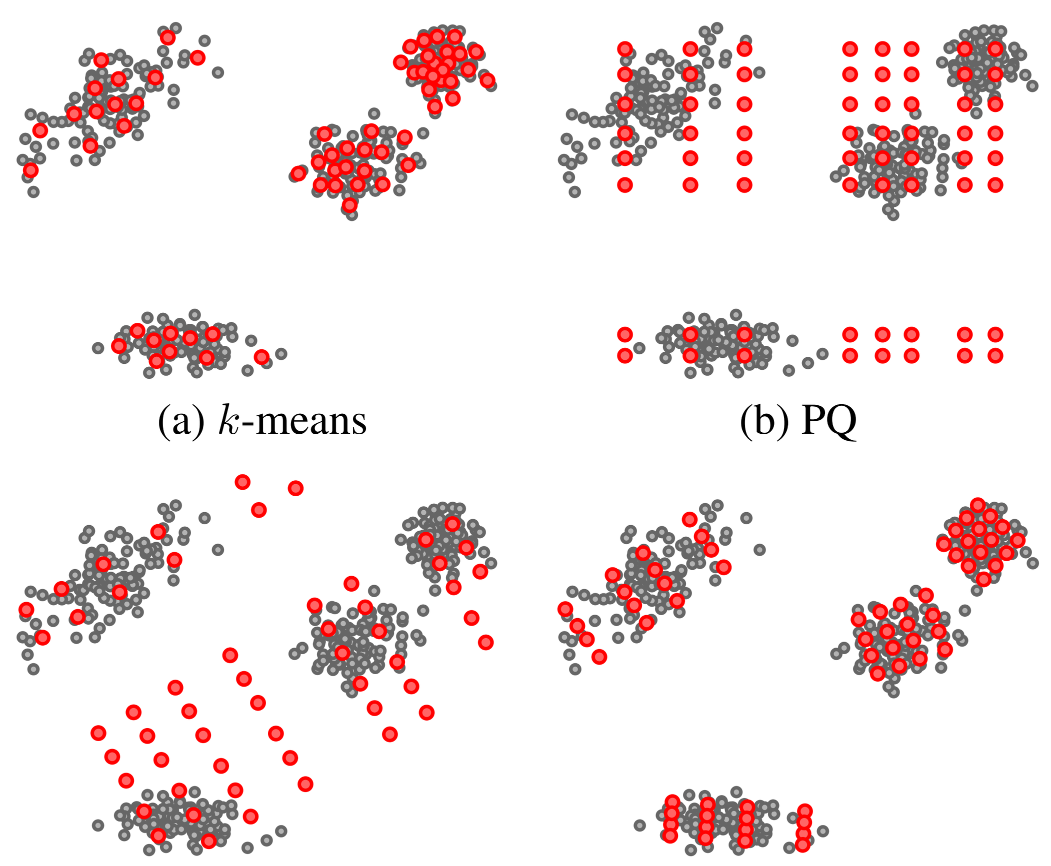
Locally Optimized Product Quantization for Approximate Nearest Neighbor Search

Yannis Kalantidis and Yannis Avrithis
National Technical University of Athens

Background

- **Vector quantization:** Minimize distortion $E = \sum_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x} - q(\mathbf{x})\|^2$, where quantizer $q : \mathbf{x} \mapsto q(\mathbf{x}) = \arg\min_{\mathbf{c}\in\mathcal{C}} \|\mathbf{x} - \mathbf{c}\|$.
- **Product quantization** [1]: $\mathcal{C} = \mathcal{C}^1 \times \cdots \times \mathcal{C}^m$, *i.e.*: k^m centroids of the form $\mathbf{c} = (\mathbf{c}^1, \ldots, \mathbf{c}^m)$ with each sub-centroid $\mathbf{c}^j \in \mathcal{C}^j$ for $j \in \mathcal{M} = \{1, \ldots, m\}$. m independent sub-problems: $q(\mathbf{x}) = (q^1(\mathbf{x}^1), \ldots, q^m(\mathbf{x}^m))$.
- **Optimized product quantization** [2]: $\mathcal{C} = \{R\hat{\mathbf{c}} : \hat{\mathbf{c}} \in \mathcal{C}^1 \times \cdots \times \mathcal{C}^m, R^{\mathrm{T}}R = I\}$, where orthogonal $d \times d$ matrix R optimized for subspace decomposition (rotation + permutation).

