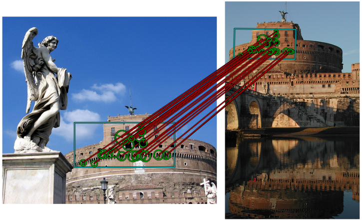
Image retrieval, vector quantization and nearest neighbor search

Yannis Avrithis

National Technical University of Athens

Rennes, October 2014

Part I: Image retrieval

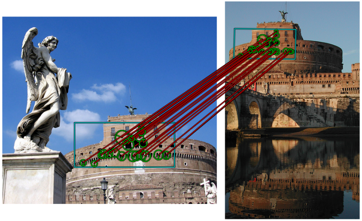


- Particular object retrieval
- Match images under different viewpoint/lighting, occlusion
- Given local descriptors, investigate match kernels beyond Bag-of-Words

Part II: Vector quantization and nearest neighbor search

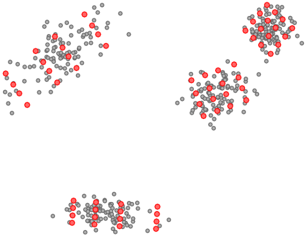
- Fast nearest neighbor search in high-dimensional spaces
- Encode vectors based on vector quantization
- Improve fitting to underlying distribution

Part I: Image retrieval



- Particular object retrieval
- Match images under different viewpoint/lighting, occlusion
- Given local descriptors, investigate match kernels beyond Bag-of-Words

Part II: Vector quantization and nearest neighbor search



- Fast nearest neighbor search in high-dimensional spaces
- Encode vectors based on vector quantization
- Improve fitting to underlying distribution

Part I: Image retrieval

To aggregate or not to aggregate: selective match kernels for image search

Joint work with Giorgos Tolias and Hervé Jégou, ICCV 2013

Overview

- Problem: particular object retrieval
- Build common model for matching (HE) and aggregation (VLAD) methods; derive new match kernels
- Evaluate performance under exact or approximate descriptors

Related work

- In our common model:
  - Bag-of-Words (BoW) [Sivic & Zisserman '03]
  - Descriptor approximation (Hamming embedding) [Jégou *et al.* '08]
  - Aggregated representations (VLAD, Fisher) [Jégou *et al.* '10][Perronnin *et al.* '10]
- Relevant to Part II:
  - Soft (multiple) assignment [Philbin *et al.* '08][Jégou *et al.* '10]
- Not discussed:
  - Spatial matching [Philbin *et al.* '08][Tolias & Avrithis '11]
  - Query expansion [Chum *et al.* '07][Tolias & Jégou '13]
  - Re-ranking [Qin *et al.* '11][Shen *et al.* '12]

Related work

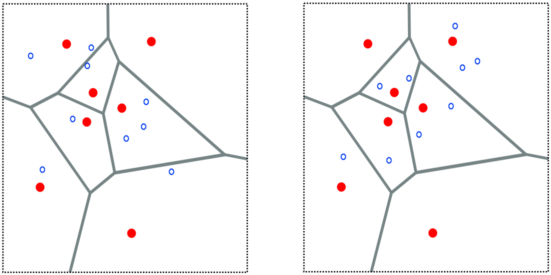
- In our common model:
  - Bag-of-Words (BoW) [Sivic & Zisserman '03]
  - Descriptor approximation (Hamming embedding) [Jégou *et al.* '08]
  - Aggregated representations (VLAD, Fisher) [Jégou *et al.* '10][Perronnin *et al.* '10]
- Relevant to Part II:
  - Soft (multiple) assignment [Philbin *et al.* '08][Jégou *et al.* '10]
- Not discussed:
  - Spatial matching [Philbin *et al.* '08][Tolias & Avrithis '11]
  - Query expansion [Chum *et al.* '07][Tolias & Jégou '13]
  - Re-ranking [Qin *et al.* '11][Shen *et al.* '12]

Related work

- In our common model:
  - Bag-of-Words (BoW) [Sivic & Zisserman '03]
  - Descriptor approximation (Hamming embedding) [Jégou *et al.* '08]
  - Aggregated representations (VLAD, Fisher) [Jégou *et al.* '10][Perronnin *et al.* '10]
- Relevant to Part II:
  - Soft (multiple) assignment [Philbin *et al.* '08][Jégou *et al.* '10]
- Not discussed:
  - Spatial matching [Philbin *et al.* '08][Tolias & Avrithis '11]
  - Query expansion [Chum *et al.* '07][Tolias & Jégou '13]
  - Re-ranking [Qin *et al.* '11][Shen *et al.* '12]

Image representation

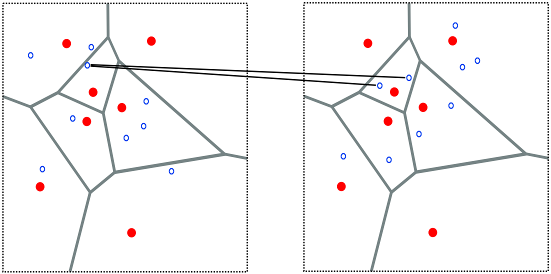
- Entire image: set of local descriptors $\mathcal{X} = \{x_1, \ldots, x_n\}$
- Descriptors assigned to cell c: $\mathcal{X}_c = \{x \in \mathcal{X} : q(x) = c\}$



Set similarity function

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

normalization factor | cell weighting | cell similarity



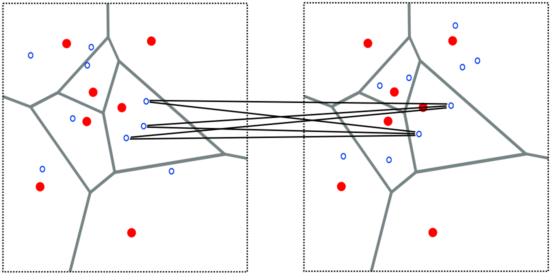
Set similarity function

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c\in\mathcal{C}} w_c \mathrm{M}\left(\mathcal{X}_c,\mathcal{Y}_c\right)$$

normalization factor

cell weighting

cell similarity



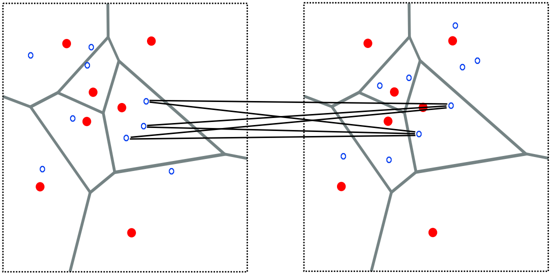
Set similarity function

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y})\sum_{c\in\mathcal{C}} w_c \mathrm{M}\left(\mathcal{X}_c,\mathcal{Y}_c\right)$$

normalization factor

cell weighting

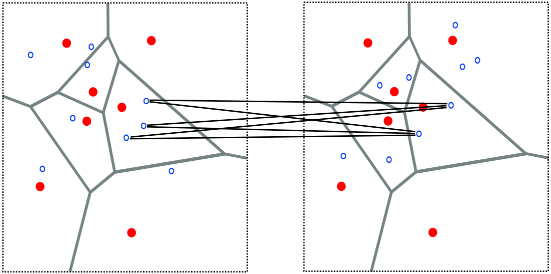
cell similarity



Set similarity function

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c\in\mathcal{C}} w_c \mathrm{M}\left(\mathcal{X}_c,\mathcal{Y}_c\right)$$

