How to evaluate search methods for vector quantization

Citation for the original published paper (version of record):

Agrell, E., Hedelin, P. (1994)
How to evaluate search methods for vector quantization
Proc. Nordic Signal Processing Symposium, Ålesund, Norway: 258-263

How to Evaluate Search Methods for Vector Quantization

Erik Agrell *Per Hedelin*

Department of Information Theory
Chalmers University of Technology
S-412 61Göteborg, Sweden

agrell@it.chalmers.se per@it.chalmers.se

ABSTRACT

Fast search methods for vector quantization are a necessity for benefiting from the performance gains of large-sized codebooks in real-time applications. We take on two 4096 vector quantizer codebooks as illustrative examples for our study. The performance of a set of search procedures is compared. Several aspects of complexity are discussed. We compare average computational complexity and maximum computational complexity in the light of real-time usage. We address storage requirement and the computational complexity required to set up the search procedures. Moreover we illustrate distortion as a function of computational effort. One main conclusion is that an accurate comparison will not yield a single answer. Depending on what aspects that are highlighted in a test, either of the three search procedures in study can be elected as the winner.

1. INTRODUCTION

At a constant transmission rate, distortion of vector quantization decreases monotonically with increasing dimension of the vectors to quantize.

There are several factors that have prohibited employment of large codebooks, including storage and training problems, but the issue of searching a large set of codevectors is often the limiting factor for applications. Hence, a major goal of source coding research is to establish methods of handling large-dimensional codebooks. In this contribution we focus on aspects of searching large sized codebooks, i.e. on the search complexity.

1.1 The Search Problem

The search problem in vector quantization occurs each time an input vector is to be encoded. The codebook has to be examined for the best representation of the input, according to some criterion. The obvious way is an exhaustive search, that is, calculating the distortion for each of the possible output codewords. In the last two decades, however, several approaches have been suggested to increase the encoding speed. This is possible by employing alternative tests, computationally cheaper than full distortion calculations, to exclude codewords, one or several at a time, from further consideration. For most methods, this type of strategy requires the precomputation of some codebook properties.

One of the earliest ideas was to precompute distances between all codewords and a set of fixed (anchor) points. The triangle inequality, or similar tests, is utilized to select a subset of the codewords, for which the exact distortion has to be calculated [1]. Recursive application of some subset-selection method gives rise to tree-organized algorithms [2-4]. A different approach is to exploit the geometrical properties of the Voronoi diagram, as in the neighbor descent methods [5-7].

In general, two prices are paid for obtaining a fast search: *i*) additional memory is required, and *ii*) an additional preparatory analysis of the codebook is required. Since this latter step is made once, prior to the employment of the encoder, a large computational burden can often, but not always, be tolerated for this analysis. Also, a further speed gain can be obtained by accepting *iii*) an increased output distortion.

1.2 Paper Outline

Because of the aforementioned multi-dimensional trade-off between different qualities, between aspects as computational burden and memory requirements, it is not trivial to summarize the performance of a search method in a single table or diagram. Highlighted in this study is exactly this evaluation problem. As an example, we encode Gaussian data with a pdf-optimized vector quantizer, using three different search algorithms. We perform a few different tests, some of which measure related properties. Nevertheless, the various tests indicate quite different performance. Each particular test would have given a fairly clear (but possibly misleading) image of how good the methods are, if published alone.

A comparison of three different search algorithms is the essence of our contribution. We discuss primarily the computational complexity. We also comment on the additional storage requirement of the algorithms, and, since one of the methods does not perform optimal codebook look-up, we compare the performance.

Finally, we relax the requirement of finding the very best codeword by introducing a constraint on the encoding time.

1.3 Notation

A d-dimensional vector $\mathbf{X}$ drawn from a random source (with a known probability density function) is to be encoded with a codebook $\{\mathbf{c}^{(i)}\}$ of $N = 2^k$ entries. The object is to minimize the conventional Euclidean distortion measure

$$D = E\left[\left\|\mathbf{X} - \mathbf{c}^{(i)}\right\|^2\right]$$

One of the procedures discussed below utilizes an eigenvalue analysis. We denote the eigenvalues of the source correlation matrix by $\{\lambda_m\}$. A transformed source vector $\mathbf{Y}'$ is $\mathbf{Y}' = \mathbf{AY}$ where $\mathbf{A}$ is the matrix of eigenvectors. Thus, the components of $\mathbf{Y}'$ are uncorrelated.

