Searching with expectations

Harsimrat Sandhawalia, Hervé Jégou

SEARCHING WITH EXPECTATIONS

Harsimrat Sandhawalia, Hervé Jégou

INRIA

ABSTRACT

Handling large amounts of data, such as large image databases, requires the use of approximate nearest neighbor search techniques. Recently, Hamming embedding methods such as spectral hashing have addressed the problem of obtaining compact binary codes optimizing the trade-off between the memory usage and the probability of retrieving the true nearest neighbors. In this paper, we formulate the problem of generating compact signatures as a rate-distortion problem. In the spirit of source coding algorithms, we aim at minimizing the reconstruction error on the squared distances with a constraint on the memory usage. The vectors are ranked based on the distance estimates to the query vector. Experiments on image descriptors show a significant improvement over spectral hashing.

Index Terms— nearest neighbor search, quantization, source coding

1. INTRODUCTION

Nearest neighbor (NN) search is inherently expensive due to the *curse of dimensionality* [2, 1]. Formally, given a space $\mathcal{Y}$, for instance the d-dimensional Euclidean space $\mathcal{Y} = \Re^d$, and a distance $D(.,.) : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re^+$, the problem consists in finding the element NN(q) in a set $\mathcal{X} \subset \mathcal{Y}$ that minimizes the distance to the query vector $q \in \mathcal{Y}$: NN$(q) = \arg\min_{x \in \mathcal{X}} D(q, x)$. Several multi-dimensional indexing methods, such as the popular KD-tree [4] or similar branch and bound techniques, have been proposed to reduce the search time. For large dimensions, it turns out [15] that such approaches are no more efficient than the brute-force exhaustive distance calculation, whose complexity is $\mathcal{O}(nd)$.

There is a large body of literature [5, 3, 10] on algorithms that overcome this issue by performing approximate nearest neighbor (ANN) search. The key idea shared by these algorithms is to find the NN with high probability "only", instead of probability 1, which dramatically improves the search speed. Most of the effort has been devoted to the Euclidean distance, i.e., the case $\mathcal{Y} = \Re^d$ and $D(x, y) = L_2(x, y) = ||x - y||_2$. In this paper, we will only consider this distance, which is of high practical interest, for instance for vector quantization. In that case, one of the most popular ANN algorithm is the Euclidean Locality-Sensitive Hashing (LSH) [3, 12], which provides theoretical guarantees on the search quality with limited assumptions. It has been successfully used for local descriptors [7] and 3D object indexing [12, 9]. However, for real data, LSH is outperformed by heuristic methods [10], which take into account the distribution of the vectors.

ANN algorithms are compared using the trade-off between search quality and efficiency. However, this trade-off does not take into account the memory requirements of the indexing structure. In the case of E2LSH, the memory usage may be even higher than that required to store the vectors themselves, i.e., several hundreds of bytes. Only recently, researchers have tried to design methods limiting this memory usage. This is a key criterion for problems involving large amounts of data [13, 14], for instance in large-scale scene recognition problems, where millions to billions of images have to be indexed. In [14, 16], Torralba et al. propose to use a single global GIST descriptor [11] which is mapped to a short binary code to represent an image. This mapping is learned such that the neighborhood in the embedded space, with respect to the Hamming distance, reflects the similarity (in particular, Euclidean distances) between the original features. The search of the Euclidean nearest neighbors is then approximated by the search of the database binary signatures having the smaller Hamming distance to the query signature. In [16], spectral hashing (SH) is shown to outperform the binary codes generated by the restricted Boltzmann machine [14], boosting and LSH. Another related work is the Hamming embedding method of [6], where the binary signature is used to refine quantized SIFT descriptors in a bag-of-features framework.