normalization factor cell weighting cell similarity



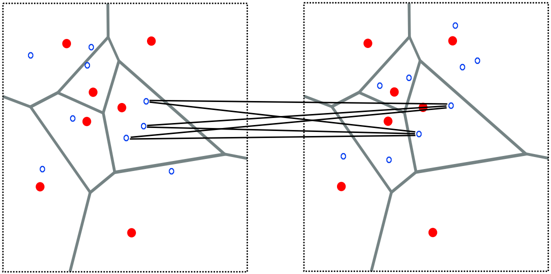
Set similarity function

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c\in\mathcal{C}} w_c \mathrm{M}\left(\mathcal{X}_c,\mathcal{Y}_c\right)$$

normalization factor

cell weighting

cell similarity



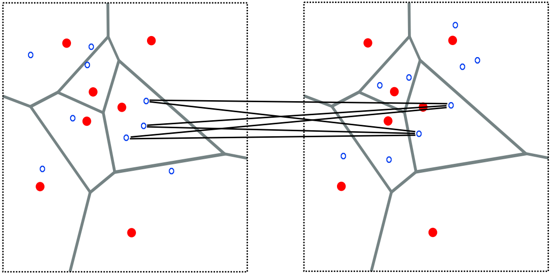
Set similarity function

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c\in\mathcal{C}} w_c \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

normalization factor

cell weighting

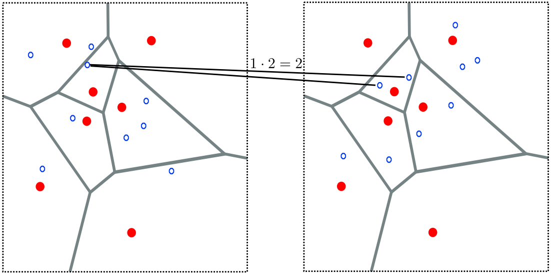
cell similarity



Bag-of-Words (BoW) similarity function

Cosine similarity

$$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = |\mathcal{X}_c| \times |\mathcal{Y}_c| = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} 1$$



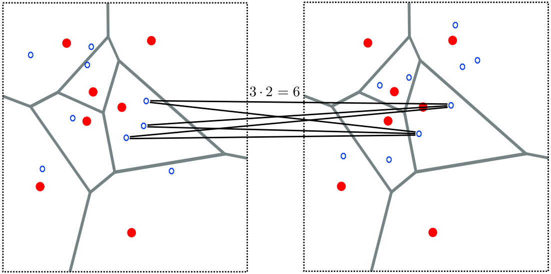
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \, \mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$$

Bag-of-Words (BoW) similarity function

Cosine similarity

$$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = |\mathcal{X}_c| \times |\mathcal{Y}_c| = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} 1$$



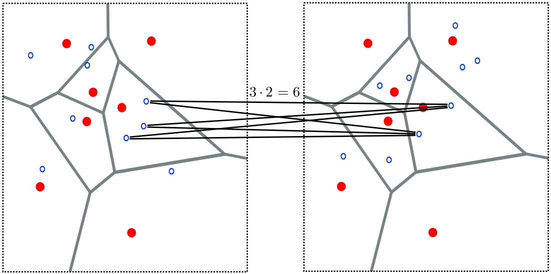
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

Bag-of-Words (BoW) similarity function

Cosine similarity

$$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = |\mathcal{X}_c| \times |\mathcal{Y}_c| = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} 1$$



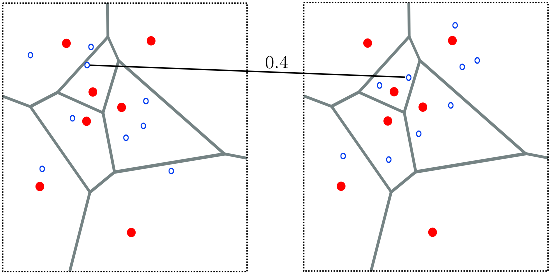
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

Hamming Embedding (HE)

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

weight function   Hamming distance   binary codes



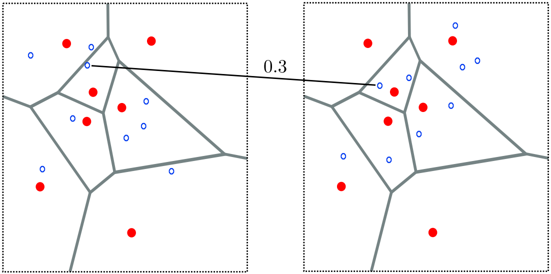
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

Hamming Embedding (HE)

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

weight function   Hamming distance   binary codes



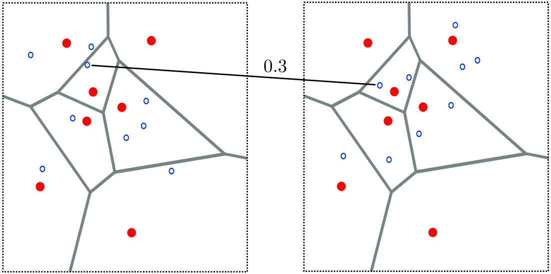
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

Hamming Embedding (HE)

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

weight function   Hamming distance   binary codes



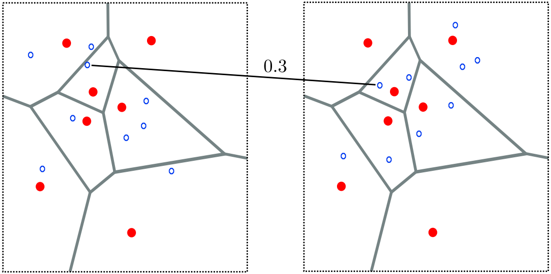
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

Hamming Embedding (HE)

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

weight function Hamming distance binary codes



Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \,\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

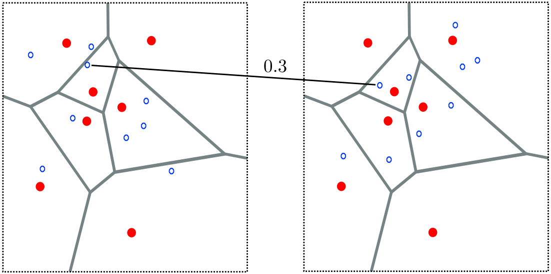
Hamming Embedding (HE)

$$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

weight function

Hamming distance

binary codes



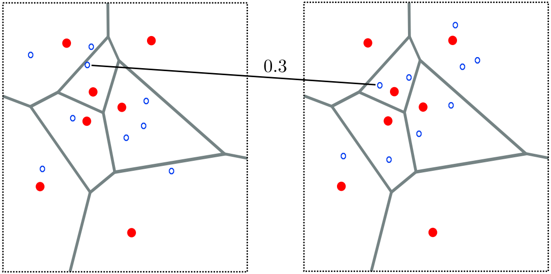
Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$$

Hamming Embedding (HE)

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

weight function | Hamming distance | binary codes



Generic set similarity

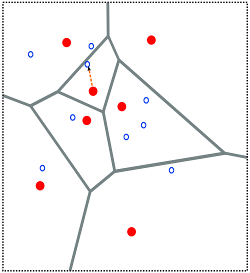
$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

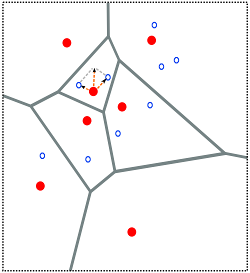
VLAD

$$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y)$$

aggregated residual $\sum_{x \in \mathcal{X}_c} r(x)$

residual $x - q(x)$





Generic set similarity

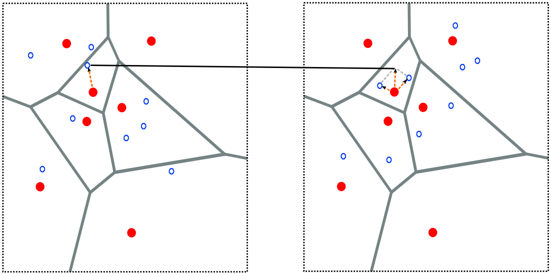
$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$$