2. SEARCH METHODS

We compare three different search methods, two neighbor descent methods, using different adjacency tables, and one tree search method.

2.1 Neighbor Descent Methods

The neighbor descent (ND) methods utilize an *adjacency table*, which is computed during the preencoding analysis. It gives a list of adjacent codewords for each codeword in the codebook. The encoding of an input vector can then be done by iterative improvement of an initial guess. An iteration consists of computing the distortions for the adjacent codewords of the current hypothesis. If any of these codewords turns out to be better than the hypothesis, further examination of adjacent codewords is aborted. The current hypothesis is abandoned and the found codeword immediately becomes the new hypothesis. The overall procedure continues until none of the neighbors provide lower distortion.

Two neighbor descent algorithms were compared. Their difference lies in the definition of adjacent codewords. The first algorithm (referred to as RND in [7]) uses the Voronoi diagram—two codewords are regarded as adjacent if their Voronoi regions have a facet in common. A linear programming approach is described in [8] for efficiently establishing the adjacency table for each vector of an arbitrary codebook.

The second algorithm employs Gabriel neighbors, which is a more restricted condition [9, 10] than the Voronoi neighbor concept. Two codewords are Gabriel neighbors if they are Voronoi neighbors and if their common facet is intersected by the straight line between the two codevectors.

The Gabriel approach reduces the memory requirement and increases encoding speed, as demonstrated in this paper. However, the price to pay is distortion. In contrast to Voronoi neighbor descent, which can be proved to find the optimal codeword for every input [8], the Gabriel adjacency is not sufficient to guarantee optimality.

Before comparing the complexity of building the adjacency tables using the Voronoi and the Gabriel approaches, it is important to stress that the creation of the tables is done as a preparatory step, i.e. before the employment of the encoding algorithm. Still we find it worth mentioning that finding the set of Gabriel neighbors is far simpler than finding the corresponding Voronoi neighbors. A flavor of the computational complexity is obtained by comparing the CPU-time for the particular 6-dimensional codebook we take on as example in this study. For the Voronoi neighbors approximately 25 hours were required whereas the Gabriel neighbors were established in one hour.

2.2 Tree Search Procedures

There exists a variety of tree search (TS) procedures for vector quantization. Several important TS procedures apply only to certain (constrained) sets of codevectors. The particular version employed in our study is general in the sense that it is applicable for an arbitrary set of (unconstrained) codevectors. The overall architecture is that of a balanced d-level tree with 2^{k_m} branches from each node at level $m-1$ [4]. Each level, m, corresponds to one component of the transformed d-dimensional codevectors $\mathbf{d}$, where $\mathbf{d} = \mathbf{Ac}$.

The total number of nodes, K, depends on the branching, i.e. on $\{k_m\}$, but for any allocation, $K < 2N$. Associated with each node at level m are the leftmost $r_m = r_{m-1} + k_m$ bits of a codeword index i. Hence, each leaf of the tree corresponds to a codeword index i. Associated with each node (m, r) at any level $m < d$ is also a triplet

$$\left\{\bar{d}_m^{(r)}, d_m^{-(r)}, d_m^{+(r)}\right\}$$

consisting of the mean, the minimum and the maximum respectively of the mth component of the transformed codevectors $\mathbf{d}$, i.e. $d_m^{(i)}$, for those i only that are descendent leafs of the given node.

Preparatory to codebook employment the branching parameters, $\{k_m\}$, must be determined. For this we have employed a conventional bit-allocation algorithm utilizing the eigenvalues $\{\lambda_m\}$.

