Product quantization for vector retrieval with no error

Andrzej Wichert
Department of Informatics, INESC-ID / IST - Technical University of Lisboa, Portugal
andreas.wichert@ist.utl.pt

Keywords: high dimensional indexing, multi-resolution, quantization, vector data bases

Abstract: We propose a coding mechanism for less costly exact vector retrieval for data bases representing vectors. The search starts at the subspace with the lowest dimension. In this subspace, the set of all possible similar vectors is determined. In the next subspace, additional metric information corresponding to a higher dimension is used to reduce this set. We demonstrate the method performing experiments on image retrieval on one thousand gray images of the size 128×96. Our model is twelve times less complex than a list matching.

1 INTRODUCTION

In this paper we propose a hierarchical product quantization for less costly vector retrieval. The space in which a vector is represented is decomposed into low dimensional subspaces and quantize each subspace separately. Each subspace corresponds to a subvector described by a mask. A global feature corresponds to a subvector of a higher dimension, a local feature to a subvector of a lower dimension. During similarity-based vector retrieval, the search starts from the images represented by global features. In this representation, the set of all possible similar images is determined. In the next stage, additional information corresponding to the representation of more local feature is used to reduce this set. This procedure is repeated until the similar vectors can be determined. The described method represents a vector indexing method that speeds up the search considerably and does not suffer from the curse of dimensionality. The method is related to the subspace trees (Wichert et al., 2010).

We describe a mathematical model of the hierarchical product quantization. The paper is organized as follows:

- We show why the search with the product quantization is an improvement to simple quantization due to the curse of dimensionality.
- We introduce a new indexing method, called hierarchical product quantization and preform experiments on image retrieval on one thousand gray images of the size 128×96 resulting in vectors of dimension 12288.

2 HIERARCHICAL PRODUCT QUANTIZATION

2.1 Euclidean ε-similarity

Two vectors $\vec{x}$ and $\vec{y}$ are similar if their Euclidian distance is smaller or equal to ε, $d(\vec{x},\vec{y}) \leq \varepsilon$. Let DB be a database of s vectors $\vec{x}^{(i)}$ dimension m in which the index i is an explicit key identifying each vector,

$$\{\vec{x}^{(i)} \in DB | i \in \{1..s\}\}. \tag{1}$$

The set DB can be ordered according to a given query vector $\vec{y}$ using an Euclidian distance function d. This is done by a monotone increasing sequence corresponding to the increasing distance of $\vec{y}$ to $\vec{x}^{(i)}$ with an explicit key that identifies each vector indicated by the index i,

$$d[y]_t := \{d(x^{(i_n)},y)_t \mid \forall t \in \{1..s\}, \forall i_t \in \{1..s\} :$$
$$d(x^{(i_1)},y)_1 \leq d(x^{(i_2)},y)_2$$
$$\leq ... \leq d(x^{(i_n)},y)_t .... \leq d(x^{(i_s)},y)_s\} \tag{2}$$

if $\vec{y} \in DB$, then $d[y]_1 := 0$. The set of similar vectors in correspondence to $\vec{y}$, $DB[y]_\varepsilon$, is the subset of DB, $DB[y]_\varepsilon \subseteq DB$ with size $\sigma = |DB[y]_\varepsilon|$, $\sigma \leq s$:

$$DB[y]_\varepsilon := \{x^{(i)} \in DB \mid d[y]_t = d(x^{(i)},y) \leq \varepsilon\}. \tag{3}$$

2.2 Related work

Could we take advantage of the grouping of the vectors into clusters?

The idea would be to determine the most similar cluster center which represents the most similar category. In the next step we would search for the most similar vectors $DB[y]_\varepsilon$ only in this cluster. By doing so we could save some considerable computation. Such a structure can be simply modeled by a clustering algorithm, as for example k-means. We group the images into clusters represented by the cluster centers c_j. After the clustering cluster centers $c_1, c_2, c_3, ..., c_k$ with clusters $C_1, C_2, C_3, ..., C_k$ are present with:

$$C_j = \{x | d(x, c_j) = \min_i d(x, c_i)\} \tag{4}$$

$$c_j = \{\frac{1}{|C_j|} \sum_{x \in C_j} x\}. \tag{5}$$

Suppose $s = \min_i d(y, c_i)$ is the distance to the closest cluster center and r_{max} the maximal radius of all the clusters. Only if $s \geq \varepsilon \geq r_{max}$ we are guaranteed to determine $DB[y]_\varepsilon$. Otherwise we have to analyze other clusters as well. When a cluster with a minimum distance s was determined, we know that the images in this cluster have the distance between $s + r_{max}$ and $s - r_{max}$. Because of that we have to analyze additionally all the clusters with $\{\forall i | d(y, c_i) < (s + r_{max})\}$. It means that in the worst case we have to analyze all the clusters. The worst case is present when the dimension of the images is high. High dimensional spaces (like for example dimensions > 100) have negative implications on the number of clusters we have to analyze. These negative effects are named as the "curse of dimensionality." Most problems arise from the fact that the volume of a sphere with the constant radius grows exponentially with increasing dimension.