VLAD

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y)$$

aggregated residual $\sum_{x \in \mathcal{X}_c} r(x)$

residual $x - q(x)$



Generic set similarity

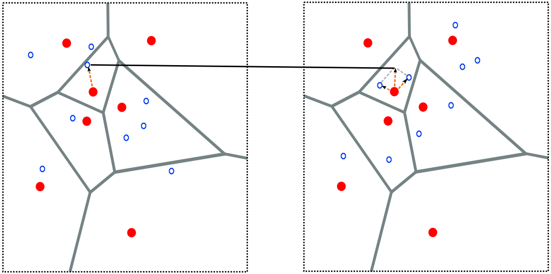
$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c\, \mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

VLAD

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y)$$

aggregated residual $\sum_{x \in \mathcal{X}_c} r(x)$

residual $x - q(x)$



Generic set similarity

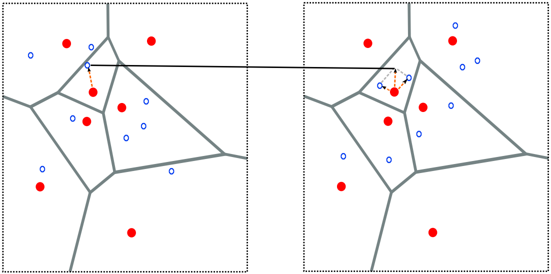
$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\, \gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \,\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

VLAD

$$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y)$$

aggregated residual $\sum_{x \in \mathcal{X}_c} r(x)$

residual $x - q(x)$



Generic set similarity

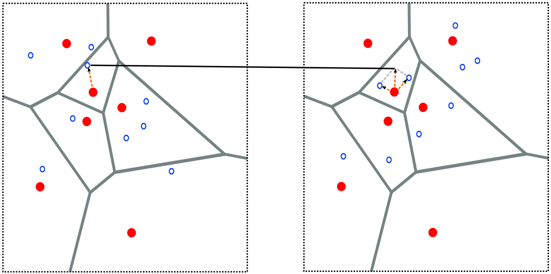
$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \,\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$$

VLAD

$$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y)$$

aggregated residual $\sum_{x \in \mathcal{X}_c} r(x)$

residual $x - q(x)$



Generic set similarity

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) = \gamma(\mathcal{X})\,\gamma(\mathcal{Y}) \sum_{c \in \mathcal{C}} w_c \,\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right)$$

Design choices

Hamming embedding

- Binary signature & voting per descriptor (not aggregated)
- Selective: discard weak votes

VLAD

- One aggregated vector per cell
- Linear operation

Questions

- Is aggregation good with large vocabularies (e.g. $65k$)?
- How important is selectivity in this case?

Design choices

Hamming embedding

- Binary signature & voting per descriptor (not aggregated)
- Selective: discard weak votes

VLAD

- One aggregated vector per cell
- Linear operation

Questions

- Is aggregation good with large vocabularies (e.g. $65k$)?
- How important is selectivity in this case?

Common model

Non aggregated

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma\left(\phi(x)^\top \phi(y)\right)$$

selectivity function

descriptor representation (residual, binary, scalar)

Aggregated

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x \in \mathcal{X}_c} \phi(x)\right)^\top \psi\left(\sum_{y \in \mathcal{Y}_c} \phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c)\right)$$

normalization (ℓ_2, power-law)

cell representation

Common model

Non aggregated

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma\left(\phi(x)^\top \phi(y)\right)$$

selectivity function

descriptor representation (residual, binary, scalar)

Aggregated

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x \in \mathcal{X}_c} \phi(x)\right)^\top \psi\left(\sum_{y \in \mathcal{Y}_c} \phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c)\right)$$

normalization (ℓ_2, power-law)

cell representation

Common model

Non aggregated

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma\left(\phi(x)^\top \phi(y)\right)$$

selectivity function

descriptor representation (residual, binary, scalar)

Aggregated

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x \in \mathcal{X}_c} \phi(x)\right)^\top \psi\left(\sum_{y \in \mathcal{Y}_c} \phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c)\right)$$

normalization (ℓ_2, power-law)

cell representation

Common model

Non aggregated

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma \left(\phi(x)^\top \phi(y) \right)$$

selectivity function

descriptor representation (residual, binary, scalar)

Aggregated

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma \left\{ \psi \left(\sum_{x \in \mathcal{X}_c} \phi(x) \right)^\top \psi \left(\sum_{y \in \mathcal{Y}_c} \phi(y) \right) \right\} = \sigma \left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c) \right)$$

normalization (ℓ_2, power-law)

cell representation

Common model

Non aggregated

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma\left(\phi(x)^\top \phi(y)\right)$$

selectivity function

descriptor representation (residual, binary, scalar)

Aggregated

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x \in \mathcal{X}_c} \phi(x)\right)^\top \psi\left(\sum_{y \in \mathcal{Y}_c} \phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c)\right)$$

normalization (ℓ_2, power-law)

cell representation

Common model

Non aggregated

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma \left(\phi(x)^\top \phi(y) \right)$$

selectivity function

descriptor representation (residual, binary, scalar)

Aggregated

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma \left\{ \psi \left(\sum_{x \in \mathcal{X}_c} \phi(x) \right)^\top \psi \left(\sum_{y \in \mathcal{Y}_c} \phi(y) \right) \right\} = \sigma \left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c) \right)$$

normalization (ℓ_2, power-law)

cell representation

BoW, HE and VLAD in the common model

Model	$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$	$\phi(x)$	$\sigma(u)$	$\psi(z)$	$\Phi(\mathcal{X}_c)$
BoW	$\mathrm{M_N}$ or $\mathrm{M_A}$	1	u	z	$\lvert\mathcal{X}_c\rvert$
HE	$\mathrm{M_N}$ only	$\hat{b}_x$	$w\left(\frac{B}{2}(1-u)\right)$	—	—
VLAD	$\mathrm{M_N}$ or $\mathrm{M_A}$	$r(x)$	u	z	$V(\mathcal{X}_c)$

BoW $$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x\in\mathcal{X}_c}\sum_{y\in\mathcal{Y}_c} 1 = \lvert\mathcal{X}_c\rvert \times \lvert\mathcal{Y}_c\rvert$$

HE $$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x\in\mathcal{X}_c}\sum_{y\in\mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

VLAD $$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x\in\mathcal{X}_c}\sum_{y\in\mathcal{Y}_c} r(x)^\top r(y) = V\left(\mathcal{X}_c\right)^\top V\left(\mathcal{Y}_c\right)$$

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x\in\mathcal{X}_c}\sum_{y\in\mathcal{Y}_c} \sigma\left(\phi(x)^\top\phi(y)\right)$$

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x\in\mathcal{X}_c}\phi(x)\right)^\top \psi\left(\sum_{y\in\mathcal{Y}_c}\phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top\Phi(\mathcal{Y}_c)\right)$$

BoW, HE and VLAD in the common model

Model	$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$	$\phi(x)$	$\sigma(u)$	$\psi(z)$	$\Phi(\mathcal{X}_c)$
BoW	$\mathrm{M_N}$ or $\mathrm{M_A}$	1	u	z	$\lvert\mathcal{X}_c\rvert$
HE	$\mathrm{M_N}$ only	$\hat{b}_x$	$w\left(\frac{B}{2}(1-u)\right)$	—	—
VLAD	$\mathrm{M_N}$ or $\mathrm{M_A}$	$r(x)$	u	z	$V(\mathcal{X}_c)$

BoW $$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} 1 = |\mathcal{X}_c| \times |\mathcal{Y}_c|$$

HE $$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

VLAD $$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c)$$

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma\left(\phi(x)^\top \phi(y)\right)$$

