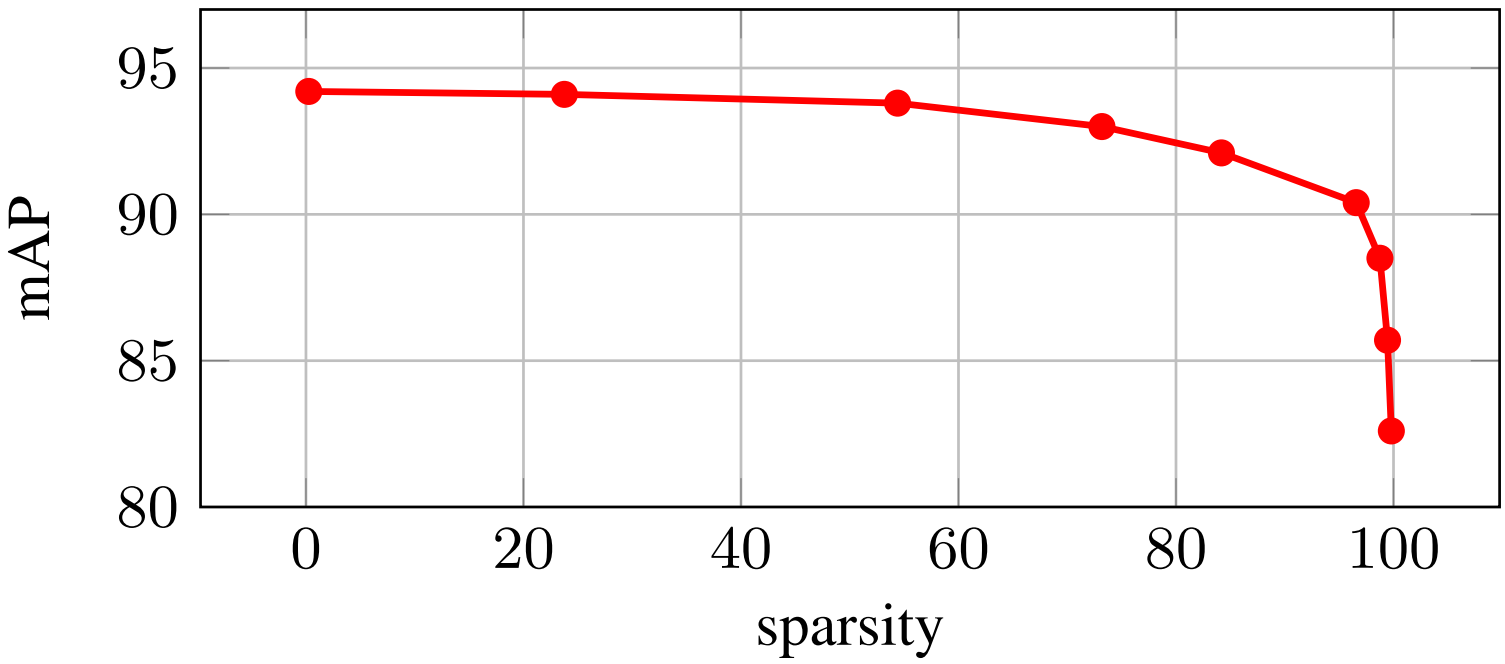
Oxford105k per landmark (FSR)



Oxford105k per landmark (weighted FSR)



sparse U (Oxford105k, ResNet101)



state of the art (global)

Method	d	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
global descriptors - Euclidean search						
R-MAC	512	47.7	77.7	70.1	84.1	76.8
R-MAC	2,048	62.6	83.9	80.8	93.8	89.9
global descriptors - manifold search						
Diffusion	512	70.3	85.7	82.5	94.1	92.5
FSR.RANK-r	512	70.3	85.8	85.0	93.8	92.4
Diffusion	2,048	**80.5**	87.1	86.8	**96.5**	**95.4**
FSR.RANK-r	2,048	**80.5**	**87.5**	**87.9**	96.4	95.3

state of the art (regional)

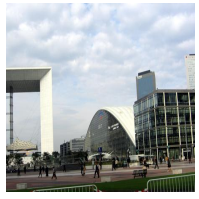
Method	$m \times d$	INSTRE	Oxf5k	Oxf105k	Par6k	Par106k
regional descriptors - Euclidean search						
R-match	21×512	55.5	79.8	76.5	86.1	79.9
R-match	21×2,048	71.0	88.1	85.7	94.9	91.3
regional descriptors - manifold search						
Diffusion	5×512	77.5	91.5	84.7	95.6	93.0
FSR.APPROX	5×512	78.4	89.9	86.5	95.6	92.4
Diffusion	21×512	80.0	93.2	90.3	96.5	92.6
FSR.APPROX	21×512	80.4	90.6	-	96.5	-
Diffusion	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	21×2,048	**89.6**	**95.8**	**94.2**	96.9	**95.3**
FSR.APPROX	21×2,048	89.2	**95.8**	-	**97.0**	-

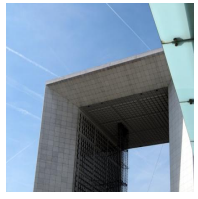
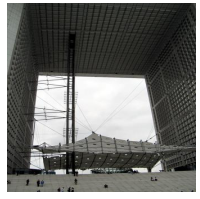
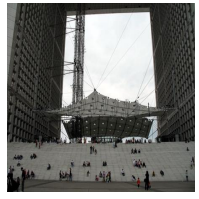
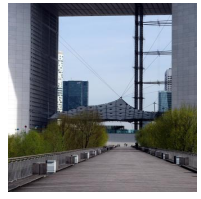
query time (Oxford105k)

- rank $r = 5$k: 0.14s
- rank $r = 10$k: 0.30s
- CG: 14s
- CG (truncated): 1s

hard examples?