Could hierarchical clustering overcome those problems? Indeed, traditional indexing methods are based on the principle of hierarchical clustering of the data space, in which metric properties are used to build a tree that then can be used to prune branches while processing the queries. Traditional indexing trees can be described by two classes, trees derived from the kd-tree and the trees composed by derivatives of the R-tree. Trees in the first class divides the data space along predefined hyper-planes regardless of data distribution. The resulting regions are mutually disjoint and most of them do not represent any objects. In fact with the growing dimension of space we would require exponential many objects to fill the space. The second class tries to overcome this problem by dividing the data space according to the data distribution into overlapping regions, as described in the second section. An example of the second class is the M-tree (Paolo Ciaccia, 1997). It performs exact retrieval with 10 dimensions. However its performance deteriorates in high dimensional spaces. Most indexing methods operate efficiently only when the number of dimensions is small (< 10). The growth in the number of dimensions has negative implications in the performance; these negative effects are also known as the "curse of dimensionality."

A solution to this problem consists of approximate queries which allow a relative error during retrieval. M-tree (Ciaccia and Patella, 2002) and A-tree (Sakurai et al., 2002) with approximate queries perform retrieval in dimensions of several hundreds. A-tree uses approximated MBR instead of a the MBR of the R-tree. Approximate metric trees like NV-trees (Olafsson et al., 2008), locality sensitive hashing (LSH) (Andoni et al., 2006) or product quantization for nearest neighbor search (Jegou et al., 2011) work with an acceptable error up to dimension 1000.

We introduce hierarchical product quantizer, who preforms exact vector queries in high dimensions. The hierarchical product quantizer model is related to the subspace-tree. In a subspace-tree instead of quamtizing the subvectors defined by masks the mean value is computed (Wichert, 2008), (Wichert et al., 2010). Mathematical methods and tools that were developed for the analysis of the subspace tree, like the correct estimation of ε (Wichert, 2008) and the algorithmic complexity (Wichert, 2008) can be as well applied for the hierarchical quantization method. Hierarchical quantization for image retrieval was first proposed by (Wichert, 2009).

2.3 Searching with product quantizers

The vector $\vec{x}$ of dimension m is split into f distinct subvectors of dimension $p = dim(m/f)$. The subvectors are quantizied using f quantiziers:

$$\vec{x} = \underbrace{x_1, , x_2, \cdots, x_p}_{u_1(\vec{x})}, \cdots, \underbrace{x_{m-p+1}, \cdots, x_m}_{u_f(\vec{x})} \tag{6}$$

$$u_t(x) \in x | t \in \{1..f\}$$

We group the subvectors of dimension $p = dim(m/f)$ into clusters represented by the cluster centers c_j of dimension p. After the clustering cluster centers $c_1, c_2, c_3, ..., c_k$ with clusters $C_1, C_2, C_3, ..., C_k$ are present with

$$C_j = \{u_t(x) | d(u_t(x), c_j) = \min_i d(u_t(x), c_i)\} \tag{7}$$

$$c_j = \{\frac{1}{|C_j|} \sum_{u_t(x) \in C_j} u_t(x)\}. \tag{8}$$

We assume that all subquantuzers have the same number k of clusters. To a query vector y we determine the most similar vector x of the database using the qunatizied codes and the Euclidean distance function d.

$$d(U(\vec{x}),U(\vec{y})) = \sqrt{\sum_{t=1}^{f} d(u_t(\vec{x}),(u_t(\vec{y}))^2}$$

$$d(U(\vec{x}),U(\vec{y})) = \sqrt{\sum_{t=1}^{f} d(c_{t(x)},c_{t(y)})^2} \quad (9)$$