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x \in \mathcal{X}_c} \phi(x)\right)^\top \psi\left(\sum_{y \in \mathcal{Y}_c} \phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c)\right)$$

BoW, HE and VLAD in the common model

Model	$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c)$	$\phi(x)$	$\sigma(u)$	$\psi(z)$	$\Phi(\mathcal{X}_c)$
BoW	$\mathrm{M_N}$ or $\mathrm{M_A}$	1	u	z	$\lvert\mathcal{X}_c\rvert$
HE	$\mathrm{M_N}$ only	$\hat{b}_x$	$w\left(\frac{B}{2}(1-u)\right)$	—	—
VLAD	$\mathrm{M_N}$ or $\mathrm{M_A}$	$r(x)$	u	z	$V(\mathcal{X}_c)$

BoW $$\mathrm{M}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} 1 = |\mathcal{X}_c| \times |\mathcal{Y}_c|$$

HE $$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} w\left(\mathrm{h}\left(b_x, b_y\right)\right)$$

VLAD $$\mathrm{M}\left(\mathcal{X}_c, \mathcal{Y}_c\right) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} r(x)^\top r(y) = V(\mathcal{X}_c)^\top V(\mathcal{Y}_c)$$

$$\mathrm{M_N}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma\left(\phi(x)^\top \phi(y)\right)$$

$$\mathrm{M_A}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma\left\{\psi\left(\sum_{x \in \mathcal{X}_c} \phi(x)\right)^\top \psi\left(\sum_{y \in \mathcal{Y}_c} \phi(y)\right)\right\} = \sigma\left(\Phi(\mathcal{X}_c)^\top \Phi(\mathcal{Y}_c)\right)$$

Selective Match Kernel (SMK)

$$\mathrm{SMK}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma_\alpha(\hat{r}(x)^\top \hat{r}(y))$$

- Descriptor representation: ℓ_2-normalized residual

$$\phi(x) = \hat{r}(x) = r(x)/\|r(x)\|$$

- Selectivity function

$$\sigma_\alpha(u) = \begin{cases} \mathrm{sign}(u)|u|^\alpha, & u > \tau \\ 0, & \text{otherwise} \end{cases}$$

Selective Match Kernel (SMK)

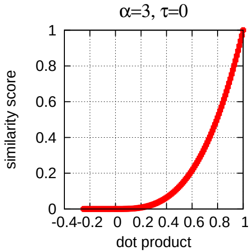
$$\mathsf{SMK}(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma_\alpha(\hat{r}(x)^\top \hat{r}(y))$$

- Descriptor representation: ℓ_2-normalized residual

$$\phi(x) = \hat{r}(x) = r(x)/\|r(x)\|$$

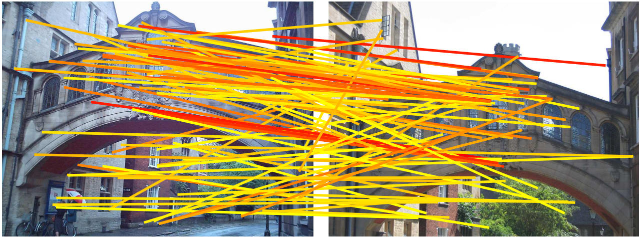
- Selectivity function

$$\sigma_\alpha(u) = \begin{cases} \operatorname{sign}(u)|u|^\alpha, & u > \tau \\ 0, & \text{otherwise} \end{cases}$$



Matching example—impact of threshold

$$\alpha = 1, \ \tau = 0.0$$

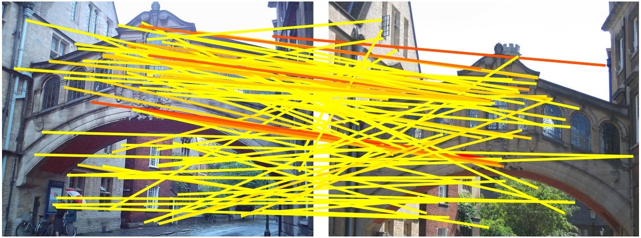


$$\alpha = 1, \ \tau = 0.25$$

thresholding removes false correspondences

Matching example—impact of shape parameter

$\alpha = 3, \ \tau = 0.0$



$\alpha = 3, \ \tau = 0.25$

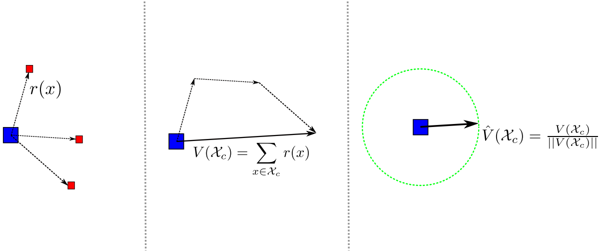
weighs matches based on confidence

Aggregated Selective Match Kernel (ASMK)

$$\mathsf{ASMK}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma_\alpha\left(\hat{V}(\mathcal{X}_c)^\top \hat{V}(\mathcal{Y}_c)\right)$$

- Cell representation: ℓ_2-normalized aggregated residual

$$\Phi(\mathcal{X}_c) = \hat{V}(\mathcal{X}_c) = V(\mathcal{X}_c)/\|V(\mathcal{X}_c)\|$$



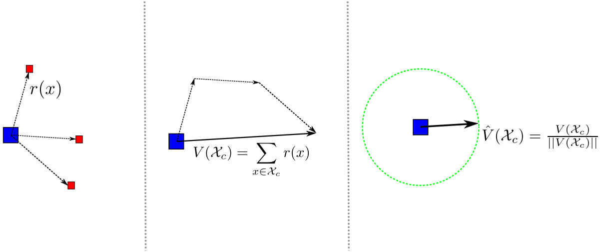
- Similar to [Arandjelovic & Zisserman '13], but:
  - with selectivity function σ_α
  - used with large vocabularies

Aggregated Selective Match Kernel (ASMK)

$$\text{ASMK}(\mathcal{X}_c, \mathcal{Y}_c) = \sigma_\alpha\left(\hat{V}(\mathcal{X}_c)^\top \hat{V}(\mathcal{Y}_c)\right)$$

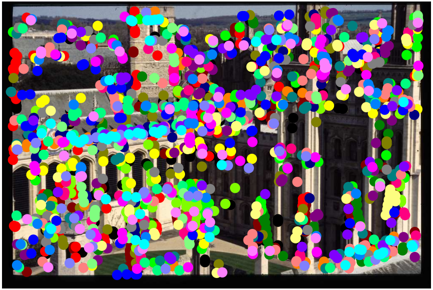
- Cell representation: ℓ_2-normalized aggregated residual

$$\Phi(\mathcal{X}_c) = \hat{V}(\mathcal{X}_c) = V(\mathcal{X}_c)/\|V(\mathcal{X}_c)\|$$



- Similar to [Arandjelovic & Zisserman '13], but:
  - with selectivity function σ_α
  - used with large vocabularies

Aggregated features: $k = 128$ as in VLAD



Aggregated features: $k = 65\text{K}$ as in ASMK

Why to aggregate: burstiness

- Burstiness: non-*iid* statistical behaviour of descriptors
- Matches of bursty features dominate the total similarity score
- Previous work: [Jégou *et al.* '09][Chum & Matas '10][Torii *et al.* '13]

In this work

- Aggregation and normalization per cell handles burstiness
- Keeps a single representative, similar to max-pooling

Why to aggregate: burstiness

- Burstiness: non-*iid* statistical behaviour of descriptors
- Matches of bursty features dominate the total similarity score
- Previous work: [Jégou *et al.* '09][Chum & Matas '10][Torii *et al.* '13]

In this work

- Aggregation and normalization per cell handles burstiness
- Keeps a single representative, similar to max-pooling