In this paper, we formulate the problem of generating compact signatures as a rate-distortion problem on (square) distances, in the spirit of the constrained optimization algorithms used in source coding. Compression techniques aim at minimizing the vector reconstruction error, in terms of mean square distortion error, with a constraint on the memory usage. We apply this methodology to generate a compact representation minimizing the average error on the squared reconstructed distances. The vectors are then ranked based on the expectation of their distances to the query vector. The advantages of our scheme are twofold. First, our signature generation does not necessarily allocate an integer number of bits to each dimension. This is a restriction of spectral hashing [14] and of [6]. Second, the number of possible distances is significantly higher than with the aforementioned methods, for which this number is the binary code length plus one. Finally, as a byproduct of the method, we get an estimation of the expected square distance, which is, in particular, interesting for ε-radius search. One of the arguments mentioned in [14] and [6] for using the Hamming space is the efficient computation of distances. Note however that, one of the fastest way of computing Hamming distances on computers consists of using table lookups. Our method is implemented using such a strategy.

Our paper is organized as follows. In Section 2 we introduce the source coding system upon which our method is built, and introduce a few notations. Section 3 presents our approach, which is shown in Section 4 to outperform the state-of-the-art of [16] with respect to the compromise between memory and search accuracy.

2. PRELIMINARIES: SOURCE CODING

Source coding produces a compact representation $f(x)$ from a vector x, in a way that optimizes the trade-off between the size of $f(x)$ in memory, expressed in bits, and an error criterion. The reconstruction quality is typically measured by the mean square error, defined as the average (or the expectation if the density is known) square distortion between the vector and its reconstruction. The source coder we consider consists of the three following stages.

A decorrelation transform maps the input vector to a space in which a few components gather most of the energy. This can be done using the Karhunen-Loeve transform, which represents the vector in the basis provided by principal component analysis (PCA). In practice, image and video compression schemes use predefined transfor-

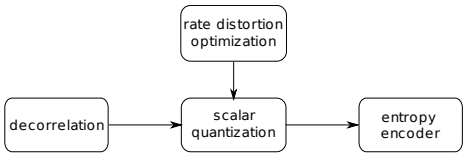

Fig. 1. A generic source code system

mations that are adapted to the data to be compressed, for instance the discrete cosine transform used in JPEG image compression.

Quantization is the only destructive process in source coding. Its purpose is to reduce the cardinality of the representation space, in particular when the input data is real-valued. A quantizer is a function mapping an input scalar or vector x to an index, in a set of n possible values. From now on, we assume that all quantization indices start at 0. Without any further processing (entropy coding), the index value is stored by $\lceil \log_2 n \rceil$ bits. In this paper, we will focus on scalar quantization, i.e., the components $x_1, \ldots, x_d$ of a vector x are quantized separately. The relative importance of the components (after decorrelation) is taken into account by using a different quantizer for each component. The vector x is then mapped as follows:

$$(x_1, \ldots, x_j, \ldots, x_d) \rightarrow (q_1(x_1), \ldots, q_j(x_j), \ldots, q_d(x_d)), \quad (1)$$

where q_j is the scalar quantizer associated with the j^{th} vector component. Let us denote by n_j the number of possible index values associated with the quantizer q_j. The cost of storing the set of index values $(q_1(x_1), \ldots, q_j(x_j), \ldots)$ without entropy coding is $\lceil \sum_j \log_2 n_j \rceil$ bits.

Each scalar quantizer q_j is fully defined by the set of reconstruction values $\{r_j(i)\}$ associated with the indices i, $0 \leq i < n_j$. The quantity $r_j(i)$ associated with the index i for the j^{th} component is often referred to as a *centroid*. Quantizing the value x_j amounts to finding the nearest neighbor in this set, as it minimizes the square reconstruction error, i.e.,

$$q_j(x_j) = \arg \min_{0 \leq i < n_j} L_2\big(x_j, r_j(i)\big). \quad (2)$$

The set of centroids corresponds to a set of intervals $\{C_i^j\}$ (including infinite endpoints) partitioning $\Re$. Let us denote by $p_j(x_j) = \mathbb{P}(X_j = x_j)$ the marginal probability density function associated with the j^{th} vector component. Assuming the intervals C_i^j fixed, a first necessary condition for the quantizer q_j to be optimal is that the reconstruction value $r_j(i)$ is equal to the expectation of the values in the corresponding interval C_i^j, i.e.,

$$r_j(i) = \frac{1}{P_j(i)} \int_{C_i^j} x\, p_j(x)\, dx \quad (3)$$

where

$$P_j(i) = \mathbb{P}(q(X_j) = i) = \int_{C_i^j} p_j(x)\, dx \quad (4)$$

is the probability to observe the index i on the j^{th} component, i.e., that the component X_j falls into the interval C_i^j. Another interesting quantity that will be used afterwards is the mean quantization error $m_j(i)$ obtained when reconstructing a value falling into C_i^j by the centroid $r_j(i)$:

$$m_j(i) = \frac{1}{P_j(i)} \int_{C_i^j} \big(x - r_j(i)\big)^2\, p_j(x)\, dx. \quad (5)$$