Next an index assignment for the codebook vectors is obtained by sorting the transformed codebook vectors $\{\mathbf{d}^{(i)}\}$ according to the bit-allocation. The triplets $\{\bar{d}_m^{(r)}, d_m^{-(r)}, d_m^{+(r)}\}$ are obtained as a natural part of this sorting procedure.

At run-time, a transformed source vector $\mathbf{Y}$ is processed by first finding an initial guess i^* for the index. This is accomplished by descending the tree with hard pruning based on utilizing the mean positions $\bar{d}_m^{(r)}$ for scoring. The true distortion D^* is evaluated for the winner of this step. The tree is thereafter descended level by level until the leaf nodes are encountered while utilizing $\{d_m^{-(r)}, d_m^{+(r)}\}$ for keeping track of a lower bound $D_{\min}(m,s)$ of the distortion associated with a node.

$$D_{\min}(m,s) = D_{\min}(m-1,s') + \beta(m,s,y_m)$$

$$\beta(m,s,y_m) = \begin{cases} (y_m - d_m^{-(s)})^2 & y_m < d_m^{-(s)} \\ 0 & d_m^{-(s)} < y_m < d_m^{+(s)} \\ (y_m - d_m^{+(s)})^2 & y_m > d_m^{+(s)} \end{cases}$$

where $(m-1,s')$ is the parent node of node (m,s). Propagation is inhibited for any node (r,m) that reaches a distortion $D_{\min}(m,s)$ that exceeds D^*. Finally, the true distortion of the surviving nodes are evaluated. The selected codeword i is the one yielding least true distortion within the set of survivors.

3. TWO GAUSSIAN CODEBOOKS

We have studied several different codebooks in various dimensions and for various rates. Below we report on the performance for a six-dimensional Gaussian source, i.e. $d = 6$, coded at a rate $R = 2$. Thus the codebook size is $N = 4096$. Two sample-to-sample correlations, ρ, were selected, namely $\rho = 0$ and $\rho = 0.75$. The codebooks were trained employing an LBG-type of approach using random samples from the respective sources. The iterations encompassed five million samples. Performance as discussed below was measured on additional random sets of one million vectors (independent of the training sets).

The high-rate approximation for Gaussian vector quantization (cf. [11]) states that

$$\sigma_Q^2 \geq \sigma_{\mathbf{X}}^2 \cdot \Theta(d) \cdot f(d) \cdot 2^{-2R}$$

where $R = k/d$ is the rate and

$$\Theta(d) = \left(\prod_{i=1}^{d} \lambda_i\right)^{1/d} \cdot \left(\sum_{i=1}^{d} \lambda_i / d\right)^{-1}$$

$$f(d) = \left(\frac{d\Gamma(d/2)}{2}\right)^{2/d} \frac{2}{d} \left(\frac{d+2}{d}\right)^{d/2}$$

For dimension $d = 6$ this formula predicts a signal-to-noise ratio (SNR) of $12.04 - 1.57 = 10.47$ dB for $\rho = 0$ and $12.04 - 1.57 + 2.99 = 13.46$ dB for $\rho = 0.75$, both at rate $R = 2$. We measured an SNR of 10.41 dB for $\rho = 0$ and 13.47 dB for $\rho = 0.75$ for our trained codebooks. The discrepancy to the prediction given by the high-rate approximation is in reasonable agreement with the accuracy of this approximation for a moderate rate R. For our example we thus have codebooks that are close to optimal for the given sources.

3.1 Voronoi Properties of the Codebooks

The complexity of neighbor descent, regarding encoding time as well as memory, is directly dependent on the size of the adjacency table. This is the twofold motivation for using Gabriel neighbors instead of the complete Voronoi description. Table 1 shows some statistics of the adjacency tables for both codebooks.

The number of Gabriel neighbors is approximately equal to half the number of Voronoi neighbors for the uncorrelated codebook, and considerably less than that for $\rho = 0.75$. Corresponding improvements in search time and memory requirement will be noted in the next section.

These results are typical for trained codebooks,

Table 1. Average and maximum number of neighbors, for two neighbor types and two codebooks.