We represent vectors by the corresponding cluster centers:

$$U(\vec{x}) = \underbrace{c_{i1},c_{i2},\cdots,c_{ip}}_{u_1(\vec{x})=c_{1(x)}=c_i},\cdots,\underbrace{c_{j1},\cdots,c_{jp}}_{u_f(\vec{x})=c_{f(x)}=c_j} \quad (10)$$

By using $d(U(\vec{x}),U(\vec{y}))$ instead of $d(\vec{x},\vec{y})$ an estimation *error* is produced:

$$d(U(\vec{x}),U(\vec{y})) + error = d(\vec{x},\vec{y}) \quad (11)$$

To speed up the computation of $d(U(\vec{x}),U(\vec{y}))$ all the possible $d(c_{t(x)},c_{t(y)})^2$ are pre-computed and stored in a look-up table. The size of the look-up table depends on the number k, it is k^2. The bigger the value of k, the slower the computations due to the size of the look-up table. However the bigger the value of k the smaller is the estimation error. To determine ε similar vectors according to the Euclidean distance to a given query y, we have to compute $d(\vec{x},\vec{y})$ for all vectors x out of the database. If the distances computed by the quantizied product $d(U(\vec{x}),U(\vec{y}))$ are smaller or equal than the distances in the original space $d(\vec{x},\vec{y})$, a lower bound which is valid in both spaces can be determined. The distance of similar objects is smaller or equal to ε in the original space and, consequently, it is smaller or equal to ε in the quantizied product as well. The use of a lower bound between different spaces was first suggested by
(Faloutsos et al., 1994), (Faloutsos, 1999). Because of the estimation error the lower bound is only valid for a certain ω value:

$$d(U(\vec{x}),U(\vec{y})) - \omega \leq d(\vec{x},\vec{y}). \quad (12)$$

How can we estimate the ω value for all $\{\vec{x}^{(i)} \in DB | i \in \{1..s\}\}$? Suppose $s = \min_i d(u_t(y),c_i)$ is the distance to the closest cluster center and r_{max} the maximal radius of all the clusters. That means that in the worst case we have d to subtract r_{max} for each subvector before computing the Euclidian distance function, $\omega = k \times r_{max}$.

If we compute the Euclidian distance between all vectors and $\{\vec{x}^{(i)} \in DB | i \in \{1..s\}\}$ compare it to the Euclidian distance between$\{U(\vec{x})^{(i)}$, we find that $\omega << k \times r_{max}$.

We can estimate the ω value by computing the Euclidian distance between a a random sample of vectors and their product quantizier representation. If the lower bound is satisfied with the correct value ω, all vectors at a distance lower than ε in the original space are also at a lower distance in the product quantizier representation. The distance of some that are above ε in the original space may be below ε in the product quantizier representation. This vectors are called false hits. The false hits are separated from the selected objects through comparison in the original space.

The set of ε similar vectors in correspondence to a query vector $\vec{y}$ is computed in two steps. In the first step the set of possible candidates is determined using product quantizier representation. The speed results from the usage of the look-up table. In the second step the false hits are separated from the selected objects through comparison in the original space. A saving compared to a simple list matching is achieved if the set of possible candidates is sufficiently small in comparison to the size of database. An even greater saving can be achieved, if one applies this method hierarchically.

2.4 Searching with hierarchical product quantizers

We apply the product quantizier recursively. The vector $\vec{x}$ of dimension m is split into f distinct subvectors of dimension $p = dim(m/f)$. The subvectors are quantizied using f quantiziers, the resulting quantizied vectror are quantizied using e quantiziers with $g = dim(m/e)$ and $f > e$

$$\vec{x} = \underbrace{\underbrace{x_1,x_2,\cdots,x_p}_{u1_1(\vec{x})},\cdots,\cdots,\cdots,\underbrace{x_{m-p+1},\cdots,x_m}_{u1_f(\vec{x})}}_{u2_1(U(\vec{x}))\qquad\qquad u2_e(U(\vec{x}))} \quad (13)$$

with following hierarchical representation,

$$\begin{aligned} &U1(\vec{x}) = U(\vec{x}) = c_{i1},c_{i2},\cdots,c_{ip},\cdots,c_{j1},\cdots,c_{jp} \\ &U2(\vec{x}) = U(U1(\vec{x})) = c_{i1},c_{i2},\cdots,c_{ig},\cdots,c_{j1},\cdots,c_{jg} \\ &\cdots \\ &Un(\vec{x}) = U(U(n-1)(\vec{x})) = c_{i1},c_{i2},\cdots,c_{il},\cdots \\ &\cdots,c_{j1},\cdots,c_{jl} \end{aligned} \quad (14)$$