Assuming without loss of generality that $r_j(i) < r_j(i+1)$, a second necessary condition for optimality is that the finite boundaries of the intervals are $\big(r_j(i) + r_j(i+1)\big)/2$. The Lloyd-Max quantizer finds the optimal solution by iteratively computing the boundaries and the reconstruction values. In the following, we assume that the two necessary conditions hold, as we generate the quantizer using scalar k-means, a sampled version of Lloyd-Max.

Entropy coding minimizes the expectation of the code length by adjusting the amount of bits used to represent the different index values. This step is typically performed using Huffman or arithmetic coding. In this paper, we will not consider this step, which involve an additional computation cost.

Discussion: In source coding, a central question is how to make best use of the bits with the least possible distortion. Hence, the number of intervals n_j used for each component has to be adjusted optimally[1]. This is the purpose of rate distortion optimization (RDO), which minimizes the distortion associated with a given number of bits. The optimization usually consists in adjusting the number of quantized indexes of the different quantizers.

3. SEARCHING WITH EXPECTATIONS

NN search depends solely on the distances between the query vector and the database vectors, or equivalently the square distances. Our method compares the vectors based on their quantization indices. For each vector $x = (x_1, \ldots, x_j, \ldots)$ expressed in the PCA basis, we only need to store the quantized indexes $q(x) = \big(q_1(x_1), \ldots, q_j(x_j), \ldots\big)$. In this section, we first explain how the expected square distances are estimated from these indices, assuming fixed scalar quantizers. Then we introduce the optimization problem we solve to optimize the quantizers, i.e., to allocate bits to dimensions. Finally, we explain how the indices provided by the quantizers are combined to bijectively generate a compact signature.

As our method is derived from the source coding framework presented in Section 2, we assume the vector to be decorrelated using PCA. The orthonormal basis ensures that the square distance between vectors is identical before and after decorrelation. We also assume that each component is quantized using a Lloyd-Max quantizer q_j (or its k-means sampled version), hence Eqn 3 holds. The construction of quantizer q_j is solely parametrized by n_j.

3.1. Expected square distance

Let us consider two vectors X and Y, seen here as random variables whose components are expressed in the orthonormal basis provided by the PCA. In this subsection, we assume that the scalar quantizers are known. In others terms n_j, the number of intervals, are fixed (not necessary equal from one component to another). Thanks to the linearity of the expectation, we have

$$\mathbb{E}\big[L_2(X,Y)^2\big] = \sum_{1 \leq j \leq d} \mathbb{E}\left[(X_j - Y_j)^2\right]. \quad (6)$$

Recall that, for a particular vector component j, we have stored the quantized indices $q_j(x_j)$ and $q_j(y_j)$. These indices identify the cells C_i^j and $C_{i'}^j$ into which x_j and y_j are quantized. The conditional expected square distance $e_j(i,i')$ between X_j and Y_j, knowing that $i = q_j(X_j)$ and $i' = q_j(Y_j)$, is computed as

$$e_j(i,i') = \int_{C_i^j} \int_{C_{i'}^j} (x-y)^2\, p(x|i)\, p(y|i')\, dx\, dy. \quad (7)$$

where $(x-y)^2$ is subsequently re-written as

$$(x-y)^2 = \big((x - r_j(i)) - (y - r_j(i'))^2 + (r_j(i) - r_j(i'))\big)^2.$$

[1] In the JPEG norm, the optimization of the scalar quantizers associated with the components of the DCT-transformed signal also takes into account the psycho-visual impact of the frequency bands.