	Average number of neighbors		*Maximum number of neighbors*	
Codebook	*Voronoi*	*Gabriel*	*Voronoi*	*Gabriel*
$\rho = 0$	117	62	178	99
$\rho = 0.75$	118	50	305	87

Table 2. Search complexity and storage requirement for the Gaussian codebook with $\rho = 0$. The last column shows the loss in SNR compared to a full search, which gives an SNR of 10.41 dB.

	Number of distance computations			*SNR*
Method	*Average*	*Maximum*	*Storage*	*difference (dB)*
Voronoi ND	170	326	732 kbyte	0
Gabriel ND	108	233	394 kbyte	–0.06
Tree search	78	693	12 kbyte	0

i.e. codebooks that are close to optimal for a given source. To our experience there are some notable differences when searching, for instance, random codebooks in comparison to (close to) optimal codebooks. The underlying property is that Voronoi regions are more regular for trained codebooks than for randomized codebooks.

4. EVALUATION OF SEARCH METHODS

4.1 Searching for Optimum

As a measure of how fast a search algorithm is, the number of vectorial distance computations is often used. In our 12-bit examples a full search obviously requires the computation of 4096 distances, but for most fast search methods, the number of distances depends on the particular input vector **X**. The maximum number of distance computations, over all **X**, is a relevant measure in real-time applications where a fixed amount of time must be assigned for the encoding of an input vector. In applications where a large delay is allowed, the average number is more appropriate.

The three search methods were used in conjunction with the two vector quantizers for Gaussian data described above. Table 2 shows both maximum and average number of distance computations for the codebook with a sample-to-sample correlation of $\rho = 0$, together with the memory requirement for storage of the precomputed codebook structure. According to these tests, the TS method is faster than ND, and it also requires much less memory.

The signal-to-noise ratio of the quantizer is also shown in the table. This is because Gabriel ND theoretically does not guarantee optimal encoding. (The other two algorithms do.) As it turns out, however, the SNR decrease is minor, namely .06 dB. Gabriel ND practically always finds the truly optimal codeword, or one with almost as low distortion.

The same set of results for $\rho = 0.75$ are shown in table 3. The distortion is still very close to that obtained with optimal encoding, but the difference is larger than for the uncorrelated codebook. The explanation to this is that the Gabriel neighbors are fewer for the correlated codebook (see table 1), which also is the cause of the lower search complexity in this case.

Summarizing the results of tables 2 and 3, we see that the tree search method outperforms either of the two neighbor methods as regards average computational complexity as well as in storage requirements. The two neighbor methods, on the other hand, have a considerably lower maximum computational complexity.

4.2 Constraining the Encoding Time

However, tables 2 and 3 do not reveal all aspects of importance for applications. It is also relevant to address how rapidly the distortion decreases during the encoding process, that is, we need to find out how significant it is to actually continue until the algorithms terminate, guaranteeing optimal codebook look-up. If the last distance computations have a small probability of improving the output codeword, we can buy considerable time for just a small distortion increase by imposing a bound on the number of distance computations that are allowed. In real-time applications where the available time for encoding is limited, such a bound is a necessity.

Table 3. Search complexity, storage requirement, and SNR loss (compared to a full search yielding 13.47 dB) when $\rho = 0.75$.

	Number of distance computations			*SNR*
Method	*Average*	*Maximum*	*Storage*	*difference (dB)*
Voronoi ND	165	425	737 kbyte	0
Gabriel ND	90	198	323 kbyte	–0.10
Tree search	64	457	24 kbyte	0

SNR as a function of such a bound is depicted in figure 1. The diagram shows that both ND methods gives a higher SNR than TS if more than 26 distortion computations are allowed, or conversely, that ND reaches any SNR value greater than 6.4 dB faster. The TS complexity displays a step after a time corresponding to 5 distance computations. This is the time needed for the initial pass through the tree, with hard pruning, before which not even an initial guess is known. If only very little time is available, this is an efficient method, skipping the second pass.

Figure 2 shows similar results for the correlated codebook. The encoding algorithms reach a higher SNR value, but the same qualitative differences between the algorithms as in figure 1 can be observed.