Overview



(a) k-means (b) PQ (c) OPQ (d) LOPQ

Contribution

- **Locality:** Partitioning data in cells with a coarse quantizer of K cells, we *locally optimize* one product quantizer per cell on the residual distribution.
- **Efficient training:** Local distributions are easier to optimize via a simple OPQ variant.
- **Multiple search frameworks:** Fits naturally to either a *single* or a *multi-index* [3].
- **Product optimization:** For an n^{th}-order multi-index, we only optimize nK product quantizers for a total of K^n cells.

References

[1] Jegou *et al.*. Product quantization for nearest neighbor search. PAMI, 2011.
[2] Ge *et al.*. Optimized product quantization for approximate nearest neighbor search. CVPR, 2013.
[3] Babenko and Lempitsky. The inverted multi-index. CVPR, 2012.

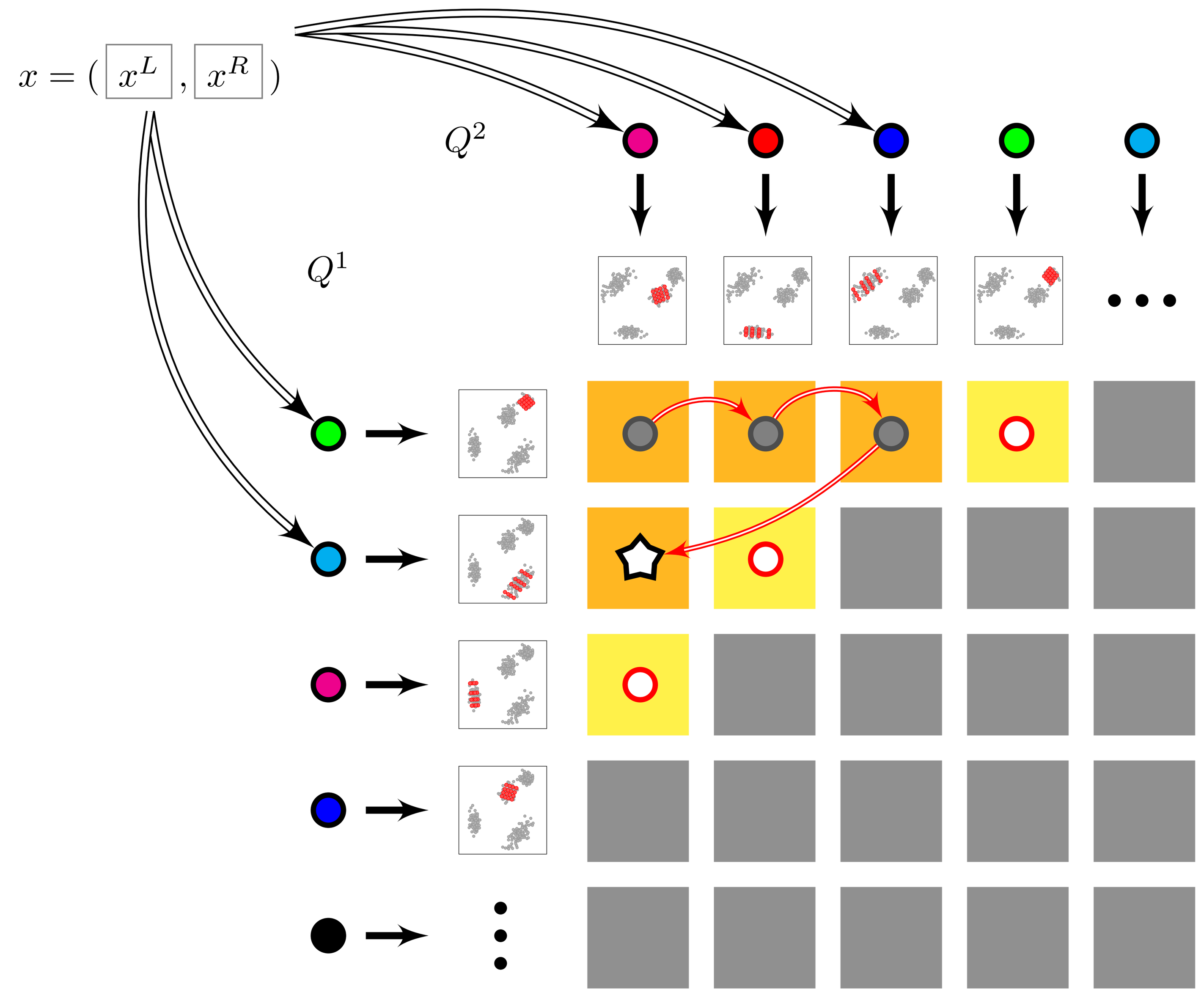
Local Optimization

- If $\mathcal{Z}$ is the set of residuals of data points quantized to some cell and $|\mathcal{C}^j| = k$ for $j \in \mathcal{M} = \{1, \ldots, m\}$, we locally optimize both space decomposition and sub-quantizers per cell, using OPQ:

$$\begin{aligned} \text{minimize} \quad & \sum_{\mathbf{z}\in\mathcal{Z}} \min_{\hat{\mathbf{c}}\in\hat{\mathcal{C}}} \|\mathbf{z} - R\hat{\mathbf{c}}\|^2 \\ \text{subject to} \quad & \hat{\mathcal{C}} = \mathcal{C}^1 \times \cdots \times \mathcal{C}^m \\ & R^{\mathrm{T}}R = I, \end{aligned} \tag{1}$$

- **Parametric solution:** Assuming a normal distribution, minimize the theoretical lower distortion bound as a function of R alone via PCA alignment and *eigenvalue allocation*. Sub-quantizer optimization follows as in PQ.
- Residual distributions are closer to normal, so parametric solution fits better to LOPQ.