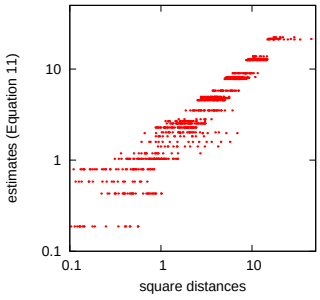

Fig. 2. Comparison of square distances with their estimates, for 2D Gaussian vectors. We have used 5 bits to represent the vectors (n_1=8 and n_2=4). Notice the discrete number of possible values for the estimates. Therefore, there is a lower and an upper bound on the estimated square distances produced by the algorithm.

Developing this squared expression and observing, from Eqn 3, that

$$\int_{C_i^j} \big(x - r_j(i)\big)\, p(x|i)\, dx = 0 \text{ and } \int_{C_{i'}^j} \big(y - r_j(i')\big)\, p(y|i')\, dy = 0, \tag{8}$$

Eqn 7 simplifies to

$$e_j(i,i') = (r_j(i) - r_j(i'))^2 + m_j(i) + m_j(i'), \tag{9}$$

where we recognize the mean square errors $m_j(i)$ and $m_j(i')$ associated with the reconstruction of X_j and Y_j by $r_j(i)$ and $r_j(i')$, respectively. Using Eqn 6 and Eqn 9, comparing two vectors x and y with the knowledge of the scalar quantized indices $q(x)$ and $q(y)$ simply consists in computing the following sum:

$$\mathbb{E}\big[L_2(X,Y)^2 \,|\, q(X), q(Y)\big] = \sum_j e_j\big(q_j(x_j), q_j(y_j)\big). \tag{10}$$

This summation is only performed on the components which have been selected by the rate-distortion optimization introduced in the next subsection, i.e., the most energetic components. The number of interesting components heavily depends on the number of bits b to allocate. Note that the values $r_j(i)$ and m_i^j are byproducts of the k-means quantizer. They are stored into lookup tables that are accessed to compute the expected square distances. To improve the efficiency at the cost of larger lookup tables, one can also directly store the tables $e_j(i,i')$ for all $1 \leq j \leq d$ and $0 \leq i, i' < n_j$, and compound several components in Eqn 10.

Figure 2 illustrates the quality of the estimates provided by this equation on the toy example of 2D anisotropic gaussian vectors.

3.2. Rate distortion optimization

We now focus on the optimization of the different scalar quantizers. The constraint on the total number of bits being fixed, the objective is to minimize the error of the estimator in Eqn 10. As we have fixed the quantizer generation procedure (Lloyd-Max/k-means), we only need to allocate the numbers of intervals $(n_1, \ldots, n_j, \ldots, n_d)$ to the different components. The optimization aims at minimizing, on average, a given distortion criterion, for instance the absolute value of the square distance error:

$$\mathbb{E}\left[\Big|(X-Y)^2 - \mathbb{E}\big[L_2(X,Y)^2 \,|\, q(X), q(Y)\big]\Big|\right] \tag{11}$$

subject to a constraint on the maximum number of bits b used to represent the vector. As we do not use entropy coding, this memory usage constraint is given by

$$\sum_{1 \leq j \leq d} \log_2(n_j) \leq b. \tag{12}$$

We simplify the problem of Eqn 11 by minimizing instead the upper bound

$$\mathbb{E}\left[\sum_{1 \leq j \leq d} \Big|(X_j - Y_j)^2 - e_j\big(q_j(X_j), q_j(Y_j)\big)\Big|\right], \tag{13}$$

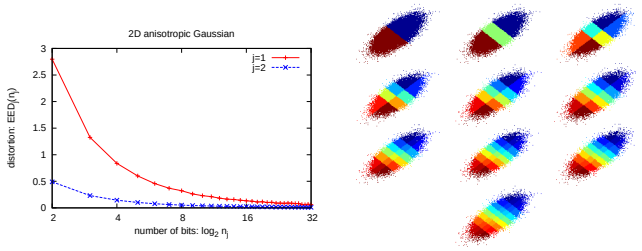

Fig. 3. Illustration of the rate-distortion greedy optimization approach. *Left:* rate distortion curves associated with the 2 different components. Each curve corresponds to the set of points $(\log_2(n_j), \text{EED}_j(n_j))$, $j = 1, 2$. *Right:* illustration of the different scalar quantizers obtained at each step of the algorithm: the number of intervals is increased for one of the two dimensions.

obtained by applying the triangular inequality to Eqn 11. Doing so, the impact of n_j on the distortion does not depend on the other components. This is convenient when applying a discrete optimization algorithm. The rate-distortion approach is then performed as follows.