(AP: 92.1)

#5

#32

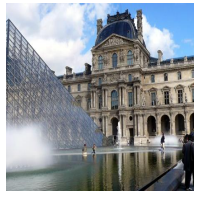
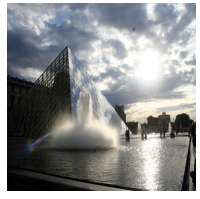
#51

#70

#71

#76

#79

#126

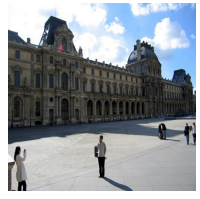

(AP: 92.7)

#2

#4

#8

#61

#68

#72

#75

#108

interpretation: random fields

- a Gaussian Markov random field (GMRF) with precision A and mean $\boldsymbol{\mu}$ can be parametrized as

$$p(\mathbf{x}) := \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A^{-1}) \propto e^{-E(\mathbf{x}|\mathbf{b},A)}$$

  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
- our solution $\mathbf{x}^* = \mathcal{L}_\alpha^{-1}\mathbf{y}$ is the expectation of a GMRF with energy

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

- if $\hat{\mathbf{x}} := D^{-1/2}\mathbf{x}$, this energy has the same minimizer as

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

interpretation: random fields

- a Gaussian Markov random field (GMRF) with precision A and mean $\boldsymbol{\mu}$ can be parametrized as

$$p(\mathbf{x}) := \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A^{-1}) \propto e^{-E(\mathbf{x}|\mathbf{b},A)}$$

  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
- our solution $\mathbf{x}^* = \mathcal{L}_\alpha^{-1}\mathbf{y}$ is the expectation of a GMRF with energy

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

- if $\hat{\mathbf{x}} := D^{-1/2}\mathbf{x}$, this energy has the same minimizer as

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

interpretation: random fields

- a Gaussian Markov random field (GMRF) with precision A and mean $\boldsymbol{\mu}$ can be parametrized as

$$p(\mathbf{x}) := \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, A^{-1}) \propto e^{-E(\mathbf{x}|\mathbf{b},A)}$$

  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
- our solution $\mathbf{x}^* = \mathcal{L}_\alpha^{-1}\mathbf{y}$ is the expectation of a GMRF with energy

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

- if $\hat{\mathbf{x}} := D^{-1/2}\mathbf{x}$, this energy has the same minimizer as

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

interpretation: graph filtering

- a signal of period n is a vector $\mathbf{s} \in \mathbb{R}^n$ where $s_{\bar{i}} := s_{(i \bmod n)+1}$
- a shift of $\mathbf{s}$ is the mapping $s_{\bar{i}} \mapsto s_{\overline{i-1}}$; also represented by $\mathbf{s} \mapsto C_n\mathbf{s}$ where C_n is an $n \times n$ circulant zero-one matrix
- A linear, shift invariant, causal filter is the mapping $\mathbf{s} \mapsto H\mathbf{s}$ where

$$H := h(C_n) = \sum_{t=0}^{\infty} h_t C_n^t$$

- 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

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

interpretation: graph filtering

- a signal of period n is a vector $\mathbf{s} \in \mathbb{R}^n$ where $s_{\bar{i}} := s_{(i \bmod n)+1}$
- a shift of $\mathbf{s}$ is the mapping $s_{\bar{i}} \mapsto s_{\overline{i-1}}$; also represented by $\mathbf{s} \mapsto C_n\mathbf{s}$ where C_n is an $n \times n$ circulant zero-one matrix
- A linear, shift invariant, causal filter is the mapping $\mathbf{s} \mapsto H\mathbf{s}$ where

$$H := h(C_n) = \sum_{t=0}^{\infty} h_t C_n^t$$

- 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

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

interpretation: graph filtering

- a signal of period n is a vector $\mathbf{s} \in \mathbb{R}^n$ where $s_{\bar{i}} := s_{(i \bmod n)+1}$
- a shift of $\mathbf{s}$ is the mapping $s_{\bar{i}} \mapsto s_{\overline{i-1}}$; also represented by $\mathbf{s} \mapsto C_n\mathbf{s}$ where C_n is an $n \times n$ circulant zero-one matrix
- A linear, shift invariant, causal filter is the mapping $\mathbf{s} \mapsto H\mathbf{s}$ where

$$H := h(C_n) = \sum_{t=0}^{\infty} h_t C_n^t$$

- 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

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

summary

- do not inject query into dataset; search for its neighbors instead
- diffusion is a (slow) iterative solver; use CG instead
- still expensive at large scale: truncate
- cast retrieval as linear graph filtering in the frequency domain
- efficiently compute an approximate Fourier basis of the graph offline
- reduce manifold search to Euclidean followed by dot product similarity search

summary

- do not inject query into dataset; search for its neighbors instead
- diffusion is a (slow) iterative solver; use CG instead
- still expensive at large scale: truncate
- cast retrieval as linear graph filtering in the frequency domain
- efficiently compute an approximate Fourier basis of the graph offline
- reduce manifold search to Euclidean followed by dot product similarity search

diffusion on region manifolds (CVPR 2017)

https://arxiv.org/abs/1611.05113

fast spectral ranking

https://arxiv.org/abs/1703.06935

thank you!