and

$$d^*(Uk(\vec{x}),Uk(\vec{y}) = d(Uk(\vec{x}),Uk(\vec{y})) - \omega_k \leq d(\vec{x},\vec{y}), \quad (15)$$

$$k \in \{1..n\}$$

The DB is mapped by the first product quantizers $U1$ into $U1(DB)$, by the k th quantizers Uk into $Uk(DB)$. The set $Uk(DB)$ can be ordered according to a given query vector $Uk(\vec{y})$ using an Euclidian distance function with ω_k as explained before

$$d[Uk(y)]_t := \{d^*(Uk(x^{(i)}), Uk(y)) \mid \forall t \in \{1..s\} : d^*[Uk(y)]_t \leq d^*[Uk(y)]_{t+1}\}$$

for a certain ε value,

$$Uk(DB[y])_\varepsilon := \{Uk(x)_t^{(i)} \in Uk(DB) \mid d[Uk(y)]_t = d^*(Uk(x)^{(i)}, Uk(y)) \leq \varepsilon\}$$

with the size $Uk(\sigma) = |Uk(DB[y]_\varepsilon)|$ and $\sigma < U1(\sigma) < U2(\sigma) < \cdots < s$.

To speed up the computation of $d(U(\vec{x}), U(\vec{y}))$ all the possible $d(c_{j(x)}, c_{j(y)})^2$ are pre-computed and stored in a look-up table For simplicity we assume that the the cost for a look-up operation is a constant $c = 1$. This is the case, given the size of the look-up tables for each hierarchy is constant. Consequently the computational dimensions of the quantized vector $\vec{x}$ is $dim(Uk) = m/f$, where $dim(Uk)$ is the number of distinct subvectors of dimension f of the vector $\vec{x}$. It follows, that $dim(Uk)$ is the number of quantiziers. The higher the hierarchy, the lower the number of the used quantiziers. Given that $dim(U0) =: m$ (the dimension of the vector $\vec{x}$), the computational cost of a hierarchy on n level is:

$$cost_n = \sum_{i=1}^{n} Ui(\sigma) \cdot dim(U(i-1) + s \cdot dim(Un) + n \tag{16}$$

where the last summand n represents the cost of the look-up operation. The cost $cost_n$ of retrieving a dozen most similar vectors out of the database DB to a query vector $\vec{y}$, is significantly lower as the cost of simple list matching $s \cdot m$.

To estimate ε we define a mean sequence $d[Uk(DB)]_n$ which describes the characteristics of an vector database of size s:

$$d_s[Uk(DB)]_n := \sum_{i=1}^{s} \frac{d[Uk(x^{(i)})]_n}{s}. \tag{17}$$

We will demonstrate this principle on an example of high dimensional vectors representing gray images.

2.5 Example: Image retrieval

The high dimensional vectors correspond to the scaled gray images, representing the gray level distribution and the layout information. Two images $\vec{x}$ and $\vec{y}$ are similar if their distance is smaller or equal to ε, $d(\vec{x}, \vec{y}) \leq \varepsilon$. The result of a range query computed by this method is a set of images that have gray level distribution that are similar to the query image.

We preform experiments on image retrieval on one thousand ($s = 1000$) gray images of the size 128×96 resulting in vectors of dimension 12288. Each gray level is represented by 8 bits, leading to 256 different gray values. The image database consists images with photos of landscapes and people, with several outliers consisting of drawings of dinosaurs or photos of flowers (Wang et al., 2001). We use a hierarchy of four $n = 4$.

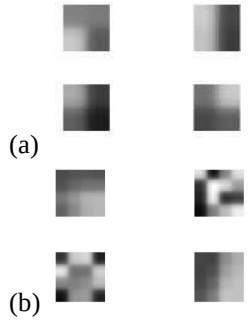

Figure 1: (a) Two examples of of squared masks M of a size 2×2. (b) Two examples of of squared masks M of a size 4×4.