Binary counterparts SMK* and ASMK*

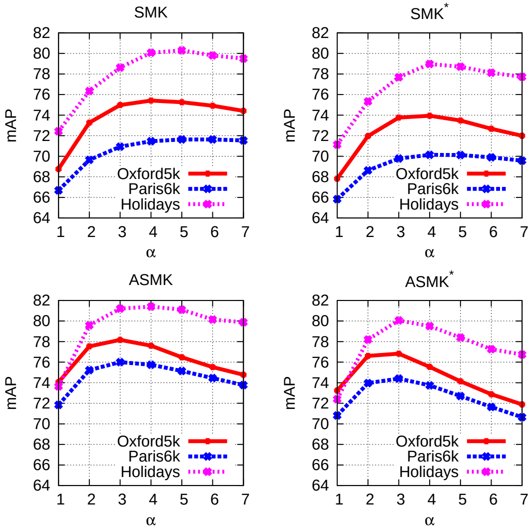
- Full vector representation: high memory cost
- Approximate vector representation: binary vector

$$\text{SMK}^\star(\mathcal{X}_c, \mathcal{Y}_c) = \sum_{x \in \mathcal{X}_c} \sum_{y \in \mathcal{Y}_c} \sigma_\alpha \left\{ \hat{b}(r(x))^\top \hat{b}(r(y)) \right\}$$

$$\text{ASMK}^\star(\mathcal{X}_c, \mathcal{Y}_c) = \sigma_\alpha \left\{ \hat{b} \left(\sum_{x \in \mathcal{X}_c} r(x) \right)^\top \hat{b} \left(\sum_{y \in \mathcal{Y}_c} r(y) \right) \right\}$$

$\hat{b}$ includes centering and rotation as in HE, followed by binarization and ℓ_2-normalization

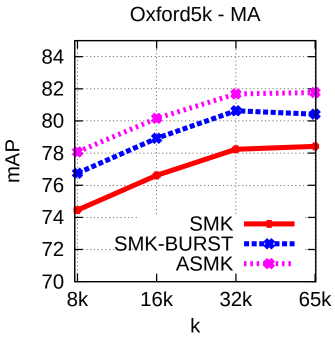
Impact of selectivity



Impact of aggregation

- Improves performance for different vocabulary sizes
- Reduces memory requirements of inverted file

k	memory ratio
8k	$69\,\%$
16k	$78\,\%$
32k	$85\,\%$
65k	$89\,\%$



with $k = 8$k on Oxford5k

- VLAD $\rightarrow 65.5\%$
- SMK $\rightarrow 74.2\%$
- ASMK $\rightarrow 78.1\%$

Comparison to state of the art

Dataset	MA	Oxf5k	Oxf105k	Par6k	Holiday
ASMK*		76.4	69.2	74.4	80.0
ASMK*	×	80.4	75.0	77.0	81.0
ASMK		78.1	-	76.0	81.2
ASMK	×	81.7	-	78.2	82.2
HE [Jégou *et al.* '10]		51.7	-	-	74.5
HE [Jégou *et al.* '10]	×	56.1	-	-	77.5
HE-BURST [Jain *et al.* '10]		64.5	-	-	78.0
HE-BURST [Jain *et al.* '10]	×	67.4	-	-	79.6
Fine vocab. [Mikulík *et al.* '10]	×	74.2	67.4	74.9	74.9
AHE-BURST [Jain *et al.* '10]		66.6	-	-	79.4
AHE-BURST [Jain *et al.* '10]	×	69.8	-	-	81.9
Rep. structures [Torri *et al.* '13]	×	65.6	-	-	74.9

Discussion

- Aggregation is also beneficial with large vocabularies $\rightarrow$ burstiness
- Selectivity always helps (with or without aggregation)
- Descriptor approximation reduces performance only slightly

Part II: Vector quantization and nearest neighbor search

Locally optimized product quantization

Joint work with Yannis Kalantidis, CVPR 2014

Overview

- Problem: given query point $\mathbf{q}$, find its nearest neighbor with respect to Euclidean distance within data set $\mathcal{X}$ in a d-dimensional space
- Focus on large scale: encode (compress) vectors, speed up distance computations
- Fit better underlying distribution with little space & time overhead

Applications

- Retrieval (image as point) [Jégou *et al.* '10][Perronnin *et al.* '10]
- Retrieval (descriptor as point) [Tolias *et al.* '13][Qin *et al.* '13]
- Localization, pose estimation [Sattler *et al.* '12][Li *et al.* '12]
- Classification [Boiman *et al.* '08][McCann & Lowe '12]
- Clustering [Philbin *et al.* '07][Avrithis '13]

Related work

- Indexing
  - Inverted index (image retrieval)
  - Inverted multi-index [Babenko & Lempitsky '12] (nearest neighbor search)
- Encoding and ranking
  - Vector quantization (VQ)
  - Product quantization (PQ) [Jégou *et al.* '11]
  - Optimized product quantization (OPQ) [Ge *et al.* '13] Cartesian k-means [Norouzi & Fleet '13]
  - Locally optimized product quantization (LOPQ) [Kalantidis and Avrithis '14]
- Not discussed
  - Tree-based indexing, e.g., [Muja and Lowe '09]
  - Hashing and binary codes, e.g., [Norouzi *et al.* '12]

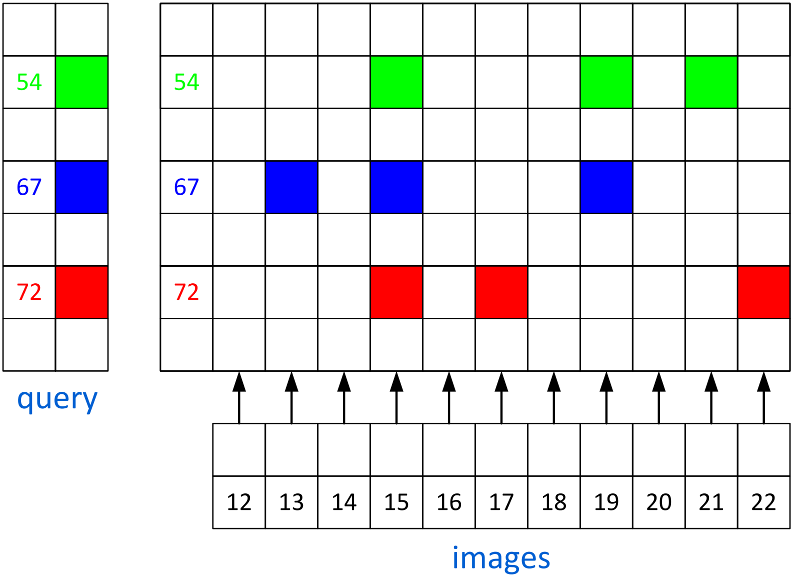
Related work

- Indexing
  - Inverted index (image retrieval)
  - Inverted multi-index [Babenko & Lempitsky '12] (nearest neighbor search)
- Encoding and ranking
  - Vector quantization (VQ)
  - Product quantization (PQ) [Jégou *et al.* '11]
  - Optimized product quantization (OPQ) [Ge *et al.* '13] Cartesian k-means [Norouzi & Fleet '13]
  - Locally optimized product quantization (LOPQ) [Kalantidis and Avrithis '14]
- Not discussed
  - Tree-based indexing, e.g., [Muja and Lowe '09]
  - Hashing and binary codes, e.g., [Norouzi *et al.* '12]