1– Expected distortion. For each component j and for all[2] possible values of n_j, we compute the expected distortion $\text{EED}_j(n_j)$ resulting from the use of n_j intervals for the quantizer q_j associated with the component j. This is done by estimating, using Monte-Carlo sampling, the expected error on the square distance:

$$\text{EED}_j(n_j) = \int_{\Re}\int_{\Re} \Big|(x-y)^2 - e_j(q_j(x), q_j(y))\Big|\, p(x)\, p(y)\, dx\, dy. \tag{14}$$

2– Optimization. The quantities $\text{EED}_j(n_j)$ being estimated for all j and n_j (and stored into tables), minimizing Eqn 13 on average amounts to solving

$$(\hat{n}_1, \ldots, \hat{n}_j, \ldots, \hat{n}_d) = \arg \min_{(n_1,\ldots,n_j,\ldots,j_d)} \sum_{1 \leq j \leq d} \text{EED}_j(n_j) \tag{15}$$

subject to the constraint of Eqn 12. A reasonable (though not necessarily optimal) solution of this discrete optimization problem is obtained by a greedy-best algorithm. Figure 3 illustrates the behavior of such an algorithm on a set of 2D Gaussian vectors, where the values n_1 and n_2 are increased until the termination constraint of Eqn 12 is not satisfied anymore.

Remark: The algorithm may output $n_j = 1$ for the least energetic components. In that case $q_j(x_j)$ is deterministically equal to 0 and is not stored. As such a component has no impact on the nearest neighbor search, it is ignored in Eqn 10. However, the quantity m_j, which is equal to the component variance $\mathbb{E}[(X_j - E[X_j])^2]$ in that case, should still be added if the final objective is to have a square distance estimate.

3.3. Code construction

In this subsection, we detail how the binary code is obtained from a given input vector x. The rate-distortion optimization performed in the previous subsection has fixed the quantizers, providing the numbers of intervals n_j used for each of the vector component. A vector x is then represented by the vector of quantized indices $q(x) = (q_1(x_1), \ldots, q_j(x_j), \ldots, q_d(x_d))$. As we have $0 \leq i < n_j$

[2] By "all", we mean the integers from 1 to 2^b, as higher values do not satisfy the constraint of Eqn 12. Note that in practice, most of the components are not likely to receive more than few bits.

for component j, the total number Λ of possible vectors is given by $\Lambda = \prod_{1 \leq j \leq d} n_j$. A vector is therefore represented by a code of length

$$\lceil \log_2 \Lambda \rceil = \left\lceil \sum_{1 \leq j \leq d} \log_2 n_j \right\rceil \text{ bits.} \quad (16)$$

Note that a scalar quantizer such that $n_j = 1$ does not impact the memory cost, as the index is always zero (and not stored). In order to bijectively map $q(x)$ to an integer $\lambda(x)$ such that $0 \leq \lambda(x) < \Lambda$, we use the factorization

$$\lambda(x) = q_1(x_1) + n_1 \left(q_2(x_2) + ...(q_{d-1}(x_{d-1}) + n_{d-1}\, q_d(x_d))...\right). \quad (17)$$

The reconstruction of the indices from the code $\lambda(x)$ is performed iteratively: starting with j=1 and l=$\lambda(x)$, each iteration computes

$$q_j(x_j) = l \mod n_j, \quad (18)$$

$$l := l \,/\, n_j \quad \text{(integer division).} \quad (19)$$

The efficiency of this process is improved by compounding several components into a single one. The components such that n_j=1 are discarded. In the extreme case where Λ is small (for instance if $\Lambda \leq 1024$, which corresponds to $b \leq 10$), all the components are compounded. For this purpose, we replace the set of tables $e_j(i, i')$ by a single lookup, which is directly indexed by the tuple $(\lambda(x), \lambda(y))$ associated with the vectors to compare.