The curves illustrate the three phases of a search procedure: finding an initial guess, improving the guess, and verifying that the last guess cannot be further improved. The algorithms perform differently in each of these phases. The first pass in tree search provides an excellent initial guess, but the improvement thereafter is slow. Of the three algorithms, Gabriel ND is the fastest in the second phase, but Voronoi ND performs a stronger test in the third phase, guaranteeing that the optimal codeword was indeed found, which is important at least from a theoretical point of view.

In passing we compose a new search algorithm

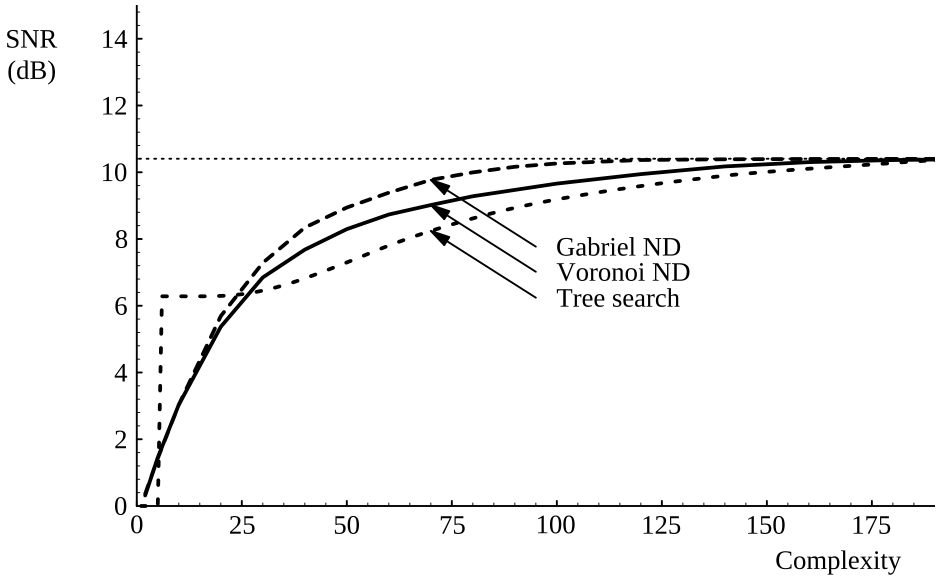

Figure 1. SNR as a function of complexity given by the number of distance measurements, for the vector quantizer with $\rho = 0$.

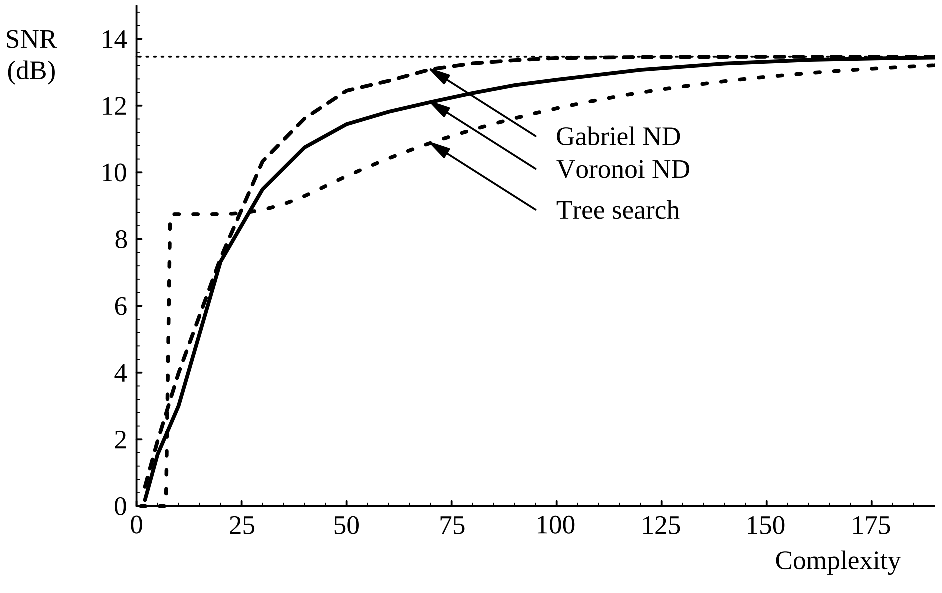

Figure 2. SNR as a function of complexity for $\rho = 0.75$.