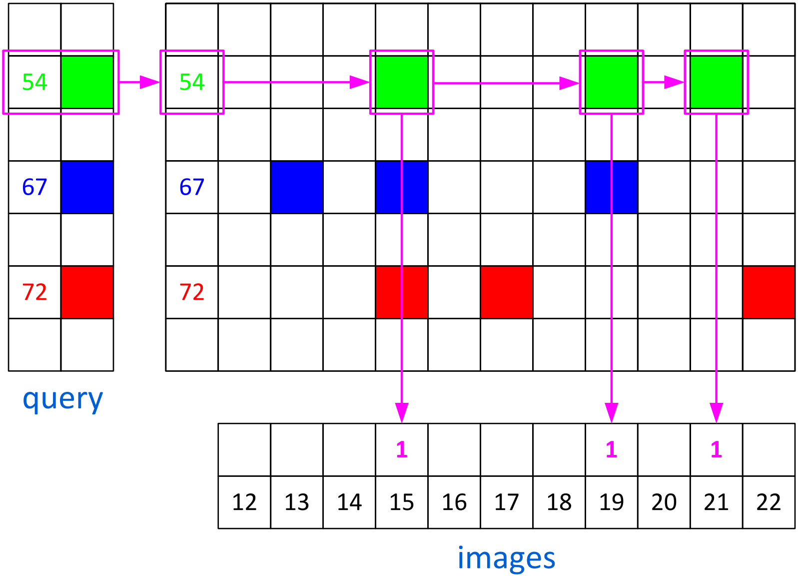
Related work

- Indexing
  - Inverted index (image retrieval)
  - Inverted multi-index [Babenko & Lempitsky '12] (nearest neighbor search)
- Encoding and ranking
  - Vector quantization (VQ)
  - Product quantization (PQ) [Jégou *et al.* '11]
  - Optimized product quantization (OPQ) [Ge *et al.* '13]
    Cartesian k-means [Norouzi & Fleet '13]
  - Locally optimized product quantization (LOPQ) [Kalantidis and Avrithis '14]
- Not discussed
  - Tree-based indexing, e.g., [Muja and Lowe '09]
  - Hashing and binary codes, e.g., [Norouzi *et al.* '12]

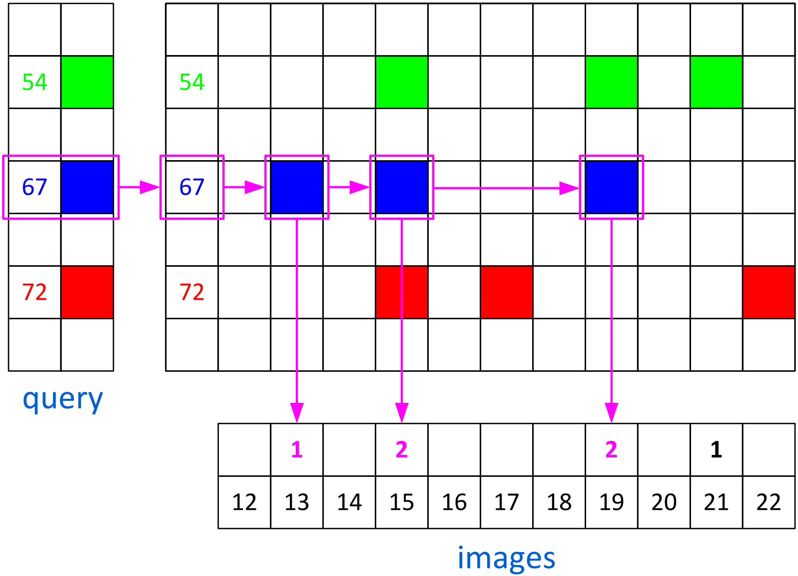
Inverted index



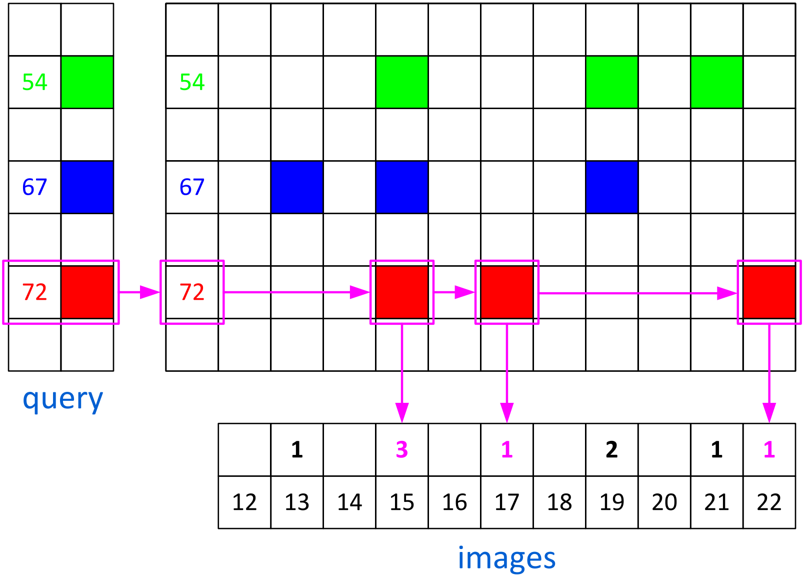
Inverted index



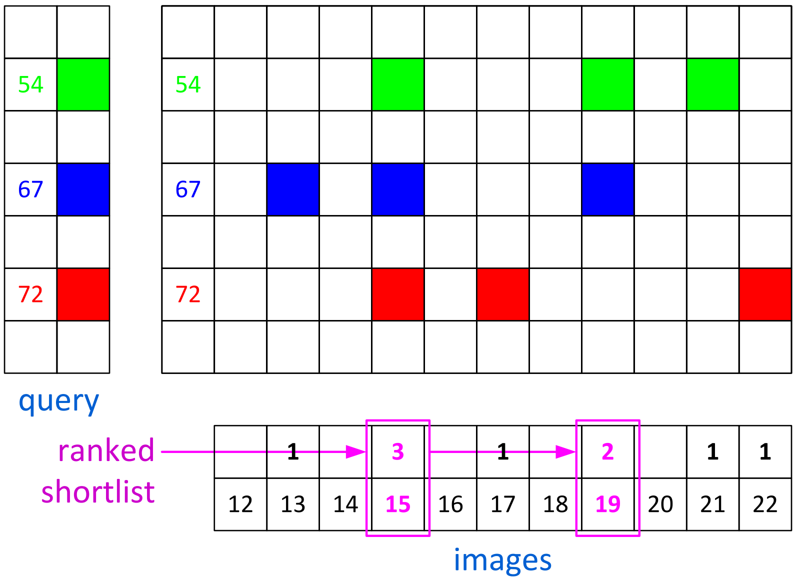
Inverted index



Inverted index



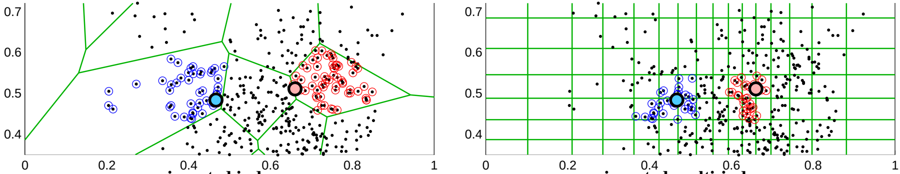
Inverted index



Inverted index—issues

- Are items in a postings list equally important?
- What changes under soft (multiple) assignment?
- How should vectors be encoded for memory efficiency and fast search?