Multi-LOPQ

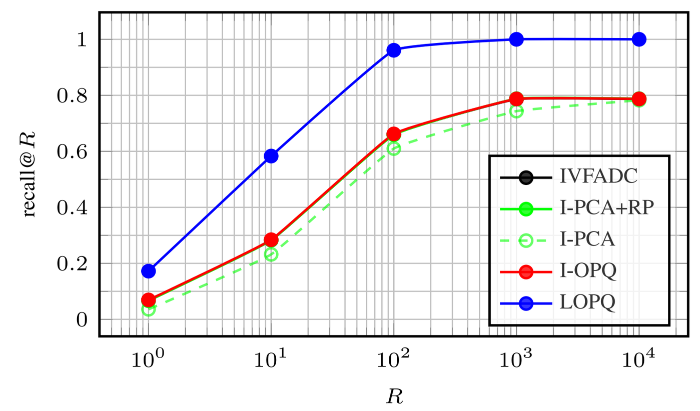


Indexing and Search

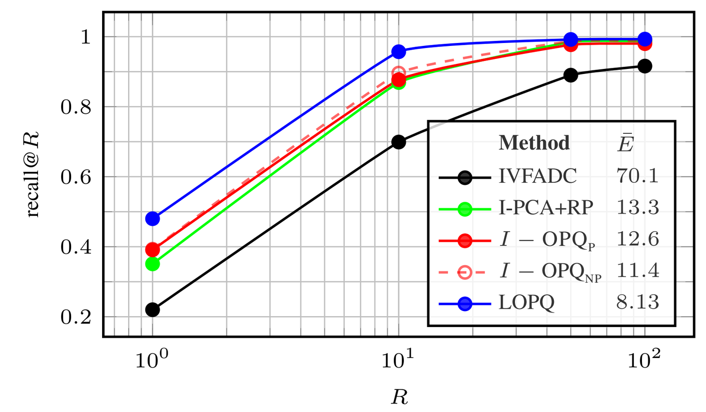
- **Single index:** For each of K cells, residual is individually rotated and encoded. The query point is soft-assigned to its w nearest cells. Asymmetric distances are computed exhaustively via lookup tables.
- **Multi-index:** Two subspace quantizers Q^1, Q^2 of K centroids each are built. Residuals are encoded per row and column: $2K$ local rotations and sub-quantizers for a total of K^2 cells. Search follows *multi-sequence* algorithm, with lazy evaluation of row/column-rotated query residuals.

Experiments

- **Protocol:** We measure Recall@R, we set $k = 256$ in all cases and $m = 8$ unless otherwise stated.
- **Multi-modal datasets:** SYNTH1M (synthetic, 1M 128-dimensional vectors from 1K anisotropic Gaussians), MNIST.

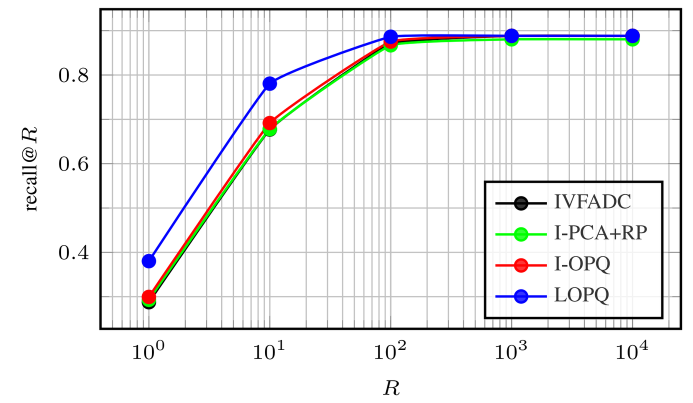


SYNTH1M ($K = 1024$, $w = 8$)

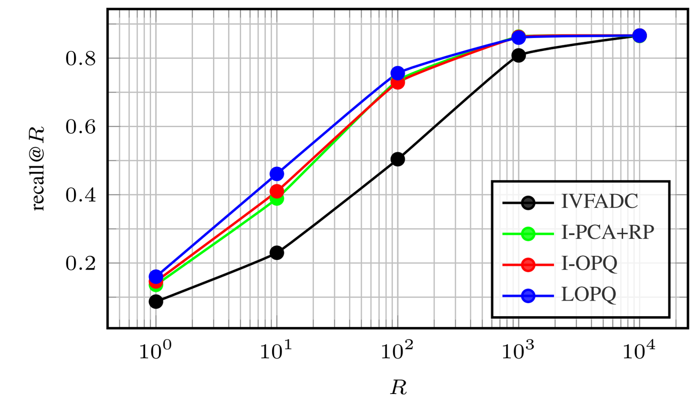


MNIST ($K = 64$, $w = 8$)

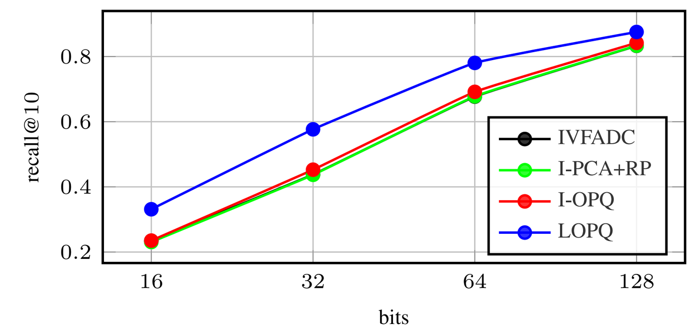
- **Common datasets:** SIFT1M, GIST1M.