of the three studied algorithms, employing each one in the phase for which it is most favorable. Thus we use TS to find an initial guess, then iterate Gabriel ND until it terminates, and finally use Voronoi ND to check if the solution is optimal and improve it if possible. This hybrid search algorithm outperforms its three components in all aspects but storage. The memory requirement is equal to that of Voronoi ND, plus one third of the TS memory (only the first element of the triplets is needed). The Gabriel adjacency table can be included as a part of the Voronoi table and does not require any extra memory.

5. SUMMARY AND CONCLUSIONS

Neighbor descent methods are attractive for vector quantization since such methods isolate a good candidate at an early stage. The examples taken on illustrate this property well. The examples also show that neighbor descent algorithms are less effective in their final phase when verifying that they actually have retrieved the correct entry.

Employing only Gabriel neighbors performs surprisingly well. The loss in performance by utilizing only a subset of the true Voronoi neighbors is so marginal that it falls below the accuracy of SNR measurements over one million sample vectors.

The tree search algorithm is effective in storage. Its average computational complexity in terms of distance computations is the lowest of the methods, for optimal codebook look-up.

Whenever optimal search is mandatory the maximum number of computations is of importance. The descent methods have a maximum that exceeds the maximum number of neighbors accounted for in the method.

“Fast” is an ambiguous term. What appears to be a competitive search method in one test, may come out as a slow method in another. It depends on what properties you measure. Therefore, it is vital to employ a test procedure that is in correspondence with the intended application and implementation. That a method has been found to be fast in one type of source coding system does not automatically imply that is suitable to use in another. Especially, the relative performance may change dramatically when a time constraint is imposed. We emphasize that an evaluation of a search algorithm should incorporate several different tests in order to accurately describe the “fastness” of the algorithm.

REFERENCES

[1] W. A. Burkhard and R. M. Keller, “Some approaches to best-match file searching,” *Communications of the ACM*, vol. 16, no. 4, pp. 230–236, Apr. 1973.

[2] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, Sept. 1977.

[3] V. Ramasubramanian and K. K. Paliwal, “Fast *K*-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding,” *IEEE Transactions on Signal Processing*, vol. 40, no. 3, pp. 518–531, Mar. 1992.

[4] P. Hedelin, “Single stage spectral quantization at 20 bits,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 525–528, Adelaide, Australia, Apr. 1994.

[5] P. J. Green and R. Sibson, “Computing Dirichlet tessellations in the plane,” *The Computer Journal*, vol. 21, no. 2, pp. 168–173, May 1978.

[6] R. L. Joshi and P. G. Poonacha, “A new MMSE encoding algorithm for vector quantization,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 645–648, Toronto, Ontario, Canada, May 1991.

[7] E. Agrell, “Spectral coding by fast vector quantization,” in *Proc. IEEE Workshop on Speech Coding for Telecommunications*, pp. 61–62, Sainte-Adèle, Québec, Canada, Oct. 1993.

[8] E. Agrell, “A method for examining vector quantizer structures,” in *Proc. IEEE International Symposium on Information Theory*, p. 394, San Antonio, TX, Jan. 1993.

[9] A. Okabe, B. Boots, and K. Sugihara, *Spatial tessellations: Concepts and applications of Voronoi diagrams*. Chichester, England, U.K.: John Wiley & Sons, 1992.

[10] S. Arya and D. M. Mount, “Algorithms for fast vector quantization,” in *Proc. Data Compression Conference*, J. A. Storer, M. Cohn, eds., pp. 381–390, Snowbird, UT, Mar.–Apr. 1993.

[11] T. D. Lookabaugh and R. M. Gray, “High-resolution quantization theory and the vector quantizer advantage,” *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 1020–1033, 1989 1989.