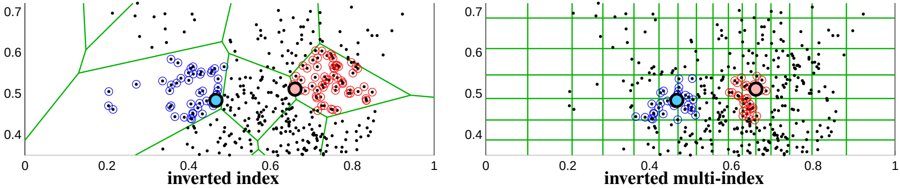
Inverted multi-index



inverted index **inverted multi-index**

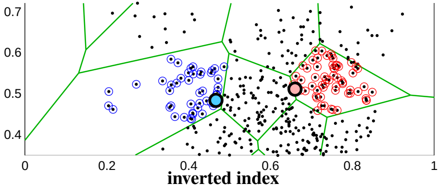
- decompose vectors as $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2)$
- train codebooks $\mathcal{C}^1, \mathcal{C}^2$ from datasets $\{\mathbf{x}_n^1\}, \{\mathbf{x}_n^2\}$
- induced codebook $\mathcal{C}^1 \times \mathcal{C}^2$ gives a finer partition
- given query $\mathbf{q}$, visit cells $(\mathbf{c}^1, \mathbf{c}^2) \in \mathcal{C}^1 \times \mathcal{C}^2$ in ascending order of distance to $\mathbf{q}$ by multi-sequence algorithm

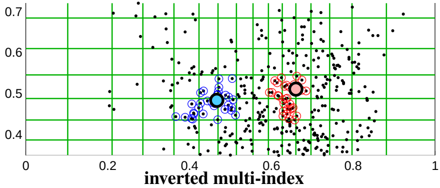
Inverted multi-index



- decompose vectors as $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2)$
- train codebooks $\mathcal{C}^1, \mathcal{C}^2$ from datasets $\{\mathbf{x}_n^1\}, \{\mathbf{x}_n^2\}$
- induced codebook $\mathcal{C}^1 \times \mathcal{C}^2$ gives a finer partition
- given query $\mathbf{q}$, visit cells $(\mathbf{c}^1, \mathbf{c}^2) \in \mathcal{C}^1 \times \mathcal{C}^2$ in ascending order of distance to $\mathbf{q}$ by multi-sequence algorithm

Inverted multi-index





- decompose vectors as $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2)$
- train codebooks $\mathcal{C}^1, \mathcal{C}^2$ from datasets $\{\mathbf{x}_n^1\}, \{\mathbf{x}_n^2\}$
- induced codebook $\mathcal{C}^1 \times \mathcal{C}^2$ gives a finer partition
- given query $\mathbf{q}$, visit cells $(\mathbf{c}^1, \mathbf{c}^2) \in \mathcal{C}^1 \times \mathcal{C}^2$ in ascending order of distance to $\mathbf{q}$ by multi-sequence algorithm

Multi-sequence algorithm

$\mathcal{C}^1 \rightarrow$

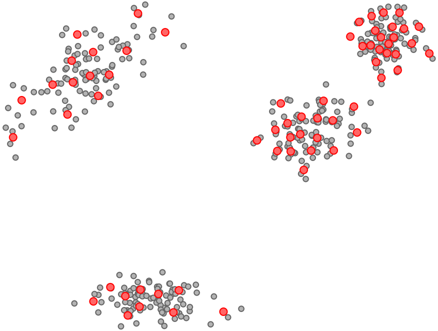
$\mathcal{C}^2 \downarrow$

0.6	0.8	4.1	6.1	8.1	9.1
2.5	2.7	6	8	10	11
3.5	3.7	7	9	11	12
6.5	6.7	10	12	14	15
7.5	7.7	11	13	15	16
11.5	11.7	15	17	19	20

Vector quantization (VQ)

$$\text{minimize} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 = \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - q(\mathbf{x})\|^2 = E(\mathcal{C})$$

dataset codebook quantizer distortion



Vector quantization (VQ)

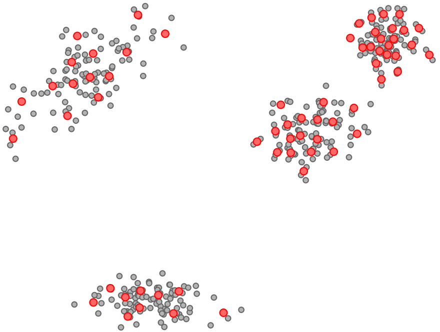
$$\text{minimize} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 = \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - q(\mathbf{x})\|^2 = E(\mathcal{C})$$

dataset

codebook

quantizer

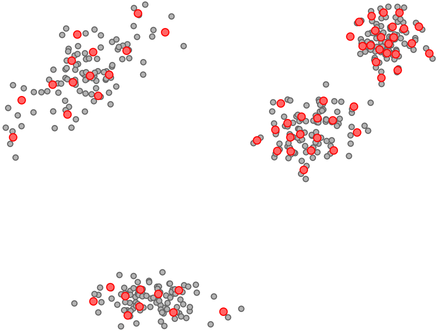
distortion



Vector quantization (VQ)

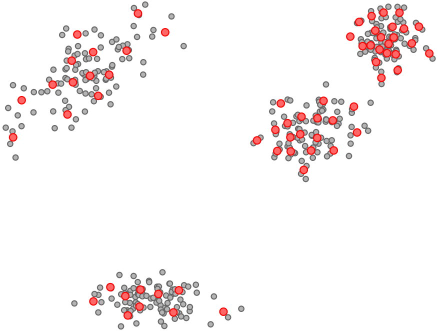
$$\text{minimize} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 = \sum_{\mathbf{x} \in \mathcal{X}} \|\mathbf{x} - q(\mathbf{x})\|^2 = E(\mathcal{C})$$

dataset codebook quantizer distortion



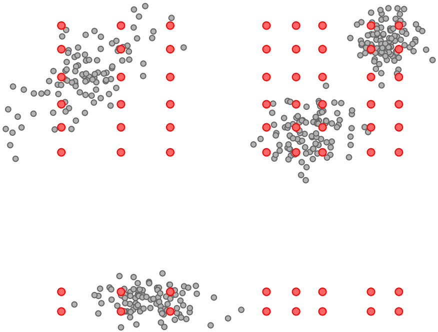
Vector quantization (VQ)

- For small distortion $\rightarrow$ large $k = |\mathcal{C}|$:
  - hard to train
  - too large to store
  - too slow to search



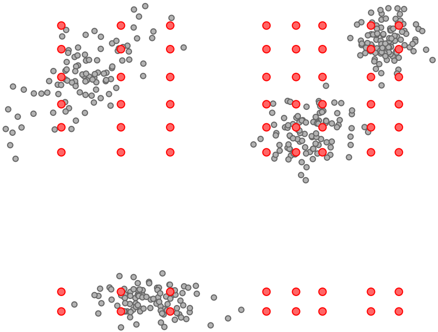
Product quantization (PQ)

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 \\ \text{subject to} \quad & \mathcal{C} = \mathcal{C}^1 \times \cdots \times \mathcal{C}^m \end{aligned}$$



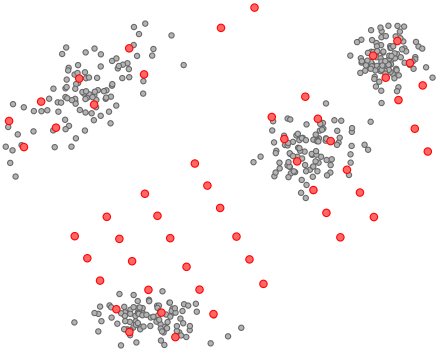
Product quantization (PQ)

- train: $q = (q^1, \dots, q^m)$ where $q^1, \dots, q^m$ obtained by VQ
- store: $|\mathcal{C}| = k^m$ with $|\mathcal{C}^1| = \cdots = |\mathcal{C}^m| = k$
- search: $\|\mathbf{y} - q(\mathbf{x})\|^2 = \sum_{j=1}^{m} \|\mathbf{y}^j - q^j(\mathbf{x}^j)\|^2$ where $q^j(\mathbf{x}^j) \in \mathcal{C}^j$