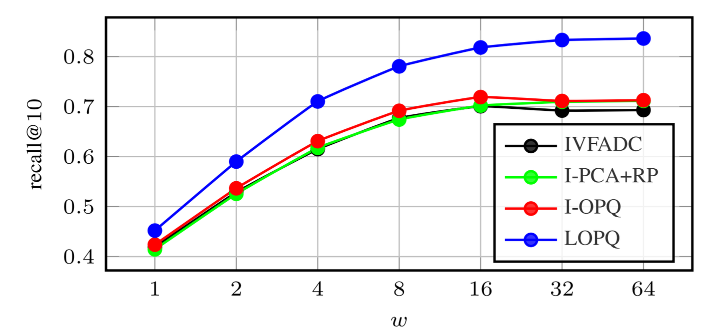
SIFT1M ($K = 1024$, $w = 8$)



GIST1M ($K = 1024$, $w = 16$)



SIFT1M - bit allocation ($K = 1024$, $w = 8$)



SIFT1M - soft assignment ($K = 1024$, $m = 8$)

- **Billion-scale experiments: SIFT1B**

T	Method	$R = 1$	10	100
	Multi-I-Hashing [16]	–	–	0.420
	KLSH-ADC [17]	–	–	0.894
	Joint-ADC [19]	–	–	0.938
20K	IVFADC+R [13]	0.262	0.701	0.962
	LOPQ+R	**0.350**	**0.820**	**0.978**
10K	Multi-D-ADC [3]	0.304	0.665	0.740
	OMulti-D-OADC [8]	0.345	0.725	**0.794**
	Multi-LOPQ	**0.430**	**0.761**	0.782
30K	Multi-D-ADC [3]	0.328	0.757	0.885
	OMulti-D-OADC [8]	0.366	0.807	**0.913**
	Multi-LOPQ	**0.463**	**0.865**	0.905
100K	Multi-D-ADC [3]	0.334	0.793	0.959
	OMulti-D-OADC [8]	0.373	0.841	**0.973**
	Multi-LOPQ	**0.476**	**0.919**	**0.973**

SIFT1B with 128-bit codes and $K = 2^{13} = 8192$ (resp. $K = 2^{14}$) for single index (resp. multi-index). For IVFADC+R and LOPQ+R, $m' = 8$, $w = 64$.

Method	$R = 1$	$R = 10$	$R = 100$
Ck-means [15]	–	–	0.649
IVFADC	0.106	0.379	0.748
IVFADC [13]	0.088	0.372	0.733
I-OPQ	0.114	0.399	0.777
Multi-D-ADC [3]	0.165	0.517	0.860
LOR+PQ	0.183	0.565	0.889
LOPQ	**0.199**	**0.586**	**0.909**

SIFT1B with 64-bit codes, $K = 2^{13} = 8192$ and $w = 64$. For Multi-D-ADC, $K = 2^{14}$ and $T = 100$K.

- **Overhead** on top of IVFADC (resp. Multi-D-ADC):
  - **Space**: Kd^2 (resp. $2K(d/2)^2$) for rotation matrices, *i.e.* around 500MB on SIFT1B and Kdk for sub-quantizers, *i.e.* 2GB on top of 21GB for SIFT1B.
  - **Query time:** The time needed to rotate the query for each soft-assigned cell (row/column). Average overhead on SIFT1B for Multi-LOPQ is 0.776, 1.92, 4.04ms respectively for $T = 10$K, 30K, 100K.

Project page: http://image.ntua.gr/iva/research/lopq/
Contact: {ykalant, iavr}@image.ntua.gr