4. EXPERIMENTS

Evaluation protocol. Our method is compared with the state-of-the-art spectral hashing of Weiss et al. [16], which maps the Euclidean vectors to binary signatures. The search consists in comparing the Hamming distances of the database signatures to the query vector signature. We have used the publicly available code.

The evaluation is performed on two types of popular image descriptors, namely the SIFT local descriptor [8] and the color GIST global descriptor [11]. Both datasets[3] were constructed using publicy available data and software: we extracted the GIST descriptors from a subsample of the tiny image dataset [13] and used SIFT descriptors available from [6]. The learning stage is performed on a separate set of vectors (out-of-sample evaluation) for all the methods.

dataset:	SIFT (d=128)	GIST (d=960)
learning set size	100,000	500,000
database set size	1,000,000	1,000,000
queries set size	10,000	500

The search quality is measured by the recall@k, defined as the proportion of query vectors whose nearest neighbor is ranked in the first k positions. This measures indicates the fraction of queries for which the nearest neighbor is retrieved, if the short-list formed by the first k vectors is verified using Euclidean distances.

Results. Figure 4 shows the retrieval performance of the different methods as a function of the number of bits b used to represent a vector. Our approach significantly outperforms spectral hashing. For instance, for SIFT descriptors and $b = 128$, the nearest neighbor is ranked in the first 100 positions for 94% of the queries, against less than 70% for spectral hashing.

Figure 5 shows the rank repartition of the nearest neighbor for b=64. Our expectation-based approach clearly ranks nearest neighbors better (note the log-scale for the ranks). For a fixed signature size, the search quality is better for the SIFT than for the GIST dataset, due to the larger dimension of the GIST descriptor. The improvement of our method is comparatively larger for the GIST dataset.

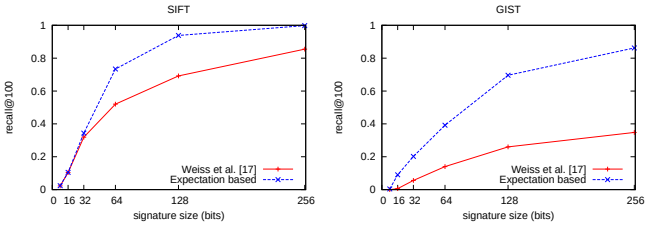

Fig. 4. Search quality as a function of the memory usage: proportion of the queries whose nearest neighbor is returned in the top 100 positions for datasets (SIFT and GIST) of one million images.

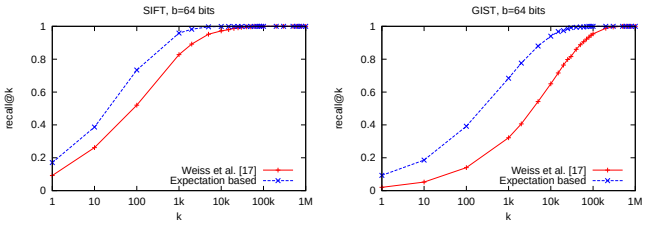

Fig. 5. recall@k for varying values of k. Note that this curve represents the repartition of the ranks of the nearest neighbors.

5. REFERENCES

[1] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the International Conference on Database Theory*, pages 217–235, August 1999.

[2] C. Böhm, S. Berchtold, and D. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, October 2001.

[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Symposium on Computational Geometry*, pages 253–262, 2004.

[4] J. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transaction on Mathematical Software*, 3(3):209–226, 1977.

[5] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimension via hashing. In *Proceedings of the International Conference on Very Large DataBases*, pages 518–529, 1999.

[6] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the European Conference on Computer VIsion*, October 2008.

[7] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *ACM Multimedia*, pages 869–876, 2004.

[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[9] B. Matei, Y. Shan, H. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3D-signature space estimation. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1111 – 1126, July 2006.

[10] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP*, 2009.

[11] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[12] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, chapter 3. MIT Press, March 2006.

[13] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large database for non-parametric object and scene recognition. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970, November 2008.

[14] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[15] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the International Conference on Very Large DataBases*, pages 194–205, 1998.

[16] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[3] Online at http://www.irisa.fr/texmex/people/jegou/ann.php