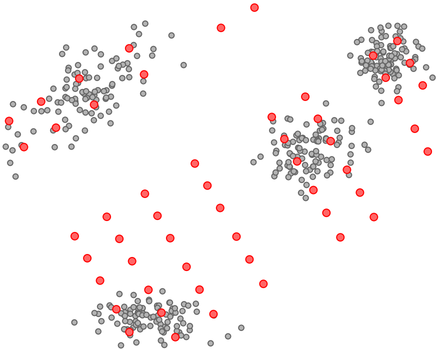
Optimized product quantization (OPQ)

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{x} \in \mathcal{X}} \min_{\hat{\mathbf{c}} \in \hat{\mathcal{C}}} \|\mathbf{x} - R\hat{\mathbf{c}}\|^2 \\ \text{subject to} \quad & \hat{\mathcal{C}} = \mathcal{C}^1 \times \cdots \times \mathcal{C}^m \\ & R^\top R = I \end{aligned}$$



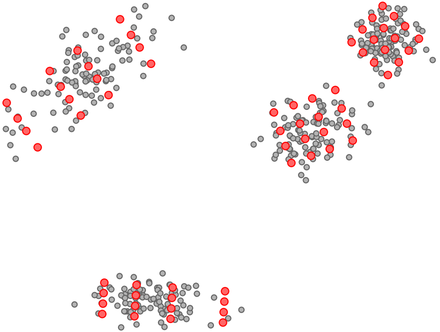
OPQ, parametric solution for $\mathcal{X} \sim \mathcal{N}(\mathbf{0}, \Sigma)$

- independence: PCA-align by diagonalizing Σ as $U \Lambda U^\top$
- balanced variance: permute Λ such that $\prod_i \lambda_i$ is constant in each subspace; $R \leftarrow U P_\pi^\top$
- find $\hat{\mathcal{C}}$ by PQ on rotated data $\hat{\mathbf{x}} = R^\top \mathbf{x}$



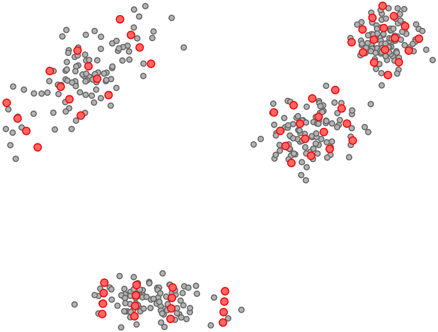
Locally optimized product quantization (LOPQ)

- compute residuals $r(\mathbf{x}) = \mathbf{x} - q(\mathbf{x})$ on coarse quantizer q
- collect residuals $\mathcal{Z}_i = \{r(\mathbf{x}) : q(\mathbf{x}) = \mathbf{c}_i\}$ per cell
- train $(R_i, q_i) \leftarrow \mathrm{OPQ}(\mathcal{Z}_i)$ per cell

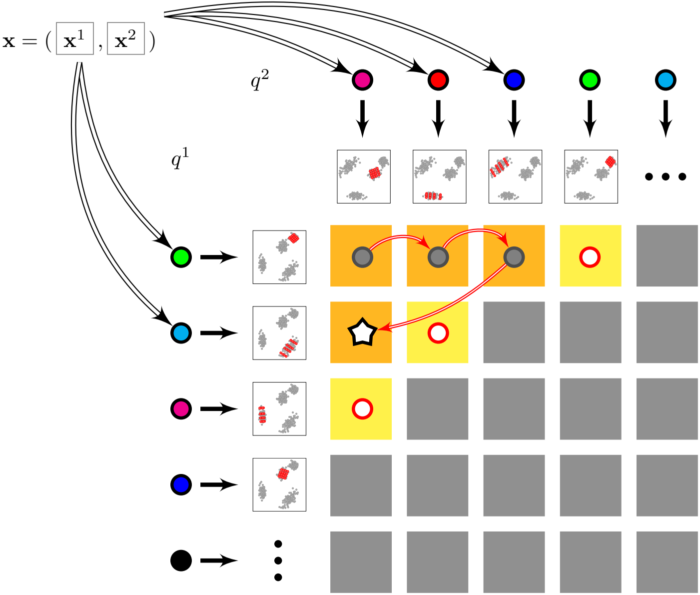


Locally optimized product quantization (LOPQ)

- better capture support of data distribution, like local PCA [Kambhatla & Leen '97]
  - multimodal (e.g. mixture) distributions
  - distributions on nonlinear manifolds
- residual distributions closer to Gaussian assumption



Multi-LOPQ



Comparison to state of the art

SIFT1B, 64-bit codes

Method	$R = 1$	$R = 10$	$R = 100$
Ck-means [Norouzi & Fleet '13]	–	–	0.649
IVFADC	0.106	0.379	0.748
IVFADC [Jégou *et al.* '11]	0.088	0.372	0.733
OPQ	0.114	0.399	0.777
Multi-D-ADC [Babenko & Lempitsky '12]	0.165	0.517	0.860
LOR+PQ	0.183	0.565	0.889
LOPQ	0.199	0.586	0.909

Most benefit comes from locally optimized rotation!

Comparison to state of the art

SIFT1B, 64-bit codes

Method	$R = 1$	$R = 10$	$R = 100$
Ck-means [Norouzi & Fleet '13]	–	–	0.649
IVFADC	0.106	0.379	0.748
IVFADC [Jégou *et al.* '11]	0.088	0.372	0.733
OPQ	0.114	0.399	0.777
Multi-D-ADC [Babenko & Lempitsky '12]	0.165	0.517	0.860
LOR+PQ	0.183	0.565	0.889
LOPQ	0.199	0.586	0.909

Most benefit comes from locally optimized rotation!

Comparison to state of the art

SIFT1B, 128-bit codes

T	Method	$R = 1$	10	100
20K	IVFADC+R [Jégou *et al.* '11]	0.262	0.701	0.962
	LOPQ+R	0.350	0.820	0.978
10K	Multi-D-ADC [Babenko & Lempitsky '12]	0.304	0.665	0.740
	OMulti-D-OADC [Ge *et al.* '13]	0.345	0.725	0.794
	Multi-LOPQ	0.430	0.761	0.782
30K	Multi-D-ADC [Babenko & Lempitsky '12]	0.328	0.757	0.885
	OMulti-D-OADC [Ge *et al.* '13]	0.366	0.807	0.913
	Multi-LOPQ	0.463	0.865	0.905
100K	Multi-D-ADC [Babenko & Lempitsky '12]	0.334	0.793	0.959
	OMulti-D-OADC [Ge *et al.* '13]	0.373	0.841	0.973
	Multi-LOPQ	0.476	0.919	0.973

Residual encoding in related work

- PQ (IVFADC) [Jégou *et al.* '11]: single product quantizer for all cells
- [Uchida *et al.* '12]: multiple product quantizers shared by multiple cells
- OPQ [Ge *et al.* '13]: single product quantizer for all cells, globally optimized for rotation (single/multi-index)
- LOPQ: with/without one product quantizer per cell, with/without rotation optimization per cell (single/multi-index)
- [Babenko & Lempitsky '14]: one product quantizer per cell, optimized for rotation per cell (multi-index)

Residual encoding in related work

- PQ (IVFADC) [Jégou *et al.* '11]: single product quantizer for all cells
- [Uchida *et al.* '12]: multiple product quantizers shared by multiple cells
- OPQ [Ge *et al.* '13]: single product quantizer for all cells, globally optimized for rotation (single/multi-index)
- LOPQ: with/without one product quantizer per cell, with/without rotation optimization per cell (single/multi-index)
- [Babenko & Lempitsky '14]: one product quantizer per cell, optimized for rotation per cell (multi-index)

http://image.ntua.gr/iva/research/

Thank you!