- In the first level of hierarchy each image corresponds to a of dimension 12288. It is is split into 3072 distinct subvectors of dimension $4 = dim(12288/3072)$. Each subvector corresponds to a squared mask M of a size 2×2. A natural grouping of the components into subvectors is achieved by the coverage of the image with 3072 masks M (see, Figure 1 (a)). The subvectors of dimension four are grouped into clusters represented by 256 cluster centers.
- In the second level of the hierarchy the resulting quantizied images are quantizied using 768 quantiziers with $16 = dim(12288/768)$. Each subvector corresponds to a squared mask M of a size 4×4 (see, Figure 1 (b)). The subvectors of dimension sxten are grouped into clusters represented by 256 cluster centers. We follow the procedure recursively additionally two times.
- The resulting quantizied images are quantizied using 192 quantiziers with $64 = dim(12288/192)$. Each subvector corresponds to a squared mask M of a size 8×8 (see, Figure 2 (a)). The subvectors of dimension 64 are grouped into clusters represented by 256 cluster centers.
- The resulting quantizied images are quantizied us-

ing 48 quantiziers with $256 = dim(12288/48)$. Each subvector corresponds to a squared mask M of a size 16×16 (see, Figure 2 (b)). The subvectors of dimension256 are grouped into clusters represented by 256 cluster centers. (Note: The number of cluster centers remains constant through the hierarchy.)

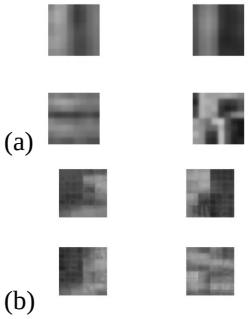

Figure 2: (a) Two examples of of squared masks M of a size 8×8. (b) Two examples of of squared masks M of a size 16×16.

The computational dimensions of the quantized vectors are $dim(U1) = 3072$, $dim(U2) = 768$, $dim(U3) = 192$ and $dim(U4) = 48$. An example of the quantizied representation of an image is indicated in the Figure 3. In each layer the image is described with less accuracy, so that the following layers represent less information. The set of ε similar vectors in correspondence to a query vector $\vec{y}$ is computed in 5 steps. For the estimation of the value of ε we use the characteristics, see Equation 17 and Figure 4

- In the first step the set of possible candidates is determined using product quantizier representation in the $U4(DB)$ of the computational dimension 48 and determine the subset $U4(DB[y])_\varepsilon$.
- Recursively out of the set $U4(DB[y])_\varepsilon$ we determine $U3(DB[y])_\varepsilon =$ $U3(U4(DB[y])_\varepsilon)_\varepsilon$,
- $U2(DB[y])_\varepsilon = U2(U3(DB[y])_\varepsilon)_\varepsilon$,
- $U1(DB[y])_\varepsilon = U2(U2(DB[y])_\varepsilon)_\varepsilon$ and
- finally the set $DB[y]_\varepsilon$ in which the false hits are separated from the selected objects through comparison in the original space by $DB[y]_\varepsilon = U0(U1(DB[y])_\varepsilon)_\varepsilon$.

To retrieve in the mean 5 most similar images the estimated ε value is 6036. We estimate the values ω_k and ε by a random sample of hundred vectors and their product quantizier representation. We computed the mean value $Uk(\overline{\sigma})$ over all possible queries (one thousand queries, each time we take an element out of the database out and and preform a query). For

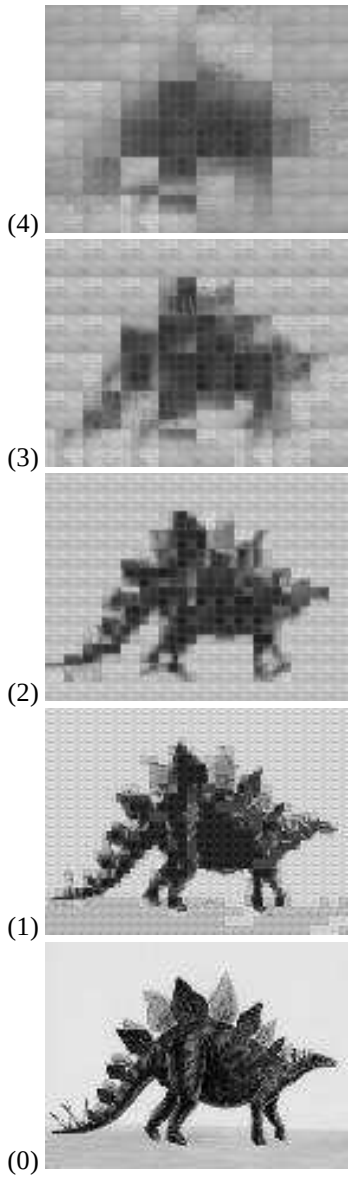

Figure 3: Gray image and its quantizied representation, (0) is equal to the original space, (1) corresponds to U1(DB), (2) to U2(DB), (3) to U3(DB) and (4) to U4(DB).

$\varepsilon = 6036$ the values are $U0(\overline{\sigma}) = 5$, $U1(\overline{\sigma}) = 20$, $U2(\overline{\sigma}) = 95$, $U_3(\overline{\sigma}) = 315$ and $U4(\overline{\sigma}) = 835$. To retrieve the 5 most similar images to a given query image of the image test database, the mean computation costs are according to Equation 16:

$$(12288 \cdot 20 + 3072 \cdot 95 + 768 \cdot 315 + 192 \cdot 835 +$$
$$+48 \cdot 1000 + 4 = 987844$$

which is 12.4 times less complex than a list matching which requires $12288 \cdot 1000$ operations. Further optimization of our results could be achieved by better quantization training (clustering algorithms).

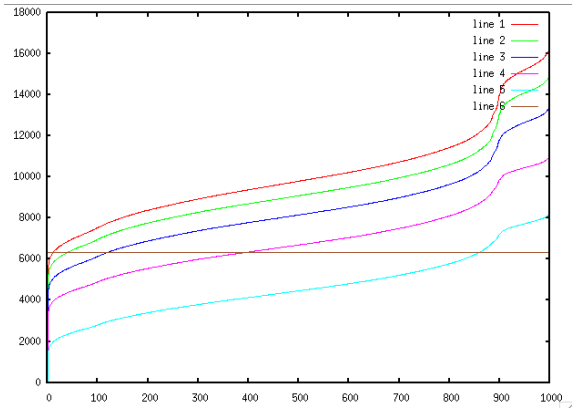

Figure 4: Characteristics of $s = 1000$, $d_s[U0(DB)]_n =$ line 1, $d_s[U1(DB)]_n =$ line 2, $d_s[U2(DB)]_n =$ line 3, $d_s[U3(DB)]_n =$ line 4 and $d_s[U4(DB)]_n =$ line 5. Line 5 represents ε.

3 CONCLUSION

We propose hierarchical product quantization for vector retrieval with no error for vector based databases. Through quantization by hierarchical clustering the distribution of the points in the high dimensional vector space can be estimated. Our method is exact and not approximative. It means we are guaranted to find the most similar vector according to a distance or similarity function. We demonstrated the working principles of our model by empirical experiment on one thousand gray images which correspond to 12288 dimensional vectors.

ACKNOWLEDGEMENTS

This work was supported by Fundação para a Ciência e Tecnologia (FCT): PTDC/EIA-CCO/119722/2010

REFERENCES

Andoni, A., Dater, M., Indyk, P., Immorlica, N., and Mirrokni, V. (2006). Locality-sensitive hashing using stable distributions. In MIT-Press, editor, *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, chapter 4. T. Darrell and P. Indyk and G. Shakhnarovich.

Ciaccia, P. and Patella, M. (2002). Searching in metric spaces with user-defined and approximate distances. *ACM Transactions on Database Systems*, 27(4).

Faloutsos, C. (1999). Modern information retrieval. In Baeza-Yates, R. and Ribeiro-Neto, B., editors, *Modern Information Retrieval*, chapter 12, pages 345–365. Addison-Wesley.

Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D., and Equitz, W. (1994). Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262.

Jegou, H., Douze, M., and Schmid, S. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Olafsson, A., Jonsson, B., and Amsaleg, L. (2008). Dynamic behavior of balanced nv-trees. In *International Workshop on Content-Based Multimedia Indexing Conference Proceedings, IEEE*, pages 174–183.

Paolo Ciaccia, Marco Patella, P. Z. (1997). M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435.

Sakurai, Y., Yoshikawa, M., Uemura, S., and Kojima, H. (2002). Spatial indexing of high-dimensional data based on relative approximation. *VLDB Journal*, 11(2):93–108.

Wang, J., Li, J., and Wiederhold, G. (2001). Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963.

Wichert, A. (2008). Content-based image retrieval by hierarchical linear subspace method. *Journal of Intelligent Information Systems*, 31(1):85–107.

Wichert, A. (2009). Image categorization and retrieval. In *Proceedings of the 11th Neural Computation and Psychology Workshop*. World Scientific.

Wichert, A., Teixeira, P., Santos, P., and Galhardas, H. (2010). Subspace tree: High dimensional multimedia indexing with logarithmic temporal complexity. *Journal of Intelligent Information Systems*, 35(3):495–516.